CƠ SỞ DỮ LIỆU

◊ ◊

PHẦN

I

♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦ ♦

- **Các khái niệm cơ bản**
- **Kiến trúc hệ thống cơ sở dữ liệu**
- **Mô hình quan hệ thực thể**
- **Mô hình quan hệ**
- **Chuẩn hoá quan hệ**
- **Thiết kế cơ sở dữ liệu vật lý**

GV. Phạm Thị Hoàng Nhung
Bộ môn Công nghệ phần mềm
Đại học Thủy lợi

MỤC LỤC

1 Chương 1. CÁC KHÁI NIỆM CƠ BẢN

Trong nhiều năm gần đây, thuật ngữ Cơ sở dữ liệu - Database đã trở nên quen thuộc trong nhiều lĩnh vực. Các ứng dụng tin học vào quản lý ngày càng nhiều và đa dạng, hầu hết các lĩnh vực kinh tế, xã hội… đều đã ứng dụng các thành tựu mới của tin học vào phục vụ công tác chuyên môn của mình. Chính vì lẽ đó mà ngày càng nhiều người quan tâm đến thiết kế, xây dựng và ứng dụng cơ sở dữ liệu (CSDL).

Trong chương này, chúng ta sẽ tìm hiểu thế nào là cơ sở dữ liệu và các khái niệm liên quan đến nó. Trước hết, chúng ta sẽ tìm hiểu lý do tại sao cần phải quản lý dữ liệu bằng CSDL?

1.1 Tại sao phải có một cơ sở dữ liệu

Hệ thống các tệp tin cổ điển

Cho đến nay vẫn còn một số đơn vị kinh tế, hành chính sự nghiệp… sử dụng mô hình hệ thống các tệp tin cổ điển: chúng 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:***

Việc xây dựng hệ thống các tệp tin riêng tại từng đơn vị quản lý ít tốn thời gian bởi khối lượng thông tin cần quản lý và khai thác là nhỏ, không đòi hỏi đầu tư vật chất và chất xám nhiều, do đó triển khai ứng dụng nhanh.

Thông tin được khai thác chỉ phục vụ mục đích hẹp nên khả năng đáp ứng nhanh chóng, kịp thời.

- ***Nhược điểm:***

Thông tin được tổ chức riêng rẽ ở nhiều nơi nên việc cập nhật dễ làm mất tính nhất quán dữ liệu.

Hệ thống thông tin được tổ chức thành các hệ thống file riêng lẻ nên thiếu sự chia sẻ thông tin giữa các nơi.

Qua phân tích trên, chúng ta nhận thấy việc tổ chức dữ liệu theo hệ thống tệp tin hoàn toàn không phù hợp với những hệ thống thông tin lớn. Việc xây dựng một hệ thống thông tin đảm bảo được tính nhất quán dữ liệu, đáp ứng được nhu cầu khai thác đồng thời của nhiều người là thực sự cần thiết.

1.2 Định nghĩa một cơ sở dữ liệu

1.2.1 Khái niệm

CSDL và công nghệ CSDL đã có những tác động to lớn trong việc phát triển sử dụng máy tính. Có thể nói rằng CSDL ảnh hưởng đến tất cả các nơi có sử dụng máy tính:

Kinh doanh (thông tin về sản phẩm, khách hàng, …)

Giáo dục (thông tin về sinh viên, điểm, ..)

Thư viện (thông tin về tài liệu, tác giả, độc giả…)

Y tế (thông tin về bệnh nhân, thuốc….)…

Như vậy, cơ sở dữ liệu là gì?

CSDL là tập hợp các dữ liệu có cấu trúc và liên quan với nhau được lưu trữ trên máy tính, được nhiều người sử dụng và được tổ chức theo một mô hình.

Ví dụ:

Danh bạ điện thoại là một ví dụ về CSDL.

- Là các thông tin có ý nghĩa
- Là tập hợp các thông tin có cấu trúc.
- Các thông tin này có liên quan với nhau và có thể hệ thống được.

Trong khái niệm này, chúng ta cần nhấn mạnh, CSDL là tập hợp các thông tin có tính chất hệ thống, không phải là các thông tin rời rạc, không có liên quan với nhau. Các thông tin này phải có cấu trúc và tập hợp các thông tin này phải có khả năng đáp ứng nhu cầu khai thác của nhiều người sử dụng một cách đồng thời. Đó cũng chính là đặc trưng của CSDL.

1.2.2 Ưu điểm

Từ khái niệm trên, ta thấy rõ ưu điểm nổi bật của CSDL là:

Giảm sự trùng lặp thông tin xuống mức thấp nhất và do đó đảm bảo được tính nhất quán và toàn vẹn dữ liệu (Cấu trúc của cơ sở dữ liệu được định nghĩa một lần. Phần định nghĩa cấu trúc này gọi là meta-data, và được Catalog của HQTCSDL lưu trữ).

Đảm bảo sự độc lập giữa dữ liệu và chương trình ứng dụng (Insulation between programs and data): Cho phép thay đổi cấu trúc, dữ liệu trong cơ sở dữ liệu mà không cần thay đổi chương trình ứng dụng.

Trừu tượng hoá dữ liệu (Data Abstraction): Mô hình dữ liệu được sử dụng để làm ẩn lưu trữ vật lý chi tiết của dữ liệu, chỉ biểu diễn cho người sử dụng mức khái niệm của cơ sở dữ liệu.

Nhiều khung nhìn (multi-view) cho các đối người dùng khác nhau: Đảm bảo dữ liệu có thể được truy xuất theo nhiều cách khác nhau. Vì yêu cầu của mỗi đối tượng sử dụng CSDL là khác nhau nên tạo ra nhiều khung nhìn vào dữ liệu là cần thiết.

Đa người dùng (multi-user): Khả năng chia sẻ thông tin cho nhiều người sử dụng và nhiều ứng dụng khác nhau.

1.2.3 Vấn đề cần giải quyết

Để đạt được các ưu điểm trên, CSDL đặt ra những vấn đề cần giải quyết. Đó là:

Tính chủ quyền của dữ liệu: Do tính chia sẻ của CSDL nên chủ quyền của CSDL dễ bị xâm phạm.

Tính bảo mật và quyền khai thác thông tin của người sử dụng: Do có nhiều người được phép khai thác CSDL nên cần thiết phải có một cơ chế bảo mật và phân quyền hạn khai thác CSDL.

Tranh chấp dữ liệu: Nhiều người được phép cùng truy cập vào CSDL với những mục đích khác nhau: Xem, thêm, xóa hoặc sửa dữ liệu. Cần phải có cơ chế ưu

tiên truy cập dữ liệu hoặc giải quyết tình trạng xung đột trong quá trình khai thác cạnh tranh. Cơ chế ưu tiên có thể được thực hiện bằng việc cấp quyền (hay mức độ) ưu tiên cho từng người khai thác.

Đảm bảo dữ liệu khi có sự cố: Việc quản lý dữ liệu tập trung có thể làm tăng nguy cơ mất mát hoặc sai lệnh thông tin khi có sự cố mất điện đột xuất hoặc đĩa lưu trữ bị hỏng. Một số hệ điều hành mạng có cung cấp dịch vụ sao lưu ảnh đĩa cứng (cơ chế sử dụng đĩa cứng dự phòng - RAID), 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 an toàn cho CSDL, nhất thiết phải có một cơ chế khôi phục dữ liệu khi có sự cố xảy ra.

1.3 Hệ quản trị cơ sở dữ liệu (DataBase Management System_DBMS)

1.3.1 Ví dụ

Như chúng ta đã biết, kích thước và độ phức tạp của CSDL rất khác nhau.

Ví dụ:

Danh bạ điện thoại của một quốc gia, một thành phố.. chứa tới hàng triệu số và những thông tin cần thiết về khách hàng.

Trong trường đại học có tới hàng ngàn sinh viên. Nhà trường phải quản lý tất cả những thông tin liên quan đến sinh viên như: tên, ngày sinh, quê quán, địa chỉ, kết quả học tập…

Xét một Ví dụ về CSDL quản lý tài liệu và độc giả trong thư viện quốc gia. Giả sử rằng có 100 triệu cuốn sách, mỗi cuốn sách cần lưu 10 thông tin liên quan, mỗi thông tin chứa tối đa 400 kí tự thì CSDL sẽ phải có tối thiểu 100 $*10^{6*}$ 400 *10 kí tự (bytes). Như vậy, dung lượng bộ nhớ cần dùng là: 100 $*10^{6*}$ 400 *10= 400 GB.

Ta thấy, bộ nhớ cũng là vấn đề cần phải được giải quyết. Tuy nhiên, vấn đề quan trọng hơn ở đây lại là cách thức tổ chức dữ liệu trong một cơ sở dữ liệu để phục vụ cho việc truy cập, tìm kiếm, cập nhật,….nhanh chóng và an toàn hơn.

Việc tổ chức dữ liệu như thế nào được thực hiện thông qua Hệ quản trị cơ sở dữ liệu(HQTCSDL).

Vậy hệ quản trị cơ sở dữ liệu (HQTCSDL) là gì?

1.3.2 Khái niệm.

HQTCSDL là tập hợp các phần mềm cho phép định nghĩa các cấu trúc để lưu trữ thông tin trên máy, nhập dữ liệu, thao tác trên các dữ liệu đảm bảo sự an toàn và bí mật của dữ liệu.

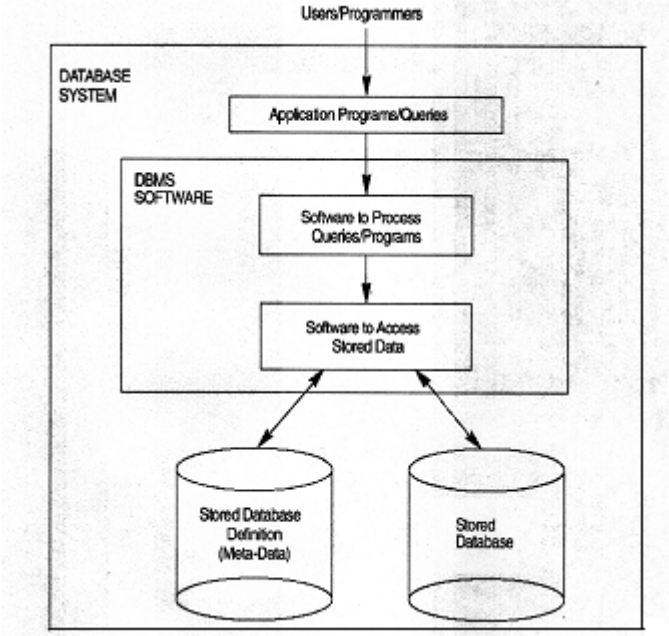
Định nghĩa cấu trúc: Định nghĩa cấu trúc CSDL bao gồm việc xác định kiểu dữ liệu, cấu trúc và những ràng buộc cho dữ liệu được lưu trữ trong CSDL.

Nhập dữ liệu: Là việc lưu trữ dữ liệu vào các thiết bị lưu trữ trung gian được điều khiển bằng HQTCSDL.

Thao tác dữ liệu: thao tác trên CSDL bao gồm những chức năng như truy xuất cơ sở dữ liệu để tìm kiếm thông tin cần thiết, cập nhật cơ sở dữ liệu và tổng hợp những báo cáo từ dữ liệu.

1.4 Hệ thống cơ sở dữ liệu (Database System)

Là phần mềm HQTCSDL cùng với dữ liệu của bản thân cơ sở dữ liệu đó.



Hình 1.1. Môi trường hệ thống cơ sở dữ liệu đơn giản

1.5 Các đối tượng sử dụng CSDL

Đối với các cơ sở dữ liệu nhỏ, mang tính cá nhân như lịch làm việc, danh bạ điện thoại cá nhân… thì chỉ cần một người để tạo ra và thao tác trên nó. Tuy nhiên, đối với các CSDL lớn như: quản lý tài chính của ngân hàng nhà nước, điều hành các chuyến bay cho các sân bay quốc tế… cần phải có rất nhiều người tham gia thiết kế, xây dựng, bảo trì CSDL và hàng trăm người sử dụng. Trong phần này, chúng ta tìm hiểu xem ai là người thao tác với CSDL hàng ngày. Và trong phần sau, chúng ta xem xét những người không trực tiếp tham gia một CSDL cụ thể, họ là người duy trì môi trường hệ thống CSDL.

1.5.1 Đối tượng trực tiếp

1.5.1.1 Quản trị cơ sở dữ liệu

Trong những tổ chức có nhiều người cùng sử dụng chung một nguồn dữ liệu thì nhất thiết phải có một người đứng đầu quản lý, chịu trách nhiệm đối với nguồn dữ liệu này. Đó chính là người quản trị cơ sở dữ liệu (Database Administrators _ DBA).

DBA có nhiệm vụ tổ chức nội dung của cơ sở dữ liệu, tạo và phân quyền cho người sử dụng, đưa ra yêu cầu về phần cứng và phần mềm… nếu cần thiết. DAB chịu trách nhiệm bảo vệ an toàn, Backup thông tin…khi có sự cố.

1.5.1.2 Thiết kế cơ sở dữ liệu

Người thiết kế CSDL chịu trách nhiệm:

- Xác định những dữ liệu nào cần lưu trữ trong CSDL
- Lựa chọn những cấu trúc thích hợp để biểu diễn và lưu trữ những dữ liệu này.

- Phỏng vấn tất cả những người sử dụng CSDL sau này để hiểu được những yêu cầu của họ đối với CSDL
- Tiến hành phân tích thiết kế hệ thống sau khi thống nhất được tất cả các yêu cầu của người sử dụng

1.5.1.3 Người sử dụng cuối

Người sử dụng cuối là những người truy cập CSDL để:

- Truy vấn
- Cập nhật
- Thống kê, báo cáo

1.5.1.4 Phân tích hệ thống và Lập trình ứng dụng

Phân tích hệ thống để định rõ những yêu cầu của người sử dụng cuối cùng, thống nhất để đưa ra khung nhìn cho từng đối tượng người sử dụng, quản lý các giao tác (transactions)…

Lập trình ứng dụng:

- Thực hiện các yêu cầu thông qua lập trình bằng những ngôn ngữ phù hợp
- Chạy thử chương trình (test)
- Chữa lỗi và gỡ rối chương trình (debug)
- Viết tài liệu, hướng dẫn sử dụng.
- Bảo trì hệ thống

1.5.2 Đối tượng gián tiếp

Ngoài những đối tượng trực tiếp tham gia vào một CSDL cụ thể như đã nói ở trên, còn có một đội ngũ những người phân tích, phát triển, và thực hiện tạo ra môi trường hệ thống và phần mềm của hệ quản trị cơ sở dữ liệu. Những người này không trực tiếp thao tác trên một hệ quản trị CSDL nào cụ thể. Họ là:

- Người phân tích và thực hiện tạo ra hệ thống của HQTCSDL
- Những nhà phát triển hệ công cụ (Tool developers)
- Người kiểm thử và bảo trì hệ thống

1.6 Lợi ích của việc sử dụng HQTCSDL

- Hạn chế dư thừa dữ liệu.
- Ngăn cản truy cập dữ liệu bất hợp pháp (bảo mật và phân quyền sử dụng).
- Cung cấp khả năng lưu trữ lâu dài cho các đối tượng và cấu trúc dữ liệu.
- Cho phép suy dẫn dữ liệu (từ dữ liệu này suy ra dữ liệu khác) sử dụng Rules.
- Cung cấp giao diện đa người dùng.
- Cho phép biểu diễn mối quan hệ phức tạp giữa các dữ liệu.

- Đảm bảo ràng buộc toàn vẹn dữ liệu (Enforcing Integrity Constraints).
- Cung cấp thủ tục sao lưu và phục hồi (backup và recovery)

2 Chương 2. NHỮNG KHÁI NIỆM VÀ KIẾN TRÚC CỦA HỆ THỐNG CƠ SỞ DỮ LIỆU

2.1 Mô hình dữ liệu, lược đồ và trường hợp (Data Models, Schemas, Instances)

Một trong những đặc điểm cơ bản của cơ sở dữ liệu là cung cấp một số mức độ trừu tượng hoá dữ liệu bằng cách làm ẩn đi cách thức tổ chức dữ liệu- cái mà hầu hết người dùng không cần biết đến.

Mô hình dữ liệu (Data Model): Là một tập những khái niệm dùng để biểu diễn cấu trúc của cơ sở dữ liệu-cung cấp những điều kiện cần thiết để đạt được mức độ trừu tượng dữ liệu. Cấu trúc dữ liệu bao gồm kiểu dữ liệu (data types) và mối quan hệ giữa các dữ liệu (relationships) và những ràng buộc (constraints) mà cơ sở dữ liệu phải tuân theo.

Hầu hết mô hình dữ liệu đều có một tập hợp các thao tác cơ bản (basic operations) để truy vấn và cập nhật dữ liệu.

- Thao tác chung (generic operations): Thêm (insert), Xoá (delete), Sửa (modify), Truy cập (retrieve)
- Thao tác do người dùng định nghĩa (user-defined operations)

2.1.1 Phân loại mô hình dữ liệu

Có rất nhiều mô hình dữ liệu đã được đưa ra, chúng ta có thể phân loại chúng theo những kiểu khái niệm mà họ đã dùng để biểu diễn cấu trúc cơ sở dữ liệu. Mô hình dữ liệu được chia làm 3 loại sau:

a. ***Mô hình khái niệm (Conceptual (high-level, semantic) data models)***:

Cung cấp những khái niệm gần gũi với đa số người sử dụng, mô hình này chỉ ra cái gì được đưa vào để quản lý. Mô hình này là phương tiện để những người phân tích thiết kế giao tiếp với người sử dụng, nhằm thu thập thông tin, xác định đúng đắn và đầy đủ yêu cầu của hệ thống.

Mô hình này sử dụng cấu trúc dữ liệu là: thực thể (entity), thuộc tính (attribute) và mối liên kết (relationship)

b. ***Mô hình dữ liệu vật lý (Physical (low-level, internal) data models):***

Cung cấp những khái niệm để biểu diễn chi tiết cách thức dữ liệu được lưu trữ trong máy tính. Mô hình này chỉ ra định dạng bản ghi (record formats), thứ tự sắp xếp các bản ghi (record ordering) và đường dẫn để truy cập dữ liệu (access paths).

c. Mô hình dữ liệu thể hiện (Implementation (record-oriented) data models):

Mô tả các dữ liệu bằng cách sử dụng những ký pháp tương ứng với mô hình dữ liệu mà một hệ quản trị cơ sở dữ liệu sử dụng.

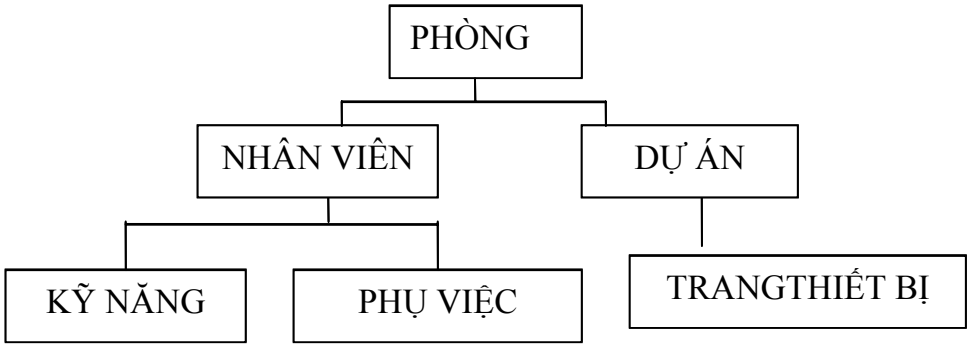
Các loại mô hình cơ sở dữ liệu thể hiện:

c1. Mô hình phân cấp

Mô hình CSDL phân cấp được biểu diễn dưới dạng cây và các đỉnh của cây là các bản ghi. Các bản ghi liên kết với nhau theo mối quan hệ cha-con.

- Một cha có nhiều con
- Một con chỉ có một cha

Ví dụ:



Hình 2.1. Minh họa mô hình cơ sở dữ liệu phân cấp

Ưu điểm:

- Thể hiện dễ dàng quan hệ 1-N.
- Việc phân chia dữ liệu dễ thể hiện, đảm bảo an toàn dữ liệu
- Tính độc lập của chương trình và các dữ liệu được đảm bảo

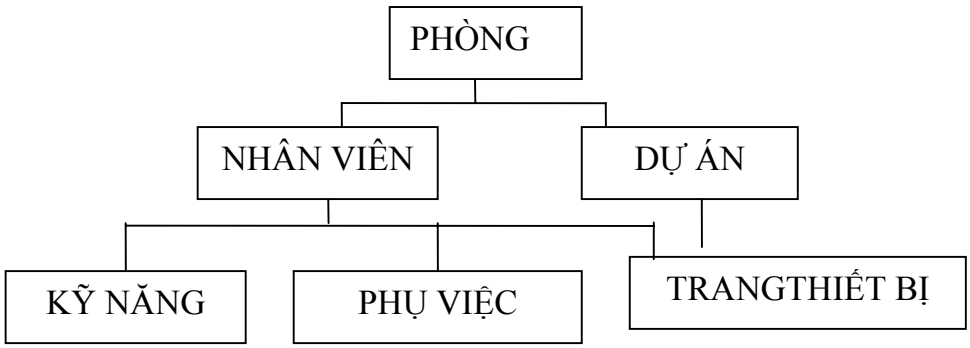
Nhược điểm:

- Không thể hiện được mối quan hệ M-N
- Trong một hệ thống phân cấp, dữ liệu được tổ chức như trên dễ đến khó sửa đổi dữ liệu.

c2. Mô hình mạng

Cấu trúc cơ bản trong mô hình mạng là những tập hợp và mỗi tập hợp có bản ghi là bản ghi chủ và một số bản ghi thành viên. Mỗi thành viên có thể thuộc về nhiều tập hợp.

Ví dụ:



Hình 2.2. Minh họa mô hình cơ sở dữ liệu mạng

Ưu điểm:

- Dễ thể hiện mối liên kết M-N
- Kiểu truy cập dữ liệu mềm dẻo hơn kiểu phân cấp

Nhược điểm:

- Việc sửa đổi số liệu khó khăn.
- Với những lập trình viên, việc thiết kế CSDL khó.

c3. Mô hình quan hệ

Trong mô hình quan hệ, các dữ liệu được biểu diễn ở dạng các bảng với các dòng và các cột.

Trong mô hình quan hệ không có một cấu trúc vật lý nào của dữ liệu mô tả sự kết nối giữa các bảng. Thay vào đó, sự kết nối giữa các bảng được mô tả logic bằng các giá trị được lưu trữ trong các dòng của bảng. Chẳng hạn trong hình dưới đây, thuộc tính *ProCode(Mã tỉnh)* được lưu trong cả 2 bảng PROVINCE và bảng STUDENT, giá trị chung này cho phép người dùng liên kết được 2 bảng.

PROVINCE	
ProCode	**ProName**
04	Hà Nội
08	Tp Hồ Chí Minh
…	…

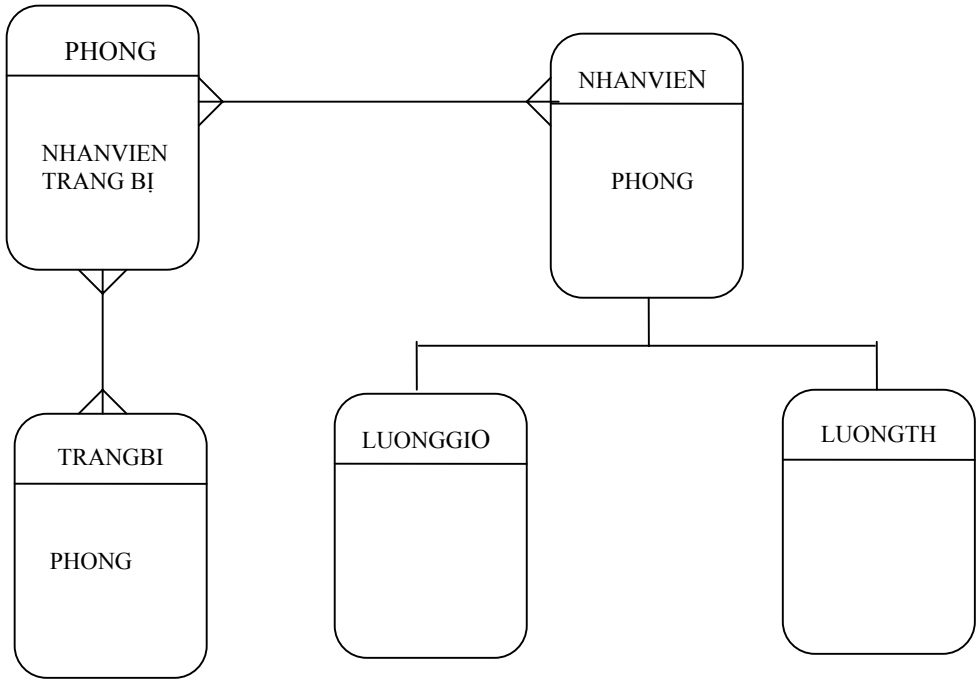
STUDENT			
StdNo	**StdName**	**StdBird**	**ProCode**
TD001	AA	9/16/1979	04
TD002	BB	6/19/1979	08
…	…	…	….

Hình 2.3. Minh họa mô hình cơ sở dữ liệu quan hệ

c4. Mô hình hướng đối tượng

Trong mô hình hướng đối tượng, các thuộc tính dữ liệu và các thao tác trên các dữ liệu này được bao gói trong một cấu trúc gọi là đối tượng.

Đối tượng có thể chứa các dữ liệu phức hợp như văn bản, hình ảnh, tiếng nói và hình ảnh động. Một đối tượng có thể yêu cầu hoặc xử lý dữ liệu từ một đối tượng khác bằng việc gửi đi một thông báo đến đối tượng đó. Mô hình hướng đối tượng biểu diễn một sơ đồ mới để lưu trữ và thao tác dữ liệu. Từ một đối tượng có thể sinh ra một đối tượng khác.



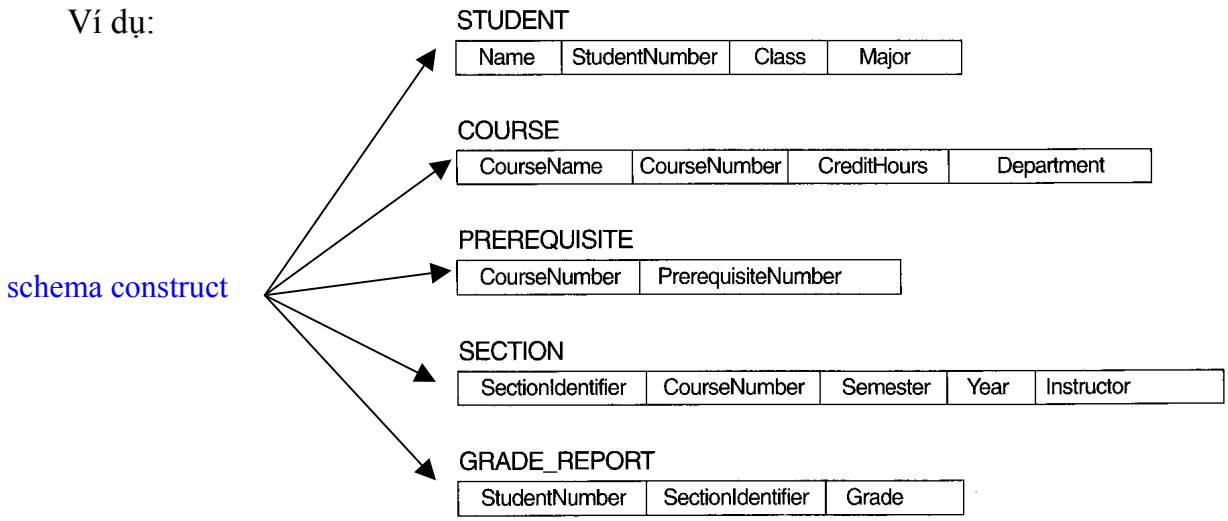
Hình 2.4. Minh họa mô hình cơ sở dữ liệu hướng đối tượng

2.1.2 Lược đồ(Schema) , minh hoạ (instances), và trạng thái (State)

Lược đồ cơ sở dữ liệu (Database Schema): là biểu diễn của cơ sở dữ liệu, bao gồm cấu trúc cơ sở dữ liệu và những ràng buộc trên dữ liệu.

Sơ đồ của lược đồ cơ sở dữ liệu (Schema Diagram): Là lược đồ cơ sở dữ liệu được biểu diễn thông qua sơ đồ.

Ví dụ:



Hình 2.5. Lược đồ cơ sở dữ liệu UNIVERSITY

Minh học cơ sở dữ liệu (Database Instance): Là dữ liệu thực sự được lưu trữ trong cơ sở dữ liệu ở thời điểm hiện tại. Database Instance cũng được gọi là trạng thái của cơ sở dữ liệu (**database state)**

Ví dụ:

STUDENT	Name	StudentNumber	Class	Major
	Smith	17	1	CS
	Brown	8	2	CS

COURSE	CourseName	CourseNumber	CreditHours	Department
	Intro to Computer Science	CS1310	4	CS
	Data Structures	CS3320	4	CS
	Discrete Mathematics	MATH2410	3	MATH
	Database	CS3380	3	CS

SECTION	SectionIdentifier	CourseNumber	Semester	Year	Instructor
	85	MATH2410	Fall	98	King
	92	CS1310	Fall	98	Anderson
	102	CS3320	Spring	99	Knuth
	112	MATH2410	Fall	99	Chang
	119	CS1310	Fall	99	Anderson
	135	CS3380	Fall	99	Stone

GRADE_REPORT	StudentNumber	SectionIdentifier	Grade
	17	112	B
	17	119	C
	8	85	A
	8	92	A
	8	102	B
	8	135	A

PREREQUISITE	CourseNumber	PrerequisiteNumber
	CS3380	CS3320
	CS3380	MATH2410
	CS3320	CS1310

Hình 2.5. Cơ sở dữ liệu UNIVERSITY

Như vậy, **Database Schema** rất hiếm khi thay đổi, còn **Database State** thay đổi bất kỳ khi nào có sự cập nhập dữ liệu.

2.2 Lược đồ kiến trúc của hệ quản trị cơ sở dữ liệu (DBMS Architecture) và sự độc lập dữ liệu (Data Independence)

Như chúng ta đã biết, các tính chất quan trọng nhất của cơ sở dữ liệu là: (1) Đảm bảo sự độc lập giữa chương trình ứng dụng và dữ liệu. (2) Hỗ trợ nhiều khung nhìn cho các đối tượng người dùng khác nhau. (3) Sử dụng danh mục để lưu trữ biểu diễn dữ liệu (schema). Trong phần này, chúng ta sẽ tìm hiểu kiến trúc của hệ quản trị

cơ sở dữ liệu, gọi là Lược đồ kiến trúc 3 mức mức(three –schema architecture). Sau đó chúng ta sẽ tìm hiểu về khái niệm độc lập dữ liệu.

2.2.1 Lược đồ kiến trúc 3 mức của HQTCSDL

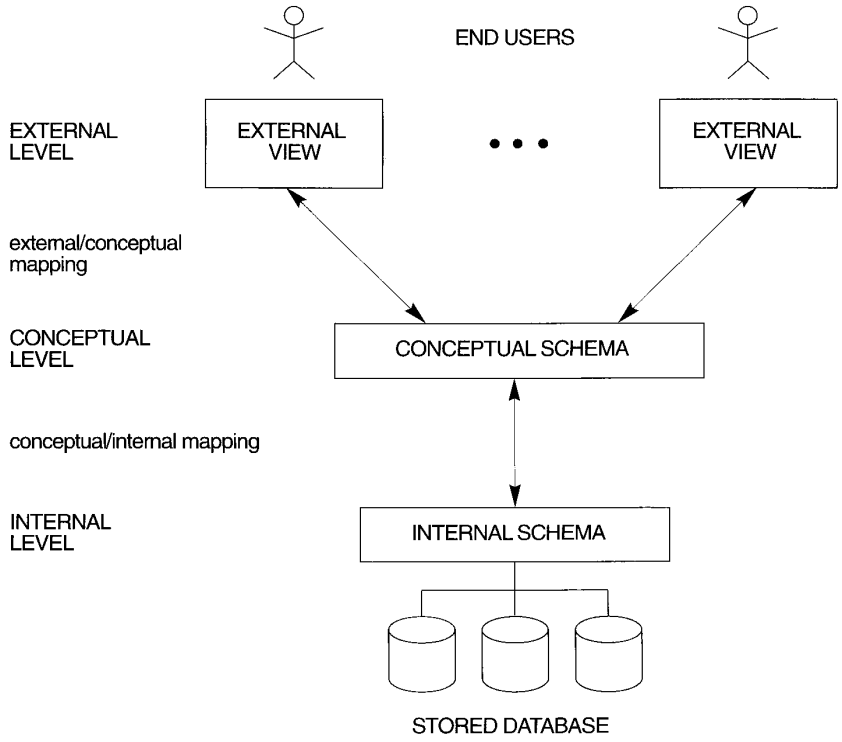
Mục đích của việc phân thành 3 mức trong kiến trúc của mô hình HQTCSDL là để tách biệt các ứng dụng của người sử dụng với cơ sở dữ liệu vật lý. Trong kiến trúc này, lược đồ có thể được định nghĩa ở 3 mức sau:

Lược đồ mức trong (Internal schema) ở Mức trong (Internal level) để biểu diễn chi tiết cấu trúc lưu trữ dữ liệu và cách thức truy cập dữ liệu. Lược đồ mức trong sử dụng mô hình dữ liệu vật lý (physical data model).

Lược đồ khái niệm (Conceptual schema) ở Mức khái niệm (Conceptual level) để biểu diễn cấu trúc và các ràng buộc trong toàn bộ cơ sở dữ liệu phục vụ cho việc giao tiếp với người sử dụng. Lược đồ khái niệm ẩn đi cách thức tổ chức vật lý của dữ liệu, chỉ tập trung vào việc biểu diễn các thực thể, các kiểu dữ liệu, mối quan hệ giữa các thực thể, các thao tác của người sử dụng và các ràng buộc giữa các dữ liệu. Mô hình dữ liệu mức khái niệm (Conceptual data model) hoặc Mô hình dữ liệu thể hiện (Implementation data model) có thể được sử dụng ở mức này.

Lược đồ mức ngoài (External Level) ở Mức ngoài (External level hoặc View level) để biểu diễn hàng loạt những khung nhìn của người sử dụng (user views). Mô hình dữ liệu mức cao (High-level data model) hoặc Mô hình dữ liệu thể hiện (Implementation data model) có thể được sử dụng ở mức này.

Ánh xạ giữa các mức này là cần thiết. Những chương trình làm việc với dữ liệu ở mức ngoài và được HQTCSDL ánh xạ tới dữ liệu vật lý ở mức trong để thực hiện.



Hình 2.6. Lược đồ kiến trúc 3 mức của HQTCSDL

2.2.2 Độc lập dữ liệu

Kiến trúc 3 mức của HQTCSDL có thể được sử dụng để giải thích khái niệm về độc lập dữ liệu. Độc lập dữ liệu là khả năng thay đổi lược đồ ở một mức nào đó của hệ thống cơ sở dữ liệu mà không cần phải thay đổi lược đồ ở mức cao hơn. Chúng ta có thể định nghĩa 2 kiểu của độc lập dữ liệu:

Độc lập dữ liệu logic (Logical data independence): cho phép thay đổi lược đồ khái niệm mà không cần phải thay đổi lược đồ mức ngoài hoặc những chương trình ứng dụng. Chúng ta có thể thay đổi lược đồ khái niệm để mở rộng (thêm các trường dữ liệu, các bản ghi) hoặc thu nhỏ cơ sở dữ liệu (xóa các trường dữ liệu, các bản ghi).

Độc lập dữ liệu vật lý (Physical data independence): cho phép thay đổi lược đồ mức trong mà không cần thay đổi lược đồ khái niệm. Có khi chúng ta cần thay đổi lược đồ mức trong vì các file vật lý đôi khi cần tổ chức lại để tăng hiệu quả thực hiện. Nếu kiểu dữ liệu không thay đổi thì chúng ta không cần thay đổi lại lược đồ khái niệm.

2.3 Ngôn ngữ của HQTCSDL

Vì HQTCSDL phục vụ có nhiều đối tượng người sử dụng khác nhau nên nó phải hỗ trợ ngôn ngữ để người sử dụng tương tác với nó. Trong phần này chúng ta sẽ tìm hiểu một số những ngôn ngữ được HQTCSDL hỗ trợ.

Ngôn ngữ định nghĩa dữ liệu (Data Definition Language - DDL): Là ngôn ngữ được các nhà quản trị cơ sở dữ liệu (DBA) và các nhà thiết kế cơ sở dữ liệu (database designers) dùng để xây dựng lược đồ khái niệm của cơ sở dữ liệu. Trong nhiều HQTCSDL, DDL cũng được sử dụng để định nghĩa lược đồ mức trong và mức ngoài (các khung nhìn). Một số HQTCSDL chia thành 2 ngôn ngữ: Ngôn ngữ định nghĩa lưu trữ (storage definition language – SDL) và Ngôn ngữ định nghĩa khung nhìn (view definition language -VDL) được dùng để định nghĩa lược đồ mức trong và mức ngoài.

Ngôn ngữ thực hiện dữ liệu (Data Manipulation Language -DML): Là ngôn ngữ được sử dụng để thao tác với dữ liệu bao gồm việc truy cập đến bản ghi và cập nhật dữ liệu cho bản ghi (thêm, sửa, xoá). Các lệnh DML có thể được nhúng trong ngôn ngữ lập trình hoặc thực hiện độc lập (ngôn ngữ truy vấn).

2.4 Các tính năng của HQTCSDL

- Nạp dữ liệu đang lưu trữ ở các tệp tin vào cơ sở dữ liệu (Conversion Tool).
- Cung cấp các thao tác truy xuất
- Đảm bảo tính độc lập dữ liệu
- Cung cấp thủ tục sao lưu và phục hồi (backup và recovery)
- Cung cấp các thủ tục điều khiển đồng thời (Do tính truy xuất đồng thời và cạnh tranh)
- Cung cấp các thủ tục kiểm soát bản quyền, kiểm tra tính đúng đắn của dữ liệu (để đảm bảo tính an toàn và toàn vẹn dữ liệu)

2.5 Phân loại HQTCSDL

N**gười ta phân loại HQTCSDL dựa trên một số tiêu chí:**

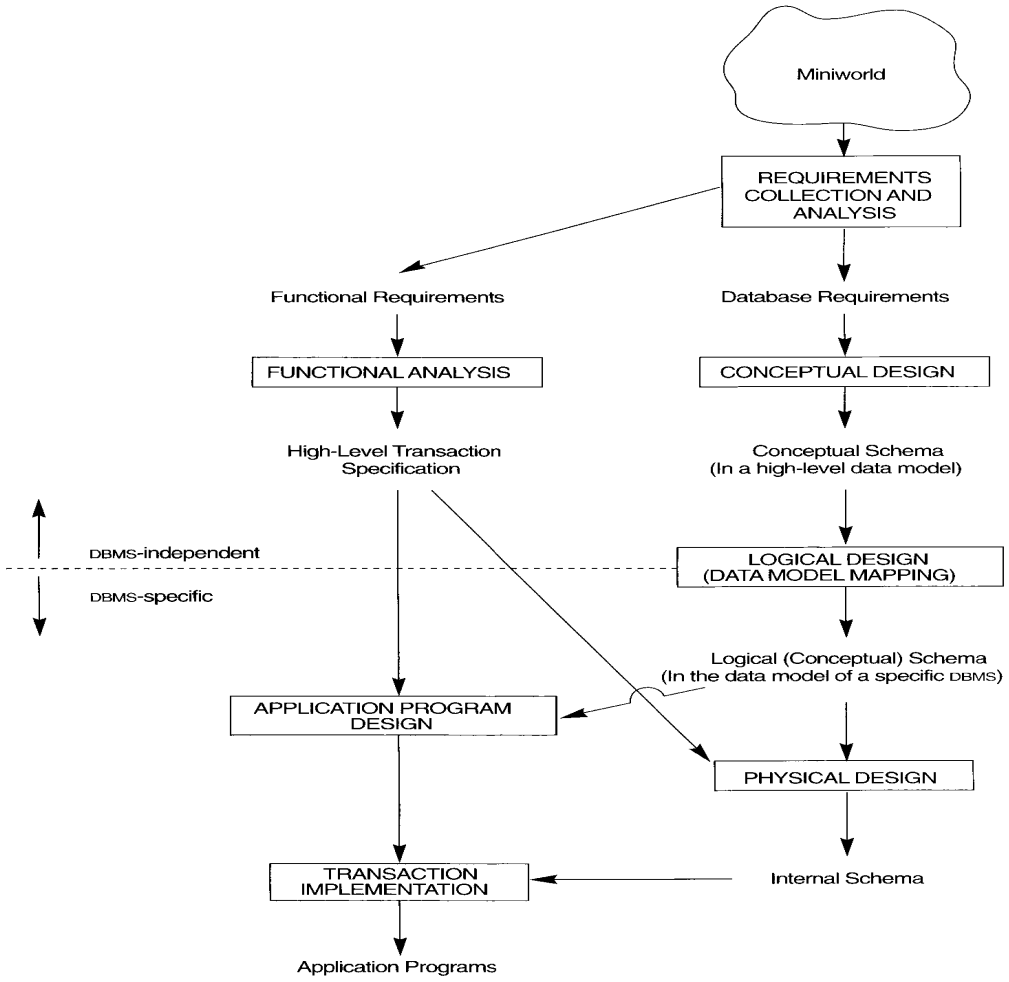
Dựa trên mô hình dữ liệu được HQTCSDL sử dụng: Mô hình quan hệ (Relational Data Model), Mô hình hướng đối tượng (Object Data Model), Mô hình mạng (Network Data Model), mô hình phân cấp (Hierarchical Data Model), mô hình quan hệ thực thể (Entity – Relationship Data Model)...

Tiêu chí khác:

- Số người sử dụng HQTCSDL: Đơn người dùng (Single-user), Đa người dùng (multi-user). Hầu hết các HQTCSDL hiện nay đều là HQTCSDL đa người dùng.
- Vị trí HQTCSDL: tập trung (sử dụng một máy đơn) hay phân tán (sử dụng nhiều máy tính).
- Giá của HQTCSDL: 100$- 300$; 10000$- 100000$

3 Chương 3. MÔ HÌNH QUAN HỆ - THỰC THỂ (Entity – Relationship Model)

3.1 Sử dụng mô hình dữ liệu khái niệm mức cao để thiết kế cơ sở dữ liệu



Hình 3.1. Các giai đoạn thiết kế cơ sở dữ liệu

Hình trên chỉ ra tiến trình thiết kế cơ sở dữ liệu một cách đơn giản. Bước đầu tiên là *tập hợp và phân tích yêu cầu hệ thống*. Trong bước này, người thiết kế cơ sở dữ liệu phải tiến hành thu thập yêu cầu của người sử dụng, sau đó viết tài liệu *những yêu cầu dữ liệu.* Kết quả của bước này là viết ra được tập hợp những yêu cầu tất cả các đối tượng người dùng một cách xúc tích. Từ đó, ta xác định được yêu cầu chức năng *(Funtional Requirements)* của hệ thống.

Sau khi tất cả các yêu cầu đã được tập hợp và phân tích, bước tiếp theo là xây dựng *lược đồ khái niệm(conceptual schema)* cho cơ sở dữ liệu. Lược đồ khái niệm là nơi biểu diễn xúc tích những yêu cầu của người sử dụng và biểu diễn chi tiết những kiểu thực thể *(entity types),*quan hệ *(relationships)* và những ràng buộc *(constraints)* của dữ liệu, phần này sử dụng những khái niệm được cung cấp trong mô hình dữ liệu mức cao *(High level data model)*. Mô hình dữ liệu mức cao là mô hình dữ liệu được dùng để giao tiếp với người sử dụng bình thường vì thế nó rất dễ hiểu, nó chỉ ra *cái gì*

cần được lưu trong cơ sở dữ liệu chứ không chỉ ra cụ thể dữ liệu được thực hiện *như thế nào*.

Bước tiếp theo trong quá trình thiết kế là cài đặt cơ sở dữ liệu trên một *mô hình dữ liệu thực hiện*, sử dụng các Hệ quản trị cơ sở dữ liệu nào đó (Hầu hết các hệ quản trị cơ sở dữ liệu hiện nay sử dụng mô hình dữ liệu quan hoặc hướng đối tượng). Vì thế, chúng ta cần thiết phải chuyển từ mô hình dữ liệu mức cao sang những mô hình dữ liệu thực hiện. Bước này được gọi là thiết kế logic *(Logical design)* hoặc ánh xạ mô hình dữ liệu *(Data model mapping)* và kết quả của bước này là *lược đồ cơ sở dữ liệu* trong mô hình cơ sở dữ liệu thực hiện.

Bước cuối cùng là thiết kế vật lý cho cơ sở dữ liệu (*physical design*), bao gồm việc thiết kế những cấu trúc lưu trữ dữ liệu bên trong, đường dẫn truy cập, tổ chức file của các file dữ liệu.

Trong chương này, chúng ta sẽ sử dụng *mô hình khái niệm ER* cho thiết kế lược đồ khái niệm (*Conceptual Schema*).

3.2 Mục đích của mô hình khái niệm ER(Entity – Relationship Model)

Qua bước xem xét tổng quát trên, ta thấy rằng, mô hình E-R là một mô tả logic chi tiết dữ liệu của một tổ chức hay một lĩnh vực nghiệp vụ. Nó giúp người thiết kế cơ sở dữ liệu mô tả thế giới thực gần gũi với quan niệm và cách nhận nhìn nhận bình thường của con người. Nó là công cụ để phân tích thông tin nghiệp vụ.

Mục đích của mô hình E – R:

- Làm thống nhất quan điểm về dữ liệu của những người tham gia hệ thống: Người quản lý, người dùng cuối, người thiết kế hệ thống
- Xác định các xử lý về dữ liệu cũng như các ràng buộc trên các dữ liệu.
- Giúp đỡ việc thể hiện cơ sở dữ liệu về mặt cấu trúc: Sử dụng thực thể và các mối liên kết giữa các thực thể. Biểu diễn mô hình quan hệ thực thể bằng một sơ đồ.

3.3 Ví dụ về một cơ sở dữ liệu ứng dụng

Trong phần này, chúng ta sẽ tìm hiểu một ví dụ về cơ sở dữ liệu ứng dụng - gọi là *COMPANY* để minh hoạ các khái niệm trong mô hình ER và sử dụng nó trong khi thiết kế lược đồ khái niệm.

Cơ sở dữ liệu *COMPANY* cần lưu trữ thông tin về nhân viên *(employees)*, phòng/ban *(departments)*, và các dự án *(projects)* trong công ty. Sau khi tập hợp được tất cả các yêu cầu của hệ thống, người thiết kế cơ sở dữ liệu tiến hành mô tả lại *miniworld* bằng mô hình ER.

Sau đây là các quy tắc nghiệp vụ của hệ thống cơ sở dữ liệu *COMPANY*, chúng ta sẽ xây dựng mô hình ER từng bước để giới thiệu các khái niệm của mô hình ER.

1. Công ty (COMPANY) có nhiều phòng/ban (DEPARTMENTs). Mỗi phòng/ban có tên (*name*), mã số (*number*) duy nhất và có một nhân viên (*employee*) làm quản lý (*manages*) phòng/ban. Chúng ta lưu lại ngày bắt đầu (*start date*) làm quản lý phòng/ban của nhân viên đó.

2. Mỗi phòng/ban có thể có nhiều địa điểm khác nhau (*locations*).

3. Mỗi phòng/ban điều hành một số dự án (PROJECTs). Mỗi dự án có tên *(name)*, mã số *(number)* duy nhất và chỉ có một địa điểm *(location)*.

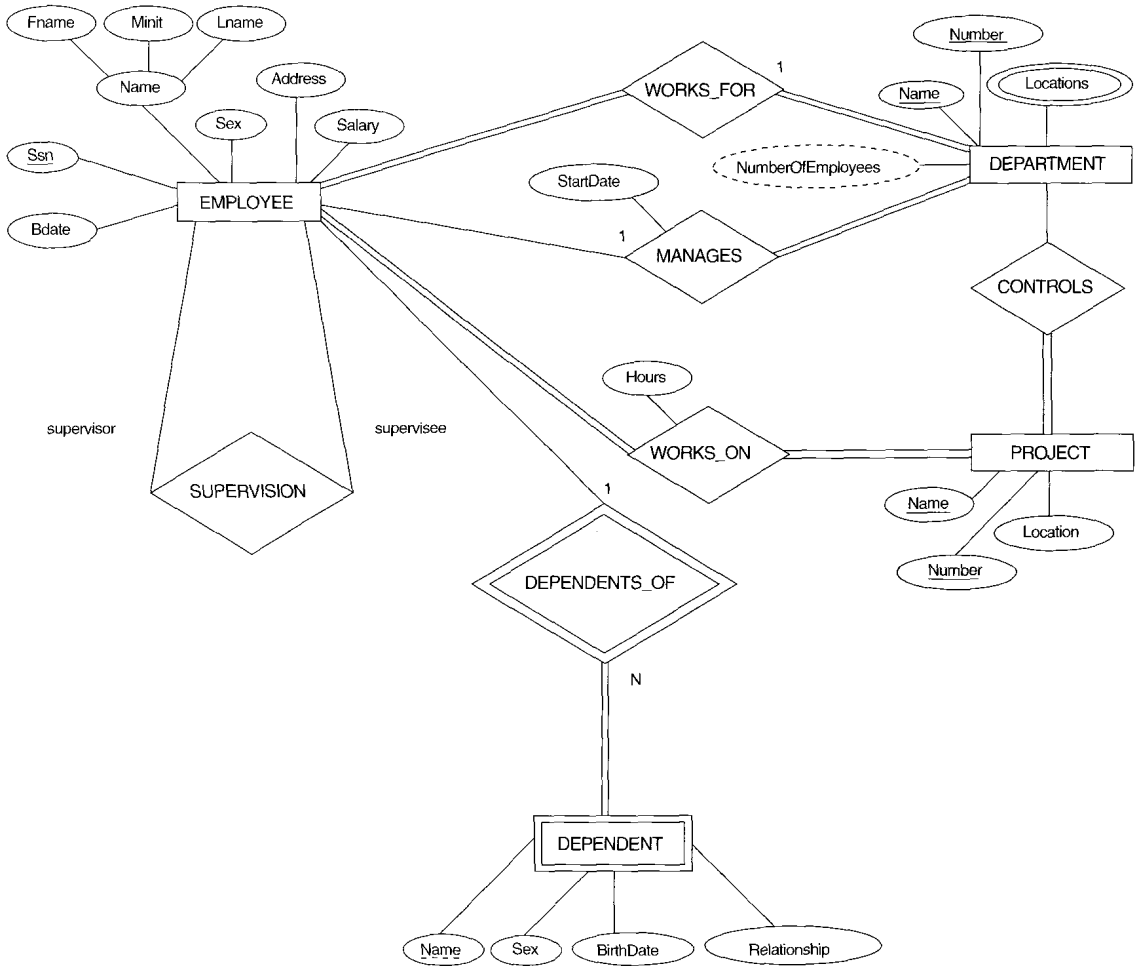
4. Với mỗi nhân viên, chúng ta lưu lại những thông tin sau: tên *(name)*, số bảo hiểm xã hội *(social security number)*, địa chỉ *(address)*, lương *(salary)*, giới tính *(sex)*, và ngày sinh *(birth date)*.

5. Mỗi nhân viên làm việc ở một phòng/ban, nhưng có thể làm việc cho nhiều dự án. Chúng ta lưu lại số giờ làm việc *(the number of hours per week)* của từng nhân viên trong từng dự án.

6. Chúng ta lưu lại thông tin về người quản lý trực tiếp *(direct supervisor)*, của mỗi nhân viên. Người quản lý trực tiếp cũng là một nhân viên.

7. Mỗi nhân viên có những người phụ thuộc vào họ (DEPENDENTs). Mỗi người phụ thuộc ta lưu lại thông tin về tên *(name)*, giới tính *(sex)*, ngày sinh *(birth date)* và quan hệ *(relationship)*.

Hình minh hoạ sau sẽ chỉ ra sơ đồ ER của cơ sở dữ liệu COMPANY dựa trên những ký hiệu đồ hoạ được quy định trong mô hình ER.



Hình 3.2. Sơ đồ ER của cơ sở dữ liệu COMPANY

Lưu ý, trên đây chỉ là giới thiệu sơ bộ về mô hình ER, các bạn chưa cần phải hiểu được toàn bộ sơ đồ trên.

Phần tiếp theo chúng ta sẽ giới thiệu về mô hình ER. Các khái niệm cơ bản trong mô hình ER gồm có **Thực thể** *(Entity)*, **Thuộc tính** *(Attributes)*, và **Quan hệ** *(Relationship)*.

3.4 Kiểu thực thể(Entity Type), Thuộc tính (Attributes), Khoá (Keys)

3.4.1 Thực thể (Entities) và thuộc tính (Attributes)

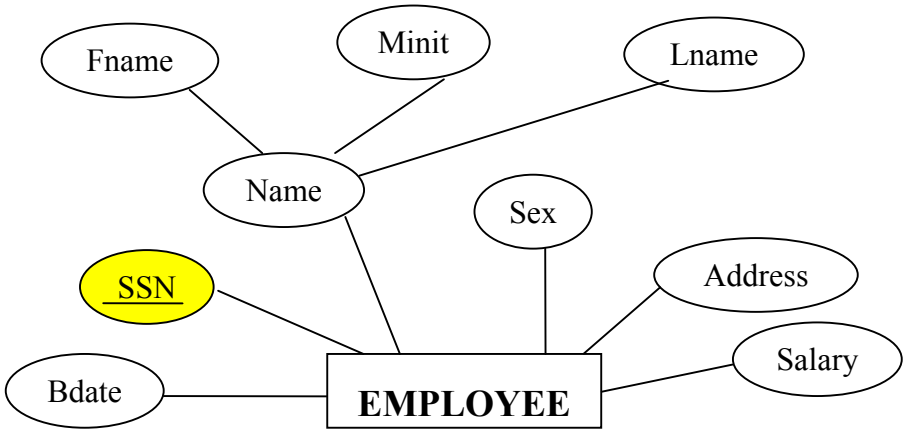
3.4.1.1 Thực thể

- Các thực thể vốn tồn tại trong thế giới thực. Một thực thể là một khái niệm để chỉ một lớp các đối tượng cụ thể hay các khái niệm có cùng những đặc tính chung mà ta quan tâm.
- Tên thực thể: Là tên của một lớp đối tượng. Trong 1 CSDL, tên thực thể không được trùng nhau.
- Ký hiệu: Trong mô hình E-R, thực thể được biểu diễn bằng một hình chữ nhật có tên bên trong.
- Ví dụ:

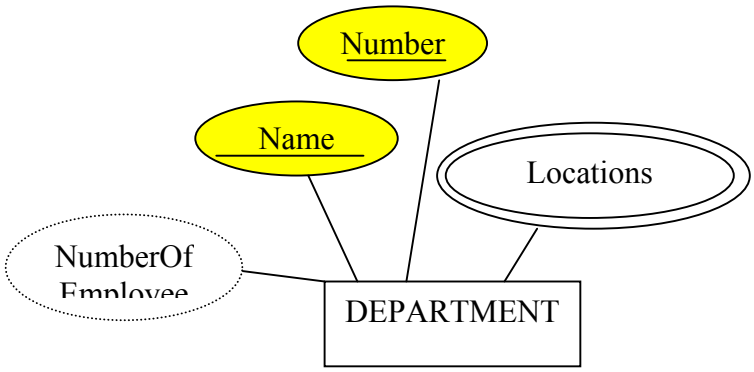
EMPLOYEE DEPARTMENT

3.4.1.2 Thuộc tính

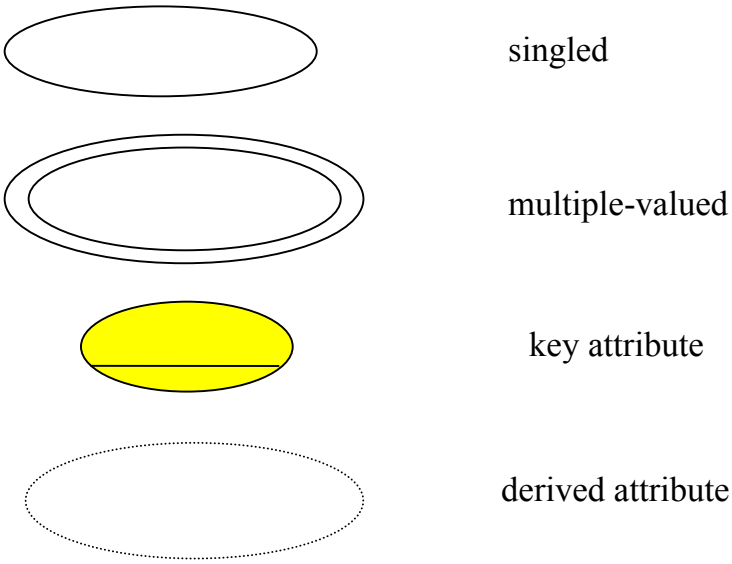
- Thuộc tính là các đặc trưng (*properties)* được sử dụng để biểu diễn thực thể.
- Ví dụ: Thực thể EMPLOYEE có các thuộc tính: *Name, SSN, Address, Sex, BirthDate.*
- Thuộc tính được ký hiệu bằng hình oval, bên trong ghi tên của thuộc tính. Thuộc tính của thực thể nào thì sẽ được gắn với thực thể đó.
- Ví dụ:



Hình 3.3. Thực thể EMPLOYEE và các thuộc tính của nó



Hình 3.4. Thực thể DEPARTMENT và các thuộc tính của nó



Hình 3.5. Một số ký hiệu của thuộc tính

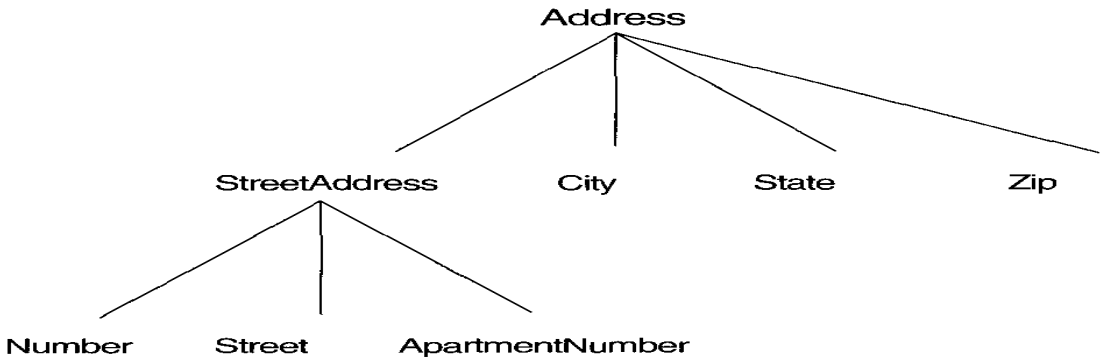
Các kiểu thuộc tính trong mô hình ER:

Thuộc tính đơn (*simple*) đối lập với **thuộc tính tổ hợp** (*composite*), **thuộc tính đơn trị** (*single-value*) đối lập với **thuộc tính đa trị** (*multivalued*), **thuộc tính lưu trữ** (*stored*) đối lập với **thuộc tính suy diễn** *(derived)***.** Sau đây ta sẽ tìm hiểu chi tiết về các loại thuộc tính này:

Thuộc tính đơn (simple) hay còn gọi là thuộc tính nguyên tử (Atomic): Chỉ có một giá trị trong một thuộc tính của một thực thể. Ví dụ: Thuộc tính *Birthdate, Sex… của Employee* là thuộc tính nguyên tử.

Thuộc tính tổ hợp (Composite): là thuộc tính được kết hợp của một số thành phần. Ví dụ: *Address(Apt#, House#, Street, City, State, ZipCode, County)* hoặc *Name (FirstName, MiddleName, LastName)* là thuộc tính tổ hợp.

Thuộc tính tổ hợp có thể được biểu diễn phân cấp như sau:



Hình 3.6. Biểu diễn phân cấp của thuộc tính tổ hợp

Thuộc tính đơn trị (single-value): Là thuộc tính chỉ có một giá trị duy nhất ở một thời điểm. Ví dụ: *Sex, Birthdate,...*

Thuộc tính đa trị (multivalued): Là thuộc tính có thể có nhiều giá trị tại một thời điểm. Ví dụ: PreviousDegrees…

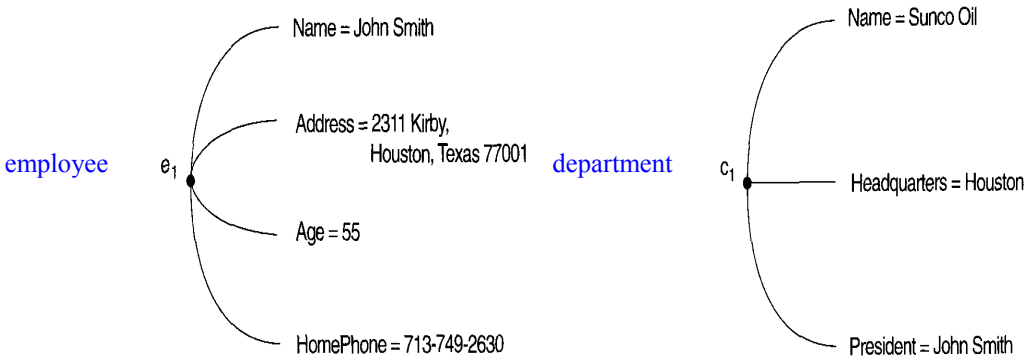
Ký hiệu: {PreviousDegrees}

Thuộc tính tổ hợp và thuộc tính đa trị có thể làm tổ ở nhiều mức, mặc dù ít gặp trường hợp này. Ví dụ: *PreviousDegrees* của thực thể STUDENT là loại thuộc tính đó. Ký hiệu: *{PreviousDegrees (College, Year, Degree, Field)}*.

Thuộc tính lưu trữ (stored attribute) và thuộc tính suy diễn (derived attribute): Thuộc tính lưu trữ là thuộc tính mà giá trị của nó phải được lưu trữ, còn thuộc tính suy diễn là thuộc tính mà giá trị của nó có thể suy ra từ giá trị của những thuộc tính khác. Ví dụ: *Age(derived attribute)* được suy diễn từ *BirthDate (stored attribute)*

Giá trị rỗng của thuộc tính (Null Values): Trong một vài trường hợp, một thực thể có thể không có giá trị tương ứng cho một thuộc tính, ví dụ thuộc tính *NameDependent*(Tên của người phụ thuộc), nếu một nhân viên nào đó trong thực thể EMPLOYEE chưa có người phụ thuộc thì thuộc tính NameDependent tương ứng với nhân viên đó sẽ không có giá trị. Hoặc trong trường hợp thuộc tính có giá trị nhưng chưa được biết, ví dụ thuộc tính *PhoneNumber* (Số điện thoại). Trong trường hợp này, một giá trị đặc biệt được tạo ra, đó là giá trị **Null**.

Mỗi thuộc tính trong thực thể luôn có giá trị, ví dụ các thuộc tính trong thực thể EMPLOYEE có các giá trị sau: Name=*'John Smith'*, SSN=*'123456789'*, Address='*731 Fondren, Houston, TX*', Sex='*M*', BirthDate= '*09-JAN-55*'. Một bộ giá trị của một thực thể được gọi là một **bản ghi** *(record)*.



Hình 3.7. Hai thực thể và giá trị thuộc tính của nó

3.4.2 Kiểu thực thể, Khoá và tập giá trị

Kiểu thực thể: là tập hợp những thực thể có cùng các thuộc tính cơ bản. Ví dụ, kiểu thực thể EMPLOYEE, kiểu thực thể PROJECT.

Khoá: Mỗi một kiểu thực thể phải có một hoặc một tập các thuộc tính mang giá trị duy nhất (unique value) để phân biệt giữa bản ghi này với bản ghi khác. Thuộc tính đó gọi là khoá của kiểu thực thể (*Key attribute*).Ví dụ: thuộc tính *SSN* của kiểu thực thể EMPLOYEE, hoặc thuộc tính *NumberStudent(Mã sinh viên)* của kiểu thực thể STUDENT. Chú ý là khoá có thể gồm một hoặc một tập các thuộc tính.

Tập giá trị hay còn gọi là miền xác định(Domain): là tập những giá trị mà thuộc tính có thể nhận được. Ví dụ: Miền xác định của thuộc tính Sex là {Male, Female}, hoặc của Mark(Điểm) là từ 0..10.

Tập giá trị không được biểu diễn trong lược đồ ER.

3.5 Liên kết, Kiểu liên kết và các Ràng buộc liên kết

3.5.1 Định nghĩa liên kết và kiểu liên kết

Liên kết (Relationship) dùng để chỉ mối quan hệ giữa hai hay nhiều thực thể khác nhau. Ví dụ: Nhân viên (A) làm việc cho dự án (X), nhân viên B làm việc cho dự án (X)…

Những liên kết của cùng một kiểu được nhóm lại gọi là kiểu liên kết (Relationship Type), ví dụ kiểu liên kết WORK_ON (làm việc cho), kiểu liên kết MANAGES (làm quản lý)…

Trong lược đồ ER, người ta sử dụng hình thoi và bên trong ghi tên kiểu liên kết để ký hiệu kiểu liên kết.

Ký hiệu:

Lưu ý: Kiểu liên kết cũng có thể có thuộc tính của nó.

3.5.2 Bậc của kiểu liên kết

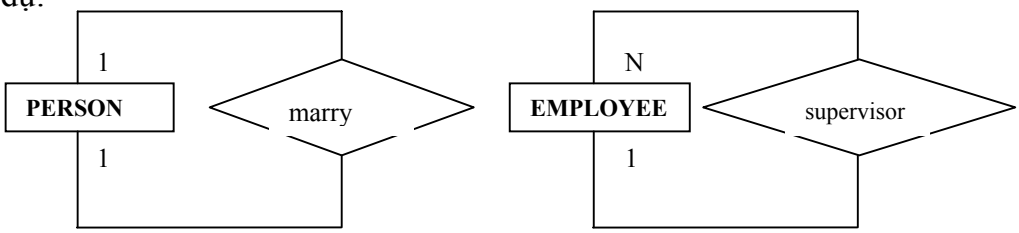
Là số lượng các kiểu thực thể tham gia vào liên kết. Có các kiểu liên kết sau:

- Kiểu liên kết bậc 1 (đệ quy) là mối quan hệ giữa cùng 1 kiểu thực thể.
- Kiểu liên kết bậc 2 là mối liên kết giữa hai kiểu thực thể
- Kiểu liên kết bậc 3 là mối liên kết giữa 3 kiểu thực thể

3.5.2.1 Mối quan hệ bậc 1

Mối quan hệ bậc 1 (đệ quy) là mối quan hệ giữa cùng 1 kiểu thực thể

Ví dụ:

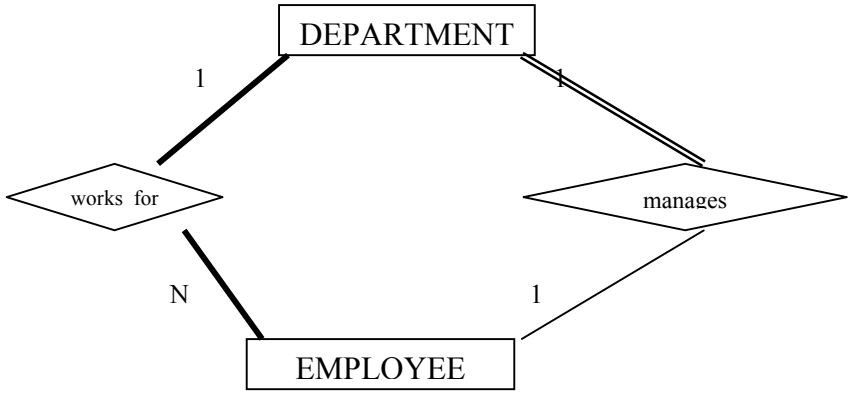


Hình 3.8. Minh họa mối quan hệ bậc 1

3.5.2.2 Mối quan hệ bậc 2

Là mối quan hệ giữa 2 kiểu thực thể khác nhau.

Ví dụ:



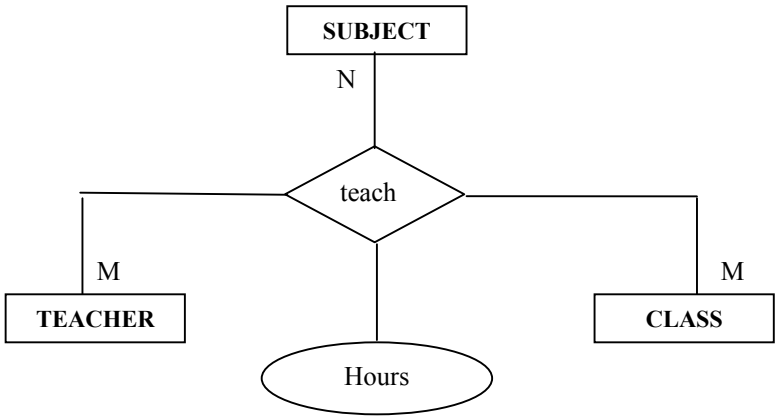
Hình 3.9. Minh họa mối quan hệ bậc 2

Ví dụ: Mối liên kết giữa hai kiểu thực thể DEPARTMENT và EMPLOYEE sau đây là kiểu liên kết bậc 2 vì nó có sự tham gia của hai kiểu thực thể.

Hình minh hoạ trên còn cho ta thấy, có thể có nhiều hơn một kiểu liên kết giữa hai kiểu thực thể khác nhau.

3.5.2.3 Mối quan hệ bậc 3

Là mối quan hệ giữa 3 kiểu thực thể khác kiểu.



Hình 3.10. Minh họa mối quan hệ bậc 3

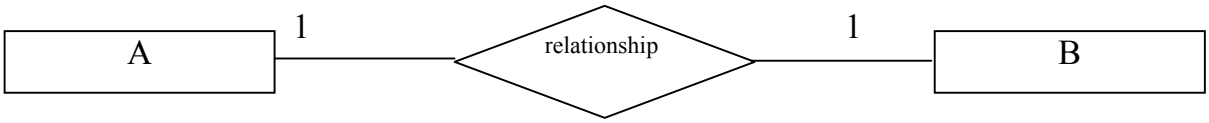
3.5.3 Ràng buộc liên kết

Các kiểu liên kết thường có một số ràng buộc nào đó về các thực thể có thể kết hợp với nhau tham gia trong một liên kết phù hợp. Các ràng buộc này xác định từ tình huống thực tế mà liên kết thể hiện.Có các loại ràng buộc như sau:

3.5.3.1 Tỷ số lực lượng:

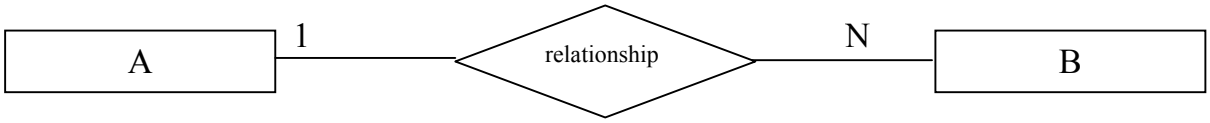
Trong các kiểu liên kết bậc 2, tỷ số lực lượng chỉ rõ số thực thể tham gia vào liên kết. Các tỷ số lực lượng có thể là: 1:1, 1:N, N:1 và M:N.

Tỷ số 1:1: Một thực thể của kiểu A có liên kết với một thực thể của kiểu B và ngược lại.

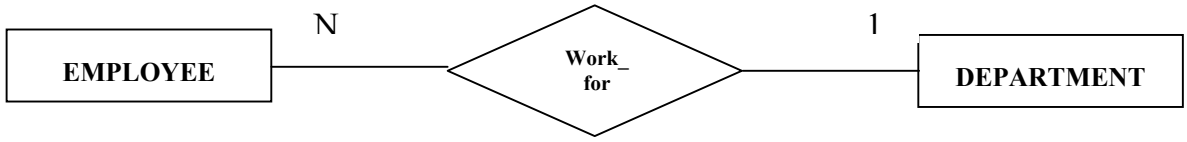


Ví dụ: Một nhân viên (EMPLOYEE) quản lý một phòng (DEPARTMENT) , và một phòng chỉ có một nhân viên quản lý.

Tỷ số 1:N: Một thực thể của kiểu A có liên kết với nhiều thực thể của kiểu B. Nhưng một thực thể của kiểu B lại có liên kết duy nhất với thực thể của kiểu A.



Ví dụ: Một nhân viên (EMPLOYEE) làm việc cho một phòng (DEPARTMENT), và một phòng có nhiều nhân viên làm việc.



Tỷ số M:N: Một thực thể của kiểu A có liên kết với nhiều thực thể của kiểu B và ngược lại.

Ví dụ:

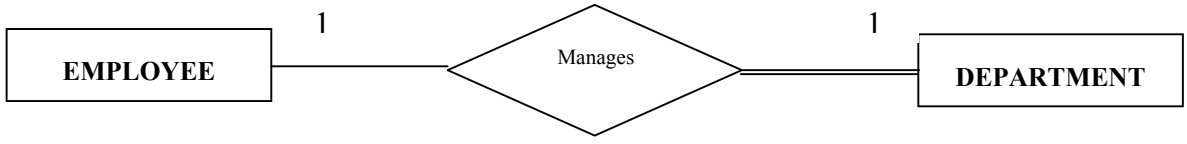
3.5.3.2 Ràng buộc về sự tham gia liên kết (Participation constraint)

Ràng buộc về sự tham gia liên kết được xác định trên từng thực thể trong từng kiểu liên kết mà thực thể đó tham gia, bao gồm: lực lượng tham gia toàn bộ (*total participation*) và lực lượng tham gia bộ phận (*partial participation*).

Ví dụ: Trong kiểu liên kết Manages giữa hai kiểu thực thể EMPLOYEE và DEPARTMENT, lực lượng tham gia của kiểu thực thể DEPARTMENT là toàn bộ, vì DEPARTMENT nào cũng có người quản lý, còn lực lượng tham gia của kiểu thực thể EMPLOYEE là bộ phận vì không phải EMPLOYEE nào cũng làm quản lý (manages) của DEPARTMENT.

Trong sơ đồ ER, kiểu thực thể có lực lượng tham gia liên kết toàn bộ được nối với kiểu liên kết bằng gạch nối kép, còn kiểu thực thể có lực lượng tham gia bộ phận được nối với kiểu liên kết bằng gạch nối đơn.

Ví dụ:



3.5.3.3 Lực lượng tham gia liên kết

Trong mối liên kết giữa các thực thể, ta cần quan tâm đến lực lượng tham gia liên kết, đó là số bản ghi lớn nhất và nhỏ nhất của thực thể tham gia vào liên kết đó.

Ký hiệu: Thêm (min,max) vào mối liên kết.

Trong đó:

- min là số bản ghi nhỏ nhất tham gia vào liên kết

- max là số bản ghi lớn nhất tham gia vào liên kết
- Mặc định, min=0, max=n
- Chúng ta xác định lực lượng này từ khảo sát thực tế bài toán.

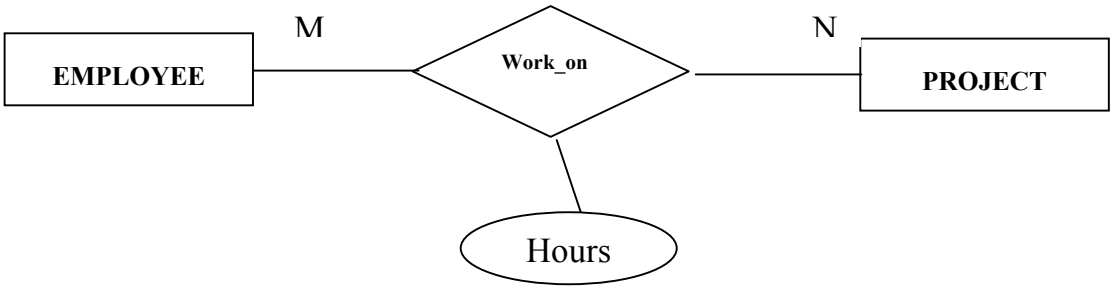
Ví dụ:

a. Tại một thời điểm, một phòng có duy nhất một người quản lý-người đó là nhân viên, một nhân viên chỉ quản lý duy nhất một phòng. Vì vậy, (0,1) là lực lượng của EMPLOYEE và (1,1) là lực lượng của DEPARTMENT tham gia trong liên kết Manages(quản lý).

b. Một nhân viên chỉ có thể làm việc cho một phòng nhưng một phòng có thể có bất kỳ số lượng nhân viên nào. Vì thế, (1,1) là lực lượng của EMPLOYEE và (0,n) là lực lượng của DEPARTMENT tham gia trong liên kết Works_For(làm việc cho).

3.5.3.4 Thuộc tính của kiểu liên kết

Kiểu liên kết cũng có thể có thuộc tính. Ví dụ: Số giờ nhân viên làm việc cho dự án (Hours) là thuộc tính của mối liên kết giữa hai kiểu thực thể EMPLOYEE và PROJECT.



3.6 Kiểu thực thể yếu(Weak Entity)

Kiểu thực thể yếu là kiểu thực thể tồn tại phụ thuộc vào thực thể khác (thực thể làm chủ hay còn gọi là xác định nó). Kiểu thực thể yếu không có khoá.

Kiểu thực thể yếu được xác định bằng:

- Một hay một tập các thuộc tính xác định kiểu thực thể yếu
- Và thực thể làm chủ (xác định) thực thể yếu.

Kiểu thực thể yếu luôn có lực lượng tham gia liên kết toàn bộ.

3.7 Tổng quát hóa và chuyên biệt hóa

Tổng quát hóa là khái niệm cho phép ta xem một vật thể nào đó (các thực thể) là một thực thể con của một vật thể khác tổng quát hơn.

Ví dụ: SACH là một loại con của loại tổng quát hơn là TAILIEU nói chung.

Chuyên biệt hóa là khái niệm ngược lại với tổng quát hóa.

Ví dụ:

Ôtô, xeca, taxi gộp lại thành một thực thể tổng quát hơn là Phuongtienvantai.

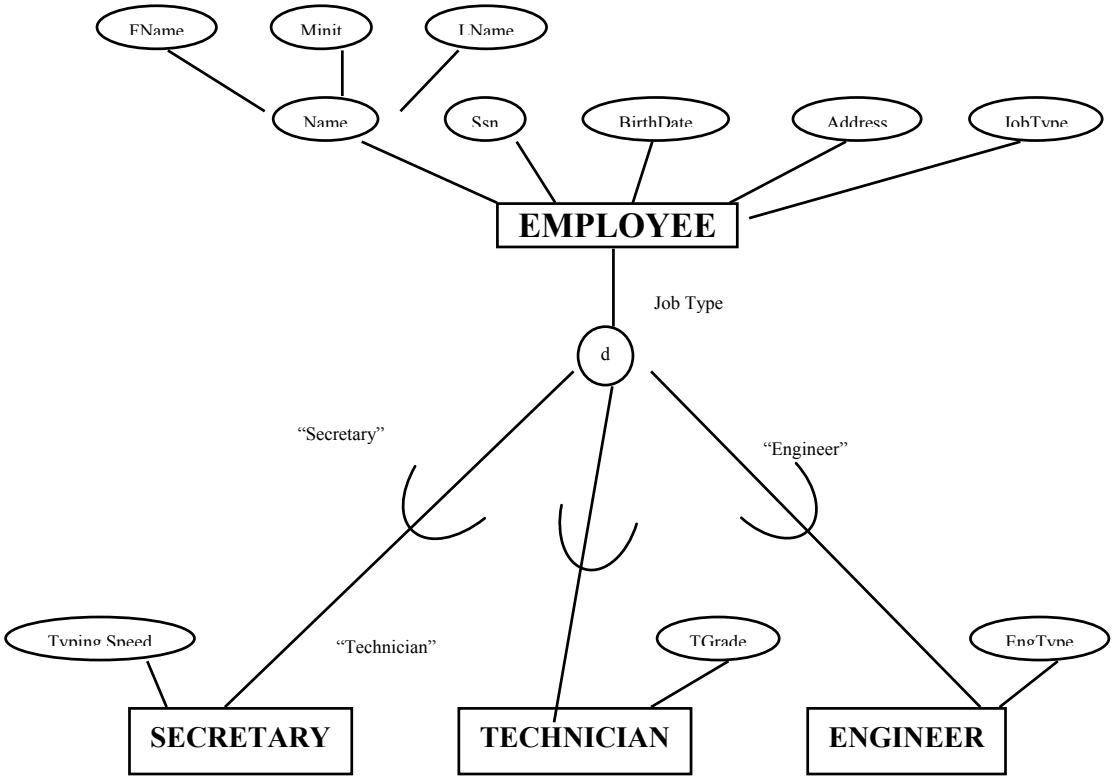
3.7.1 Thực thể con và thực thể chính

Trong mô hình dữ liệu, cần phải mô tả một cách rõ ràng các thực thể gần như nhau: đó là các thực thể có các thuộc tính chung, nhưng cũng có một số các thuộc tính khác nhau.

Ví dụ: Trong một Đơn vị có 3 loại nhân viên: SECRETARY, TECHNICIAN, ENGINEER. Các thực thể này có một số thuộc tính chung và một số thuộc tính riêng.

Trong trường hợp này, có thể có 3 hướng giải quyết sau:

- Cách đơn giản nhất là gộp tất cả các loại nhân viên vào một thực thể EMPLOYEE. Cách này dẫn đến dư thừa thông tin, vì có những thuộc tính luôn rỗng đối với mỗi loại nhân viên.
- Cách thứ hai, định nghĩa riêng rẽ từng loại thực thể: SECRETARY, TECHNICIAN, ENGINEER. Cách này không khai thác được những thuộc tính chung của nhân viên.
- Cách thứ ba, định nghĩa một thực thể chính gọi là EMPLOYEE, với 3 thực thể con là: SECRETARY, TECHNICIAN, ENGINEER. Những thuộc tính chung nằm trong thực thể chính, còn thực thể con sẽ chứa thuộc tính riêng của nó. Hình sau đây minh hoạ cách giải quyết này:



Hình 3.11. Minh họa mối quan hệ giữa thực thể con và thực thể chính

3.7.2 Các thực thể con loại trừ

Trong mô hình trên, các thực thể con là loại trừ lẫn nhau.

Thực thể con loại trừ gồm 2 loại:

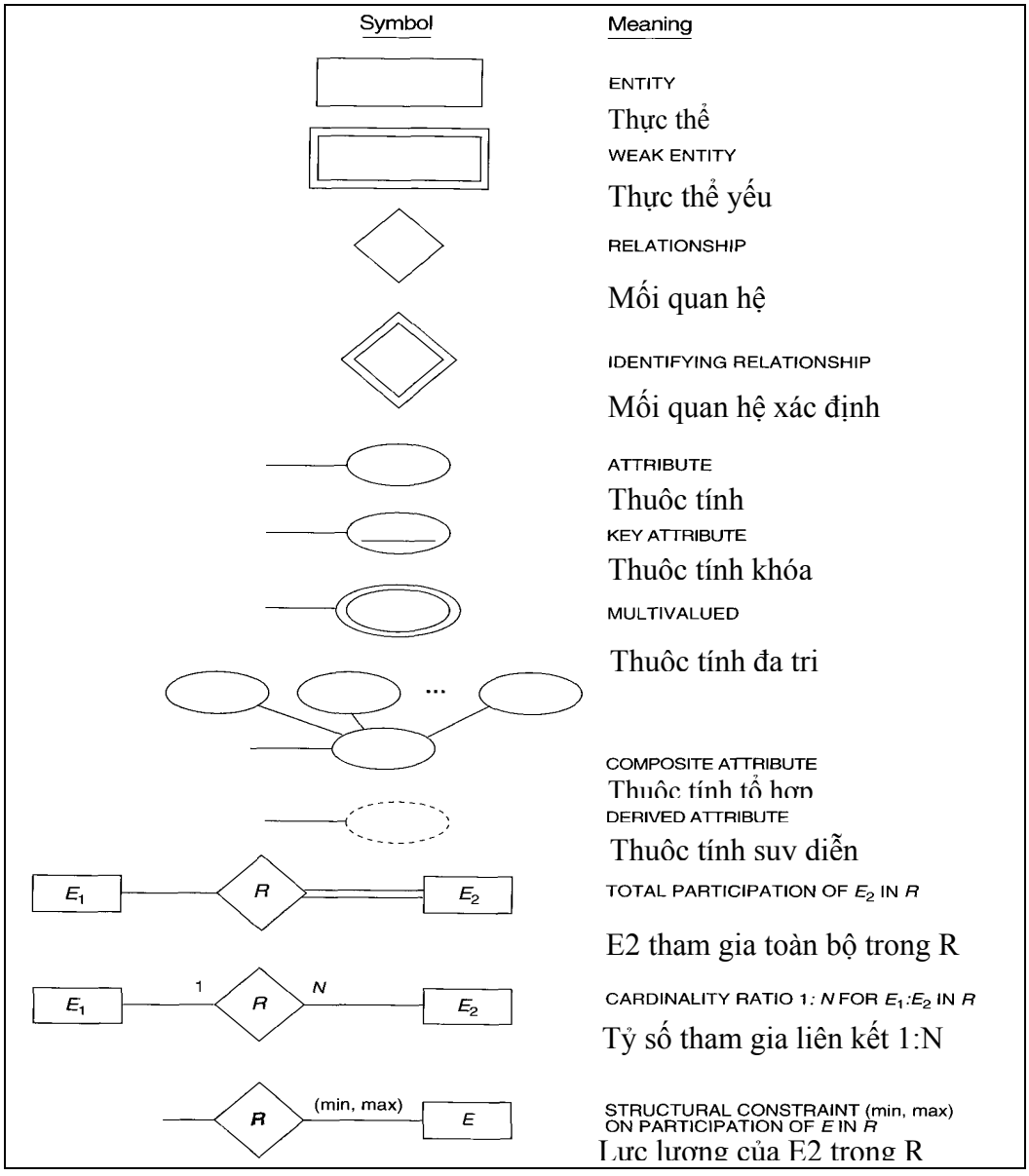
Thực thể con đầy đủ: Là tất cả các thực thể con xác định một thực thể chính. Trong ví dụ trên, tất cả các thực thể con là đầy đủ vì không thể bổ sung thực thể nào vào thực thể chính EMPLOYEE.

Thực thể con không đầy đủ: Tập các thực thể con không đầy đủ xác định thực thể chính. Ví dụ: Tập thực thể ÔTÔ, XEMAY chưa xác định được thực thể chính là PHUONGTIEN.

3.8 Các ký hiệu và quy ước đặt tên trong mô hình ER

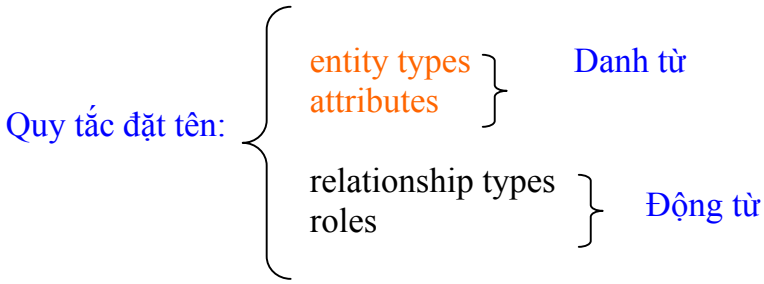
3.8.1 Các ký hiệu

Trong xây dựng mô hình E-R, ta sử dụng các ký hiệu trong hình 3.12:



Hình 3.12. Bảng tổng hợp các ký hiệu trong mô hình ER

3.8.2 Quy tắc đặt tên



3.9 Xây dựng một mô hình ER.

3.9.1 Các bước xây dựng sơ đồ ER

3.9.1.1 Liệt kê, chính xác hóa và lựa chọn thông tin cơ sở

Xác định một từ điển bao gồm tất cả các thuộc tính (không bỏ sót bất cứ thông tin nào).

Chính xác hóa các thuộc tính đó. Thêm các từ cần thiết để thuộc tính đó mang đầy đủ ý nghĩa, không gây lầm lẫn, hiểu nhầm.

Chú ý: Để lựa chọn các đặc trưng cần thiết , ta duyệt từ trên xuống và chỉ giữ lại những thuộc tính đảm bảo yêu cầu sau:

Thuộc tính cần phải đặc trưng cho một lớp các đối tượng được xét.

Chọn một thuộc tính một lần, nếu lặp lại thì bỏ qua.

Một thuộc tính phải là sơ cấp (Nếu giá trị của nó có thể suy ra từ các thuộc tính khác thì bỏ qua).

3.9.1.2 Xác định các thực thể và các thuộc tính của nó, sau đó xác định thuộc tính định danh cho từng thực thể.

Duyệt danh sách các thuộc tính từ trên xuống để tìm ra thuộc tính tên gọi. Mỗi thuộc tính tên gọi sẽ tương ứng với một thực thể.

Gán các thuộc tính cho từng thực thể.

Xác định thuộc tính định danh cho từng thực thể.

3.9.1.3 Xác định các mối quan hệ và các thuộc tính riêng của nó

Xét danh sách các thuộc tính còn lại, hãy tìm tất cả các động từ (ứng với thuộc tính đó). Với mỗi động từ, hãy trả lời các câu hỏi: Ai? Cái gì? Ở đâu? Khi nào? Bằng cách nào?

3.9.1.4 Vẽ sơ đồ mô hình thực thể- mối quan hệ, xác định lực lượng tham gia liên kết cho các thực thể.

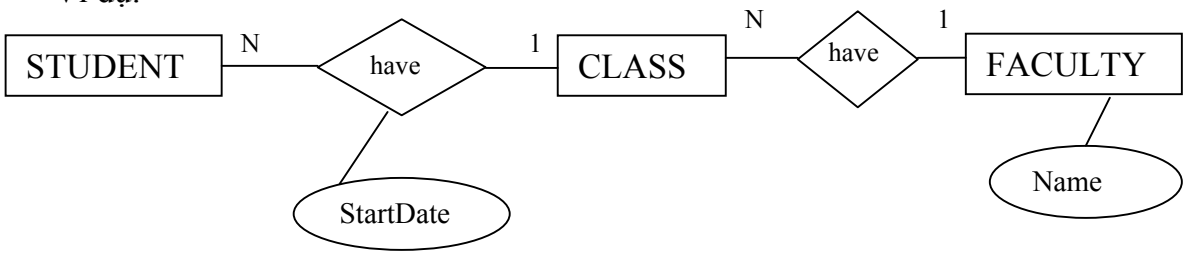
3.9.1.5 Chuẩn hóa sơ đồ và thu gọn sơ đồ

- Vẽ sơ đồ.
- Chuẩn hóa sơ đồ, nếu trong đó còn có chứa: các thuộc tính lặp, nhóm lặp và các thuộc tính phụ thuộc thời gian → sơ đồ chỉ còn các thực thể đơn và các thuộc tính đơn.

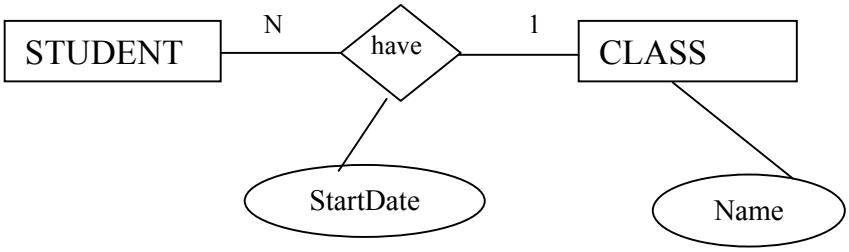
Thu gọn sơ đồ: Nếu một thực thể có tất cả các đặc trưng:

- Là thực thể treo: là thực thể chỉ tham gia vào một mối quan hệ và chỉ chứa một thuộc tính duy nhất thực sự là của nó (có thể có thuộc tính thứ 2 thêm vào làm định danh).
- Mối quan hệ là bậc hai và không có thuộc tính riêng.
- Mối quan hệ là 1: N hay 1:1.

Ví dụ:



Được thu gọn thành sơ đồ sau:



3.9.2 Mô hình ER cho cơ sở dữ liệu COMPANY

a. Qua Bước 1 và Bước 2 ta xác định được danh sách **các thực thể và các thuộc tính** của từng thực thể.

b. Qua bước 3 ta xác định được các **kiểu liên kết** như sau:

1. **MANAGES**: là kiểu liên kết 1:1 giữa hai kiểu thực thể EMPLOYEE và DEPARTMENT. Lực lượng tham gia kiên kết của kiểu thực thể EMPLOYEE là bộ phận, vì không phải nhân viên nào cũng tham gia quản lý. Còn lực lượng tham gia của DEPARTMENT là toàn bộ, vì tại bất kỳ thời điểm nào một phòng cũng có một nhân viên làm quản lý. Thuộc tính StartDate được gắn vào kiểu liên kết để ghi lại thời điểm bắt đầu làm quản lý của nhân viên cho phòng đó.

2. **WORKS_FOR:** là kiểu liên kết 1:N giữa hai kiểu thực thể DEPARTMENT và EMPLOYEE. Cả hai kiểu thực thể này đều có lực lượng tham gia toàn bộ vào liên kết.

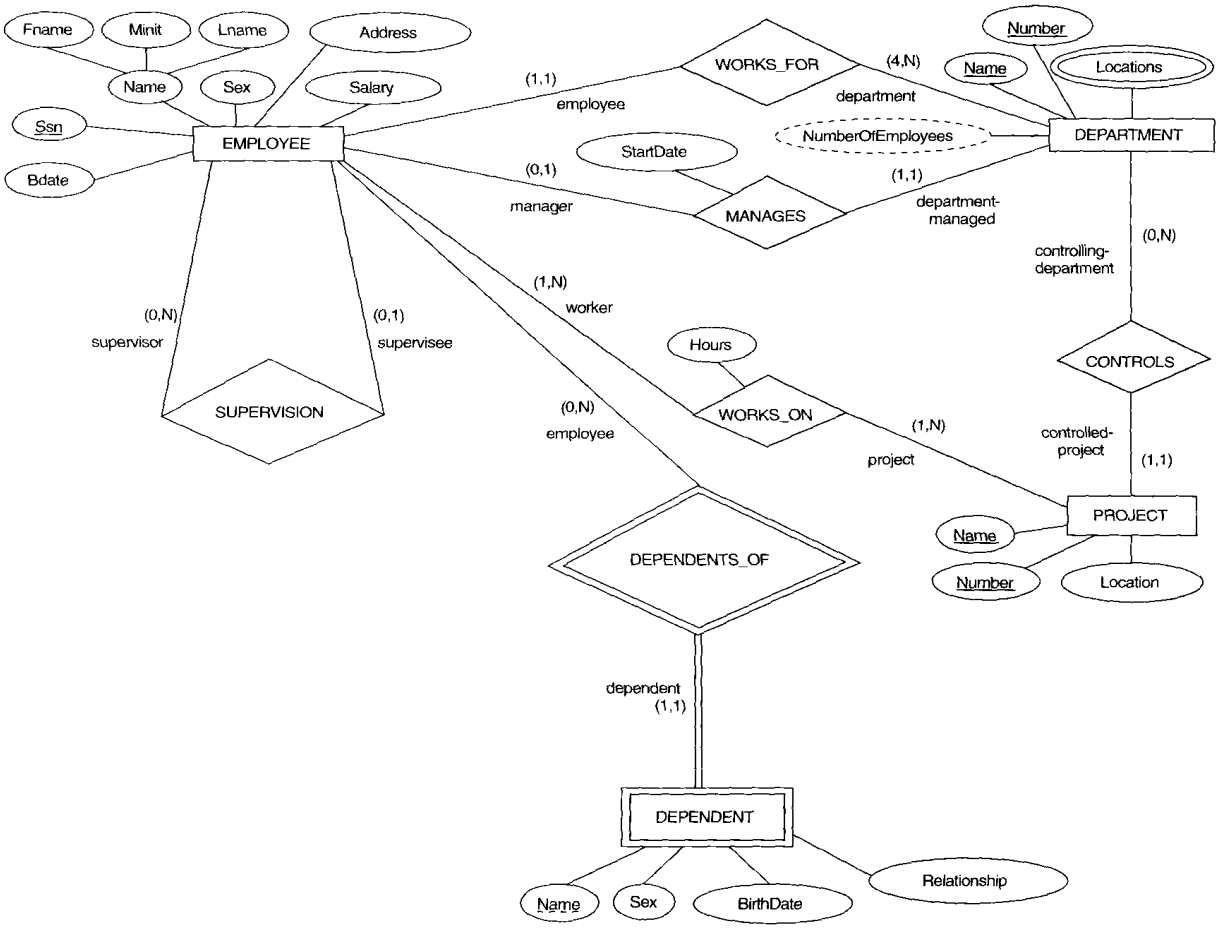
3. **CONTROLS**: là kiểu liên kết 1:N giữa hai kiểu thực thể DEPARTMENT và PROJECT. Lực lượng tham gia của PROJECT là toàn bộ, của DEPARTMENT là bộ phận.

4. **SUPERVISOR**: là kiểu liên kết 1:N giữa hai kiểu thực thể EMPLOYEE và EMPLOYEE (Mỗi nhân viên có người quản lý cấp trên của mình, người đó cũng là một nhân viên). Trong quá trình phỏng vấn các đối tượng người dùng, người thiết kế được trả lời rằng: Không phải nhân viên nào cũng làm quản lý nhân viên khác, và không phải nhân viên nào cũng có người quản lý trực tiếp mình. Vì vậy, cả hai kiểu thực thể này có lực lượng tham gia bộ phận.

5. **WORK_ON**: là kiểu liên kết M:N giữa hai kiểu thực thể EMPLOYEE và PROJECT, vì một dự án có nhiều nhân viên làm việc và một nhân viên có thể làm việc cho nhiều dự án. Thuộc tính Hours là thuộc tính của kiểu liên kết được dùng để ghi lại số giờ mỗi nhân viên làm việc cho một dự án nào đó. Cả hai kiểu thực thể này có lực lượng tham gia toàn bộ.

6**. DEPENDENTS_OF**: là kiểu liên kết 1:N giữa hai kiểu thực thể EMPLOYEE và DEPENDENT. Kiểu thực thể DEPENDENT là kiểu thực thể yếu, vì nó không tồn tại nếu không có sự tồn tại của kiểu thực thể EMPLOYEE. Lực lượng tham gia của EMPLOYEE là bộ phận, vì không phải nhân viên nào cũng có người phụ thuộc. Lực lượng tham gia của DEPENDENT là toàn bộ vì nó là kiểu thực thể yếu.

c. Qua bước 4 ta vẽ được mô hình ER:



Hình 3.13. Mô hình ER của bài toán COMPANY

3.9.3 Bài tập

Xây dựng mô hình ER để quản lý các đơn vị sau:

Bài tập 1: Quản lý hoạt động của một trung tâm đại học

Qua quá trình khảo sát, điều tra hoạt động của một trung tâm đại học ta rút ra các quy tắc quản lý sau:

- Trung tâm được chia làm nhiều **trường** và mỗi trường có 1 **hiệu trưởng** để quản lý nhà trường.
- Một trường chia làm nhiều **khoa**, mỗi khoa thuộc về một trường.
- Mỗi khoa cung cấp nhiều **môn học**. Mỗi môn học thuộc về 1 khoa (thuộc quyền quản lý của 1 khoa).
- Mỗi khoa thuê nhiều giáo viên làm việc. Nhưng mỗi **giáo viên** chỉ làm việc cho 1 khoa. Mỗi khoa có 1 chủ nhiệm khoa, đó là một giáo viên.
- Mỗi giáo viên có thể dạy nhiều nhất 4 môn học và có thể không dạy môn học nào.
- Mỗi sinh viên có thể học nhiều môn học, nhưng ít nhất là môn. Mỗi môn học có thể có nhiều sinh viên học, có thể không có sinh viên nào.
- Một khoa quản lý nhiều sinh viên chỉ thuộc về một khoa.
- Mỗi giáo viên có thể được cử làm chủ nhiệm của lớp, lớp đó có thể có nhiều nhất 100 sinh viên.

Bài tập 2:

Cho các thuộc tính, các quy tắc quản lý của một đơn vị.

1. Thuộc tính:

- Mã đơn vị, Tên đơn vị, Số điện thoại đơn vị, Địa chỉ đơn vị.
- Mã nhân viên, Tên nhân viên, Giới tính nhân viên, Địa chỉ nhân viên, Số điện thoại của nhân viên.
- Mã dự án, Tên dự án
- Mã khách hàng, tên khách hàng, Địa chỉ khách hàng, Số điện thoại của khách hàng.
- Mã hàng, Tên hàng, Số lượng trong kho.
- Lượng đặt hàng, Ngày đặt hàng

2. Các quy tắc

- Một đơn vị thuê 1 hoặc nhiều nhân viên
- Một đơn vị được quản lý bởi 1 người quản lý. Đó là một nhân viên.
- Một nhân viên chỉ làm việc cho 1 đơn vị
- Một nhân viên có thể làm việc cho 1 dự án
- Mỗi dự án có thể thuê 1 hoặc nhiều nhân viên

- Một nhân viên có thể phục vụ cho 1 hoặc nhiều khách hàng
- Một khách hàng có thể được 1 hoặc nhiều nhân viên phục vụ
- Một khách hàng có thể đặt 1 hoặc 1 vài hàng hóa (Khách hàng nào cũng đặt hàng: 1 hoặc nhiều mặt hàng)
- Mọi mặt hàng đều có ít nhất một khách hàng đặt mua
- Một đơn đặt hàng chỉ có 1 mặt hàng.

4 Chương 4. MÔ HÌNH CƠ SỞ DỮ LIỆU QUAN HỆ

4.1 Khái niệm mô hình quan hệ

Mô hình CSDL quan hệ lần đầu tiên được E.F.Codd và tiếp sau đó được công ty IBM giới thiệu vào năm 1970. Ngày nay, hầu hết các tổ chức đã áp dụng CSDL quan hệ để quản lý dữ liệu trong đơn vị mình.

Mô hình cơ sở dữ liệu quan hệ là cách thức biểu diễn dữ liệu dưới dạng bảng hay còn gọi là quan hệ, mô hình được xây dựng dựa trên cơ sở lý thuyết đại số quan hệ.

Cấu trúc dữ liệu: dữ liệu được tổ chức dưới dạng quan hệ hay còn gọi là bảng.

Thao tác dữ liệu: sử dụng những phép toán mạnh (bằng ngôn ngữ SQL).

4.2 Các thành phần cơ bản của mô hình

4.2.1 Một số khái niệm của mô hình quan hệ

Mô hình quan hệ là cách thức biểu diễn dữ liệu dưới dạng các quan hệ (các bảng). Một quan hệ là một bảng dữ liệu 2 chiều (cột và dòng), mô tả một thực thể. Mỗi cột tương ứng với một thuộc tính của thực thể. Mỗi dòng chứa các giá trị dữ liệu của một đối tượng cụ thể thuộc thực thể

Một số khái niệm cơ bản:

Lược đồ quan hệ: R(A1,…,An), trong đó R là tên quan hệ, Ai là các thuộc tính, mỗi Ai có miền giá trị tương ứng dom(Ai).

Lược đồ quan hệ được sử dụng để mô tả một quan hệ, bao gồm: Tên quan hệ, các thuộc tính và bậc của quan hệ (số lượng các thuộc tính)

4.2.2 Quan hệ:

Một quan hệ r của R(A1,...,An), ký hiệu r(R) là một tập hợp n-bộ r={ t1, ..., tm}

Trong đó:

Mỗi ti =<v1, ...,vn>, vi ∈ dom(Ai).

r(R) ⊆ dom(A1) x .... x dom(An)

r = { (vi_1,vi_2,...,vin) / i=1,...,m}

v_{11}	v_{12}	v_{1n}
v_{21}	v_{22}	v_{2n}
.....		
vm_1	vm_2	vmn
A_1	A_2	Am

Ta có Ai là các thuộc tính và miền giá trị của Ai là:

D1=dom(A_1), D_2=dom(A_2),...., Dn=dom(An).

Chú ý: - Các tập (D1,D2,...,Dn) là tập các miền trị của R

- n được gọi là bậc của quan hệ r.
- m được gọi là lực lượng của r.
- Quan hệ bậc 1 là quan hệ nhất nguyên, bậc 2 là quan hệ nhị nguyên, bậc n là quan hệ n nguyên.

Ví dụ: Quan hệ EMPLOYEE trên tập các thuộc tính R={SSN, Name, BDate, Address, Salary} là một quan hệ 5 ngôi.

SSN	Name	BDate	Address	Salary	
001	Đỗ Hoàng Minh	1960	Hà nội	425	t1
002	Đỗ Như Mai	1970	Hải Phòng	390	t2
003	Đặng Hoàng Nam	1973	Hà nội	200	t3

t1(001, 'Đỗ Hoàng Minh', 1960, 'Hà nội' , 425) = t1(R) là một bộ của quan hệ EMPLOYEE

4.2.3 Các tính chất của một quan hệ

- Giá trị đưa vào cột là đơn nhất
- Các giá trị trong cùng một cột phải thuộc cùng một miền giá trị (cùng kiểu)
- Thứ tự dòng cột tuỳ ý.

4.2.4 Các ràng buộc toàn vẹn trên quan hệ

Ràng buộc là những quy tắc được áp đặt lên trên dữ liệu đảm bảo *tính tin cậy* và *độ chính xác* của dữ liệu. Các luật toàn vẹn được thiết kế để giữ cho dữ liệu phù hợp và đúng đắn.

Có 4 kiểu ràng buộc chính: Ràng buộc miền giá trị (Domain Constraints), Ràng buộc khoá (Key Constraints), Ràng buộc thực thể (Entity Integrity Constraints), và Ràng buộc toàn vẹn tham chiếu (Referential Integrity Constraints).

4.2.4.1 Ràng buộc miền giá trị

Là một hợp các kiểu dữ liệu và những giá trị giới hạn mà thuộc tính có thể nhận được. Thông thường việc xác định miền giá trị của các thuộc tính bao gồm một số các yêu cầu sau: ***Tên thuộc tính, Kiểu dữ liệu, Độ dài dữ liệu, khuôn dạng của dữ liệu, các giá trị giới hạn cho phép, ý nghĩa, có duy nhất hay không, có cho phép giá trị rỗng hay không.***

4.2.4.2 Ràng buộc khoá

4.2.4.2.1 Khóa chính (Primary Key)

Khóa chính là một (hoặc một tập) các thuộc tính đóng vai trò là nguồn của một phụ thuộc hàm mà đích lần lượt là các thuộc tính còn lại.

Ví dụ: R={SSN, Name, BDate, Address, Salary}

SSN→ Name, BDate, Address, Salary

(Nguồn)→ (Đích)

Ta thấy, từ SSN ta có thể suy ra toàn bộ các thuộc tính ứng. Vậy SSN được gọi là khóa chính.

Một số gợi ý khi chọn khóa:

- Khóa không nên là tập hợp của quá nhiều thuộc tính. Trong trường hợp khóa có nhiều thuộc tính, có thể thêm một thuộc tính "nhân tạo" thay chúng làm khóa chính cho quan hệ.

- Nếu khóa chính được cấu thành từ một số thuộc tính, thì các thành phần nên tránh sử dụng thuộc tính có giá trị thay đổi theo thời gian: như tên địa danh, phân loại.

4.2.4.2.2 Khóa dự tuyển (Candidate Key)

Trong tập hợp các thuộc tính của một bảng, có thể có nhiều thuộc tính có thể dùng được làm khóa chính. Các thuộc tính đó được gọi là khóa dự tuyển.

Khóa dự tuyển cần thỏa mãn 2 tính chất sau:

- Xác định duy nhất.
- Không dư thừa: Khi xóa đi bất kỳ một thuộc tính nào của khóa đều phá hủy tính xác định duy nhất của khóa.

4.2.4.2.3 Khóa ngoại (Foreign Key)

Trong nhiều trường hợp, khóa chính của một bảng được đưa sang làm thuộc tính bên bảng khác, thuộc tính đó gọi là khóa ngoại. Khóa ngoại đóng vai trò thể hiện liên kết giữa 2 bảng.

4.2.4.2.4 Khóa phụ (Second Key)

Đóng vai trò khi ta muốn sắp xếp lại dữ liệu trong bảng.

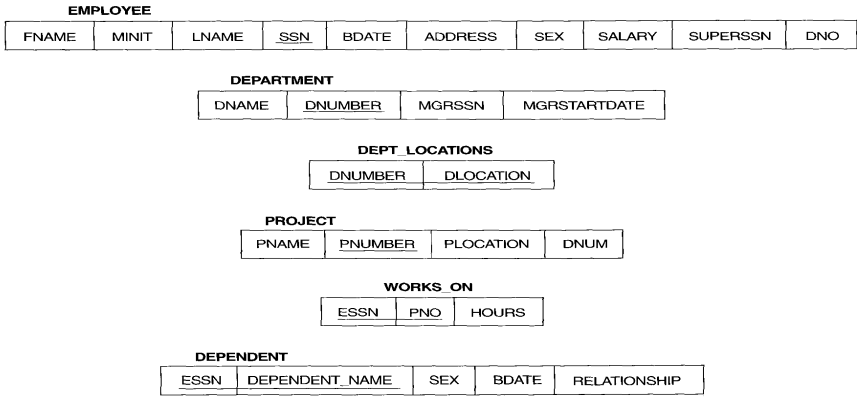
Ví dụ: Ta có bảng SINHVIEN (MaSV, Hoten, GioiTinh, Diem).

Muốn sắp xếp lại danh sách sinhviên theo thứ tự a, b, c.. của Họ tên. Khi đó thuộc tính Hoten được gọi là khóa phụ.

4.2.4.3 Ràng buộc thực thể

Mỗi một lược đồ quan hệ R, chúng ta phải xác định khoá chính của nó. Khoá chính trong lược đồ quan hệ được gạch chân ở phía dưới của thuộc tính.

Sau đây là danh sách các lược đồ quan hệ trong cơ sở dữ liệu COMPANY sau khi xác định ràng buộc thực thể:



Hình 4.1. Lược đồ cơ sở dữ liệu COMPANY

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
	1	Houston
	4	Stafford
	5	Bellaire
	5	Sugarland
	5	Houston

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

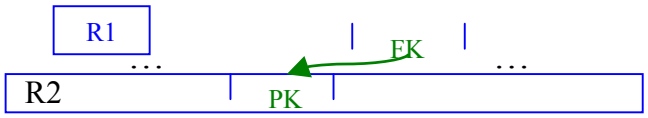
Hình 4.2. Một thể hiện của cơ sở dữ liệu COMPANY

Lưu ý: Ràng buộc khoá và ràng buộc thực thể được xác định cho từng quan hệ.

4.2.4.4 Ràng buộc toàn vẹn tham chiếu

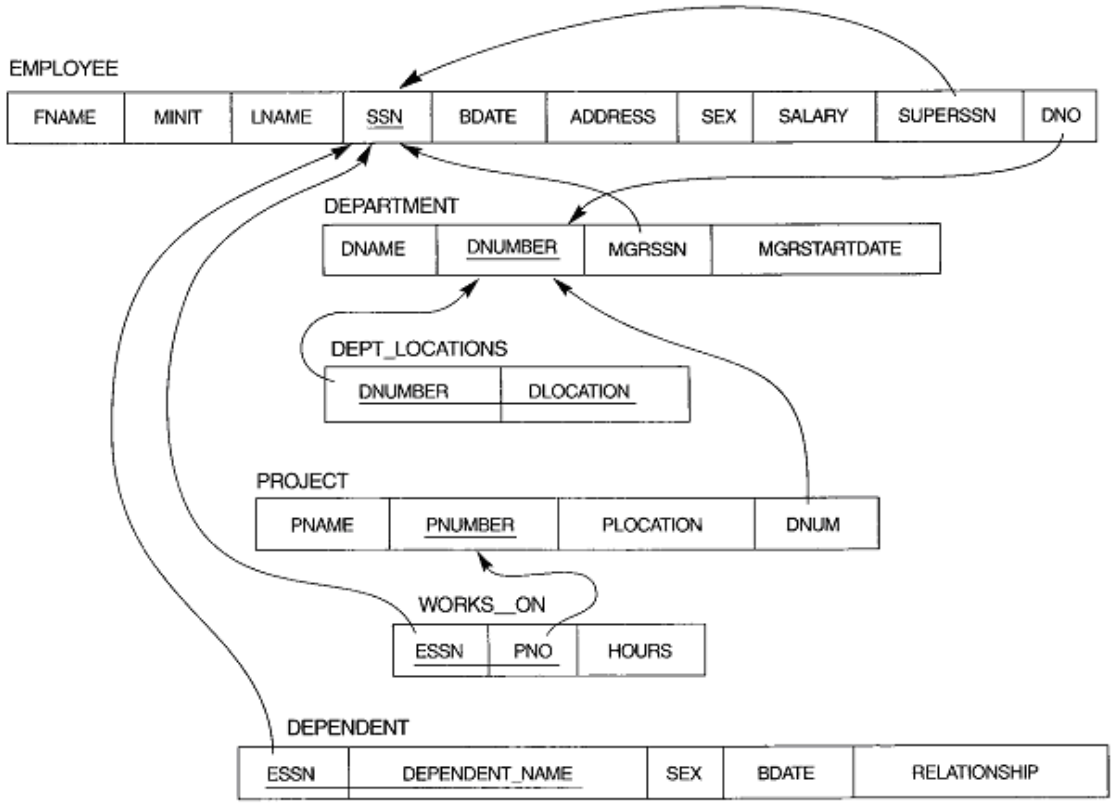
Một bộ giá trị trong một quan hệ tham chiếu tới một bộ giá trị đã tồn tại trong một quan hệ khác.

Ràng buộc toàn vẹn tham chiếu phải xác định trên 2 quan hệ: quan hệ tham chiếu (**referencing relation**) và quan hệ được tham chiếu (**referenced relation**).



Ràng buộc toàn vẹn tham chiếu còn được gọi là ràng buộc khoá ngoại.

Ví dụ: Thuộc tính DNo của quan hệ EMPLOYEE tham chiếu tới thuộc tính DNumber của quan hệ DEPARTMENT.



Hình 4.3. Các ràng buộc tham chiếu trong cơ sở dữ liệu COMPANY

4.2.5 Các phép toán trên CSDL quan hệ

4.2.5.1 Phép toán cập nhật

a. Phép chèn (INSERT): Là phép bổ xung thêm một bộ vào quan hệ r cho trước.

+ Biểu diễn: INSERT(r; A_1=d_1,A_2=d_2,...,An=dn) với Ai là thuộc tính, di thuộc dom(Ai), i=1,..,n.

Nếu thứ tự các trường là cố định, có thể biểu diễn phép chèn dưới dạng không tường minh INSERT(r; d_1,d_2,..., dn).

+ Ví dụ : Chèn thêm một bộ t_4=('004', 'Hoàng Thanh Vân',1969, 'Hà nội', 235) vào quan hệ EMPLOYEE(SSN, Name, BDate, Address, Salary) ta có thể viết:

INSERT(EMPLOYEE; SSN= '004', Name= 'Hoàng Thanh Vân', BDate=1969, Address= 'Hà nội', Salary=235).

+ Chú ý : Kết quả của phép chèn có thể gây ra một số sai sót là :

- Bộ mới được thêm không phù hợp với lược đồ quan hệ cho trước

- Một số giá trị của một số thuộc tính nằm ngoài miền giá trị của thuộc tính đó.

- Giá trị khoá của bộ mới có thể là giá trị đã có trong quan hệ đang lưu trữ.

b. Phép loại bỏ (DEL): Là phép xoá một bộ ra khỏi một quan hệ cho trước.

- Biểu diễn : DEL(r; A_1=d_1,A_2=d_2,...,An=dn) hay DEL((r, d_1,d_2,..., dn).

Nếu K=(E_1,E_2,...,Em) là khoá thì có thể viết DEL(r; E_1=e_1,E_2=e_2,...,Em=em)

- Ví dụ :

+ Để xoá bộ t1 ra khỏi quan hệ r:

DEL(EMPLOYEE; SSN= '004', Name= 'Hoàng Thanh Vân', BDate=1969, Address= 'Hà nội', Salary=235).

+ Cần loại bỏ một nhân viên trong quan hệ EMPLOYEE mà biết SSN đó là '004' thì chỉ cần viết: DEL(EMPLOYEE; SSN= '004')

c. Phép cập nhật (UPDATE): Là phép tính dùng để sửa đổi một số giá trị nào đó tại một số thuộc tính.

+ Biểu diễn :

UPD (r; A_1=d_1,A_2=d_2,...,An=dn; B_1=b_1,B_2=b_2,...,Bk=bk)

Với {B_1,B_2,...,Bk} là tập các thuộc tính mà tại đó các giá trị của bộ cần thay đổi. {B_1,B_2,...,Bk} ứng với tập thuộc tính {A_1,A_2,...,An}

Hay UPD(r; E_1=e_1,E_2=e_2,...,Em=e; B_1=b_1,B_2=b_2,...,Bk=bk) với K=(E_1,E_2,...,Em) là khoá.

+ Ví dụ : Để thay đổi tên nhân viên có SSN= '003' trong quan hệ EMPLOYEE thành Nguyễn Thanh Mai ta có thể viết :

CH (EMPLOYEE; SSN= '03'; Name= 'Nguyễn Thanh Mai')

4.2.5.2 Phép toán đại số quan hệ

Đại số quan hệ gồm một tập các phép toán tác động trên các quan hệ và cho kết quả là một quan hệ.

Có 8 phép toán được chia làm 2 nhóm : Nhóm các phép toán tập hợp (hợp, giao, trừ, tích đề các), nhóm các phép toán quan hệ (chọn, chiếu, kết nối, chia).

Định nghĩa : Hai quan hệ r và s được gọi là khả hợp nếu chúng được xác định trên cùng một tập các miền giá trị (Có nghĩa là chúng được xác đinh trên cùng một tập các thuộc tinh).

a. Phép hợp:

- Phép hợp của hai quan hệ khả hợp r U s = {t / t thuộc r hoặc t thuộc s}

Ví dụ**:**

r (A, B , C)	s(A, B , C)	r U s (A, B , C)
a1 b1 c1	a1 b1 c1	a1 b1 c1
a2 b2 c2	a1 b2 c2	a2 b2 c2
		a1 b2 c2

- Phép hợp của hai quan hệ là phép gộp các bộ của hai bảng của một quan hệ thành một bảng và bỏ đi các bộ trùng.

Ví dụ**:** EMPLOYEE1

SSN	Name	DNo
001	Hoàng	P001
002	Thiện	P002

EMPLOYEE2

SSN	Name	DNo
003	Huy	P001
002	Thiện	P002
004	Thiện	P003

EMPLOYEE1 ∪ EMPLOYEE2 = EMPLOYEE3

SSN	Name	DNo
001	Hoàng	P001
002	Thiện	P002
003	Huy	P001
004	Thiện	P003

Hình 4.4. Minh họa dữ liệu phép hợp 2 quan hệ

b. Phép giao

- Phép giao của hai quan hệ khả hợp r ∩ s ={t / t thuộc r và t thuộc s}

Ví dụ **:**

r ∩ s (A, B , C)
a1 b1 c1

- Phép giao của hai quan hệ là lấy ra các bộ cùng có mặt ở cả hai bảng của một quan hệ.

Ví dụ: EMPLOYEE1 ∩ EMPLOYEE2 = 002, Thiện, P002

c. Phép trừ

- Phép trừ của hai quan hệ khả hợp r - s = {t / t thuộc r và t không thuộc s}

Ví dụ :

r - s (A, B , C)
a2 b2 c2

- Phép trừ của hai quan hệ A và B là lấy các bộ có trong bảng A mà không có trong bảng B.

Ví dụ: EMPLOYEE1 - EMPLOYEE2 = 001, Hoàng, P001

EMPLOYEE2 - EMPLOYEE1

SSN	Name	DNo
003	Huy	P001
004	Thiện	P003

d. Phép tích đề các :

- Cho quan hệ r(R), R={A_1,A_2,...,An} và quan hệ s(U), U={B_1,B_2,...,Bm}

- Tích đề các :

r x s ={t=(a_1,a_2,...,an, b_1,b_2,...,bm) /a_1,a_2,...,an ∈r và b_1,b_2,...,bm ∈s}

Ví dụ **:**

r (A , B) ; s(C , D) ;

a1 1	1 d1
a2 2	2 d2
a3 3	

r x s = k (A, B, C, D)

a1	1	1	d1
a1	1	2	d2
a2	2	1	d1
a2	2	2	d2
a3	3	1	d1
a3	3	2	d2

- Chú ý: Bậc k = bậc r + bậc s , lực lượng k = lực lượng r x lực lượng s

Phép tích đề các là phép toán đắt nhất trong các phép toán của đại số quan hệ.

e. Phép chọn (cắt ngang) - một ngôi

- Là phép toán lọc ra một tập con các bộ của quan hệ đã cho theo biểu thức chọn F.

- Biểu thức chọn F là một tổ hợp logic các toán hạng, mỗi toán hạng là một phép so sánh đơn giản giữa hai thuộc tính hoặc giữa một thuộc tính và một giá trị hằng.

- Phép toán logic: AND (và), OR (hoặc), NOT (phủ định).

- Phép toán so sánh : <, >, =, >=, <=, <>

- Phép chọn trên quan hệ r với biểu thức chọn F

$$\sigma_F(r) = \{ \text{ t thuộc r / F(t) đúng} \}$$

r (A , B) ; F1 = (A=a1) OR (B=3) → $\sigma_{F1}(r)$ = r' (A ,B)

a1	1	a1	1
a2	2	a3	3
a3	3		

Bậc r = bậc r'; lực lượng của r >= lực lượng của r'

-Phép chọn trên quan hệ là lấy ra các dòng của bảng quan hệ thoả mãn một điều kiện nào đó trên tập các cột thuộc tính.

Ví dụ : Chọn trên quan hệ EMPLOYEE3 các nhân viên thuộc phòng có DNo=P001

SSN	Name	DNo
001	Hoàng	P001
003	Huy	P001

f. Phép chiếu (cắt dọc) - 1 ngôi

- Là phép toán loại bỏ đi một số thuộc tính và chỉ giữ lại một số thuộc tính được chỉ ra của một quan hệ.

- Cho quan hệ r(R), X là tập con của tập thuộc tính R. Phép chiếu của quan hệ r trên X : $\Pi_X(r)$ = { t[X]/ thuộc r}; t[X] là bộ t lấy trên tập thuộc tính X.

Ví dụ : Cho r(A,B) như trên , X={A};

$\Pi_X(r)$ = u(A)
a1
a2
a3

- Bậc của r > bậc của k. Lực lượng của r > lực lượng của k

- Phép chiếu trên quan hệ là lấy một số cột (thuộc tính) nào đó của bảng quan hệ.

Ví dụ : Lấy danh sách mã NV của quan hệ NHANVIEN

Π_{SSN} (EMPLOYEE3) =

SSN
001
002
003
004

g. Phép kết nối - 2 ngôi

1. Phép kết nối

- Cho hai quan hệ r(R), R={A_1,A_2,...,An} và quan hệ s(U), U={B_1,B_2,...,Bm}.

- Phép xếp cạnh nhau: cho hai bộ d = (d_1,d_2,...,dn) và e = (e_1,e_2,...,em) phép xếp cạnh nhau của d và e là : (d^e) = (d_1,d_2,...,dn, e_1,e_2,...,em)

- Phép kết nối giữa quan hệ r có thuộc tính A và quan hệ s có thuộc tính B với một phép so sánh θ là :

r >< s = {a^b / a thuộc r, b thuộc s và a(A) θ b(B)}

Ví dụ : Xét quan hệ r và s trong ví dụ phép tích đề các

r (A , B)	;	s(C , D)	;	r x s = k (A, B, C, D)
a1 1		1 d1		(B>=C) a1 1 1 d1
a2 2		2 d2		a1 1 2 d2
a3 3				a2 2 2 d2
				a3 3 1 d1
				a3 3 2 d2

- Lực lượng của phép kết nối k' <= lực lượng của phép tích đề các k.

$$k' = \sigma_{B>=C}(k) \rightarrow r \overset{F}{><} s = \sigma_F(r \times s)$$

- Chú ý **:**

+ Để phép kết nối có nghĩa, miền trị dom(A) phải so sánh được qua phép so sánh θ với miền trị dom(B)

+ Nếu phép so sánh θ là "=" thì phép kết nối gọi là kết nối bằng.

2. Phép kết nối tự nhiên

Phép toán kết nối bằng trên những thuộc tính cùng tên của hai quan hệ và sau khi kết nối thì cắt bỏ đi một thuộc tính cùng tên bằng phép chiếu của đại số quan hệ được gọi là phép kết nối tự nhiên ký hiệu *.

-Ví du :

r(A, B, C)	s(C, D, E)	r*s (A, B, C, D, E)
a1 1 1	1 d1 e1	a1 1 1 d1 e1
a2 2 1	2 d2 e2	a2 2 1 d1 e1
a1 2 2	3 d3 e3	a1 2 2 d2 e2

$$r*s = \Pi_{ABCDE}(r \overset{C=C}{><} s)$$

- **Ví dụ** : Cho hai quan hệ NHANVIEN và PHONG

SSN	Name	DNo
001	Hoàng	P001
002	Thiện	P002
003	Huy	P001
004	Thiện	P003

DNo	DName
P001	Tổ chức
P002	Kinh doanh
P003	Nhân sự
P004	Tiếp thị

Kết nối tự nhiên của hai quan hệ :

SSN	Name	DNo	DName
001	Hoàng	P001	Tổ chức
002	Thiện	P002	Kinh doanh
003	Huy	P001	Tổ chức
004	Thiện	P003	Nhân sự

Hình 4.5. Minh họa dữ liệu phép kết nối tự nhiên 2 quan hệ

h. Phép chia

- Cho r là một quan hệ n- ngôi, s là quan hệ m- ngôi (n>m, s khác rỗng). Phép chia quan hệ r cho quan hệ s là tập tất cả các n-m bộ t sao cho với mọi bộ u thuộc s thì bộ (t^u) thuộc r : r ÷ s = {t / với mọi u thuộc s thì (t^u) thuộc r}

Ví dụ :

r(A, B, C, D)

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e

s(C, D)

c	d
e	f

r ÷ s (A, B)

a	b
e	d

- Ví dụ với hai quan hệ : PRODUCT và SUPPORT

PNo	PName
h1	Đài
h2	TV
h3	tủ lạnh

SNo	PNo	PName
n1	h1	Đài
n1	h2	TV
n1	h3	Tủ lạnh
n2	h3	Tủ lạnh
n2	h1	Đài
n3	h1	Đài
n3	h2	TV
n3	h3	Tủ lạnh
n4	h1	Đài

Cung cấp Mặt hàng =

SNo
n1
n3

Hình 4.6. Minh họa dữ liệu phép chia 2 quan hệ

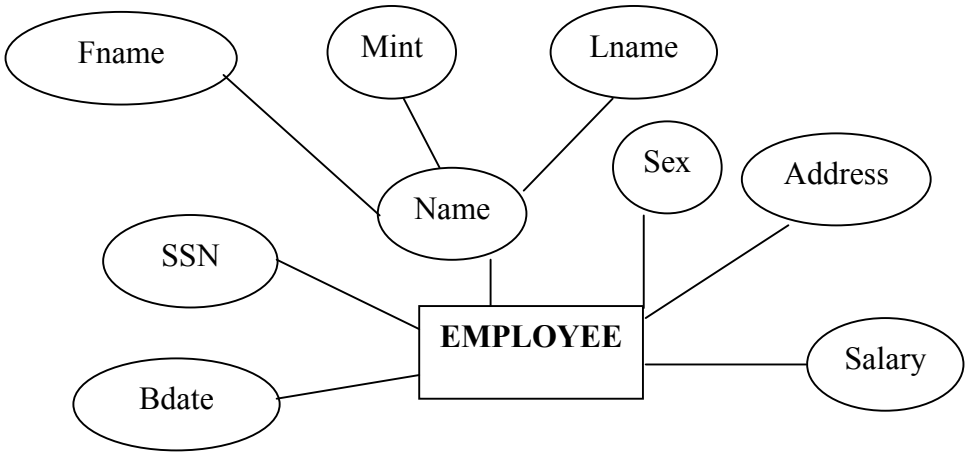
5 Chương 5. CHUYỂN TỪ MÔ HÌNH ER SANG MÔ HÌNH QUAN HỆ

Như chúng ta đã biết, mô hình ER là mô hình dữ liệu mức khái niệm. Sau quá trình khảo sát thiết kế, ta thu được mô hình này. Từ mô hình này ta có thể sử dụng các quy tắc chuyển sang mô quan hệ để thực hiện quản lý cơ sở dữ liệu trên máy tính.

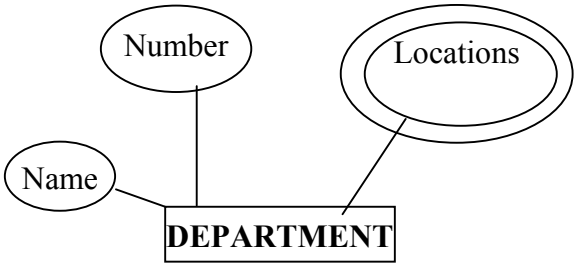
Sau đây là 8 bước được sử dụng để chuyển từ mô hình ER sang mô hình quan hệ:

Bước 1: Mỗi kiểu thực thể bình thường (không phải kiểu thực thể yếu) trong mô hình ER trở thành một quan hệ. Quan hệ đó bao gồm tất cả các thuộc tính đơn giản và thuộc tính tổ hợp của thực thể. Thuộc tính định danh của thực thể là khóa chính của quan hệ.

Ví dụ: Kiểu thực thể: EMPLOYEE, DEPARTMENT, PROJECT
==> quan hệ: EMPLOYEE, DEPARTMENT, PROJECT

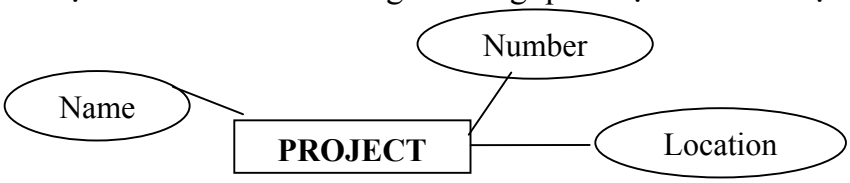


→ **Quan hệ: EMPLOYEE (Ssn, fname, minit, lname, bdate, sex, address, salary)**



→ **Quan hệ: DEPARTMENT (Dnumber, Dname)**

Lưu ý: Thuộc tính Locations không có trong quan hệ vì nó là thuộc tính đa trị.

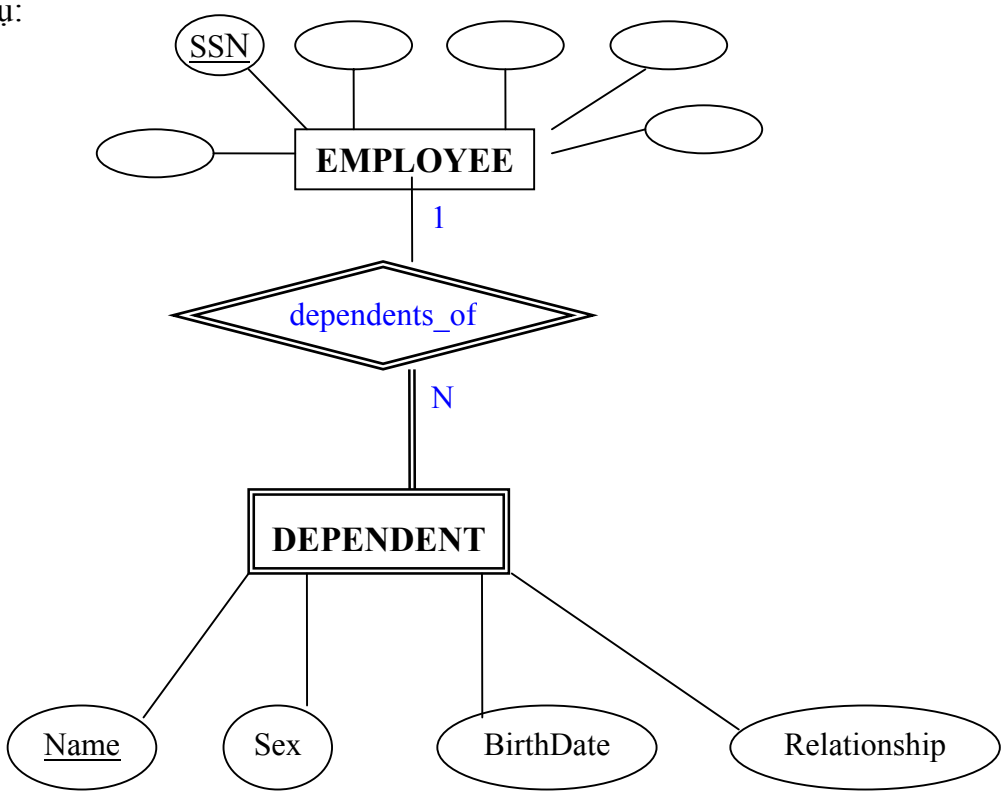


→ **Quan hệ: PROJECT (Pnumber, Pname, Plocation)**

Bước 2: Cho mỗi thực thể yếu (Weak Entity) trong mô hình ER, tạo thành một quan hệ R, tất cả thuộc tính đơn giản của thực thể yếu trở thành thuộc tính của R. Thêm vào đó, thuộc tính định danh của thực thể chủ trở thành khóa ngoại của R.

Khoá chính của R là sự kết hợp giữa thuộc tính định danh của thực thể chủ và thuộc tính định danh của thực thể yếu.

Ví dụ:

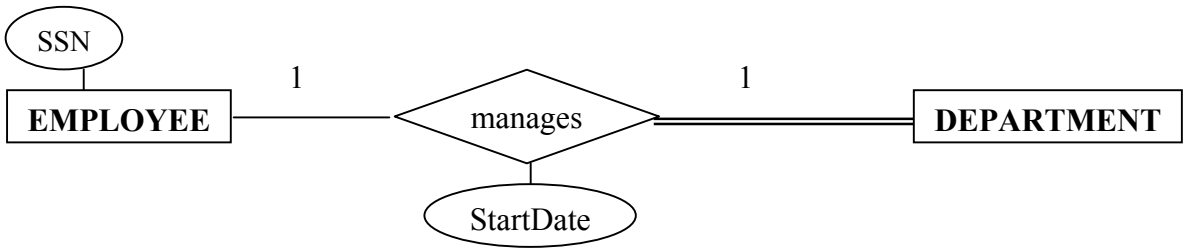


→ DEPENDENT (Essn, Dependent_name, sex, bdate, relationship)

Bước 3: Cho mỗi mối liên kết 1-1 trong mô hình ER:

- Xác định một quan hệ S_T. Kiểu thực thể có sự tham gia toàn bộ vào liên kết trở thành quan hệ S, thực thể còn lại trở thành quan hệ T.
- Đưa khóa chính của T sang làm khóa ngoại của S.
- Thuộc tính của mối quan hệ S_T trở thành thuộc tính của S.

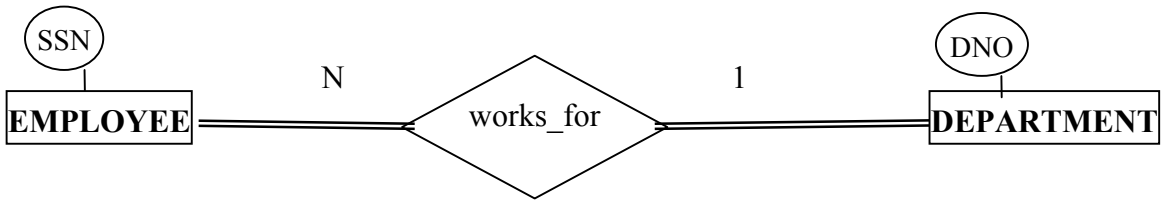
Ví dụ:



→ DEPARTMENT(…, MGRSSN, MGRSTARTDATE)

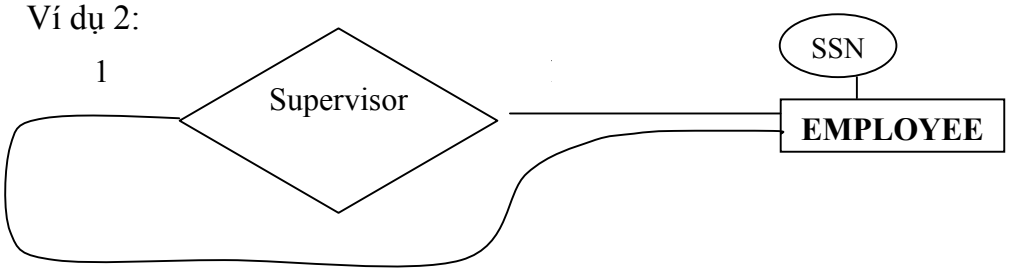
Bước 4: Cho mỗi mối liên kết 1_N trong mô hình ER. Chuyển khóa chính của quan hệ phía 1 sang làm khóa ngoại của quan hệ phía N.

Ví dụ 1:



→ EMPLOYEE(…, DNO)

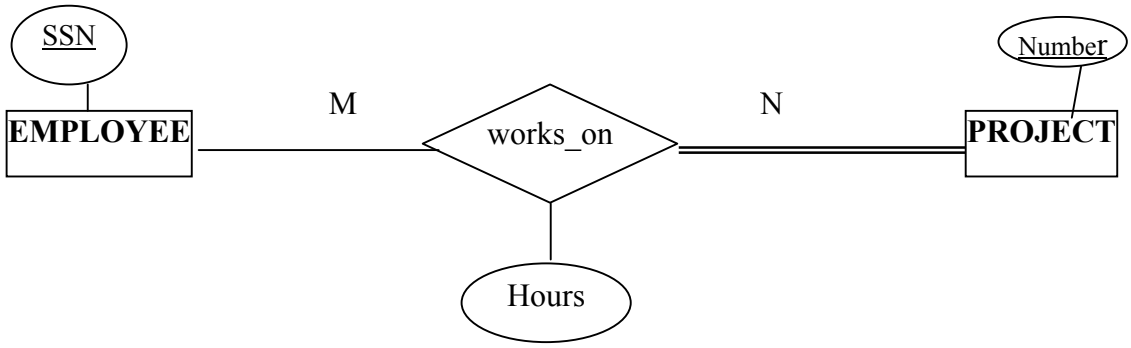
Ví dụ 2:



→ EMPLOYEE(…, SuperSSN)

Bước 5:Cho mỗi mối liên kết MN, sinh ra một quan hệ mới R, chuyển khóa chính của hai quan hệ phía M và N thành khóa ngoại của quan hệ R. Khóa chính của R là sự kết hợp của hai khóa ngoại.

Ví dụ:



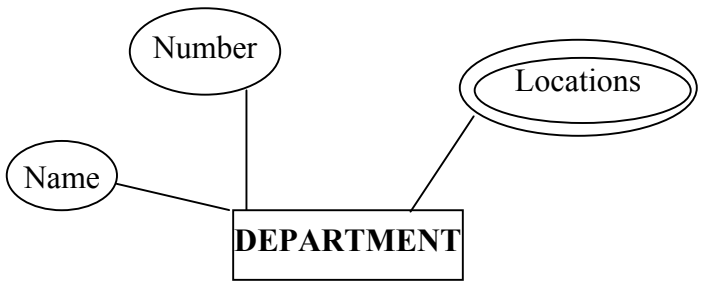
→ WORKS_ON(PNO, ESSN, Hours)

Bước 6: Nếu gặp thuộc tính đa trị:

- Chuyển thuộc tính đa trị thành quan hệ mới.
- Thuộc tính định danh (hoặc 1 phần thuộc tính định danh) của thực thể chính chuyển thành khóa ngoại của quan hệ mới.

- Khóa chính của quan hệ mới là khóa chính của bản thân quan hệ + khóa ngoại do thực thể chính chuyển sang.

Ví dụ:

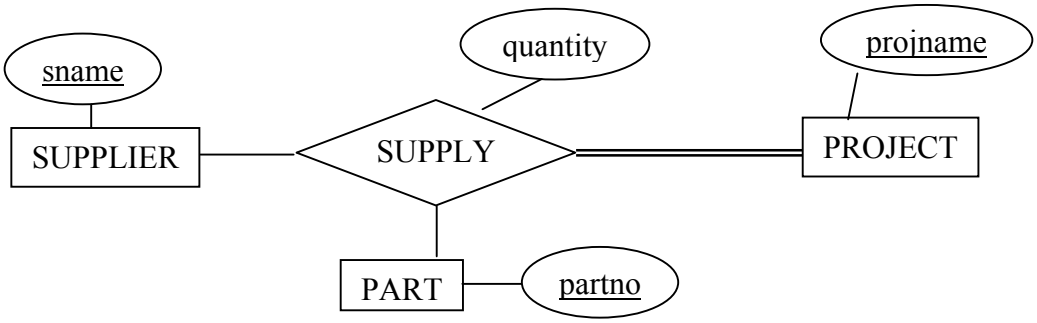


→ **DEPT_LOCATIONS (DNumber, DLocation)**

Bước 7:

Cho mỗi mối liên kết có bậc (>2), tạo ra quan hệ mới (R), khóa chính của các quan hệ tham gia liên kết được đưa làm khóa ngoại của quan hệ R và các khóa ngoại này đồng thời đóng vai trò là khóa chính của R.

Ví dụ:



→ **SUPPLY (SName, ProjName, PartNo, Quantity)**

Bước 8: Xử lý quan hệ giữa lớp cha/ lớp con và chuyên biệt hoá hoặc tổng quát hoá.

Các lựa chọn khác nhau cho việc chuyển đổi một số lượng các lớp con từ cùng một chuyên biệt (hoặc tổng quát hoá thành lớp cha). Ngoài 7 bước đã trình bày ở trên trong bước 8 dưới đây đưa ra một lựa chọn phổ biến nhất và các điều kiện mà mỗi lựa chọn có thể sử dụng. Sử dụng ký hiệu Attr(R) để biểu thị các thuộc tính của R và PK(R) là khoá chính của R.

Cách thực hiện:

Chuyển đổi mỗi chuyên biệt hoá có:

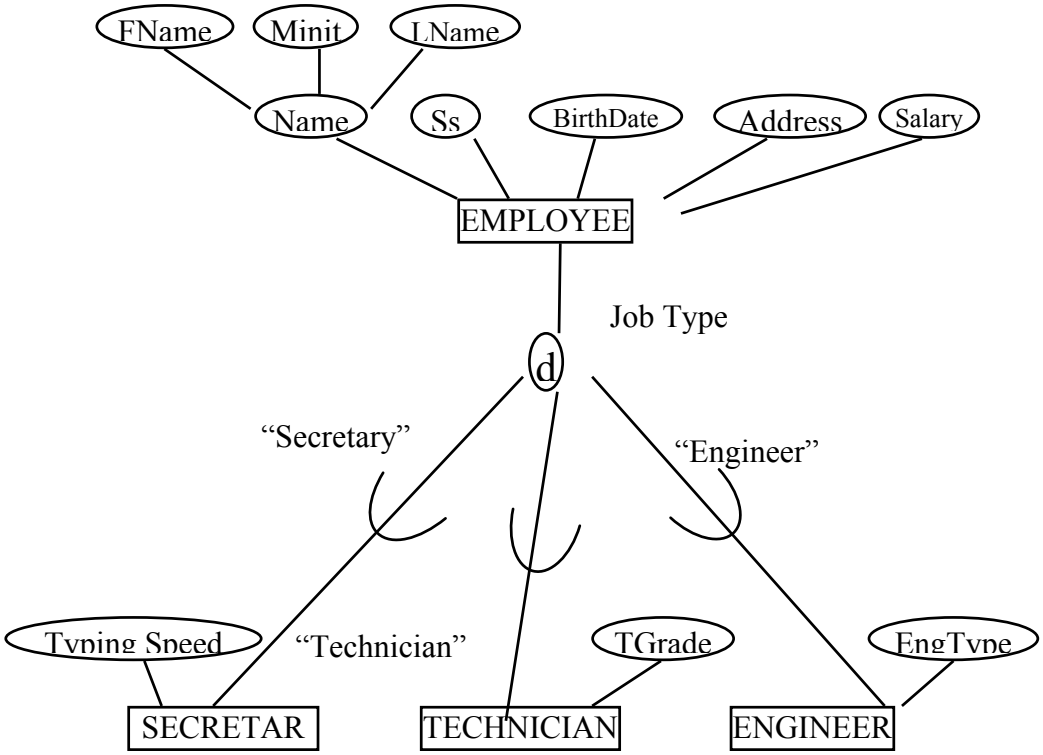
- m lớp con { S_1, S_2 , … , Sm} và lớp cha C, thuộc tính của C là { k, a_1 , a_2 , …, an} và k là khoá chính thành những lược đồ quan hệ, chúng ta có thể sử dụng một trong 4 lựa chọn sau:

1. Lựa chọn 8A:

- Tạo quan hệ L cho lớp cha C với các thuộc tính ***Attrs(L)={k, a1, …, an}*** và khoá chính của L là: ***PK(L)=k.***

- Tạo quan hệ Li cho mỗi lớp con tương ứng Si với các thuộc tính ***Attrs(Li)={k} U {thuộc tính của Si}*** và ***PK(Li)=k***.

Ví dụ:



→ Chuyển chuyên biệt hoá trên thành các quan hệ sau:

EMPLOYEE(SSN, FName, Minit, LName, BirthDate, Address, Salary)

SECRETARY(SSN, TypingSpeed)

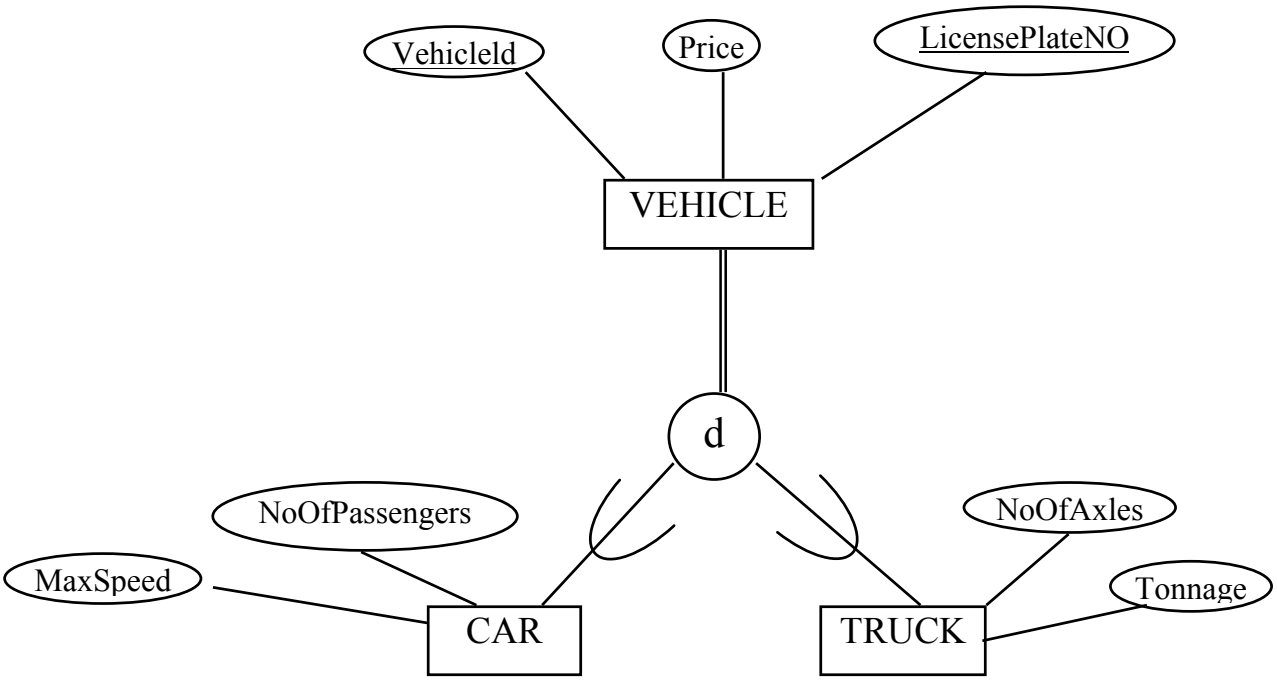
TECHNICIAN(SSN, TGrade)

ENGINEER(SSN, EngType)

2. Lựa chọn 8B:

Tạo một quan hệ Li cho mỗi lớp con Si , với các thuộc tính ***Attr(Li) = {k, a_1, a_2,…, am} U {thuộc tính của Si}*** và ***PK(Li) = k***.

Ví dụ:



→ Chuyển chuyên biệt hoá trên thành các quan hệ sau:

CAR(Vehicleld, LicensePlateNo, Price, MaxSpeed, NoOfPassengers)

TRUCK(Vehicleld, LicensePlateNo, Price, NoOfAxles, Tonnage)

3. *Lựa chọn 8C:*

Tạo một quan hệ L với các thuộc tính ***Attr(L) = {k, a_1 , a_2 , …, an } U {thuộc tính của S_1} U… U {thuộc tính của Sm} U {t}*** và ***PK(L) = k***. Trong đó, t là thuộc tính phân biệt chỉ ra bản ghi thuộc về lớp con nào, vì thế miền giá trị của t ={1,2,…,m}.

Ví dụ: Đối với chuyên biệt hoá của EMPLOYEE, ta chỉ tạo ra một quan hệ L như sau:

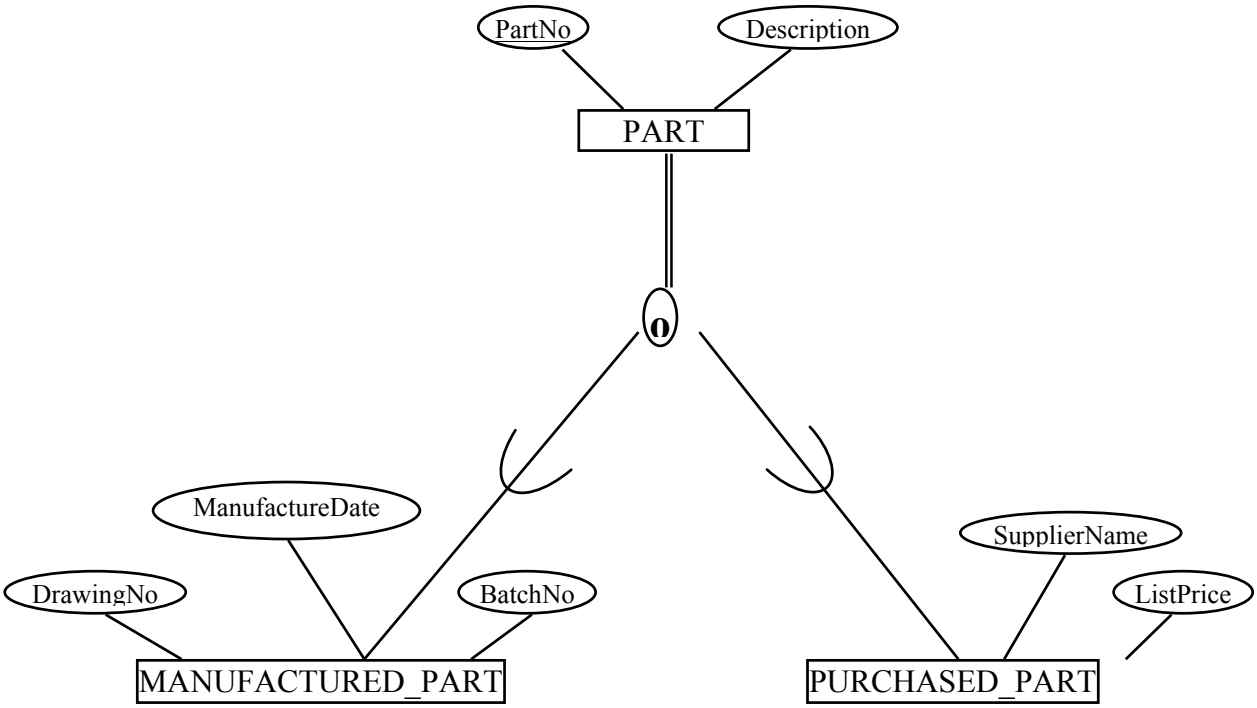
EMPLOYEE(SSN, FName, Minit, LName, BirthDate, Address, Salary, TypingSpeed, Tgrad, EngType, JobType)

Thuộc tính phân biệt

4. *Lựa chọn 8D:*

Tạo một quan hệ L với các thuộc tính ***Attr(L) = {k, a_1 , a_2 , …, an } U {thuộc tính của S_1} U… U {thuộc tính của Sm} U {t_1 , t_2 , …, tmj} và PK(L) = k***. Lựa chọn này cho chuyên biệt hoá của các lớp con được nạp chồng (nhưng cũng áp dụng cho một chuyên biệt tách rời), và với mỗi ti, 1≤ i ≤ m, là thuộc tính BOOLEAN chỉ ra bộ theo lớp con Si.

Ví dụ:



→ PART (<u>ParNo</u>, Description, Mflag, DrawingNo, ManufactureDate, BatchNo, PFlag, SupplierName, ListPrice)

Thuộc tính phân biệt

6 Chương 6. PHỤ THUỘC HÀM VÀ CHUẨN HOÁ QUAN HỆ

6.1 Một số hướng dẫn khi thiết kế cơ sở dữ liệu quan hệ

Việc quan trọng nhất khi thiết kế cơ sở dữ liệu quan hệ là ta phải chọn ra tập các lược đồ quan hệ tốt nhất dựa trên một số tiêu chí nào đó. Và để có được lựa chọn tốt, thì chúng ta cần đặc biệt quan tâm đến mối ràng buộc giữa các dữ liệu trong quan hệ, đó chính là các phụ thuộc hàm.

Để hiểu hơn về câu hỏi tại sao phải thiết kế một cơ sở dữ liệu tốt, chúng ta hãy cùng tìm hiểu ví dụ sau:

RESULT(StNo, StName, SubNo,SubName, Credit, Mark)

Quan hệ RESULT(Kết quả học tập) có các thuộc tính: StNo(Mã sinh viên), StName(Tên sinh viên), SubNo(Mã môn học), SubName(Tên môn học), Credit (Số đơn vị học trình) và Mark (điểm thi của sinh viên trong môn học).

Sau đây là minh hoạ dữ liệu của quan hệ RESULT:

StNo	StName	SubNo	SubName	Credit	Mark
St01	Mai	Sub04	CSDL	3	9
St01	Mai	Sub01	TRR	5	10
St01	Mai	Sub02	PPS	4	8
St02	Vân	Sub04	CSDL	3	10
St02	Vân	Sub01	TRR	5	9
St03	Thanh	Sub07	Tiếng Anh	4	8

Hình 6.1. Minh họa dữ liệu của quan hệ RESULT

Quan hệ trên thiết kế chưa tốt vì:

1. **Dư thừa dữ liệu (Redundancy):** Thông tin về sinh viên và môn học bị lặp lại nhiều lần. Nếu sinh viên có mã St01 thi 10 môn học thì thông tin về sinh viên này bị lặp lại 10 lần, tương tự đối với môn học có mã Sub04, nếu có 1000 sinh viên thi thì thông tin về môn học cũng lặp lại 1000 lần
2. **Không nhất quán (Inconsistency):** Là hệ quả của dư thừa dữ liệu. Giả sử sửa bản ghi thứ nhất, tên sinh viên được chữa thành Nga thì dữ liệu này lại không nhất quán với bản ghi thứ 2 và 3 (vẫn có tên là Mai).
3. **Dị thường khi thêm bộ (Insertion anomalies):** Nếu muốn thêm thông tin một sinh viên mới nhập trường (chưa có điểm môn học nào) vào quan hệ thì không được vì khoá chính của quan hệ trên gồm 2 thuộc tính StNo và SubNo.

4. **Dị thường khi xoá bộ (Deletion anomalies):** Giả sử xoá đi bản ghi cuối cùng, thì thông tin về môn học có mã môn học là SubNo=Sub07 cũng mất.

Nhận xét: Qua phân tích trên, ta thấy chúng ta nên tìm cách tách quan hệ trên thành các quan hệ nhỏ hơn.

Trong chương này chúng ta sẽ nghiên cứu về những khái niệm và các thuật toán để có thể thiết kế được những lược đồ quan hệ tốt.

6.2 Phụ thuộc hàm(Functional Dependencies)

- Phụ thuộc hàm (FDs) được sử dụng làm thước đo để đánh giá một quan hệ tốt.
- FDs và khoá được sử dụng để định nghĩa các dạng chuẩn của quan hệ.
- FDs là những ràng buộc dữ liệu được suy ra từ ý nghĩa và các mối liên quan giữa các thuộc tính.

6.2.1 Định nghĩa phụ thuộc hàm

Cho r(U), với r là quan hệ và U là tập thuộc tính.

*Cho A,B ⊆U, phụ thuộc hàm X → **Y** (đọc là X xác định Y) được định nghĩa là:*

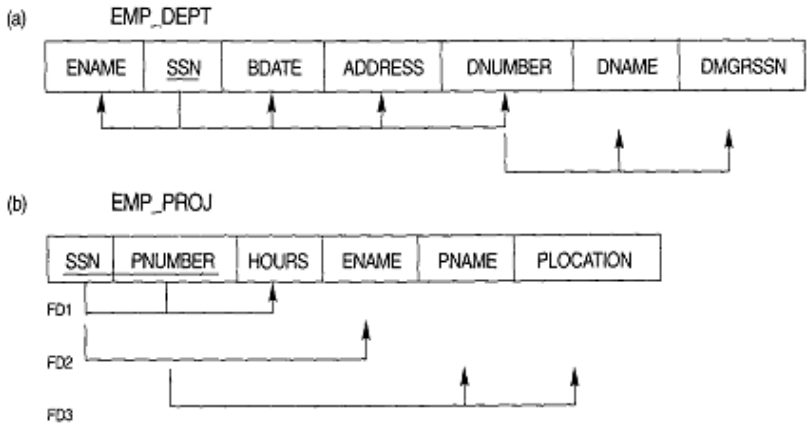
∀ t, t' ∈ r nếu t.X = t'.X thì t.Y = t'.Y

***(Có nghĩa là:** Nếu hai bộ có cùng trị X thì có cùng trị Y)*

Phụ thuộc hàm được suy ra từ những quy tắc dữ liệu khi ta khảo sát yêu cầu của bài toán.

Ví dụ:

Từ mã số bảo hiểm xã hội, ta có thể suy ra được tên của nhân viên (Ssn→ Ename).Từ mã dự án, ta có thể suy ra tên và vị trí của dự án (PNumber→{PName, PLcation})



Hình 6.2. Biểu diễn FDs của 2 lược đồ quan hệ EMP_DEPT và EMP_PROJ

6.2.2 Hệ tiên đề Armstrong

Cho lược đồ quan hệ r(U), U là tập thuộc tính, F là tập các phụ thuộc hàm được định nghĩa trên quan hệ r.

Ta có phụ thuộc hàm A → B **được suy diễn logic từ F** nếu quan hệ r trênU thỏa các phụ thuộc hàm trong F thì cũng thỏa phụ thuộc hàm A → B.

Ví dụ:

Tập phụ thuộc hàm: F = { A → B, B → C}

Ta có phụ thuộc hàm A → C là phụ thuộc hàm được suy từ F.

Hệ tiên đề Armstrong được sử dụng để tìm ra các phụ thuộc hàm suy diễn từ F.

Hệ tiên đề Armstrong bao gồm:

1. **Phản xạ:** Nếu Y ⊂X thì X → Y
2. **Tăng trưởng:** Nếu Z ⊂U và X → Y thì XZ → YZ (Ký hiệuXZ là X∪Z)
3. **Bắc cầu:** Nếu X → Y và Y → Z thì X → Z
4. **Giả bắc cầu:** Nếu X → Y và WY → Z thì XW → Z
5. **Luật hợp:** Nếu X → Y và X → Z thì X →YZ
6. **Luật phân rã:** Nếu X → Y và Z ⊂Y thì X → Z

Trong sáu luật trên thì a4, a5, a6 suy được từ a1, a2, a3.

6.2.3 Bao đóng của tập phụ thuộc hàm

- Ta gọi f là một phụ thuộc hàm được suy dẫn từ F, ký hiệu là F ⊢ f nếu tồn tại một chuỗi phụ thuộc hàm: f1, f2,…., fn sao cho fn=f và mỗi fi là một thành viên của F hay được suy dẫn từ những phụ thuộc hàm j=1,…,i-1 trước đó nhờ vào luật dẫn.
- **Bao đóng của F:** ký hiệu là F+ là tập tất cả các phụ thuộc hàm được suy từ F nhờ vào hệ tiên đề Armstrong. F+ được định nghĩa:

$$F^+ = \{ X \rightarrow Y \mid F \vdash X \rightarrow Y \}$$

6.2.4 Bao đóng của tập thuộc tính X trên F

Bao đóng của tập thuộc tính X xác định trên tập phụ thuộc hàm F ký hiệu là X^+ là tập hợp tất cả các thuộc tính có thể suy ra từ X. Ký hiệu:

$$X^+ = \{ Y \mid F \vdash X \rightarrow Y \}$$

X^+ có thể được tính toán thông qua việc lặp đi lặp lại cá quy tắc 1, 2, 3 của hệ tiên đề Armstrong.

Thuật toán xác định bao đóng của tập thuộc tính X+:

X^+ := X;
repeat

oldX$^+$:= X$^+$;
for (mỗi phụ thuộc hàm Y →Z trong F) do
if Y ⊆ X$^+$ then X$^+$ ∪ Z
until (oldX$^+$ = X$^+$);

Ví dụ: Cho tập phụ thuộc hàm:

F = { SSN→ENAME, PNUMBER→{PNAME, PLOCATION}, {SSN, PNUMBER} → HOURS}

Suy ra:

{SSN}$^+$ = {SSN, ENAME}

{PNUMBER}$^+$ = {PNUMBER, PNAME, PLOCATION}

{SSN, PNUMBER}$^+$ = {SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS}

Như vậy, tập thuộc tính {SSN, PNUMBER} là khoá của quan hệ.

6.2.5 Khoá của quan hệ

Cho quan hệ r(R), tập K⊂R được gọi là khóa của quan hệ r nếu: K+=R và nếu bớt một phần tử khỏi K thì bao đóng của nó sẽ khác R.

Như thế tập K⊂R là khoá của quan hệ nếu K+=R và (K \A)+ ≠R , ∀A⊂R.

Ví dụ: ChoR = { A, B, C, D, E, G } và tập phụ thuộc hàm:

F= { AB → C , D → EG , BE → C , BC → D , CG → BD, ACD → B, CE → AG}

Ta sẽ thấy các tập thuộc tính:

K1 = { A, B } , K2 = {B,E} , K3={C,G} , K4={C,E} , K5 = {C,D}, K6={B,C} đều là khóa của quan hệ.

Như vậy, một quan hệ có thể có nhiều khóa.

Thuật toán tìm khoá:

Ý tưởng: Bắt đầu từ tập U vì Closure(U+,F) = U. Sau đó ta bớt dần các phần tử của U để nhận được tập bé nhất mà bao đóng của nó vẫn bằng U.

Thuật toán:

Input: Lược đồ quan hệ r(U), tập phụ thuộc hàm F.

Output: Khoá K

Bước 1: Gán K = U

Bước 2: Lặp lại các bước sau:

Loại phần tử A khỏi K mà **Closure(K -A,F)** =U

Nhận xét:

- Thuật toán trên chỉ tìm được một khóa. Nếu cần tìm nhiều khóa, ta thay đổi trật tự loại bỏ các phần tử của K.
- Chúng ta có thể cải thiện tốc độ thực hiện thuật toán trên bằng cách: Trong bước 1 ta chỉ gán K=Left (là tập các phần tử có bên tay trái của các phụ thuộc hàm)

Ví dụ:

Cho lược đồ quan hệ R = { A,B,C,D,E,G,H,I} và tập phụ thuộc hàm:

F= { AC → B, BI → ACD, ABC → D , H → I , ACE → BCG , CG → AE }

Tìm khoá K?

Ta có Left={A,B,C,H,E,G}

Bước 1: K=Left={A,B,C,H,E,G}

Bước 2:

Tập thuộc tính	A	B	C	D	E	G	H	I	Ghi chú
ABCHEG	x	x	x	x	x	x	x	x	
BCHEG	x	x	x	x	x	x	x	x	Loại A
CHEG	x	x	x	x	x	x	x	x	Loại B
CHG	x	x	x	x	x	x	x	x	Loại E

Như vậy, {C,H,G} là một khoá của R.

Nếu muốn tìm tất cả các khoá của R, ta cần thay đổi trật tự loại bỏ phần tử của khoá K.

6.2.6 Tập phụ thuộc hàm tương đương

Hai tập phụ thuộc hàm **F và G là tương đương** nếu:

- Tất cả các phụ thuộc hàm trong F có thể được suy ra từ G, và
- Tất cả các phụ thuộc hàm trong G có thể suy ra từ F.

Vì thế, F và G là tương đương nếu $F^+ = G^+$

Nếu F và G là tương đương thì ta nói **F phủ G hay G phủ F**.

Vì thế, thuật toán sau đây sẽ kiểm tra sự tương đương của hai tập phụ thuộc hàm:

- F phủ E: $\forall X \rightarrow Y \in E$, tính X^+ từ F, sau đó kiểm tra xem $Y \in X^+$
- E phủ F: $\forall X \rightarrow Y \in F$, tính X^+ từ E, sau đó kiểm tra xem $Y \in X^+$

6.2.7 Tập phụ thuộc hàm tối thiểu

Tập phụ thuộc hàm là tối thiểu nếu nó thoả mãn các điều kiện sau:

1. Chỉ có một thuộc tính nằm ở phía bên tay trái của tất cả các phụ thuộc hàm trong F.

2. Không thể bỏ đi bất kỳ một phụ thuộc hàm nào trong F mà vẫn có được một tập phụ thuộc hàm tương đương với F (tức là, không có phụ thuộc hàm dư thừa).

3. Không thể thay thế bất kỳ phụ thuộc hàm X→A nào trong F bằng phụ thuộc hàm Y→A, với Y⊂X mà vẫn có được một tập phụ thuộc hàm tương đương với F (tức là, không có thuộc tính dư thừa trong phụ thuộc hàm)

Nhận xét:

- Tất cả các tập phụ thuộc hàm đều có phụ thuộc hàm tối thiểu tương đương với nó.
- Có thể có nhiều phụ thuộc hàm tối thiểu

Thuật toán: Tìm tập phụ thuộc hàm tối thiểu G của F

1. Đặt G:=F.

2. Thay thế tất cả các phụ thuộc hàm $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ trong G bằng n phụ thuộc hàm: $X \rightarrow A_1$, $X \rightarrow A_2, \ldots$, $X \rightarrow A_n$.

3. Với mỗi phụ thuộc hàm X → A trong G, với mỗi thuộc tính B trong X nếu ((G - {X → A}) ∪ {(X - {B}) →A}) là tương đương với G, thì thay thế X→ A bằng (X - {B}) → A trong G. *(Loại bỏ thuộc tính dư thừa trong phụ thuộc hàm)*

4. Với mỗi phụ thuộc hàm X → A trong G, nếu (G - {X → A}) tương đương với G, thì loại bỏ phụ thuộc hàm X → A ra khỏi G.*(Loại bỏ phụ thuộc hàm dư thừa)*

6.3 Các dạng chuẩn của quan hệ

6.3.1 Định nghĩa các dạng chuẩn

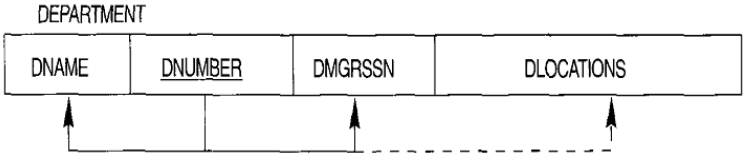
6.3.1.1 Dạng chuẩn 1(First Normal Form)

a. Định nghĩa

Một quan hệ ở dạng chuẩn 1 nếu các giá trị của tất cả thuộc tính trong quan hệ là **nguyên tử (**tức là chỉ có 1 giá trị tại một thời điểm**).**

b. Ví dụ:

- Quan hệ sau đây không phải ở dạng chuẩn 1:



DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

Vi phạm dạng chuẩn 1

Hình 6.3. Dữ liệu của quan hệ DEPARTMENT vi phạm 1NF

- Chuyển quan hệ trên thành dạng chuẩn 1 (bằng cách xác định tập thuộc tính {DNumber, DLocation} là khoá chính), ta có:

DEPARTMENT

Dư thừa

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Hình 6.4. Dư thừa dữ liệu trong quan hệ ở dạng chuẩn 1

c. Nhận xét:

- Quan hệ ở dạng chuẩn 1 có tồn tại sự dư thừa dữ liệu, trong quan hệ DEPARTMENT, nếu như một phòng có nhiều địa điểm khác nhau thì dữ liệu của 3 thuộc tính (DName, DNumber, DMgrSsn) bị lặp lại nhiều lần.
- Chúng ta có thể tách quan hệ DEPARTMENT thành 2 quan hệ:

DEPARTMENT

DNAME	DNUMBER	DMGRSSN

DEPT_LOCATIONS

DNUMBER	DLOCATION

Hình 6.5. Quan hệ DEPARTMENT được tách thành 2 quan hệ

Mô tả dữ liệu của 2 quan hệ này:

DEPARTMENT:

DName	DNumber	DMgrSsn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS:

DNumber	DLocation
5	Bellaire
5	Sugarland
5	Houston
4	Stafford
1	Houston

Hình 6.6. Minh họa dữ liệu của DEPARTMENT và DEPT_LOCATIONS

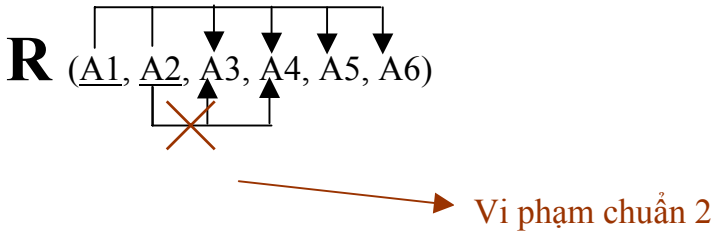
6.3.1.2 Dạng chuẩn 2(Second Normal Form_2NF)

a. Định nghĩa:

Một quan hệ ở dạng chuẩn 2 nếu:

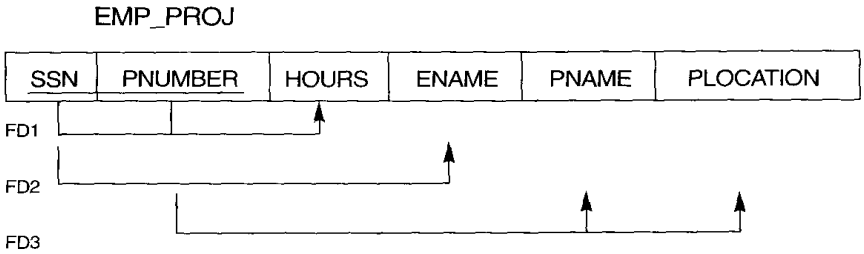
- Quan hệ đó ở dạng chuẩn 1
- Tất cả các thuộc tính không phải là khóa **phụ thuộc đầy đủ** vào khóa.
- Phụ thuộc đầy đủ: Phụ thuộc hàm Y →Z là phụ thuộc hàm đầy đủ nếu: $\forall A \in Y, (Y - \{A\}) \not\rightarrow Z$

b. Sơ đồ mô tả:



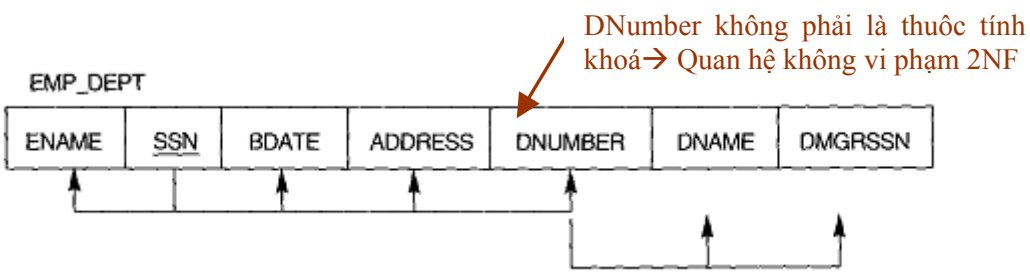
c. Ví dụ:

Ví dụ 1: Quan hệ EMP_PROJ **không phải ở dạng chuẩn 2** vì tồn tại 2 phụ thuộc hàm FD2, FD3 là **phụ thuộc hàm bộ phận** (trái với phụ thuộc hàm đầy đủ)



Hình 6.7. Lược đồ quan hệ EMP_PROJ và các phụ thuộc hàm

Ví dụ 2: Quan hệ sau đây **ở dạng chuẩn 2:**



Hình 6.8. Quan hệ EMP_DEPT ở dạng chuẩn 2

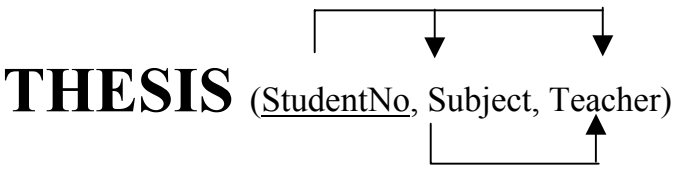
EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

Dư thừa dữ liệu

Hình 6.9. Minh hoạ dữ liệu của quan hệ EMP_DEPT

Ví dụ 3: Quan hệ sau đây **ở dạng 2NF:**



StudentNo	Subject	Teacher
SV01	1	Nguyễn Văn Hiệu
SV02	2	Ngô Lan Phương
SV03	1	Nguyễn Văn Hiệu
SV04	1	Nguyễn Văn Hiệu

Hình 6.10. Minh hoạ dữ liệu của quan hệ THESIS

d. Nhận xét:

- Quan hệ ở dạng chuẩn 2 có sự dư thừa thông tin.

- Dạng chuẩn 2 có thể bị vi phạm khi quan hệ có khóa **gồm hơn một thuộc tính**.

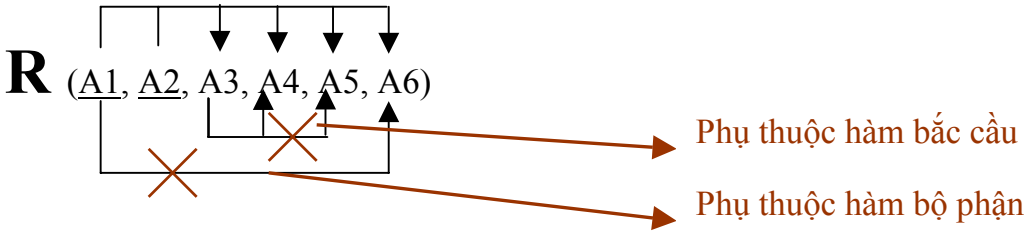
6.3.1.3 Dạng chuẩn 3 (Third Normal Form)

a. Định nghĩa

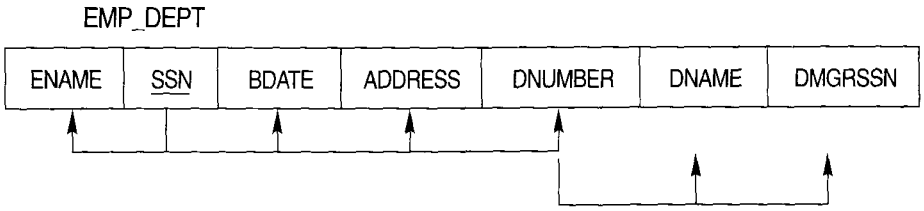
Một quan hệ ở dạng chuẩn 3 nếu:

- Quan hệ ở dạng chuẩn 2
- Và không có chứa các phụ thuộc hàm **phụ thuộc bắc cầu vào khoá.**
- **Phụ thuộc hàm phụ thuộc bắc cầu:** Phụ thuộc hàm Y→Z là phụ thuộc hàm bắc cầu nếu tồn tại hai phụ thuộc hàm:Y→X và X →Z.

b. Biểu diễn bằng sơ đồ

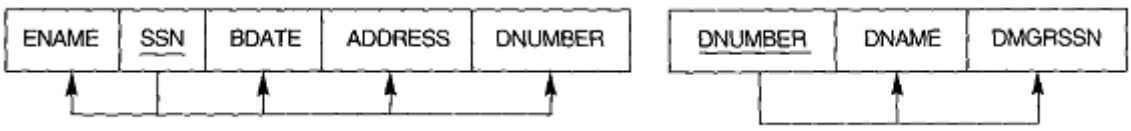


c. Ví dụ: Quan hệ EMP_DEPT **không phải ở dạng chuẩn 3** vì còn tồn tại phụ thuộc hàm DNumber→DName, DMgrSsn là phụ thuộc hàm phụ thuộc bắc cầu vào khoá.



Hình 6.11. Quan hệ EMP_DEPT không phải ở dạng chuẩn 3

Tách quan hệ trên thành 2 quan hệ: EMPLOYEE và DEPARTMENT. 2 quan hệ sau đều ở dạng chuẩn 3:



Hình 6.12. Tách quan hệ EMP_DEPT thành 2 quan hệ mới

EMPLOYEE

ENAME	SSN	BDATE	ADDRESS	DNUMBER
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5
Zelaya,Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4
Narayan,Remesh K.	666884444	1962-09-15	975 Fire Oak,Humble,TX	5
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Hình 6.13. Mô tả dữ liệu của quan hệ EMPLOYEE và DEPARTMET

d. Nhận xét:

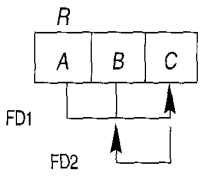
- Trong một cơ sở dữ liệu tốt, các quan hệ nên được chuyển về dạng chuẩn 3.
- Tuy nhiên, dữ liệu vẫn có khả năng dư thừa khi quan hệ có hai tập khóa dự tuyển gối lẫn nhau, hoặc quan hệ có thuộc tính không khóa xác định một thuộc tính khóa .

6.3.1.4 Dạng chuẩn Boyce _Codd(Boyce-Codd Normal Form)

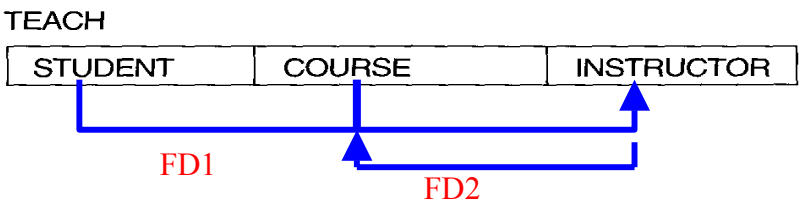
a. Định nghĩa

Quan hệ R ở dạng chuẩn BCNF khi tất cả các phụ thuộc hàm X →A trong R đều phải có X là khoá của R.

b. Ví dụ: Quan hệ sau ở dạng 3NF nhưng không phải BCNF.



A, B: thuộc tính khoá
C: không phải là thuộc tính khoá



STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Hình 6.14. Minh hoạ dữ liệu của quan hệ TEACH vi phạm chuẩn Boyce -Codd

Để nhận được quan hệ ở BCNF, ta có thể tách quan hệ trên:

Cách 1: R1(Student, Instructor) và R2(Student, Course)

Cách 2: R1(Couse, Instructor} và R2(Course, Student)

Cách 3: R1(Instructor, Course} và R2(Instructor, Student)

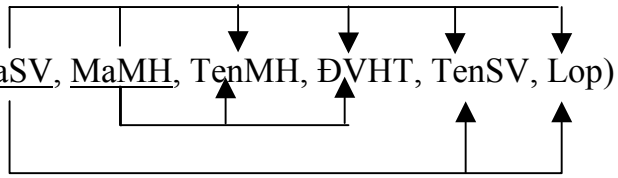
Lưu ý: Việc tách quan hệ như trên sẽ làm mất đi phụ thuộc hàm FD1.

6.3.2 Phép phân rã các lược đồ quan hệ

6.3.2.1 Định nghĩa

Phép phân rã các lược đồ quan hệ R={A1, A2, . . . , An}là việc thay thế lược đồ quan hệ R thành các lược đồ con {R1, . . . , Rk}, trong đó Ri⊆R và R=R1 ∪ R2…∪ Rk

Vídụ: Cho quan hệ R với các phụ thuộc hàm như sau:



Ta có thể phân rã thành 3 lược đồ R1(MaSV, TenSV, Lop) và R2(MaMH,TenMH, ĐVHT) và R3(MaSV, MaMH).

6.3.2.2 Phép phân rã không mất mát thông tin

Cho R là một lược đồ quan hệ, phép rã ρ=(R1,R2, . . .,Rn) và D là tập các phụ thuộc dữ liệu. Phép phân rã ρ không mất mát thông tin nếu khi thực hiện phép toán kết nối tự nhiên các quan hệ thành phần R1, R2,…,Rn ta vẫn nhận được kết quả của quan hệ ban đầu.

Ví dụ về một phép phân rã có mất mát thông tin:

Cho quan hệ:

MaSV	MaMH	Điem
1	A	3
2	A	5
3	A	6
4	B	6
5	C	9

Nếu ta phân rã quan hệ trên thành 2 quan hệ: R1(MaSV, MaMH) và R2(MaMH, Điem) như sau:

R1:

MaSV	MaMH
1	A
2	A
3	A
4	B
5	C

R2:

MaMH	Điem
A	3
A	5
A	6
B	6
C	9

Thực hiện phép kết nối tự nhiên 2 quan hệ R1 và R2:

R1*R2=

MaSV	MaMH	Điem
1	A	3
1	A	5
1	A	6
2	A	3
2	A	5
2	A	6
3	A	3
3	A	5
3	A	6
4	B	6
5	C	9

Như vậy, khi nối tự nhiên 2 bảng, ta nhận được quan hệ không giống quan hệ ban đầu→ Phép phân rã trên là mất mát thông tin.

Vấn đề đặt ra đối với người thiết kế là phải tìm ra những phép phân rã không làm mất mát thông tin (chi tiết sẽ được trình bày ở phần sau). Bây giờ chúng ta sẽ tìm hiểu một thuật toán để kiểm tra một phép phân rã có mất mát thông tin hay không.

6.3.2.3 Thuật toán kiểm tra phép phân rã không mất mát thông tin

Input:

- Lược đồ quan hệ R={A1, A2, . . . , An}
- Tập các phụ thuộc hàm F
- Phép tách ρ(R1, R2, . . . , Rk)

Output: Kết luận phép tách ρ không mất mát thông tin.

Các bước của thuật toán:

Bước 1:

- Thiết lập một bảng với n cột (tương ứng với n thuộc tính) và k dòng (tương ứng với k quan hệ), trong đó cột thứ j ứng với thuộc tính Aj, dòng thứ i ứng với lược đồ Ri.
- Tại dòng i và cột j, ta điền ký hiệu aj nếu thuộc tinh Aj∈Ri.Ngược lại ta điền ký hiệu bij.

Bước 2:

- Xét các phụ thuộc hàm trong F và áp dụng cho bảng trên.
- Giả sử ta có phụ thuộc hàm X→Y∈F, xét các dòng có giá trị bằng nhau trên thuộc tính X **thì làm bằng** các giá trị của chúng trên Y. Ngược lại làm bằng chúng bằng ký hiệu bij. Tiếp tục áp dụng các pth cho bảng (kể cả việc lặp lại các phụ thuộc hàm đã áp dụng) cho tới khi không còn áp dụng được nữa.

Bước 3:

Xem xét bảng kết quả. Nếu xuất hiện một dòng chứa toàn giá trị **a1, a2 ,…,an** thì kết luận phép tách ρ không mất mát thông tin.

Vi dụ: Cho quan hệ:

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

Hình 6.15. Minh họa dữ liệu của quan hệ EMP_DEPT

Tách quan hệ trên thành 2 quan hệ:

EMPLOYEE

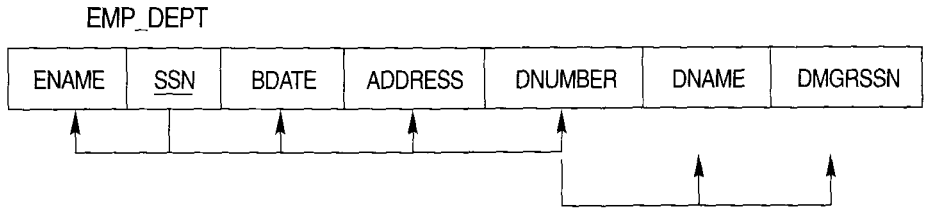
ENAME	SSN	BDATE	ADDRESS	DNUMBER
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5
Zelaya,Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4
Narayan,Ramesh K.	666884444	1962-09-15	975 Fire Oak,Humble,TX	5
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1

DEPARTMENT

DNAME	DNUMBER	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Hình 6.16. Quan hệ EMPLOYEE được phân rã (tách) thành 2 quan hệ

Tập phụ thuộc hàm F:



Kiểm tra phép tách trên là không mất mát thông tin:

Bước 1:

	EName	SSN	BDate	Address	DNumber	DName	DMgrSsn
EMPLOYEE	a1	a2	a3	a4	a5	b16	b17
DEPARTMENT	b21	b22	b23	b24	a5	a6	a7

Bước 2: Xét phụ thuộc hàm DNumber→ DName, DMgrSsn. Ta nhận thấy có giá trị a5 ở dòng thứ 2, nên ta sẽ làm bằng giá trị a6, a7 cho dòng thứ 1.

Bước 3: Tồn tại một dòng chứa giá trị a1, a2,..a7. Kết luận, phép phân rã trên không mất mát thông tin.

	EName	SSN	BDate	Address	DNumber	DName	DMgrSsn
EMPLOYEE	a1	a2	a3	a4	a5	a6	a7
DEPARTMENT	b21	b22	b23	b24	a5	a6	a7

Ghi chú: Sinh viên thực hiện phép nối tự nhiên 2 quan hệ EMPLOYEE và DEPARTMENT trên để kiểm tra có bằng quan hệ ban đầu EMP_DEPT

6.4 Chuẩn hoá quan hệ

Chuẩn hoá quan hệ là việc phân rã một lược đồ quan hệ thành các lược đồ con ở dạng chuẩn 3 hoặc ở BCNF sao cho vẫn bảo toàn phụ thuộc và không mất mát dữ liệu.

6.4.1 Thuật toán phân rã lược đồ quan hệ thành các lược đồ quan hệ con ở BCNF

Input:

- Lược đồ quan hệ R
- Tập phụ thuộc hàm F

Output:

Phép phân rã của R không mất thông tin và mỗi lược đồ quan hệ trong phép tách đều ở dạng BCNF đối với phép chiếu của F trên lược đồ đó.

Các bước của thuật toán:

- Ban đầu phép tách ρ chỉ bao gồm R.
- Nếu S là một lược đồ thuộc ρ và S chưa ở dạng BCNF thì chọn phụ thuộc hàm X → A thỏa trong S, trong đó X không chứa khóa của S và A∉X. {phụ thuộc hàm vi phạm định nghĩa dạng chuẩn BCNF}. Thay thế S trong ρ bởi S1 và S2 như sau S1 = XA, S2 = S\A.
- Quá trình trên tiếp tục cho đến khi tất cả các lược đồ quan hệ đều ở dạng BCNF

Ví dụ:

Cho lược đồ quan hệ R(CTHRSG).

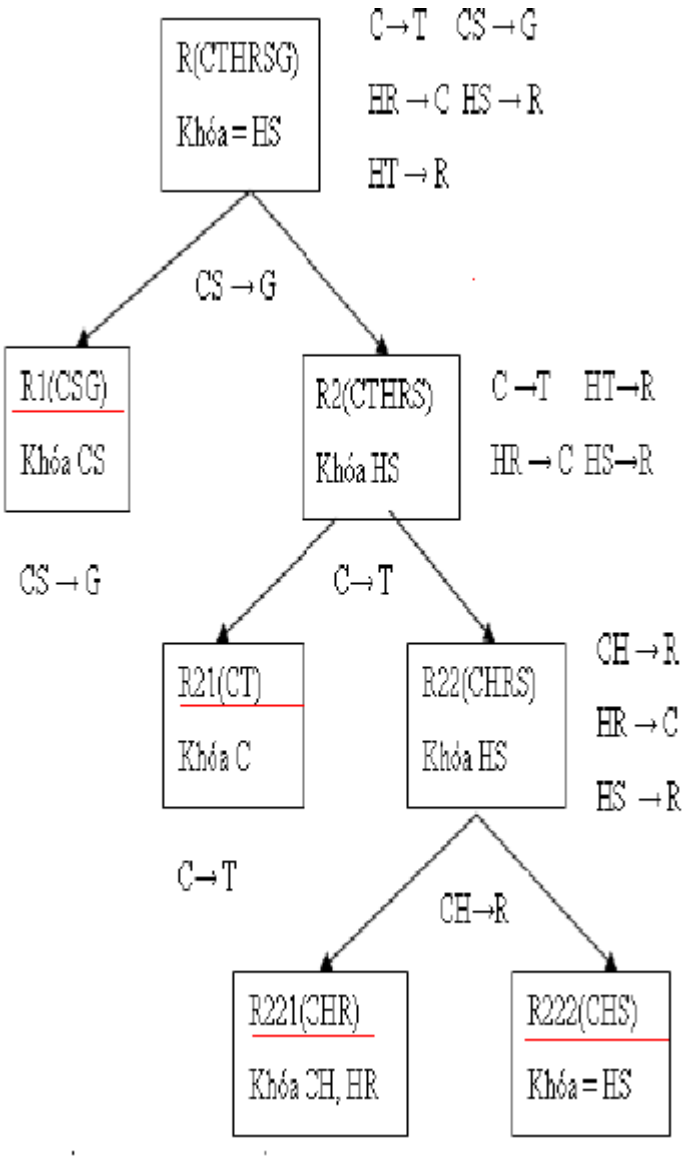
Trong đó:

- C: Course; T: Teacher; H: Hour; R: Room; S: Student; G:Group).
- Và tập các phụ thuộc hàm F:
  - C → T: Mỗi khoá học (course) có một thầy (teacher) duy nhất.
  - HR →C: Tại một thời điểm (Hour) ở tại phòng học (room) chỉ có một khoá học duy nhất.

- HT→ R: Tại một thời điểm và một giáo viên chỉ ở một phòng duy nhất
- CS→G: Một sinh viên học một course thì chỉ ở một lớp duy nhất.
- HS → R: Một sinh viên, ở một thời điểm nhất định chỉ ở trong một phòng duy nhất.

Dựa vào thuật toán tìm khoá→Khóa của R là **HS.**

Yêu cầu: Tách lược đồ R thành các lược đồ con ở dạng BCNF.



Hình 6.17. Biểu diễn quá trình tách quan hệ R thành các quan hệ ở BCNF

Như vậy, quan hệ R được tách thành 4 quan hệ R1, R21, R221, R222 đều ở BCNF.

6.4.2 Thuật toán phân rã một lược đồ quan hệ thành các lược đồ con ở 3NF.

Input:

- Lược đồ quan hệ R
- Tập các phụ thuộc hàm F, không làm mất tính tổng quát giả sử đó là phủ tối thiểu.

Output:

Phép tách không mất mát thông tin trên R thành các lược đồ con ở dạng chuẩn 3 sao cho vẫn bảo toàn các phụ thuộc hàm.

Các bước của thuật toán:

- Bước 1: Loại bỏ các thuộc tính của R nếu thuộc tính đó không liên quan đến phụ thuộc hàm nào của F.(không có mặt ở cả hai vế của phụ thuộc hàm).
- Bước 2: Nếu có một phụ thuộc hàm của F liên quan đến tất cả các thuộc tính của R thì kết quả chính là R.
- Bước 3: Ngoài ra, phép tách ρ đưa ra các lược đồ gồm các thuộc tính XA ứng với phụ thuộc hàm $X \rightarrow A \in F$. Nếu tồn tại các phụ thuộc hàm X→A1, X→A2, …,X→An thuộc F thì thay thế XAi (1<=i<= n) bằng XA1A2. . .An. Quá trình tiếp tục.
- Chú ý: Tại mỗi bước kiểm tra lược đồ R, **nếu mỗi thuộc tính không khóa không phụ thuộc bắc cầu vào khóa chính,** thì R đã ở dạng 3NF, ngược lại cần áp dụng bước 3 để tách tiếp.

Ví dụ:

Cho lược đồ quan hệ R(C,T,H,R,S,G) với tập phụ thuộc hàm tối thiểu F:

C→T, HR→C, HT→R, CS→G, HS→ R.

Yêu cầu: Phân rã lược đồ quan hệ trên thành các quan hệ con đều ở dạng 3NF.

- Sử dụng thuật toán tìm khoá→ Khoá chính của R là HS.
- Thực hiện thuật toán:
  - Bước 1: Không có thuộc tính bị loại bỏ
  - Bước 2: Không có phụ thuộc hàm nào liên quan tới tất cả các thuộc tính
  - Bước 3:
    - Phụ thuộc hàm C→ T vi phạm 3NF (phụ thuộc bắc cầu vào khoá), vì vậy tách R thành R1(C,T) và R2(C,H,R,S,G).
    - Phụ thuộc hàm CS→G vi phạm 3NF(phụ thuộc bộ phận vào khoá), tách R2 thành R21(C,S,G) và R22(C,H,R,S).

- Phụ thuộc hàm HR→C vi phạm 3NF, tách R22 thành R221(H,R,C) và R222(H,S,R)

Như vậy, quan hệ R được tách thành các quan hệ sau: R1, R21, R221, R222

Lưu ý:

- Kết quả của phép tách có thể khác nhau phụ thuộc vào thứ tự áp dụng các phụ thuộc hàm khi thực hiện thuật toán.
- Sinh viên tự kiểm tra xem việc tách quan hệ như trên có mất mát thông tin không.

Bài tập:

1. Cho một quan hệ R ={A, B, C, D, E, F, G, H, I, J} và tập phụ thuộc hàm

F = { A,B→ C
A→ D, E
B→ F
F→ G, H
D→ I, J
}

Yêu cầu:

- Tìm {A}+ ={D, E, I ,J }
- Tìm khóa của quan hệ R.
- Tách quan hệ R thành BCNF.
- Kiểm tra xem việc tách trên có mất mát thông tin không?

2. Lặp lại yêu cầu ở bài 1 với tập phụ thuộc hàm sau:

G= {A,B → C
B, D→ E, F
A, D → G, H
A→ I
H→J}

3. Cho một quan hệ R ={CourseNo, SecNo, OfferingDept, Credit_Hours, CourseLevel, InstructorSSN, Semester, Year, Days_Hours, RoomNo, NoOfStudents} và tập phụ thuộc hàm:

F ={ CourseNo → OfferingDept, Credit_Hours, CourseLevel;

CourseNo, SecNo, Semester, Year →Days_Hours, RoomNo, NoOfStudents, InstructorSSN;

RoomNo, Days_Hours, Semester, Year → InstructorSSN, CourseNo, SecNo }

Yêu cầu:

- Tìm khóa của quan hệ R.
- Quan hệ trên thuộc dạng chuẩn mấy?
- Tách quan hệ về dạng 3NF.
- Kiểm tra xem việc tách trên có mất mát thông tin không?

7 Chương 7. THIẾT KẾ CƠ SỞ DỮ LIỆU VẬT LÝ (Tham khảo)

Thiết kế cơ sở dữ liệu vật lý là quá trình chuyển các đặc tả dữ liệu lôgic thành các đặc tả kỹ thuật để lưu trữ dữ liệu. Gồm 2 nội dung sau:

Lựa chọn công nghệ lưu trữ (Hệ điều hành, HQTCSDL, các công cụ truy nhập dữ liệu).

Chuyển các quan hệ của mô hình logic thành các thiết kế vật lý.

Trong chương này sẽ trình bày những phần sau:

Thiết kế các trường, bản ghi vật lý

Thiết kế file vật lý

Thiết kế cơ sở dữ liệu vật lý

7.1 Nội dung thiết kế file vật lý và cơ sở dữ liệu vật lý

7.1.1 Quá trình thiết kế

Trong quá trình thiết kế hệ thống vật lý, vấn đề đặt ra hàng đầu là phải làm thế nào tối thiểu hóa không gian lưu trữ và thời gian người dùng tương tác với hệ thống.

Tuy nhiên, do dung lượng các thiết bị nhớ tăng nhanh, nên người ta tập trung nhiều vào việc xử lý các file và dữ liệu sao cho hiệu quả hơn đối với người sử dụng.

Các thông tin cần thiết để thiết kế file vật lý:

Các quan hệ đã được chuẩn hóa, kể cả ước lượng về số lượng dữ liệu cần lưu trữ

Định nghĩa chi tiết các thuộc tính

Các mô tả cho biết ở đâu và khi nào dữ liệu được sử dụng (xem, thêm, sửa, xóa).

Các yêu cầu và mong đợi về sử dụng dữ liệu và tích hợp dữ liệu, bao gồm các yêu cầu về thời gian đáp ứng, các mức độ an toàn, ghi tạm, phục hồi….

Các mô tả về công nghệ được sử dụng để triển khai file và CSDL (thiết bị lưu trữ, hệ điều hành, HQTCSDL…)

Một số các quyết định cơ bản có ý nghĩa đối với sự tích hợp và hiệu năng của hệ thống ứng dụng cần thực hiện:

Chọn định dạng lưu trữ (kiểu dữ liệu) cho mỗi thuộc tính sao cho tối thiểu hóa dư thừa thông tin và tối đa sự tích hợp dữ liệu.

Nhóm gộp các thuộc tính từ mô hình dữ liệu lô gic vào bản ghi vật lý.

Sắp xếp các bản ghi có quan hệ với nhau vào bộ nhớ ngoài sao cho từng bản ghi hay nhóm các bản ghi lưu trữ, cập nhật và lấy ra nhanh chóng (gọi là tổ chức file)

Lựa chọn phương tiện và cấu trúc để lưu trữ dữ liệu đảm bảo truy nhập hiệu quả hơn.

7.1.2 Sản phẩm thiết kế

Sản phẩm thiết kế là một tập các đặc tả mà các nhà lập trình và các nhà phân tích dữ liệu sẽ sử dụng để xác định định dạng và cấu trúc các file trong bộ nhớ thứ cấp của máy tính (bộ nhớ ngoài).

Khi sử dụng các công cụ CASE, kho dữ liệu của CASE hay từ điển dữ liệu dự án là nơi lưu trữ tất cả các đặc tả nêu ra ở trên.

Sau đây là các phần tử tiêu biểu của thiết kế được lưu trữ trong kho dữ liệu của CASE khi thiết kế file và cơ sở dữ liệu vật lý:

BẢNG MÔ TẢ CÁC TRƯỜNG

Loại đặc tả	Mô tả nội dung
Tên trường (field name)	Theo quy định về cách đặt tên trường của HQTCSDL.
Kiểu trường (data type)	Chọn kiểu dữ liệu mà HQTCSDL đó hỗ trợ
Kích cỡ (size)	Là kích thước tối đa dùng để lưu trữ dữ liệu của trường đó
Mã hóa (Coding)	Cách viết tắt giá trị của trường. Ví dụ, mỗi nước được biểu diễn bằng hai ký tự
Các quy tắc toàn vẹn dữ liệu (data integrity rules)	Các đặc tả về các hạn chế đặt lên giá trị của trường
Các kiểm soát bảo trì (maintenance controls)	Chỉ ra những giá trị nào được phép thay đổi
Công thức (Formular)	Mô tả công thức tính toán giá trị với những trường số cần tính toán.
Toàn vẹn tham chiếu (references integrity)	Đặc tả giá trị của trường có liên quan đến giá trị của trường khác
Sở hữu (Ownership)	Ai là người sở hữu trường đó (có quyền đối với dữ liệu)

BẢNG CÁC ĐẶC TẢ TIÊU BIỂU ĐỐI VỚI THIẾT KẾ BẢN GHI

Các trường (fields)	Danh sách các trường trong một bản ghi
Dữ liệu có cấu trúc (Structure Data)	Định nghĩa cấu trúc dữ liệu dùng để lư trữ bản ghi (Thứ tự các trường, khóa chính, khóa ngoại…)
Sự lưu trữ lại (retention)	Đặc tả những bản ghi nào đó được giữ lại trong file bao lâu (dữ liệu về sinh viên không được lưu trữ quá 10 năm sau khi ra trường).

BẢNG CÁC ĐẶC TẢ TIÊU BIỂU ĐỐI VỚI THIẾT KẾ FILE

Tên file và định vị	Tên file theo quy định của HQTCSDL và thiết bị lưu trữ nó.
Các bản ghi (record)	Những bản ghi nào được lưu trữ trong file.
Khóa chính (Primary Key)	Là một hay một số trường được dùng để định danh duy nhất cho bản ghi.
Chỉ số hóa (index)	Chỉ ra các trường được dùng để lập chỉ số
Yếu tố khóa bản ghi (Record blocking factor)	Số các bản ghi theo mỗi trang hoặc khóa của bản ghi (Ví dụ: 10 bản ghi của ITEM được lưu trữ trong một trang bộ nhớ ngoài)
Lưu giữ lại và sao lưu (Retention and Backup)	File được lưu trữ trong bao lâu và các thủ tục sao lưu, thời gian định kỳ cần sao lưu (để đảm bảo an toàn khi có sự cố).
Tổ chức file (file organization)	Phương pháp truy nhập dữ liệu và sắp xếp các bản ghi trong file
Kiểm soát (controls)	Đặc tả về kiểm soát và phương pháp mã hóa

BẢNG CÁC ĐẶC TẢ TIÊU BIỂU ĐỐI VỚI THIẾT KẾ CSDL

Các file	Các file trong CSDL và nơi định vị nó
Kiến trúc (Architecture)	Loại hình cấu trúc (bao gồm cả mô hình) CSDL được dùng để tổ chức file.
Các mối quan hệ	Cơ chế để liên kết file với nhau.

7.2 Thiết kế các trường

Một thuộc tính trong mô hình dữ liệu logic được biểu diễn bằng một số trường (fields).

Ví dụ: HoTenSV được biểu diễn thành 2 trường HodemSV và TenSV

7.2.1 Yêu cầu thiết kế trường

Mỗi HQTCSDL sử dụng những kiểu dữ liệu nhất định để lưu trữ dữ liệu.

Trong yêu cầu thiết kế trường, quan trọng nhất là phải chọn kiểu dữ liệu phù hợp, ta thường quan tâm đến các mục tiêu sau khi chọn kiểu dữ liệu:

Tiết kiệm không gian lưu trữ

Biểu diễn được mọi giá trị có thể thuộc miền giá trị

Cải thiện tính toàn vẹn (tổ chức việc nhập dữ liệu, kiểm tra dữ liệu đầu vào)

Hỗ trợ thao tác dữ liệu (Ví dụ: thao tác với dữ liệu số nhanh hơn với ký tự)

7.2.2 Chọn kiểu và cách biểu diễn dữ liệu

7.2.2.1 Kiểu dữ liệu

Các kiểu dữ liệu mà HQTCSDL SQL hỗ trợ và ý nghĩa của nó

DECIMAL(m,n)	Số thập phân có độ dài là m chữ số và n số thập phân
INTEGER	Số nguyên lớn (độ dài tối đa là 11 chữ số)
SMALLINT	Số nguyên nhỏ (độ dài tối đa là 6 chữ số)
FLOAT(m,n)	Số thực có độ dài là m chữ số và n số thập phân
CHAR	Xâu ký tự có độ dài là m ký tự
DATE	Kiểu dữ liệu thời gian và có rất nhiều cách biểu diễn
LOGICAL	Giá trị logic (đúng/sai)

7.2.2.2 Các trường tính toán

Khi giá trị của một trường là giá trị nhận được từ các giá trị của trường khác thì trường đó gọi là trường tính toán.

Có các loại tính toán sau:

+ Tính toán số học: Lương= Hệ số lương * 210.

+ Tính toán lôgic: Tiền trợ cấp = | 50.000 đ nếu cán bộ là nữ.
| 0 nếu cán bộ là nam.

+ Tính toán hỗn hợp:

Tiền điện= S ố điện * 500đ nếu số điện < 100.

Số điện *500 + (Số điện -100)* 750 nếu số điện >100.

7.2.2.3 Các kỹ thuật mã hóa dữ liệu và nén dữ liệu

Một số phương pháp mã hóa dùng để biểu diễn dữ liệu trong các trường lưu trữ:

Mã hóa phân cấp: để mô tả các dữ liệu phân cấp người ta dùng nhiều nhóm, mỗi nhóm đại diện cho cấp và các nhóm được sắp xếp lần lượt từ trái sang phải.

Ví dụ: Hệ thống phân loại sách trong thư viện:

Các cấp			Mã số	Tên tài khoản
1	2	3		
500			500	Khoa học tự nhiên
	1		5001	Toán học
	2		5002	Vật lý
	1	1	50011	Toán giải tích
	1	2	50012	Toán rời rạc

Mã liên tiếp: Mã này được tạo ra theo quy tắc một dãy liên tục, như 1, 2, 3 … A, B, C…. Mã loại này dùng cho những dữ liệu là danh sách như danh sách sinh viên. Nó đơn giản, dễ tự động hóa, không nhầm lẫn. Tuy nhiên nó không gợi nhớ về đối tượng được mã hóa và không cho phép chèn thêm vào giữa.

Mã gợi nhớ: Căn cứ vào đối tượng được mã hóa để cấu tạo mã. Ví dụ: VND (Đồng Việt Nam), TL001 (Thủy lợi 001)…Loại này giúp ta nhận ra đối tượng được mã hóa, có thể nới rộng hoặc thu hẹp số lượng mã. Tuy nhiên khó tổng hợp và phân tích.

Mã thành phần ngữ nghĩa: Theo phương pháp này, mã được chia làm nhiều thành phần, mỗi phần mô tả một đặc trưng nhất định của đối tượng như phân loại, địa danh… Những phần này có thể sử dụng các nhóm ký tự khác nhau. Mã loại này rất thông dụng và được sử dụng nhiều trong công nghiệp cũng như giao tiếp quốc tế.

Ví dụ: Địa chỉ miền trên internet có dạng:

<Tên tổ chức>.<Loại tổ chức>.<Tên nước>

Ví dụ : **hwru.edu.vn**: Đại học Thủy Lợi, Tổ chức giáo dục, Tên nước

Mã loại này cồng kềnh, và cần chọn các thành phần sao cho ổn định, nếu không việc sử dụng mã sẽ gặp nhiều khó khăn.

7.2.2.4 Kiểm tra tính toàn vẹn của dữ liệu

Để đảm bảo tính đúng đắn của dữ liệu người ta đặt các ràng buộc trên các dữ liệu đó.

Thường dùng các phương pháp sau để kiểm tra tính toàn vẹn:

Giá trị ngầm định (default value): Là giá trị được gán sẵn cho một trường nào đó khi bản ghi mới được nhập vào. Ví dụ: Trong hóa đơn bán hàng, trường ngày bán được mặc định là ngày hiện tại.

Kiểm tra khuôn dạng (picture control): Là mẫu định dạng bao gồm độ rộng, các giá trị có thể trong từng vị trị. Ví dụ: TLA006, $999,999.99.

Kiểm tra giới hạn (range control): Các trường có thể đưa ra các giới hạn đối với các giá trị của nó. Ví dụ: Điểm mộn học được giới hạn là các số và được giới hạn từ 0..10.

Tính toàn vẹn tham chiếu (reference integrity): là giá trị của thuộc tính đã cho có thể bị hạn chế bởi giá trị của những thuộc tính khác. Ví dụ: Trong mối quan hệ 1_N, nếu giá trị của bảng bên 1 chưa có thì sẽ không được có bên N.

Kiểm tra giá trị rỗng (Null value control): Nếu đặt một thuộc tính nào đó là khác rỗng thì bắt buộc ta phải thêm giá trị cho trường đó.

Quản lý dữ liệu mất: Trong khi vận hành, nếu vì một lý do nào đó mà dữ liệu có thể bị mất. Khi thiết kế file vật lý, các nhà thiết kế phải chỉ ra cách thức mà hệ thống quản lý dữ liệu bị mất. Balad và Hofer đã đưa ra một số phương pháp sau đây dùng để quản lý dữ liệu của 1 trường bị mất:

Cho quy trình để ước lượng giá trị bị mất.

Theo dõi dữ liệu bị mất để báo cáo và sử dụng một phần tử hệ thống giúp con người mau chóng thay thế giá trị bị mất này.

Thực hiện một số kiểm tra để có thể bỏ qua dữ liệu bị mất hay phải phục hồi nó nếu nó thực sự ảnh hưởng đến kết quả của hệ thống.

7.3 Thiết kế các bản ghi vật lý

Một bản ghi vật lý là một nhóm các trường được lưu trữ ở các vị trí liền kề nhau và được gọi ra cùng nhau như một đơn vị thống nhất.

Thiết kế bản ghi vật lý là chọn một nhóm các trường của nó sẽ lưu trữ ở những vị trí liền kề nhau nhằm 2 mục tiêu: sử dụng hiệu quả không gian lưu trữ và tăng tốc độ truy nhập. Hệ điều hành đọc hay ghi dữ liệu vào bộ nhớ thứ cấp theo một đơn vị gọi là trang. Một trang này có dung lượng cụ thể phụ thuộc vào hệ điều hành và máy tính cụ thể.

Vấn đề đặt ra ở đây là phải thiết kế các bản ghi thế nào để tận dụng được dung lượng chứa của trang. Nếu dung lượng của trang tận dụng được càng nhiều thì số lần đọc càng ít và tốc độ truy cập càng nhanh.

Để làm được điều này người ta thường phi chuẩn hóa một số quan hệ nhận được.

7.3.1 Phi chuẩn

Việc phi chuẩn hóa các quan hệ đã chuẩn hóa trong nhiều trường hợp là cần thiết để tận dụng dung lượng trang của máy.

BENHNHAN(MaBN, TenBN, Diachi_BN, Ngay_nhap, Giuong_phong, Khoa, Tinh_trang, Ngayra, ThanhToan)

Ta có thể phân chia nó thành 2 quan hệ mới để có độ dài gần với dung lượng trang:

BENHNH1(MaBN, TenBN, Diachi_BN, Khoa)

BENHNH2(MaBN, Ngay_nhap, Giuong_phong, Tinh_trang, Ngayra, ThanhToan)

Có một số dạng phi chuẩn hóa, nhưng không có một quy tắc chặt chẽ nào. Rodger đã thảo luận đến một số trường hợp chung có thể xét phi chuẩn:

Hai thực thể có quan hệ một – một.

Ví dụ: Có 2 quan hệ có mối liên kết 1_1 như sau:

SINHVIEN(MaSV, TenSV, MaThe)

THEDOC(MaThe, DiaChi, NgayCap, MaSV)

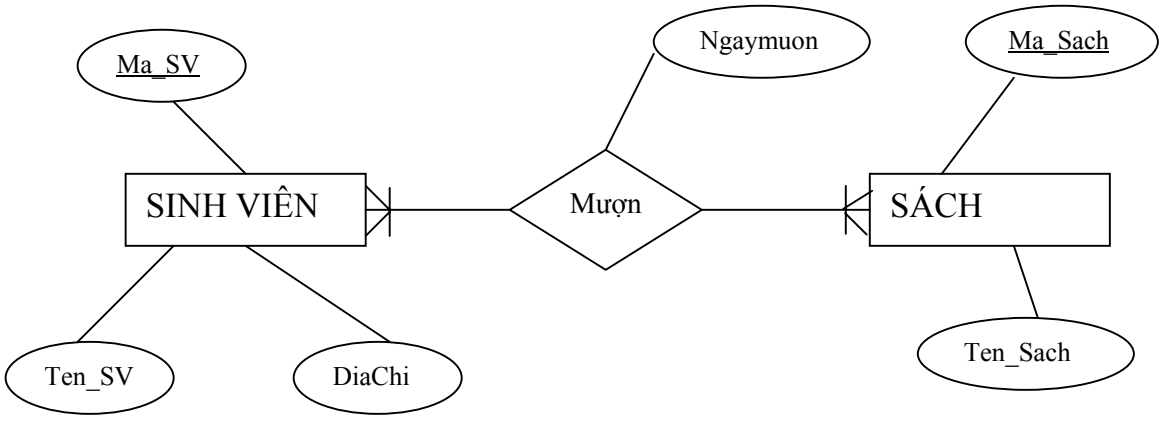
Phi chuẩn hóa ta có quan hệ sau:

SINHVIEN(MaSV, TenSV, MaThe, DiaChi, NgayCap)

Và trong trường hợp này MaThe, DiaChi, NgayCap có thể bỏ trống đối với những SV không có thẻ.

Hai thực thể có mối quan hệ M_N trong đó liên kết có thuộc tính riêng.

Ví dụ: Có quan hệ như sau:



Sau khi chuẩn hóa, ta nhận được 3 quan hệ sau:

SINHVIÊN(Ma_SV, Ten_SV, DiaChi)

SÁCH(Ma_Sach, Ten_Sach)

MƯỢN(Ma_SV, Ma_Sach, Ngay_muon)

Phi chuẩn hóa ta được:

SINHVIÊN(Ma_SV, Ten_SV, DiaChi)

MƯỢN(Ma_SV, Ma_Sach, TenSach, NgayMuon)

Dữ liệu tham chiếu: Trong quan hệ 1_N nếu bảng ở bên 1 không tham gia vào một quan hệ nào khác thì ta có thể hợp nhất 2 thực thể này thành 1.

Ví dụ:

KHO (Ma_Kho, Ten_Kho, Loai_Kho)

HANG(Ma_Hang, Ten_Hang)

Phi chuẩn hóa ta được:

HANG(Ma_Hang, Ten_Hang, Ma_Kho, Ten_Kho, Loai_Kho)

7.3.2 Quản lý trường có độ dài cố định

Việc thiết kế bản ghi sẽ rất dễ dàng nếu trường có độ dài cố định (vì tính ngay được độ dài bản ghi).

Trong trường hợp này việc xác định vị trí của một trường chỉ bằng phép toán: Vị trí con trỏ hiện thời + (độ dài bản ghi * số bản ghi)

7.3.3 Quản lý trường có độ dài biến đổi

Một trường có độ dài thay đổi như trường Memo thì định vị trí của một trường hay một bản ghi cụ thể không đơn giản.

Một cách chung để quản lý trường có độ dài thay đổi là đưa các bản ghi có độ dài cố định vào một bản ghi vật lý có độ dài cố định và đưa những bản ghi vật lý có độ dài thay đổi vào một bản ghi vật lý có độ dài thay đổi. Đó chính là kỹ thuật thiết kế

bản ghi vật lý tự động được sử dụng trong hầu hết các hệ quản trị CSDL cho máy tính nhỏ.

7.4 Thiết kế file vật lý

File vật lý là một phần nhỏ của bộ nhớ thứ cấp (đĩa cứng, băng từ…) lưu các bản ghi vật lý một cách độc lập. Việc lưu trữ các bản ghi vật lý ở vị trí nào đối với người dùng không quan trọng nhưng lại được các nhà thiết kế đặc biệt quan tâm.

7.4.1 Các loại file

Một hệ thống thông tin có thể cần đến 6 loại file sau:

File dữ liệu (Data file- master file): là file chứa dữ liệu liên quan với mô hình dữ liệu vật lý và lôgic. File này luôn tồn tại, nhưng nội dung thay đổi.

File lấy từ bảng (look up table file): La danh sách các dữ liệu tham chiếu lấy từ một hay một số file khác theo một yêu cầu nào đó.

File giao dịch (Transaction file): là file dữ liệu tạm thời phục vụ các hoạt động hàng ngày của một tổ chức. File này thường được thiết kế để phục vụ các yêu cầu xử lý nhanh.

File làm việc (Work file): Là file tạm thời dùng để lưu kết quả trung gian, file này sẽ tự động xóa đi mỗi khi không cần thiết.

File bảo vệ (Protection file): là file được thiết kế để khắc phục những sai sót trong quá trình hệ thống hoạt động. Các file này cho hình ảnh của file dữ liệu trước và sau những hoạt động nhất định (cập nhật, sửa đổi, xử lý…) của hệ thống.

File lịch sử (History file): File này ghi lại quá trình hoạt động của hệ thống, cũng có thể là các dữ liệu cũ hiện không cần sử dụng.

Việc tổ chức các loại file khác nhau không chỉ liên quan đến việc tổ chức lưu trữ và khai thác dữ liệu, mà còn liên quan đến các hoạt động xử lý dữ liệu trong quá trình hoạt động của hệ thống. Về nguyên tắc, việc sử dụng càng ít file càng tốt. Tuy nhiên, việc đưa vào các file là cần thiết cho việc đảm bảo an toàn dữ liệu (file bảo vệ, file lịch sử), tăng tốc độ truy cập hay xử lý (file giao dịch, file lấy từ bảng, file làm việc) ….

7.4.2 Các phương pháp truy cập

Mỗi hệ điều hành trợ giúp một số kỹ thuật khác nhau để tìm kiếm và lấy thông tin ra gọi là các phương pháp truy cập.

Về cơ bản, có 2 loại phương pháp truy cập:

Phương pháp truy cập trực tiếp, sử dụng tính toán để xác định địa chỉ chính xác của một bản ghi và truy nhập trực tiếp đến bản ghi đó.

Phương pháp gián tiếp, hỗ trợ việc tìm kiếm bản ghi thứ n xuất phát từ một vị trí hiện thời của con trỏ hay điểm bắt đầu của 1 file.

Mặc dù có tồn tại 2 phương pháp như vậy, nhưng ít khi chúng ta sử dụng trực tiếp mà thường thông qua các công cụ có sẵn mà phần mềm hệ thống trợ giúp.

7.4.3 Tổ chức file

Cách tổ chức file là kỹ thuật sắp xếp các bản ghi vật lý của một file trên một thiết bị nhớ thứ cấp. Tổ chức một file cụ thể cần tính toán đến các yếu tố sau:

Lấy dữ liệu nhanh.

Thông lượng các giao dịch xử lý lớn

Sử dụng hiệu quả không gian nhớ.

Tránh được sai sót khi mất dữ liệu

Tối ưu hóa nhu cầu tổ chức file.

Đáp ứng được nhu cầu khi tăng dữ liệu

Trước khi nghiên cứu thiết kế tổ chức các CSDL cần xem xét sơ qua cách tổ chức của dữ liệu trong bộ nhớ ngoài.

Mô hình tổ chức bộ nhớ ngoài.

Bộ nhớ ngoài (hay còn gọi là bộ nhớ thứ cấp) là các thiết bị lưu trữ như đĩa từ, băng từ…

Đĩa từ được phân thành các khối vật lý (tổ chức đống) có kích cỡ như nhau: khoảng 512 bytes đến 4K (4*1024=4096 bytes) và được đánh địa chỉ khối. Địa chỉ này gọi là địa chỉ tuyệt đối trên đĩa.

Mỗi tệp dữ liệu trên đĩa từ chiếm 1 hoặc nhiều khối, mỗi khối chứa 1 hoặc nhiều bản ghi. Việc thao tác với tệp thông qua tên tệp thực chất là thông qua địa chỉ tuyệt đối của các khối.

Mỗi bản ghi đều có địa chỉ và thường được xem là địa chỉ tuyệt đối của byte đầu tiên của bản ghi hoặc là địa chỉ của khối chứa bản ghi đó.

Các phép toán đặc trưng trên tệp dữ liệu là:

Thêm một bản ghi.

Xóa một bản ghi

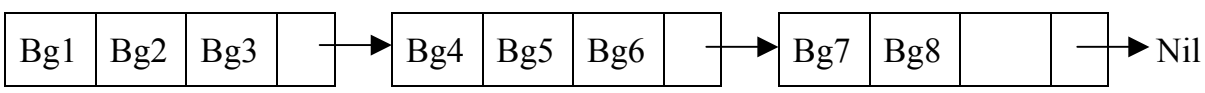
Sửa một bản ghi

Tìm một bản ghi theo điều kiện

7.4.3.1 Tổ chức file tuần tự (Sequential file)

Trong tổ chức file tuần tự, các bản ghi được sắp xếp tuần tự.

Giả sử: Hiện tại tệp cơ 8 bản ghi. 1 khối chứa 3 bản ghi. Con trỏ tệp trỏ đến bản ghi đầu tiên.



Thêm bản ghi: lần theo con trỏ đến kiểm tra khối cuối cùng (khối 3) xem có đủ bộ nhớ không. Nếu không đủ thì phải cấp thêm khối mới.

Xóa bản ghi (có giá trị khóa =k):

- Tìm đến bản ghi đó
- Xóa :
- Xóa logic: chỉ đánh dấu xóa
- Xóa vật lý: Xóa hẳn bản ghi đó

Tìm kiếm bản ghi (có giá trị khóa =k): Tìm lần lượt từ trên xuống dưới cho đến khi gặp khóa k.

Sửa đổi giá trị thuộc tính:

- Tìm đến thuộc tính cần sửa
- Ghi đè giá trị mới lên giá trị cũ

7.4.3.2 Tổ chức file băm (Hashed File)

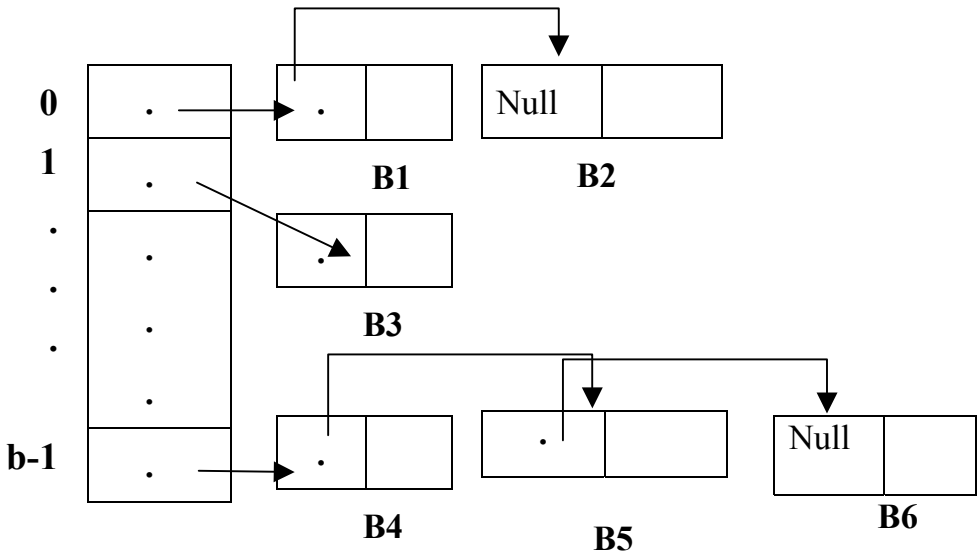
- Khái niệm hàm băm: Mỗi bản ghi đều có 1 khóa là giá trị số (giả sử là k)
- Hàm băm H(k)=b. Trong đó, b là số cụm (hàm băm sẽ tác động lên giá trị khóa và trả lại 1 số nguyên là số cụm)

Như vậy, tư tưởng của hàm băm là phân chia tập hợp các bản ghi của tệp dữ liệu thành các cụm (Buckets). Mỗi cụm bao gồm một hoặc nhiều khối, mỗi khối chứa một số lượng cố định các bản ghi.

Mỗi cụm ứng với một địa chỉ băm được đánh số từ 0..b-1. Ở mỗi đầu của khối đều chứa con trỏ trỏ đến khối tiếp theo trong cụm, khối cuối cùng trong cụm chứa con trỏ rỗng.

Có một bảng chỉ dẫn cụm (bucket directory): chứa k con trỏ, mỗi con trỏ chứa địa chỉ khối đầu tiên của từng cụm.

Ví dụ:



Hình 7.1. Tổ chức file băm

Thêm bản ghi mới (với giá trị khóa k):

Tính H(k): Nếu H(k)=b thì bổ sung vào nhóm b (Lần theo con trỏ, thêm vào như tổ chức tuần tự).

Tìm kiếm bản ghi (với giá trị khóa k):

Tính H(k): Nếu H(k)=b thì b chính là nhóm chứa bản ghi có khóa là k. Sau đó tìm kiếm tuần tự trên nhóm đó.

Xóa bản ghi (với giá trị khóa k):

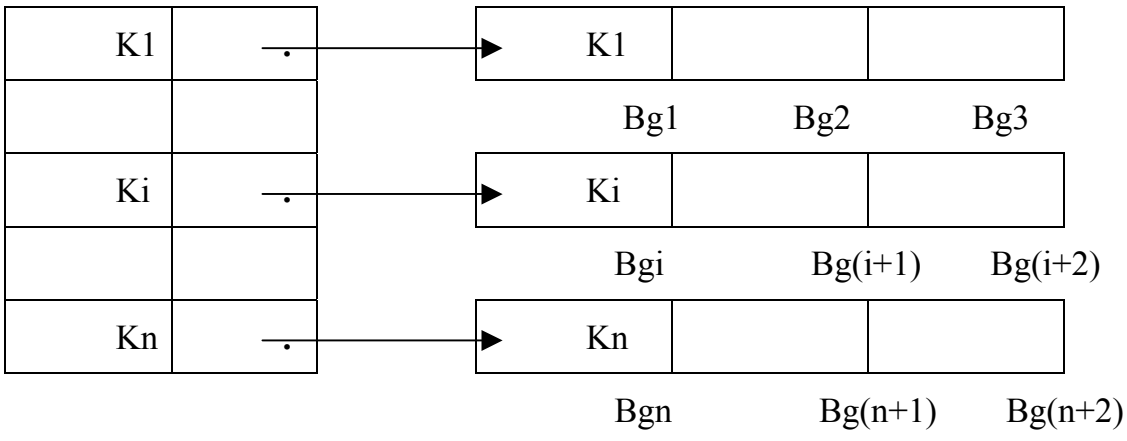
- Tìm đến bản ghi có khóa k bằng cách tính H(k): Nếu H(k)= b thì tìm bản ghi đó trong nhóm b.
- Xóa logic hoặc xóa vật lý.
- Nếu bản ghi là duy nhất trong khối thì khi xóa bản ghi sẽ đồng thời giải phóng khối khỏi cụm chứa khối.

Sửa đổi thuộc tính:

- Sửa đổi thuộc tính không phải là khóa: Ghi đè giá trị mới lên giá trị cũ.
- Sửa đổi thuộc tính khóa: Xóa bản ghi cũ, thêm bản ghi mới vào.
- Tính H(k').
  - Nếu h(k) =h(k') → Ghi đè lên
  - Nếu h(k) <> h(k') → Xóa bản ghi cũ, thêm bản ghi mới vào.

7.4.3.3 Tổ chức file chỉ số

Một kiểu tổ chức tệp dữ liệu truy nhập khóa rất thường dùng trong CSDL là tệp chỉ số.



Hình 7.2. Tổ chức file chỉ số

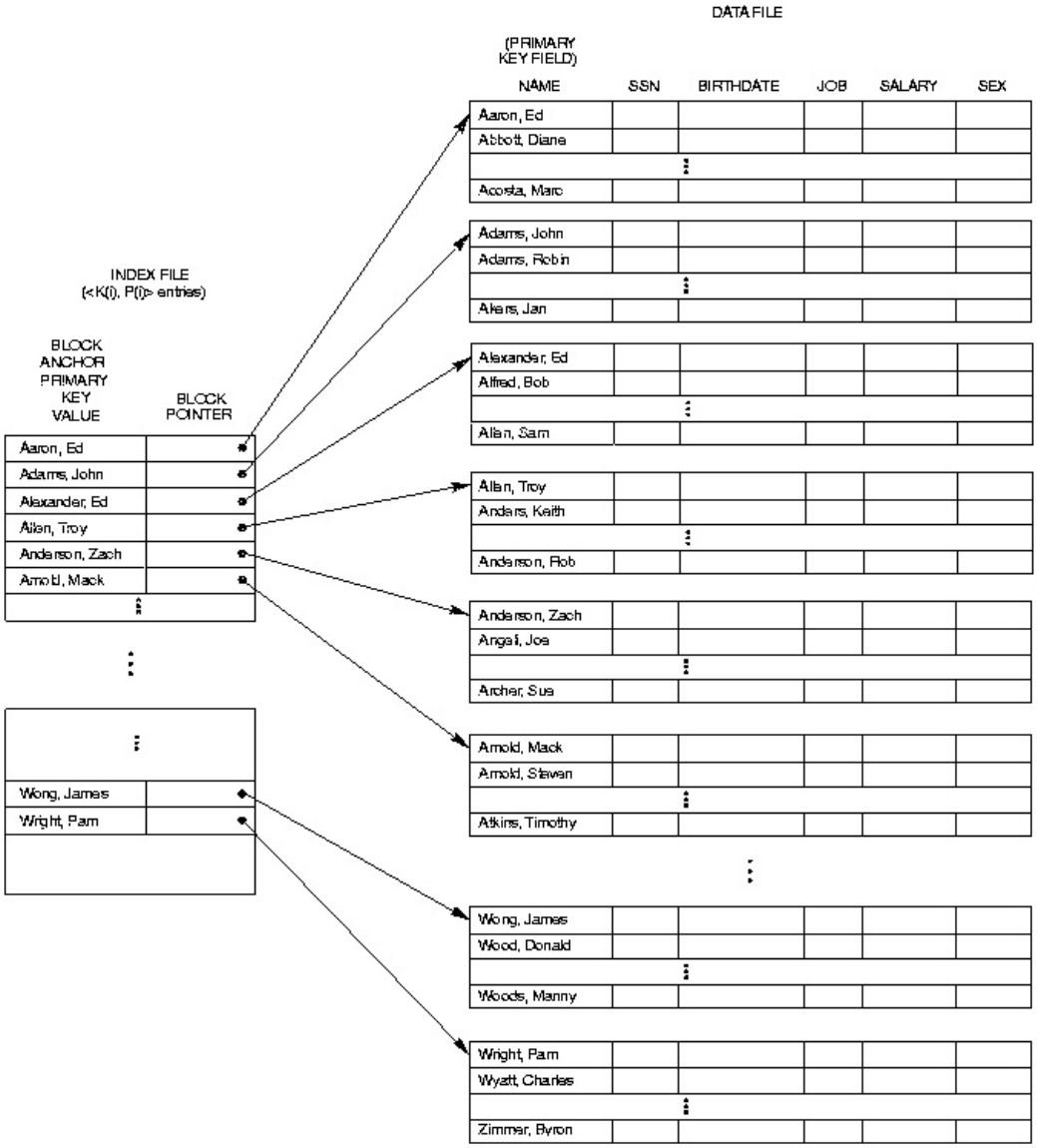
Trong đó:

K1: Giá trị khóa của bản ghi đầu tiên của khối thứ nhất.

Tệp chỉ dẫn có 2 trường: khóa và con trỏ

Tệp chính có n khối, trong khối chứa các bản ghi.

Một minh họa về tệp chỉ số:



Hình 7.3. Minh họa tệp chỉ số

Thêm bản ghi (có giá trị khóa k):

- Tìm vị trí khối cần thêm (H(k)=b).
- Nếu khối đó còn chỗ trống thì thêm bản ghi vào theo đúng thứ tự sắp xếp và chú ý thay đổi khóa trong bảng chỉ dẫn khối nếu có sự thay đổi.

Xóa bản ghi (có giá trị khóa k):

- Quá trình giống như thêm một bản ghi. Tuy nhiên, nếu khi xóa tạo ra khối rỗng, khi đó có thể xóa bỏ toàn bộ khối.
- Tìm kiếm bản ghi (có giá trị khóa k): Có thể tìm kiếm tuần tự hoặc nhị phân.

Sửa đổi thuộc tính:

- Thuộc tính khóa: Xóa cũ, thêm mới.
- Thuộc tính không khóa: Ghi đè lên thuộc tính cũ.

Bảng so sánh các cách tổ chức file khác nhau

Yếu tố	Cách tổ chức file		
	Tuần tự	Chỉ số	Băm
Không gian lưu trữ	Không lãng phí	Cần nhiều không gian hơn vì phải lưu file chỉ số	Cần nhiều không gian khi thêm và xóa bản ghi (phải tính H(k))
Truy nhập tuần tự theo khóa chính	Rất nhanh	Trung bình	Không thực tế
Truy nhập ngẫu nhiên theo khóa chính	Không thực tế	Trung bình	Rất nhanh
Xóa bản ghi	Tạo ra khoảng trống và có thể yêu cầu tổ chức lại	Nếu không gian nhớ được bố trí động thì dễ dàng, nhưng yêu cầu phải bảo trì các chỉ số.	Rất dễ dàng
Thêm bản ghi	Yêu cầu sắp xếp lại file sau khi thêm	Nt	Nt
Sửa bản ghi	Yêu cầu sắp xếp lại file sau khi sửa	Dễ dàng, nhưng yêu cầu phải bảo trì các chỉ số	Rất dễ dàng

7.4.4 Ví dụ về thiết kế file

Xét Ví dụ sau: Có 2 chứng từ

ĐƠN ĐẶT HÀNG

Số hóa đơn: xxxxxx

Người đặt hàng: (30 ký tự……..)

Địa chỉ: (50 ký tự……..)

Ngày đặt: dd/mm/yyyy

Số tt	Tên hàng	Mô tả hàng	Đơn vị tính	Số lượng
Xx	Char(15)	Char(50)	Char(10)	xxxxx
…	….	….	….	….

PHIẾU GIAO HÀNG

Số phiếu: xxxxxx

Tên khách hàng: (30 ký tự……..)

Địa chỉ: (50 ký tự……..)

Nơi giao hàng: (50 ký tự……..)

Ngày giao: dd/mm/yyyy

Số tt	Tên hàng	Đơn vị tính	Đơn giá	Số lượng	Thành tiền
Xx	Char(15)	Char(10)	Number	Number	Number
…	….	….	….	….	….

Sau quá trình thiết kế logic ta được các quan hệ sau:

KHACH(Ma_Khach, Ten_kh, DiaChi_kh)

HANG(Ma_hang, Ten_hang, Mota_hang, Donvi)

DONHANG(So_Don, Ma_Khach, NgayDon)

PHIEUGIAO(So_Phieu, Ma_Khach , Noi_Giao, Ngay_Giao)

DONGDON(So_Don, Ma_Hang, So_luongD)

DONGPHIEUGIAO(So_Phieu, MaHang, So_luonggi, DonGia)

Khi tiến hành thiết kế vật lý, ta được các file sau:

1. KHACH

Tên thuộc tính	**Kiểu dữ liệu**	**Kích cỡ**	**Khuôn dạng**
Ma_khach	Ký tự	6	Chữ hoa +số
Ten_kh	Ký tự	30	Chữ đầu viết hoa
Diachi_kh	Ký tự	30	Chữ đầu viết hoa

2. HANG

Tên thuộc tính	**Kiểu dữ liệu**	**Kích cỡ**	**Khuôn dạng**
Ma_hang	Ký tự	6	Chữ hoa+số
Ten_hang	Ký tự	15	Chữ đầu viết hoa
Mota_hang	Ký tự	30	Chữ đầu viết hoa
Don_vi	Ký tự	10	Chữ thường

3. DONHANG

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
So_don	Ký tự	6	Chữ hoa +số
Ma_khach	Ký tự	15	Chữ hoa +số
Ngay_don	Ngày	8	Dd/mm/yy

4. DONGDON

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
So_don	Ký tự	6	Chữ hoa +số
Ma_hang	Ký tự	6	Chữ hoa +số
So_luongd	Số	7	Số nguyên

5. PHIEUGIAO

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
So_phieu	Ký tự	6	Chữ hoa +số
Ma_khach	Ký tự	6	Chữ hoa +số
Ngay_giao	Ngày	30	Dd/mm/yy
Noi_giao	Ký tự	30	
Tong_tien	Số	9	Số thực

6. DONGPHIEU

Tên thuộc tính	Kiểu dữ liệu	Kích cỡ	Khuôn dạng
So_phieu	Ký tự	6	Chữ hoa+ số
Ma_hang	Ký tự	6	Chữ hoa + số
Don_gia	Số	6	Số thực
So_luonggi	Số	7	Số thực

TÀI LIỆU THAM KHẢO

1. Elmasri & Navathe: *Fundamentals of Database Systems*, International Edition.
2. Nguyễn Tiến Vương: *Cơ sở dữ liệu quan hệ*
3. Date, C.J., and Darwen, H.: *A Guide to the SQL Standard, 3rd ed*., Addison-Wesley.

……..o0o……..

Cơ sở dữ liệu nâng cao

CHƯƠNG 1
TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU QUAN HỆ

Trong chương này chúng ta hệ thống lại các khái niệm cơ bản của cơ sở dữ liệu quan hệ. Mục đích là định nghĩa các thuật ngữ, đưa ra bộ khung cơ sở cho các phần sau. Việc chọn mô hình cơ sở dữ liệu quan hệ làm hệ thống cơ sở có nhiều lý do: thứ nhất là cơ sở toán học vững chắc của mô hình quan hệ làm nó trở thành mô hình lý tưởng trong việc giải quyết các vấn đề lý thuyết, hai là phần lớn các vấn đề trình bày trong các chương sau có thể mô tả dễ dàng bởi mô hình quan hệ, ba là thị trường hệ quản trị cơ sở dữ liệu quan hệ rất phát triển và vẫn đang mở rộng và cuối cùng là phần lớn các hệ cơ sở dữ liệu sau này cũng thuộc loại quan hệ.

Mô hình quan hệ có ba đặc trưng cơ bản:

1. Cấu trúc dữ liệu đơn giản. Chúng là các quan hệ (relation), được biểu diễn như các bảng 2 chiều với phần tử là các bản ghi (record, data item). Nó cung cấp đặc tính độc lập ở mức độ cao so với dạng biểu thị vật lý của dữ liệu.
2. Mô hình quan hệ cung cấp một nền tảng vững chắc, bảo đảm được tính nhất quán dữ liệu (data consistency). Thiết kế cơ sở dữ liệu được hỗ trợ bởi quá trình chuẩn hoá, giúp loại bỏ các bất thường dữ liệu. Các trạng thái nhất quán của một cơ sở dữ liệu có thể được định nghĩa và duy trì một cách thống nhất qua các quy tắc toàn vẹn cơ sở dữ liệu (integrity rules).
3. Mô hình quan hệ cho phép các thao tác trên quan hệ theo kiểu tập hợp. Đặc tính này đã dẫn đến việc phát triển các ngôn ngữ phi thủ tục mạnh mẽ với nền tảng là lý thuyết tập hợp (đại số quan hệ) hoặc lôgic (phép tính quan hệ).

1. Khái niệm cơ sở dữ liệu

1.1. Cơ sở dữ liệu

Dữ liệu được lưu trữ trên các thiết bị lưu trữ theo một cấu trúc nào đó để có thể phục vụ cho nhiều người sử dụng với nhiều mục đích khác nhau gọi là *cơ sở dữ liệu.*

1.2. Hệ quản trị cơ sở dữ liệu

Phần mềm cho phép một hoặc nhiều người tạo lập, lưu trữ, cập nhật và khai thác cơ sở dữ liệu gọi là *hệ quản trị cơ sở dữ liệu* (DataBase Management Systems - DBMS).

Vai trò chính của hệ quản trị cơ sở dữ liệu là cho phép người dùng thao tác với dữ liệu thông qua các thuật ngữ trìu tượng, khác với việc máy tính lưu trữ dữ liệu. Theo nghĩa này hệ quản trị cơ sở dữ liệu có nhiệm vụ như là một bộ thông dịch (interpreter) với ngôn ngữ bậc cao nhằm giúp người dùng sử dụng hệ thống mà không cần quan tâm đến cách biểu diễn dữ liệu trong máy hoặc các thuật toán chi tiết. Ví dụ người dùng không cần biết hệ quản trị cơ sở dữ liệu *Access* tổ chức dữ liệu theo kiểu hàm băm, kiểu file chỉ mục hay kiểu cây cân bằng, và cũng không cần biết thuật toán thực hiện lệnh sắp xếp là Quick Sort, thuật toán nổi bọt hay sắp xếp nhị phân ...

Một cơ sở dữ liệu gồm một hoặc nhiều tập tin được thiết kế theo một cấu trúc nhất định và có quan hệ chặt chẽ với nhau. Cơ sở dữ liệu được dùng chung cho

nhiều người và nhiều mục đích khác nhau, vì vậy sẽ tiết kiệm được tài nguyên, giảm thiểu sự trùng lặp thông tin, bảo đảm tính nhất quán thông tin.

1.3. Các đặc trưng của phương pháp cơ sở dữ liệu

+ *Chia sẻ dữ liệu.* Mục đích chính của cách tiếp cận cơ sở dữ liệu là dữ liệu được chia sẻ bởi nhiều người dùng hợp pháp.

+ *Giảm thiểu dư thừa dữ liệu.* Dữ liệu dùng chung cho nhiều bộ phận, thay vì được lưu trữ phân tán trùng lặp nay được lưu trữ tập trung một chỗ theo một cấu trúc thống nhất.

+ *Tính tương thích dữ liệu.* Việc loại bỏ sự dư thừa dữ liệu kéo theo hệ quả là sự tương thích dữ liệu. Ví dụ khi địa chỉ nhân viên thay đổi thì tất cả các bộ phận đều được cập nhật địa chỉ mới.

+ *Tính toàn vẹn dữ liệu* (data integrity). Mỗi cơ sở dữ liệu cần đảm bảo một số loại ràng buộc toàn vẹn (integrity constraints). Đặc biệt khi người dùng thực hiện các thao tác như chèn, xoá hay sửa đổi dữ liệu thì các ràng buộc đó cần phải được kiểm tra một cách chặt chẽ.

+ *Bảo mật dữ liệu (data security).* Khi có nhiều người cùng chia sẻ dữ liệu, việc bảo đảm an toàn dữ liệu và bảo mật thông tin là tối quan trọng. Cần phải có cơ chế bảo mật như mật khẩu (*password*) và phân quyền truy cập dữ liệu (*data access rights*).

+ *Tính đồng bộ dữ liệu* (synchronization). Thông thường cơ sở dữ liệu được nhiều người dùng truy cập đồng thời, gây nên sự cạnh tranh dữ liệu. Vì vậy cần có cơ chế bảo vệ chống sự không tương thích bởi các thao tác cùng lúc lên dữ liệu.

+ *Tính độc lập dữ liệu.* Sự tách biệt cấu trúc mô tả dữ liệu khỏi chương trình ứng dụng sử dụng dữ liệu gọi là độc lập dữ liệu. Điều này cho phép phát triển tổ chức dữ liệu mà không cần sửa đổi chương trình ứng dụng. Sự độc lập dữ liệu là một trong mục tiêu chính của cách tiếp cận cơ sở dữ liệu.

Tương tự như một phần mềm, *vòng đời* của cơ sở dữ liệu gồm có các giai đoạn chính sau:

- Lập kế hoạch cơ sở dữ liệu (database planning).
- Khảo sát, phân tích (study and analysis).
- Thiết kế cơ sở dữ liệu (database design).
- Cài đặt cơ sở dữ liệu (database implementation).
- Bảo trì cơ sở dữ liệu (post-implementation).

1.4. Lược đồ dữ liệu và thể hiện dữ liệu

Khi thiết kế cơ sở dữ liệu ta tạo ra cấu trúc cơ sở dữ liệu, cái đó gọi là *lược đồ dữ liệu.* Ví dụ lược đồ dữ liệu hồ sơ nhân sự gồm các thành phần sau:

Họ tên, ngày sinh, hệ số lương

Các thành phần của lược đồ dữ liệu gọi là *thuộc tính* hoặc *trường*.

Khi sử dụng cơ sở dữ liệu thì ta làm việc với dữ liệu thật sự, đó là sự *thể hiện dữ liệu*. Ví dụ đối với lược đồ trên ta có thể có các thể hiện dữ liệu sau:

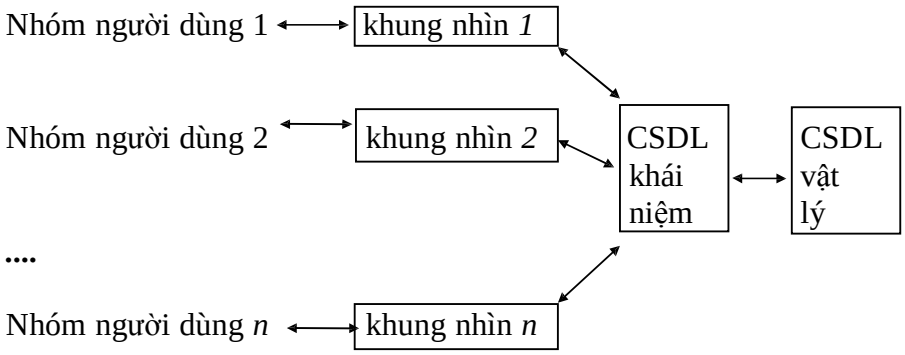
Nguyễn Văn A, 20/08/1970, 3.40

Trần Thị B, 15/05/1962, 4.12

Mỗi thể hiện dữ liệu gọi là *bộ* hay *bản ghi*.

1.5. Các mức trìu tượng dữ liệu

Giữa máy tính thao tác với các bit, người thiết kế cơ sở dữ liệu và người dùng có cách nhìn khác nhau đối với dữ liệu, đó chính là các *mức trìu tượng dữ liệu*. Sơ đồ chuẩn về các mức trìu tượng như sau:



a. Cơ sở dữ liệu vật lý: nằm cố định trong các thiết bị lưu trữ như đĩa và băng từ. Bản thân cơ sở dữ liệu vật lý cũng có nhiều mức trìu tượng khác nhau: từ mức bản ghi và file trong ngôn ngữ lập trình như PASCAL, qua mức bản ghi lôgic được hỗ trợ bởi hệ điều hành, đến mức các bit và địa chỉ vật lý trong các thiết bị lưu trữ.

b. Cơ sở dữ liệu khái niệm (schema): là sự trìu tượng thế giới thực đối với một đối tượng nào đó. Hệ quản trị cơ sở dữ liệu cung cấp ngôn ngữ thiết kế dữ liệu để thiết kế sơ đồ khái niệm. Đây là ngôn ngữ bậc cao cho phép mô tả cơ sở dữ liệu khái niệm bằng ngôn ngữ "mô hình dữ liệu". Một ví dụ điển hình là đồ thị có hướng trong mô hình mạng, trong đó các nút biểu diễn các đơn thể và các cung biểu diễn quan hệ.

c. Khung nhìn hoặc *lược đồ con (subschema)*: là mô hình trìu tượng một phần của cơ sở dữ liệu khái niệm. Có những hệ trang bị công cụ gọi là ngôn ngữ thiết kế khung nhìn cho phép khai báo khung nhìn, và công cụ gọi là ngôn ngữ thao tác dữ liệu khung nhìn để diễn tả câu hỏi và các thao tác đối với khung nhìn.

Theo một nghĩa nào đó, khung nhìn là cơ sở dữ liệu khái niệm và cùng mức trìu tượng như cơ sở dữ liệu khái niệm. Mặt khác khung nhìn có thể trìu tượng hơn theo nghĩa dữ liệu của nó được suy ra từ cơ sở dữ liệu khái niệm.

d. Độc lập dữ liệu

Độc lập dữ liệu dữ liệu giữa các mức trìu tượng có ý nghĩa rất quan trọng đối với cơ sở dữ liệu.

- *Độc lập dữ liệu mức vật lý*: Sự thay đổi lược đồ dữ liệu vật lý không làm thay đổi lược đồ dữ liệu mức khái niệm và mức khung nhìn.

Ta cần hiểu rằng sự thay đổi tổ chức vật lý dữ liệu có thể làm ảnh hưởng đến hiệu quả chương trình ứng dụng. Sự độc lập dữ liệu mức vật lý đảm bảo không phải viết lại chương trình chỉ vì lý do thay đổi cách tổ chức dữ liệu. ý nghĩa của tính độc lập dữ liệu mức vật lý là nó cho phép ta tinh chỉnh cơ sở dữ liệu mức vật lý để tăng hiệu quả sử dụng trong khi các chương trình ứng dụng vẫn chạy bình thường như không có vấn đề gì xảy ra.

- *Độc lập dữ liệu lôgic*: Sự thay đổi lược đồ dữ liệu khái niệm không làm thay đổi khung nhìn.

Trong quá trình sử dụng cơ sở dữ liệu có thể ta phải sửa đổi hiệu chỉnh lược đồ khái niệm, chẳng hạn thêm thông tin về thực thể mà cơ sở dữ liệu mô tả.

1.6. Ngôn ngữ dữ liệu

Mỗi hệ quản trị cơ sở dữ liệu cần phải có ngôn ngữ riêng của mình. Có hai loại ngôn ngữ cơ sở dữ liệu.

a. Ngôn ngữ mô tả dữ liệu (Data Definition Language - DDL). Gồm các lệnh cho phép khai báo, hiệu chỉnh cấu trúc cơ sở dữ liệu, mô tả các mối quan hệ của dữ liệu cũng như các quy tắc áp đặt lên dữ liệu. Ngôn ngữ mô tả dữ liệu được xây dựng dựa trên loại mô hình dữ liệu (mô hình quan hệ, mô hình mạng, mô hình phân cấp, mô hình hướng đối tượng ...) mà hệ quản trị cơ sở dữ liệu tương ứng được thiết kế.

b. Ngôn ngữ thao tác dữ liệu (Data Manipulation Language - DML). Là bộ lệnh cho phép người dùng thực hiện các công việc:

- Cập nhật dữ liệu như thêm, sửa, xoá.
- Truy vấn, tổng hợp dữ liệu.
- Các hàm tính toán.
- Bảo mật dữ liệu.

* *Giao tiếp ngôn ngữ chủ* : là khả năng cho phép chương trình viết trong các ngôn ngữ bậc cao như COBOL, C , ... có thể truy cập xử lý dữ liệu trong cơ sở dữ liệu.

* *Ngôn ngữ truy vấn SQL* : là ngôn ngữ có cú pháp tiếng Anh, giúp người dùng có thể thao tác dữ liệu dễ dàng mà không cần lập trình. Đây là loại ngôn ngữ thế hệ thứ 4 với đặc trưng *phi thủ tục*.

2. Khái niệm cơ sở dữ liệu quan hệ

2.1. Miền

Miền là tập hợp các giá trị. Người ta thường dùng chữ hoa để ký hiệu miền.

◊ *Ví dụ*. Các tập hợp sau là các miền:

- Tập các số nguyên.
- Tập các xâu ký tự độ dài không quá 30 ký tự.
- A = {0,1}.

2.2. Tích Đề-các

Tích Đề-các của các miền $D_1, D_2,..., D_n$, ký hiệu là

$$D_1 \times D_2 \times ... \times D_n,$$

là tập hợp tất cả *n-bộ* $(v_1, v_2, ..., v_n)$ thoả mãn

$$v_1 \in D_1, v_2 \in D_2, ..., v_n \in D_n.$$

Tức là

$$D_1 \times D_2 \times ... \times D_n = \{(v_1, v_2,..., v_n) \mid v_1 \in D_1, v_2 \in D_2, ..., v_n \in D_n \}.$$

◊ *Ví dụ.* Cho $D_1 = \{0,1\}$, $D_2 = \{a,b,c\}$. Khi đó tích đề-các

$$D_1 \times D_2 = \{(0,a),(0,b),(0,c),(1,a),(1,b),(1,c)\}$$

2.3. Quan hệ

Quan hệ là tập con của tích đề-các của 1 hoặc nhiều miền. Quan hệ có thể có hữu hạn hoặc vô hạn số phần tử. Trong giáo trình này ta giả thiết rằng quan hệ có hữu hạn phần tử.

◊ *Ví dụ*

Cho $D_1 = \{0,1\}$, $D_2 = \{a,b,c\}$. Tập **r** $= \{(0,a),(1,b),(1,c)\} \subset D_1 \times D_2$, vậy **r** là quan hệ trên D_1 và D_2.

Ta nói quan hệ **r** có *bậc n* nếu **r** là tập con của tích đề-các của *n* miền.

Mỗi phần tử của quan hệ gọi là *bộ*. Mỗi bộ của quan hệ bậc *n*, còn gọi là *n-bộ*, có *n thành phần*. Mỗi thành phần của bộ là *nguyên tố*, có nghĩa không thể phân tách được thành các thành phần nhỏ hơn.

Để trực quan ta có thể coi quan hệ như một *bảng* trong đó mỗi *hàng* là một bộ và mỗi *cột* ứng với một thành phần.

Dữ liệu được tổ chức dưới dạng các quan hệ có liên quan với nhau gọi là *cơ sở dữ liệu quan hệ.*

Mỗi cột của quan hệ được gán một tên gọi là *thuộc tính*.

Tập hợp tất cả các tên thuộc tính của quan hệ gọi là *lược đồ quan hệ*.

Tập hợp các lược đồ quan hệ của một cơ sở dữ liệu gọi là *lược đồ cơ sở dữ liệu quan hệ.*

· ***Các tính chất của quan hệ***

- Các giá trị trên cột phải cùng một miền giá trị và đơn trị.

Giá trị trên giao của cột và hàng đơn trị, không chấp nhận nhiều giá trị.

- Mỗi hàng là duy nhất.

Không cho phép hai hàng hoàn toàn giống nhau.

◊ *Ký hiệu*

Lược đồ quan hệ R có các thuộc tính $A_1, A_2, ..., A_n$ ký hiệu là

$$R = (A_1, A_2, ..., A_n)$$

Quan hệ **r** với lược đồ $R = (A_1, A_2, ..., A_n)$ có thể viết

r(R) hoặc **r**(A_1, A_2, ..., A_n)

Cho **r** là quan hệ với lược đồ R = (A_1, A_2, ..., A_n), *t* là một bộ của **r**, A ⊂ {A_1, A_2, ..., A_n}. Khi đó *t*(A) ký hiệu bộ các thành phần của *t* ứng với các thuộc tính trong tập A. Nếu A là 1 thuộc tính thì *t*(A) chính là giá trị thành phần ứng với thuộc tính A.

◊ *Ví dụ*

Đây là ví dụ sẽ dùng làm cơ sở dữ liệu mẫu để mô hình hoá một công ty. Các thực thể được mô hình hoá là:

- Các *nhân viên*, ký hiệu EMP, viết tắt từ *employee*.
- Các *dự án*, ký hiệu PROJ, viết tắt từ *project*.

Đối với mỗi nhân viên chúng ta muốn theo dõi các thông tin sau:

- *Mã số nhân viên*, ký hiệu *ENO*, viết tắt từ *employee number*.
- *Tên nhân viên*, ký hiệu *ENAME*, viết tắt từ *employee name*.
- *Chức vụ*, ký hiệu *TITLE*.
- *Lương*, ký hiệu *SAL*, viết tắt từ *salary*.
- *Mã số dự án*, ký hiệu *PNO*, viết tắt từ *project number*.
- *Nhiệm vụ dự án*, ký hiệu *RESP*, viết tắt từ *responsibility*.
- *Thời gian làm việc trong dự án*, ký hiệu *DUR*, viết tắt từ *duration*.

Đối với mỗi dự án chúng ta muốn theo dõi các thông tin sau:

- *Mã số dự án*, ký hiệu *PNO*, viết tắt từ *project number*.
- *Tên dự án*, ký hiệu *PNAME*, viết tắt từ *project name*.
- *Kinh phí dự án*, ký hiệu *BUDGET*.

Các lược đồ quan hệ (relation scheme) cho cơ sở dữ liệu này có thể định nghĩa như sau:

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)

PROJ(*PNO, PNAME, BUDGET*)

Lược đồ EMP có 7 thuộc tính (attribute): *ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*.

Giá trị của *ENO* lấy tự miền chứa các mã số nhân viên, giả sử là D_1, Giá trị của *ENAME* lấy tự miền chứa các tên nhân viên hợp lệ, giả sử là D_2, ...

Đây là một thể hiện cơ sở dữ liệu mẫu của chúng ta gồm hai bảng như sau:

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E2	M.Smith	Syst.Anal.	34000	P1	Analyst	24
E2	M.Smith	Syst.Anal.	34000	P2	Analyst	6
E3	A.Lee	Mech.Eng.	27000	P3	Consultant	10
E3	A.Lee	Mech.Eng.	27000	P4	Engineer	48
E4	J.Miller	Programmer	24000	P2	Programmer	18
E5	B.Casey	Syst.Anal.	34000	P2	Manager	24

E6	L.Chu	Elect.Eng.	40000	P4	Manager	48
E7	R.David	Mech.Eng.	27000	P3	Engineer	36
E8	J.Jones	Syst.Anal.	34000	P3	Manager	40

PROJ

PNO	*PNAME*	*BUDGET*
P1	Instrumentation	150000
P2	Database Development	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Các cột của bảng tương ứng các thuộc tính của quan hệ. Các thông tin nhập theo hàng tương ứng với các bộ hay bản ghi. Dòng trên cùng biểu diễn lược đồ quan hệ của bảng.

Một giá trị của thuộc tính, chẳng hạn lương của nhân viên, thời gian tham gia dự án,... tại thời điểm nào đó có thể chưa được xác định. Khi đó có nhiều cách diễn giải khác nhau như "chưa được biết" hay "chưa áp dụng được". "Giá trị đặc biệt" thường được gọi là ***null***. Giá trị ***null*** phải khác các giá trị trong miền thuộc tính, và cũng cần phân biệt nó với giá trị *zero* (với thuộc tính kiểu số) hay giá trị *rỗng* (với thuộc tính kiểu ký tự).

2.4. Khoá

a) Siêu khoá

Cho lược đồ quan hệ $R=(A_1, A_2, ..., A_n)$ và tập con $S \subset \{ A_1, A_2, ..., A_n \}$. Tập S gọi là *siêu khoá* (superkey) của lược đồ R nếu các thuộc tính của S xác định duy nhất các bộ của mỗi quan hệ của lược đồ R, tức là với mọi quan hệ **r** của lược đồ R phải thoả mãn:

$$\forall\, t_1, t_2 \in \mathbf{r}: t_1 \neq t_2 \Rightarrow \exists\, A \in S: t_1(A) \neq t_2(A)$$

Lưu ý rằng theo định nghĩa, mỗi bộ là duy nhất nên đối với mỗi lược đồ quan hệ, *tập hợp tất cả thuộc tính là siêu khoá*. Siêu khoá là cơ sở để phân biệt 2 bộ khác nhau trong 1 quan hệ. Một lược đồ có thể có nhiều siêu khoá. Tính chất của siêu khoá là quy luật được xác định trong quá trình phân tích thiết kế cơ sở dữ liệu.

◊ *Ghi chú*

(1) Tập tất cả thuộc tính R là siêu khoá (tầm thường).

(2) $S \subset T \subset R$ & S là siêu khoá $\Rightarrow$ T là siêu khoá.

b) Khoá

Tập K các thuộc tính của lược đồ R là *khoá (key)* nếu K là siêu khoá *cực tiểu*, tức là mọi tập con thực sự của K không phải là siêu khoá.

· *Mệnh đề*: Mọi lược đồ quan hệ luôn có khóa.

Chứng minh. Mệnh đề suy ra từ sự tồn tại phần tử cực tiểu trong tập có quan hệ thứ tự.

◊ *Ví dụ*

Lược đồ PROJ có khoá là *PNO*.

Lược đồ EMP có khoá là (*ENO, PNO*).

Các thuộc tính thuộc khoá nào đó gọi là *thuộc tính khoá* hay *thuộc tính nguyên tố*.

Thuộc tính không phải thuộc tính khoá gọi là *thuộc tính không khoá*.

Mỗi quan hệ có ít nhất một khoá. Trường hợp có nhiều khoá thì gọi các khoá đó là *khoá dự tuyển* (candidate key), trong đó có một khoá là *khoá chính* (primary key).

c) Khoá ngoại

Cho lược đồ R và lược đồ Q. Tập con H các thuộc tính của R gọi là *khoá ngoại* của R *tham chiếu* đến lược đồ Q, nếu Q có khoá K gồm các thuộc tính (có thể dưới tên khác) của H thoả mãn:

Với mọi quan hệ ***r*** và ***q*** là các quan hệ của 1 cơ sở dữ liệu ứng với lược đồ R và Q ta có:

$$\forall x \in \mathbf{r}\ \exists y \in \mathbf{q}: x(H) = y(K)$$

◊ *Ví dụ*

Lược đồ EMP có khoá ngoại *PNO* tham chiếu đến khoá *PNO* của lược đồ PROJ.

Ở dạng bảng cơ sở dữ liệu mẫu của chúng ta gồm 2 bảng như sau

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
				P1		

PROJ

PNO	*PNAME*	*BUDGET*
P1		

3. Quy tắc toàn vẹn

Qui tắc toàn vẹn (*integrity rule*) là các ràng buộc đảm bảo trạng thái nhất quán của cơ sở dữ liệu. Chúng thường được diễn tả như là các ràng buộc toàn vẹn.

Có các loại qui tắc toàn vẹn sau: *Toàn vẹn thực thể* (Entity integrity), *Miền giá trị* (Domains integrity), *Toàn vẹn tham chiếu* (Referential integrity), *Thao tác bẫy* (Triggering operations).

3.1. Qui tắc toàn vẹn thực thể

Qui tắc *toàn vẹn thực thể* yêu cầu thực thể phải có khoá chính, các thuộc tính khoá phải có giá trị duy nhất và khác *null*. Qui tắc này không cho phép hai bản ghi trùng khoá.

◊ *Ví dụ*. Xét cơ sở dữ liệu

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)
PROJ(*PNO, PNAME, BUDGET*)

Qui tắc toàn vẹn thực thể ràng buộc:

- Trong lược đồ PROJ thuộc tính khoá *PNO* không thể nhận giá trị *null* và có giá trị không trùng nhau.

- Trong lược đồ EMP cặp thuộc tính khoá (*EMP, PNO*) không thể nhận giá trị *null* và có giá trị không trùng nhau.

3.2. Qui tắc miền giá trị

Đây là loại ràng buộc lên các giá trị hợp lệ của thuộc tính. *Miền giá trị* là tập hợp tất cả các loại dữ liệu và phạm vi giá trị được thuộc tính thừa nhận. Định nghĩa miền giá trị xác định các tham số đặc trưng của thuộc tính: kiểu dữ liệu (data type), độ dài (length), khuôn dạng (format), phạm vi (range), giá trị cho phép (allowable values), ý nghĩa (meaning), tính duy nhất (uniqueness), chấp nhận giá trị *null* (null support).

◊ *Ví dụ*. Xét quan hệ

PROJ(*PNO, PNAME, BUDGET*)

Các thuộc tính *PNAME* và *BUDGET* có ràng buộc miền giá trị như sau

Tên thuộc tính	:	*PNAME*	*BUDGET*
Ý nghĩa	:	Tên dự án	Kinh phí dự án
Kiểu dữ liệu	:	Ký tự (Character)	Số (numeric)
Độ dài	:	20	10
Format	:		9,999,999
Phạm vi	:		> 1000 & <10,000,000
Giá trị cho phép	:		
Duy nhất	:	Có	Không
Null support	:	Non-null	Null

3.3. Qui tắc toàn vẹn tham chiếu

Toàn vẹn tham chiếu là ràng buộc đảm bảo tính hợp lệ của sự tham chiếu của một đối tượng trong cơ sở dữ liệu (gọi là đối tượng tham chiếu) đến đối tượng khác (gọi là đối tượng được tham chiếu) trong cơ sở dữ liệu đó. Các thuộc tính tương ứng gọi *thuộc tính cặp ghép* của ràng buộc tham chiếu.

◊ *Ví dụ*. Xét các quan hệ

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)

PROJ(*PNO, PNAME, BUDGET*)

Với mỗi bộ **e** ∈ EMP phải tồn tại bộ **p**∈ PROJ sao cho

$$\mathbf{e}(PNO) = \mathbf{p}(PNO)$$

Thuộc tính *PNO* của quan hệ EMP là khoá ngoại tham chiếu đến khoá chính *PNO* của quan hệ PROJ: EMP.PNO→PROJ.PNO

Quan hệ EMP là quan hệ tham chiếu và quan hệ PROJ là quan hệ được tham chiếu, với thuộc tính cặp ghép là (*EMP.PNO*, *PROJ.PNO*).

Qui tắc toàn vẹn tham chiếu được xét đến trong khi cập nhật quan hệ tham chiếu hoặc quan hệ được tham chiếu. Ta xét các qui tắc con sau.

· ***Qui tắc chèn***: Không thể chèn hàng mới vào quan hệ tham chiếu nếu quan hệ được tham chiếu chưa có dữ liệu thuộc tính cặp ghép tương ứng.

◊ *Ví dụ*. Xét các quan hệ EMP và PROJ. Giả sử ta muốn chèn bản ghi

e = (E1, J.Doe, Elect.Eng., 40000, P5, Manager, 20)

trong đó P5 là dự án Elect.Commerce (thương mại điện tử) với kinh phí 500000 USD.

Khi đó, nếu quan hệ PROJ chưa có bản ghi

p = (P5, Elect.Commerce, 500000)

thì qui tắc chèn đảm bảo không thể chèn bản ghi **e** vào quan hệ EMP được.

Muốn chèn bản ghi **e** vào quan hệ EMP, trước hết ta phải chèn bản ghi **p** vào quan hệ PROJ.

· ***Qui tắc xoá***: Không thể xoá hàng của quan hệ được tham chiếu nếu hàng đó có dữ liệu thuộc tính cặp ghép tương ứng trong quan hệ tham chiếu.

◊ *Ví dụ*. Xét các quan hệ EMP và PROJ.

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E2	M.Smith	Syst.Anal.	34000	P1	Analyst	24
E2	M.Smith	Syst.Anal.	34000	P2	Analyst	6
...	...	...	...	...	...	...

PROJ

PNO	*PNAME*	*BUDGET*

P1	Instrumentation	150000
P2	Database Development	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Giả sử dự án P1 đã kết thúc và ta muốn xoá nó khỏi quan hệ PROJ. Tuy nhiên các bản ghi một và hai của EMP tham chiếu đến dự án P1, vì thế ta không thể xoá dự án P1 được.

Tuy nhiên, qui tắc toàn vẹn tham chiếu cho phép các phương án lựa chọn sau:

1) *Restrict*: Không cho xoá.
2) *Nullify*: Gán giá trị *null* cho thuộc tính cặp ghép của các bản ghi của quan hệ tham chiếu tham chiếu đến bản ghi bị xoá.
3) *Cascade*: Xoá tất cả các bản ghi của quan hệ tham chiếu tham chiếu đến bản ghi bị xoá.

3.4. Qui tắc thao tác bẫy

Qui tắc thao tác bẫy là qui tắc yêu cầu tính hợp pháp của dữ liệu trong các tác nghiệp cập nhật như xoá, chèn và sửa.

Thao tác bẫy có thể liên quan đến các thuộc tính của một quan hệ hoặc nhiều quan hệ. Các ràng buộc phức tạp thường được phát biểu dạng thao tác bẫy.

Một thao tác bẫy thường có các thành phần sau:

1) *Qui tắc người dùng*: là yêu cầu ngắn gọn của ràng buộc.
2) *Sự kiện*: là các thao tác xử lý dữ liệu (chèn, sửa hoặc xoá) kích hoạt thao tác bẫy.
3) *Tên quan hệ*: tên các quan hệ liên quan.
4) *Điều kiện*: là các lý do dẫn đến việc kích hoạt thao tác bẫy.
5) *Hành động*: là công việc thực thi khi thao tác bẫy được kích hoạt.

◊ *Ví dụ*. Cho quan hệ

NHANVIEN(*Manv, HoTen, NgaySinh, NgayBC, ...*).

Hiển nhiên là *NgayBC* (ngày vào biên chế) không được sớm hơn *NgaySinh*. Ta có thể đảm bảo điều kiện này bằng thao tác bẫy sau:

Qui tắc người dùng: *NgayBC* không sớm hơn *NgaySinh.*
Sự kiện: Chèn, Sửa.
Tên quan hệ: NHANVIEN
Điều kiện: *NgayBC < NgaySinh.*
Hành động: Phủ nhận thao tác cập nhật.

◊ *Ví dụ*. Xét hai quan hệ

KHACH(*Makhach, TenKhach, TaiKhoan, SoDu*)
THANHTOAN(*MaKhach, SoTien*)

Ta thấy rằng *SoTien* của THANHTOAN không thể vượt quá *SoDu* của KHACH. Ta có thể đảm bảo điều kiện này bằng thao tác bẫy sau:

Qui tắc người dùng: *SoTien* không lớn hơn *SoDu*.
Sự kiện: Chèn, Sửa.
Tên quan hệ: THANHTOAN, KHACH
Điều kiện: THANHTOAN.*SoTien* > KHACH.*SoDu*
Hành động: Phủ nhận thao tác cập nhật.

4. Các ngôn ngữ dữ liệu quan hệ

Các ngôn ngữ thao tác dữ liệu được phát triển cho mô hình quan hệ (thường gọi là *ngôn ngữ vấn tin*, query language) được chia làm hai nhóm căn bản: Các ngôn ngữ dựa trên đại số quan hệ (relational algebra) và các ngôn ngữ dựa trên phép tính quan hệ (relational calculus). Khác biệt giữa chúng là cách thức người sử dụng đưa ra câu vấn tin. Đại số quan hệ thuộc loại thủ tục (procedural), trong đó người dùng cần phải đặc tả, nhờ một số toán tử, bằng cách nào đạt được kết quả. Ngược lại phép tính quan hệ thuộc loại phi thủ tục (nonprocedural), người dùng chỉ cần đặc tả các mối liên hệ cần phải đảm bảo trong kết quả. Cả hai ngôn ngữ được Codd đưa ra năm 1970 và ông đã chứng minh rằng chúng tương đương về khả năng diễn tả.

4.1. Đại số quan hệ

Đại số quan hệ có một tập các phép toán trên các quan hệ. Chúng có nguồn gốc từ lý thuyết tập hợp (mỗi quan hệ thực chất là một tập hợp). Mỗi toán tử nhận một hoặc hai quan hệ làm toán hạng và cho ra một quan hệ mới (quan hệ kết quả), đến lượt nó quan hệ kết quả có thể dùng làm toán hạng cho một toán tử khác. Những phép toán này cho phép vấn tin và cập nhật cơ sở dữ liệu quan hệ.

Có năm phép toán đại số cơ bản và năm phép toán khác có thể định nghĩa theo các phép toán cơ bản.

Các phép toán cơ bản là: *phép chọn, phép chiếu, phép hợp, phép hiệu* và *tích Descartes*. Hai phép toán đầu thuộc loại một ngôi, và ba phép toán sau thuộc loại hai ngôi.

Các phép toán bổ sung có thể định nghĩa bởi các phép toán cơ bản là: *phép giao, phép nối, phép nối tự nhiên, phép bán nối*, và *phép thương*.

Trong thực hành đại số quan hệ được mở rộng để có thể nhóm hoặc sắp xếp kết quả và có thể thực hiện các phép gộp phần (aggregation) hoặc các phép tính số học. Một số phép toán khác như nối ngoài (outer join) cũng được hỗ trợ.

Các toán hạng của một số phép toán hai ngôi phải *ứng hợp* (*union compartible*), tức là chúng cùng bậc (cùng số thuộc tính) và các thuộc tính tương ứng có cùng miền giá trị.

a. Phép chiếu

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$. Cho S là lược đồ con của R, $S \subset R$, S có m thuộc tính ($m < n$). Chiếu của **r** lên lược đồ S, ký hiệu $\pi_S(\mathbf{r})$, được định nghĩa là quan hệ với lược đồ S gồm các m-bộ u sao cho tồn tại n-bộ $v \in \mathbf{r}$ thoả mãn

$$v(S) = u$$

tức là

$$\pi_S(\mathbf{r}) = \{\text{m-bộ } u : \exists\, v \in \mathbf{r}, v(S) = u \}$$

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C. Sau đây là ví dụ cụ thể về chiếu của **r** lên hai thuộc tính A và C.

r

A	**B**	**C**
a	b	c
d	a	f
c	b	d
c	f	d

⇒

$\pi_{A,C}(\mathbf{r})$

A	**C**
a	c
d	f
c	d

◊ *Ví dụ*: Xét quan hệ PROJ

PROJ

PNO	*PNAME*	*BUDGET*
P1	Instrumentation	150000
P2	Database Development	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Chiếu của PROJ lên các thuộc tính PNO và BUDGET có kết quả như sau

$\pi_{PNO,BUDGET}$(PROJ)

PNO	*BUDGET*
P1	150000
P2	135000
P3	250000
P4	310000

b. Phép chọn

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1,..., A_n)$ và F là biểu thức lôgic bao gồm

(i) Các toán hạng là hằng hoặc (số hiệu) thành phần,

(ii) Các phép toán so sánh : <, =, >, ≠ , ≤, ≥

(iii) Các phép toán logic : &, ∧ (và), ∨ (hoặc) , ¬ (phủ định).

Phép chọn của **r** theo biểu thức F, ký hiệu $\sigma_F(\mathbf{r})$, là quan hệ với lược đồ R gồm tất cả các bộ t trong **r** sao cho khi thay các thành phần của t vào biểu thức F thì ta có giá trị đúng. Tức là

$$\sigma_F(\mathbf{r}) = \{ t \in \mathbf{r} : F(t) = true \}$$

◊ *Ví dụ:* Cho quan hệ **r** với các thuộc tính A,B,C. Sau đây là ví dụ cụ thể về chọn của **r** theo biểu thức B=b

r

A	**B**	**C**
a	b	c

⇒

$\sigma_{B=b}(\mathbf{r})$

A	**B**	**C**
a	b	c

d	a	f		c	b	d
c	b	d			
c	f	d			

◊ *Ví dụ*: Xét quan hệ EMP

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E2	M.Smith	Syst.Anal.	34000	P1	Analyst	24
E2	M.Smith	Syst.Anal.	34000	P2	Analyst	6
E3	A.Lee	Mech.Eng.	27000	P3	Consultant	10
E3	A.Lee	Mech.Eng.	27000	P4	Engineer	48
E4	J.Miller	Programmer	24000	P2	Programmer	18
E5	B.Casey	Syst.Anal.	34000	P2	Manager	24
E6	L.Chu	Elect.Eng.	40000	P4	Manager	48
E7	R.David	Mech.Eng.	27000	P3	Engineer	36
E8	J.Jones	Syst.Anal.	34000	P3	Manager	40

Các bộ của các kỹ sư điện (electrical engineer) được biểu diễn bằng phép chọn như sau

$\sigma_{TITLE = \text{'Elect. Eng.'}}(EMP)$

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E6	L.Chu	Elect.Eng.	40000	P4	Manager	48

c. Phép hợp

Cho quan hệ **r**, **s** với lược đồ quan hệ R=(A_1,..., A_n).

Hợp của **r** và **s**, ký hiệu **r** ∪ **s** , là quan hệ với lược đồ R gồm tất cả các bộ thuộc **r** hoặc thuộc **s**.

◊ *Ví dụ*

Cho quan hệ **r** và **s** với các thuộc tính A,B,C . Sau đây là ví dụ cụ thể về hợp của **r** và **s**.

r

A	**B**	**C**
a	b	c
d	a	f
c	b	d

s

A	**B**	**C**
b	g	a
d	a	f

⇒

r ∪ **s**

A	**B**	**C**
a	b	c
d	a	f
c	b	d
b	g	a

d. Phép hiệu

Cho quan hệ **r**, **s** với lược đồ quan hệ R=(A_1,..., A_n).

Hiệu của **r** và **s**, ký hiệu **r** − **s**, là quan hệ với lược đồ R gồm tất cả các bộ thuộc **r** nhưng không thuộc **s**.

◊ *Ví dụ*

Cho quan hệ **r** và **s** với các thuộc tính A,B,C . Sau đây là ví dụ cụ thể về hiệu của **r** và **s**.

r

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

s

A	B	C
b	g	a
d	a	f

⇒ **r − s**

A	B	C
a	b	c
c	b	d

e. Tích Đề-các

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(B_1,..., B_m)$.

Tích Đề các của **r** và **s**, ký hiệu **r** × **s**, là quan hệ với lược đồ $=(A_1, ..., A_n, B_1,..., B_m)$ gồm tất cả các (n+m)-bộ, trong đó n thành phần đầu là bộ thuộc **r** và m thành phần sau là bộ thuộc **s**.

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính D,E,F. Sau đây là ví dụ cụ thể về tích Đề-các của **r** và **s**.

r

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

s

D	E	F
b	g	a
d	a	f

⇒ **r × s**

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

f. Phép giao

Cho quan hệ **r**, **s** với lược đồ quan hệ $R=(A_1,..., A_n)$.

Giao của **r** và **s**, ký hiệu **r** ∩ **s**, là quan hệ với lược đồ R gồm tất cả các bộ thuộc **r** và thuộc **s**.

◆ *Công thức*. Phép giao suy ra từ phép hiệu

$$\mathbf{r} \cap \mathbf{s} = \mathbf{r} - (\mathbf{r} - \mathbf{s}) = \mathbf{s} - (\mathbf{s} - \mathbf{r})$$

◊ *Ví dụ*

Cho quan hệ **r** và **s** với các thuộc tính A,B,C . Sau đây là ví dụ cụ thể về giao của **r** và **s**.

r

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

s

A	B	C
b	g	a
d	a	f

⇒ **r ∩ s**

A	B	C
d	a	f

g. Phép nối

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(B_1, ..., B_m)$.

Cho biểu thức lôgic F. Phép *nối* của quan hệ **r** và quan hệ **s** theo điều kiện nối F, ký hiệu

$$\mathbf{r} \underset{F}{><} \mathbf{s}$$

là quan hệ với lược đồ $=(A_1, ..., A_n, B_1, ..., B_m)$ gồm tất cả các (n+m)-bộ (t,u) thoả $t \in \mathbf{r}$, $u \in \mathbf{s}$ và khi thế các giá trị của *t* và u vào F thì ta được giá trị đúng.

Ở đây F là biểu thức lôgic gồm các toán hạng dạng $A\theta B$, trong đó A là thuộc tính của **r** và B là thuộc tính **s** và θ là phép toán so sánh.

◆ *Công thức.* Ta có thể biểu diễn

$$\mathbf{r} \underset{F}{><} \mathbf{s} = \sigma_F(\mathbf{r} \times \mathbf{s})$$

◆ *Phép đẳng nối*

Nếu các toán tử so sánh trong biểu thức F đều là phép bằng (=), thì phép nối theo F được gọi là phép *đẳng nối* (*equijoin*).

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính D,E. Sau đây là ví dụ cụ thể về phép nối và đẳng nối của **r** và **s**.

r

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

s

D	E
3	1
6	2

$\Rightarrow$ $\mathbf{r} \underset{B<D}{><} \mathbf{s}$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

r

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

s

D	E
3	1
6	2

$\Rightarrow$ $\mathbf{r} \underset{C=D}{><} \mathbf{s}$

A	B	C	D	E
1	2	3	3	1
4	5	6	6	2

◊ *Ví dụ.* Xét các quan hệ

EMP(*ENO, ENAME, TITLE*)

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

PAY(*TITLE, SAL*)

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000
Operator	15000

với ràng buộc toàn vẹn tham chiếu EMP.*TITLE*→PAY.*TITLE*.

Phép nối EMP với PAY theo EMP.*TITLE*=PAY.*TITLE* có kết quả sau

EMP >< PAY
EMP.*TITLE*=PAY.*TITLE*

ENO	*ENAME*	*EMP.TITLE*	*PAY.TITLE*	*SAL*
E1	J.Doe	Elect.Eng.	Elect.Eng.	40000
E2	M.Smith	Syst.Anal.	Syst.Anal.	34000
E3	A.Lee	Mech.Eng.	Mech.Eng.	27000
E4	J.Miller	Programmer	Programmer	24000
E5	B.Casey	Syst.Anal.	Syst.Anal.	34000
E6	L.Chu	Elect.Eng.	Elect.Eng.	40000
E7	R.David	Mech.Eng.	Mech.Eng.	27000
E8	J.Jones	Syst.Anal.	Syst.Anal.	34000

h. Phép nối tự nhiên

Nối tự nhiên giữa 2 quan hệ **r** và **s**, ký hiệu là **r** >< **s**, là phép đẳng nối trên các thuộc tính cụ thể có cùng miền giá trị. Tuy nhiên khác với đẳng nối là các thuộc tính dùng để nối tự nhiên chỉ xuất hiện một lần trong bảng kết quả.

◆ *Cách tính*.

Cho các quan hệ **r** và **s** với các cột được đặt tên theo thuộc tính. *Nối tự nhiên* **r** >< **s** được tính như sau:

(i) Tính tích Đề-các **r** × **s**.

(ii) Với mỗi thuộc tính A có cả trong **r** và **s**, chọn các bộ trong **r** × **s** có R.A = S.A, trong đó R.A (S.A) là tên cột của R × S tương ứng với cột A của R (S).

(iii) Với mỗi thuộc tính A như trên ta loại bỏ cột S.A.

Một cách hình thức, giả sử A_1, A_2, ... , A_k là tên các thuộc tính dùng chung cho cả R và S. Gọi i_1, i_2, ... , i_m là danh sách các thành phần của R × S , trừ các thuộc tính $S.A_1$, $S.A_2$, ..., $S.A_k$. Khi đó ta có thể biểu diễn nối tự nhiên bằng công thức sau.

◆ *Công thức*

$$\mathbf{r} >< \mathbf{s} = \pi_{i_1, i_2, ..., i_m} \sigma_{R.A_1=S.A_1 \wedge ... \wedge R.A_k=S.A_k} (\mathbf{r} \times \mathbf{s})$$

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính B,C,D. Sau đây là ví dụ cụ thể về phép nối tự nhiên của **r** và **s**.

r			**s**				**r** >< **s**			
A	**B**	**C**	**B**	**C**	**D**	⇒	**A**	**B**	**C**	**D**
a	b	c	b	c	d		a	b	c	d
d	b	c	b	c	e		a	b	c	e
b	b	f	a	d	b		d	b	c	d

c	a	d		d	b	c	e
				c	a	d	b

◊ *Ví dụ.* Các quan hệ

EMP(*ENO, ENAME, TITLE*)

PAY(*TITLE, SAL*)

cho ở ví dụ trước, có thuộc tính chung là *TITLE*.

Nối tự nhiên của hai quan hệ cho ở bảng sau

EMP >< PAY

ENO	*ENAME*	*TITLE*	*SAL*
E1	J.Doe	Elect.Eng.	40000
E2	M.Smith	Syst.Anal.	34000
E3	A.Lee	Mech.Eng.	27000
E4	J.Miller	Programmer	24000
E5	B.Casey	Syst.Anal.	34000
E6	L.Chu	Elect.Eng.	40000
E7	R.David	Mech.Eng.	27000
E8	J.Jones	Syst.Anal.	34000

i. Phép bán nối

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1,..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(B_1, ..., B_m)$.

Ký hiệu θ là toán tử so sánh (<, =, >, ≠ , ≤, ≥). Cho F là biểu thức lôgic gồm các toán hạng dạng AθB, trong đó A là thuộc tính của **r** và B là thuộc tính **s**.

Phép *bán nối* của quan hệ **r** và quan hệ **s** theo điều kiện nối F, ký hiệu

$$\mathbf{r} \underset{F}{><} \mathbf{s}$$

là quan hệ với lược đồ R gồm các bộ của **r** có tham gia vào nối của **r** và **s** theo F**.**

◆ *Công thức.* Ta có thể biểu diễn

$$\mathbf{r} \underset{F}{><} \mathbf{s} = \pi_R(\mathbf{r} \underset{F}{><} \mathbf{s})$$

Ưu điểm của phép bán nối là giảm số lượng các bộ cần xử lý để thực hiện nối. Trong hệ cơ sở dữ liệu phân tán điều này có ý nghĩa rất quan trọng vì nó làm giảm số lượng dữ liệu cần truyền giữa các vị trí để ước lượng câu vấn tin.

◆*Bán nối tự nhiên*

$$\mathbf{r} >< \mathbf{s} = \pi_R(\mathbf{r} >< \mathbf{s})$$

◊ *Ví dụ.* Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính D,E. Sau đây là ví dụ cụ thể về phép bán nối của **r** và **s**.

r			**s**			$\mathbf{r} \underset{B<D}{><} \mathbf{s}$		
A	**B**	**C**	**D**	**E**	⇒	**A**	**B**	**C**
1	2	3	3	1		1	2	3

4	5	6		6	2		4	5	6
7	8	9							

r **s** **r** $\overset{A=E}{><}$ **s**

A	**B**	**C**		**D**	**E**	⇒	**A**	**B**	**C**
1	2	3		3	1		1	2	3
4	5	6		6	2				
7	8	9							

◊ *Ví dụ*. Xét các quan hệ

EMP(*ENO, ENAME, TITLE*)

PAY(*TITLE, SAL*)

ở ví dụ trước. Phép bán nối của hai quan hệ theo EMP.*TITLE*=PAY.*TITLE* có kết quả sau

EMP >< PAY

EMP.*TITLE*=PAY.*TITLE*

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

PAY >< EMP

EMP.*TITLE*=PAY.*TITLE*

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000

j. Phép chia (thương)

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(A_1, ..., A_m)$, trong đó $m < n$.

Ta định nghĩa *phép chia* **r** ÷ **s** là quan hệ với lược đồ $(A_{m+1}, ..., A_n)$ gồm tất cả (n-m)-bộ *v* sao cho với mọi m-bộ *u* thuộc **s**, bộ (*u*,*v*) thuộc **r**.

Để lập công thức cho phép chia ta ký hiệu

$$\mathbf{t} = \pi_{A_{m+1},...,A_n}(r)$$

Khi đó

$$(\mathbf{s} \times \mathbf{t}) - \mathbf{r}$$

là tập hợp các n-bộ không thuộc **r**, tạo ra bằng cách lấy các phần tử của **s** kết hợp với n-m thành phần của các phần tử thuộc **r**. Đặt

$$\mathbf{q} = \pi_{A_{m+1},\ldots,A_n}\left((s \times t) - r\right)$$

Như vậy **q** là tập tất cả các (n−m)-bộ *v* gồm n−m thành phần cuối của các phần tử của **r** và tồn tại phần tử $u \in \mathbf{s}$ sao cho bộ (*u*,*v*) không thuộc **r**. Suy ra

◆ *Công thức*

$$\mathbf{r} \div \mathbf{s} = \mathbf{t} - \pi_{A_{m+1},\ldots,A_n}\left((s \times t) - r\right)$$

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C,D và **s** với các thuộc tính C,D. Sau đây là ví dụ cụ thể về phép chia của **r** cho **s**.

r				**s**			**r ÷ s**	
A	**B**	**C**	**D**	**C**	**D**	⇒	**A**	**B**
a	b	c	d	c	d		a	b
a	b	e	f	e	f		e	d
b	c	e	f					
e	d	c	d					
e	d	e	f					
a	b	d	e					

◊ *Ví dụ*. Xét các quan hệ sau:

ASG'

ENO	*PNO*	*PNAME*	*BUDGET*
E1	P1	Instrumentation	150000
E2	P1	Instrumentation	150000
E2	P2	Database Deve.	135000
E3	P3	CAD/CAM	250000
E3	P4	Maintenance	310000
E4	P2	Database Deve.	135000
E5	P2	Database Deve.	135000
E6	P4	Maintenance	310000
E7	P3	CAD/CAM	250000
E8	P3	CAD/CAM	250000

PROJ'

PNO	*PNAME*	*BUDGET*
P3	CAD/CAM	250000
P4	Maintenance	310000

Để tìm mã số nhân viên của những nhân viên được phân vào tất cả dự án trong PROJ', ta phải thực hiện phép chia ASG' cho PROJ' và được kết quả là

ASG' ÷ PROJ'

ENO
E3

· ***Các chương trình đại số quan hệ***

Vì tất cả các phép toán đại số đều nhận quan hệ làm đối biến và sinh ra các quan hệ kết quả, chúng ta có thể lồng ghép các phép toán này bằng các dấu ngoặc đơn và sinh ra các *chương trình đại số quan hệ*.

Dưới đây là một số ví dụ minh hoạ sử dụng các quan hệ

EMP(*ENO, ENAME, TITLE*)
PAY(*TITLE, SAL*)
PROJ(*PNO, PNAME, BUDGET*)
ASG(*ENO, PNO, RESP, DUR*)

◊ *Ví dụ*. Sau đây là chương trình tìm tên tất cả nhân viên đang làm việc cho dự án CAD/CAM

$$\pi_{ENAME}(((\sigma_{PNAME='CAD/CAM'}(PROJ)) >< ASG) >< EMP)$$

Thứ tự thực hiện như sau: thực hiện phép chọn trên PROJ, sau đó nối tự nhiên với ASG, theo sau là nối tự nhiên với EMP, và cuối cùng là chiếu lên ENAME.

Một chương trình tương đương nhưng kích thước quan hệ trung gian nhỏ hơn là

$$\pi_{ENAME}(EMP >< (\pi_{ENO}(ASG >< (\sigma_{PNAME='CAD/CAM'}(PROJ))))$$

◊ *Ví dụ*. Sau đây là chương trình tăng lương tất cả các lập trình viên (programmer) lên 25000 USD.

$$(PAY - (\sigma_{TITLE='Programmer'}(PAY)) \cup (<'Programmer',25000>)$$

4.2. Đại số quan hệ

Trong các ngôn ngữ dựa trên phép tính quan hệ, thay vì xác định xem phải làm thế nào để thu được kết quả, chúng ta sẽ xác định xem kết quả là gì bằng cách đưa ra mối liên hệ được giả sử là đúng đối với kết quả.

Ngôn ngữ phép tính quan hệ được phân làm 2 nhóm: *phép tính quan hệ bộ (tuple relational calculus)* và *phép tính quan hệ miền (domain relational calculus)*. Sự khác biệt giữa chúng là ở các biến nguyên thuỷ được dùng khi xác định các câu vấn tin.

Ngôn ngữ phép tính quan hệ có cơ sở lý thuyết vững chắc bởi vì chúng xây dựng trên logic vị từ bậc nhất. Ngữ nghĩa được gán cho các công thức bằng cách diễn giải chúng như các phán đoán trên cơ sở dữ liệu. Một cơ sở dữ liệu quan hệ có thể xem như tập các bộ hoặc tập các miền. Phép tính quan hệ bộ diễn giải các biến trong công thức như một bộ của quan hệ, còn phép tính quan hệ miền diễn giải biến như giá trị của miền.

a) Phép tính quan hệ bộ (Codd 1970)

Biến nguyên thuỷ dùng trong phép tính quan hệ bộ là *biến bộ* (tuple variable), biểu thị một bộ của quan hệ. Nói cách khác biến này biến thiên trên các bộ của quan hệ.

Trong phép tính quan hệ bộ, câu vấn tin được đặc tả là

$$\{t \mid F(t)\}$$

trong đó t là biến bộ và F là công thức chỉnh dạng. Công thức nguyên tử có hai dạng:

(1) *Biểu thức kiểu phần tử biến bộ.*

Nếu t là một biến bộ biến thiên trên các bộ của một quan hệ R, biểu thức "*bộ t thuộc quan hệ R*" là công thức nguyên tử, và được viết là $R.t$ hoặc $R(t)$.

(2) *Điều kiện.*

Loại công thức nguyên tử này có thể định nghĩa như sau:

(i) $s[A]\ \theta\ t[B]$, trong đó s và t là các biến bộ và A và B là các thành phần tương ứng của s và t , θ là một trong các toán tử so sánh $<, >, =, \leq, \geq$ và $\neq$.

(ii) $s[A]\ \theta\ c$, trong đó s , A và θ định nghĩa giống như trên và c là hằng.

Hiện có nhiều ngôn ngữ dựa trên phép tính quan hệ bộ, và ngôn ngữ thông dụng nhất là SQL và QUEL. SQL hiện là chuẩn quốc tế (duy nhất) với các phiên bản chuẩn hoá đã được đưa ra năm 1986 (SQL1), năm 1992 (SQL2) và năm 1998 (SQL3).

· ***Ngôn ngữ truy vấn SQL***

SQL thuộc loại *ngôn ngữ thế hệ thứ tư (4GL)* được nghiên cứu nhiều năm và trở thành tiêu chuẩn quốc tế về kiểm soát dữ liệu. SQL kế thừa *tính phi thủ tục* của 4GL: Xử lý đồng thời hàng loạt câu lệnh. Người dùng chỉ cần nêu ra yêu cầu về dữ liệu mà không cần biết máy tính xử lý bên trong như thế nào. Người dùng có thể truy xuất nhanh chóng với những CSDL lớn, yêu cầu những xử lý phức tạp tinh vi mà không cần lập trình.

SQL là *ngôn ngữ có cấu trúc*. Trong câu lệnh của SQL có một số mệnh đề tuân theo những cú pháp riêng của nó. Có 4 loại lệnh trong SQL :

- Các lệnh truy vấn dữ liệu.
- Các lệnh định nghĩa dữ liệu (DDL).
- Các lệnh xử lý cập nhật dữ liệu (DML).
- Các lệnh kiểm soát dữ liệu.

◆ Các lệnh truy vấn dữ liệu, gọi là câu vấn tin có cú pháp tổng quát như sau

```
SELECT [DISTINCT] <biểu thức 1> AS <tên 1>] [,...] | *
FROM  <bảng 1> [<bí danh 1>] [,...]
[INTO <dbf đích>]
[WHERE  <điều kiện nối >  [AND | OR <điều kiện lọc>]]
[GROUP BY  <cột nhóm 1> [,...]
   [HAVING <điều kiện nhóm>]]
[ORDER BY <biểu thức sắp xếp 1> [ASC | DESC] [,...]]
[UNION | INTERSECT | MINUS  < câu truy vấn khác>]
```

Dưới đây là một số ví dụ minh họa sử dụng các quan hệ

EMP(*ENO, ENAME, TITLE*)
PAY(*TITLE, SAL*)
PROJ(*PNO, PNAME, BUDGET*)
ASG(*ENO, PNO, RESP, DUR*)

◊ *Ví dụ*. Tìm tên tất cả nhân viên đang làm việc cho dự án CAD/CAM

```
Select   EMP.ENAME
From     EMP, ASG, PROJ
Where   (EMP.ENO = ASG.ENO)
         AND (ASG.PNO = PROJ.PNO)
         AND (PROJ.PNAME = "CAD/CAM")
```

◊ *Ví dụ*. Tìm tên tất cả nhân viên đang quản lý dự án (Manager)

```
SELECT      ENAME
FROM        EMP, ASG
WHERE       (EMP.ENO = ASG.ENO)
        AND    (RESP = "Manager")
```

◊ *Ví dụ*. Tìm tên tất cả nhân viên đang làm việc trong dự án P3 và P4. (bài tập)

◆ Các lệnh cập nhật dữ liệu gồm có lệnh UPDATE (hiệu chỉnh), INSERT (thêm) và DELETE (xoá).

◊ *Ví dụ.* Tăng lương các lập trình viên (programmer) lên 25000 USD.

```
UPDATE  PAY
SET     SAL = 25000
WHERE   PAY.TITLE = "Programmer"
```

◊ *Ví dụ.* Thêm nhân viên mới vào EMP

```
INSERT INTO EMP
   VALUE ('E10', 'John Smith', 'Programmer')
```

◊ *Ví dụ.* Xoá dự án 'P1'

```
DELETE FROM PROJ
   WHERE  PNO = 'P1'
```

b) Phép tính quan hệ miền (Lacroix, Pirotte 1977)

Biến nguyên thuỷ dùng trong phép tính quan hệ miền là *biến miền* (*domain variable*), xác định một thành phần của bộ biến thiên trong tập giá trị của miền. Nói cách khác, miền xác định của biến miền bao gồm các miền trên đó quan hệ được định nghĩa. Câu vấn tin có dạng sau:

$$x_1, \ldots, x_n \mid F(x_1, \ldots, x_n)$$

trong đó F là công thức chỉnh dạng còn $x_1, \ldots, x_n$ là các biến tự do.

Thành công của ngôn ngữ phép tính quan hệ miền chủ yếu do QBE (Zloof, 1977) đem lại. Đây là ứng dụng kiểu trực quan của phép tính miền. QBE (Query by example) được thiết kế dành cho kiểu làm việc tương tác từ thiết bị đầu cuối trực quan và thân thiện.

Khái niệm cơ bản là *example*: người sử dụng đưa ra các câu vấn tin bằng cách cung cấp một example có thể có của câu trả lời. Hành động gõ tên quan hệ sẽ kích hoạt việc hiển thị các lược đồ của chúng lên màn hình. Sau đó bằng cách cung cấp các từ khoá trong các cột (miền), người dùng đặc tả câu vấn tin.

Chẳng hạn các thuộc tính của quan hệ chiếu được cho bằng từ P (Project).

Theo mặc định tất cả các câu vấn tin đều là kiểu truy xuất. Câu vấn tin cập nhật đòi hỏi phải có đặc tả U dưới tên quan hệ cần cập nhật.

◊ *Ví dụ.* Tìm tên tất cả nhân viên đang làm việc cho dự án CAD/CAM

EMP	*ENO*	*ENAME*	*TITLE*
	E2	P	

ASG	*ENO*	*PNO*	*RESP*	*DUR*
	E2	P3		

PROJ	*PNO*	*PNAME*	*BUDGET*

P3	CAD/CAM	

◊ *Ví dụ.* Tăng lương các lập trình viên (programmer) lên 25000 USD.

PAY	TITLE	SAL
	Programmer	U.25000

5. Thiết kế cơ sở dữ liệu quan hệ

5.1. Dư thừa dữ liệu

Khi thiết kế cơ sở dữ liệu quan hệ ta thường đứng trước vấn đề lựa chọn giữa các lược đồ quan hệ: lược đồ nào tốt hơn ? Tại sao ? Mục này sẽ nghiên cứu một số tiêu chuẩn đánh giá lược đồ quan hệ và các thuật toán giúp chúng ta xây dựng được lược đồ cơ sở dữ liệu quan hệ có cấu trúc tốt.

Có thể nói tổng quát một lược đồ quan hệ có *cấu trúc tốt* là lược đồ không chứa đựng sự *dư thừa dữ liệu*, tức là là sự trùng lặp thông tin trong cơ sở dữ liệu.

5.1.1. Sự dư thừa dữ liệu

Dư thừa dữ liệu là sự trùng lặp thông tin trong cơ sở dữ liệu.

◊ *Ví dụ*

Xét quan hệ EMP(*ENO*, *ENAME, TITLE, SAL*, *PNO*, *RESP*, *DUR*). Nếu một nhân viên tham gia trong nhiều dự án, thì các dữ liệu như *ENAME, TITLE, SAL* phải lặp lại nhiều lần và kéo theo dư thừa dữ liệu.

Ngoài việc gây lãng phí dung lượng lưu trữ, sự dư thừa dữ liệu có thể gây ra những hậu quả nghiêm trọng đối với dữ liệu khi người dùng cập nhật dữ liệu làm cho dữ liệu không tương thích, bất định hoặc mất mát. Các sự cố như vậy gọi là những *dị thường*.

5.1.2. Các dị thường cập nhật dữ liệu

Ta sẽ minh hoạ các dị thường bằng các lược đồ

EMP(*ENO*, *ENAME, TITLE, SAL*, *PNO*, *RESP, DUR*)
PROJ(*PNO*, *PNAME, BUDGET*)

a. Dị thường do dữ liệu lặp: Một số thông tin có thể được lặp lại một cách vô ích.

◊ *Ví dụ*: Trong quan hệ EMP tên (*ENAME*), chức vụ (*TITLE*), và lương (*SAL*) của nhân viên được lặp lại trong mỗi dự án mà họ tham gia. Điều này rõ ràng là làm lãng phí chỗ lưu trữ và đối nghịch với các nguyên lý của cơ sở dữ liệu.

b. Dị thường chèn bộ: Không thể chèn bộ mới vào quan hệ, nếu không có đầy đủ dữ liệu.

◊ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên mới được nhận vào công ty và chưa được phân công vào dự án nào cả. Khi đó chúng ta không thể nhập các thông tin về tên, chức vụ, lương của nhân viên này vào quan hệ, vì khoá của EMP là (*ENO, PNO*).

c. Dị thường xoá bộ: Trường hợp này ngược với dị thường chèn bộ. Việc xoá bộ có thể kéo theo mất thông tin.

◊ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên làm việc trong một dự án duy nhất. Khi dự án chấm dứt, chúng ta không thể xoá thông tin về dự án đó trong EMP được, vì nếu làm thế ta sẽ mất luôn thông tin về nhân viên đó.

d. Dị thường sửa bộ: Việc sửa đổi dữ liệu dư thừa có thể dẫn đến sự không tương thích dữ liệu.

◊ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên làm việc trong nhiều dự án. Khi có sự thay đổi về lương, rất nhiều bộ phải cập nhật sự thay đổi này. Điều đó gây lãng phí thời gian công sức và là nguy cơ gây ra sự không thống nhất dữ liệu.

Trong các ví dụ trên ta thấy tác hại của sự dư thừa dữ liệu và sự cần thiết phải loại bỏ chúng khỏi các lược đồ quan hệ. Quá trình từng bước thay thế một lược đồ quan hệ bằng các tập lược đồ quan hệ đơn giản và chuẩn tắc hơn gọi là ***chuẩn hoá***. Mục đích của chuẩn hoá là loại bỏ các dị thường (hoặc các khía cạnh không mong muốn khác) để có những quan hệ tốt hơn.

Cơ sở lý thuyết của việc thiết kế lược đồ cơ sở dữ liệu quan hệ tốt là khái niệm *phụ thuộc dữ liệu*. Phụ thuộc dữ liệu biểu diễn các quan hệ nhân quả giữa các thuộc tính trong quan hệ. Ví dụ trong bảng EMP, thuộc tính SAL phụ thuộc vào thuộc tính ENO, vì mỗi nhân viên chỉ có một lương duy nhất.

Cũng dựa trên khái niệm phụ thuộc dữ liệu người ta định nghĩa các *dạng chuẩn* của lược đồ dữ liệu quan hệ. Mỗi dạng chuẩn đáp ứng một yêu cầu nhất định đối với lược đồ quan hệ.

Quá trình biến đổi một lược đồ thành lược đồ *tương đương* (bảo toàn thông tin và phụ thuộc dữ liệu) thoả mãn dạng chuẩn gọi là quá trình *chuẩn hoá lược đồ quan hệ*.

Khái niệm phụ thuộc dữ liệu sẽ được nghiên cứu chi tiết ở phần sau.

5.2. Cấu trúc phụ thuộc dữ liệu

Có ba dạng phụ thuộc dữ liệu, *phụ thuộc hàm* (*functional dependancy*- FD) , *phụ thuộc đa trị* (*multivalued dependancy* - MVD) và *phụ thuộc chiếu nối* (*projection-join dependancy* - PJD)

a. Phụ thuộc hàm

Cho lược đồ quan hệ R=(A_1, A_2, ..., A_n) và X, Y là các tập con của {A_1, A_2, ..., A_n}. Ta nói rằng X *xác định hàm* Y hay Y *phụ thuộc hàm* X, ký hiệu $X \rightarrow Y$, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

$$\forall u, v \in \mathbf{r} : u(X) = v(X) \Rightarrow u(Y) = v(Y)$$

Cần nhấn mạnh rằng tính chất phụ thuộc hàm phải thoả với mọi quan hệ **r** của lược đồ R. Ta không thể chỉ xét một quan hệ đặc biệt (quan hệ rỗng chẳng hạn) rồi quy nạp cho toàn lược đồ. Nhưng ta có thể phủ nhận phụ thuộc hàm qua một quan hệ cụ thể nào đó.

Phụ thuộc hàm $X \rightarrow Y$ gọi là phụ thuộc hàm *tầm thường* nếu $Y \subset X$ (hiển nhiên là nếu $Y \subset X$ thì theo định nghĩa ta có $X \rightarrow Y$).

Phụ thuộc hàm $X \rightarrow Y$ gọi là phụ thuộc hàm *nguyên tố* nếu không có tập con thực sự $Z \subset X$ thoả $Z \rightarrow Y$.

Tập thuộc tính $K \subset R$ gọi là *khoá* nếu nó xác định hàm tất cả các thuộc tính và $K \rightarrow R$ là phụ thuộc hàm nguyên tố.

◊ *Ví dụ*: Xét quan hệ PROJ. Ta có thể chấp nhận rằng mỗi dự án có tên và kinh phí xác định. Vậy có thể khẳng định

$$PNO \rightarrow (PNAME, BUDGET)$$

Trong quan hệ EMP ta có

$$(ENO, PNO) \rightarrow (ENAME, TITLE, SAL, RESP, DUR)$$
$$ENO \rightarrow (ENAME, TITLE, SAL)$$

Hoàn toàn hợp lý khi chúng ta khẳng định rằng lương của mỗi chức vụ là cố định, do đó sẽ tồn tại phụ thuộc hàm

$$TITLE \rightarrow SAL$$

b. Phụ thuộc đa trị

Cho lược đồ quan hệ $R=(A_1, A_2, ..., A_n)$ và X, Y là các tập con của $\{A_1, A_2, ..., A_n\}$. Ta nói rằng X *xác định đa trị* Y hay Y *phụ thuộc đa trị* vào X, ký hiệu $X \rightarrow\rightarrow Y$, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

Ứng với mỗi giá trị của miền giá trị các thuộc tính trong X, có một tập giá trị các thuộc tính trong Y liên quan và tập này độc lập với các thuộc tính trong $Z=R\backslash(X \cup Y)$, tức là:

$$\forall x \in D(X)\ \forall y, y' \in D(Y)\ \forall z, z' \in D(Z): (x,y,z), (x,y',z') \in \mathbf{r} \Rightarrow (x,y,z'), (x,y',z) \in \mathbf{r}$$

với D(X), D(Y) và D(Z) là miền giá trị của X, Y và Z.

· Chú ý rằng *phụ thuộc hàm là trường hợp riêng của phụ thuộc đa trị*, tức là

$$X \rightarrow Y \quad \Rightarrow \quad X \rightarrow\rightarrow Y$$

Thật vậy, nếu

$$(x,y,z), (x, y', z') \in \mathbf{r} \text{ và } X \rightarrow Y$$

thì y=y', và kéo theo

$$(x, y, z'), (x, y', z) \in \mathbf{r}$$

◊ *Ví dụ*

Trở lại ví dụ đang xét. Giả sử ta muốn duy trì thông tin về tập nhân viên và về tập dự án có liên quan đến công ty cũng như về chi nhánh (*PLACE*) thực hiện dự án. Yêu cầu này có thể được thực hiện bằng cách định nghĩa quan hệ

SKILL**(***ENO, PNO, PLACE***)**

Ta giả sử (có thể không thực tế cho lắm) (1) mỗi nhân viên đều có thể làm việc cho mọi dự án, (2) mỗi nhân viên đều có thể làm việc tại mọi chi nhánh và (3) mỗi dự án đều có thể được thực hiện tại bất kỳ chi nhánh nào. Một quan hệ mẫu thoả các điều kiện này cho ở bảng sau:

SKILL

ENO	*PNO*	*PLACE*
E1	P1	Toronto
E1	P1	New York
E1	P1	London
E1	P2	Toronto
E1	P2	New York
E1	P2	London
E2	P1	Toronto
E2	P1	New York
E2	P1	London
E2	P2	Toronto
E2	P2	New York
E2	P2	London

Chú ý rằng không có phụ thuộc hàm không tầm thường nào trong quan hệ SKILL; tất cả thuộc tính là thuộc tính khoá. Quan hệ SKILL có hai phụ thuộc đa trị

$$ENO \rightarrow\rightarrow PNO$$
$$ENO \rightarrow\rightarrow PLACE$$

c. Phụ thuộc chiếu-nối

Cho lược đồ quan hệ $R=(A_1, A_2, ..., A_n)$ và $R_1, R_2, ..., R_k$ là các tập con của $\{A_1, A_2, ..., A_n\}$. Ta nói rằng $\{R_1, R_2, ..., R_k\}$ xác định một *phụ thuộc chiếu-nối* của R, nếu mọi quan hệ **r** của R là nối tự nhiên của các chiếu của nó lên $R_1, R_2, ..., R_k$, tức là

$$\mathbf{r} = \pi_{R1}(\mathbf{r}) >< \pi_{R2}(\mathbf{r}) >< \ ... >< \pi_{Rk}(\mathbf{r})$$

· Chú ý rằng *phụ thuộc đa trị là trường hợp riêng của phụ thuộc chiếu-nối*, tức là

$$X \rightarrow\rightarrow Y \quad \Rightarrow \quad \{X \cup Y, X \cup Z\}$$ xác định một phụ thuộc chiếu-nối,

trong đó $Z=R\backslash(X \cup Y)$.

Thật vậy, cho quan hệ **r** trên lược đồ R thỏa phụ thuộc đa trị X→→Y. Hiển nhiên $\pi_{XY}(\mathbf{r}) >< \pi_{XZ}(\mathbf{r}) \supset \mathbf{r}$. Ta chỉ cần chứng minh $\pi_{XY}(\mathbf{r}) >< \pi_{XZ}(\mathbf{r}) \subset \mathbf{r}$. Cho $(x,y) \in \pi_{XY}(\mathbf{r})$ và $(x,z) \in \pi_{XZ}(\mathbf{r})$. Khi đó tồn tại z' và y' thỏa $(x,y,z') \in \mathbf{r}$ và $(x,y',z) \in \mathbf{r}$ (theo định nghĩa phép chiếu), kéo theo $(x,y,z) \in \mathbf{r}$ (vì X→→Y). Từ đó suy ra $\pi_{XY}(\mathbf{r}) >< \pi_{XZ}(\mathbf{r}) \subset \mathbf{r}$.

◊ *Ví dụ*

Xét quan hệ SKILL(*ENO, PNO, PLACE*) ở trên. Do *ENO*→→*PNO*, nên {(*ENO,PNO*), (*ENO,PLACE*)} xác định phụ thuộc chiếu nối.
Ta có

$\pi_{ENO,PNO}$(SKILL)

ENO	*PNO*
E1	P1
E1	P2
E2	P1
E2	P2

$\pi_{ENO,PLACE}$(SKILL)

ENO	*PLACE*
E1	Toronto
E1	New York
E1	London
E2	Toronto
E2	New York
E2	London

Suy ra

$$\text{SKILL} = \pi_{ENO,PNO}(\text{SKILL}) >< \pi_{ENO,PLACE}(\text{SKILL})$$

5.3. Phụ thuộc đa trị

Trong phần này chúng ta sẽ nghiên cứu sâu hơn về phụ thuộc đa trị, mối quan hệ giữa phụ thuộc đa trị và phụ thuộc hàm.

5.3.1. Định nghĩa

Ta nhắc lại định nghĩa phụ thuộc đa trị.

Cho lược đồ quan hệ $R(A_1, A_2, ..., A_n)$ và X, Y là các tập con của $\{A_1, A_2, ..., A_n\}$. Ta nói rằng X *xác định đa trị* Y hay Y *phụ thuộc đa trị* vào X, ký hiệu X→→Y, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

Ứng với mỗi giá trị của miền giá trị các thuộc tính trong X, có một tập giá trị các thuộc tính trong Y liên quan và tập này độc lập với các thuộc tính trong Z=R\(X∪Y), tức là:

$$\forall x \in D(X) \forall y,y' \in D(Y)\ \forall z, z' \in D(Z): (x,y,z), (x,y',z') \in \mathbf{r} \Rightarrow (x,y,z'), (x,y',z) \in \mathbf{r}$$

◊ *Ví dụ*

Xét lược đồ CTHRSG=(*C,T,H,R,S,G*), trong đó C (Course) là môn học, T (Teacher) là giáo viên, H (Hour) là tiết học, R (Room) là phòng học, S (Student) là sinh viên và G (Grade) là điểm số. Một quan hệ mẫu cho ở bảng sau

C	*T*	*H*	*R*	*S*	*G*
CS101	Trần	M9	222	Hùng	9
CS101	Trần	W9	333	Hùng	9
CS101	Trần	F9	222	Hùng	9
CS101	Trần	M9	222	Dũng	7
CS101	Trần	W9	333	Dũng	7
CS101	Trần	F9	222	Dũng	7

Trong ví dụ đơn giản này ta thấy *môn học* có nhiều giờ học, trong các phòng học khác nhau, trong tuần. Mỗi *sinh viên* có một bản ghi cho mỗi giờ học và điểm của sinh viên cũng được lặp lại tương ứng.

Như vậy ta suy ra phụ thuộc đa trị $C \rightarrow\rightarrow H,R$, tức là sẽ có tập *giờ-phòng* ứng với mỗi môn học, độc lập với các thuộc tính khác. Ví dụ, cho hai bản ghi

t = (CS101, Trần, M9, 222, Hùng, 9)
s = (CS101, Trần, W9, 333, Dũng, 7)

chúng ta chờ đợi rằng có thể hoán chuyển (M9, 222) của *t* với (W9, 333) của *s* để nhận được các bộ sau

u = (CS101, Trần, M9, 222, Dũng, 7)
v = (CS101, Trần, W9, 333, Hùng, 9)

Và ta thấy *u, v* cũng có mặt trong quan hệ trên.

Cần nhấn mạnh rằng, $C \rightarrow\rightarrow H,R$ đúng bởi vì với mỗi môn học *c*, nếu tồn tại các bộ

(c, h1, r1, t1, s1, g1)

và

(c, h2, r2, t2, s2, g2)

thì cũng sẽ tồn tại

(c, h1, r1, t2, s2, g2) và (c, h2, r2, t1, s1, g1).

Lưu ý rằng $C \rightarrow\rightarrow H$ và $C \rightarrow\rightarrow R$ không đúng, bởi vì, nếu ngược lại, từ *t* và *s* suy ra bản ghi

(CS101, Trần, M9, 333, Dũng, 7)

phải có trong quan hệ trên.

Tồn tại nhiều phụ thuộc đa trị khác như $C \rightarrow\rightarrow S,G$ và $H,R \rightarrow\rightarrow S,G$ (bài tập).

5.3.2. Các tiên đề phụ thuộc đa trị và phụ thuộc hàm

Ta sẽ trình bày tập hợp đầy đủ các tiên đề phụ thuộc hàm và phụ thuộc đa trị trên tập thuộc tính U. Các tiên đề Armstrong được nhắc lại vì tính hệ thống.

(A1) ***Quy tắc phản xạ phụ thuộc hàm***:

$$Y \subset X \subset U \Rightarrow X \rightarrow Y$$

(A2) ***Quy tắc tăng trưởng phụ thuộc hàm*:**

$$X \rightarrow Y \ \& \ Z \subset U \Rightarrow X \cup Z \rightarrow Y \cup Z$$

(A3) ***Quy tắc bắc cầu phụ thuộc hàm***:

$$X \rightarrow Y \ \& \ Y \rightarrow Z \Rightarrow X \rightarrow Z$$

(M1) ***Quy tắc bù phụ thuộc đa trị***:

$$X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow (U \setminus (X \cup Y))$$

(M2) ***Quy tắc tăng trưởng phụ thuộc đa trị***:

$$X \rightarrow\rightarrow Y \ \& \ V \subset W \Rightarrow X \cup W \rightarrow\rightarrow Y \cup V$$

(M3) ***Quy tắc bắc cầu phụ thuộc đa trị***:

$$X \rightarrow\rightarrow Y \ \& \ Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Z \setminus Y)$$

(M4) ***Quy tắc phụ thuộc hàm-đa trị***:

$$X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$$

(M5) ***Quy tắc phụ thuộc đa trị-hàm***:

$$X \rightarrow\rightarrow Y \ \& \ Z \subset Y \ \& \ W \rightarrow Z \ \& \ W \cap Y = \varnothing \Rightarrow X \rightarrow Z$$

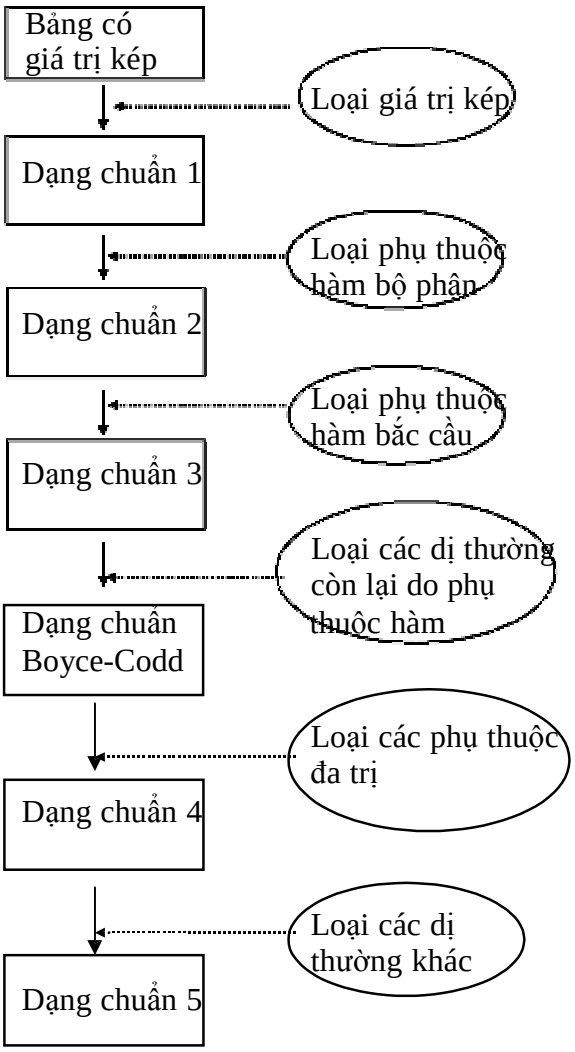
· ***Định lý*:** Các tiên đề A1-A3 và M1-M5 là đúng và đủ cho phụ thuộc hàm và phụ thuộc đa trị. Tức là, nếu D là tập hợp các phụ thuộc hàm và phụ thuộc đa trị trên tập thuộc tính U, và D^+ là tập hợp các phụ thuộc hàm và phụ thuộc đa trị suy diễn lôgic từ D (theo nghĩa mỗi quan hệ thoả D thì cũng thoả D^+), thì D^+ chính là tập hợp các phụ thuộc hàm và phụ thuộc đa trị suy ra từ D bằng các tiên đề trên.

Chứng minh. (công nhận)

5.4. Chuẩn hoá lược đồ quan hệ

Chúng ta đã chỉ ra rằng sự dư thừa dữ liệu là nguyên nhân của các dị thường khi cập nhật dữ liệu dẫn đến sự không tương thích dữ liệu và các hậu quả nghiêm trọng khác. Một lược đồ cơ sở dữ liệu được cho là *tốt* là phải loại bỏ được sự dư thừa dữ liệu. Tuy nhiên ta cần đưa ra định nghĩa chính xác thế nào là lược đồ cơ sở dữ liệu tốt cùng với quá trình thiết kế chúng. Quá trình biến đổi một lược đồ cơ sở dữ liệu thành *lược đồ tương đương*, tức phải bảo toàn thông tin và bảo toàn phụ thuộc dữ liệu, thoả mãn những tiêu chuẩn nhất định gọi là quá trình *chuẩn hoá lược đồ quan hệ*.

Chuẩn hoá lược đồ quan hệ thường được thực hiện qua các giai đoạn tương ứng với các dạng chuẩn (xem sơ đồ dưới). *Dạng chuẩn* là trạng thái quan hệ được xác định bằng cách áp dụng các quy tắc đối với phụ thuộc hàm của quan hệ.



5.4.1. Dạng chuẩn thứ nhất (1NF)

Quan hệ gọi là ở *dạng chuẩn thứ nhất* hay *quan hệ chuẩn hoá* nếu miền giá trị của mỗi thuộc tính chỉ chứa những giá trị *nguyên tử*, tức là không phân chia được nữa. Như vậy mỗi giá trị trong quan hệ cũng là nguyên tử.

Dạng chuẩn 1 chỉ có ý nghĩa ở mức thể hiện của lược đồ quan hệ, vì chỉ liên quan đến giá trị các thuộc tính của các bộ trong một quan hệ được định nghĩa trên lược đồ quan hệ đó.

5.4.2. Dạng chuẩn thứ 2 (2NF)

Thuộc tính A gọi là *phụ thuộc đầy đủ* vào tập thuộc tính X, nếu $X \rightarrow A$ là phụ thuộc hàm nguyên tố.

Giả sử K là khoá của lược đồ R. Khi đó mọi thuộc tính không khoá A của R đều phụ thuộc hàm vào khoá K: $K \rightarrow A$. Nếu A không phụ thuộc đầy đủ vào K thì tồn tại tập con thực sự H của K xác định hàm A, tức $H \rightarrow A$. Khi đó phụ thuộc hàm $H \rightarrow A$ gọi là *phụ thuộc hàm bộ phận*.

Một lược đồ quan hệ gọi là *ở dạng chuẩn thứ 2* nếu nó ở dạng chuẩn thứ 1 và không có phụ thuộc hàm bộ phận, tức là mọi thuộc tính không khoá đều phụ thuộc đầy đủ vào các khoá của lược đồ.

◊ *Ví dụ*

- Xét các quan hệ sau:

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)
PROJ(*PNO, PNAME, BUDGET*)

Lược đồ của EMP có khoá là (*ENO, PNO*).

Phụ thuộc hàm *ENO→*(*ENAME, TITLE*) là phụ thuộc hàm bộ phận vì vế phải là tập con thực sự của khoá. Vậy EMP không ở dạng chuẩn thứ 2.

Lược đồ của PROJ không có phụ thuộc hàm bộ phận, vậy nó ở dạng chuẩn 2.

- Xét quan hệ KHO_HANG(*Kho, Hang, QuayHang, NhanVien*). Lược đồ của quan hệ này có hai phụ thuộc hàm sau:

Kho,Hang→QuayHang: Mỗi mặt hàng ở mỗi kho chỉ được bán ở 1 quầy hàng;

Kho,QuayHang→NhanVien: Mỗi quầy hàng của mỗi kho chỉ có 1 nhân viên phụ trách.

Khoá của lược đồ này là (*Kho, Hang*).

Vậy lược đồ này ở dạng chuẩn thứ 2 vì không có phụ thuộc hàm bộ phận.

5.4.3. Dạng chuẩn thứ 3 (3NF)

Phụ thuộc hàm X→A gọi là *phụ thuộc hàm bắc cầu*, nếu nó là phụ thuộc hàm nguyên tố, A là thuộc tính không khoá, A∉X, và X chứa thuộc tính không khoá.

Khi đó với mọi khoá K ta có các phụ thuộc hàm không tầm thường K→X & X→A. Mặt khác không thể có X→K, vì X chứa các thuộc tính không khoá và không chứa khoá (vì X→A là nguyên tố).

Nói một cách khác phụ thuộc hàm bắc cầu là sự phụ thuộc không tầm thường giữa các thuộc tính không khoá.

Một lược đồ quan hệ gọi là *ở dạng chuẩn thứ 3* nếu nó ở dạng chuẩn thứ 2 và không có phụ thuộc hàm bắc cầu.

◊ *Ví dụ*

- Lược đồ của quan hệ

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)

có khoá là (*ENO, PNO*).

Phụ thuộc hàm *TITLE→SAL* là phụ thuộc hàm bắc cầu. Vậy EMP không ở dạng chuẩn thứ 3.

- Lược đồ của quan hệ

PROJ(*PNO, PNAME, BUDGET*)

không có phụ thuộc hàm bắc cầu, vậy nó ở dạng chuẩn 3.

- Xét quan hệ KHO_HANG(*Kho, Hang, QuayHang, NhanVien*). Ta có hai phụ thuộc hàm sau:

Kho,Hang→QuayHang: Mỗi mặt hàng ở mỗi kho chỉ được bàn ở 1 quầy hàng;

Kho,QuayHang→NhanVien: Mỗi quầy hàng của mỗi kho chỉ có 1 nhân viên phụ trách.

Khoá của lược đồ này là (*Kho, Hang*).

Phụ thuộc hàm thứ hai là phụ thuộc hàm bắc cầu, vì thế lược đồ không ở dạng chuẩn thứ 3, mặc dù nó ở dạng chuẩn thứ 2.

5.4.4. Dạng chuẩn Boyce-Codd (BCNF)

Một lược đồ quan hệ gọi là ở *dạng chuẩn Boyce-Codd* nếu mọi phụ thuộc hàm không tầm thường đều có vế trái là siêu khoá

◊ *Ví dụ*:

- Lược đồ của quan hệ

PROJ(*PNO, PNAME, BUDGET*)

chỉ có phụ thuộc hàm duy nhất *PNO→*(*PNAME, BUDGET*), vậy nó ở dạng chuẩn Boyce-Codd.

- Xét lược đồ

LOPHOC(*Lop,MonHoc,GiaoVien*) với 2 phụ thuộc hàm sau:

$$GiaoVien \rightarrow MonHoc \quad \text{và} \quad (Lop,MonHoc) \rightarrow GiaoVien$$

Lược đồ có 2 khoá

$$K_1 = (Lop, MonHoc) \text{ và } K_2 = (Lop, GiaoVien),$$

nên tất cả thuộc tính đều là thuộc tính khoá. Như vậy lược đồ ở dạng chuẩn thứ 3. Tuy nhiên lược đồ không ở dạng chuẩn Boyce-Codd vì phụ thuộc hàm

$$GiaoVien \rightarrow MonHoc$$

không thoả yêu cầu vế trái phải là siêu khoá.

Sự dị thường khi thêm bộ hay sửa bộ thể hiện ở chỗ nếu một giáo viên dạy nhiều lớp (cùng một môn học) thì thông tin về giáo viên đó lặp lại nhiều lần gây dư thừa dữ liệu.

Sự dị thường khi xoá bộ thể hiện ở chỗ nếu giáo viên T chỉ dạy lớp C nào đó, thì thông tin về giáo viên T (môn học mà giáo viên đó dạy) sẽ bị mất nếu ta xoá bản ghi tương ứng (chẳng hạn vì giáo viên T thôi không dạy lớp C nữa).

5.4.5. Dạng chuẩn thứ 4 (4NF)

Một quan hệ R được gọi là *ở dạng chuẩn thứ 4*, nếu với mỗi phụ thuộc đa trị X→→Y trong R, X cũng xác định hàm tất cả thuộc tính của R.

Như vậy, *nếu quan hệ ở dạng chuẩn BCNF và các phụ thuộc đa trị cũng là phụ thuộc hàm thì quan hệ này ở dạng chuẩn 4*.

◊ *Ví dụ*: Xét quan hệ

SKILL(*ENO, PNO, PLACE*)

ENO	*PNO*	*PLACE*
E1	P1	Toronto
E1	P1	New York
E1	P1	London
E1	P2	Toronto
E1	P2	New York
E1	P2	London
E2	P1	Toronto
E2	P1	New York
E2	P1	London
E2	P2	Toronto
E2	P2	New York
E2	P2	London

Chú ý rằng không có phụ thuộc hàm nào trong quan hệ SKILL; tất cả thuộc tính là thuộc tính khoá. Quan hệ SKILL có hai phụ thuộc đa trị

ENO→→PNO
ENO→→PLACE

Vì quan hệ không có phụ thuộc hàm nên nó ở dạng BCNF. Tuy nhiên nó không ở dạng chuẩn 4, vì *ENO* không phải là khoá.

Để đạt dạng chuẩn 4, cần phân rã SKILL thành hai quan hệ

EP(*ENO, PNO*) và EL(*ENO, PLACE*)

5.4.6. Dạng chuẩn thứ 5 (5NF)

Một quan hệ R được gọi là *ở dạng chuẩn thứ 5*, còn gọi là *dạng chuẩn chiếu-nối* PJNF, nếu mỗi phụ thuộc chiếu nối được xác định bởi các khoá của R.

◊ *Ví dụ*

Với quan hệ PROJ(*PNO, PNAME, BUDGET*) ta có phụ thuộc chiếu-nối

{(*PNO, PNAME*), (*PNO, BUDGET*)}

và mỗi thành phần đều có khoá chính PNO. Vì vậy PROJ ở dạng chuẩn 5.

CHƯƠNG 2

CƠ SỞ DỮ LIỆU PHÂN TÁN

1. Hệ quản trị cơ sở dữ liệu phân tán

1.1. Khái niệm hệ quản trị cơ sở dữ liệu phân tán

Công nghệ các *hệ quản trị cơ sở dữ liệu phân tán* (*distributed database management system - distributed DBMS*) là sự hợp nhất của hai hướng tiếp cận đối với quá trình xử lý dữ liệu: *Công nghệ cơ sở dữ liệu* và *công nghệ mạng máy tính*. Xử lý dữ liệu chuyển từ hệ thống xử lý file cổ điển sang dạng cơ sở dữ liệu, quản lý tập trung. Điều này dẫn đến tính *độc lập dữ liệu*, nghĩa là các ứng dụng được "miễn nhiễm" đối với những thay đổi về tổ chức lôgic hoặc vật lý của dữ liệu và ngược lại.

Một trong những động lực chủ yếu thúc đẩy sử dụng cơ sở dữ liệu là nhu cầu tích hợp các dữ liệu hoạt tác của một xí nghiệp và cho phép truy xuất tập trung. Công nghệ mạng máy tính đặc trưng ở chỗ phi tập trung hoá thiết bị. Tuy nhiên điểm mấu chốt của ý tưởng cơ sở dữ liệu phân tán là *tích hợp* (*integration*), chứ không phải *tập trung hoá* (*centralization*). Cần hiểu rằng có thể tích hợp mà không cần tập trung. Và đây chính là mục tiêu của công nghệ cơ sở dữ liệu phân tán.

Tại sao chúng ta phải thực hiện phân tán ? Câu trả lời kinh điển cho câu hỏi này là việc xử lý phân tán nhằm thích ứng tốt hơn với việc phân bố ngày càng rộng rãi các công ty, xí nghiệp. Nhiều ứng dụng hiện tại của công nghiệp máy tính được phân tán. Thương mại điện tử, các ứng dụng đa phương tiện, giáo dục từ xa, chữa bệnh từ xa, điều khiển sản xuất từ xa, ... là các ví dụ minh hoạ.

Tuy nhiên từ góc độ tổng quát hơn, xử lý phân tán là một biến thể của qui tắc "*chia để trị*" nhằm giải quyết tốt hơn các bài toán lớn và phức tạp. Từ quan điểm kinh tế, cách tiếp cận này có ưu điểm cơ bản là việc tính toán phân tán tận dụng sức mạnh của nhiều bộ phận xử lý một cách tối ưu.

- ***Hệ cơ sở dữ liệu phân tán là gì ?***

Chúng ta có thể định nghĩa một *cơ sở dữ liệu phân tán* là tập hợp nhiều cơ sở dữ liệu có liên quan lôgic và được phân bổ trên một mạng máy tính.

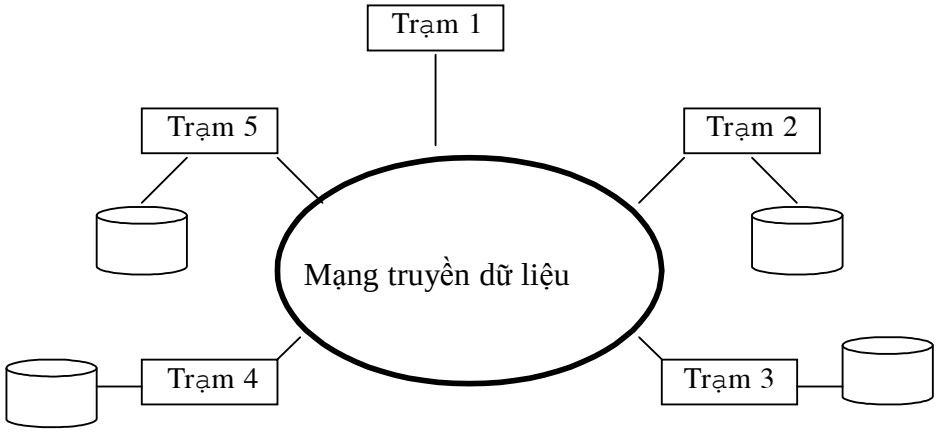
Hệ quản trị cơ sở dữ liệu phân tán (*distributed database management system - distributed DBMS*) là hệ thống phần mềm cho phép quản lý các hệ cơ sở dữ liệu phân tán và làm cho việc phân tán trở nên *vô hình* đối với người sử dụng.

Lưu ý rằng một hệ cơ sở dữ liệu phân tán không phải là tập hợp các tập tin lưu trữ riêng rẽ tại các nút của mạng máy tính. Để tạo ra một hệ cơ sở dữ liệu phân tán, các tập tin không chỉ liên đới lôgic mà còn phải có cấu trúc chung và được truy xuất qua một giao diện chung. Cũng cần phân biệt một hệ cơ sở dữ liệu phân tán với các *dữ liệu bán cấu trúc* (*semi-structrured data*) được lưu trên Internet (chẳng hạn như các trang Web).

Định nghĩa trên cũng loại bỏ các hệ thống đa bộ xử lý dùng chung bộ nhớ trong (shared memory) hoặc dùng chung bộ nhớ thứ cấp (shared disk).

Ngoài ra một hệ cơ sở dữ liệu phân tán không phải là hệ thống mà trong đó, mặc dù có sự hiện diện của mạng máy tính, cơ sở dữ liệu chỉ nằm tại một nút của mạng.

Môi trường của một hệ CSDL phân tán có thể biểu diễn bằng sơ đồ sau



• ***Các đặc trưng của cơ sở dữ liệu phân tán***

Đặc tính *vô hình* là sự tách biệt về ngữ nghĩa ở mức độ cao của hệ thống với các vấn đề cài đặt ở cấp độ thấp. Ưu điểm của hệ cơ sở dữ liệu vô hình là không cho người dùng "nhìn thấy" các chi tiết cài đặt, hỗ trợ phát triển các ứng dụng phức tạp.

Độc lập dữ liệu là dạng vô hình cơ bản cần có trong một hệ cơ sở dữ liệu. Sự độc lập dữ liệu liên quan đến khả năng "miễn nhiễm" của các ứng dụng đối với những thay đổi trong định nghĩa và tổ chức dữ liệu, và ngược lại.

Vô hình kết mạng. Trong môi trường phân tán, hệ thống mạng là một loại tài nguyên quan trọng cần quản lý. Thông thường, người dùng cần được tách khỏi mọi chi tiết hoạt động của mạng, thậm chí người ta mong muốn che dấu sự tồn tại của mạng, nếu được. Khi đó đối với người dùng sẽ không có sự khác biệt giữa các ứng dụng chạy trên cơ sở dữ liệu tập trung và các ứng dụng chạy trên cơ sở dữ liệu phân tán. Kiểu vô hình này gọi là *vô hình kết mạng* (*network transparency*) hoặc *vô hình phân bổ* (*distribution transparency*).

Vô hình nhân bản. Vì những lý do về hiệu năng (performance), độ tin cậy (reliability) và tính sẵn sàng (availability), người ta mong muốn có thể nhân dữ liệu thành nhiều bản (nhân bản) trên các máy mạng. Việc nhân bản giúp tăng hiệu năng vì những yêu cầu sử dụng có xung đột và nằm rải rác có thể đáp ứng kịp thời. Thí dụ, dữ liệu thường được một người truy xuất có thể được đặt tại máy của người đó và trên máy của những người khác có cùng nhu cầu truy xuất, như thế sẽ làm tăng khu vực truy xuất. Ngoài ra nếu một máy phải ngưng hoạt động, một bản sao khác của dữ liệu vẫn có sẵn trên máy khác của mạng. Tuy nhiên việc nhân bản sẽ gây khó khăn khi cập nhật cơ sở dữ liệu. Vì vậy việc nhân bản và qui mô nhân bản do các ứng dụng quyết định.

Vô hình phân mảnh. Phân hoạch dữ liệu cho các vị trí khác nhau là yêu cầu tất yếu của hệ phân tán. Quá trình này gọi là *quá trình phân mảnh* (*fragmentation*). Có hai kiểu phân mảnh. *Phân mảnh ngang* (*horizontal fragmentation*), trong đó mỗi quan hệ được phân hoạch thành tập các quan hệ con, mỗi quan hệ con này chứa một tập con các bộ của quan hệ ban đầu. *Phân mảnh dọc* (*vertical fragmentation*), trong đó mỗi quan hệ được phân hoạch thành tập các quan hệ

con, mỗi quan hệ con này được định nghĩa trên một tập con các thuộc tính của quan hệ ban đầu).

Khi các đối tượng cơ sở dữ liệu bị phân mảnh, chiến lược xử lý vấn tin là dựa trên các mảnh chứ không phải quan hệ. Như vậy *câu vấn tin toàn cục* (*global query*) phải được dịch thành *câu vấn tin theo mảnh* (*fragment query*).

1.2. Mô hình kiến trúc hệ quản trị cơ sở dữ liệu phân tán

Kiến trúc của một hệ thống xác định cấu trúc của nó. Tức là các thành phần của hệ thống được xác định, chức năng mỗi thành phần được mô tả, các mối tương liên (interrelationship) và tương tác (interaction) giữa các thành phần được định nghĩa.

Chúng ta hãy xem xét một số cách kết hợp nhiều cơ sở dữ liệu lại để dùng chung cho nhiều hệ quản trị cơ sở dữ liệu. Ta phân loại hệ thống theo các đặc điểm (1) *tính tự trị* (*autonomy*) của các hệ thống cục bộ, (2) *tính phân tán* (*distribution*) của chúng, (3) *tính đa chủng* (*heterogeneity*) của chúng.

1.2.1. Tính tự trị

Tính tự trị (*autonomy*) muốn nói đến sự *phân bổ quyền điều khiển*, chứ không phải phân bố dữ liệu. Tính tự trị chỉ ra mức độ hoạt tác độc lập của từng hệ quản trị cơ sở dữ liệu. Tính tự trị biểu hiện qua một số yếu tố sau:

- Các hệ thống thành viên có trao đổi thông tin với nhau không.
- Các hệ thống thực hiện các giao dịch một cách độc lập hay không.
- Các hệ thống có được sửa đổi hay không.

Từ đó người ta xây dựng các yêu cầu đối với một hệ thống tự trị. Chẳng hạn hệ thống tự trị phải thoả mãn:

(1) Các hoạt động cục bộ của từng hệ quản trị cơ sở dữ liệu không bị ảnh hưởng bởi sự tham gia của chúng vào trong phức hệ cơ sở dữ liệu (multidatabase system).

(2) Phương thức xử lý và tối ưu hoá vấn tin trong từng hệ quản trị cơ sở dữ liệu không bị ảnh hưởng bởi việc thực hiện các câu truy vấn toàn cục truy xuất nhiều cơ sở dữ liệu.

(3) Tính nhất quán và hoạt động của hệ thống không bị ảnh hưởng khi từng hệ quản trị cơ sở dữ liệu riêng rẽ tham gia hoặc tách ra khỏi liên minh cơ sở dữ liệu.

Ở bình diện khác, tính tự trị thể hiện ở các khía cạnh sau:

(1) *Tự trị thiết kế* (*design autonomy*): Mỗi hệ quản trị cơ sở dữ liệu tự do sử dụng các mô hình dữ liệu và các kỹ thuật quản lý giao dịch thích hợp.

(2) *Tự trị truyền thông* (*communication autonomy*): Mỗi hệ quản trị cơ sở dữ liệu tự do quyết định loại thông tin cung cấp cho hệ quản trị cơ sở dữ liệu khác hoặc cho các phần mềm điều khiển hoạt động toàn cục.

(3) *Tự trị thực thi* (*execution autonomy*): Mỗi hệ quản trị cơ sở dữ liệu có thể thực hiện các giao dịch theo phương thức của mình.

Tính tự trị có thể chia làm ba cấp độ sau:

(**0**) *Tích hợp mật thiết* (*tight integration*): Chỉ tồn tại một hình ảnh duy nhất về toàn bộ hệ thống cơ sở dữ liệu cho người dùng muốn dùng chung thông tin trong

nhiều cơ sở dữ liệu. Một trong các bộ quản lý dữ liệu (data manager) nắm quyền kiểm soát việc xử lý yêu cầu của người dùng, ngay cả khi yêu cầu đó phải được nhiều bộ quản lý dữ liệu tham gia xử lý.

(**1**) *Hệ thống bán tự trị* (*semiautonomous system*): Bao gồm các hệ quản trị cơ sở dữ liệu có thể hoạt tác độc lập, nhưng quyết định tham gia vào liên minh nhằm chia sẻ dữ liệu cục bộ của chúng. Mỗi hệ quản trị cơ sở dữ liệu phải xác định những phần cơ sở dữ liệu nào của riêng chúng mà các hệ quản trị cơ sở dữ liệu khác được truy xuất. Chúng không phải là hệ thống tự trị hoàn toàn mà cần sửa đổi lại để có thể trao đổi thông tin với những hệ thống khác.

(**2**) *Hệ thống cô lập*: Các hệ quản trị cơ sở dữ liệu hoạt động cô lập, không có giao tiếp chia sẻ dữ liệu với nhau.

1.2.2. Tính phân tán

Tính phân tán (*distribution*) chỉ khả năng *phân bố dữ liệu* ở những vị trí khác nhau. Có hai loại kiến trúc phân tán:

(**1**) *Phân tán khách/chủ* (*client/server*): Đây là hình thức phân tán chức năng. Thành phần chủ (server) chịu trách nhiệm quản trị dữ liệu; thành phần khách (client) chịu trách nhiệm cung cấp môi trường ứng dụng, kể cả giao diện người dùng.

Nhiệm vụ truyền thông được chia sẻ giữa chủ và khách.

(**2**) *Phân tán ngang hàng* (*peer-to-peer*): Còn gọi là *phân tán hoàn toàn*. Không có sự phân biệt giữa máy chủ và khách, mỗi máy đều có đầy đủ chức năng của một hệ quản trị cơ sở dữ liệu và có thể trao đổi thông tin với máy khác để thực hiện vấn tin và giao dịch.

1.2.3. Tính đa chủng

Tính đa chủng (*heterogeneous*) thể hiện dưới nhiều hình thái khác nhau trong các hệ phân tán, từ khác biệt về phần cứng, các giao thức kết nối mạng đến sự khác biệt của các bộ quản lý dữ liệu.

Tính đa chủng liên quan đến sự khác biệt các mô hình dữ liệu (data model), ngôn ngữ vấn tin (query language) và nghi thức quản lý giao dịch (transaction management protocol).

1.2.4. Các kiểu kiến trúc

Ta tổ hợp các mức độ tự trị, phân tán và đa chủng để nghiên cứu các mô hình kiến trúc khác nhau.

Ký hiệu

A0 ... hệ thống tự trị tích hợp
A1 ... hệ thống bán tự trị
A2 ... hệ thống cô lập
D0 ... hệ thống không phân tán (tập trung)
D1 ... hệ thống phân tán khách/chủ
D2 ... hệ thống phân tán ngang hàng
H0 ... hệ thống đồng chủng

H1 ... hệ thống đa chủng

• Mô hình (A0,D0,H0): *Hệ thống tích hợp, không phân tán, đồng chủng*. Được gọi là *hệ thống phức hợp* (composite system), bao gồm nhiều hệ quản trị cơ sở dữ liệu được tích hợp về mặt lôgic. Kiến trúc này phù hợp với những hệ thống đa bộ xử lý và mọi tài nguyên dùng chung.

• Mô hình (A0,D0,H1): *Hệ thống tích hợp, không phân tán, đa chủng*. Nó có nhiều hệ quản trị cơ sở dữ liệu đa chủng, nhưng cung cấp một hình ảnh tích hợp cho người dùng. Chẳng hạn trên cơ sở dữ liệu mạng cùng hiện diện cả cơ sở dữ liệu phân cấp và cơ sở dữ liệu quan hệ.

• Mô hình (A0,D1,H0): *Hệ thống tích hợp, phân tán khách/chủ, đồng chủng*. Hệ thống cung cấp một hình ảnh tích hợp cho người dùng.

• Mô hình (A0,D2,H0): *Hệ thống tích hợp, phân tán hoàn toàn, đồng chủng*. Hệ thống không phân biệt khách chủ. Mỗi vị trí đều trang bị đầy đủ chức năng.

• Mô hình (A1,D0,H0): *Hệ thống bán tự trị, không phân tán, đồng chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu liên bang* (*federated DBMS*).

Các hệ thống thành viên đều có quyền tự trị nhất định trong các hoạt động của chúng. Chúng tự do hiệp đồng với những hệ thống khác khi thực hiện các yêu cầu của người dùng truy xuất đến nhiều cơ sở dữ liệu.

◊ *Ví dụ*. Nhiều bản cài đặt một hệ quản trị cơ sở dữ liệu "*mở*" trên cùng một máy. "Mở" ở đây có nghĩa là hệ quản trị cơ sở dữ liệu có khả năng tham gia vào liên bang.

• Mô hình (A1,D0,H1): *Hệ thống bán tự trị, không phân tán, đa chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu liên bang đa chủng* (heterogeneous federated DBMS).

◊ *Ví dụ*. Một hệ quản trị cơ sở dữ liệu quan hệ lo quản lý dữ liệu có cấu trúc, một hệ quản trị cơ sở dữ liệu đồ hoạ lo quản lý hình ảnh tĩnh, và một máy chủ cung cấp các hình video. Nếu chúng ta muốn cung cấp một hình ảnh tích hợp cho người dùng thì cần phải "che dấu" tính tự trị và đa chủng của các hệ thống thành viên và thiết lập giao diện chung.

• Mô hình (A1,D1,H1): *Hệ thống bán tự trị, phân tán khách/chủ, đa chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu liên bang đa chủng phân tán* (heterogeneous federated distributed DBMS). Dữ liệu được phân tán trên các máy khác nhau.

• Kiến trúc (A2,D0,H0): *Hệ thống cô lập, không phân tán, đồng chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu phức hệ* (multidatabase system-MDBS). Các thành viên không có khái niệm hiệp đồng. Đây thực chất là tập các cơ sở dữ liệu tự trị và được kết nối lại.

• Mô hình (A2,D0,H1): *Hệ thống cô lập, không phân tán, đa chủng*. Hệ thống này được dùng để xây dựng các ứng dụng truy xuất dữ liệu từ nhiều hệ thống lưu trữ khác nhau với các đặc tính khác nhau. Một số hệ thống có thể không phải là hệ quản trị cơ sở dữ liệu.

• Mô hình (A2,D1,H1) và (A2,D2,H1): Ta xét chung hai trường hợp này do tính tương tự của những vấn đề do chúng sinh ra. Cả hai đều biểu diễn cho trường hợp các cơ sở dữ liệu thành viên tạo ra phức hệ phân tán trên một số vị trí – chúng được gọi là các *phức hệ cơ sở dữ liệu phân tán* (distributed MDBS). Khác biệt chính giữa hai kiến trúc này là ở phân tán khách/chủ, phần lớn các công việc tương tác được trao cho *hệ thống trung gian* (middleware system), tạo ra *kiến trúc ba tầng* (three layer architecture).

1.3. Kiến trúc hệ quản trị cơ sở dữ liệu phân tán

Chúng ta sẽ xem xét chi tiết ba kiến trúc hệ thống trong số các kiến trúc giới thiệu ở phần trước. Đó là các hệ thống khách/chủ (Ax,D1,Hy), các hệ phân tán (A0,D2,H0) và các phức hệ (A2,Dx,Hy).

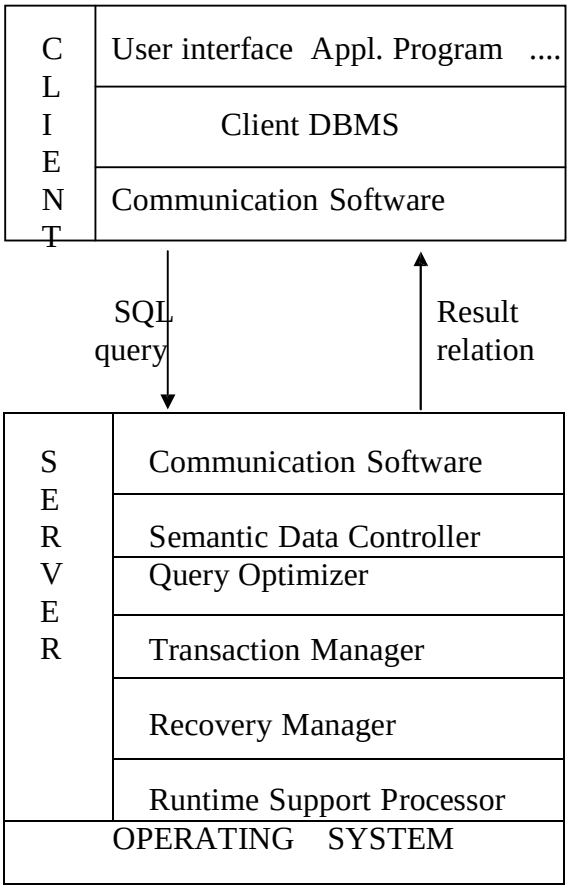
1.3.1. Các hệ khách/chủ

Các hệ quản trị cơ sở dữ liệu *khách/chủ* xuất hiện đầu những năm 90 và có ảnh hưởng lớn đến công nghệ DBMS và phương thức xử lý tính toán. ý tưởng tổng quát hết sức đơn giản và rõ ràng: phân biệt các chức năng cần được cung cấp và chia những chức năng này thành hai lớp: *chức năng chủ* (server function) và *chức năng khách* (client function). Nó cung cấp một *kiến trúc hai tầng* (*two-level architecture*), tạo điều kiện dễ dàng cho việc quản lý mức độ phức tạp của các hệ quản trị cơ sở dữ liệu hiện đại và độ phức tạp của việc phân tán dữ liệu.

Máy chủ thực hiện phần lớn công việc quản lý dữ liệu: xử lý tối ưu hoá vấn tin, quản lý giao dịch, quản lý thiết bị lưu trữ.

Máy khách quản lý các ứng dụng, giao diện, hệ quản trị cơ sở dữ liệu của khách, chịu trách nhiệm quản lý dữ liệu được gửi cho khách và có thể cả quản lý các khoá chốt giao dịch.

Kiến trúc này được mô tả trong hình sau.



kiến trúc tham chiếu khách/chủ

Kiến trúc này thông dụng trong các hệ cơ sở dữ liệu quan hệ, ở đó việc giao tiếp giữa khách và chủ nằm ở mức câu lệnh SQL. Khách hàng chuyển câu vấn tin cho máy chủ mà không cần biết nó thực hiện và tối ưu hoá như thế nào. Máy chủ thực hiện hầu hết công việc và gửi quan hệ kết quả về cho khách.

Có một số kiến trúc khách/chủ khác nhau:

• *Kiến trúc nhiều khách, một chủ.*

Từ góc độ quản lý, loại này không khác nhiều so với cơ sở dữ liệu tập trung, vì CSDL được lưu trên một máy chủ duy nhất và cũng có phần mềm quản lý.

Có một số khác biệt quan trọng ở cách thực hiện các giao dịch và quản lý bộ nhớ cache.

• *Kiến trúc nhiều khách, nhiều chủ*:

Có hai chiến lược quản lý:

- Mỗi máy khách tự quản lý kết nối của nó với các máy chủ khác nhau. Lối tiếp cận này làm đơn giản chương trình ở máy chủ nhưng đặt gánh nặng lên máy khách cùng với các trách nhiệm khác.
- Mỗi máy khách chỉ quan hệ với máy chủ đại diện của mình và giao tiếp với các máy chủ khác thông qua đại diện khi cần.

1.3.2. Các hệ phân tán ngang hàng

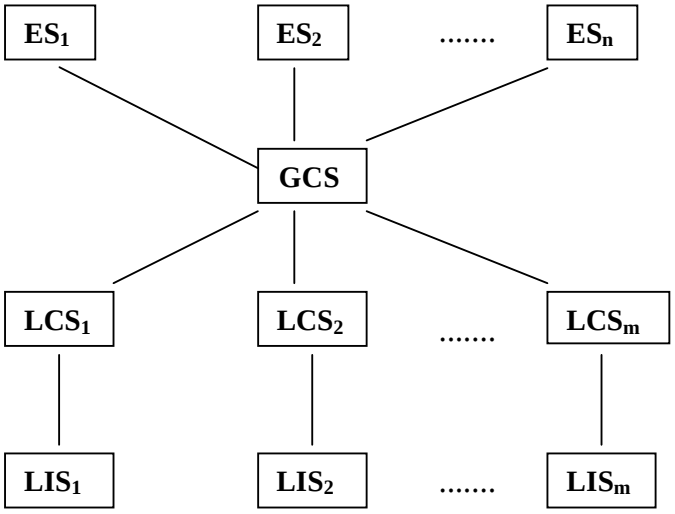
Trong kiến trúc này tổ chức dữ liệu vật lý trên mỗi máy có thể rất khác nhau. Điều này dẫn đến việc định nghĩa cấu trúc dữ liệu riêng cho mỗi vị trí, gọi là *lược đồ nội tại cục bộ* LIS (local internal schema). Cấu trúc lôgic của dữ liệu ở mọi vị trí được mô tả bằng *lược đồ khái niệm toàn cục* GCS (global conceptual schema).

Để mô tả tổ chức lôgic của dữ liệu tại mỗi vị trí cần phải có *tầng thứ ba* trong kiến trúc gọi là *lược đồ khái niệm cục bộ* LCS (local conceptual schema).

Lược đồ khái niệm toàn cục lúc này là *hợp* của các lược đồ khái niệm cục bộ.

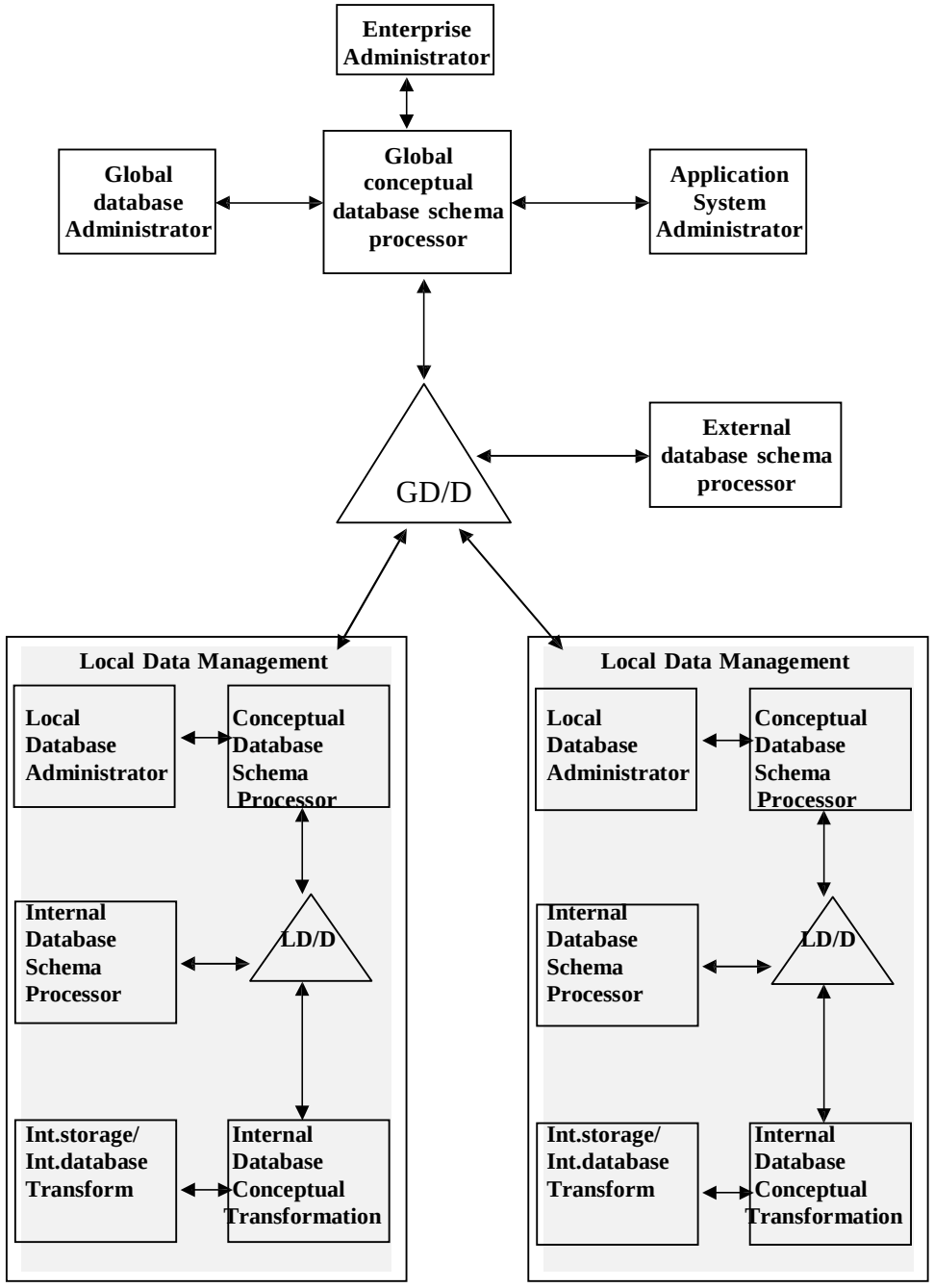
Các ứng dụng và việc truy xuất cơ sở dữ liệu được hỗ trợ qua các *lược đồ ngoại giới* ES (external schema), được định nghĩa là một tầng nằm trên tầng lược đồ khái niệm toàn cục.

Kiến trúc này được mô tả trong hình sau.



kiến trúc tham chiếu CSDL phân tán ngang hàng

Các chức năng của hệ phân tán ngang hàng được biểu diễn bằng sơ đồ sau:



sơ đồ chức năng của hệ quản trị CSDL phân tán ngang hàng

Trong hệ thống này các *từ điển/thư mục dữ liệu*, viết tắt là D/D, có vai trò trung tâm, vừa xử lý lược đồ dữ liệu vừa cung cấp các ánh xạ giữa chúng ở các cấp độ toàn cục và cục bộ.

• *Thư mục/từ điển toàn cục* GD/D (global directory/dictionary) chứa các định nghĩa lược đồ toàn cục và thực hiện các ánh xạ toàn cục.

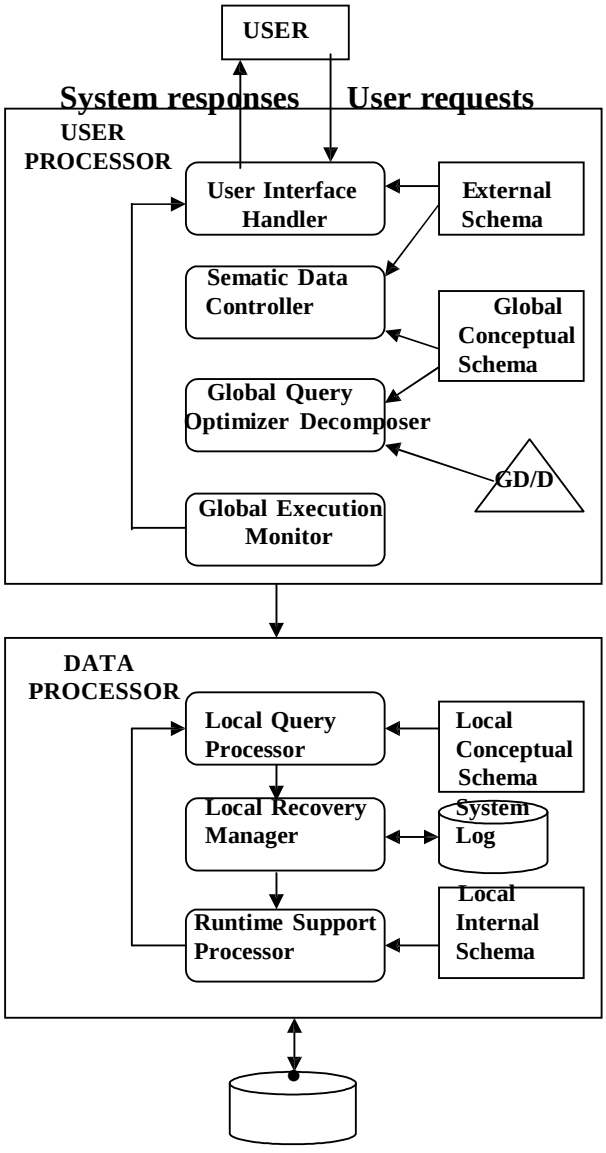
• *Thư mục/từ điển cục bộ* LD/D (local directory/dictionary) chứa các định nghĩa lược đồ cục bộ và thực hiện các ánh xạ cục bộ. Nó cũng có thể chứa các số liệu thống kê về các ứng dụng, các thông tin kiểm soát truy xuất,...

Các thành phần quản lý cơ sở dữ liệu cục bộ được tích hợp nhờ các chức năng của hệ quản trị cơ sở dữ liệu toàn cục.

Trong sơ đồ này, lược đồ khái niệm cục bộ là ánh xạ của lược đồ khái niệm toàn cục vào mỗi vị trí. Hơn nữa, những cơ sở dữ liệu loại này thường được thiết kế theo kiểu từ trên xuống, vì thế tất cả định nghĩa khung nhìn đều có phạm vi toàn cục.

Mỗi vị trí cũng có một quản trị viên cơ sở dữ liệu thể hiện mong muốn có được khả năng điều khiển cục bộ đối với hoạt động quản trị cơ sở dữ liệu.

Các thành phần của hệ quản trị cơ sở dữ liệu phân tán được trình bày trong hình sau:



Hệ thống có hai thành phần chính: Một thành phần lo xử lý mọi tương tác với người dùng gọi là *bộ phận phục vụ người dùng* (user processor), còn thành phần thứ hai lo việc lưu trữ dữ liệu, gọi là *bộ phận xử lý dữ liệu* (data processor).

- *Bộ phận phục vụ người dùng* (user processor): bao gồm bốn phần:
  (1) *Bộ phận giao tiếp* (user interface handler) chịu trách nhiệm diễn dịch các yêu cầu của người dùng (user requests) và định dạng dữ liệu kết quả để chuyển cho người dùng.
  (2) *Bộ phận kiểm soát dữ liệu ngữ nghĩa* (sematic data controller): sử dụng các *ràng buộc toàn vẹn* (integrity constraints) và thông tin quyền hạn (authorization), được định nghĩa như thành phần của lược đồ khái niệm toàn cục, để kiểm tra xem các câu vấn tin có thể xử lý được hay không.
  (3) *Bộ phận phân rã và tối ưu hoá vấn tin* (global query optimizer and decomposer) xác định chiến lược hoạt động nhằm giảm thiểu chi phí, phiên dịch các câu vấn tin toàn cục thành các câu vấn tin cục bộ bằng cách sử dụng các lược đồ khái niệm toàn cục, lược đồ khái niệm cục bộ và các thư mục toàn cục. Bộ phận tối ưu vấn tin toàn cục, ngoài những nhiệm vụ khác, còn chịu trách nhiệm tạo ra chiến lược thực thi tốt nhất cho các phép nối phân tán.
  (4) *Bộ phận theo dõi hoạt động phân tán* (distributed execution monitor) điều phối việc thực hiện phân tán các yêu cầu người dùng và cũng được gọi là *bộ quản lý giao dịch phân tán* (distributed transaction manager). Khi thực hiện các vấn tin phân tán, các bộ phận tại các vị trí có thể giao tiếp với nhau.

- *Bộ phận xử lý dữ liệu* (data processor): bao gồm ba phần.
  (1) *Bộ phận xử lý câu vấn tin cục bộ* (local query processor): hoạt động như *bộ chọn đường truy xuất* (access path selector), chịu trách nhiệm chọn ra một đường truy xuất thích hợp nhất để truy xuất các mục dữ liệu.
  (2) *Bộ phận khôi phục cục bộ* (local recovery manager): bảo đảm cho các cơ sở dữ liệu cục bộ vẫn duy trì được tính nhất quán ngay cả khi có sự cố xảy ra.
  (3) *Bộ phận hỗ trợ thực thi* (run-time support processor): truy xuất cơ sở dữ liệu tuỳ vào các lệnh trong *lịch biểu* (schedule) do bộ phận tối ưu vấn tin sinh ra. Nó chính là giao diện với hệ điều hành và chứa *bộ quản lý vùng đệm cơ sở dữ liệu* (database buffer manager), chịu trách nhiệm quản lý vùng đệm và việc truy xuất dữ liệu.

Lưu ý rằng việc sử dụng thuật ngữ *Bộ phận phục vụ người dùng* và *Bộ phận xử lý dữ liệu* không phải là sự phân chia chức năng giống như các hệ khách/chủ. Sự phân chia này chỉ thể hiện khía cạnh tổ chức và không bắt buộc phải đặt trên các máy khác nhau. Trong các hệ thống ngang hàng, người ta mong muốn có cả môdun phục vụ người dùng và môđun xử lý dữ liệu trên cùng một máy.

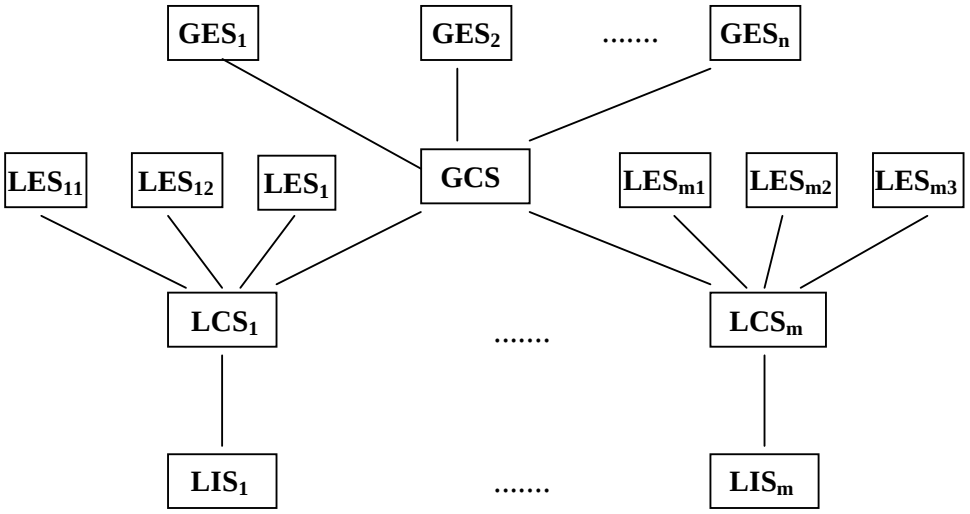
1.3.3. Các phức hệ cơ sở dữ liệu

Hệ quản trị phức hệ cơ sở dữ liệu phân tán khác biệt với hệ quản trị cơ sở dữ liệu phân tán về

- mức độ tự trị: phản ánh trong các mô hình kiến trúc
- định nghĩa lược đồ khái niệm toàn cục: trong hệ phức hợp lược đồ khái niệm toàn cục chỉ là một tập con bao gồm một số cơ sở dữ liệu cục bộ mà mỗi hệ quản trị CSDL muốn dùng chung, còn trong hệ phân tán cơ sở dữ liệu toàn cục là hợp các cơ sở dữ liệu cục bộ.

a) Các mô hình sử dụng lược đồ khái niệm toàn cục

Trong phức hệ cơ sở dữ liệu, lược đồ khái niệm toàn cục GCS được định nghĩa bằng cách tích hợp các lược đồ ngoài của các cơ sở dữ liệu tự trị hoặc các thành phần của lược đồ khái niệm cục bộ của chúng, xem hình sau



kiến trúc phức hệ CSDL với một lược đồ khái niệm toàn cục

Người dùng của hệ quản trị cơ sở dữ liệu cục bộ sẽ định nghĩa khung nhìn riêng (LES) của họ trên cơ sở dữ liệu cục bộ và không cần thay đổi các ứng dụng hiện có nếu họ không truy xuất dữ liệu của cơ sở dữ liệu khác. Đây chính là khía cạnh tự trị của kiến trúc này.

Thiết kế lược đồ khái niệm toàn cục trong phức hệ cơ sở dữ liệu bao gồm việc tích hợp các lược đồ khái niệm cục bộ (ánh xạ đi từ dưới lên, từ lược đồ khái niệm cục bộ lên lược đồ khái niệm toàn cục, ngược lại với hệ phân tán) hoặc tích hợp các lược đồ ngoài cục bộ (ánh xạ đi theo chiều từ trên xuống).

Một khi đã thiết kế xong GCS, các khung nhìn trên lược đồ có thể định nghĩa cho người dùng cần truy xuất ở phạm vi toàn cục. Các lược đồ ngoại giới toàn cục GES và lược đồ khái niệm toàn cục GCS không nhất thiết sử dụng cùng một mô hình và cùng ngôn ngữ, chúng không cần xác định xem hệ thống *đồng chủng* hay *đa chủng*.

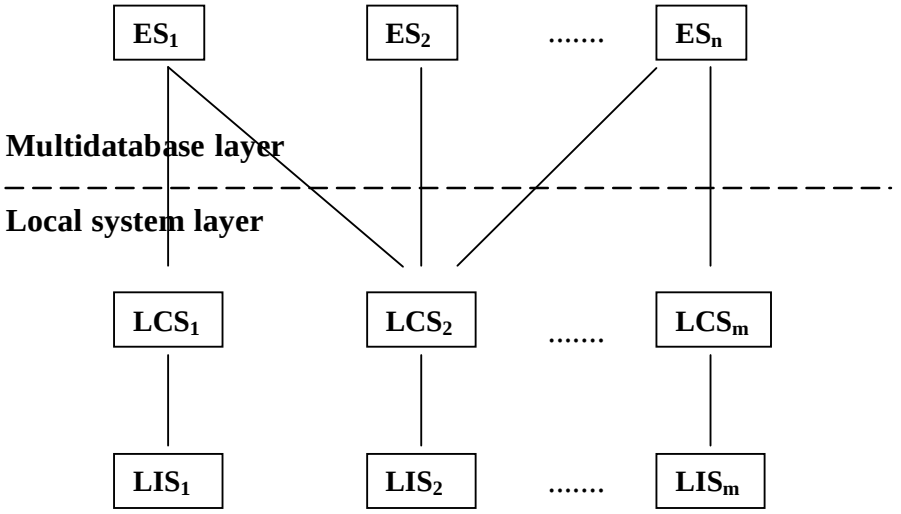
Trong trường hợp đa chủng, ta có hai lựa chọn cài đặt: *đơn ngôn* (unilingal) hoặc *đa ngôn* (multilingal).

Trong phức hệ cơ sở dữ liệu đơn ngôn, khi truy xuất toàn cục người dùng sử dụng chung một lược đồ ngoại giới toàn cục và một ngôn ngữ xử lý toàn cục.

Trong phức hệ cơ sở dữ liệu đa ngôn, khi truy xuất toàn cục mỗi người dùng sử dụng lược đồ ngoại giới toàn cục được định nghĩa bằng ngôn ngữ của hệ quản trị cơ sở dữ liệu cục bộ của mình và các câu vấn tin toàn cục cũng được tạo bằng ngôn ngữ của hệ quản trị cơ sở dữ liệu cục bộ.

b) Các mô hình không có lược đồ khái niệm toàn cục

Kiến trúc của phức hệ cơ sở dữ liệu không có lược đồ khái niệm toàn cục được trình bày trong hình sau



kiến trúc phức hệ CSDL không có lược đồ khái niệm toàn cục

Kiến trúc này có hai tầng: *tầng hệ thống cục bộ* và *tầng phức hệ CSDL* phía trên.

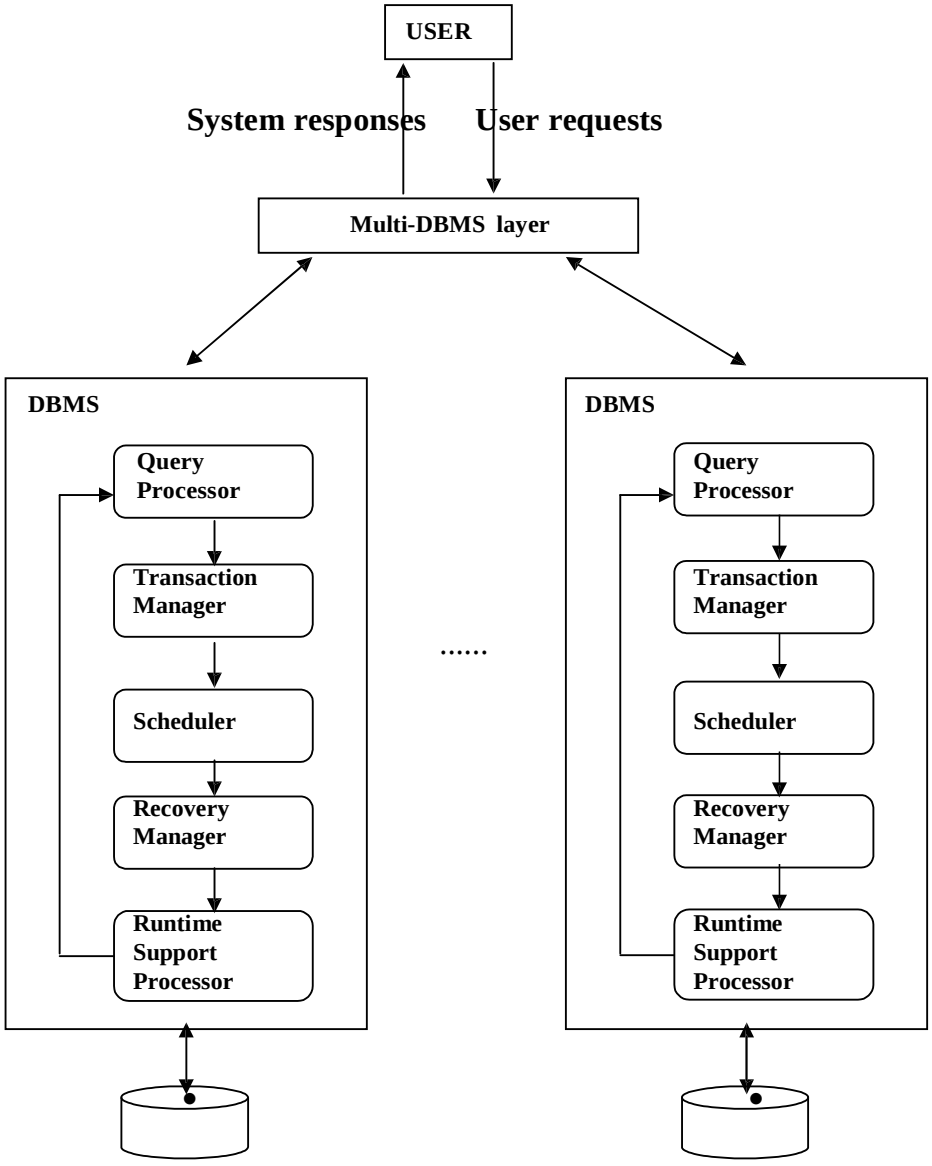
Tầng hệ thống cục bộ bao gồm một số hệ quản trị cơ sở dữ liệu, với chức năng là giới thiệu cho tầng phức hệ cơ sở dữ liệu các thành phần của cơ sở dữ liệu cục bộ có thể dùng chung với những cơ sở dữ liệu khác. Dữ liệu dùng chung này được trình bày qua lược đồ khái niệm cục bộ thực sự hoặc qua lược đồ ngoại giới cục bộ. Nếu có vấn đề đa chủng, mỗi lược đồ LCS_i có thể sử dụng mô hình dữ liệu khác nhau.

Tầng phức hệ cơ sở dữ liệu gồm các lược đồ ngoại giới, trong đó mỗi khung nhìn được định nghĩa trên một hay nhiều lược đồ khái niệm cục bộ. Vì vậy trách nhiệm cung cấp quyền truy xuất đến nhiều cơ sở dữ liệu (có thể đa chủng) được trao cho ánh xạ giữa lược đồ ngoại giới và lược đồ khái niệm cục bộ. Đây là điểm khác biệt cơ bản so với kiến trúc sử dụng lược đồ khái niệm toàn cục, trong đó quan hệ giữa lược đồ ngoại giới và lược đồ khái niệm cục bộ được thực hiện bởi các ánh xạ thông qua lược đồ khái niệm toàn cục.

Kiến trúc cơ sở dữ liệu liên bang cũng không sử dụng lược đồ khái niệm toàn cục. Trong hệ thống này mỗi hệ quản trị cơ sở dữ liệu cục bộ định nghĩa một

lược đồ xuất (export schema), trong đó định nghĩa dữ liệu muốn chia sẻ với các DBMS khác, mỗi ứng dụng truy xuất đến cơ sở dữ liệu toàn cục qua định nghĩa của *lược đồ nhập* (import schema), thực chất là hình ảnh ngoại giới toàn cục.

Các thành phần của hệ quản trị phức hợp khác biệt nhiều so với hệ quản trị cơ sở dữ liệu phân tán. ở đây có một tầng phần mềm chạy bên trên những hệ quản trị cơ sở dữ liệu riêng biệt và cung cấp cho người dùng những tiện ích để truy xuất nhiều cơ sở dữ liệu khác nhau. Tuỳ thuộc vào sự tồn tại hay không lược đồ khái niệm toàn cục và vấn đề đa chủng mà nội dung phần mềm sẽ thay đổi cho phù hợp.



các thành phần của phức hệ CSDL

1.4. Tổ chức thư mục toàn cục

Các vấn đề *tổ chức thư mục toàn cục* chỉ được đề cập đến trong các hệ phân tán và phức hệ có sử dụng lược đồ khái niệm toàn cục.

Thư mục toàn cục cũng là cơ sở dữ liệu chứa dữ liệu về các dữ liệu thực sự lưu trữ trong cơ sở dữ liệu (còn gọi là *meta dữ liệu* hay *siêu dữ liệu*). Vì thế những kỹ thuật thiết kế cơ sở dữ liệu phân tán cũng áp dụng cho việc quản lý thư mục. Như vậy thư mục có thể *toàn cục* đối với toàn bộ cơ sở dữ liệu hoặc *cục bộ* đối với từng vị trí. Nghĩa là có thể có một thư mục duy nhất chứa các thông tin về tất cả dữ liệu trong cơ sở dữ liệu, hoặc có một số thư mục, mỗi thư mục chứa thông tin được lưu ở các vị trí khác nhau. Trong trường hợp sau chúng ta có thể xây dựng hệ phân cấp thư mục để dễ dàng khi tìm kiếm hoặc cài đặt một chiến lược tìm kiếm phân tán cho phép trao đổi giữa các vị trí lưu trữ thư mục.

Vấn đề thứ hai liên quan đến vị trí chứa thư mục. Thư mục có thể được duy trì tập trung tại một vị trí hoặc phân tán đến một số vị trí. Giữ thư mục tại một vị trí làm cho việc quản lý được dễ dàng, nhưng có thể làm tăng tải trọng tại đó, gây ùn tắc lưu lượng thông báo ở vị trí đó. Ngược lại, phân tán thư mục trên nhiều vị trí làm giảm tải trọng tại một điểm nhưng sẽ làm cho vấn đề quản lý thư mục thêm phức tạp. Trong các phức hệ cơ sở dữ liệu, sự lựa chọn sẽ phụ thuộc vào vấn đề hệ thống có phân tán hay không. Nếu có, thư mục sẽ được phân tán, ngược lại nó được quản lý tập trung.

Vấn đề thứ ba là nhân bản thư mục. Có thể có một bản thư mục duy nhất hoặc nhiều bản. Có nhiều bản thư mục sẽ làm tăng độ tin cậy của hệ thống do khả năng truy xuất được một bản thư mục sẽ cao hơn. Hơn nữa thời gian trễ khi truy xuất thư mục sẽ giảm đi do ít xảy ra tranh chấp và khoảng cách đến các bản sao sẽ ngắn hơn. Tuy nhiên, việc duy trì cập nhật các bản sao cũng phức tạp và tốn kém. Vì vậy sự lựa chọn sẽ phụ thuộc vào môi trường hệ thống và phải cân bằng các yếu tố như thời gian đáp ứng, kích thước thư mục, khả năng của máy ở mỗi vị trí, yêu cầu khả tín, mức độ thay đổi của thư mục.

2. Thiết kế cơ sở dữ liệu phân tán

Thiết kế một hệ thống máy tính phân tán cần phải chọn *vị trí đặt dữ liệu* và *chương trình* trên một mạng máy tính, rất có thể phải kể luôn cả việc thiết kế mạng. Đối với hệ quản trị cơ sở dữ liệu phân tán cần phải thực hiện hai điều: *phân tán cơ sở dữ liệu* và *phân tán các chương trình ứng dụng* chạy trên hệ đó. ở đây chúng ta chỉ tập trung vào việc phân tán dữ liệu.

Việc tổ chức các hệ phân tán có thể được nghiên cứu dựa theo ba trục không gian

- *Mức độ chia sẻ dữ liệu* (level of sharing)
- *Kiểu mẫu truy xuất* (behavior of access pattern)
- *Mức độ hiểu biết về kiểu mẫu truy xuất*

(1) Theo mức độ chia sẻ có ba khả năng xảy ra:

- *Không chia sẻ dữ liệu*: mỗi ứng dụng và dữ liệu của nó thực thi tại một vị trí, không có trao đổi hoặc giao tiếp với những chương trình khác hoặc truy xuất dữ liệu ở những vị trí khác. Hình thức này đặc trưng cho các kết nối mạng ở thời kỳ sơ khai.
- *Chia sẻ dữ liệu*: tất cả chương trình đều được nhân bản cho mỗi vị trí, nhưng không nhân bản dữ liệu. Theo đấy các yêu cầu của người dùng được xử lý tại mỗi vị trí và dữ liệu cần thiết được chuyển đi trên mạng.
- *Chia sẻ dữ liệu – chương trình*: cả chương trình và dữ liệu được dùng chung. Nghĩa là chương trình nằm tại một vị trí có thể yêu cầu dịch vụ từ một chương trình nằm ở vị trí thứ hai, và đến lượt nó, chương trình này có thể truy xuất dữ liệu nằm tại vị trí thứ ba.

Ở đây cần phân biệt giữa *Chia sẻ dữ liệu* và *Chia sẻ dữ liệu - chương trình*, đặc biệt đối với hệ phân tán đa chủng. Trong môi trường đa chủng rất khó khăn, có khi không thể được, cho thực thi một chương trình trên một phần cứng khác và trong hệ điều hành khác.

(2) Theo kiểu mẫu truy xuất có hai kiểu lựa chọn:

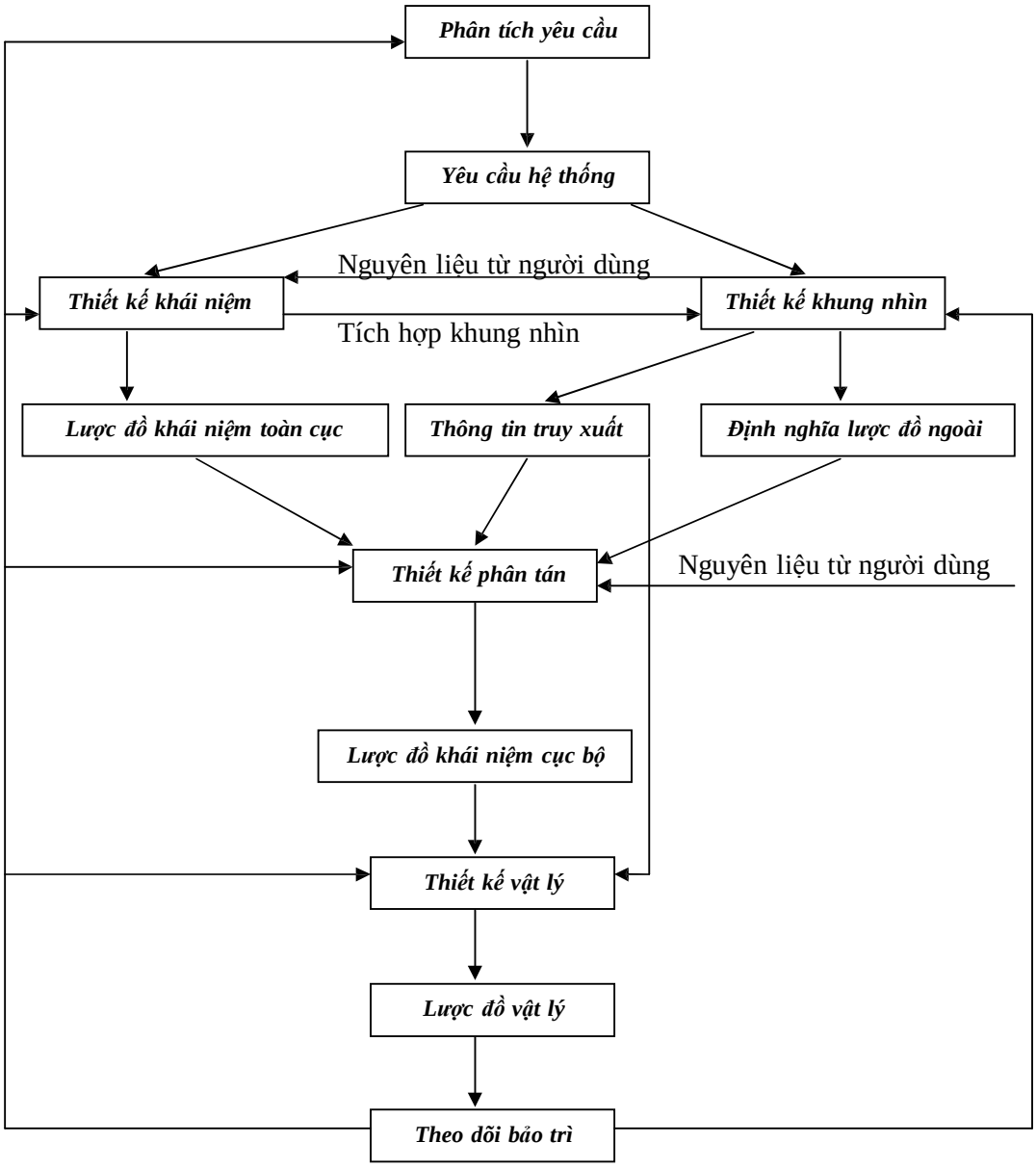
- Loại *tĩnh*, không thay đổi theo thời gian
- Loại *động*, thay đổi theo thời gian.

2.1. Các chiến lược thiết kế

Hai chiến lược chính trong việc thiết kế cơ sở dữ liệu phân tán là *tiếp cận từ trên xuống* và *tiếp cận từ dưới lên*. Trong thực tế rất hiếm các ứng dụng đơn giản để chỉ sử dụng một cách tiếp cận, vì vậy trong phần lớn thiết kế cả hai cách tiếp cần đều được áp dụng bổ sung nhau.

2.1.1. Quá trình thiết kế từ trên xuống

Sơ đồ quá trình thiết kế từ trên xuống biểu diễn trong hình sau:



Việc *phân tích yêu cầu* nhằm định nghĩa môi trường hệ thống và thu nhập các nhu cầu xử lý của tất cả người dùng, đồng thời cũng xác định *yêu cầu hệ thống.*

Hồ sơ ghi chép các yêu cầu là nguyên liệu cho hai hoạt động song song: *Thiết kế khung nhìn* (view design) và *Thiết kế khái niệm* (conceptual design).

Thiết kế khung nhìn định nghĩa các giao diện cho *người dùng đầu cuối* (end-user).

Thiết kế khái niệm là quá trình xem xét tổng thể đối tượng – xí nghiệp, nhằm xác định các loại thực thể và mối liên hệ giữa chúng với nhau. Ta có thể chia quá trình này thành 2 nhóm bao gồm các hoạt động liên quan tới nhau: *Phân tích thực*

thể (entity analysis) và *Phân tích chức năng* (functional analysis). Phân tích thực thể có liên quan đến việc xác định các thực thể, các thuộc tính và các mối liên hệ giữa chúng. Phân tích chức năng đề cập đến việc xác định các chức năng cơ bản có liên quan đến xí nghiệp cần được mô hình hoá. Kết quả của hai quá trình này cần được đối chiếu qua lại, giúp chúng ta biết được chức năng nào sẽ hoạt tác trên những thực thể nào.

Có sự liên hệ khăng khít giữa thiết kế khái niệm và thiết kế khung nhìn. Theo nghĩa nào đó thiết kế khái niệm được coi như là sự tích hợp các khung nhìn. Tuy nhiên mô hình khái niệm cần phải hỗ trợ không chỉ những ứng dụng hiện có mà còn cả những ứng dụng trong tương lai. Tích hợp khung nhìn nhằm đảm bảo các yêu cầu về thực thể và các mối liên hệ giữa các khung nhìn đều phải được bao quát trong lược đồ khái niệm.

Trong các hoạt động thiết kế khái niệm và thiết kế khung nhìn, người thiết kế cần phải đặc tả các thực thể dữ liệu và phải xác định các ứng dụng chạy trên cơ sở dữ liệu cũng như các thông tin thống kê về những ứng dụng này. Thông tin thống kê bao gồm đặc tả về tần số ứng dụng, khối lượng thông tin khác nhau,...

Lược đồ khái niệm toàn cục GCS và thông tin về kiểu mẫu truy xuất thu được trong thiết kế khung nhìn sẽ là nguyên liệu (input) cho bước *thiết kế phân tán*. Mục tiêu của giai đoạn này là thiết kế các lược đồ khái niệm cục bộ LCS bằng cách phân tán các thực thể cho các vị trí của hệ thống phân tán.

Ta chia quan hệ thành nhiều quan hệ nhỏ hơn gọi là các *mảnh* (*fragment*) và phân tán các mảnh này. Hoạt động thiết kế phân tán gồm hai bước: *Phân mảnh* (*fragmentation*) và *cấp phát* (*allocation*). Ta sẽ thảo luận về vấn đề này trong các phần sau.

Thiết kế vật lý là ánh xạ lược đồ khái niệm cục bộ sang các thiết bị lưu trữ vật lý có sẵn tại các vị trí tương ứng. Nguyên liệu cho quá trình này là lược đồ khái niệm cục bộ và thông tin về kiểu mẫu truy xuất các mảnh.

Hoạt động phát triển và thiết kế luôn là quá trình liên tục, đòi hỏi theo dõi hiệu chỉnh thường xuyên. Vì thế chúng ta đưa vấn đề quan sát và theo dõi như một hoạt động chính trong quá trình này. Cần chú ý rằng chúng ta không chỉ theo dõi vấn đề cài đặt cơ sở dữ liệu, mà còn quan sát theo dõi tính thích hợp của các khung nhìn của người dùng. Kết quả này có tác dụng phản hồi, tạo cơ sở cho việc tái thiết kế về sau.

2.1.2. Quá trình thiết kế từ dưới lên

Thiết kế từ trên xuống thích hợp cho những cơ sở dữ liệu được thiết kế từ đầu. Tuy nhiên trong thực tế cũng có khi đã có sẵn một số cơ sở dữ liệu, và chúng ta phải tích hợp chúng thành một cơ sở dữ liệu chung. Tiếp cận từ dưới lên sẽ thích hợp cho tình huống này. Khởi điểm của thiết kế từ dưới lên là các lược đồ khái niệm cục bộ, sẽ phải được tích hợp thành lược đồ khái niệm toàn cục.

2.2. Các vấn đề thiết kế phân tán

Trong mục này chúng ta sẽ trả lời các câu hỏi sau:

- Tại sao cần phân mảnh
- Làm thế nào để thực hiện phân mảnh
- Phân mảnh nên thực hiện đến mức độ nào
- Cách thức kiểm tra tính đúng đắn của phân mảnh
- Cách thức cấp phát dữ liệu
- Những thông tin nào cần thiết cho phân mảnh và cấp phát

2.2.1. Các lý do phân mảnh

Trước tiên, khung nhìn của các ứng dụng thường chỉ là tập con của quan hệ. Vì thế đơn vị truy xuất không phải toàn bộ quan hệ mà chỉ là tập con của quan hệ. Kết quả là xem tập con của quan hệ là đơn vị phân tán là thích hợp.

Thứ hai là, nếu các ứng dụng có các khung nhìn được định nghĩa trên một quan hệ cho trước lại nằm tại những vị trí khác nhau thì chỉ có hai cách chọn lựa với đơn vị phân tán là toàn bộ quan hệ, khi không có phân mảnh: Hoặc quan hệ không được nhân bản mà được lưu ở một vị trí, hoặc quan hệ được nhân bản cho tất cả hoặc một số vị trí có chạy ứng dụng. Chọn lựa đầu gây ra một số lượng lớn truy xuất không cần thiết đến dữ liệu ở xa. Còn chọn lựa sau có thể dẫn đến nhân bản không cần thiết, gây khó khăn khi cập nhật và lãng phí không gian lưu trữ.

Cuối cùng việc phân rã quan hệ thành nhiều mảnh, mỗi mảnh xử lý như một đơn vị, sẽ cho phép thực hiện nhiều giao dịch đồng thời. Việc phân mảnh quan hệ cho phép thực hiện song song câu vấn tin, bằng cách chia nó thành một tập câu vấn tin con hoạt tác trên các mảnh. Vì thế việc phân mảnh sẽ làm tăng mức độ hoạt động đồng thời và kéo theo tăng lưu lượng hoạt động của hệ thống. Kiểu hoạt động này gọi là *đồng thời nội vấn tin (intraquery concurency)*, sẽ được phân tích trong các phần sau.

2.2.2. Các kiểu phân mảnh

Có hai kiểu phân mảnh: *phân mảnh theo chiều dọc* và *phân mảnh theo chiều ngang*.

◊ *Ví dụ*

Chúng ta sử dụng lược đồ cơ sở dữ liệu đã phát triển trong chương trước. Ta thêm vào lược đồ PROJ thuộc tính LOC (vị trí) để chỉ nơi thực hiện dự án. Sau đây là một thể hiện cơ sở dữ liệu sẽ được dùng:

EMP

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

ASG

ENO	*PNO*	*RESP*	*DUR*
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	*PNAME*	*BUDGET*	*LOC*
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000

Trong hình sau trình bày quan hệ PROJ được tách ngang thành 2 quan hệ $PROJ_1$ chứa các thông tin về dự án có kinh phí dưới 200000 USD, và $PROJ_2$ chứa các thông tin về dự án có kinh phí lớn hơn 200000 USD.

$PROJ_1$

PNO	*PNAME*	*BUDGET*	*LOC*
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

$PROJ_2$

PNO	*PNAME*	*BUDGET*	*LOC*
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

Còn trong hình sau trình bày quan hệ PROJ được tách dọc thành 2 quan hệ $PROJ_1$ chứa các thông tin về kinh phí dự án, và $PROJ_2$ chứa các thông tin về tên và vị trí dự án.

$PROJ_1$

PNO	*BUDGET*
P1	150000
P2	135000
P3	250000
P4	310000

$PROJ_2$

PNO	*PNAME*	*LOC*
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAm	New York
P4	Maintenamce	Paris

Việc phân mảnh có thể lồng ghép, vừa phân mảnh ngang vừa phân mảnh dọc, thành *phân mảnh tổng hợp* (*hybrid fragmentation*).

2.2.3. Các qui tắc phân mảnh đúng đắn

Có ba qui tắc trong khi phân mảnh đảm bảo cơ sở dữ liệu sẽ không thay đổi ngữ nghĩa khi phân mảnh.

1) *Tính đầy đủ* (*completeness*): Cho quan hệ **r** bất kỳ. Giả sử **r** được phân rã thành các mảnh. Khi đó tính đầy đủ yêu cầu mỗi mục dữ liệu trong **r** cũng phải được lưu trữ trong một hoặc vài mảnh nào đó.

2) *Tính tái thiết* (*reconstruction*): Cho quan hệ **r** bất kỳ. Giả sử **r** được phân rã thành các mảnh $\mathbf{r_1}$, ..., $\mathbf{r_n}$. Khi đó tính tái thiết yêu cầu "hợp" các phân mảnh của quan hệ **r** trả lại đầy đủ dữ liệu ban đầu của quan hệ **r**. Khái niệm "hợp" ở đây là toán tử quan hệ Δ sao cho

$$\mathbf{r} = \Delta_{i=1}^{n} r_i$$

Toán tử Δ thay đổi tuỳ theo từng loại phân mảnh.

Khả năng tái thiết một quan hệ từ các mảnh của nó cũng phải đảm bảo rằng các ràng buộc định nghĩa theo phụ thuộc dữ liệu sẽ được bảo toàn.

3) *Tính tách biệt* (*disjointness*): Cho quan hệ **r** bất kỳ. Giả sử **r** được phân rã thành các mảnh $\mathbf{r_1}$, ..., $\mathbf{r_n}$. Khi đó tính tách biệt yêu cầu một mục dữ liệu *d* nào đó một khi đã xuất hiện trong mảnh $\mathbf{r_i}$ thì sẽ không xuất hiện trong mảnh $\mathbf{r_k}$ khác. Tiêu chuẩn này đảm bảo các mảnh ngang sẽ tách biệt nhau. Còn trong phân mảnh dọc thì các thuộc tính khoá chính phải được lặp lại trong mỗi mảnh, vì vậy tính tách biệt chỉ áp dụng với các thuộc tính không khoá.

2.2.4. Các kiểu cấp phát

Giả sử cơ sở dữ liệu đã được phân mảnh thích hợp và cần phải quyết định cấp phát các mảnh cho các vị trí trên mạng. Khi dữ liệu được cấp phát, nó có thể được nhân bản hoặc chỉ duy trì một bản duy nhất.

Một cơ sở dữ liệu không nhân bản, gọi là *cơ sở dữ liệu phân hoạch*, có chứa các mảnh được cấp phát cho các vị trí, trong đó chỉ tồn tại một bản duy nhất cho mỗi mảnh trên mạng.

Kiểu *cơ sở dữ liệu nhân bản* có hai dạng:

- *Cơ sở dữ liệu nhân bản hoàn toàn*, trong đó toàn bộ cơ sở dữ liệu đều có bản sao ở mỗi vị trí.
- *Cơ sở dữ liệu nhân bản một phần*, trong đó các mảnh được phân tán đến các vị trí, mỗi mảnh có thể có nhiều bản sao nằm ở các vị trí khác nhau. Số lượng các bản sao của các mảnh có thể là tham số (input) cho các thuật toán cấp phát (allocation algorithm) hoặc là biến quyết định (dicision variable) mà giá trị của nó được xác định bằng thuật toán này.

Lý do nhân bản là nhằm đảm bảo được độ tin cậy và hiệu quả cho các câu vấn tin chỉ đọc. Nếu có nhiều bản sao của một mục dữ liệu thì chúng ta vẫn có cơ hội truy xuất được dữ liệu đó ngay cả khi hệ thống có sự cố. Hơn nữa các câu vấn tin chỉ đọc truy xuất đến cùng một mục dữ liệu có thể cho thực hiện song song vì các bản sao có mặt tại nhiều vị trí.

Ngược lại, trong cơ sở dữ liệu nhân bản các câu vấn tin cập nhật có thể gây nhiều rắc rối vì phải đảm bảo các bản sao phải được cập nhật chính xác.

Vì vậy quyết định nhân bản cần được cân nhắc và phụ thuộc vào tỉ lệ giữa các câu vấn tin chỉ đọc và câu vấn tin cập nhật. Quyết định này hầu như ảnh hưởng đến tất cả các thuật toán của hệ quản trị cơ sở dữ liệu phân tán và các chức năng kiểm soát khác.

2.3. Phương pháp phân mảnh

Trong phần này chúng ta sẽ bàn đến các chiến lược và thuật toán phân mảnh. Có hai chiến lược phân mảnh cơ bản: *phân mảnh ngang* (*horizontal fragmentation*) và *phân mảnh dọc* (*vertical fragmentation*). Ngoài ra còn có khả năng phân mảnh hỗn hợp.

2.3.1. Phân mảnh ngang

Phân mảnh ngang chia quan hệ theo các bộ. Mỗi mảnh là một tập con của quan hệ. Có hai loại phân mảnh ngang: *phân mảnh nguyên thuỷ* (*primary horizontal fragmentation*), thực hiện dựa trên các vị từ định nghĩa trên chính

quan hệ đó, và *phân mảnh dẫn xuất* (*derived horizontal fragmentation*), dựa trên các vị từ định nghĩa trên quan hệ khác.

Trước khi thực hiện phân mảnh, chúng ta cần thu thập thông tin cần thiết.

a) Yêu cầu thông tin

• *Thông tin về cơ sở dữ liệu*

Thông tin này bao gồm lược đồ khái niệm toàn cục, các liên kết giữa các quan hệ, đặc biệt là phép nối.

Trong mô hình quan hệ, các mối liên hệ được biểu thị bằng các quan hệ. Tuy nhiên trong các mô hình khác, như mô hình thực thể-quan hệ, các mối liên hệ được biểu diễn tường minh. Với mục đích thiết kế phân tán, các mối liên hệ cũng được mô hình hoá trong bộ khung quan hệ. Theo cách này chúng ta sẽ vẽ các đường nối (L) có hướng giữa các quan hệ (R, S) ràng buộc nhau qua phép đẳng nối dạng

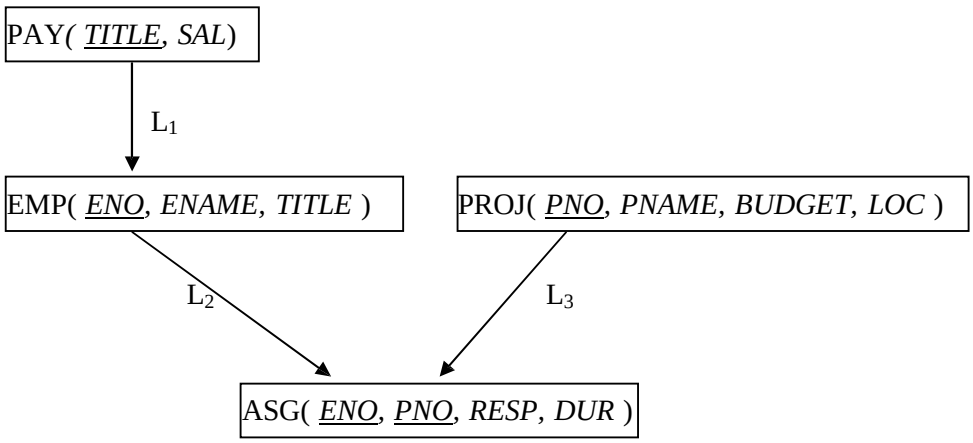
$$R \xrightarrow{L} S$$

trong đó R gọi là quan hệ chủ, S gọi là quan hệ thành viên. Người ta dùng hàm *owner* và *member* để phân biệt các quan hệ này:

$$owner(L) = R \quad \text{và} \quad member(L) = S$$

◊ *Ví dụ*

Hình sau trình bày một cách biểu diễn các đường nối giữa các quan hệ PAY, EMP, PROJ và ASG.



ở đây có ba đường nối L_1, L_2, L_3. Ta có

$owner(L_1) = PAY$ và $member(L_1) = EMP$

$owner(L_2) = EMP$ và $member(L_2) = ASG$

$owner(L_3) = PROJ$ và $member(L_3) = ASG$

Thông tin định lượng về cơ sở dữ liệu, tức là *lực lượng* (*cardinality*) của mỗi quan hệ R, ký hiệu *card*(R).

• *Thông tin về ứng dụng*

Yêu cầu thông tin định tính lẫn định lượng về các ứng dụng. Thông tin định tính định hướng cho hoạt động phân mảnh, còn thông tin định lượng sử dụng cho các mô hình cấp phát.

Những thông tin định tính cơ bản gồm các vị từ được dùng trong các câu vấn tin. Nếu không thể phân tích được hết tất cả các ứng dụng để xác định những vị từ này thì ít nhất cũng phải nghiên cứu được các ứng dụng quan trọng nhất. Một hướng dẫn quan trọng, gọi là *qui tắc 80/20*, là "*20% câu vấn tin sẽ chiếm đến 80% truy xuất dữ liệu*".

Cho quan hệ $R(A_1, ..., A_n)$, trongđó A_i là một thuộc tính được định nghĩa trên miền giá trị D_i. Một *vị từ đơn giản* p được định nghĩa trên R có dạng:

$$p: A_i \, \theta \; Value$$

Trong đó $\theta \in \{ =, <, \leq, \neq, >, \geq \}$ và *Value* là giá trị được chọn từ miền D_i.

◊ *Ví dụ*

Xét bảng quan hệ PROJ. Các biểu thức

$$PNAME = \text{"Maintenance"} \quad \text{và} \quad BUDGET \leq 200000$$

là các vị từ đơn giản.

Các câu vấn tin thường chứa nhiều vị từ phức tạp, là tổ hợp các vị từ đơn giản. Một tổ hợp cần đặc biệt chú ý được gọi là *tiểu hạng* (minterm predicate), là *hội* (conjuction) của các vị từ đơn giản. Bởi vì ta luôn có thể biến đổi một biểu thức bool thành *dạng chuẩn hội* (conjuctive normal form), việc sử dụng tiểu hạng trong thuật toán thiết kế không làm mất tính tổng quát.

Cho một tập các vị từ đơn giản $Pr = \{p_1, ..., p_m\}$ trên quan hệ R. Tập các tiểu hạng $M=\{m_1, ..., m_z\}$ gồm các vị từ dạng

$$m_j = \bigwedge_{p_k \in Pr} p_k^* \text{ , với } 1 \leq k \leq m, \; 1 \leq j \leq z$$

trong đó $p_k^* = p_k$ hoặc $p_k^* = \neg p_k$, $\forall k=1,...,m$. Như vậy mỗi vị từ đơn giản có thể xuất hiện trong vị từ tiểu hạng dạng khẳng định hoặc phủ định.

◊ *Ví dụ*

Xét bảng quan hệ PAY. Sau đây là các vị từ đơn giản định nghĩa trên PAY.

p_1 : *TITLE* = "Elect. Eng."
p_2 : *TITLE* = "Syst. Anal."
p_3 : *TITLE* = "Mech. Eng."
p_4 : *TITLE* = "Programmer"
p_5 : *SAL* ≤ 30000
p_6 : *SAL* > 30000

Từ các vị từ đơn giản trên có thể định nghĩa các vị từ tiểu hạng sau

m_1 : *TITLE* = "Elect. Eng." ∧ *SAL* ≤ 30000
m_2 : *TITLE* = "Elect. Eng." ∧ *SAL* > 30000
m_3 : ¬(*TITLE* = "Elect. Eng.") ∧ *SAL* ≤ 30000
m_4 : ¬(*TITLE* = "Elect. Eng.") ∧ *SAL* > 30000
m_5 : *TITLE* = "Programmer" ∧ *SAL* ≤ 30000
m_6 : *TITLE* = "Programmer" ∧ *SAL* > 30000

Theo những thông tin định lượng về các ứng dụng, chúng ta cần biết hai tập dữ liệu.

(1) *Độ tuyển tiểu hạng* (*minterm selectivity*): Số lượng các bộ quan hệ sẽ được truy xuất bởi câu vấn tin được đặc tả theo một vị từ tiểu hạng đã cho. Ta ký hiệu độ tuyển của vị từ tiểu hạng *m* là *sel(m).*

(2) *Tần số truy xuất* (*access frequency*): tần số ứng dụng truy xuất dữ liệu. Cho

$$Q = \{ q_1, q_2, ..., q_k \}$$

là tập các câu vấn tin, ký hiệu $acc(q_i)$ biểu thị tần số truy xuất của q_i trong một khoảng thời gian đã cho. Ta cũng ký hiệu tần số truy xuất của một vị từ tiểu hạng *m* là *acc(m).*

b) Phân mảnh ngang nguyên thuỷ

Phân mảnh ngang nguyên thuỷ được định nghĩa bằng một phép toán chọn trên các quan hệ chủ của một lược đồ CSDL.

Cho quan hệ **r**, các mảnh ngang của **r** là các quan hệ con r_i, i = 1, ..., k, với

$$\mathbf{r_i} = \sigma_{F_i}(r), \text{ i = 1, ..., k}$$

trong đó F_i , i = 1, ..., k, là công thức chọn để có mảnh $\mathbf{r_i}$.

◊ *Ví dụ*

Phân rã quan hệ PROJ thành các mảnh ngang $PROJ_1$ và $PROJ_2$ trong ví dụ trên có thể được định nghĩa như sau:

$$PROJ_1 = \sigma_{BUDGET \le 200000}(PROJ)$$
$$PROJ_2 = \sigma_{BUDGET > 200000}(PROJ)$$

Ta cũng có thể định nghĩa các mảnh ngang sau đây

$$PRJ_1 = \sigma_{LOC="Montreal"}(PROJ)$$
$$PRJ_2 = \sigma_{LOC="New\ York"}(PROJ)$$
$$PRJ_3 = \sigma_{LOC="Paris"}(PROJ)$$

PRJ_1

PNO	*PNAME*	*BUDGET*	*LOC*
P1	Instrumentation	150000	Montreal

PRJ_2

PNO	*PNAME*	*BUDGET*	*LOC*
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York

PRJ_3

PNO	*PNAME*	*BUDGET*	*LOC*
P4	Maintenamce	310000	Paris

Bây giờ ta có thể định nghĩa một mảnh ngang chặt chẽ hơn. Một mảnh ngang $\mathbf{r_i}$ của quan hệ **r** có chứa tất cả các bộ của **r** thoả vị từ tiểu hạng m_i. Như vậy, cho tập M các vị từ tiểu hạng, số mảnh ngang bằng số các vị từ tiểu hạng trong tập M. Tập các mảnh ngang này gọi là tập các *mảnh tiểu hạng* (minterm fragment).

Theo như đã phân tích, việc định nghĩa các mảnh ngang phụ thuộc vào các vị từ tiểu hạng. Vì thế bước đầu tiên của mọi thuật toán phân mảnh là xác định tập các vị từ đơn giản sẽ cấu thành các vị từ tiểu hạng.

Các vị từ đơn giản cần có các tính chất *đầy đủ* và *cực tiểu*.

• *Tính đầy đủ*. Tập các vị từ đơn giản *Pr* gọi là *đầy đủ* nếu và chỉ nếu xác suất mỗi ứng dụng truy xuất đến một bộ bất kỳ thuộc về một mảnh tiểu hạng nào đó được định nghĩa theo *Pr* đều bằng nhau.

◊ *Ví dụ*. Xét phân mảnh PRJ_1, PRJ_2, PRJ_3 ở ví dụ trước. Nếu ứng dụng duy nhất truy xuất PROJ muốn truy xuất các bộ theo vị trí, thì tập vị từ (ứng dụng)

$$Pr = \{LOC=\text{"Montreal"}, LOC=\text{"New York"}, LOC=\text{"Paris"}\}$$

là đầy đủ bởi vì mỗi bộ của mỗi mảnh PRJ_i đều có xác suất truy xuất như nhau.

Tuy nhiên, nếu có ứng dụng thứ hai chỉ truy xuất các bộ dự án có ngân sách nhỏ hơn 200000 USD, thì tập vị từ *Pr* không còn đầy đủ nữa. Một số bản ghi trong mỗi mảnh $PROJ_i$ được ứng dụng thứ hai truy xuất với xác suất cao hơn.

Để cho tập vị từ đầy đủ ta phải thêm các vị từ $BUDGET \leq 200000$, $BUDGET > 200000$ vào *Pr*, tức là

$$Pr = \{LOC=\text{"Montreal"}, LOC=\text{"New York"}, LOC=\text{"Paris"}, BUDGET \leq 200000, BUDGET > 200000\}$$

Lý do cần phải đảm bảo tính đầy đủ là vì các mảnh thu được theo tập vị từ đầy đủ sẽ nhất quán về mặt logic do tất cả chúng đều thoả vị từ tiểu hạng. Chúng cũng đồng nhất về mặt thống kê theo cách mà các ứng dụng truy xuất chúng. Vì vậy chúng ta sẽ sử dụng tập vị từ đầy đủ làm cơ sở cho phân mảnh ngang nguyên thủy.

• *Tính cực tiểu*

Cho tập các vị từ đơn giản *Pr*. Ta nói một vị từ là *có liên đới* (*relevant*) trong việc xác định phân mảnh, nếu nó ảnh hưởng đến việc mảnh f nào đó bị phân thành các mảnh con f_1 và f_2 thì phải có ít nhất một ứng dụng truy xuất đến f_1 và f_2 theo cách khác nhau.

Tập *Pr* gọi là *cực tiểu* nếu mọi vị từ trong nó là *có liên đới* và không có vị từ tương đương.

◊ *Ví dụ*. Tập vị từ *Pr* trong ví dụ trước là đầy đủ và cực tiểu. Tuy nhiên nếu chúng ta thêm vị từ

$$PNAME=\text{"Instrumentation"}$$

vào *Pr*, thì *Pr* không còn là cực tiểu nữa bởi vì vị từ trên không có liên đới ứng với *Pr*. Không có ứng dụng truy xuất khác nhau đến các mảnh ảnh hưởng bởi vị từ trên .

Bây giờ chúng ta sẽ trình bày một thuật toán lặp sinh ra một tập các vị từ đầy đủ và cực tiểu. Ta sẽ sử dụng qui tắc cơ bản về tính đầy đủ và cực tiểu gọi tắt là qui tắc 1.

♦ *Qui tắc 1*. Một quan hệ hoặc một mảnh được phân hoạch thành ít nhất hai phần và chúng được truy xuất khác nhau bởi một ứng dụng.

◊ Ký hiệu *f(p)* là mảnh sinh bởi vị từ *p* và *F* là tập các mảnh, f_k là mảnh sinh bởi vị từ p_k.

• ***Thuật toán: COM_MIN***

Đầu vào: Quan hệ **r** và tập các vị từ đơn giản *Pr*.
Đầu ra: Tập các vị từ *Pr'* đầy đủ và cực tiểu.
Khai báo: *F* là tập các mảnh tiểu hạng.
begin
Tìm vị từ $p \in Pr$ sao cho *p* phân hoạch **r** theo qui tắc 1
$Pr' \leftarrow p$
$Pr \leftarrow Pr - \{p\}$
$F \leftarrow f\{= f(p)\}$
repeat
begin
Tìm vị từ $p \in Pr$ sao cho *p* phân hoạch một mảnh f_k nào đó của *F* theo qui tắc 1 :
$Pr' \leftarrow Pr' \cup \{p\}$
$Pr \leftarrow Pr - \{p\}$
$F \leftarrow F \cup \{f\}$
if tồn tại $p_k \in Pr'$ không liên đới hoặc có vị từ tương đương **then**
begin
$Pr' \leftarrow Pr' - \{p_k\}$
$F \leftarrow F - \{f_k\}$
end
end
until *Pr'* đầy đủ
end. {COM_MIN}

Bước thứ hai trong quá trình thiết kế phân mảnh ngang nguyên thuỷ là suy dẫn ra các tập vị từ tiểu hạng có thể được định nghĩa trên các vị từ trong *Pr'*. Các vị từ tiểu hạng này xác định các mảnh "ứng cử viên" cho bước cấp phát.

Việc xác định các vị từ tiểu hạng không khó. Khó khăn chính là tập này rất lớn (thực sự chúng tỉ lệ hàm mũ theo số lượng vị từ đơn giản), cần phải giảm số lượng.

Bước thứ ba là loại bỏ một số mảnh vô nghĩa. Điều này được thực hiện bằng cách xác định những vị từ mâu thuẫn với tập các *phép kéo theo*, kí hiệu là I.

Chẳng hạn, nếu $Pr' = \{p_1, p_2\}$, trong đó

$$p_1 : att = value1 \quad \& \quad p_2 : att = value2$$

và miền giá trị của thuộc tính *att* là {*value1*,*value2*}. Như vậy sẽ phát sinh hai phép kéo theo:

$$i_1 : p_1 \Rightarrow \neg p_2 \qquad \text{và} \qquad i_2 : \neg p_1 \Rightarrow p_2$$

Bốn vị từ tiểu hạng của *Pr'* được định nghĩa như sau:

$$m_1 : (att=value1) \wedge (att=value2)\ m_2 : (att=value1) \wedge \neg(att=value2)\ m_3 : \neg(att=value1) \wedge (att=value2)\ m_4 : \neg(att=value1) \wedge \neg(att=value2)$$

Trong trường hợp này các vị từ tiểu hạng m_1 và m_4 mâu thuẫn với các phép kéo theo i_1 và i_2 , và vì thế chúng bị loại khỏi tập các vị từ tiểu hạng M.

Thuật toán phân mảnh ngang nguyên thuỷ được trình bày như sau:

- ***Thuật toán: PHORIZONTAL***

Đầu vào: Quan hệ **r** và tập các vị từ đơn giản *Pr*.
Đầu ra: Tập các vị từ tiểu hạng M.
begin
Đặt *Pr'*:=COM_MIN(**r**, *Pr*);
Xác định tập M các vị từ tiểu hạng;
Xác định tập I các phép kéo theo giữa các vị từ trong *Pr'*;
Loại vị từ tiểu hạng mâu thuẫn:
for mỗi vị từ tiểu hạng $m \in$ M do
if *m* mâu thuẫn với I **then**
M $\leftarrow$ M − {*m*} { loại *m* khỏi M}
End-if
End-for
End.

◊ *Ví dụ*

Xét thiết kế lược đồ CSDL: EMP, ASG, PROJ, PAY cho ở phần trên. Có hai quan hệ cần phân mảnh ngang nguyên thuỷ là PAY và PROJ.

Giả sử có một ứng dụng truy xuất PAY, ứng dụng này kiểm tra thông tin lương và xác định số lương sẽ tăng. Giả sử rằng các mẩu tin nhân viên được quản lý ở hai nơi. Một nơi xử lý các mẩu tin có lương ≤ 30000 USD, và nơi khác xử lý các mẩu tin của nhân viên có lương >30000 USD. Vì thế câu vấn tin được sử dụng ở cả hai nơi.

Tập vị từ đơn giản được sử dụng để phân hoạch PAY là

$$p_1 : \text{SAL} \leq 30000 \qquad \& \qquad p_2 : \text{SAL} > 30000$$

Từ đó ta có tập vị từ đơn giản khởi đầu là

$$Pr = \{p_1, p_2\}$$

Áp dụng thuật toán COM_MIN ta có tập khởi đầu *Pr*' = $\{p_1\}$. Đây là tập đầy đủ và cực tiểu vì p_2 không phân hoạch mảnh $f_1 = f(p_1)$ theo qui tắc 1. Chúng ta có thể tạo ra các vị từ tiểu hạng cho tập M:

$$m_1 : (\text{SAL} \leq 30000) = p_1 \quad \text{và} \quad m_2 : \neg(\text{SAL} \leq 30000) = (\text{SAL} > 30000) = p_2$$

Sau đó ta định nghĩa hai mảnh $F_{PAY} = \{\text{PAY}_1, \text{PAY}_2\}$ theo M:

PAY_1

TITLE	*SAL*
Mech.Eng.	27000
Programmer	24000

PAY_2

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000

Bây giờ ta xét quan hệ PROJ. Giả sử có hai ứng dụng.

Ứng dụng thứ nhất được đưa ra tại ba vị trí và cần tìm tên và ngân sách của các dự án khi biết vị trí. Theo ký pháp SQL, câu vấn tin được viết là

```
SELECT   PNAME, BUDGET
FROM     PROJ
WHERE    LOC = Value
```

Đối với ứng dụng này, các vị từ đơn giản có thể dùng là

$$p_1 : LOC = \text{"Montreal"}, \quad p_2 : LOC = \text{"New York"}, \quad p_3 : LOC = \text{"Paris"}$$

Ứng dụng thứ hai được đưa ra tại hai vị trí và liên quan tới việc điều hành dự án. Những dự án có ngân sách ≤ 200000 USD được quản lý tại một vị trí, còn những dự án có ngân sách > 200000 USD được quản lý tại vị trí thứ hai. Vì thế các vị từ đơn giản phải được sử dụng để phân mảnh theo ứng dụng thứ hai là

$$p_4 : BUDGET \leq 200000 \quad \text{và} \quad p_5 : BUDGET > 200000$$

Nếu kiểm tra bằng thuật toán COM_MIN, tập

$$\text{Pr'} = \{p_1, p_2, p_3, p_4, p_5\}$$

rõ ràng là đầy đủ và cực tiểu.

Dựa trên *Pr*' chúng ta có thể định nghĩa sáu vị từ tiểu hạng sau tạo nên M.

$$m_1 : (LOC = \text{"Montreal"}) \wedge (BUDGET \leq 200000)$$
$$m_2 : (LOC = \text{"Montreal"}) \wedge (BUDGET > 200000)$$

m_3 : (*LOC* = "New York") $\wedge$ (*BUDGET* $\leq$ 200000)
m_4 : (*LOC* = "New York") $\wedge$ (*BUDGET* > 200000)
m_5 : (*LOC* = "Paris") $\wedge$ (*BUDGET* $\leq$ 200000)
m_6 : (*LOC* = "Paris") $\wedge$ (*BUDGET* > 200000)

Đây không phải là tất cả các vị từ tiểu hạng có thể được tạo ra. Chẳng hạn có thể định nghĩa vị từ

$$p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5$$

Tuy nhiên, các phép kéo theo hiển nhiên là

i_1 : $p_1 \Rightarrow \neg p_2 \wedge \neg p_3$
i_2 : $p_2 \Rightarrow \neg p_1 \wedge \neg p_3$
i_3 : $p_3 \Rightarrow \neg p_1 \wedge \neg p_2$
i_4 : $p_4 \Rightarrow \neg p_5$
i_5 : $p_5 \Rightarrow \neg p_4$
i_6 : $\neg p_4 \Rightarrow p_5$
i_7 : $\neg p_5 \Rightarrow p_4$

cho phép loại bỏ những vị từ tiểu hạng khác và chúng ta còn lại m_1 đến m_6.
Kết quả của phân mảnh ngang nguyên thuỷ cho PROJ là tạo ra sáu mảnh

$$F_{PROJ} = \{PROJ_1, PROJ_2, PROJ_3, PROJ_4, PROJ_5, PROJ_6\}$$

của quan hệ PROJ tương ứng theo các vị từ tiểu hạng m_1 đến m_6 trong M. Chú ý rằng các mảnh $PROJ_2$ và $PROJ_5$ rỗng.

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

$PROJ_3$

PNO	PNAME	BUDGET	LOC
P2	Database Develop.	135000	New York

$PROJ_4$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York

$PROJ_6$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

c) Phân mảnh ngang dẫn xuất

Phân mảnh ngang dẫn xuất được định nghĩa trên một quan hệ thành viên của một đường nối dựa theo phép toán chọn trên quan hệ chủ nhân của đường nối đó. Ta cần lưu ý hai điểm sau. Trước tiên đường nối giữa quan hệ chủ và quan hệ thành viên được định nghĩa bằng một phép đẳng nối. Thứ hai, đẳng nối có thể

được cài đặt nhờ các phép *bán nối*. Điểm này rất quan trọng vì ta muốn phân hoạch quan hệ thành viên theo phân mảnh của quan hệ chủ, nhưng cũng muốn mảnh thu được chỉ định nghĩa trên các thuộc tính của quan hệ thành viên.

Muốn thực hiện phân mảnh ngang dẫn xuất, ta cần ba yếu tố đầu vào: tập các phân hoạch của quan hệ chủ, quan hệ thành viên, và tập các vị từ bán nối giữa quan hệ chủ và quan hệ thành viên.

Như thế, nếu cho trước đường nối L, trong đó *owner*(L)=S và *member*(L)=R, các mảnh ngang dẫn xuất của R được định nghĩa là

$$R_i = R >< S_i \text{ , } 1 \le i \le n$$

trong đó *n* là số lượng mảnh được định nghĩa trên R, và $S_i = \sigma_{F_i}(S)$ với F_i là công thức định nghĩa mảnh nguyên thuỷ S_i.

◊ *Ví dụ*. Xét đường nối L_1 trong ví dụ mục a). Ta có *owner*(L_1)=PAY và *member*(L_1)=EMP. Ta có thể nhóm các kỹ sư (engineer) thành hai nhóm tuỳ theo lương: nhóm có lương từ 30000 USD trở xuống và nhóm có lương từ 30000 USD trở lên. Hai mảnh EMP_1 và EMP_2 được định nghĩa như sau

$$EMP_1 = EMP >< PAY_1 \quad \text{và} \quad EMP_2 = EMP >< PAY_2$$

trong đó

$$PAY_1 = \sigma_{SAL \le 30000}(PAY) \quad \text{và} \quad PAY_2 = \sigma_{SAL > 30000}(PAY)$$

Kết quả được trình bày ở các bảng sau

EMP_1

ENO	*ENAME*	*TITLE*
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E7	R.David	Mech.Eng.

EMP_2

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E8	J.Jones	Syst.Anal.

Cũng có vấn đề phức tạp cần chú ý. Trong lược đồ CSDL chúng ta hay gặp nhiều đường nối đến quan hệ R (chẳng hạn có hai đường nối đến quan hệ ASG trong ví dụ trên). Như thế có thể có nhiều cách phân mảnh ngang dẫn xuất cho R. Quyết định chọn cách phân mảnh nào dựa trên 2 tiêu chuẩn.

(1) Phân mảnh nào có đặc tính nối tốt hơn.
(2) Phân mảnh nào được sử dụng trong nhiều ứng dụng hơn.

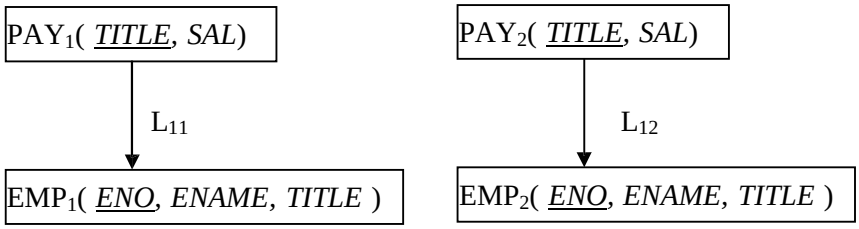
Chúng ta sẽ bàn về tiêu chuẩn thứ 2 trước. Tiêu chuẩn rất đơn giản nếu xét đến tần số truy xuất dữ liệu của các ứng dụng. Nếu được, ta nên ưu tiên những người dùng quan trọng để tác động của họ lên hiệu năng hệ thống là nhỏ nhất.

Tuy nhiên, việc áp dụng tiêu chuẩn thứ nhất không đơn giản. Chẳng hạn thử xét phân mảnh đã thảo luận trong ví dụ trước. Tác động (và mục tiêu) của phân mảnh này là nối của các quan hệ EMP và PAY để trả lời các câu vấn tin được hỗ

trợ (a) bằng cách thực hiện nó trên các quan hệ nhỏ hơn (tức các mảnh) và (b) bằng cách thực hiện các nối theo lối phân tán.

Điểm (a) là hiển nhiên. Các mảnh của EMP nhỏ hơn bản thân EMP. Vì thế khi nối một mảnh của PAY với một mảnh của EMP sẽ nhanh hơn là nối các quan hệ này.

Tuy nhiên điểm (b) lại quan trọng hơn và là trọng tâm của CSDL phân tán. Nếu ngoài việc thực hiện được nhiều câu vấn tin tại nhiều vị trí khác nhau, chúng ta có thể cho thực hiện song song một câu vấn tin, thời gian đáp ứng và lưu lượng hệ thống sẽ được cải thiện. Chẳng hạn xét đồ thị nối giữa các mảnh EMP và PAY lấy từ ví dụ trên. Mỗi mảnh chỉ có 1 đường nối đến hoặc đi khỏi.



Một đồ thị nối như thế gọi là *đồ thị đơn giản*. Thiết kế, trong đó mối liên hệ nối giữa các mảnh là đơn giản, có ưu điểm là quan hệ thành viên và quan hệ chủ có thể được cấp phát cho một vị trí, và các nối giữa các cặp mảnh khác nhau có thể tiến hành độc lập và song song.

Tuy nhiên, không phải lúc nào cũng thu được các đồ thị nối đơn giản. Khi đó thiết kế cần tạo ra *đồ thị nối phân hoạch*. Một đồ thị phân hoạch chứa 2 hoặc nhiều đồ thị con và không có đường nối giữa chúng. Các mảnh như thế không dễ phân tán để thực hiện song song như trong các đồ thị đơn giản, nhưng vẫn có thể cấp phát.

◊ *Ví dụ*. Xét tiếp ví dụ trên. Ta đã phân mảnh EMP theo phân mảnh của PAY. Bây giờ ta xét quan hệ ASG. Giả sử có hai ứng dụng sau:

(1) Ứng dụng thứ nhất tìm tên các kỹ sư có làm việc tại một nơi nào đó. Ứng dụng này chạy ở cả ba trạm và truy xuất thông tin về các kỹ sư làm việc trong các dự án tại chỗ với xác suất cao hơn các kỹ sư làm việc ở những vị trí khác.

(2) Ứng dụng thứ hai là tại mỗi trạm quản lý, nơi lưu các mẩu tin nhân viên, người dùng muốn truy xuất đến các dự án đang được các nhân viên này thực hiện và cần biết xem họ sẽ làm việc với dự án đó trong bao lâu.

Ứng dụng thứ nhất dẫn đến việc phân mảnh ASG theo các mảnh $PROJ_1$, $PROJ_3$, $PROJ_4$ và $PROJ_6$ của PROJ thu được trong ví dụ phân mảnh ngang nguyên thuỷ, trong đó

$$PROJ_1 = \sigma_{LOC=\text{"Montreal"} \wedge BUDGET \leq 200000}(PROJ)$$
$$PROJ_3 = \sigma_{LOC=\text{"New York"} \wedge BUDGET \leq 200000}(PROJ)$$
$$PROJ_4 = \sigma_{LOC=\text{"New York"} \wedge BUDGET > 200000}(PROJ)$$

$$PROJ_6 = \sigma_{LOC=\text{"Paris"} \wedge BUDGET > 200000}(PROJ)$$

Vì thế phân mảnh dẫn xuất của ASG theo $PROJ_1$, $PROJ_3$, $PROJ_4$ và $PROJ_6$ được định nghĩa như sau:

$$ASG_1 = ASG >< PROJ_1$$
$$ASG_2 = ASG >< PROJ_3$$
$$ASG_3 = ASG >< PROJ_4$$
$$ASG_4 = ASG >< PROJ_6$$

Thể hiện các mảnh này được trình bày ở các bảng sau:

ASG_1

ENO	*PNO*	*RESP*	*DUR*
E1	P1	Manager	12
E2	P1	Analyst	24

ASG_3

ENO	*PNO*	*RESP*	*DUR*
E3	P3	Consultant	10
E7	P3	Engineer	36
E8	P3	Manager	40

ASG_2

ENO	*PNO*	*RESP*	*DUR*
E2	P2	Analyst	6
E4	P2	Programmer	18
E5	P2	Manager	24

ASG_4

ENO	*PNO*	*RESP*	*DUR*
E3	P4	Engineer	48
E6	P4	Manager	48

Ứng dụng thứ hai là câu vấn tin được viết bằng SQL như sau:

```
SELECT   RESP, DUR
FROM     ASG, EMP_i
WHERE    ASG.ENO = EMP_i.ENO
```

trong đó *i* =1 hoặc *i* =2 tuỳ thuộc nơi đưa ra câu vấn tin. Phân mảnh dẫn xuất của ASG theo phân mảnh của EMP được định nghĩa dưới đây và cho ở bảng sau:

$$ASG_1 = ASG >< EMP_1$$
$$ASG_2 = ASG >< EMP_2$$

ASG_1

ENO	*PNO*	*RESP*	*DUR*
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E7	P3	Engineer	36

ASG_2

ENO	*PNO*	*RESP*	*DUR*
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E5	P2	Manager	24
E6	P4	Manager	48
E8	P3	Manager	40

Ví dụ này minh hoạ 2 điều:

(1) Phân mảnh dẫn xuất có thể xảy ra dây chuyền, trong đó một quan hệ được phân mảnh như hệ quả của phân mảnh quan hệ khác, và đến lượt nó làm cho các quan hệ khác phân mảnh (chẳng hạn như dây chuyền PAY→EMP→ASG).

(2) Thông thường một quan hệ có nhiều cách phân mảnh (chẳng hạn như ASG). Chọn một lược đồ phân mảnh là bài toán quyết định và sẽ được phân tích trong phần cấp phát.

d) Kiểm định tính đúng đắn

Bây giờ ta cần kiểm tra các tiêu chuẩn đối với các thuật toán.

• *Tính đầy đủ*

Tính đầy đủ của phân mảnh ngang nguyên thuỷ dựa vào các vị từ chọn được dùng. Với điều kiện các vị từ chọn là đầy đủ, phân mảnh thu được cũng đảm bảo đầy đủ. Bởi vì cơ sở của thuật toán phân mảnh là tập các vị từ cực tiểu và đầy đủ *Pr'*, tính đầy đủ sẽ được đảm bảo.

Tính đầy đủ của phân mảnh ngang dẫn xuất có phức tạp hơn. Trước tiên chúng ta định nghĩa lại qui tắc đầy đủ.

Cho R là quan hệ thành viên của đường nối với quan hệ chủ S. Quan hệ S được phân mảnh thành $F_S = \{S_1, ..., S_k\}$. Gọi A là thuộc tính nối giữa R và S. Ký hiệu $\{R_1, ..., R_k\}$ là tập phân mảnh dẫn xuất của R theo F_S. Khi đó với mỗi bộ t của R_i thì phải có 1 bộ t' của S_i thoả t[A]=t'[A], với mọi i =1, ..., k.

Tính đầy đủ ở đây thực chất là đảm bảo *ràng buộc toàn vẹn tham chiếu*.

• *Tính tái thiết*

Tái thiết một quan hệ toàn cục từ các mảnh được thực hiện bằng toán tử hợp trong cả phân mảnh ngang nguyên thuỷ lẫn dẫn xuất.

$$R = \mathop{\mathrm{U}}_{R_i \in F} R_i$$

• *Tính tách rời*

Trong phân mảnh ngang nguyên thuỷ tính tách rời được đảm bảo nếu các vị từ tiểu hạng xác định phân mảnh có tính loại trừ tương hỗ.

Tuy nhiên phân mảnh dẫn xuất có hàm chứa các bán nối có phức tạp hơn. Tính tách rời được đảm bảo nếu đồ thị nối thuộc loại đơn giản. Nếu đồ thị nối không đơn giản thì phải xem xét các giá trị thực sự của phân mảnh.

◊ *Ví dụ*. Khi phân mảnh quan hệ PAY ở ví dụ trước, các vị từ tiểu hạng $M=\{m_1,m_2\}$ là

$$m_1 = SAL \leq 30000$$
$$m_2 = SAL > 30000$$

Vì m_1 và m_2 loại trừ tương hỗ, phân mảnh của PAY là tách biệt.

Tuy nhiên, với quan hệ EMP ta đòi hỏi

(i) Mỗi kỹ sư có một chức vụ duy nhất.
(ii) Mỗi chức vụ có một giá trị lương duy nhất đi kèm.

Vì hai qui tắc này suy ra từ ngữ nghĩa của CSDL, phân mảnh của EMP ứng với PAY cũng tách rời.

2.3.2. Phân mảnh dọc

Một phân mảnh dọc của quan hệ R là tập các mảnh R_1, ..., R_k, trong đó mỗi mảnh chứa tập con thuộc tính của R và các khoá của R. Mục đích của phân mảnh dọc là phân hoạch một quan hệ thành tập các quan hệ nhỏ hơn để nhiều ứng dụng chỉ cần chạy trên 1 mảnh. Phân mảnh tối ưu cho phép giảm tối đa thời gian thực thi các ứng dụng chạy trên các mảnh đó.

Phân mảnh dọc phức tạp hơn so với phân mảnh ngang, do số khả năng phân mảnh dọc rất lớn. Trong phân mảnh ngang, nếu tổng số vị từ đơn giản trong *Pr* là *n* thì có 2^n vị từ tiểu hạng có thể định nghĩa trên nó. Ngoài ra có thể loại bỏ một số lớn trong đó mâu thuẫn với phép kéo theo hiện có, như vậy sẽ làm giảm đi số mảnh dự tuyển. Trong trường hợp phân mảnh dọc, nếu quan hệ có *m* thuộc tính không khoá, thì số mảnh có thể bằng *B*(*m*), *số Bell* thứ *m*. Với *m* lớn $B(m) \approx m^m$, chẳng hạn với $B(10) \approx 115000$, $B(15) \approx 10^9$, $B(30) \approx 10^{23}$.

Những giá trị này cho thấy, nỗ lực tìm lời giải tối ưu cho bài toán phân hoạch dọc không hiệu quả; vì thế phải dùng đến các phương pháp *heuristic*. Chúng ta nêu ở đây hai loại heuristic cho phân mảnh dọc các quan hệ toàn cục.

(1) *Nhóm thuộc tính*: Bắt đầu bằng cách gán mỗi thuộc tính cho một mảnh, và tại mỗi bước nối số mảnh lại cho đến khi thoả tiêu chuẩn nào đó.

(2) *Tách mảnh*: Bắt đầu bằng một quan hệ và quyết định cách phân hoạch có lợi dựa trên hành vi truy xuất của các ứng dụng trên các thuộc tính.

Trong phần tiếp chúng ta chỉ thảo luận kỹ thuật tách mảnh, vì nó thích hợp phương pháp thiết kế từ trên xuống hơn và giải pháp tối ưu gần với quan hệ đầy đủ hơn là tập các mảnh chỉ có 1 thuộc tính. Hơn nữa kỹ thuật tách mảnh sinh ra các mảnh với các thuộc tính không khoá không chồng nhau.

Việc nhân bản khoá chính cho các mảnh là đặc trưng của phân mảnh dọc, cho phép tái thiết quan hệ toàn cục. Vì thế phương pháp tách mảnh chỉ đề cập đến các thuộc tính không khoá.

Mặc dù có những vấn đề phát sinh, nhân bản các thuộc tính khoá có ưu điểm nổi bật là duy trì tính toàn vẹn ngữ nghĩa.

Một chọn lựa khác đối với nhân bản các thuộc tính khoá là sử dụng một *mã định bộ* (TID - tuple identifier). Đây là giá trị duy nhất được hệ thống gán cho mỗi bộ của quan hệ.

a) Các yêu cầu thông tin của phân mảnh dọc

Những thông tin chính cần cho phân mảnh dọc có liên quan tới các ứng dụng. Vì phân mảnh dọc đặt vào một mảnh các thuộc tính thường được truy xuất chung với nhau, ta cần xác định một độ đo khái niệm "chung với nhau". Số đo này gọi là *ái lực* (*affinity*) của thuộc tính, chỉ mức độ liên đới giữa các thuộc tính.

Thông số quan trọng về dữ liệu có liên quan đến các ứng dụng là *tần số truy xuất* (*access frequency*) của chúng. Gọi $Q = \{q_1, ..., q_k\}$ là tập các vấn tin của người dùng sẽ chạy trên quan hệ $R(A_1, ..., A_n)$. Với mỗi câu vấn tin q_i và mỗi thuộc tính A_j ta định nghĩa *giá trị sử dụng thuộc tính* (*attribute usage value*), ký hiệu $use(q_i, A_j)$, như sau

$$use(q_i, A_j) = \begin{cases} 1, \text{ nếu } q_i \text{ tham chiếu thuộc tính } A_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

Các vectơ $use(q_i, \bullet)$ cho mỗi ứng dụng sẽ được xác định nếu nhà thiết kế biết được các ứng dụng chạy trên cơ sở dữ liệu. Cần nhắc lại rằng *qui tắc 80/20* rất có ích cho công việc này.

◊ *Ví dụ.* Xét quan hệ PROJ của ví dụ phần trước. Giả sử các ứng dụng sau đây chạy trên quan hệ đó.

q_1 : Tìm ngân sách của dự án, cho biết mã số dự án.

```
SELECT   BUDGET
FROM     PROJ
WHERE    PNO = Value
```

q_2 : Tìm tên và ngân sách của tất cả dự án.

```
SELECT   PNAME, BUDGET
FROM     PROJ
```

q_3 : Tìm tên của các dự án được thực hiện ở thành phố đã cho

```
SELECT   PNAME
FROM     PROJ
WHERE    LOC = Value
```

q_4 : Tìm tổng ngân sách các dự án được thực hiện ở thành phố đã cho

```
SELECT   SUM(BUDGET)
FROM     PROJ
WHERE    LOC = Value
```

Dựa theo 4 ứng dụng này ta có thể xác định được giá trị sử dụng các thuộc tính. Để cho đơn giản ta ký hiệu

$$A_1 = PNO,\ A_2 = PNAME,\ A_3 = BUDGET \text{ và } A_4 = LOC.$$

Giá trị sử dụng cho ở ma trận sau (phần tử ô [i, j] chính là $use(q_i, A_j)$) :

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Giá trị sử dụng thuộc tính không đủ thông tin để làm cơ sở cho việc tách và phân mảnh. Lý do là chúng không biểu thị độ lớn của tần số ứng dụng. Số đo tần số có thể được chứa trong định nghĩa về số đo ái lực thuộc tính $aff(A_i, A_j)$, biểu thị cho *cầu nối* (*bond*) giữa hai thuộc tính của một quan hệ theo cách chúng được các ứng dụng truy xuất.

Cho quan hệ R(A_1, ..., A_n) và tập ứng dụng Q={q_1, ..., q_k}. Với mỗi cặp thuộc tính (A_i, A_j) ký hiệu

S_l là vị trí thứ l, l = 1, ..., L

$ref_l(q_k)$ là số lần truy xuất đến A_i, A_j cho mỗi lần thực hiện ứng dụng q_k tại vị trí S_l.

$acc_l(q_k)$ là số đo tần số thực hiện ứng dụng q_k tại vị trí S_l.

$Q(i, j) = \{k \mid use(q_k, A_i) = 1 \ \& \ use(q_k, A_j) = 1\}$

Số đo ái lực thuộc tính giữa hai thuộc tính A_i và A_j của quan hệ R(A_1 , ..., A_n) ứng với tập ứng dụng Q={q_1, ..., q_k} được xác định theo công thức

$$aff(A_i, A_j) = \sum_{k \in Q(i,j)} \sum_{l=1}^{L} ref_l(q_k).acc_l(q_k)$$

Kết quả là ma trận vuông cấp n gọi là *ma trận ái lực thuộc tính* (*AA - attribute affinity matrix*).

◊ *Ví dụ.* Ta tiếp tục ví dụ trên. Để đơn giản ta giả sử rằng có 3 vị trí S_1, S_2 và S_3 và 4 ứng dụng và $ref_l(q_k)$ =1 với mọi S_l và mọi q_k. Cho tần số sử dụng như sau

$$\begin{array}{lll} acc_1(q_1) = 15; & acc_2(q_1) = 20; & acc_3(q_1) = 10; \\ acc_1(q_2) = 5; & acc_2(q_2) = 0; & acc_3(q_2) = 0; \\ acc_1(q_3) = 25; & acc_2(q_3) = 25; & acc_3(q_3) = 25; \\ acc_1(q_4) = 3; & acc_2(q_4) = 0; & acc_3(q_4) = 0; \end{array}$$

Vì chỉ có ứng dụng q_1 truy xuất đến cả hai thuộc tính A_1 và A_3, nên ta có

$$aff(A_1, A_3) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

Tương tự ta tính được ma trận ái lực sau

$$AA = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{pmatrix} \end{array}$$

b) Thuật toán tụ nhóm

Nhiệm vụ cơ bản trong việc xây dựng thuật toán phân mảnh dọc là tìm tiêu chí nào đó để nhóm các thuộc tính của quan hệ dựa trên ma trận ái lực. *Thuật toán năng lượng nối BEA* (*bond energy algorithm*) là thích hợp vì những lý do sau:

(1) BEA gom các thuộc tính có cùng độ lớn ái lực lại với nhau.

(2) Các kết quả tụ nhóm không bị ảnh hưởng bởi thứ tự đưa các thuộc tính vào thuật toán.

(3) Độ phức tập tính toán $O(n^2)$, với n là số thuộc tính, là chấp nhận được.

(4) Có thể xác định mối liên hệ giữa các nhóm thuộc tính.

Thuật toán năng lượng nối BEA hoán vị các hàng và cột ma trận ái lực để tạo thành *ma trận ái lực tụ CA* (*clustered affinity matrix*). Hoán vị được thực hiện để cực đại *số đo ái lực chung AM* (*global affinity measure*):

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) + aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j)]$$

trong đó

$$aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0 \quad \forall i,j$$

Vì ma trận ái lực *AA* đối xứng nên hàm mục tiêu *AM* có thể rút gọn như sau

$$AM = \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})]$$

Quá trình sinh *ma trận ái lực tụ CA* được thực hiện qua 3 bước.

(1) *Khởi tạo*. Đặt cố định một trong các cột của *AA* vào *CA*, chẳng hạn cột 1. Gán số cột trong *CA* i := 1.

(2) *Thực hiện lặp*. Lấy lần lượt một trong $n-i$ cột còn lại trong *AA* đặt vào $i+1$ vị trí giữa i cột trong *CA* (kể cả hai đầu). Chọn nơi đặt sao cho nó làm tăng nhiều nhất số đo ái lực được mô tả ở trên. Tăng $i := i+1$.
Tiếp tục bước này cho đến khi $i = n$.

(3) *Sắp thứ tự hàng*. Một khi thứ tự các cột đã được xác định, các hàng cũng sắp thứ tự lại tương ứng.

Để bước (2) của thuật toán hoạt động chúng ta cần định nghĩa xem *đóng góp* (*contribution*) của thuộc tính vào số đo ái lực mang ý nghĩa gì.
Ta có

$$\begin{aligned} AM &= \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \left[aff(A_i, A_j)aff(A_i, A_{j-1}) + aff(A_i, A_j)aff(A_i, A_{j+1})\right] \end{aligned}$$

Ta định nghĩa *cầu nối* (*bond*) giữa hai thuộc tính A_x và A_y như sau

$$bond(A_x, A_y) = \sum_{z=1}^{n} aff(A_z, A_x)aff(A_z, A_y)$$

Như vậy *AM* có thể viết lại là

$$AM = \sum_{j=1}^{n} \left[bond(A_j, A_{j-1}) + bond(A_j, A_{j+1})\right]$$

Bây giờ xét n thuộc tính sau

$$\underbrace{A_1, A_2, \ldots, A_{i-1}}_{AM'} \underbrace{A_i A_j A_{j+1}, \ldots, A_n}_{AM''}$$

Số đo ái lực chung cho các thuộc tính này có thể viết như sau:

$$AM_{old} = AM' + AM'' + bond(A_{i-1}, A_i) + bond(A_i, A_j) + bond(A_j, A_i) + bond(A_j, A_{j+1})$$

$$= \sum_{l=1}^{i-1}[bond(A_l, A_{l-1}) + bond(A_l, A_{l+1})] + \sum_{l=j+1}^{n}[bond(A_l, A_{l-1}) + bond(A_l, A_{l+1})] + 2bond(A_i, A_j)$$

Bây giờ xét đến việc đặt thuộc tính mới A_k vào giữa thuộc tính A_i và A_j trong ma trận ái lực tụ:

$$\underbrace{A_1, A_2, \ldots, A_{i-1}}_{AM'} \underbrace{A_i A_k A_j A_{j+1}, \ldots, A_n}_{AM''}$$

Số đo ái lực chung mới có thể biểu diễn tương tự như sau:

$$AM_{new} = AM' + AM'' + bond(A_i, A_k) + bond(A_k, A_i) + bond(A_k, A_j) + bond(A_j, A_k)$$
$$= AM' + AM'' + 2bond(A_i, A_k) + 2bond(A_k, A_j)$$

Vì thế đóng góp thực (net contribution) cho số đo ái lực chung khi đặt thuộc tính A_k vào giữa thuộc tính A_i và A_j là

$$cont(A_i, A_k, A_j) = AM_{new} - AM_{old} = 2bond(A_i, A_k) + 2bond(A_k, A_j) - 2bond(A_i, A_j)$$

Sau đây là *thuật toán năng lượng nối* BEA

• **Thuật toán BEA**
Đầu vào: Ma trận ái lực *AA* cấp *n*.
Đầu ra: Ma trận ái lực tụ *CA*.
{*loc* là biến lưu vị trí nơi có giá trị đóng góp *cont* lớn nhất}
Begin
{khởi tạo}
CA(•,1) := *AA*(•,1); {đưa cột thứ nhất của *AA* vào cột thứ nhất của *CA*}
CA(•,2) := *AA*(•,2); {đưa cột thứ hai của *AA* vào cột thứ hai của *CA*}
index := 3;
while *index* ≤ *n* **do** {chọn vị trí tốt nhất cho cột AA_{index}}
begin
for *i:=1* to *index−1* **do**
begin
tính $cont(A_{i-1}, A_{index}, A_i)$
end {for}
tính $cont(A_{index-1}, A_{index}, A_{index+1})$

```
        loc := nơi có giá trị cont lớn nhất.
        for  j := index downto loc do
            CA(•, j) := CA(•, j−1)
            CA(•, loc) := AA(•, index)
        index := index + 1
      end;
  {sắp thứ tự các hàng theo thứ tự các cột}
End; {BEA}
```

◊ *Ví dụ*

Xét ma trận *AA* tính ở ví dụ trước

$$AA = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{pmatrix} \end{array}$$

Ta tính phần đóng góp khi di chuyển A_4 vào giữa các thuộc tính A_1 và A_2 được cho bằng công thức

$$cont(A_1, A_4, A_2) = 2bond(A_1, A_4) + 2bond(A_4, A_2) - 2bond(A_1, A_2)$$
$$= 2*135 + 2*11865 - 2*225 = 23550$$

vì

$$bond(A_1, A_4) = 45*0 + 0*75 + 45*3 + 0*78 = 135$$
$$bond(A_4, A_2) = 0*0 + 75*80 + 3*5 + 78*75 = 11865$$
$$bond(A_1, A_2) = 45*0 + 0*80 + 45*5 + 0*75 = 225$$

Trường hợp thuộc tính A_k đặt vào tận cùng bên trái hoặc tận cùng bên phải ta có

$$bond(A_0, A_k) = bond(A_k, A_{k+1}) = 0$$

◊*Ví dụ*

Ta xét tiếp ví dụ trên. Bây giờ ta gom tụ các thuộc tính của quan hệ PROJ và dùng ma trận ái lực *AA*.

Theo bước khởi đầu ta đưa cột 1 và cột 2 của *AA* vào *CA*

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} & & 45 & 0 \\ & & 5 & 75 \\ & & 53 & 3 \\ & & 3 & 78 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cc} A_1 & A_2 \end{array} \\ \begin{pmatrix} 45 & 0 \\ 0 & 80 \\ 45 & 5 \\ 0 & 75 \end{pmatrix} \end{array}$$

Tiếp theo ta di chuyển cột 3 của AA sang CA. Có 3 nơi có thể đặt cột 3:

- Bên trái cột 1 (0-3-1): ta có

  $bond(A_0,A_1) = bond(A_0,A_3) = 0$

$bond(A_3,A_1) = 45*45+5*0+53*45+3*0 = 4410$

suy ra

$cont(A_0,A_3,A_1) = 2\ bond(A_0,A_3) + 2bond(A_3,A_1) - 2\ bond(A_0,A_1) = 8820$

- Giữa cột 1 và 2 (1-3-2): ta có

$bond(A_1,A_3) = bond(A_3,A_1) = 4410$

$bond(A_3,A_2) = 890$

$bond(A_1,A_2) = 225$

suy ra

$cont(A_1,A_3,A_2) = 10150$

- Bên phải cột 2 (2-3-4): ta có

$bond(A_3,A_4) = bond(A_2,A_4) = 0$

$bond(A_1,A_4) = 890$

suy ra

$cont(A_2,A_3,A_4) = 1780$

Vì đóng góp của trường hợp thứ 2 là lớn nhất, ta chèn A_3 vào giữa A_1 và A_2, và có

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array}\begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} & & & 0 \\ & & & 75 \\ & & & 3 \\ & & & 78 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array}\begin{array}{c} \begin{array}{ccc} A_1 & A_3 & A_2 \end{array} \\ \begin{pmatrix} 45 & 45 & 0 \\ 0 & 5 & 80 \\ 45 & 53 & 5 \\ 0 & 3 & 75 \end{pmatrix} \end{array}$$

Tính toán tương tự cho A_4 chỉ ra rằng cần đặt nó bên phải A_2 .

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array}\begin{array}{c} \begin{array}{cccc} A_1 & A_3 & A_2 & A_4 \end{array} \\ \begin{pmatrix} 45 & 45 & 0 & 0 \\ 0 & 5 & 80 & 75 \\ 45 & 53 & 5 & 3 \\ 0 & 3 & 75 & 78 \end{pmatrix} \end{array}$$

Cuối cùng ta hoán chuyển thứ tự các hàng và được ma trận kết quả

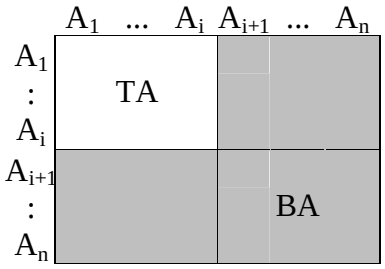
$$CA = \begin{array}{c} \\ A_1 \\ A_3 \\ A_2 \\ A_4 \end{array}\begin{array}{c} \begin{array}{cccc} A_1 & A_3 & A_2 & A_4 \end{array} \\ \begin{pmatrix} 45 & 45 & 0 & 0 \\ 45 & 53 & 5 & 3 \\ 0 & 5 & 80 & 75 \\ 0 & 3 & 75 & 78 \end{pmatrix} \end{array}$$

Ta thấy quá trình tạo ra hai tụ: một ở góc trên bên trái chứa các ái lực nhỏ, còn tụ kia ở góc dưới bên phải chứa các ái lực cao.

c) Thuật toán phân hoạch

Mục đích của hành động tách thuộc tính là tìm tập thuộc tính được truy xuất cùng nhau, hoặc với phần lớn các thuộc tính, bởi các tập ứng dụng riêng biệt. Thí dụ, nếu biết hai thuộc tính A_1 và A_2 chỉ được ứng dụng q_1 truy xuất và các thuộc tính A_3 và A_4 được các ứng dụng khác truy xuất, chẳng hạn q_2 và q_3, thì quyết định phân mảnh là đương nhiên. Vấn đề là tìm được thuật toán để xác định các nhóm này.

Xét ma trận thuộc tính tụ *CA*. Nếu một điểm nằm trên đường chéo được cố định, hai tập thuộc tính sẽ được xác định. Tập $\{A_1, ..., A_i\}$ ở góc trên trái và tập $\{A_{1+1}, ..., A_n\}$ ở góc dưới phải. Tập thứ nhất gọi là *đỉnh* (top) và ký hiệu *TA* , tập thứ hai gọi là *đáy* (bottom) và ký hiệu *BA*.



Bây giờ xét tập ứng dụng Q = $\{q_1, ..., q_k\}$ và định nghĩa các tập ứng dụng chỉ truy xuất TA, BA hoặc cả hai. Những tập này định nghĩa như sau

$$
\begin{aligned}
AQ(q_i) &= \{A_j \mid use(q_i, A_j) = 1\} \\
TQ &= \{q_j \mid AQ(q_i) \subseteq TA\} \\
BQ &= \{q_j \mid AQ(q_i) \subseteq BA\} \\
OQ &= Q - (TQ \cup BQ)
\end{aligned}
$$

Phương trình đầu định nghĩa tập thuộc tính được truy xuất bởi ứng dụng q_i ; TQ và BQ tương ứng là các tập ứng dụng chỉ truy xuất TA và BA, còn OQ là tập ứng dụng truy xuất cả hai tập TA và BA.

Ở đây nảy sinh bài toán tối ưu hoá. Nếu có *n* thuộc tính trong quan hệ, thì sẽ có $n-1$ vị trí khả hữu có thể là điểm phân chia trên đường chéo của ma trận thuộc tính tụ cho quan hệ đó. Vị trí tốt nhất để phân chia là vị trí sinh các tập TQ và BQ sao cho tổng các *truy xuất chỉ một mảnh* là lớn nhất, còn tổng các *truy xuất cả hai mảnh* là nhỏ nhất. Vì thế ta định nghĩa các phương trình chi phí như sau:

$$CQ = \sum_{q_i \in Q} \sum_{\forall S_j} ref_j(q_i) acc_j(q_i)$$

$$CTQ = \sum_{q_i \in TQ} \sum_{\forall S_j} ref_j(q_i) acc_j(q_i)$$

$$CBQ = \sum_{q_i \in BQ} \sum_{\forall S_j} ref_j(q_i) acc_j(q_i)$$

$$COQ = \sum_{q_i \in OQ} \sum_{\forall S_j} ref_j(q_i) acc_j(q_i)$$

Mỗi phương trình trên đếm tổng số truy xuất đến các thuộc tính bởi các ứng dụng trong các lớp tương ứng của chúng. Dựa trên số liệu này bài toán tối ưu hoá được định nghĩa là bài toán tìm điểm x ($1 \le x \le n$) sao cho biểu thức

$$z = CTQ*CBQ - COQ^2$$

lớn nhất.

Đặc trưng quan trọng của biểu thức này là nó định nghĩa hai mảnh sao cho giá trị của CTQ và CBQ càng gần bằng nhau càng tốt. Điều này cho phép cân bằng tải trọng xử lý khi các mảnh được phân tán đến các vị trí khác nhau. Thuật toán phân hoạch có độ phức tạp tuyến tính *O(n)*, theo số thuộc tính của quan hệ.

Để tăng thêm số phương án xét, tức hiệu quả thuật toán, chúng ta cần hiệu chỉnh thuật toán bằng thủ tục *xê dịch* (SHIFT) thuộc tính như sau. Sau mỗi bước lặp tìm vị trí tốt nhất cho một thứ tự thuộc tính, ta dịch cột tận trái sang thành cột tận phải, và hàng trên cùng xuống cuối cùng. Với thao tác xê dịch như thế này ta chỉ cần kểm tra $n-1$ vị trí trên đường chéo để tìm trị z lớn nhất. Độ phức tạp của thuật toán sẽ tăng thêm n lần, tức là $O(n^2)$.

Với giả thiết đã có thủ tục xê dịch SHIFT, ta có thuật toán phân hoạch PARTITION sau:

• **Thuật toán PARTITION**

Đầu vào: ma trận ái lực tụ *CA*
quan hệ *R*
ma trận sử dụng thuộc tính *ref*
ma trận tần số truy xuất *acc*

Đầu ra: tập các mảnh $F_R = \{R_1, R_2\}$. với $R_1 \cap R_2$ là khoá.

Begin
{xác định giá trị z cho cột thứ nhất}
{các chỉ mục trong phương trình chi phí chỉ ra điểm tách}
Tính CTQ_{n-1} ;
Tính CBQ_{n-1} ;
Tính COQ_{n-1} ;
gán *best* := $CTQ_{n-1} * CBQ_{n-1} - (COQ_{n-1})^2$;
do {xác định cách phân hoạch tốt nhất}
begin
for i := n–2 downto 1 do
begin
Tính CTQ_i ;
Tính CBQ_i ;

Tính COQ_i ;
gán $z := CTQ_i * CBQ_i - (COQ_i)^2$;
if $z > best$ *then*
begin
$best := z$;
ghi nhận điểm tách vào trong hành động xê dịch
end;
end;{for}
gọi SHIFT(CA);
end;
until không thể thực hiện SHIFT được nữa.
xây dựng lại ma trận theo vị trí xê dịch
$R_1 := \pi_{TA}(R) \cup K$ {K là tập thuộc tính khoá chính của R}
$R_2 := \pi_{BA}(R) \cup K$
$F := \{R_1, R_2\}$;
end.

◊ *Ví dụ*

Áp dụng thuật toán PARTITION cho ma trận *CA* từ quan hệ PROJ ở thí dụ trước. Kết quả là định nghĩa các mảnh $F_{PROJ} = \{PROJ_1, PROJ_2\}$, trong đó

$PROJ_1 = \{A_1, A_3\} = \{ PNO, BUDGET \}$
$PROJ_2 = \{A_1, A_2, A_4\} = \{ PNO, PNAME, LOC \}$

d) Kiểm tra tính đúng đắn

♦ *Tính đầy đủ*: được đảm bảo bằng thuật toán PARTITION vì mỗi thuộc tính của quan hệ toàn cục được đưa vào một trong các mảnh.

♦ *Tính tái thiết*: đảm bảo

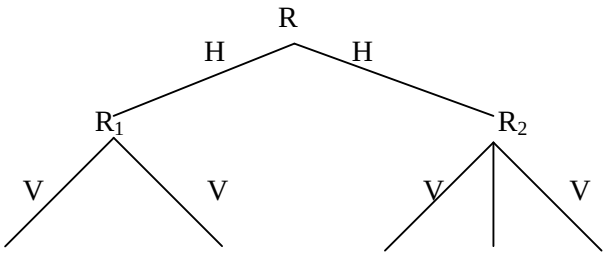
$$R = \ ><_K R_i \ \forall R_i \in F_R .$$

nếu mỗi R_i là đầy đủ và chứa thuộc tính khoá của R hoặc chứa *mã định bộ* (TID - tuple identifier) được gán bởi hệ thống.

♦ *Tính tách biệt*: đảm bảo tách biệt đối với các thuộc tính không khoá.

2.3.3. Phân mảnh hỗn hợp

Trong đa số trường hợp, phân mảnh ngang hoặc phân mảnh dọc đơn giản cho một lược đồ CSDL không đủ đáp ứng yêu cầu các ứng dụng. Khi đó phân mảnh dọc có thể được thực hiện sau một phân mảnh ngang hoặc ngược lại. Chiến lược này sinh ra một lối phân hoạch có cấu trúc cây và gọi là *phân mảnh hỗn hợp* (hybrid fragmentation).



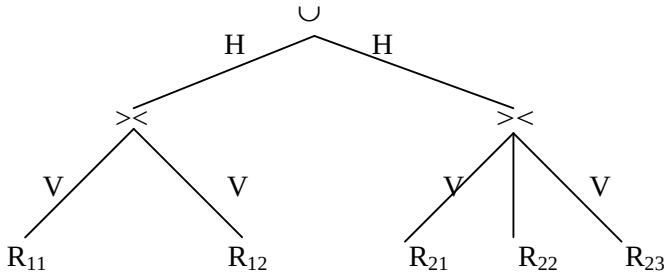
R_{11} R_{12} R_{21} R_{22} R_{23}

Phân mảnh hỗn hợp

Một ví dụ điển hình minh hoạ cho sự cần thiết phải có phân mảnh hỗn hợp là quan hệ PROJ. Trong một ví dụ trước ta đã phân hoạch nó thành sáu mảnh ngang dựa vào hai ứng dụng. Trong một ví dụ sau đó, chúng ta lại phân mảnh dọc PROJ thành hai mảnh. Như thế chúng ta có một tập các mảnh ngang, mỗi mảnh ngang lại được phân tiếp thành hai mảnh dọc.

Số mức lồng ghép có thể khá lớn, nhưng hữu hạn. Trong thực tế, do các quan hệ toàn cục đã chuẩn hoá và chi phí nối nhiều mảnh có thể quá cao, nên phần lớn người ta chỉ thực hiện tối đa hai mức lồng ghép.

Tính đúng đắn của phân mảnh hỗn hợp suy ra từ tính đúng đắn của phân mảnh ngang và phân mảnh dọc. Thí dụ để tái thiết quan hệ toàn cục, người ta bắt đầu tại các nút lá của cây phân hoạch và di chuyển lên trên bằng cách thực hiện các phép nối và hợp như ở hình sau



Tính tái thiết của phân mảnh hỗn hợp

2.4. Cấp phát

Cấp phát tài nguyên cho các nút của mạng máy tính là bài toán đã được nghiên cứu rộng rãi.

2.4.1. Bài toán cấp phát

Giả sử có tập các mảnh $FF = \{F_1, ..., F_n\}$ và mạng bao gồm các vị trí $SS = \{S_1,..., S_m\}$ trên đó có tập ứng dụng $QQ = \{q_1 ,..., q_l\}$ đang chạy.

Bài toán cấp phát (allocation problem) là tìm phân phối tối ưu của FF cho SS.

Một trong các điểm quan trọng là định nghĩa khái niệm *tối ưu* (*optimality*). Khái niệm tối ưu có thể định nghĩa ứng với hai số đo:

(1) *Chi phí nhỏ nhất*: Hàm chi phí gồm có chi phí lưu mỗi mảnh F_i tại vị trí S_j, chi phí vấn tin F_i tại vị trí S_j , chi phí cập nhật F_i tại tất cả mọi vị trí chứa nó và chi phí truyền dữ liệu. Bài toán cấp phát cố gắng tìm lược đồ cấp phát với hàm chi phí tổ hợp thấp nhất.

(2) *Hiệu năng*: Chiến lược cấp phát được thiết kế nhằm duy trì hiệu năng hữu lượng. Các chiến lược đã biết là hạ thấp thời gian đáp ứng và tăng tối đa lưu lượng hệ thống tại mỗi vị trí.

Một lược đồ cấp phát hoàn hảo là trả lời câu vấn tin trong thời gian ngắn nhất mà vẫn duy trì chi phí xử lý thấp nhất. Do tính phức tạp của bài toán, nên người ta mới chỉ giải quyết những trường hợp đơn giản.

Chúng ta xét một trường hợp đơn giản. Cố định mảnh $F \in FF$. Chúng ta đưa ra một số giả thiết và định nghĩa nhằm mô hình hoá bài toán cấp phát này.

(1) Giả sử có thể sửa đổi QQ để xác định được các *vấn tin cập nhật* (update query) và *vấn tin chỉ đọc* (retrieval-only query), và để định nghĩa các đại lượng sau đây cho mảnh F

$$T = \{t_1,...,t_m\}$$

với t_i là lưu lượng chỉ đọc được sinh ra tại S_i cho mảnh F và

$$U = \{u_1,...,u_m\}$$

với u_i là lưu lượng cập nhật được sinh ra tại S_i cho mảnh F .

(2) Giả sử chi phí truyền giữa hai vị trí S_i và S_j là cố định với mỗi đơn vị truyền. Ngoài ra cũng giả sử rằng chúng khác nhau khi cập nhật và khi truy xuất chỉ đọc. Ta định nghĩa

c_{ij} là chi phí truyền 1 đơn vị đối với các yêu cầu *chỉ đọc* giữa vị trí S_i và S_j ($i,j=1, ..., m$);

c'_{ij} là chi phí truyền 1 đơn vị đối với các yêu cầu *cập nhật* giữa vị trí S_i và S_j ($i, j=1, ..., m$);

và ký hiệu

$$C(T) = \{c_{12}, ..., c_{1m}, ..., c_{m-1,m}\}$$
$$C'(U) = \{c'_{12}, ..., c'_{1m}, ..., c'_{m-1,m}\}$$

(3) Gọi chi phí lưu trữ mảnh F tại vị trí S_i là d_i và ký hiệu

$$D = \{d_1, ..., d_m\}$$

(4) Giả thiết không có ràng buộc về khả năng lưu trữ cho các vị trí hoặc cho các đường truyền.

Khi đó bài toán cấp phát có thể được đặc tả là bài toán cực tiểu hoá chi phí, trong đó ta tìm tập $I \subseteq SS$ các vị trí lưu các bản sao của mảnh F. Ký hiệu x_i là *biến quyết định* (decision variable) chọn nơi đặt sao cho

$$x_i = \begin{cases} 1, \text{ nếu } F \text{ phân cho } S_i \\ 0, \text{ nếu ngược lại} \end{cases}$$

Bài toán cấp phát được phát biểu chính xác như sau

$$\min_{I \subset SS} \left[\sum_{i=1}^{m} \left(\sum_{S_j \in I} x_j u_j c_{ij} + t_i \min_{S_j \in I} c_{ij} \right) + \sum_{S_j \in I} x_j d_j \right]$$

Số hạng thứ nhất tương ứng với chi phí truyền các cập nhật từ các vị trí (S_i) đến mọi vị trí có giữ bản sao của mảnh F và với chi phí thực hiện các yêu cầu chỉ đọc tại vị trí đó (S_i) nhưng với chi phí truyền dữ liệu thấp nhất. Số hạng thứ hai tính tổng chi phí lưu tất cả bản sao của mảnh F.

Cách đặt vấn đề hết sức đơn giản và không phù hợp lắm với thực tế thiết kế cơ sở dữ liệu phân tán. Đây thực chất là mô hình *bài toán cấp phát tập tin* - FAP (*file allocation problem*) cho mạng máy tính. Để phân biệt với bài toán trên, ta gọi bài toán cấp phát mảnh trong thiết kế cơ sở dữ liệu phân tán là *bài toán cấp phát cơ sở dữ liệu* - DAP (*database allocation problem*). Trong thực tế, vì bài toán quá phức tạp nên ta chỉ nghiên cứu các *phương pháp heuristic* để tìm lời giải gần tối ưu.

2.4.2. Yêu cầu về thông tin

Ở giai đoạn cấp phát, ta cần thông tin định lượng về cơ sở dữ liệu, về các ứng dụng chạy trên đó, về cấu trúc mạng, khả năng xử lý và giới hạn lưu trữ của mỗi vị trí trên mạng.

a) Thông tin về cơ sở dữ liệu

Cho mảnh F_j và ứng dụng q_i. Độ tuyển của F_j ứng với ứng dụng q_i, ký hiệu $sel_i(F_j)$, là số lượng các bộ của F_j mà q_i truy xuất để xử lý.

Kích thước của mảnh F_j cũng được định nghĩa thông qua lực lượng $card(F_j)$ và độ dài $length(F_j)$ như sau

$$size(F_j) = card(F_j) * length(F_j)$$

b) Thông tin về ứng dụng

Phần lớn các thông tin về ứng dụng đều đã được biên dịch trong khi thực hiện phân mảnh. Hai số liệu quan trọng là

RR_{ij} số truy xuất chỉ đọc bởi câu vấn tin q_i thực hiện trên mảnh F_j,
UR_{ij} số truy xuất cập nhật bởi câu vấn tin q_i thực hiện trên mảnh F_j.

Ta cũng định nghĩa hai ma trận UM và RM với các phần tử tương ứng u_{ij} và r_{ij} như sau

$$u_{ij} = \begin{cases} 1, \text{ nếu } q_i \text{ cập nhật } F_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

$$r_{ij} = \begin{cases} 1, \text{ nếu } q_i \text{ cần đọc } F_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

Tiếp theo, vectơ $O = \{o(1),..., o(k)\}$ gồm các thành phần $o(i)$ đặc tả vị trí đưa ra câu vấn tin q_i (i = 1, .., k).

Cuối cùng để định nghĩa ràng buộc thời gian đáp ứng, thời gian đáp ứng tối đa được phép của mỗi ứng dụng cũng cần được đặc tả.

c) Thông tin về vị trí

Với mỗi vị trí, chúng ta cần biết về khả năng lưu trữ và xử lý của nó. Ta ký hiệu

USC_k là chi phí lưu 1 đơn vị dữ liệu tại S_k.
LPC_k là chi phí xử lý 1 đơn vị dữ liệu tại S_k.

d) Thông tin về mạng

Chi phí truyền dữ liệu được định nghĩa theo đơn vị là *bó dữ liệu* (frame). Ký hiệu

g_{ij} là chi phí truyền 1 bó dữ liệu giữa hai vị trí S_i và S_j.

2.4.3. Mô hình cấp phát

Chúng ta sẽ thảo luận một mô hình cấp phát có mục tiêu là giảm thiểu tổng chi phí xử lý và lưu trữ trong khi vẫn cố gắng đáp ứng được các đòi hỏi về thời gian đáp ứng. Mô hình có dạng sau

min (Tổng chi phí)

với ràng buộc

ràng buộc thời gian đáp ứng, ràng buộc lưu trữ, ràng buộc xử lý.

Ta sẽ khai triển các thành phần của mô hình này. Ký hiệu biến quyết định

$$x_{ij} = \begin{cases} 1, \text{ nếu } F_i \text{ phân cho } S_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

- ***Tổng chi phí***

Hàm tổng chi phí có hai thành phần: phần xử lý vấn tin và phần lưu trữ. Vì vậy nó có thể biểu diễn là

$$TOC = \sum_{q_i \in QQ} QPC_i + \sum_{S_k \in SS} \sum_{F_j \in FF} STC_{jk}$$

với

QPC_i là chi phí xử lý câu vấn tin của ứng dụng q_i

STC_{jk} là chi phí lưu mảnh F_j tại vị trí S_k.

Ta xét chi phí lưu trữ trước. Nó được cho bởi

$$STC_{jk} = USC_k * size(F_j) * x_{jk}$$

và hai ký hiệu tổng là tìm các tổng chi phí lưu trữ tại tất cả vị trí cho tất cả các mảnh.

Chi phí xử lý vấn tin khó xác định hơn. Ta tách QPC_i thành hai thành phần chi phí xử lý *PC* và chi phí truyền *TC*

$$QPC_i = PC_i + TC_i$$

Thành phần xử lý *PC* gồm 3 hệ số chi phí: chi phí truy xuất *AC,* chi phí toàn vẹn *IE* và chi phí điều khiển đồng thời *CC*:

$$PC_i = AC_i + IE_i + CC_i$$

Mô tả chi tiết cho mỗi hệ số chi phí phụ thuộc thuật toán dùng để hoàn tất các tác vụ đó. Chi tiết về chi phí truy xuất AC như sau

$$AC_i = \sum_{S_k \in SS} \sum_{F_j \in FF} \left(u_{ij} * UR_{ij} + r_{ij} * RR_{ij} \right) * x_{jk} * LPC_k$$

Hai số hạng đầu trong công thức trên tính số truy xuất của vấn tin q_i đến mảnh F_j. Chú ý rằng

$$\left(u_{ij} * UR_{ij} + r_{ij} * RR_{ij}\right)$$

là tổng số các truy xuất cập nhật và đọc mảnh F_j. Giả thiết chi phí xử lý là giống nhau với mọi mảnh. Ký hiệu tổng cho biết tổng số các truy xuất cho tất cả các mảnh được q_i tham chiếu. Nhân với LPC_k cho ra chi phí của truy xuất này tại vị trí S_k. Ta dùng x_{jk} để tách giá trị chi phí cho các vị trí có lưu các mảnh.

Hệ số chi phí duy trì tính toàn vẹn cơ sở dữ liệu có thể mô tả giống thành phần xử lý ngoại trừ chi phí xử lý cục bộ một đơn vị cần được thay đổi nhằm phản ánh chi phí thực sự để duy trì tính toàn vẹn. Ta sẽ quay lại vấn đề này ở chương sau.

Hàm chi phí truyền dữ liệu có thể biểu diễn giống hàm chi phí truy xuất. Tuy nhiên tổng chi phí truyền dữ liệu cho cập nhật và cho yêu cầu chỉ đọc sẽ khác nhau hoàn toàn. Trong vấn tin cập nhật, ta cần cho tất cả mọi vị trí biết nơi có các bản sao, còn trong vấn tin chỉ đọc thì chỉ cần truy xuất đến 1 trong các bản sao là đủ. Ngoài ra vào lúc kết thúc yêu cầu vấn tin cập nhật thì không cần truyền dữ liệu ngược lại cho vị trí đưa ra vấn tin ngoài một thông báo xác nhận, còn trong vấn tin chỉ đọc có thể phải có nhiều thông báo truyền dữ liệu.

Thành phần cập nhật hàm truyền dữ liệu là

$$TCU_i = \sum_{S_k \in SS} \sum_{F_j \in FF} u_{ij} * x_{jk} * g_{o(i),k} + \sum_{S_k \in SS} \sum_{F_j \in FF} u_{ij} * x_{jk} * g_{k,o(i)}$$

Số hạng thứ nhất để gửi thông báo cập nhật từ vị trí gốc $o(i)$ của q_i đến tất cả bản sao cần cập nhật. Số hạng thứ hai dành cho thông báo xác nhận.

Thành phần chi phí chỉ đọc là

$$TCR_i = \sum_{F_j \in FF} \min_{S_k \in SS} \left(r_{ij} * x_{jk} * g_{o(i),k} + r_{ij} * x_{jk} * \frac{sel_i(F_j) * length(F_j)}{fsize} * g_{k,o(i)} \right)$$

Số hạng thứ nhất trong TCR biểu thị chi phí truyền yêu cầu chỉ đọc đến những vị trí có bản sao của mảnh cần truy xuất. Số hạng thứ hai biểu thị chi phí để truyền các kết quả từ những vị trí này đến vị trí yêu cầu. Phương trình này khẳng định rằng trong số các vị trí có bản sao của cùng 1 mảnh, chỉ vị trí có tổng chi phí truyền thấp nhất mới được chọn để thực hiện thao tác này.

Bây giờ hàm chi phí truyền dữ liệu của câu vấn tin q_i có thể được tính là

$$TC_i = TCU_i + TCR_i$$

- ***Ràng buộc***

Các hàm ràng buộc có thể được đặc tả tương tự. Tuy nhiên thay vì mô tả những hàm này kỹ lưỡng, chúng ta chỉ nêu những ý tổng quát.

Ràng buộc thời gian đáp ứng cần được đặc tả là

$$\textit{thời gian thực thi của } q_i \leq \textit{thời gian đáp ứng lớn nhất của } q_i\ , \forall q_i \in QQ.$$

Người ta thường đặc tả số đo chi phí của hàm theo thời gian vì nó đơn giản.

Ràng buộc lưu trữ là

$$\sum_{F_j \in FF} STC_{jk} \leq \textit{khả năng lưu trữ tại vị trí } S_k, \forall S_k \in SS$$

Ràng buộc xử lý là

$$\sum_{q_i \in QQ} \textit{tải trọng xử lý của } q_i \textit{ tại } S_k \leq \textit{khả năng xử lý tại vị trí } S_k, \forall S_k \in SS$$

Mô hình bài toán cấp phát có lời giải phi đa thức, vì thế người ta tìm các phương pháp *heuristic* cho lời giải gần tối ưu. Có sự tương ứng giữa bài toán cấp phát và bài toán chọn vị trí đặt thiết bị đã được khám phá trong các nghiên cứu về quá trình điều hành sản xuất.

BÀI TẬP

1. Cho quan hệ

EMP

ENO	ENAME	TITLE
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

Ký hiệu p_1 là vị từ TITLE < "Programmer" và p_2 là vị từ TITLE > "Programmer". Giả thiết chuỗi ký tự được sắp theo thứ tự từ điển.

a) Thực hiện phân mảnh ngang cho quan hệ EMP ứng với $\{p_1, p_2\}$.

b) Giải thích tại sao phân mảnh kết quả $\{EMP_1, EMP_2\}$ không đáp ứng quy tắc đúng đắn của phân mảnh.

c) Sửa lại các vị từ p_1 và p_2 để chúng phân hoạch EMP theo các quy tắc đúng đắn của phân mảnh.

Hướng dẫn. Sửa các vị từ bằng cách xây dựng các vị từ tiểu hạng và suy ra các phép kéo theo tương ứng với thực hiện phân mảnh ngang của EMP dựa trên các vị từ tiểu hạng này. Cuối cùng chứng tỏ kết quả có tính đầy đủ, tính tái thiết và tính tách biệt.

2. Xét quan hệ

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

Giả sử có hai ứng dụng truy xuất ASG. ứng dụng thứ nhất được đưa ra tại 5 vị trí và muốn tìm thời gian phân công các nhân viên khi biết mã số nhân viên. Giả sử rằng nhà quản lý (Manager), nhà tư vấn (Consultant), kỹ sư (Engineer) và lập trình viên (Programmer) được lưu tai 4 vị trí khác nhau. ứng dụng thứ hai được đưa ra tại 2 vị trí trong đó các nhân viên có thời gian phân công dưới 20 tháng được quản lý tại một vị trí còn nhứng người trên 20 tháng được quản lý tại vị trí thứ hai. Hãy phân mảnh ngang nguyên thủy cho ASG theo các thông tin trên.

3. Xét các quan hệ sau

EMP

ENO	ENAME	TITLE
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

PAY

TITLE	SAL
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000

Giả sử EMP và PAY được phân mảnh ngang như sau:

$EMP_1 = \sigma_{TITLE="Elect.Eng."}(EMP)$
$EMP_2 = \sigma_{TITLE="Syst.Anal"}(EMP)$
$EMP_3 = \sigma_{TITLE="Mech.Eng."}(EMP)$
$EMP_4 = \sigma_{TITLE="Programmer."}(EMP)$
$PAY_1 = \sigma_{SAL >=30000}(PAY)$
$PAY_2 = \sigma_{SAL < 30000}(PAY)$

Vẽ đồ thị nối nửa của $E ><_{TITLE} PAY$. Đây là đồ thị đơn giản hay phân hoạch ? Nếu nó phân hoạch, hãy sửa lại phân mảnh của EMP hoặc PAY để có đồ thị nối nửa $E ><_{TITLE} PAY$ đơn giản.

4. Xét quan hệ PAY và EMP ở bài tập trên. Ký hiệu p_1 là vị từ SAL < 30000 và p_2 là vị từ SAL >= 30000. Thực hiện phân mảnh ngang cho PAY ứng với p_1 và p_2 để có được hai mảnh PAY_1 và PAY_2. Sử dụng phân mảnh của PAY thực hiện phân mảnh ngang dẫn xuất cho EMP. Chứng tỏ tính đầy đủ, tính tái thiết và tính tách biệt của phân mảnh EMP.

5. Xét các quan hệ EMP, ASG ở các bài tập trên. Giả sử định nghĩa khung nhìn sau

```
CREATE   VIEW    EMPVIEW(ENO, ENAME, PNO, RESP)
AS       SELECT  EMP.ENO, EMP.ENAME, ASG.PNO, ASG.RESP
         FROM    EMP, ASG
         WHERE   EMP.ENO = ASG.ENO
           AND   ASG.DUR = 24
```

được truy xuất bởi ứng dụng q_1 nằm tại các vị trí 1 và 2 với tần suất lần lượt là 10 và 20. Giả sử tiếp rằng có một vấn tin q_2 khác được định nghĩa là

```
SELECT   ENO, DUR
FROM     ASG
```

chạy tại các vị trí 2 và 3 với tần số tương ứng là 20 và 10. Dựa trên những thông tin này hãy xây dựng ma trận $use(q_i, A_j)$ cho các thuộc tính của cả hai quan hệ

EMP và ASG. Xây dựng ma trận ái lực chứa tất cả các thuộc tính của EMP và ASG. Cuối cùng, hãy biến đổi ma trận ái lực để có thể dùng nó khi tách các quan hệ này thành hai mảnh dọc bằng một heuristic hoặc thuật toán BEA.

6. Giả sử môi trường làm việc như ở bài tập 5. Giả sử 60% truy xuất của q_1 là cập nhật đến PNO và RESP của khung nhìn EMPVIEW, trong khi ASG.DUR không được cập nhật qua khung nhìn EMPVIEW. Ngoài ra, giả sử rằng tốc độ truyền dữ liệu giữa vị trí 1 và vị trí 2 chỉ bằng một nữa tốc độ truyền dữ liệu giữa vị trí 2 và vị trí 3. Dựa vào những thông tin trên hãy tìm một phân mảnh hợp lý của ASG và EMP và cách nhân bản và vị trí đặt dữ liệu tối ưu cho các mảnh, giả thiết rằng chi phí lưu trữ không đáng kể, nhưng các bản sao phải nhất quán.

Hướng dẫn. Xét phân mảnh ngang cho ASG dựa trên vị từ DUR=24 và phân mảnh ngang dẫn xuất tương ứng cho EMP. Cần xemét ma trận ái lực nhận được trong bài tập 5 cho EMP và ASG và xét xem nó có ý nghĩa gì không trong việc thực hiện phân mảnh dọc cho ASG.

7. Cho thí dụ về ma trận CA, trong đó điểm tách không duy nhất và phân hoạch nằm ngay giữa ma trận. Cho biết số các thao tác xê dịch cần thực hiện để có điểm tách duy nhất.

8. Gọi $Q = \{q_1, q_2, q_3, q_4, q_5\}$ là tập vấn tin, $A = \{A_1, A_2, A_3, A_4, A_5\}$ là tập thuộc tính và $S = \{S_1, S_2, S_3\}$ là tập vị trí. Ma trận giá trị sử dụng các thuộc tính và ma trận tần số truy xuất ứng dụng lần lượt là

	A_1	A_2	A_3	A_4	A_5
q_1	1	1	1	0	1
q_2	1	1	1	0	1
q_3	1	0	0	1	1
q_4	0	0	1	0	0
q_5	1	1	1	0	0

và

	S_1	S_2	S_3	
q_1	10	20	0	q_1
q_2	5	0	10	q_2
q_3	0	35	5	q_3
q_4	0	10	0	q_4
q_5	0	15	0	q_5

Giả thiết $ref_l(q_k) = 1$ cho mọi q_k và S_l, A_1 là thuộc tính khóa. Sử dụng các thuật toán năng lượng nối và phân hoạch dọc để có được một phân mảnh dọc cho tập thuộc tính trong A.

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 =

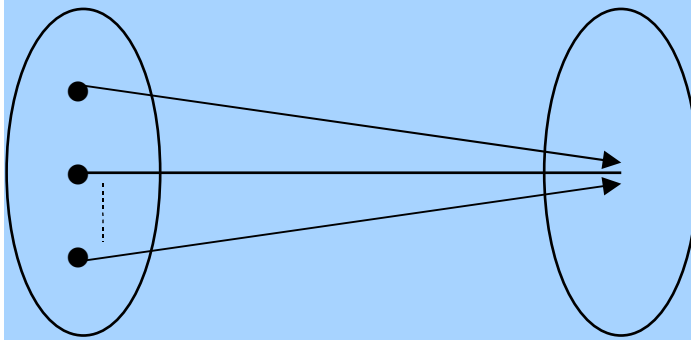
a	b
0	0
1	1
2	1
3	2

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 :

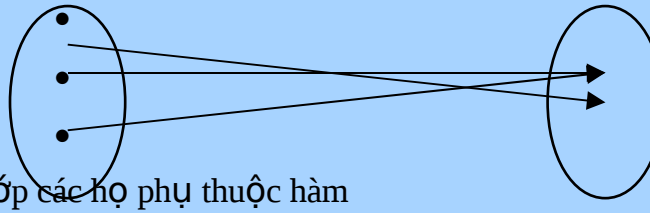
- Có thể cho ví dụ chỉ ra rằng có nhiều sơ đồ quan hệ khác nhau nhưng tập các khoá tối tiểu của chúng giống nhau. Có nghĩa là tồn tại

$s_1 = <R,F_1>, ..., s_t = <R,F_t>$ ($2 \leq t$) 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

$$\sqrt{2|K^{-1}|}\ 2 \le s(K) \le |K^{-1}|+1.$$

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 $\sqrt{2|N_{I(F)}|} \le s(F) \le |N_{I(F)}|+1$.

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

$$ab \rightarrow c \qquad d \rightarrow eg$$

$$c \rightarrow a \qquad be \rightarrow c$$

$$bc \rightarrow d \qquad cg \rightarrow bd$$

$$acd \rightarrow b \qquad ce \rightarrow ag$$

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 $M = \{B \in P(R) :$ Tồn tại $E_{ij} \in E_r : E_{ij} = B\}$

Bước 3:

$A_r^+ = \cap B$ nếu tồn tại $B \in M : A \subseteq B$,

$A \subseteq B$

R ngược lại.

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

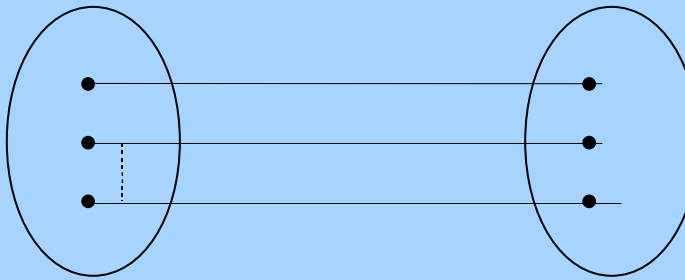
Định lí 2

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 $B \in P(R)$.

$$H(B) = \begin{cases} \cap A \text{ if } \exists A \in L(M) : B \subseteq A, \\ R \quad \text{ngược lại,} \end{cases}$$

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) (**).

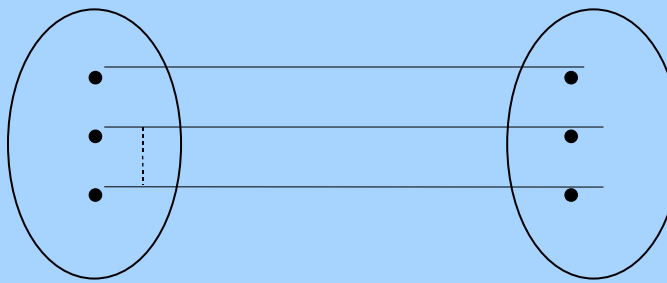
Nếu (A, {a}) $\in$ M(H) thì phù hợp với định nghĩa của M(H) B $\in$ Z(H), và A $\subset$ B kéo theo a $\in$ B. Vì a $\notin$ A, ta thấy A không là giao của các B như thế. Theo cấu trúc của H ta có A $\in$ L(M). Như vậy, nếu (A, {a}) $\in$ M(H) thì A $\in$ L(M) (***).

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

$$O\left(|R|^2 \sum_{q=1} t_q.u_q\right).$$

ở đây $l_q - t_q$, nếu $l_q > t_q$,

$u_q =$

1 nếu $l_q = t_q$

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 $L(n) \to \{a\}$, và $L(n) \subseteq ... \subseteq L(0)$ (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 $B \subset L(n)$ và $B \neq 0$. Như vậy sẽ có a_j sao cho $a_j \notin B$, $a_j \in L(n)$. Theo các xây dựng thuật toán này ta có $L(j-1) - a_j \to \{a\}$. Rõ ràng bởi (1) ta thu được $L(n) - a_j \subseteq L(j-1) - a_j$ (2). Dễ thấy $B \subseteq L(n) - a_j$.

Từ (1), (2) ta có $B \to \{a\}$. □

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, $MAX(F^+, a) = K^{-1}_a$.

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

$u_{iq} = l_{iq}$ nếu $l_{iq} > t_{iq}$ hoặc $u_{iq} = 1$ 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 $T = T(i) - b_{i+1}$

$$T(i+1) = \begin{cases} T & \text{nếu } \forall B \in K : T \notin B \\ T(i) & \text{ngược lại} \end{cases}$$

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 $B \in T^{-1}$, $B \notin B_i$ ($\forall i : 1 \leq i \leq m$).

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} → {b},{a} → {c},{b,d} → {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

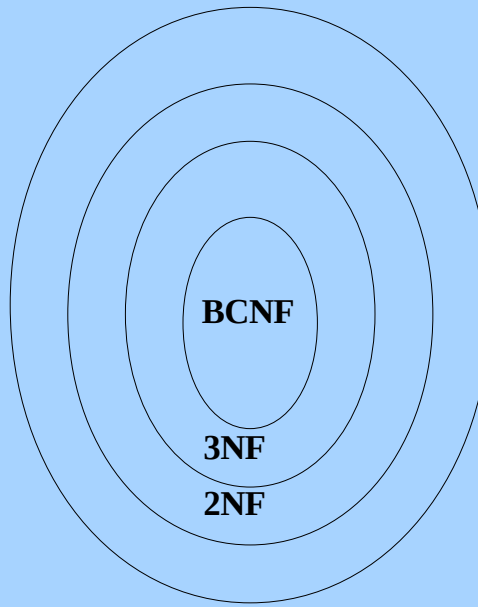
F = {{a} →{c}, {b} →{d}, {c} →{a, b, d}}.

F' = {{a, b} →{c}, {d} →{b}, {c} →{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: $C^+ \cap F_n = \varnothing$;

(3) Với mỗi B ∈ l_s và a ∈ F_n : $(B - a)^+ = B - a$.

Lời giải: Giả sử s là 2NF. Nếu $F_n = \varnothing$ thì (2) là rõ ràng. Giả thiết rằng $F_n \neq \varnothing$. Do định nghĩa của F_n và của M_s, (3) là hiển nhiên.

Giả sử rằng chúng ta có (2) và $F_n \neq \varnothing$. 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, $C^+ \cap F_n \neq \varnothing$. Điều này là vô lý. Do đó ta thu được (3).

Bây giờ, giả sử chúng ta có (3) và $F_n \neq \varnothing$. 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

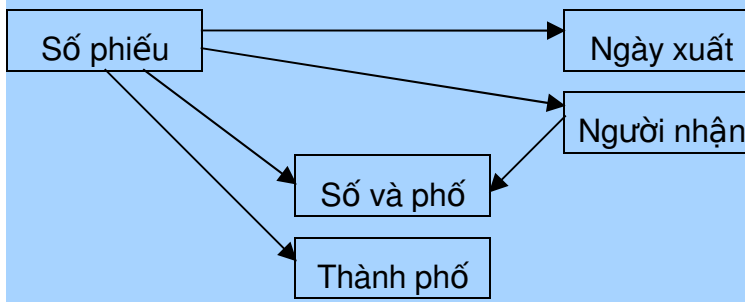
Vật tư

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.

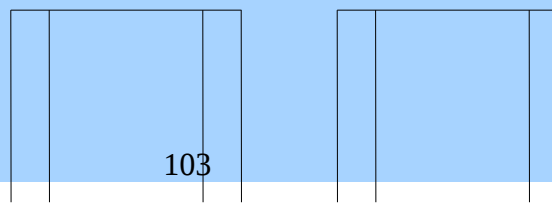
Ví dụ: r ∩ t =

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 = $\prod$ $i_1, i_2...i_{n+m-q}$ ($\delta r.A_1 = t.A_1 \wedge ... \wedge r.A_q = t.A_q$ (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

$\Pi_{\text{Ngày tháng,Tổng}}(\delta_{\text{Ngày tháng' > '210397'}}$ (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.

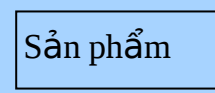
5.2.1. Thực thể

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

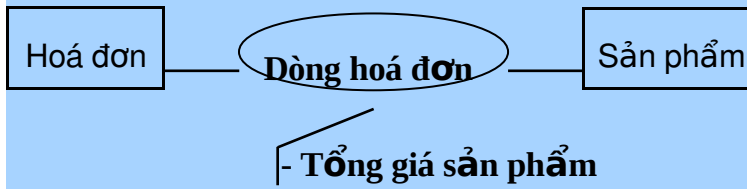
Khái niệm quan hệ ở mục này (khác với khái niệm quan hệ của Codd) được dùng để nhóm họp 2 hay nhiều thực thể với nhau nhằm biểu hiện một mối liên quan tồn tại trong thế giới thực giữa các thực thể này. Kích thước của một quan hệ là số thực thể đã cấu thành nên quan hệ, và có thể là một số

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.

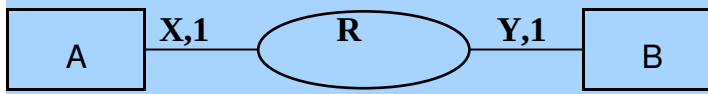
- Y là số tối đa các thể hiện tương ứng với E mà R có thể có trong thực tế.

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.

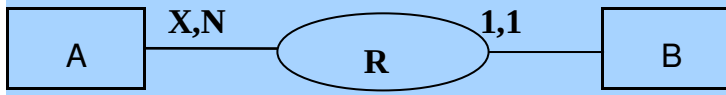


X và Y có thể lấy các giá trị 0 và 1

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.

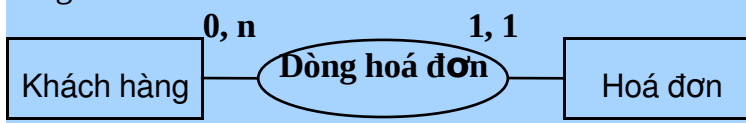


X có thể lấy các giá trị 0 và 1

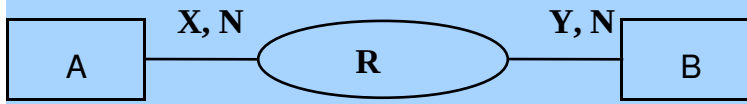
Ví dụ:

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

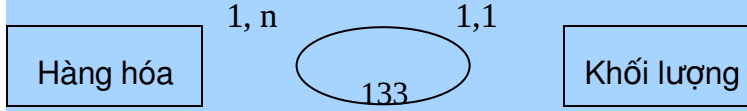
Doanh số

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:



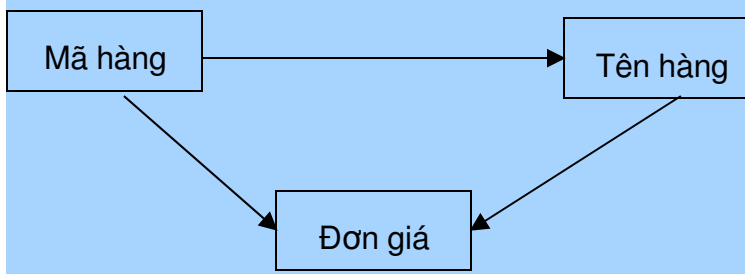
R

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

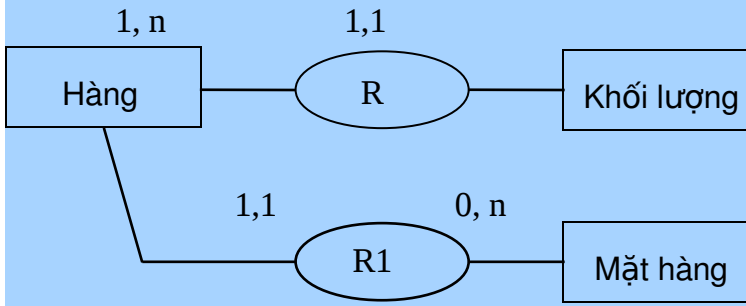
Tên hàng	Đơn giá
Radio	1000
Máy giặt	3000
TV	4000

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:

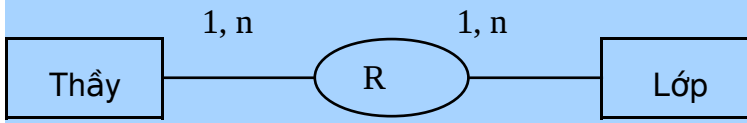
Thực thể 'Lớp':

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.....
.....

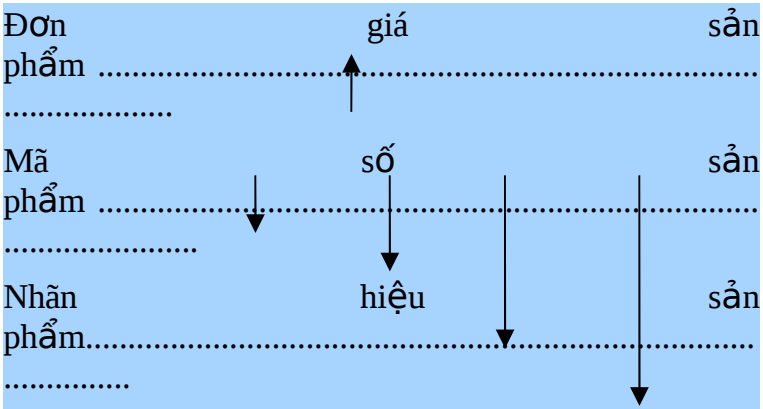
Tên kho .....
.....

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'

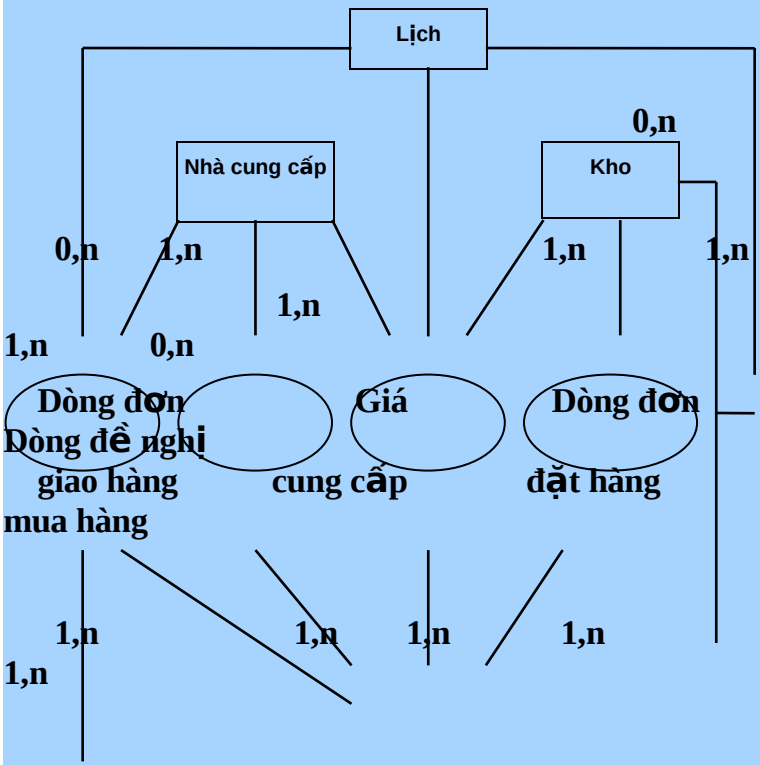
'Dòng đơn đặt hàng'

'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 .

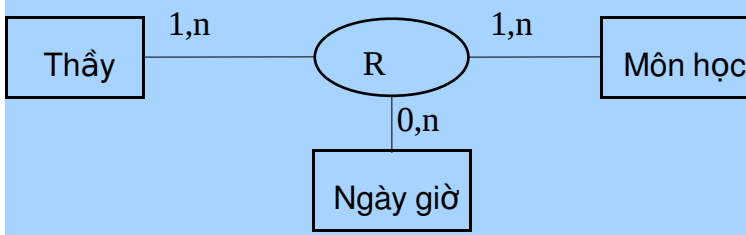
6.4.3. Biến một quan hệ thành thực thể

Một quan hệ có kích thước lớn hơn 3 nên được biến thành thực thể để đơn giản hoá.

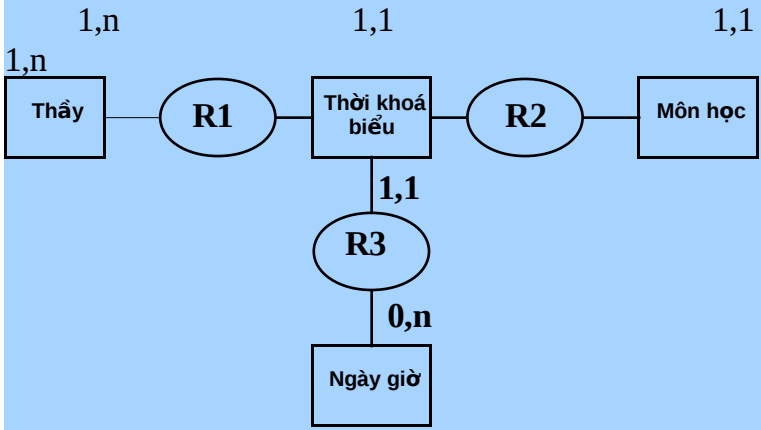
Có thể biến một quan hệ thành thực thể khi hội đủ các điều kiện sau:

- 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.



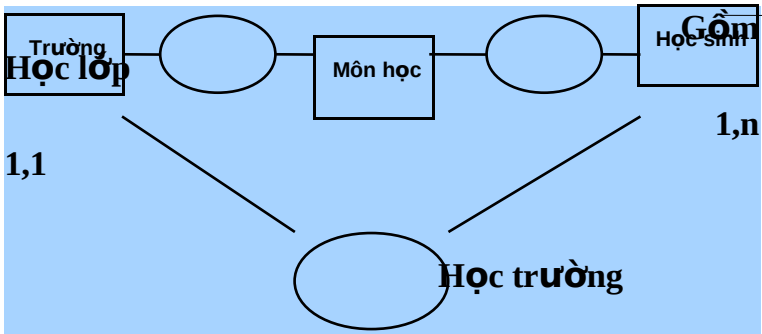
Quan hệ R được biến thành thực thể 'Thời khoá biểu':



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

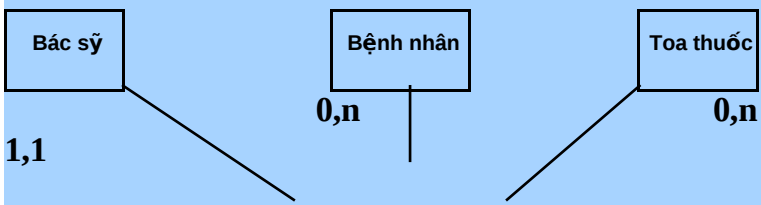
6.4.5 Phân tách một quan hệ phức tạp

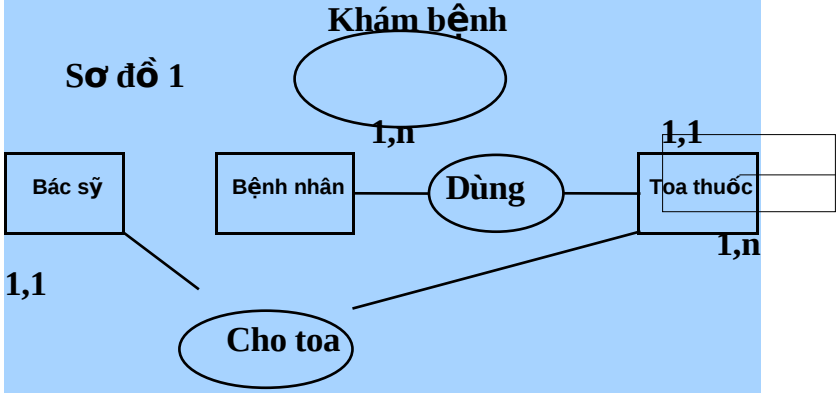
Xét một quan hệ có kích thước lớn hơn hoặc bằng 3. Quan hệ có thể được phân tách thành nhiều quan hệ khác với kích thước nhỏ hơn mà không mất thông tin nếu tồn tại ít nhất một phụ thuộc hàm giữa các thực thể cấu thành quan hệ.

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

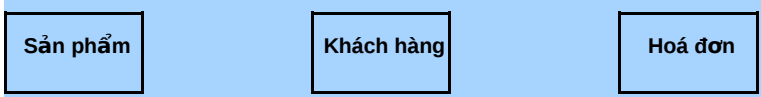
Sơ đồ 2

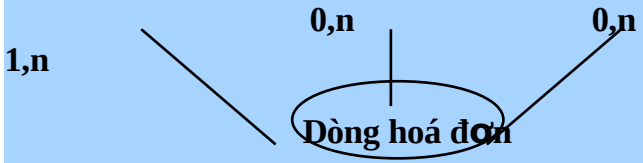
Chúng ta chọn sơ đồ 2

6.4.5.2. Trường hợp phụ thuộc hàm hiện

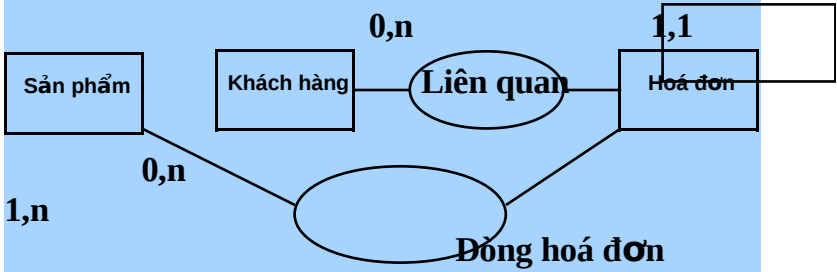
Cho một quan hệ R giữa 3 thực thể A, B, và C. Nếu tồn tại một hàm phụ thuộc A →B thì R có thể được phân thành quan hệ giữa A với B và giữa B với C.

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

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

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

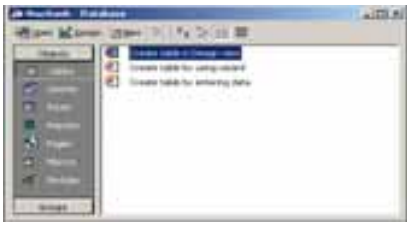
3.4. Các thành phần cơ bản của một tập tin CSDL ACCESS

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:

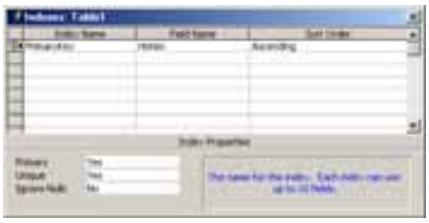
Mở bảng ở chế độ Design View

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

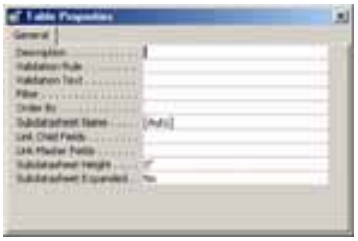
Chọn trường hiện thời là trường sẽ nằm sau trường được chen vào
Thực hiện lệnh *Insert/ Row*

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 nhấp chọn nút  thì bảng sẽ chuyển sang chế độ Datasheet, để quay trở về chế độ Design, ta nhấp 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

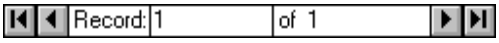
 Bản ghi mới

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

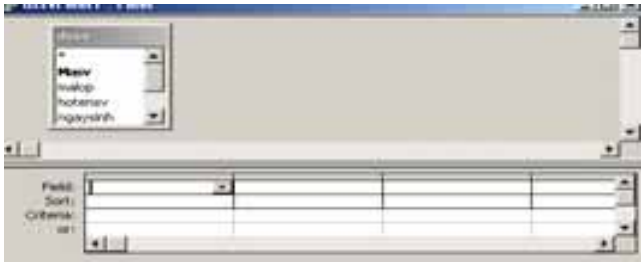
Thực hiện lệnh Records/ Sort/ Sort Ascending (Nếu sắp xếp tăng dần)

/ 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.

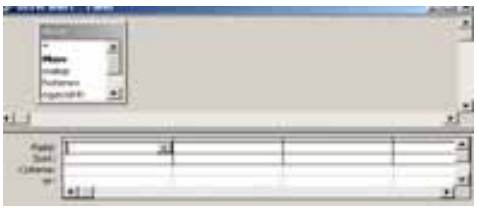
Trong hàng Criteria: Chọn điều kiện sắp xếp (nếu có).

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..

Ch□□ng 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

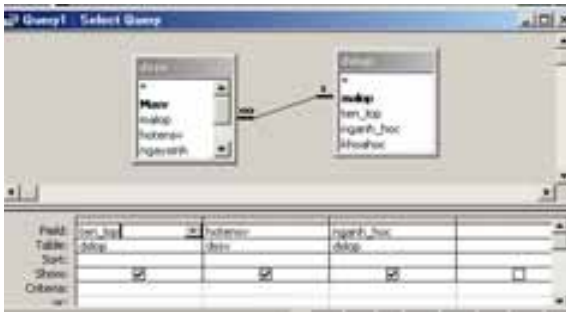
Từ cửa sổ Database, click vào đối tượng Queries.

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

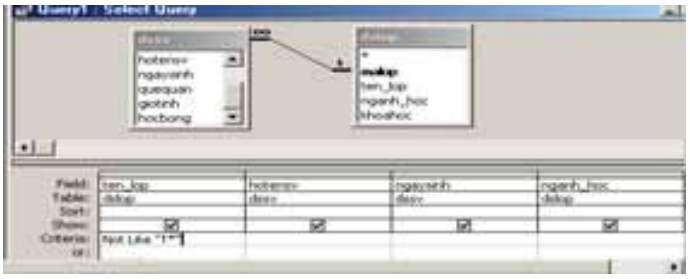
Đưa các trường Tenlop, hotensv, ngaysinh, nganh_hoc vào vùng lưới QBE

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

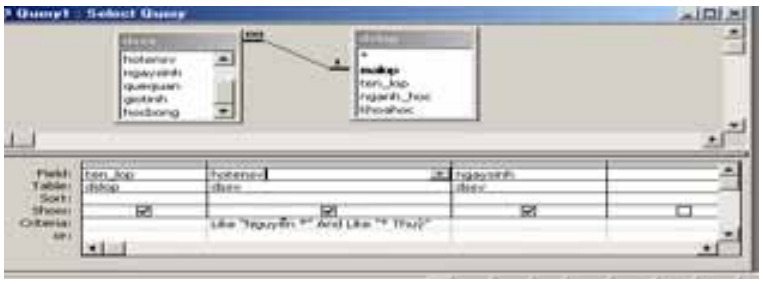
Muốn quy định nhiều tiêu chuẩn trong cùng một trường, chúng ta phải sử dụng toán tử AND (và) cùng toán tử OR (hoặc).

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

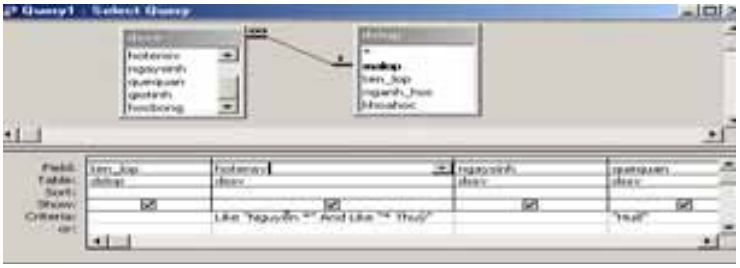
Ví dụ: Tạo một truy vấn để hiển thị danh sách những sinh viên có Họ là "**Nguyễn"** và tên "**Thuỷ**" và có quê quán ở "**Huế"** bao gồm các trường: Tenlop, Hotensv, ngaysinh Quequan.

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].**

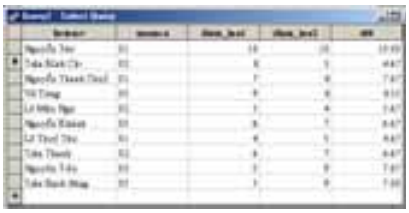
4.2.8. Tạo trường kiểu biểu thức

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)

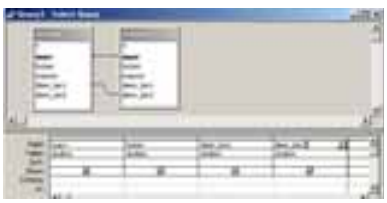
Tạo truy vấn để hiển thị danh sách những sinh viên có điểm thi Lần 1 bằng điểm thi lần 2...

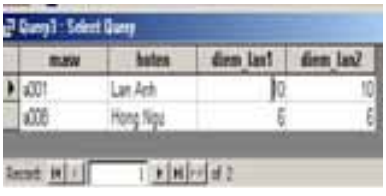
Tạo truy vấn mới và đưa bảng Dsdiem vào tham gia truy vấn 2 lần

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

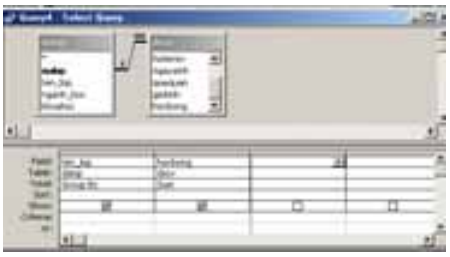
Trong vùng lưới QBE:

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

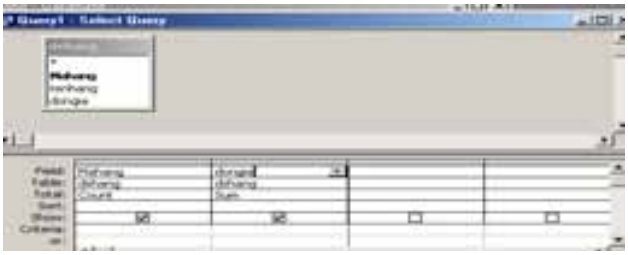
Đưa 2 trường Mahang và dongia vào vùng lưới QBE.

Chọn View/Totals

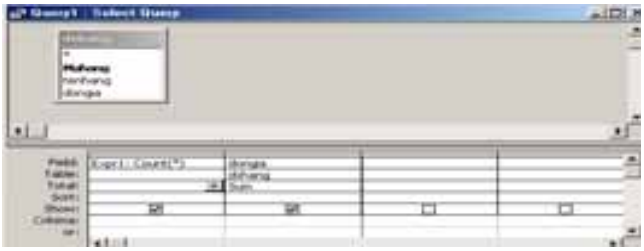
Trong hàng Total của trường Mahang chọn phép toán Count

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?

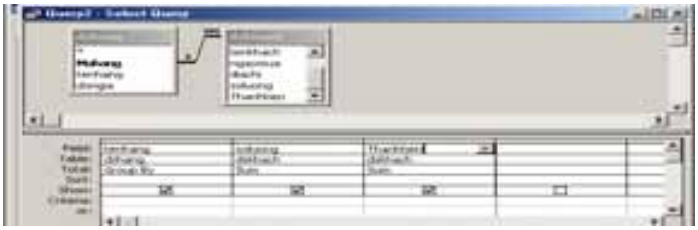
Tạo truy vấn và đưa 2 bảng dshang và dskhach vào tham gia truy vấn

Chọn View/Totals

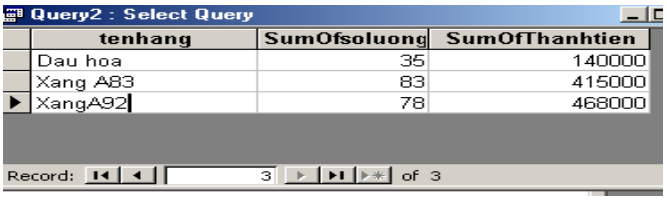
Đưa các trường Tenhang, soluong, thanhtien vào vùng lưới QBE.

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.

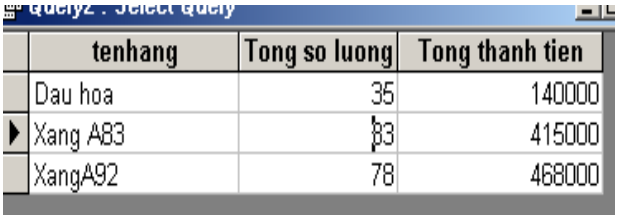


Query2 : Select Query

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.

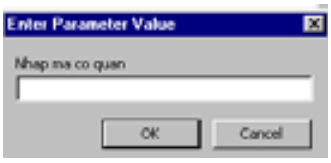
7. TRUY VẤN THAM SỐ (Parameter Query)

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 ([ ])

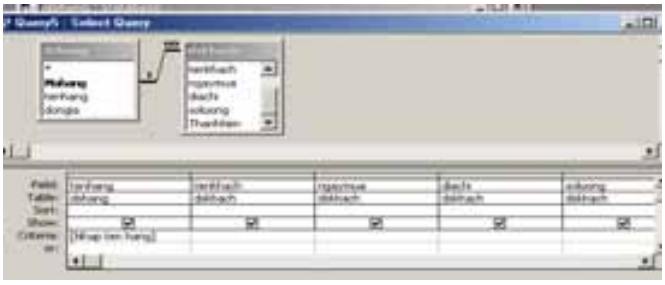
Tên tham số cũng là chuỗi nhắc nhở.Access cho phép có khoảng trắng và độ dài tối đa 255 ký tự.

Quy định kiểu dữ liệu cho tham số: Chọn queries/ parameter query.

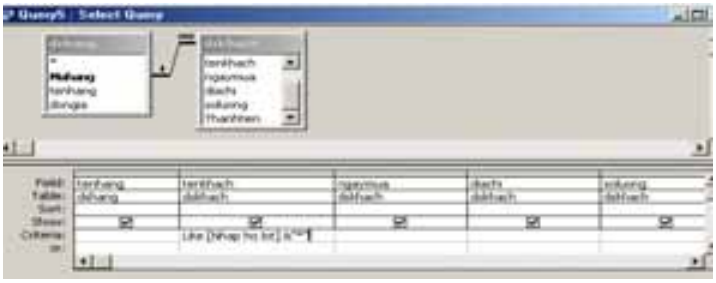
Trong hộp thoại query parameters: Trong mục Parameter chọn tham số, trong mục Data type chọn kiểu dữ liệu tương ứng.

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.

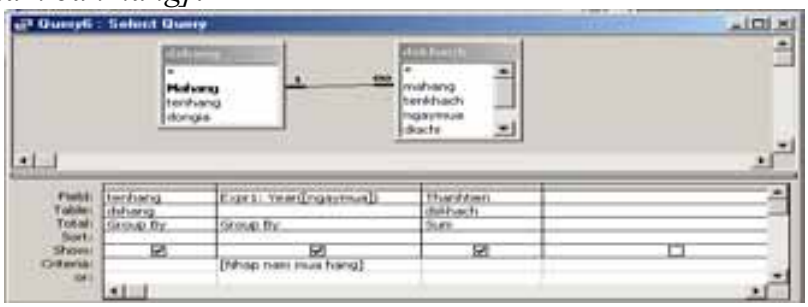
Đưa các trường tenhang, ngayban, thanhtien vào vùng lưới QBE.

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.?

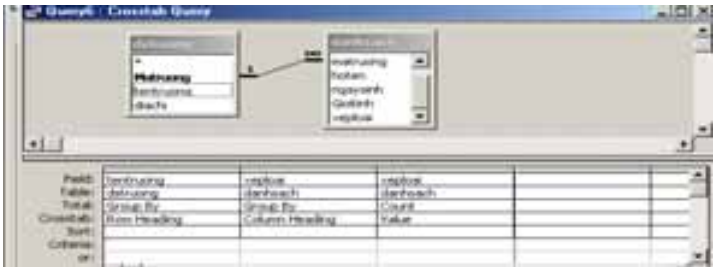
Tạo truy vấn và đưa 2 bảng dữ liệu vào tham gia truy vấn, đưa các trường tentruong và xeploai vào vùng lưới QBE. (Trường Xeploai đưa vào 2 lần)

Chọn Query/ crosstab query

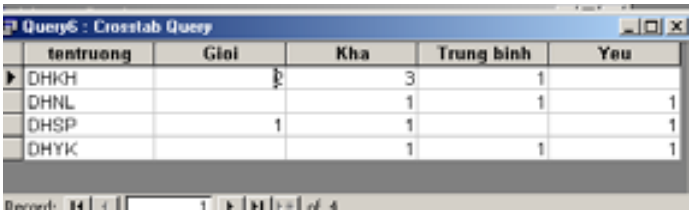
Tại hàng Total của trường tentruong: Chọn phép toán Group by, hàng crosstab: chọn Row heading

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.

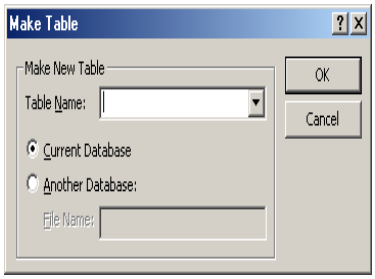
Truy vấn nối (Append query): Nối một số bản ghi từ một hoặc nhiều bảng dữ liệu vào sau một hoặc nhiều bảng dữ liệu khác.

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.

Chương 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.

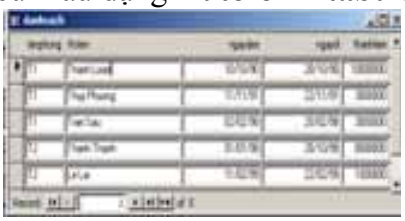
Ví dụ

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.

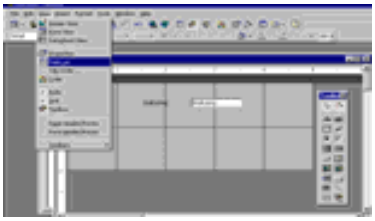
Mở hộp danh sách trường

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

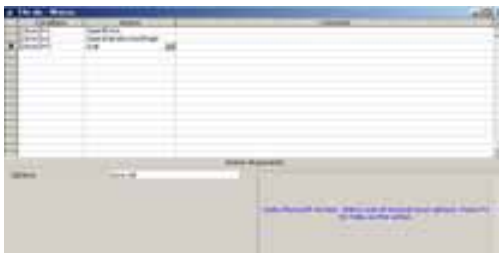
Là macro có chứa các điều kiện thi hành cho mỗi hành động.

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.

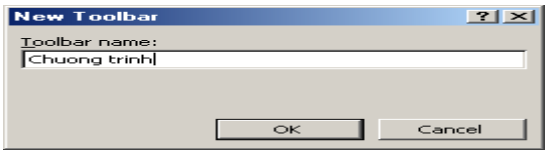
Có 2 cách hiển thị các Menu của người sử dụng (Custom Menu) trong các ứng dụng đó là:

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

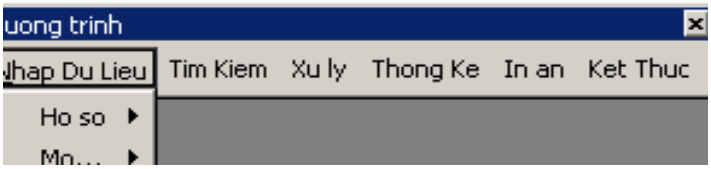


Chọn Properties và chọn Type là Menu bar, close

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**.

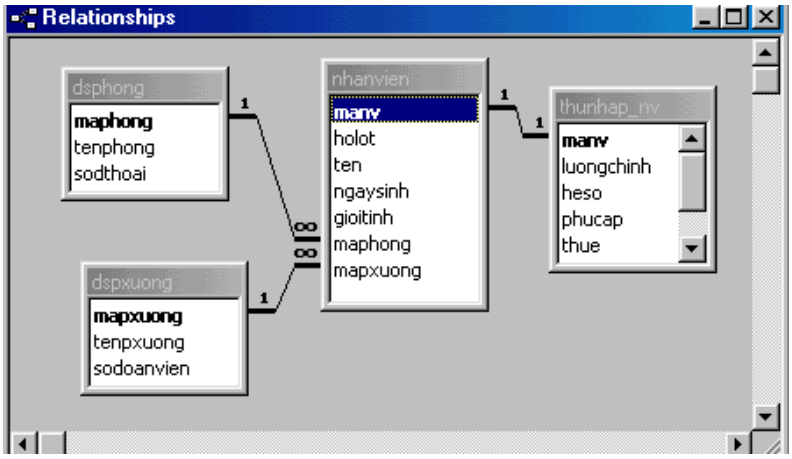
9. Lọc và hiển thị những nhân viên **nữ** có **Mapxuong** là **P1** và sinh trong tháng **10** năm **1975** hoặc nhân viên **nam** có năm sinh **1972** đến **1976**.

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

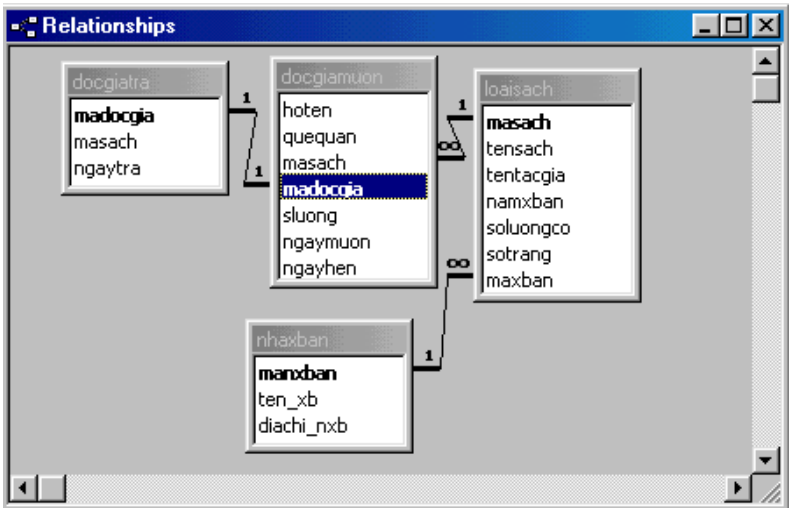
Bảng 4: : **Nhaxban**

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	

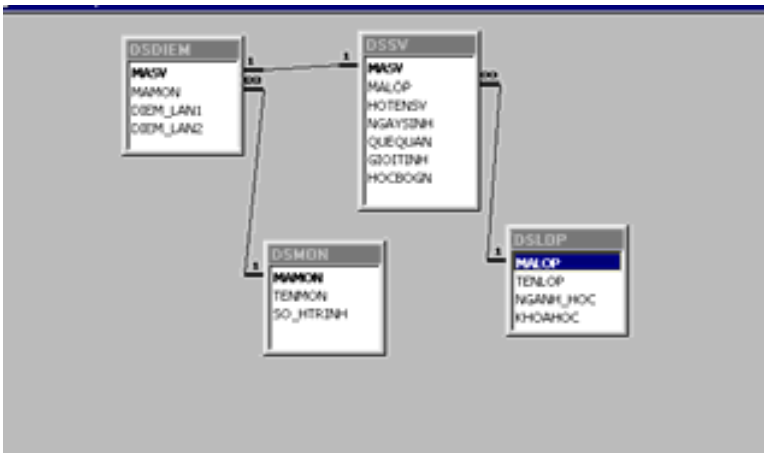
Bảng 3: DSMON

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:

Mã lớp	Mã sv	Họ và tên	Ngày sinh	Quê quán	Giới tính	Học bổng
K23A	s001	Nguyễn Tèo	10-10-78	Huế	☑	1000000
K23B	s002	Trần Bình Chi	05-05-79	Đà Nẵng	☐	2000000
K24A	s005	Lê Mậu Ngọ	08-08-79	Huế	☑	1200000
	s003	Nguyễn Thanh Thu	12-10-79	Huế	☑	2300000
K24B	s004	Võ Tòng	02-02-79	Quảng Bình	☐	2200000
K25A	s010	Trần Bạch đằng	20-10-79	Huế	☑	2000000
	s009	Nguyễn Tiêu	10-10-78	Đà Nẵng	☐	1000000
	s006	Nguyễn Khánh	10-05-75	Đà Nẵng	☑	3000000
K25B	s008	Trần Thanh	25-07-77	Huế	☑	3000000
	s007	Lê Thuỷ Thu	03-03-79	Quảng Nam	☐	2000000

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)

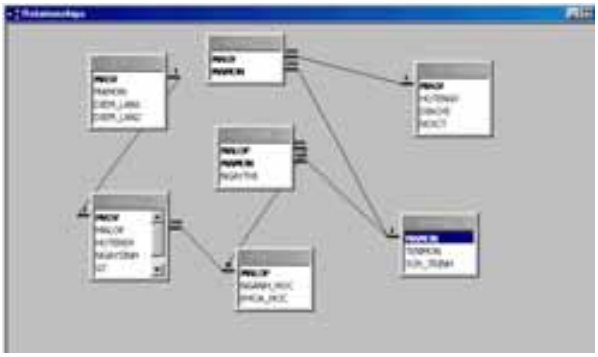
DSGIAOVIEN(**MAGV**, HOTENGVIEN, DIACHI, NOI_CT)

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.

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,…

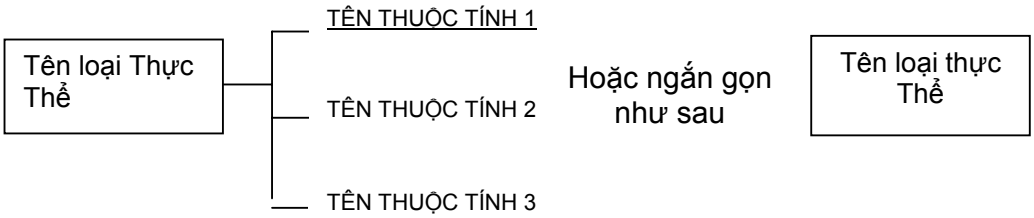
(Trong giáo trình này, tên của loại thực thể được in hoa ký tự đầu tiên, các ký tự còn lại viết thường).

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

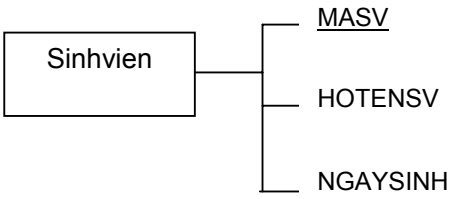
khác. Mỗi giảng viên có thể dạy nhiều môn ở nhiều khoa, nhưng chỉ thuộc về sự quản lý hành chính của một khoa.

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:



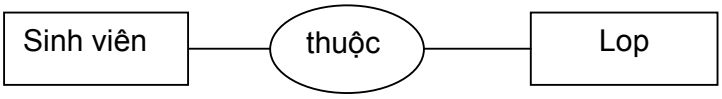
1.3.5.Mối Kết Hợp (relationship)

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:

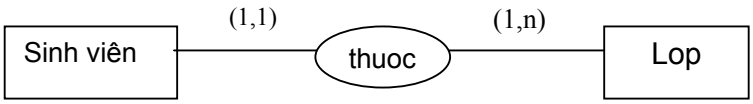


Bản số của mối kết hợp:

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.

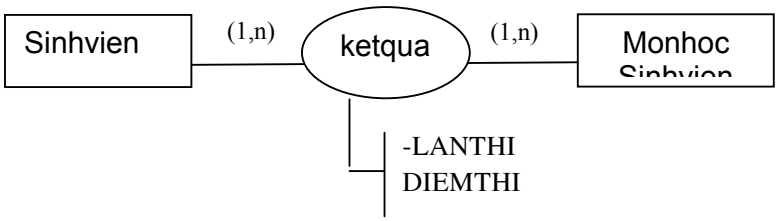
Ví dụ:



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:

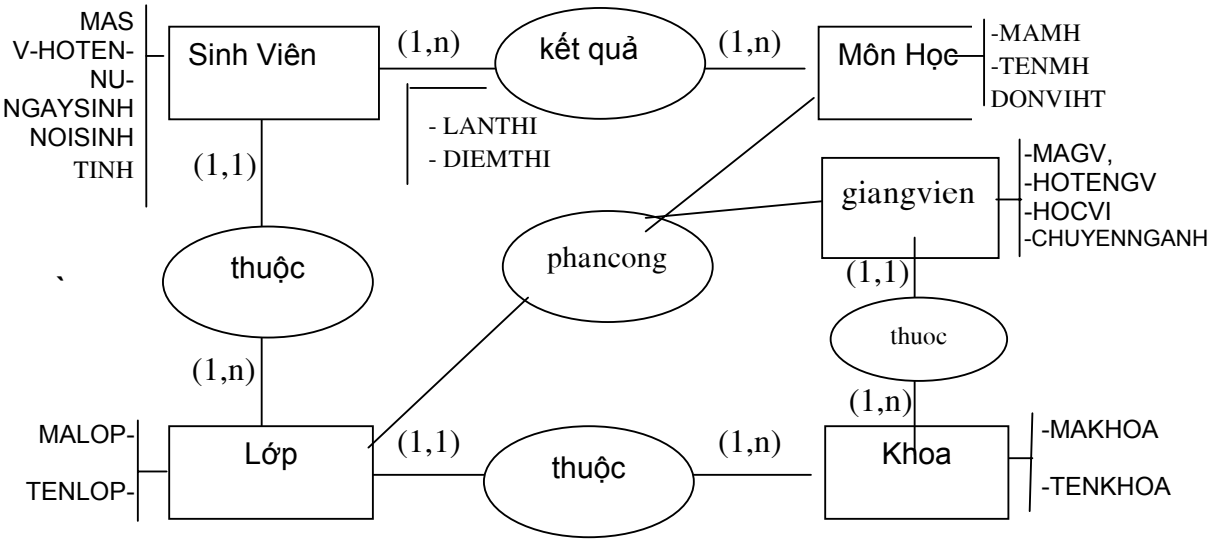
b1.Xác định danh sách các loại thực thể

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.

$$r \overset{Ai\ \theta\ Bj}{|><|} s = \{t_{12} / \exists\ t_1 \in r_1 , \exists\ t_2 \in r_2 \text{ sao cho } t_{12}.Q_1^{+} = t_1 ; t_{12}.Q_2^{+} = t_2 ; t_{12}A_i\ \theta\ t_{12}.B_j\}$$

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)

Cho $X \rightarrow Y,\ X \rightarrow Z \Rightarrow X \rightarrow YZ$

5.*Luật bắc cầu giả* (the pseudotransitivity rule)

Cho $X \rightarrow Y, WY \rightarrow Z \Rightarrow XW \rightarrow Z$

6.*Luật phân rã* (the decomposition rule):

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,…,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 $\rightarrow$ A; DA$\rightarrow$ CE; D $\rightarrow$ H; GH$\rightarrow$ C; AC$\rightarrow$ D; DG $\rightarrow$ C}

G={B$\rightarrow$ A; DA$\rightarrow$ CE; D $\rightarrow$ H; GH$\rightarrow$ C; AC$\rightarrow$ D ;BC $\rightarrow$ AC;

BC $\rightarrow$ D; DA $\rightarrow$ AH; AC $\rightarrow$ 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 $X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X^+$)

Ví dụ 5.5:

Chẳng hạn hai tập phụ thuộc hàm sau là tương đương:

Q(A,B,C)

F={ A$\rightarrow$B; A$\rightarrow$C; B$\rightarrow$A; C$\rightarrow$A; B$\rightarrow$C}

G={ A$\rightarrow$B; C$\rightarrow$A; B$\rightarrow$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: $X \rightarrow A \in F$ và $Z \subset X$ mà:

$$F^+ = (F - (X \rightarrow A) \cup (Z \rightarrow A))^+$$

Điều kiện c) Không $\exists$ $X \rightarrow A \in F$ mà:

$$F^+ = (F - (X \rightarrow A))^+$$

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 ?

$A \to B$; $A \to C$; $B \to A$; $C \to D$; $D \to C$; $D \to A$

5.2. a.Cho lược đồ quan hệ Q(ABCD) và tập phụ thuộc hàm

F = { $A \to B$;

$BC \to D$}.

Những phụ thuộc hàm nào sau đây thuộc F^+ ?

$C \to D$; $A \to D$; $AD \to C$; $AC \to D$; $BC \to A$; $B \to CD$.

b.Cho lược đồ quan hệ Q(ABCDEGH) và tập phụ thuộc hàm

F ={ $AB \to C$;

$B \to D$;

$CD \to E$;

$CE \to GH$;

$G \to A$}

Những phụ thuộc hàm nào sau đây không thuộc vào F^+ ?

$AB \to E$; $AB \to GH$; $CGH \to E$; $CB \to E$; $GB \to E$.

c)Cho lược đồ quan hệ Q,F như sau:

với Q=(ABCD)

F=[$A \to B$;

$A \to C$].

Trong các phụ thuộc hàm sau, những phụ thuộc hàm được suy ra từ F ?

$A \to D$; $C \to D$; $AB \to B$; $BC \to A$; $A \to BC$

5.3. Cho lược đồ quan hệ Q(ABCD) và tập phụ thuộc hàm

F={ $A \to D$;

$D \to A$;

$AB \to C$}

a.Tính AC^+

b.Chứng minh $BD \to C \in F^+$

5.4. a)Q(ABCDEG)

Cho F={$AB \to C$;

$C \rightarrow A$;

$BC \rightarrow D$;

$ACD \rightarrow B$;

$D \rightarrow EG$;

$BE \rightarrow C$;

$CG \rightarrow BD$;

$CE \rightarrow AG$}

X=[BD], X^+=*?*

Y=[CG], Y^+=*?*

b)Cho lược đồ quan hệ Q và tập phụ thuộc hàm F

F={ $AB \rightarrow E$;

$AG \rightarrow I$;

$BE \rightarrow I$;

$E \rightarrow G$;

$GI \rightarrow H$ }

Chứng minh rằng $AB \rightarrow GH$.

c.Tương tự cho tập phụ thuộc hàm

F = { $AB \rightarrow C$;

$B \rightarrow D$;

$CD \rightarrow E$;

$CE \rightarrow GH$;

$G \rightarrow A$}

Chứng minh rằng $AB \rightarrow E$; $AB \rightarrow G$

d.Q(ABCDEGH)

F = {$B \rightarrow A$; $DA \rightarrow CE$; $D \rightarrow H$; $GH \rightarrow C$; $AC \rightarrow D$ }

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$\rightarrow$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

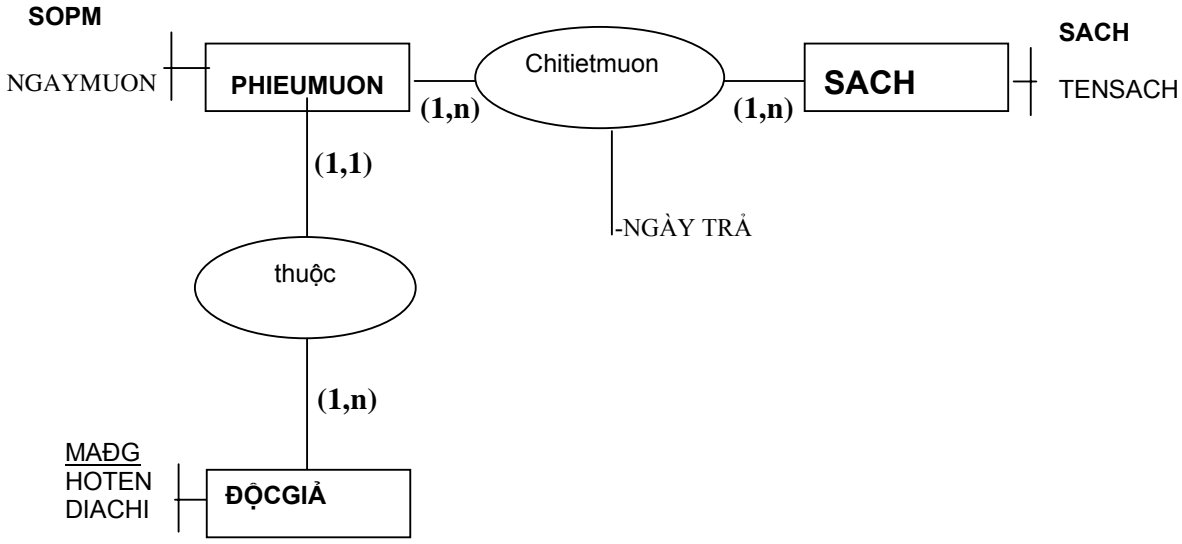
KHOA ĐIỆN TỬ - TIN HỌC

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:

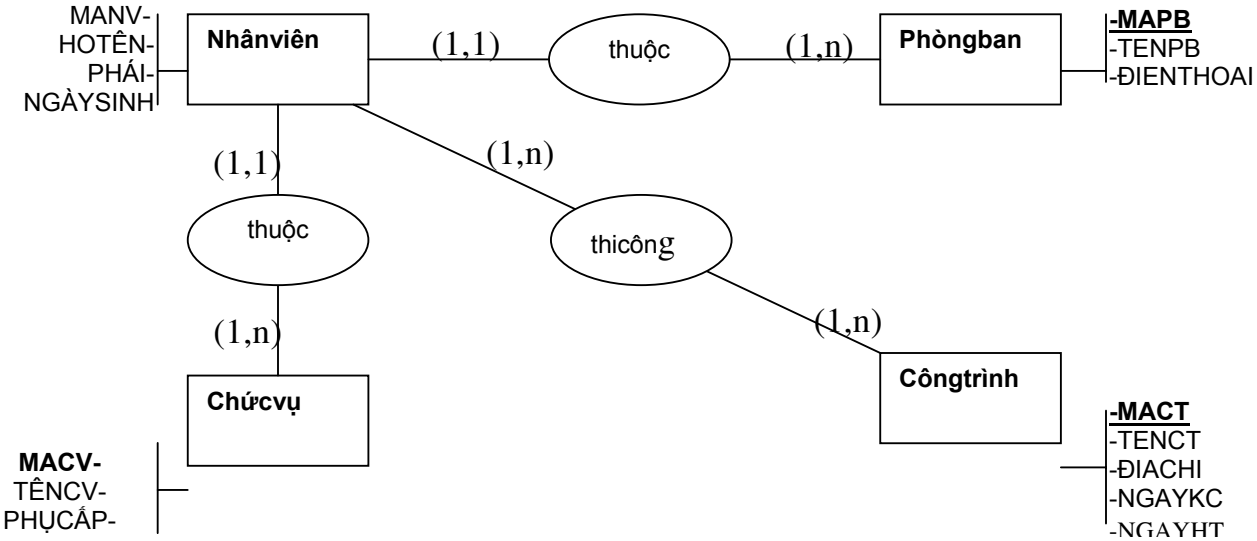
100,101,103 ? Đây là kết quả của phép toán quan hệ nào ?

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 → T; V U → A

b) O → V; A V → U

c) T → O; A V → U

d) A O → T

e) O → V; V U → 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→ C

C → 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

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

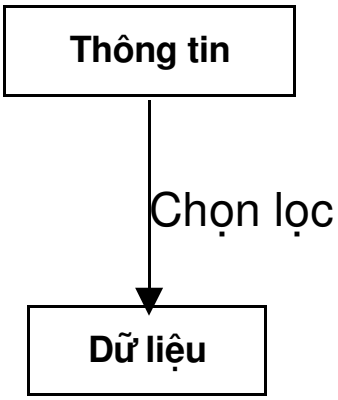
TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

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
  …

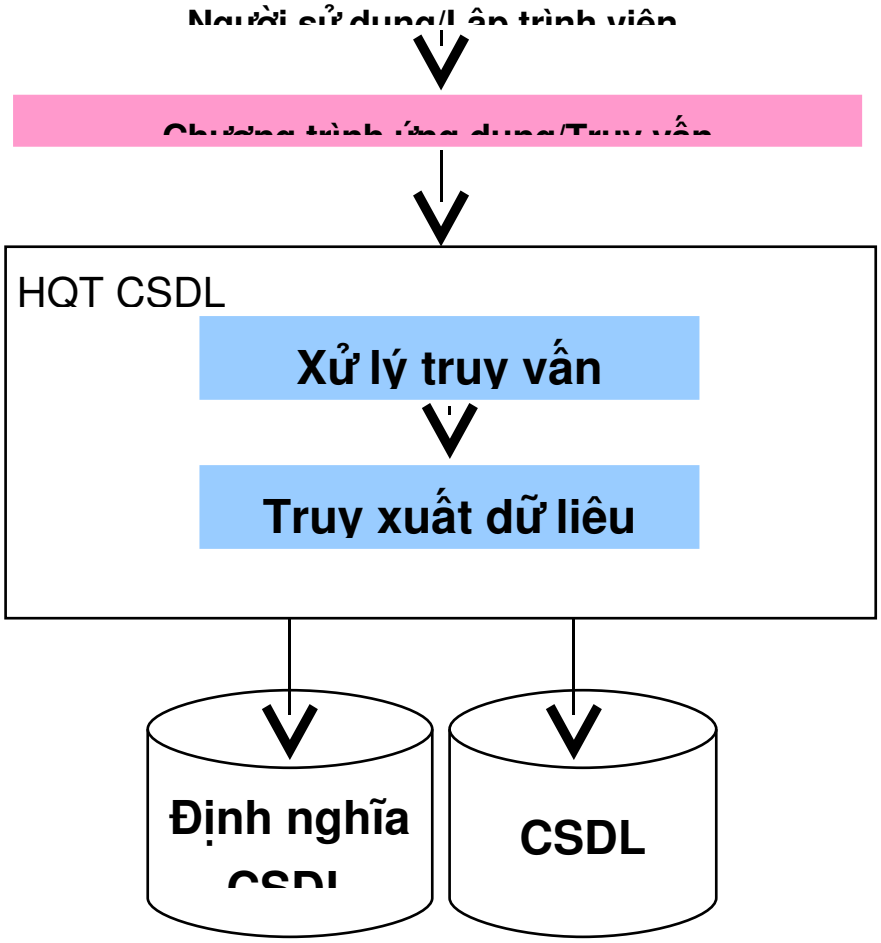


- Dữ liệu (Data)
  - Một mô tả hình thức về thông tin và hoạt động
  + Tên, địa chỉ, số điện thoại của khách hàng
  + Báo cáo doanh thu
  + Đăng ký học phần

- 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

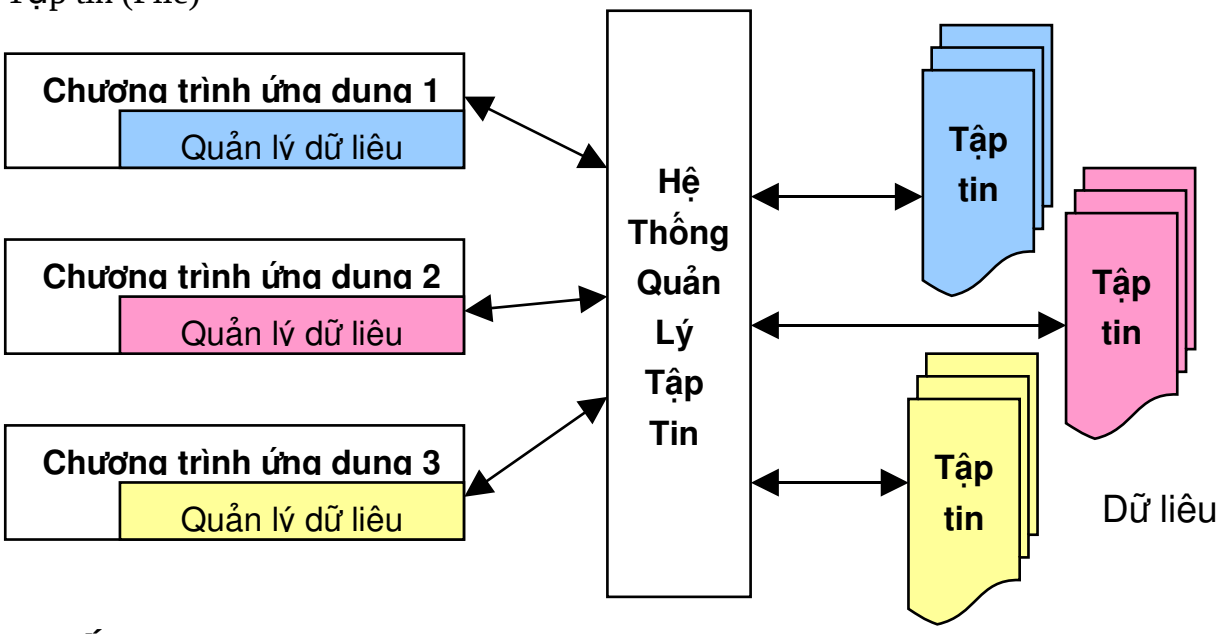
DEAN	TENDA	MADA	DDIEM_DA	PHONG
	San pham X	1	VUNG TAU	5
	San pham Y	2	NHA TRANG	5
	San pham Z	3	TP HCM	5
	Tin hoc hoa	10	HA NOI	4

PHANCONG	MA_NVIEN	SODA	THOIGIAN
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0

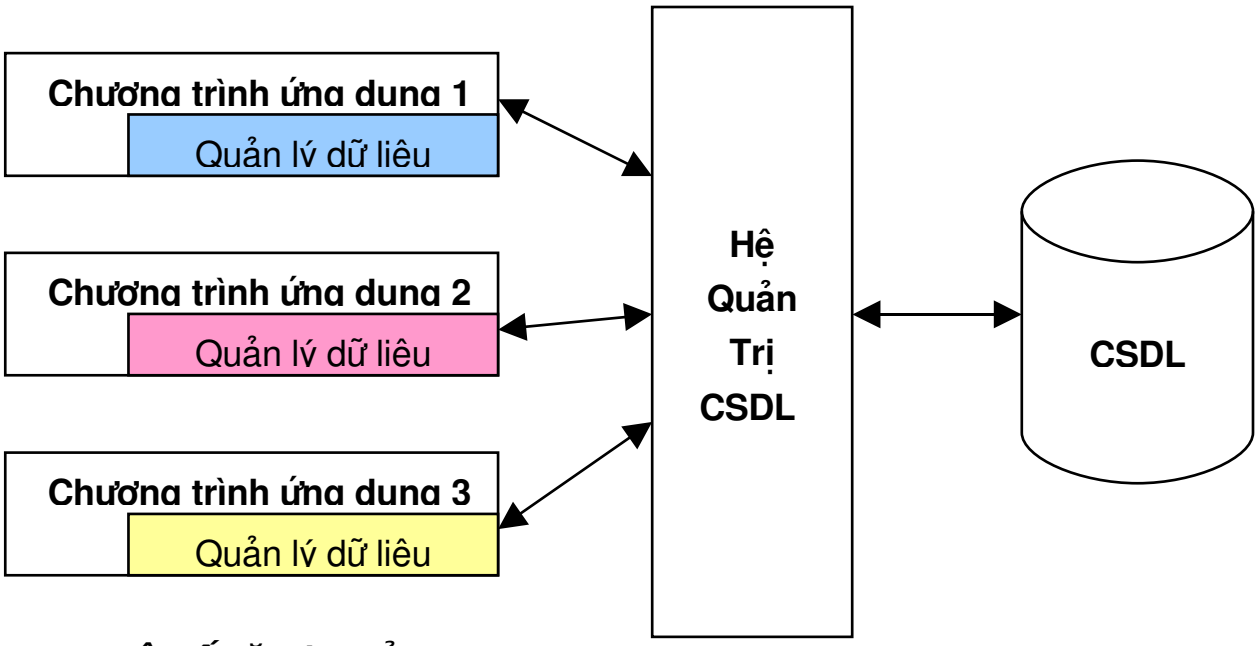
- Quản lý đề án của một công ty
  - Định nghĩa CSDL
    - Cấu trúc bảng, bao gồm các thành phần dữ liệu và kiểu dữ liệu tương ứng
  - Xây dựng CSDL
    - Đưa dữ liệu vào các bảng
  - Xử lý CSDL
    - Thực hiện các truy vấn: "Cho biết những nhân viên thuộc phòng 5"
    - Thực hiện các phép cập nhật: "Chuyển 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

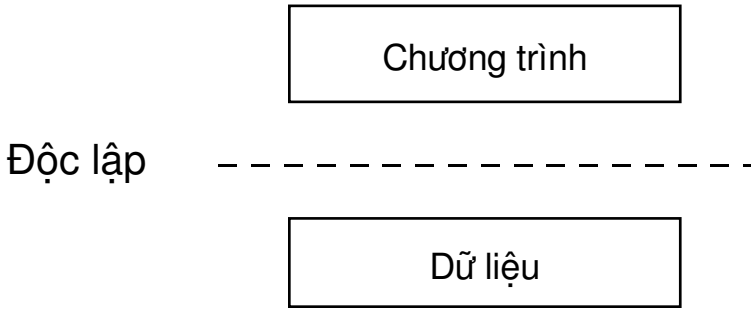
1. Tính tự mô tả

- 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

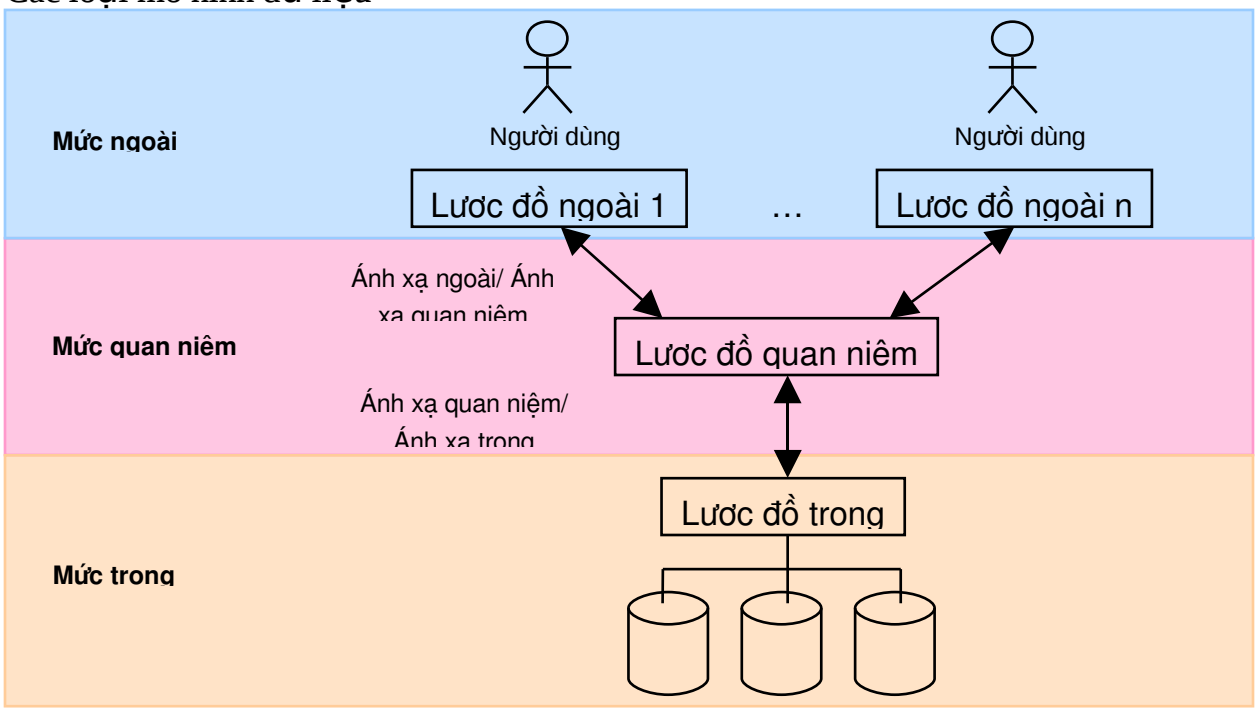
- Chịu trách nhiệm về
  - Lựa chọn cấu trúc phù hợp để lưu trữ dữ liệu
  - Quyết định những dữ liệu nào cần được lưu trữ
- Liên hệ với người dùng để nắm bắt được những yêu cầu và đưa ra một thiết kế CSDL thỏa yêu cầu này
- Có thể là 1 nhóm các DBA quản lý các CSDL sau khi việc thiết kế hoàn tất

3. Người dùng cuối

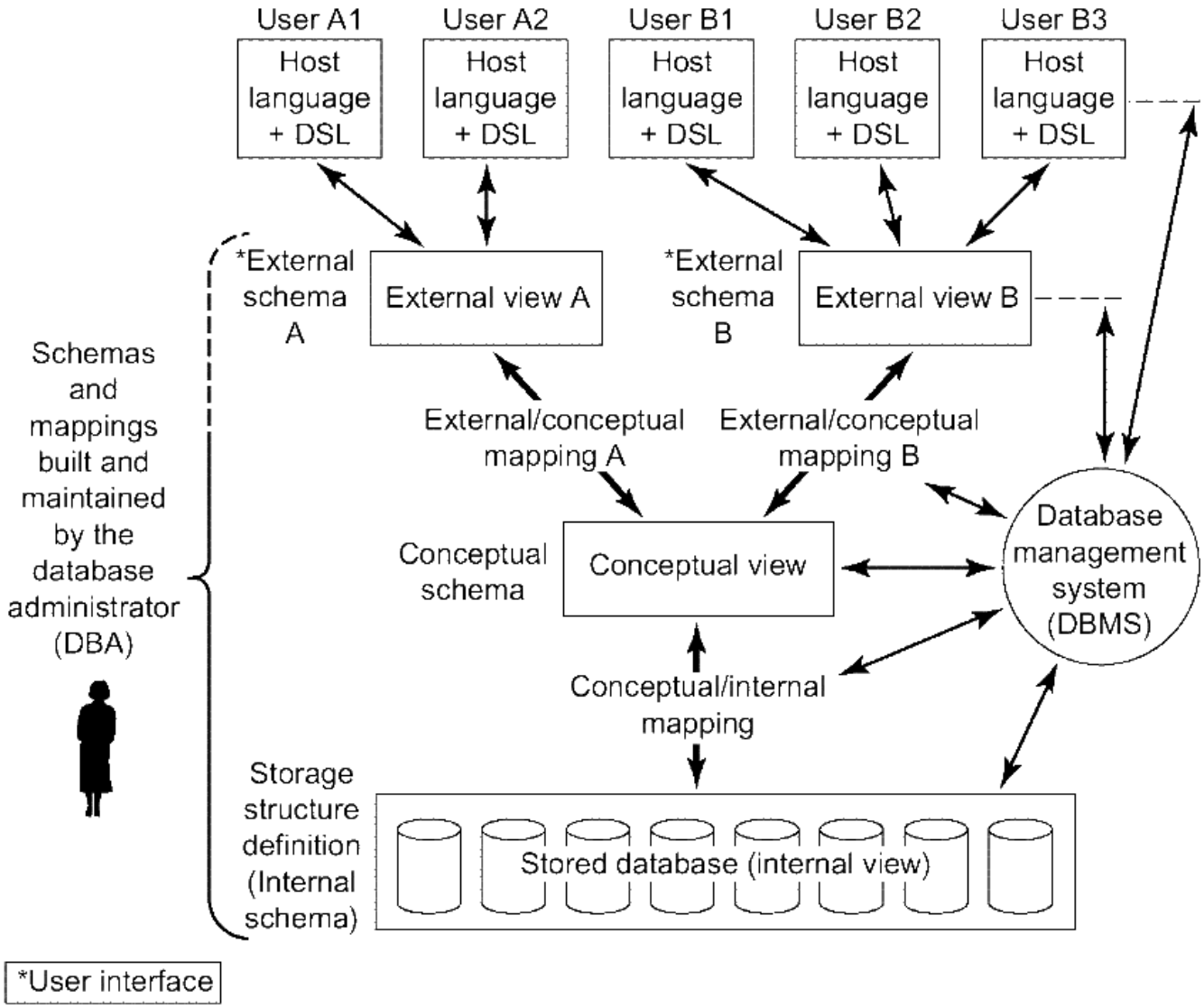
- Người ít sử dụng
  - Ít khi truy cập CSDL, nhưng cần những thông tin khác nhau trong mỗi lần truy cập và dùng những câu truy vấn phức tạp
  - Người quản lý
- Người sử dụng thường xuyên
  - Thường xuyên truy vấn và cập nhật CSDL nhờ vào một số các chức năng đã được xây dựng sẵn
  - Nhân viên
- Người 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

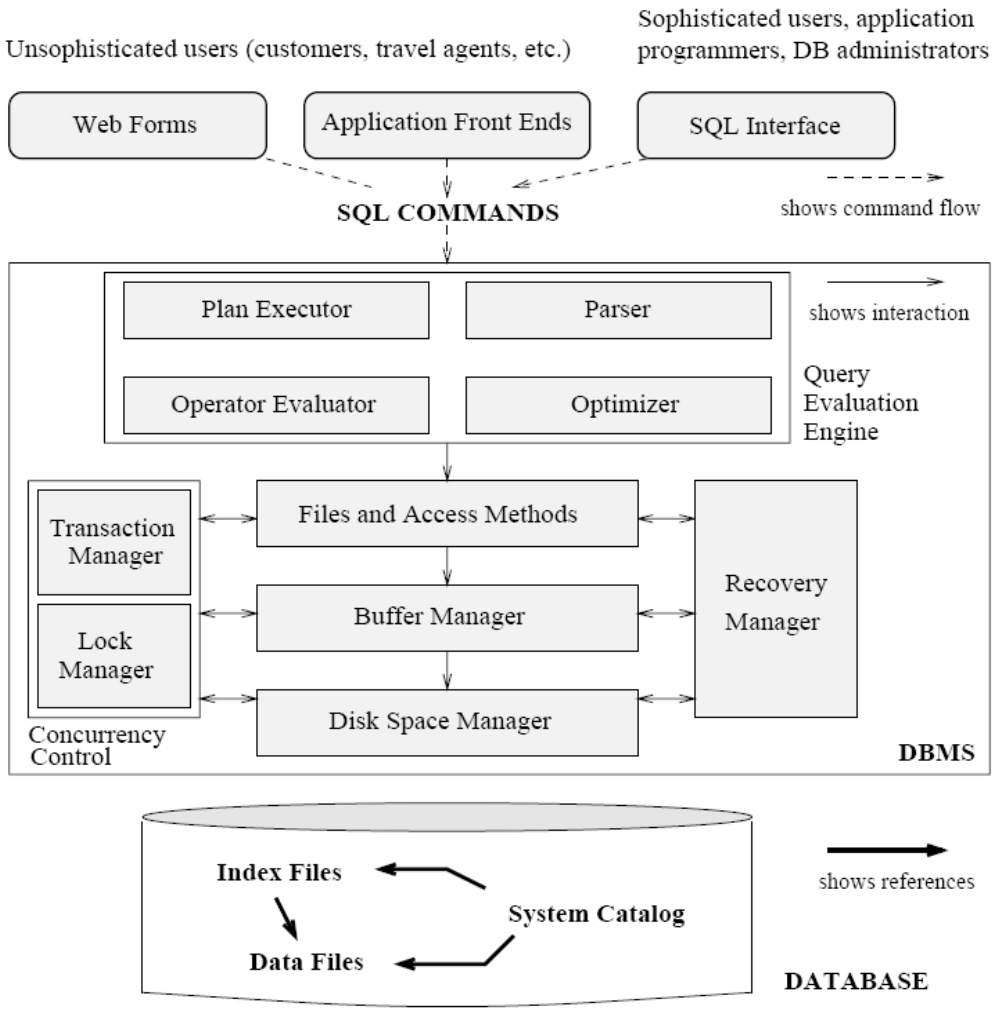
Các loại mô hình dữ liệu



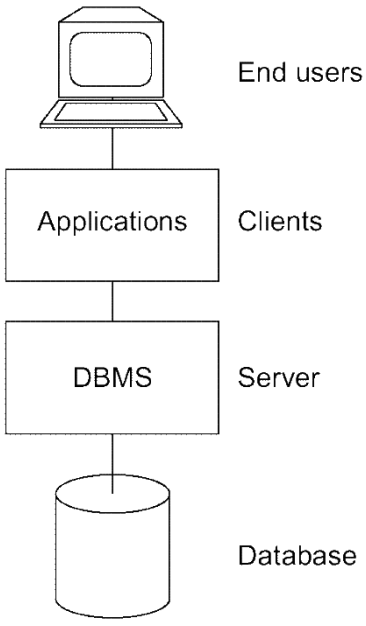
- Mức trong (lược đồ trong)
  - Mô tả cấu trúc lưu trữ vật lý CSDL
- Mức quan niệm (lược đồ quan niệm)
  - Mô tả cấu trúc của toàn thể CSDL cho 1 cộng đồng người sử dụng, gồm thực thể, kiểu dữ liệu, mối liên hệ và ràng buộc
  - Che bớt các chi tiết của cấu trúc lưu trữ vật lý
- Mức ngoài (lược đồ ngoài)
  - 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



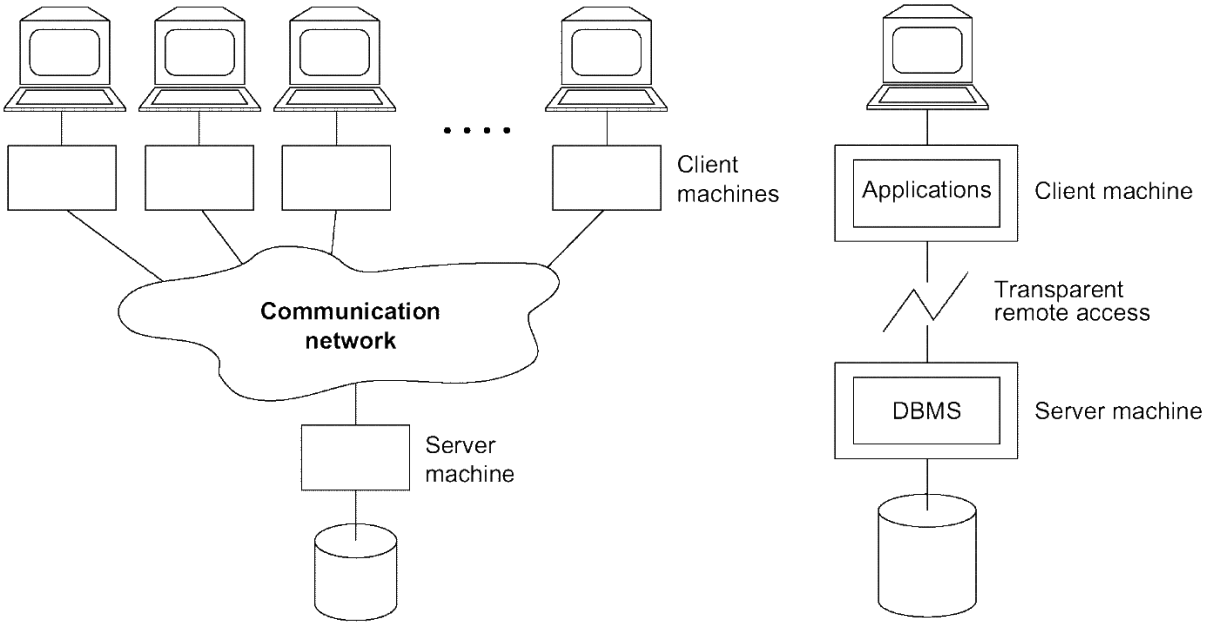
- Độc lập dữ liệu
  - Độc lập logic
    - Khả năng thay đổi lược đồ quan niệm mà không thay đổi lược đồ ngoài hoặc các chương trình ứng dụng
  - Độc lập vật lý
    - Khả năng thay đổi lược đồ trong mà không làm thay đổi lược đồ quan niệm cũng như lược đồ ngoài



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

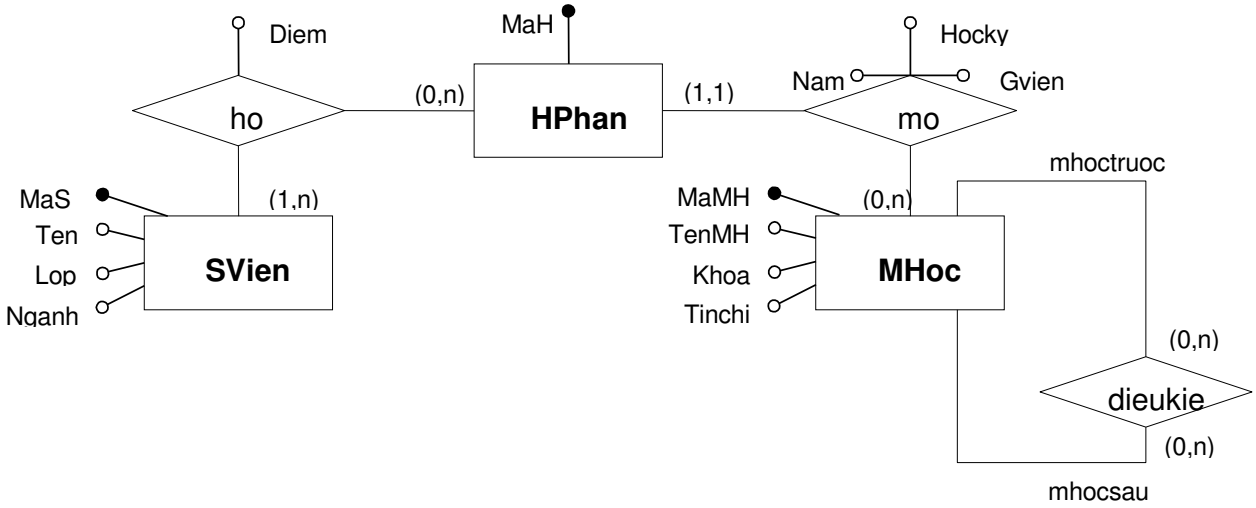
- Giảm thời gian phát triển ứng dụng
- Tính khả dụng
  - Khi có một sự thay đổi lên CSDL, tất cả người dùng đều thấy được

VII. Các KN

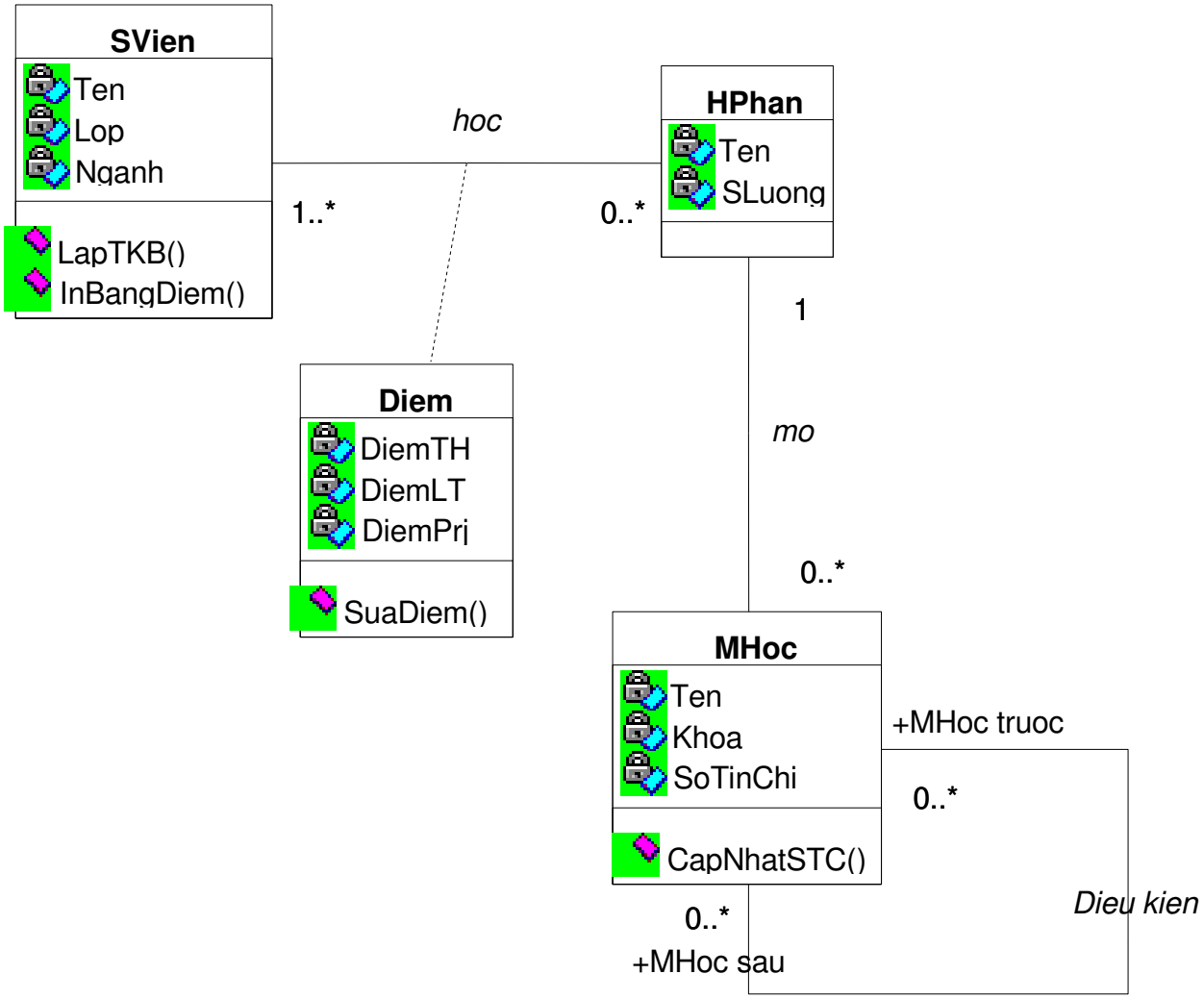
1. Mô hình dữ liệu

- 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

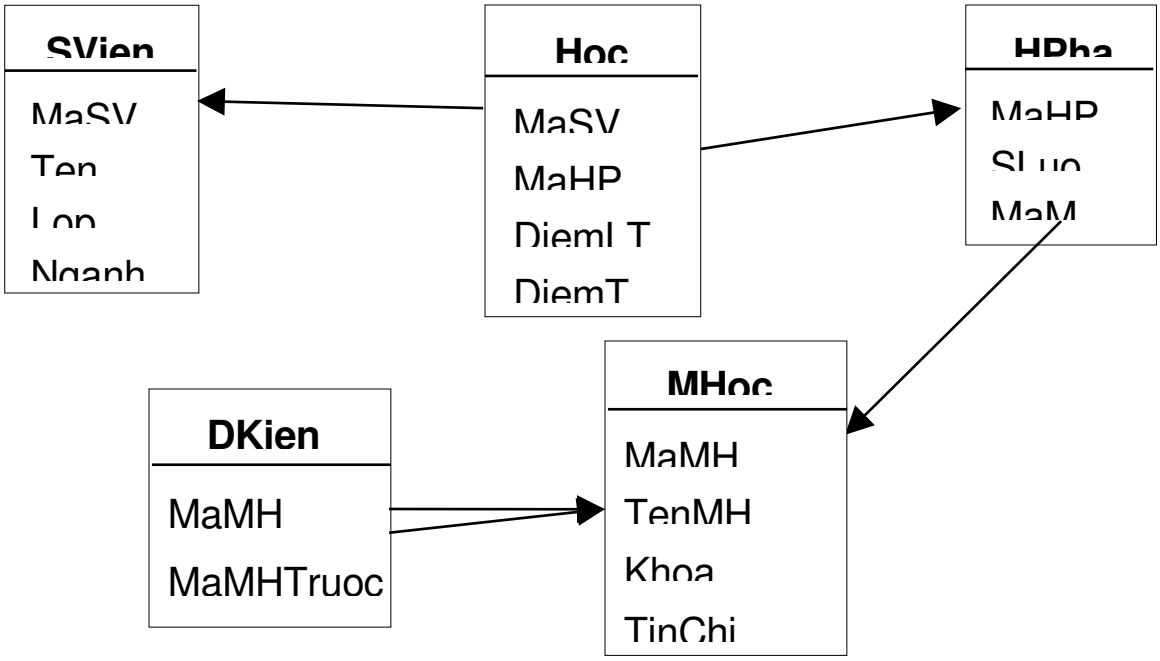
* Ví dụ mô hình ER



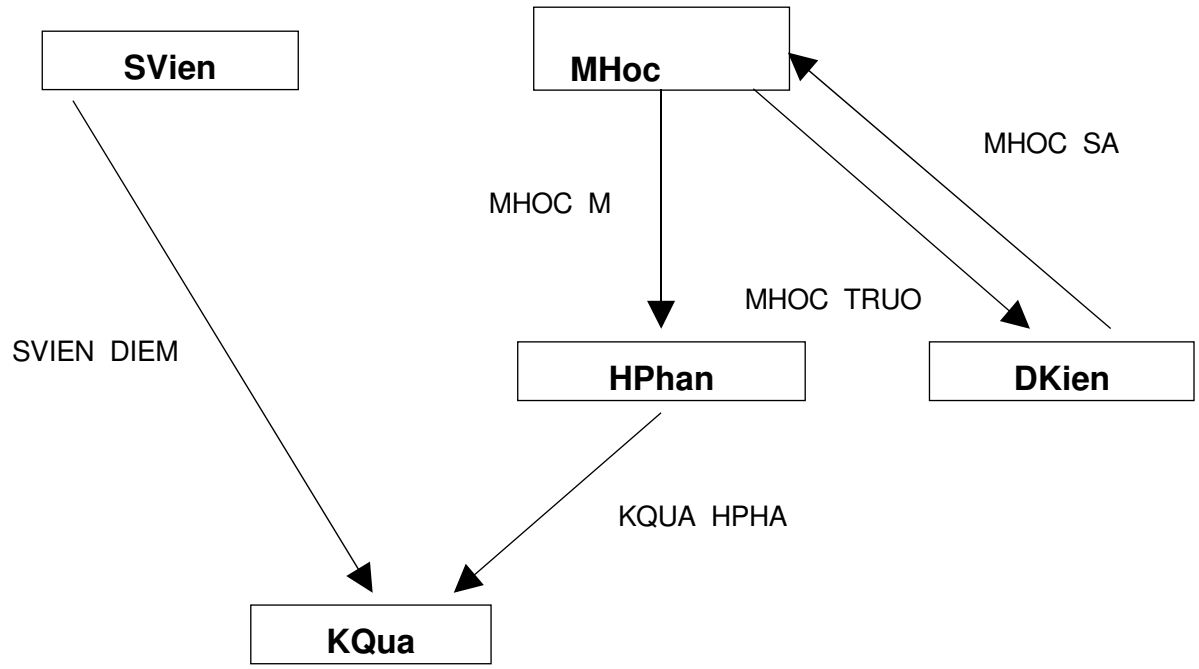
* Ví dụ mô hình đối tượng



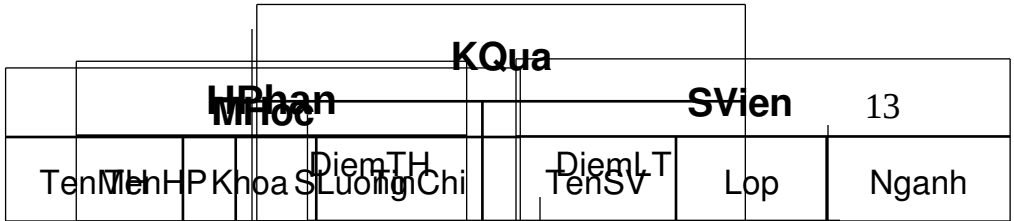
* Ví dụ mô hình quan hệ



* Ví dụ mô hình mạng

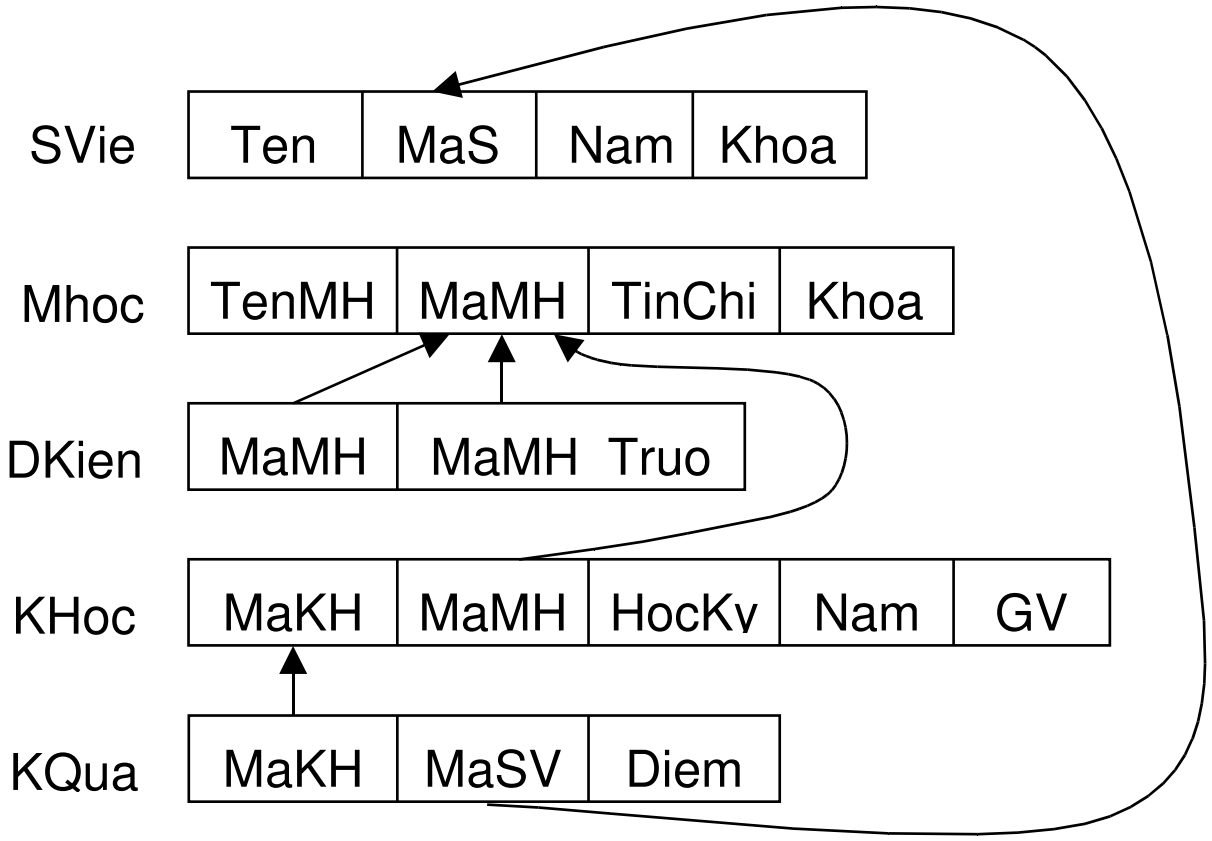


* Ví dụ mô hình phân cấp



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

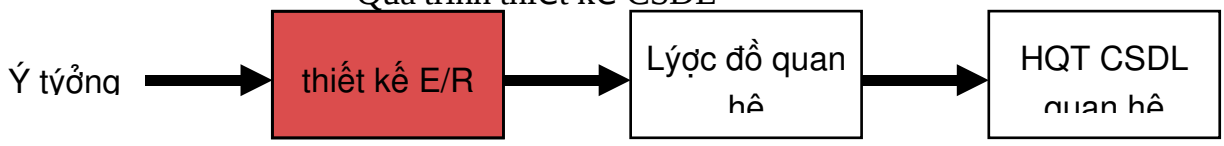
- Ví dụ
  CREATE TABLE employees (
  id INTEGER PRIMARY KEY,
  first_name CHAR(50) null,
  last_name CHAR(75) not null,
  date_of_birth DATE null);
- Ngôn ngữ thao tác dữ liệu
  (DML – Data Manipulation Language)
    - Cho phép truy xuất, thêm, xóa, sửa dữ liệu
    - Mức cao (phi thủ tục)
    - Mức thấp (thủ tục)

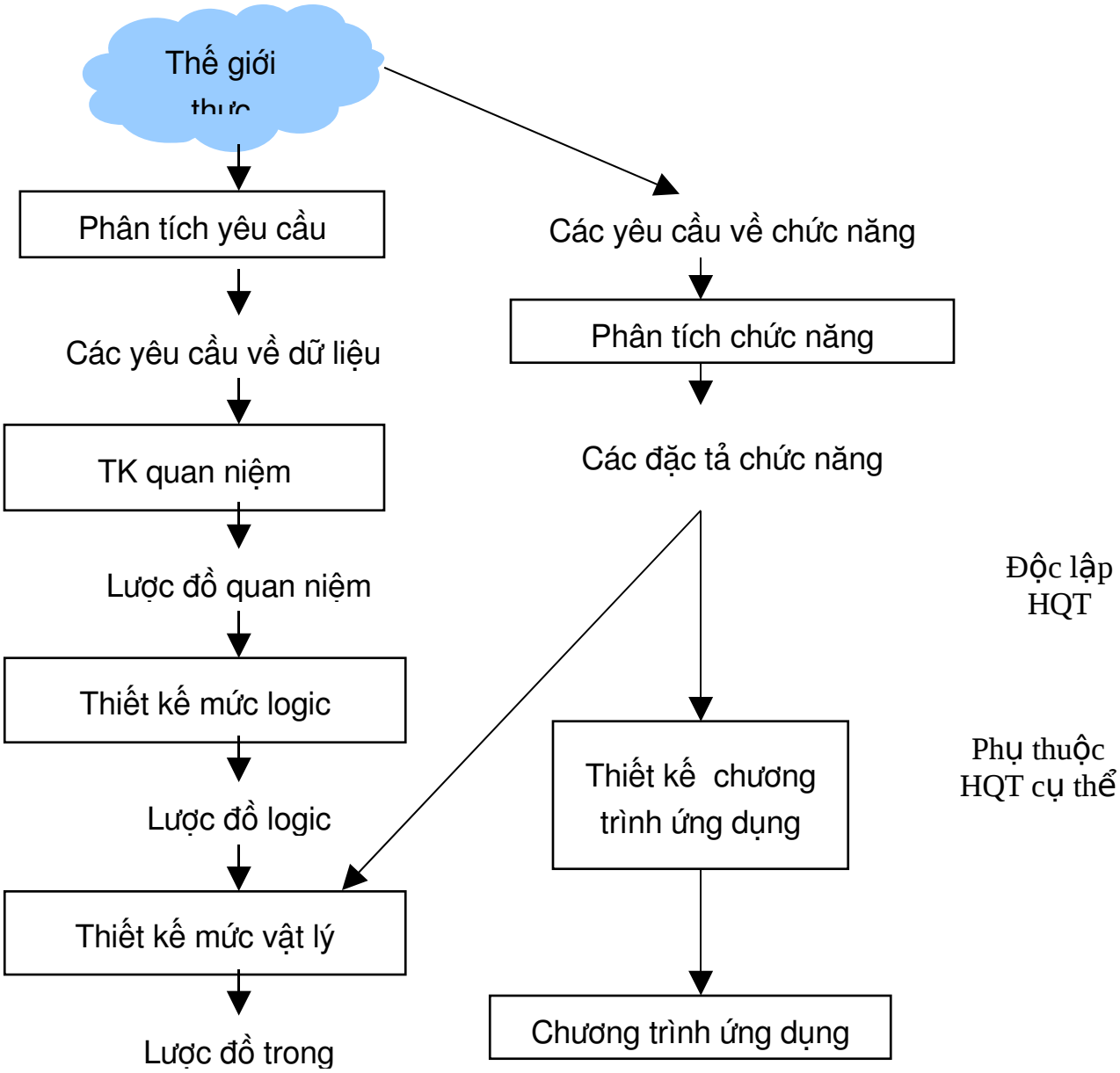
- Ví dụ
  - Các câu lệnh trong SQL: SELECT, INSERT, UPDATE, và DELETE.
  - SELECT id, last_name FROM employees
- Ngôn ngữ điều khiển giao dịch
  (Transaction Control Language - TCL)
  - Đảm bảo tính toàn vẹn dữ liệu khi thực hiện các tác vụ có sự thay đổi dữ liệu
  - Các câu lệnh SQL tương ứng:
    - 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





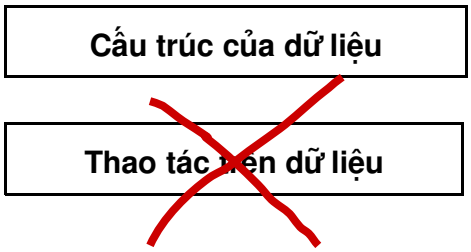
II. Mô hình thực thể - liên kết

- 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

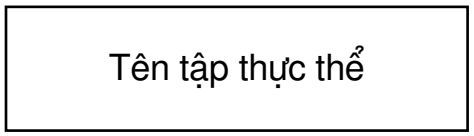
- Quan hệ: Là sự liên kết giữa 2 hay nhiều tập thực thể

- 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

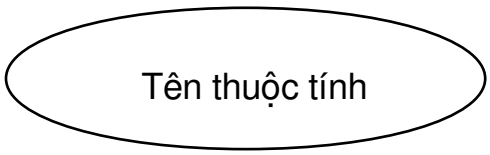
- Tập các quan hệ: là tập hợp các mối quan hệ giống nhau

III. Lược đồ ER

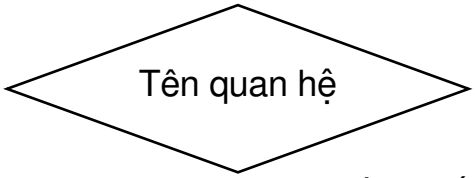
- Là đồ thị biểu diễn các tập thực thể, thuộc tính và mối quan hệ
  - Đỉnh



Tập thực thể



Thuộc tính

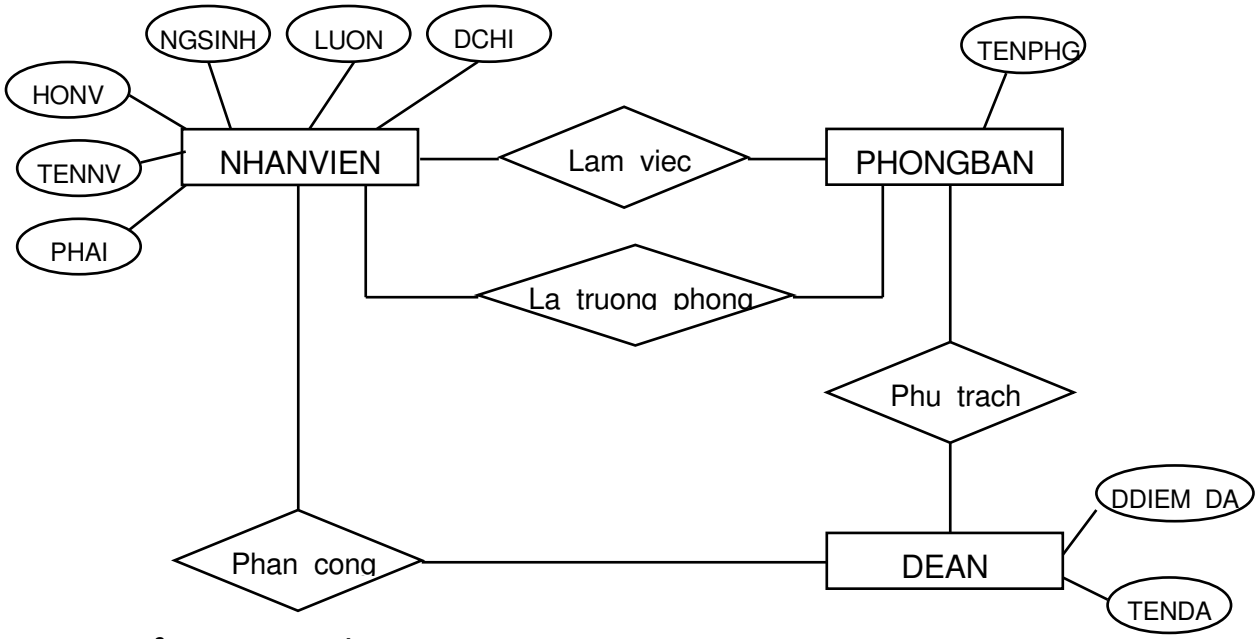


Quan hệ

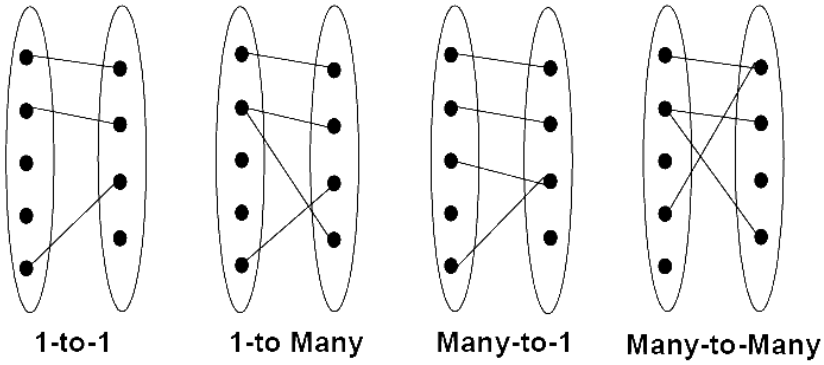
  - Cung là đường nối giữa
    - Tập thực thể và thuộc tính
    - Mối quan hệ và tập thực thể

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”

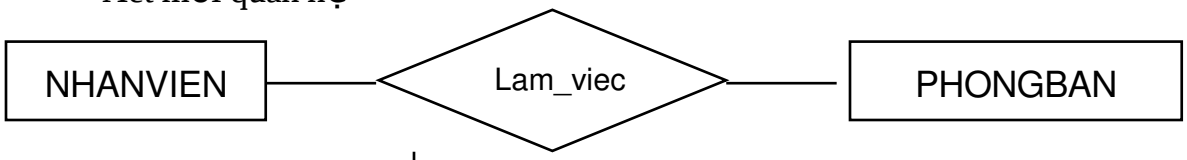
- Chú ý
  - Không lưu trữ lược đồ ER trong CSDL
    - Khái niệm trừu tượng
  - Lược đồ ER chỉ giúp ta *<u>thiết kế CSDL</u>* trước khi chuyển các quan hệ và dữ liệu xuống mức vật lý

**** 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

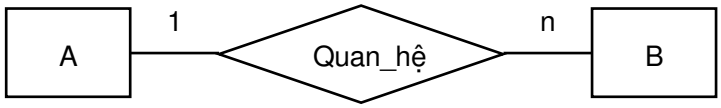


NHANVIEN	PHONGBAN
Tung	Nghien cuu
Hang	Dieu hanh
Vinh	Quan ly

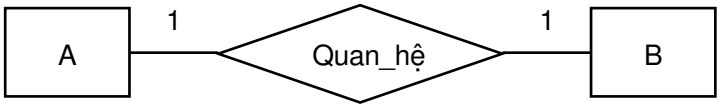
(Tung, Nghien cuu)
(Hang, Dieu hanh)
(Vinh, Quan ly)

- Xét mối quan hệ nhị phân R (binary relationship) giữa 2 tập thực thể A và B, ràng buộc liên kết bao gồm

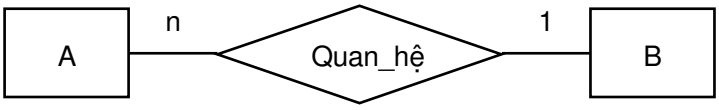
– Một-Nhiều



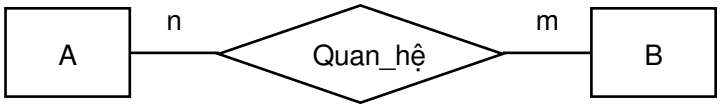
– Một-Một

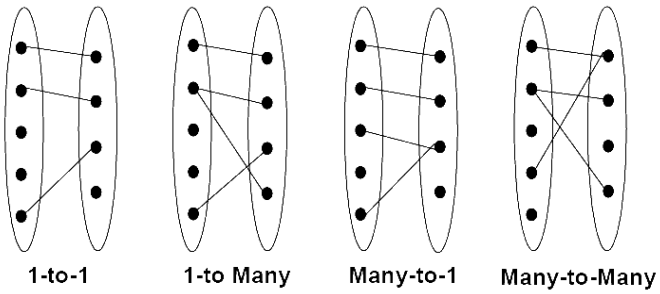


– Nhiều-Một

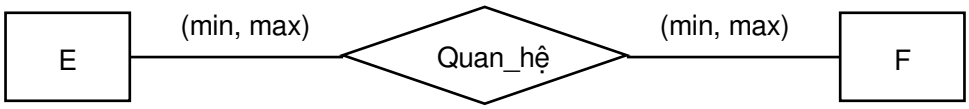


– Nhiều-Nhiều

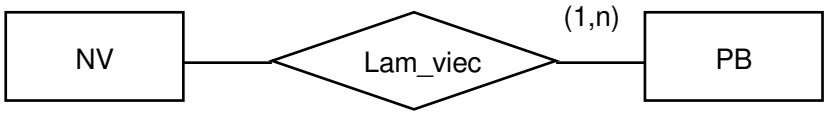




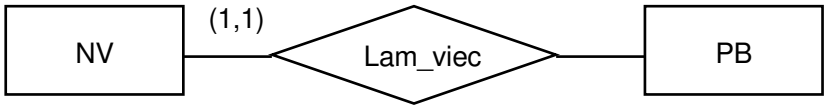
- (min, max) chỉ định mỗi thực thể e ∈ E tham gia ít nhất và nhiều nhất vào thể hiện của R



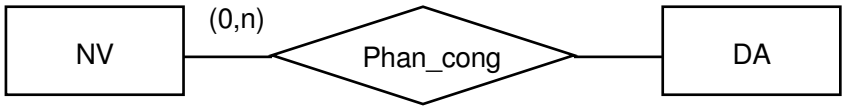
  - (0,1) – không hoặc 1
  - (1,1) – duy nhất 1
  - (0,n) – không hoặc nhiều
  - (1,n) – một hoặc nhiều
- Ví dụ
  - Một phòng ban có nhiều nhân viên



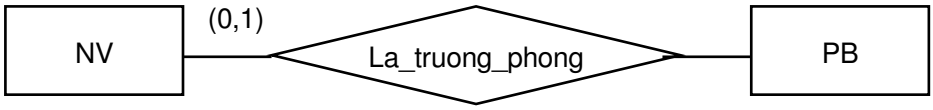
  - Một nhân viên chỉ thuộc 1 phòng ban



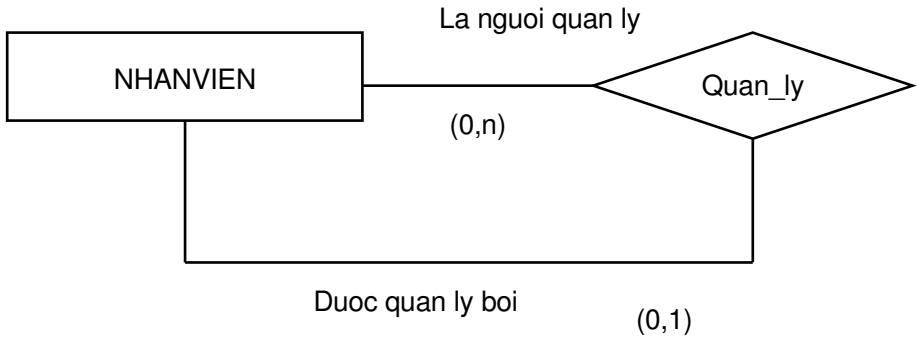
  - Một nhân viên có thể được phân công vào nhiều đề án hoặc không được phân công vào đề án nào



  - Một nhân viên có thể là trưởng phòng của 1 phòng ban nào đó

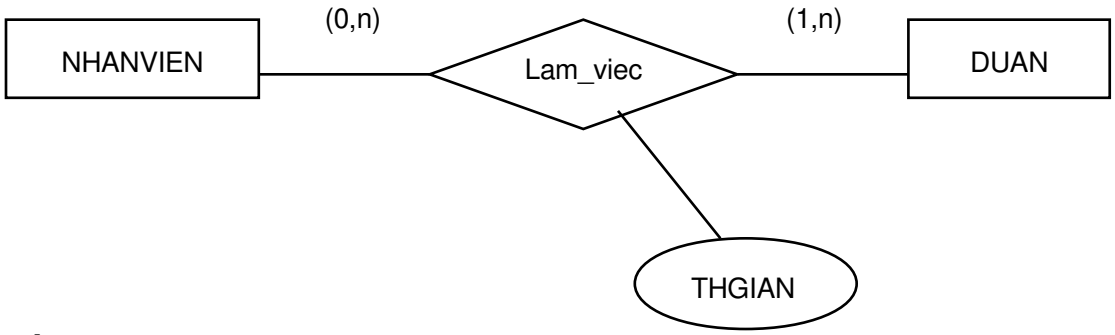


- Một loại thực thể có thể tham gia nhiều lần vào một quan hệ với nhiều vai trò khác nhau



**** Thuộc tính trên mối quan hệ***

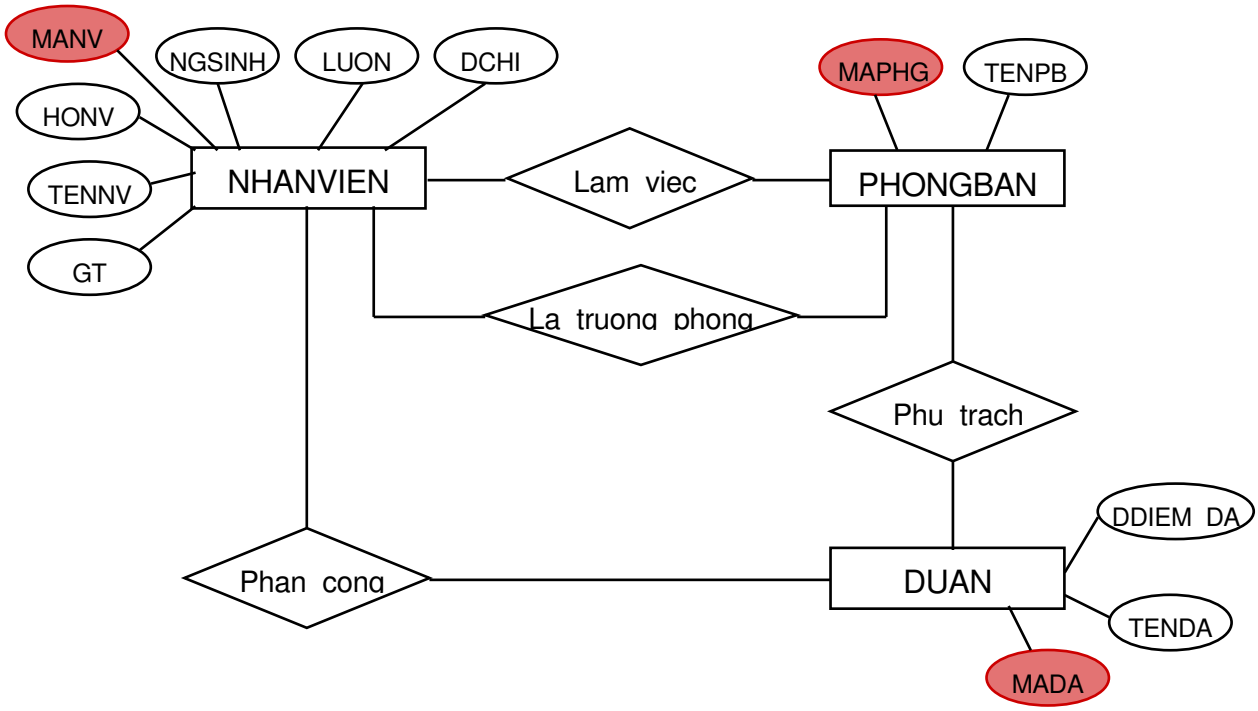
- Thuộc tính trên mối quan hệ mô tả tính chất cho mối quan hệ đó
- Thuộc tính này không thể gắn liền với những thực thể tham gia vào mối quan hệ



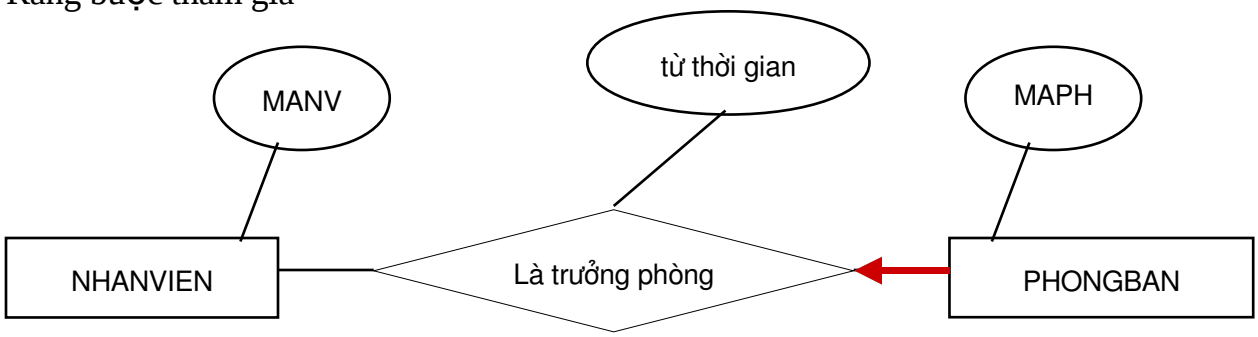
**** Thuộc tính khóa***

- 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



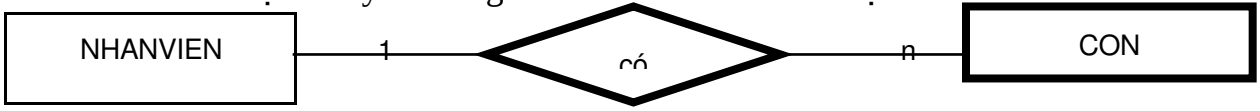
Ràng buộc tham gia



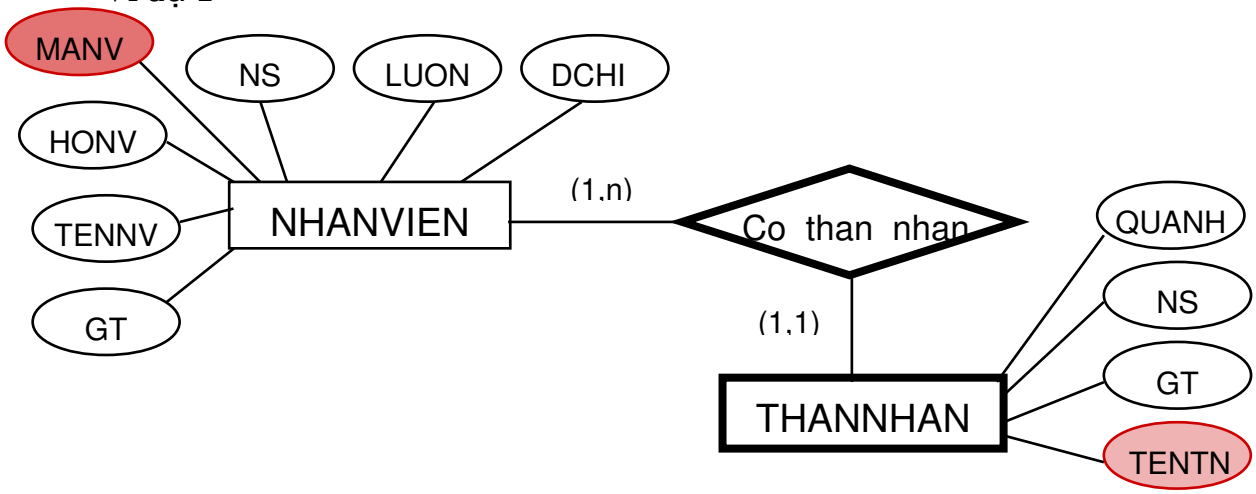
- Xét ví dụ trên
  - Có phải phòng nào cũng có trưởng phòng không?
    - Nếu có → đó là ràng buộc tham gia giữa thực thể NHANVIEN và PHONGBAN.
    - Tham gia toàn bộ vào liên kết
  - Có phải nhân viên nào cũng là trưởng phòng?
    - Sai → *tham gia bộ phận* vào liên kết
- Biểu diễn
  - ← hoặc =

**** Tập thực thể yếu***

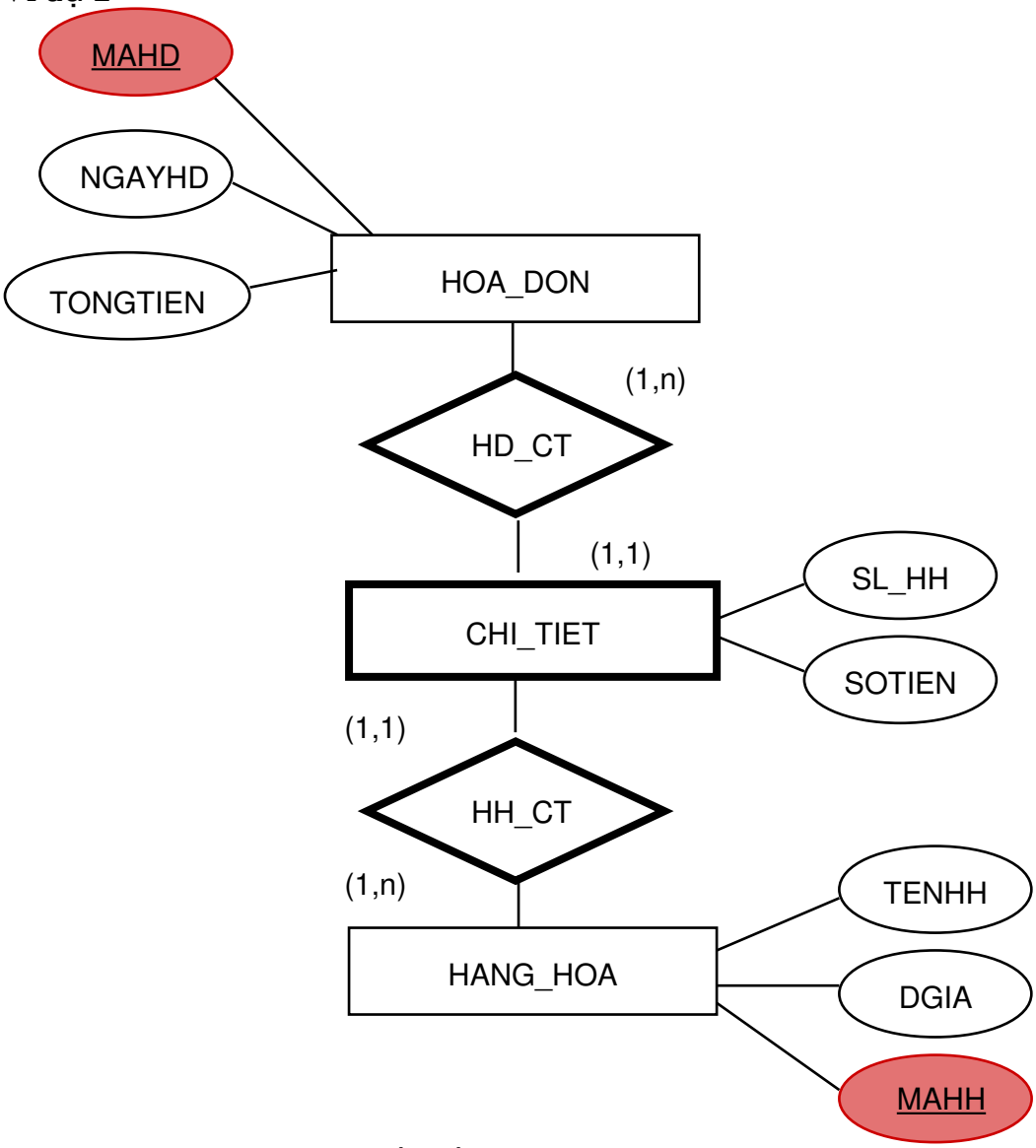
- Là thực thể mà khóa có được từ những thuộc tính của tập thực thể khác
- Thực thể yếu (weak entity set) phải tham gia vào mối quan hệ mà trong đó có một tập thực thể chính (*kiểu thực thể chủ*)
- Mô tả kiểu thực thể yếu bằng hình thoi và hình chữ nhật nét đôi.



- Ví dụ 1

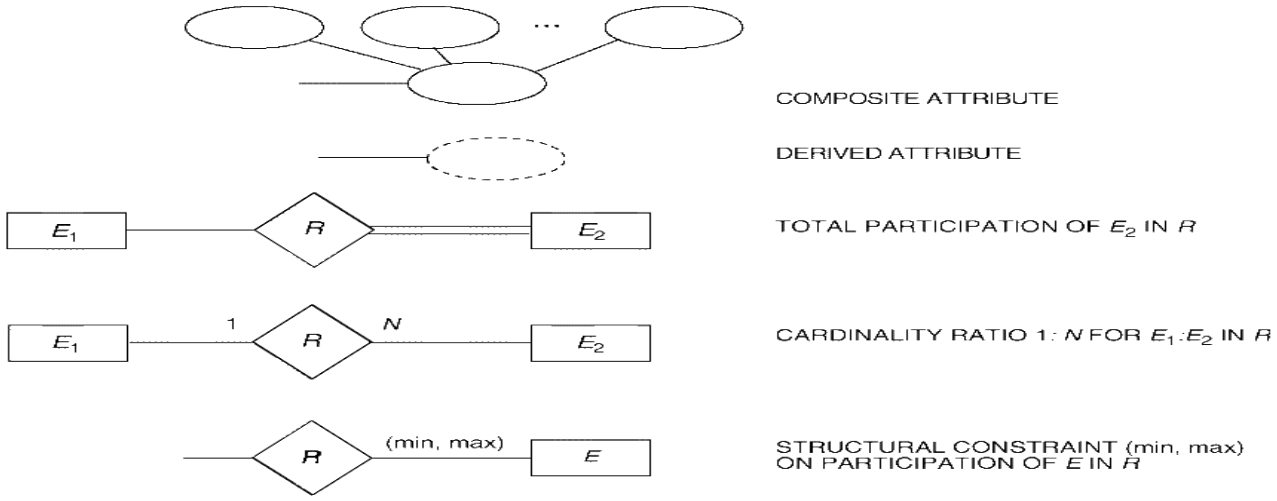


- Ví dụ 2



II. Thiết kế

Các ký hiệu



**** Các bước thiết kế***

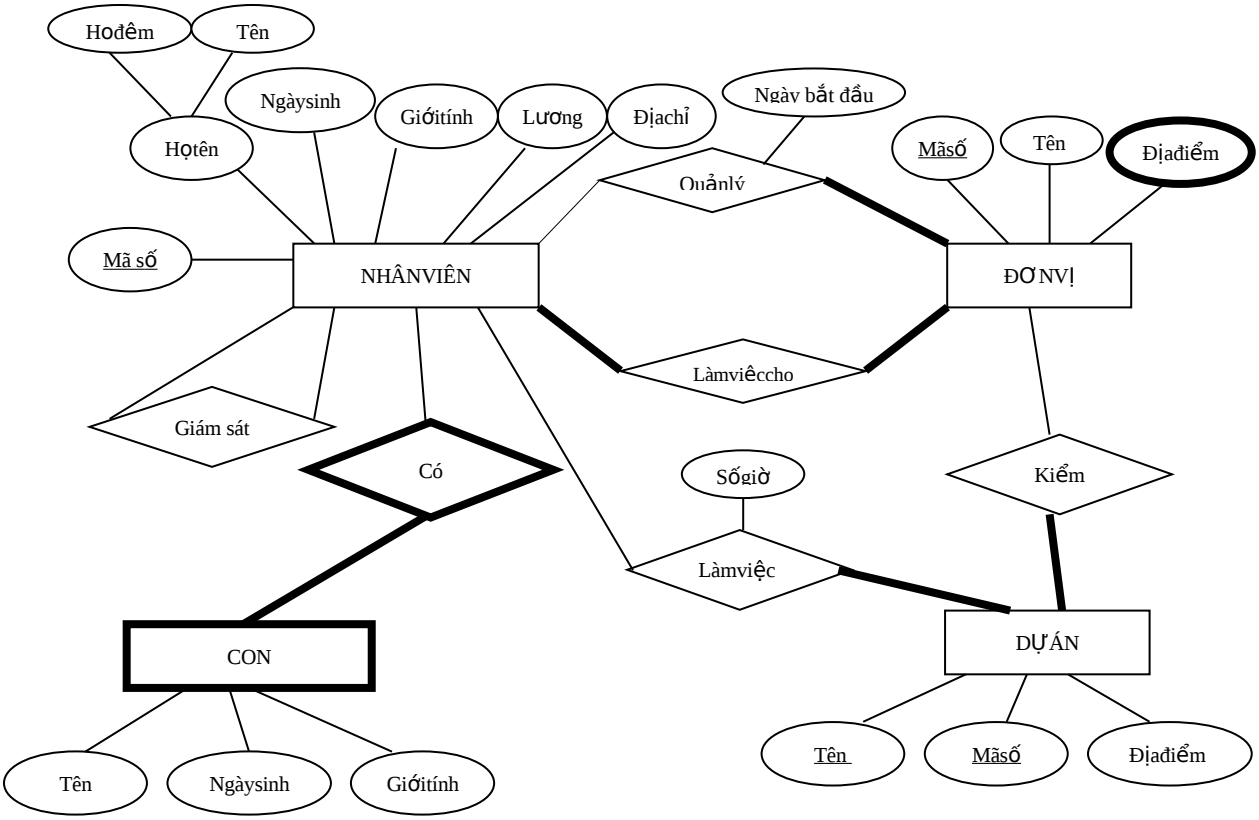
- Xác định tập thực thể
- Xác định mối quan hệ
- Xác định thuộc tính và gắn thuộc tính cho tập thực thể và mối quan hệ
- Quyết định miền giá trị cho thuộc tính
- Quyết định thuộc tính khóa
- Quyết định (min, max) cho mối quan hệ

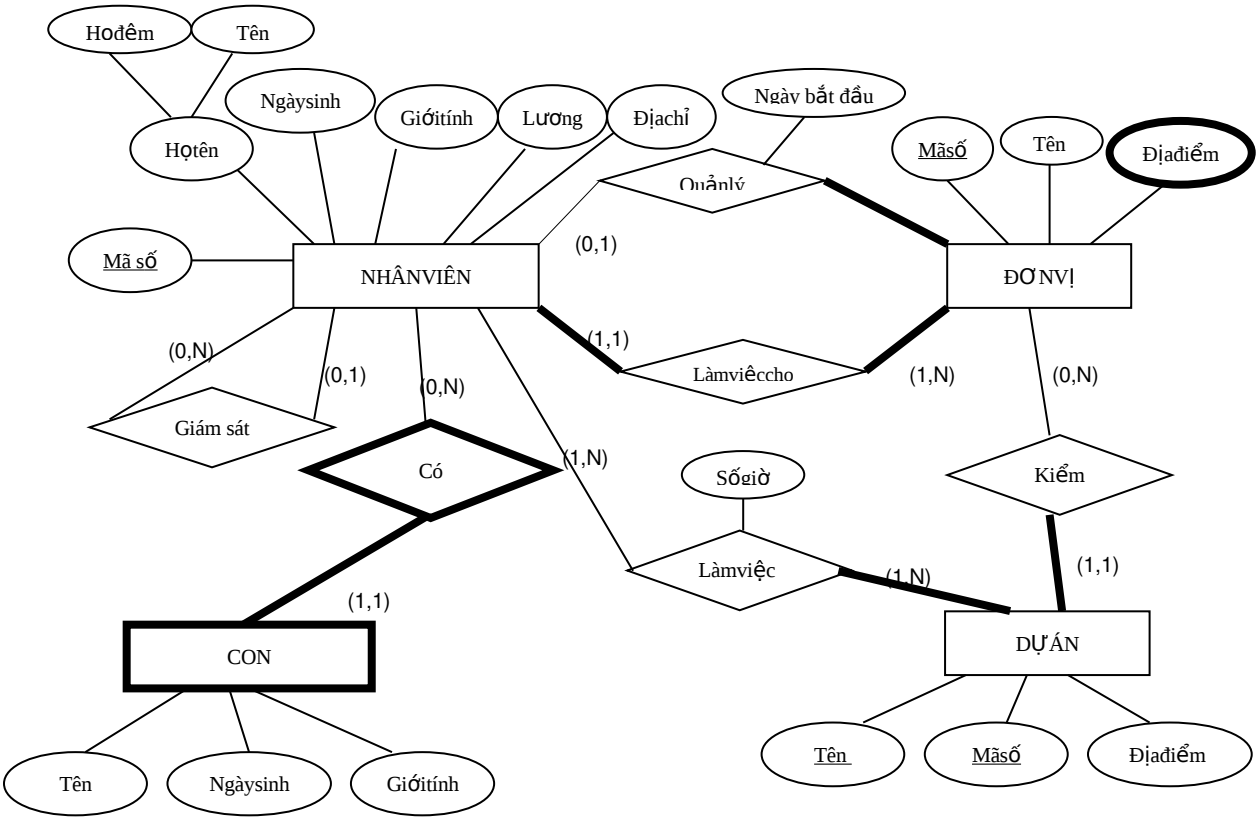
****Qui tắc thiết kế***

- 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)
(1,N)
(0,N)
(0,1)
(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.

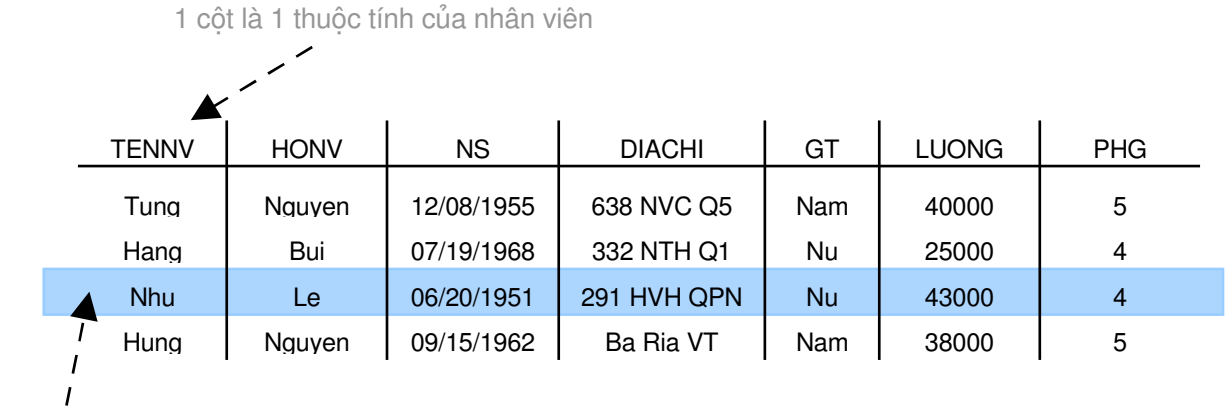
Chương 2 Mô hình dữ liệu quan hệ

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



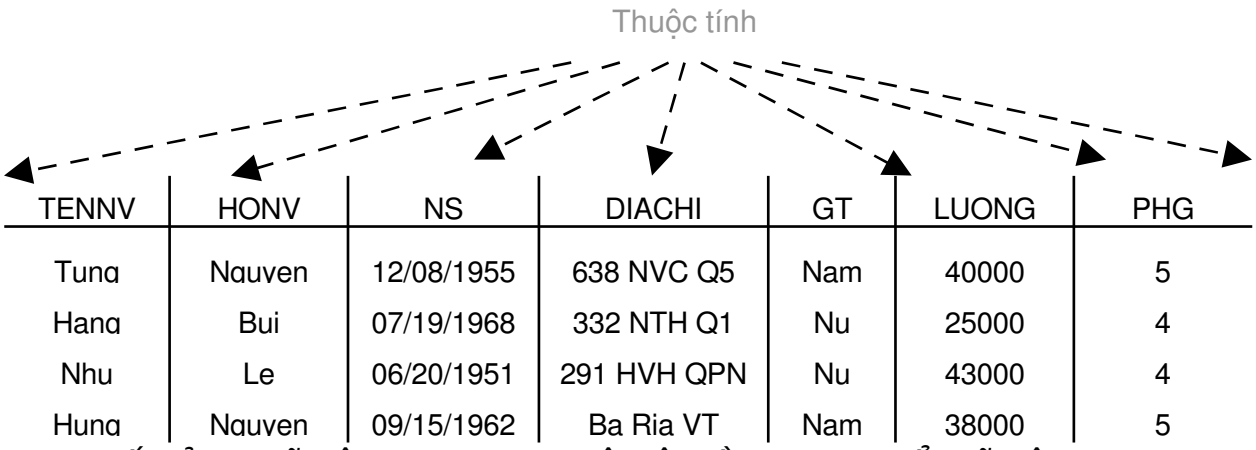
TENNV	HONV	NS	DIACHI	GT	LUONG	PHG
Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	5
Hang	Bui	07/19/1968	332 NTH Q1	Nu	25000	4
Nhu	Le	06/20/1951	291 HVH QPN	Nu	43000	4
Hung	Nguyen	09/15/1962	Ba Ria VT	Nam	38000	5

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

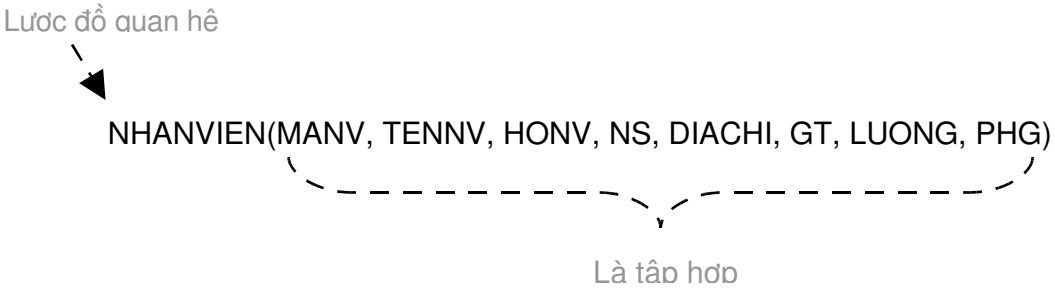


TENNV	HONV	NS	DIACHI	GT	LUONG	PHG
Tung	Nguyen	12/08/1955	638 NVC Q5	Nam	40000	5
Hang	Bui	07/19/1968	332 NTH Q1	Nu	25000	4
Nhu	Le	06/20/1951	291 HVH QPN	Nu	43000	4
Hung	Nguyen	09/15/1962	Ba Ria VT	Nam	38000	5

- 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>

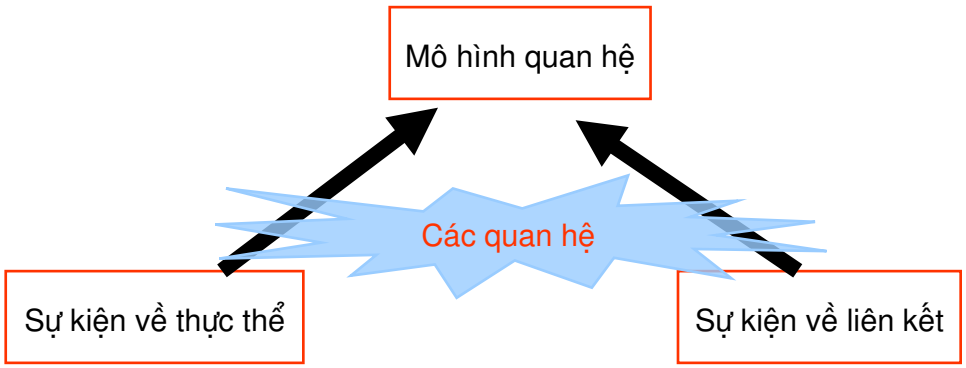
Dữ liệu cụ thể của thuộc tính

* 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 khả hợp nếu
    - Cùng bậc n
    - Và có DOM(Ai)=DOM(Bi) , $1 \leq i \leq n$
- Kết quả của $\cup$, $\cap$, và $-$ là một quan hệ 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

$$\pi_{X,Y}(R) = \pi_X(\pi_Y(R))$$

$$\pi_{A1, A2, ..., An}(\pi_{A1, A2, ..., Am}(R)) = \pi_{A1, A2, ..., An}(\mathbf{R}),\ n \le m$$

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

$\pi_{A1, A2, ..., Ak}(\sigma_P(R))$ $\qquad$ $\sigma_P(\pi_{A1, A2, ..., Ak}(R))$

  - 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

R	A	B
	α	1
	β	2

S	B	C	D
	α	10	+
	β	10	+
	β	20	-
	γ	10	-

unambiguous

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

$$R1 \leftarrow \sigma_{MAPHG \neq 5}(DIADIEM\_PHG)$$

$$R2 \leftarrow \sigma_{DIADIEM=DD}(R1 \times DD\_P5)$$

$$KQ \leftarrow \pi_{MAPHG}(R2)$$

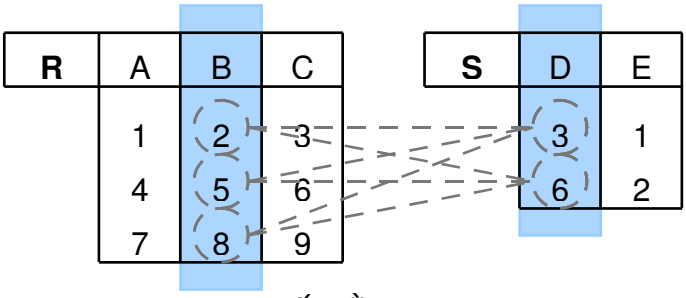
VI. Phép nối

- 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)

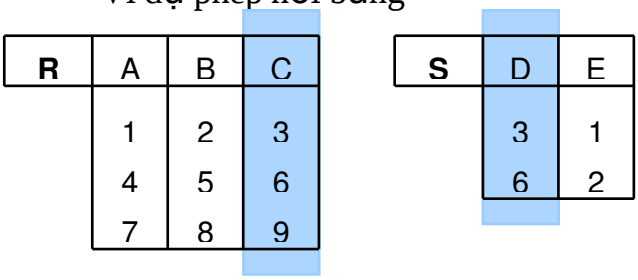
- Mỗi bộ của Q là tổ hợp của 2 bộ trong R và S, thỏa mãn một số <u>điều kiện nối</u> nào đó
  - Có dạng Ai θ Bj
  - Ai là thuộc tính của R, Bj là thuộc tính của S
  - Ai và Bj có cùng miền giá trị
  - θ là phép so sánh ≠ , = , <, >, ≤ , ≥

- 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

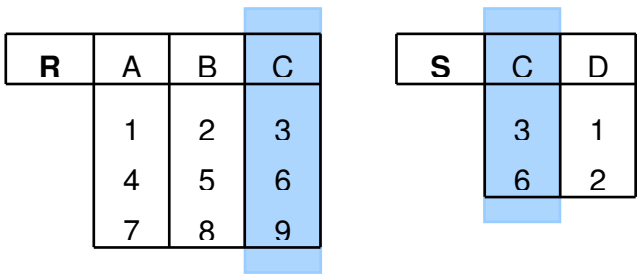
$R \bowtie_{\_\_} S$



- Ví dụ phép nối bằng



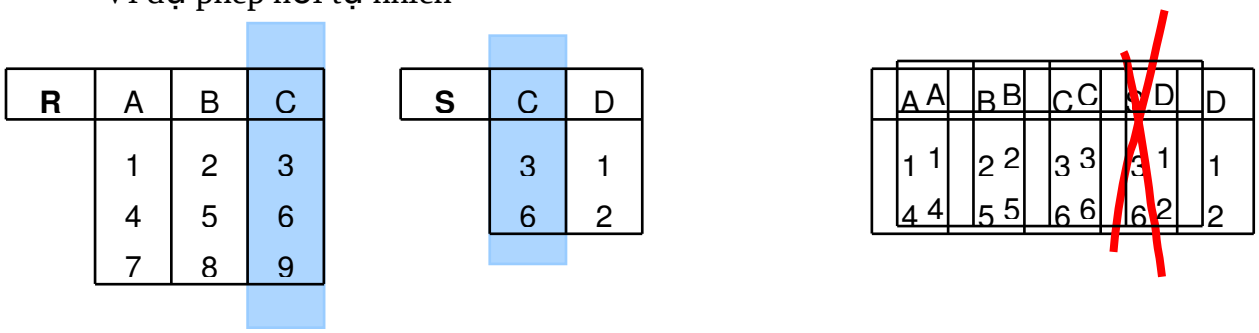
$R \bowtie_{\_\_} S$



$R \bowtie_{\_\_\_} S$

$R \bowtie S$

- 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

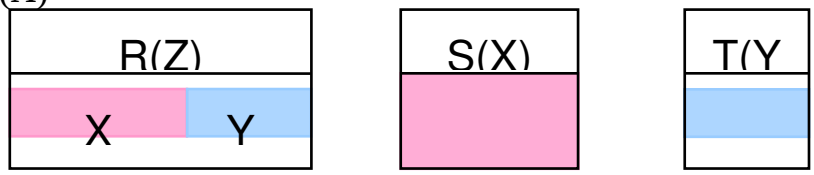
* Tập đầy đủ các phép toán ĐSQH

- 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

$R \div S$

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:

$$P4\_DA \leftarrow \pi_{MADA}(\sigma_{PHG=4}(DEAN))$$

B2:

$$NV\_DA \leftarrow \pi_{MANV, MADA}(PHANCONG)$$

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

$$\mathfrak{I}_{G1, G2, \ldots, Gn}(E)$$

  – 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

$$\mathfrak{I}_{COUNT(), AVERAGE(LUONG)}(NHANVIEN)$$

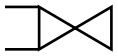
Ví dụ 19

- 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**

  $X \to X$, hay tổng quát hơn nếu $Y \subset X$ thì $X \to Y$
- **A2. Tính bắc cầu**: $X \to Y$ và $Y \to Z$ thì $X \to Z$.
- **A3. Tính mở rộng hai vế**

  $X \to Y$ thì $XZ \to YZ$. (Mở rộng hai vế Z)
- **A4. Tính tựa bắc cầu**: $X \to Y$ và $YZ \to W$ thì $XZ \to W$
- **A5. Tính mở rộng trái thu hẹp phải**

  $X \to Y$ thì $XZ \to Y - W$
- **A6. Tính cộng đầy đủ**: $X \to Y$ và $Z \to W$ thì $XZ \to YW$
- **A7. Tính tích lũy**: $X \to Y$ và $Y \to ZW$ thì $X \to YZW$

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.

$X \to Y$ theo tính mở rộng hai vế

$XZ \to YZ$

Và $YZ \to W$

Theo tính bắc cầu

$XZ \to W$

- **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

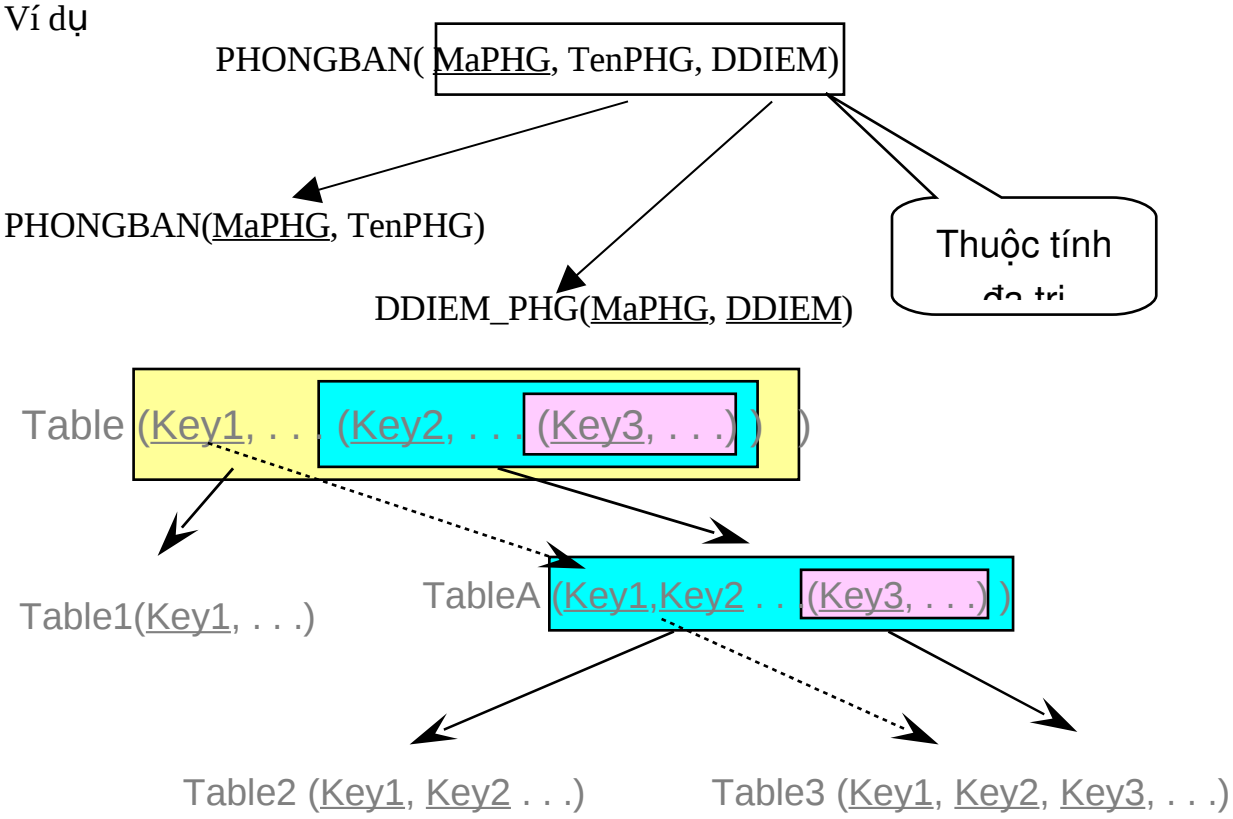
- Mỗi một dạng chuẩn là một tập các điều kiện trên lược đồ nhằm đảm bảo các tính chất của nó (liên quan tới dư thừa và bất thường trong cập nhật)
- Chuẩn hóa dữ liệu: quá trình phân tích lược đồ quan hệ dựa trên các FD và các khóa chính để đạt được
  - Cực tiểu sự dư thừa
  - Cực tiểu các phép cập nhật 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

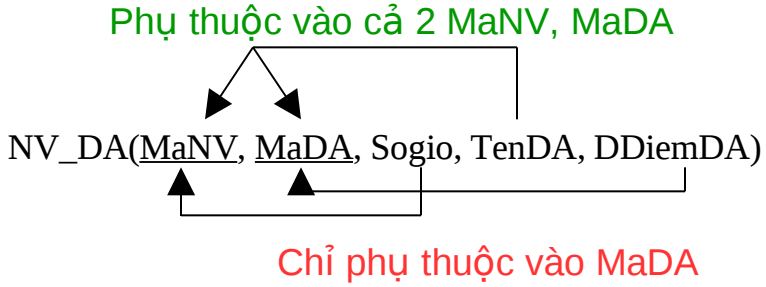
- Phụ thuộc hàm đầy đủ: Một phụ thuộc hàm X $\rightarrow$Y là một phụ thuộc hàm đầy đủ nếu loại bỏ bất kỳ thuộc tính A nào ra khỏi X thì phụ thuộc hàm không còn đúng nữa.

  $\forall$ 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

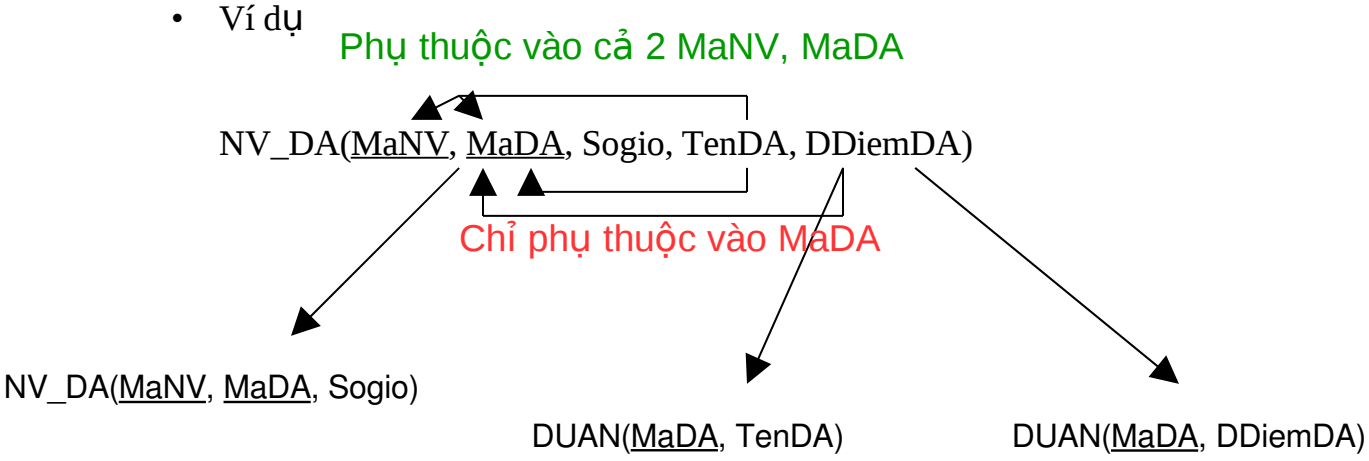
  $\forall$ A$\in$ X, (X – {A}) $\rightarrow$Y
- Tiêu chuẩn Y phụ thuộc đầy đủ vào X

$$\begin{cases} Y \subseteq X_F^+ \\ \forall X' \subset X : Y \not\subset X'^+_F \end{cases}$$

- 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

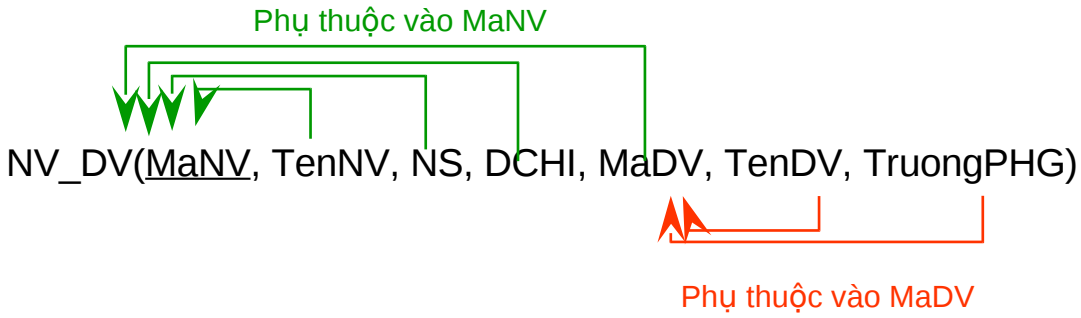


3. Dạng chuẩn 3

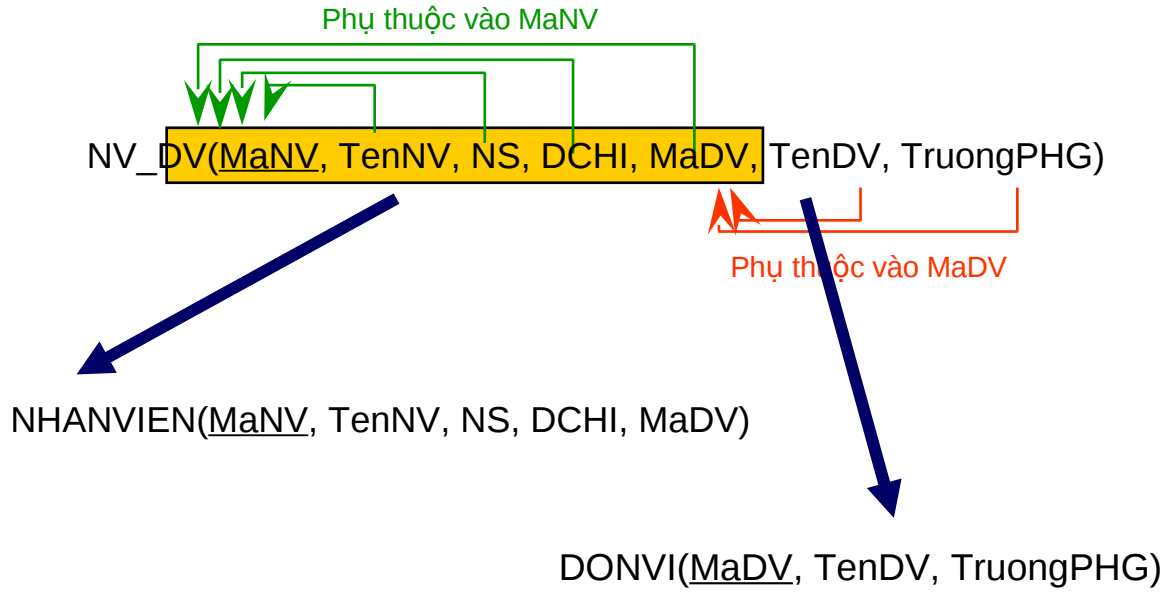
- 3NF dựa trên khái niệm phụ thuộc bắc cầu.
  - Cho lược đồ quan hệ α=<U+, F>, , A là một thuộc tính của U+.
  - Ta nói A phụ thuộc bắc cầu vào X trên α nếu có thể chèn một cầu thực sự Y vào giữa quan hệ A và X. Nghĩa là

$$\begin{cases} X \to Y, Y \to A \ (Y \ không \to X) \\ A \notin XY \end{cases}$$

- 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

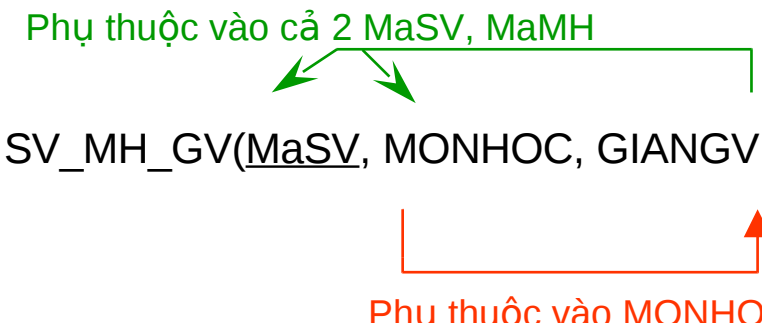
– tách chúng thành một quan hệ riêng có khoá chính là thuộc tính không khóa gây ra phụ thuộc.

- Ví dụ : R (A1,A2,A3,A4,A5)

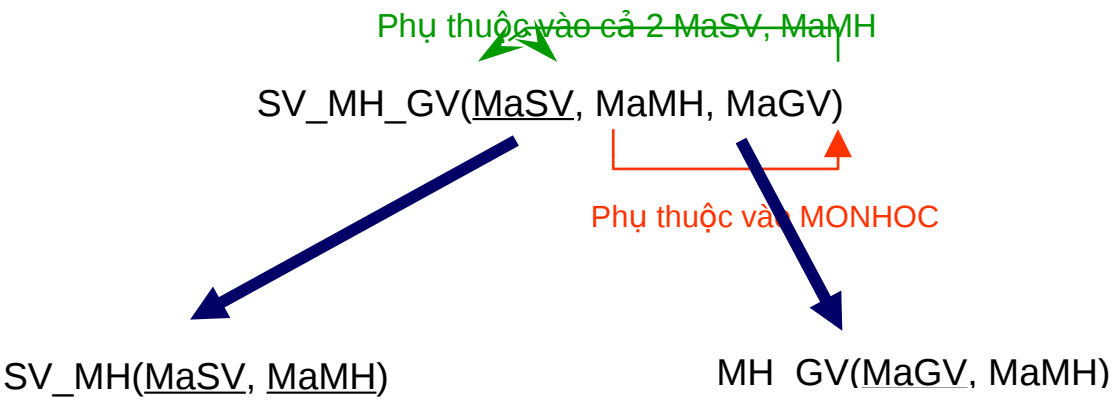
Với các phụ thuộc hàm:

– 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.

$$\alpha = <U^+, F> = \begin{cases} \alpha_1 = <U_1^+, F_1> \\ \alpha_2 = <U_2^+, F_2> \\ ..... \\ \alpha_m = <U_m^+, F_m> \end{cases}$$

- 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 $X \subseteq U_1^+$, $X \neq U_1^+$, $\emptyset$ 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 côt> <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

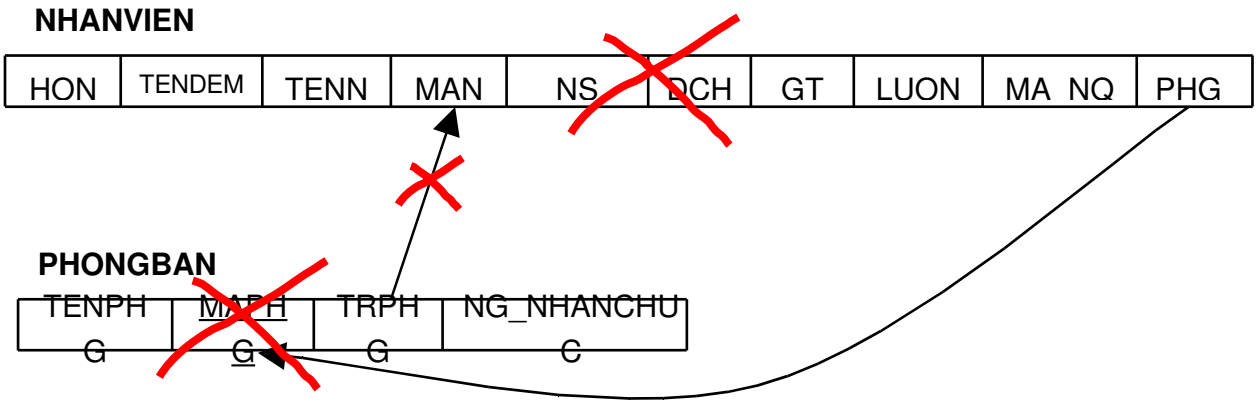
```
CREATE TABLE PHONGBAN (
        TENPB VARCHAR(20),
        MAPHG INT NOT NULL,
        TRPHG CHAR(9),
        NG_NHANCHUC DATETIME
)
ALTER TABLE PHONGBAN ADD
        CONSTRAINT PB_MAPHG_PK PRIMARY KEY (MAPHG),
        CONSTRAINT PB_TRPHG FOREIGN KEY (TRPHG)
            REFERENCES NHANVIEN(MANV),
        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

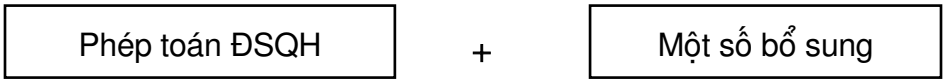
CREATE DOMAIN <Tên_kdl_mới> **AS** <Kiểu_dữ_liệu>

- 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 nhiều dòng trùng nhau
  - Bảng là *bag* ≠ quan hệ là *set*

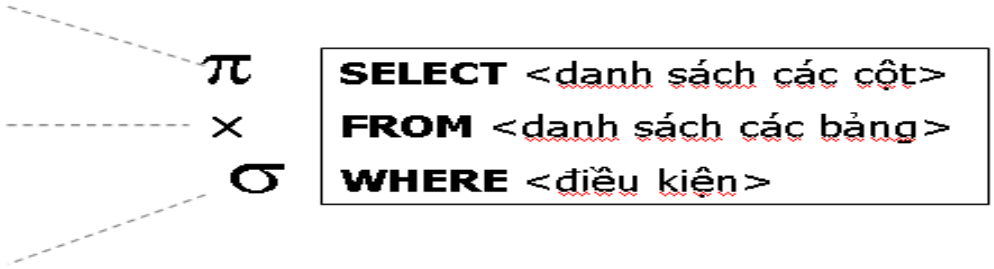
1. Truy vấn cơ bản

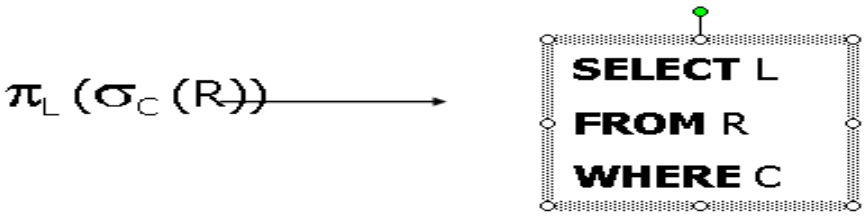
- Gồm 3 mệnh đề

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'
```

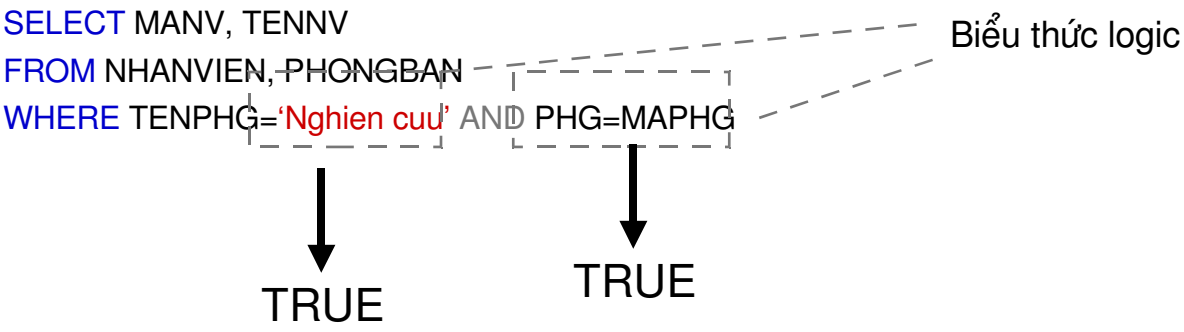
LUONG
30000
25000
38000

- 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)
```

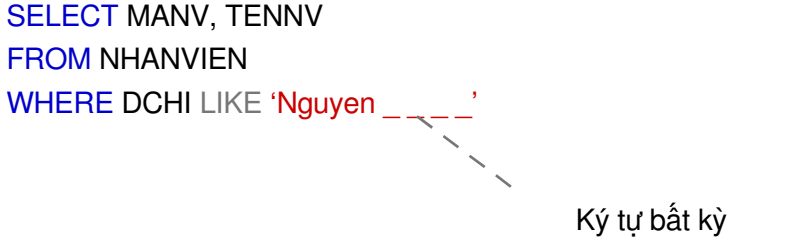
BETWEE

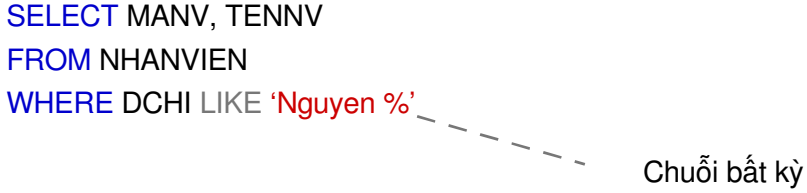
```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE LUONG>20000 AND LUONG<30000
SELECT MANV, TENNV
FROM NHANVIEN
WHERE LUONG BETWEEN 20000 AND 30000
```

NOT BETWEEN

```
SELECT MANV, TENNV
FROM NHANVIEN
WHERE LUONG NOT BETWEEN 20000 AND 30000
```

LIKE





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

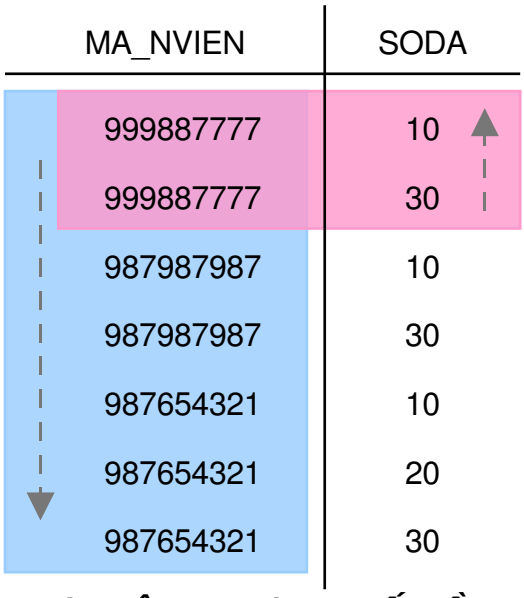
- Dùng để hiển thị kết quả câu truy vấn theo một thứ tự nào đó
- Cú pháp

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 môn Cơ sở dữ liệu

MỘT SỐ KÝ HIỆU VÀ QUY ƯỚC

A, B, C ,…. là tên các thuộc tính đơn.
X, Y, Z, …. là tập hợp các thuộc tính.
t, t_1, t_2,…là các bộ giá trị.
t.[A]: giá trị tại bộ t ứng với thuộc tính A.
t.A: giá trị tại bộ t ứng với thuộc tính A.
F: là tập các phụ thuộc hàm.
f: là kí hiệu của một phụ thuộc hàm
U : Tập hữu hạn các thuộc tính
R, S..: Ký hiệu các quan hệ
r, s...: Ký hiệu lược đồ quan hệ hoặc sơ đồ quan hệ.
R(f) : Ta nói quan hệ R thoả mãn phụ thuộc hàm f
PTH : Phụ thuộc hàm
F ├ f: ta gọi f là một phụ thuộc hàm được suy dẫn logic từ F
X →Y : Y phụ thuộc hàm vào X
X! →Y : Y không phụ thuộc hàm vào X
Sơ đồ quan hệ (lược đồ quan hệ).

DANH SÁCH CÁC HÌNH VẼ VÀ CÁC BẢNG DỮ LIỆU

Chương 1

NHẬP MÔN CƠ SỞ DỮ LIỆU

1.1 Giới thiệu về hề thống quản lý tệp truyền thống

Hệ thống quản lý tệp truyền thống thường được tổ chức riêng rẽ, phục vụ cho một mục đích của một đơn vị hoặc một đơn vị con trực thuộc cụ thể.

Hệ thống quản lý tệp truyền thống cho phép ta tạo các tệp, truy cập và xử lý thông tin trong các tệp thông qua các chương trình ứng dụng. Các phần mềm ứng dụng này được viết bằng các ngôn ngữ lập trình đa năng như PASCAL, C ...

- Ưu điểm:

- Việc xây dựng hệ thống các tệp tin riêng tại từng đơn vị quản lý ít tốn thời gian bởi khối lượng thông tin cần quản lý và khai thác là nhỏ, không đòi hỏi đầu tư vật chất và chất xám nhiều, do đó triển khai ứng dụng nhanh.
- Thông tin được khai thác chỉ phục vụ mục đích hẹp nên khả năng đáp ứng nhanh chóng, kịp thời.

- Nhược điểm:

- Thông tin được tổ chức riêng rẽ ở nhiều nơi nên việc cập nhật dễ làm mất tính nhất quán dữ liệu.
- Hệ thống thông tin được tổ chức thành các hệ thống file riêng lẻ nên thiếu sự chia sẻ thông tin giữa các nơi.
- Có sự dư thừa dữ liệu rất lớn qua việc trùng lặp các tệp tin trong các ứng dụng khác nhau.
- Không gian đĩa bị lãng phí, khó khăn trong việc bảo trì hệ thống.
- Khó khăn trong việc truy xuất dữ liệu.

Một ví dụ điển hình về sự trùng lắp dữ liệu như trong Hệ quản lý nguồn nhân lực bao gồm ba hệ chính:

1. Hệ lương, hệ này duy trì ngày công và lương cho tất cả nhân viên.

2. Hệ nhân sự, hệ này duy trì lý lịch cá nhân, dữ liệu về tổ chức, công việc đào tạo và vị trí thăng tiến.

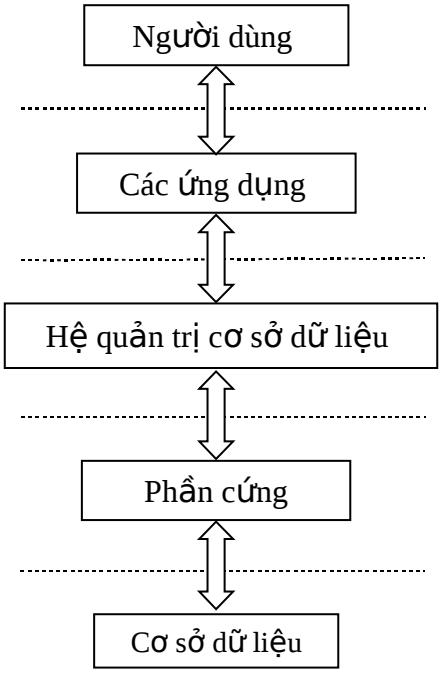
3. Hệ hưu, hệ này quản trị các qui tắc liên quan đến nghỉ hưu, loại nghỉ hưu. Chi tiết về hưu của từng nhân viên.

Vấn đề bất lợi là Hệ quản lý lương thông thường được quản lý bởi phòng Tài chính, trong khi Hệ quản lý nhân sự và Hệ quản lý hưu được quản lý bởi phòng Tổ chức cán bộ. Rõ ràng, có nhiều dữ liệu về nhân viên là chung cho cả ba hệ. Thường những hệ này thực hiện và lưu trữ riêng biệt nên chúng tạo ra sự trùng lặp dữ liệu.

Qua phân tích trên, chúng ta nhận thấy việc tổ chức dữ liệu theo hệ thống tệp hoàn toàn không phù hợp với những hệ thống thông tin lớn. Việc xây dựng một hệ thống thông tin đảm bảo được tính nhất quán dữ liệu, đáp ứng được nhu cầu khai thác đồng thời của nhiều người là thực sự cần thiết.

1.2. Hệ cơ sở dữ liệu

1.2.1 Các thành phần của hệ cơ sở dữ liệu.



Hình 1.1 : Các thành phần của một hệ cơ sở dữ

Các thành phần trong hệ CSDL gồm:

- ***Người dùng (User), gồm có 4 đối tượng sử dụng:***

+ *Người quản trị cơ sở dữ liệu:* Trong những tổ chức có nhiều người cùng sử dụng chung một nguồn dữ liệu thì nhất thiết phải có một người đứng đầu quản lý, chịu trách nhiệm đối với nguồn dữ liệu này. Đó chính là người quản trị cơ sở dữ liệu (Database Administrators - DBA). DBA có nhiệm vụ tổ chức nội dung của cơ sở dữ liệu, tạo và cấp quyền truy cập cơ sở dữ liệu cho người dùng, đưa ra yêu cầu về phần cứng và phần mềm… nếu cần thiết. DAB cũng phải chịu trách nhiệm bảo vệ an toàn, Backup thông tin…khi có sự cố.

+ *Người phân tích và thiết kế hệ thống:* Là người chịu trách nhiệm: (a) xác định những dữ liệu nào cần lưu trữ trong CSDL; (b) lựa chọn những cấu trúc thích hợp để biểu diễn và lưu trữ; (c) phỏng vấn tất cả những người sử dụng CSDL sau này để hiểu được những yêu cầu của họ đối với CSDL; (d) tiến hành phân tích thiết kế hệ thống sau khi thống nhất được tất cả các yêu cầu của người sử dụng.

+ *Người viết chương trình ứng dụng:* Là người viết phần mềm phục vụ cho việc thực hiện các chức năng của hệ thống bằng những ngôn ngữ phù hợp, ngoài ra còn có các nhiệm vụ: (a) chạy thử chương trình (test); (b) chữa lỗi và gỡ rối chương trình (debug); (c) viết tài liệu, hướng dẫn sử dụng; (d) bảo trì hệ thống

+ *Người dùng cuối (EndUser):* Người dùng cuối là những người truy cập CSDL để: (a) cập nhật dữ liệu; (b) cruy vấn dữ liệu; (c) thống kê, báo cáo. Mỗi EndUse chỉ có một quyền hạn trong phạm vi nhất định đỗi với cơ sở dữ liệu như quyền đọc, ghi, copy...)

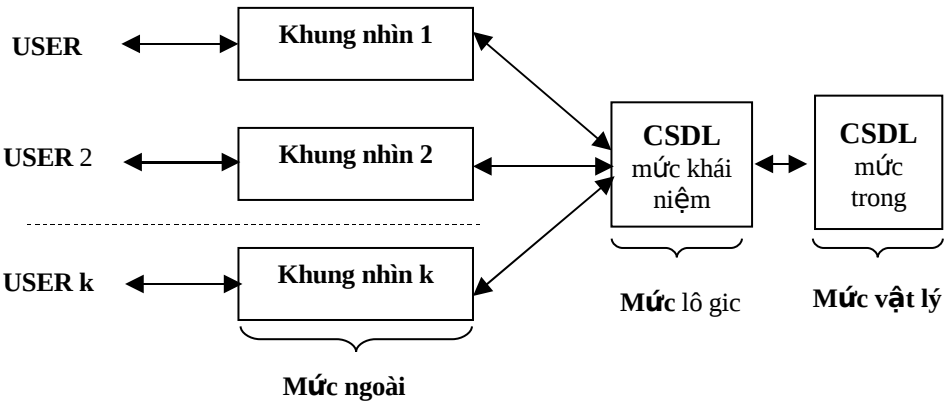
- ***Các ứng dụng:*** Các thao tác cần thiết truy cập vào cơ sở dữ liệu như tạo lập, xử lý, cập nhật dữ liệu.

- ***Hệ quản trị cơ sở dữ liệu:*** Hệ quản trị cơ sở dữ liệu là phần mềm cho phép định nghĩa các cấu trúc để lưu trữ dữ liệu và các thao tác trên dữ liệu sao cho đảm bảo sự an toán và bí mật của dữ liệu. Hiện nay có một số hệ quản trị cơ sở dữ liệu thông dụng như FOXPRO, ACCESS, SQL SERVER, ORACLE...

- ***Phần cứng:*** Phần cứng là các thiết bị và các phương tiện được sử dụng để lưu trữ và truy cập vào cơ sở dữ liệu.

- ***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 (như băng từ, đĩa từ…), để có thể thoả mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với những mục đích sử dụng khác nhau.

1.2.2. Kiến trúc của một hệ cơ sở dữ liệu



Hình 1.2: Cấu trúc của một hệ cơ sở dữ liệu

Cấu trúc một hệ cơ sở dữ liệu gồm ba mức:

+ *Mức ngoài:* Là mức sát với người sử dụng nhất, là cách nhìn, là quan niệm của từng người sử dụng đối với cơ sở dữ liệu mức khái niệm. Khả năng truy nhập tuỳ thuộc vào quyền hạn từng USER.

+ *Mức logic (CSDL mức khái niệm):* Là tập các dữ liệu được biểu diễn dưới dạng trừu tượng của cơ sở dữ liệu vật lý.

+ *Mức vật lý:* Là tập các dữ liệu được biểu diễn theo một cấu trúc nào đó, được lưu trên các thiết bị nhớ thứ cấp (như đĩa từ, băng từ …).

1.3. Phân loại các hệ cơ sở dữ liệu

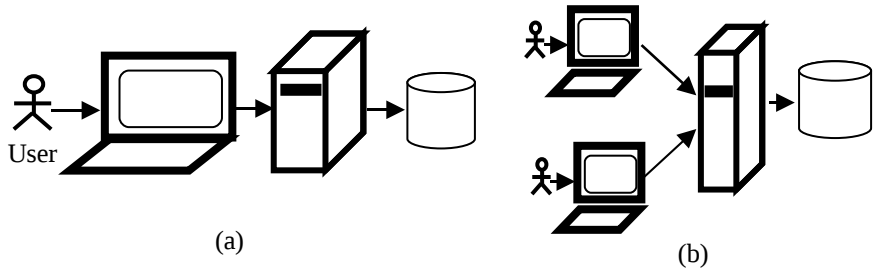
1.3.1. Các hệ tập trung

Hệ cơ sở dữ liệu tập trung là hệ trong đó CSDL được lưu trữ tại một vị trí nhất định, gồm các hệ cơ sở dữ liệu sau:

- *Hệ cơ sở dữ liệu cá nhân (Personal Database):* Mô hình này là một hệ cơ sở dữ liệu nhỏ chỉ gồm một máy tính cá nhân với một vài người sử dụng làm nhiệm vụ đơn lẻ với quy mô nhỏ.

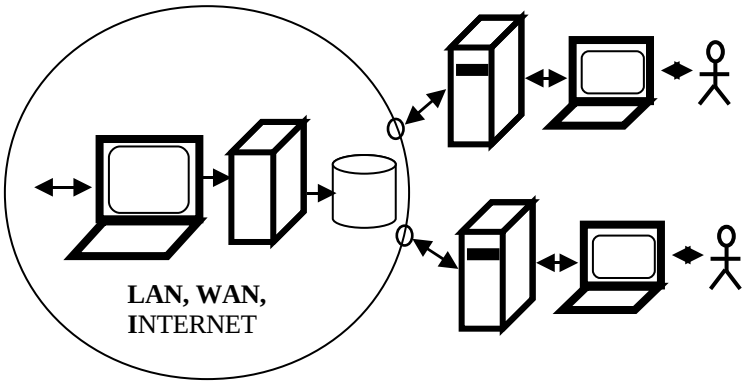
- *Hệ cơ sở dữ liệu trung tâm (Central Database):* Hệ cơ sở dữ liệu trung tâm là một hệ đa người dùng từ thiết bị đầu cuối (terminal) tại đó có màn hình và phím để trao đổi thông tin. Mọi xử lý, tính toán được thực hiện tại trung tâm với một máy tính mạnh có thể xử lý nhiều yêu cầu.

Một hệ như vậy khi máy tính trung tâm có sự cố, thì toàn bộ hệ thống sẽ ngừng hoạt động.



Hình 1.3: Hệ cơ sở dữ liệu; (a) Cá nhân; (b) Trung tâm

- *Hệ cơ sở dữ liệu Client/Server (Client/Server Database):* Cơ sở dữ liệu được lưu trữ tại máy chủ (Server) và nhiều máy trạm (Client) kết nối sử dụng chung cơ sở dữ liệu này.

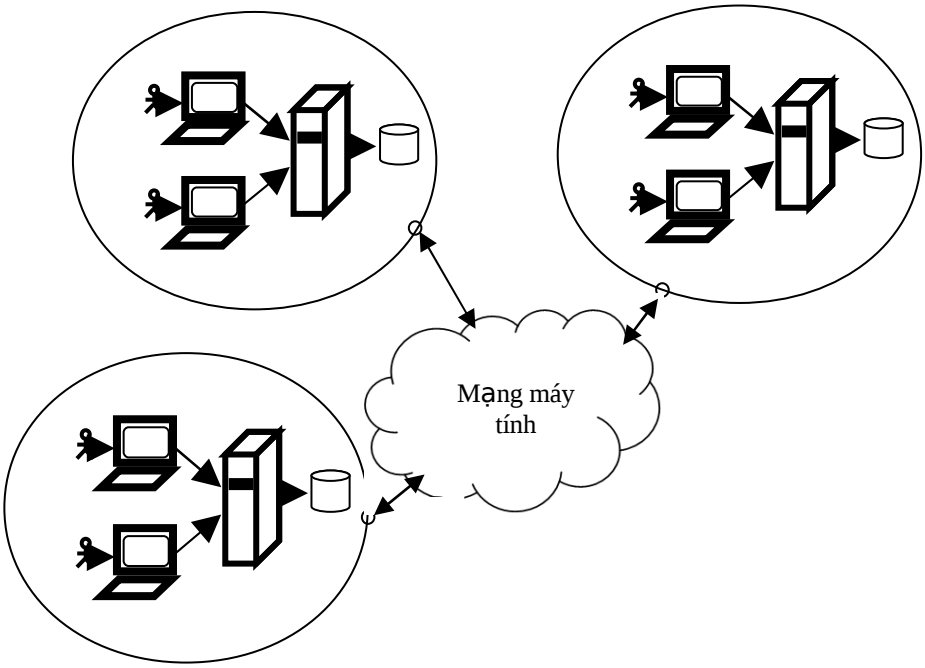


Hình 1.4 Kiến trúc Client/Server

1.3.2. Các hệ cơ sở dữ liệu phân tán

Hệ cơ sở dữ liệu phân tán là hệ CSDL trong đó cơ sở dữ liệu được tổ chức phân bố thành nhiều cơ sở dữ liệu địa phương, được lưu trữ trên các máy tính ở các vị trí địa lý khác nhau nhưng vẫn cùng thuộc một hệ thống. Các cơ sở

dữ liệu này được liên kết với nhau qua mạng máy tính phục vụ nhu cầu người dùng ở nhiều địa điểm khác nhau ở mức trong suốt.



Hình 1.5: Hệ cơ sở dữ liệu phân tán

Hệ cơ sở dữ liệu phân tán được phân thành hai loại

+ *Hệ thuần nhất:* Trong các hệ CSDL địa phương biểu diễn theo những mô hình giống nhau phương thức truy cập giống nhau.

+ *Hệ không thuần nhất:* Ngược lại với mô hình hệ thuần nhất là hệ không thuần nhất.

1.4. Những ưu điểm của việc xây dựng một hệ cơ sở dữ liệu

- Đảm bảo sự độc lập dữ liệu: Dữ liệu độc lập với chương trình làm cho dữ liệu được sử dụng rộng rãi và thuận lợi hơn.

- Giảm thiểu việc dư thừa dữ liệu: Khác với hệ thống tệp, hệ thống cơ sở dữ liệu tổ chức theo cấu trúc thống nhất, hợp lý hạn chế việc lưu trữ tại nhiều nơi.

- Đảm bảo tính nhất quán và toàn vẹn dữ liệu: Do ít dư thừa nên hạn chế được sự dị thường khi thay đổi, cập nhật.

- Tăng tính dùng chung: Cơ sở dữ liệu có khả năng cho nhiều người truy cập sử dụng mỗi người nhìn vào cơ sở dữ liệu như nó là của riêng mình không bị ảnh hưởng bởi người khác.

- Tăng khả năng phát triển các ứng dụng: Do có sự mở rộng giao lưu nên khả năng sáng tạo cải tiến thuận lợi hơn.

- Tính chuẩn hoá cao.

- Chất lượng dữ liệu được cải thiện.

- Giảm bớt chi phí bảo trì hệ thống.

1.5. Tính độc lập dữ liệu

Tính độc lập dữ liệu là sự bất biến của chương trình ứng dụng đối với các thay đổi trong cấu trúc lưu trữ và chiến lược truy nhập vào cơ sở sữ liệu. Tính độc lập dữ liệu ở đây có hai mặt:

- Độc lập về vật lý: Là sự độc lập trong lưu trữ, chương trình ứng dụng không phụ thuộc vào việc dữ liệu được lưu giữ ở đâu hoặc lưu giữ như thế nào trên thiết bị nhớ thứ cấp.

- Độc lập về lôgic: Sự thay đổi, thêm bớt thông tin về các thực thể ở mức quan niệm không đòi hỏi thay đổi các khung nhìn của người sử dụng dẫn tới không cần thay đổi chương trình ứng dụng.

1.6 Hệ quản trị cơ sở dữ liệu

1.6.1 Các chức năng của một hệ quản trị CSDL

Một hệ quản trị CSDL thực hiện các chức năng sau:

+ Tạo cấu trúc dữ liệu tương ứng với mô hình dữ liệu được chọn.

+ Đảm bảo tính độc lập dữ liệu.

+ Cho phép cập nhật dữ liệu.

+ Kết xuất ra được các báo cáo từ các dữ liệu trong CSDL.

+ Đảm bảo tính an toàn và toàn vẹn dữ liệu trong CSDL.

+ Cung cấp các tiện ích sao lưu phục hồi dữ liệu.

+ Cung cấp các thủ tục điều khiển tương tranh.

1.6.2. Các thành phần của một hệ QTCSDL

Một hệ quản trị thông thường có các thành phần chính sau:

+ Ngôn ngữ định nghĩa dữ liệu (Data Definition Language).

+ Ngôn ngữ thao tác dữ liệu (Data Manipulation Language).

+ Ngôn ngữ hỏi đáp dữ liệu (Query Language).

+ Bộ viết báo cáo.

+ Từ điển dữ liệu.

+ Bộ phát sinh đồ hoạ.

1.7. Các mô hình dữ liệu.

Mô hình dữ liệu cho phép người dùng biểu diễn cơ sở dữ liệu dưới cấu trúc thuật ngữ dễ hiểu. Một mô hình dữ liệu là một hình thức mô tả toán học bao gồm:

+ Một hệ thống các ký hiệu để mô tả dữ liệu.

+ Tập các phép toán để thao tác trên cơ sở dữ liệu.

Vào những năm đầu của thập kỷ 60 (thế kỷ 20), mô hình mạng và mô hình phân cấp là thế hệ đầu tiên của họ các mô hình dữ liệu. Sang đầu thập kỷ 70. E.F. Codd đề xuất mô hình quan hệ mới, đó chính là thế hệ thứ hai. Mô hình quan hệ này có cấu trúc chặt chẽ, sáng sủa, nhất quán và có tính trực quan cao.

1.6.1 Khái niệm về thực thể và liên kết.

a) Thực thể

Thực thể (entity) là đối tượng cụ thể hay trừu tượng mà ta cần quan tâm trong công tác quản lý. Tên thực thể là danh từ.

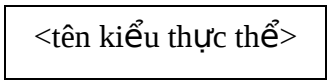
Thí dụ 1.1: Quản lý thư viện ta có các thực thể như: “Sách”, “Độc giả”... là các đối tượng cụ thể. Các đối tượng trừu tượng có thể là: Khoa công nghệ thông tin, Ngành toán ứng dụng….

b) Kiểu thực thể

Kiểu thực thể là tập hợp các thực thể (đối tượng) cùng được mô tả bằng những đặc trưng, tính chất giống nhau.

Thí dụ 1.2: Một nhân viên là một thực thể, tập hợp các nhân viên của cùng một hệ thống tạo thành một kiểu thực thể.

Biểu diễn một kiểu thực thể: Là một hình chữ nhật bên trong ghi tên của kiểu thực thể.



Thí dụ 1.3: Biểu diễn các thực thể « Nhân viên », « Sách », « Độc giả »:

Ghi chú: Thể hiện của kiểu thực thể là một thực thể, nó là một phần tử trong tập hợp hay lớp của kiểu thực thể. Vì vậy trong các ứng dụng để tránh sử dụng nhiều khái niệm ta đồng nhất thực thể và kiểu thực thể.

c) Thuộc tính (Attribute)

Thuộc tính là dữ liệu dùng để mô tả một đặc trưng của thực thể. Mỗi thực thể có một tập các thuộc tính. Tên thuộc tính phải là danh từ.

Thí dụ 1.4: Thực thể "Sách" có các thuộc tính: Tên sách, tên tác giả, nhà xuất bản...

1.6.2. Liên kết và kiểu liên kết

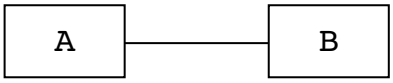
Liên kết: Là một sự ghép nối giữa hai hay nhiều thực thể phản ánh một thực tế quản lý.

Thí dụ 1.5: Ông Nguyễn Văn Hưng làm việc ở phòng Đào tạo; Hoá đơn số 60 gửi cho khách hàng Trần Văn Hùng; Sinh viên Dương Văn Việt thuộc lớp CNTT1A

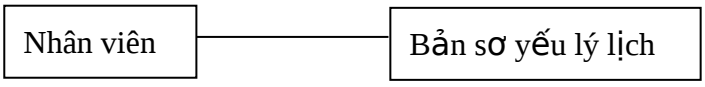
Phân loại liên kết:

+ Liên kết 1-1 (liên kết một - một): Hai kiểu thực thể A và B có mối liên kết 1-1 nếu một thực thể kiểu A tương ứng với một thực thể kiểu B và ngược lại.

Kí hiệu:



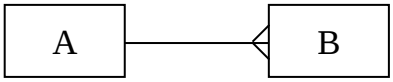
Thí dụ 1.7:



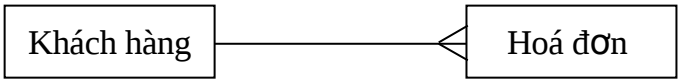
Ghi chú: Trong biểu đồ cấu trúc dữ liệu hai kiểu thực thể có mối liên kết 1-1 có thể được đồng nhất thành một kiểu thực thể.

+ Liên kết 1-n (một - nhiều): Hai kiểu thực thể A và B có mối liên kết 1-n nếu một thực thể kiểu A tương ứng với nhiều thực thực thể kiểu B và ngược lại một thực thể kiểu B tương ứng với duy nhất một thực thể kiểu A.

Kí hiệu:

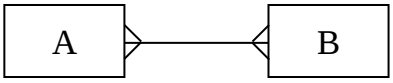


Thí dụ 1.8:

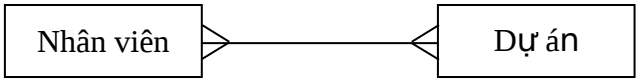


+ Liên kết n-n (nhiều - nhiều): Hai kiểu thực thể A và B có mối liên kết n - n nếu một thực thể kiểu A tương ứng với nhiều thực thể kiểu B và ngược lại.

Kí hiệu:

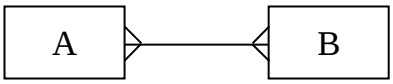


Thí dụ 1.9:

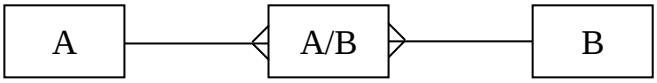


Ghi chú: Trong biểu đồ cấu trúc dữ liệu nếu tồn tại mối liên kết n-n giữa các kiểu thực thể, ta cần chuẩn hoá nó đưa về dạng liên kết một-nhiều:

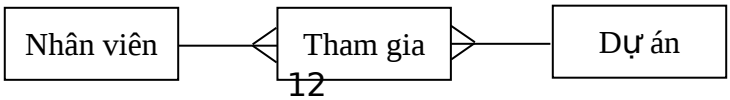
Dạng nhiều- nhiều



Đưa về dạng một - nhiều



Thí dụ 1.10:



1.6.3. Các mô hình dữ liệu

- *Mô hình dữ liệu phân cấp:* Mô hình dữ liệu phân cấp (Hierachical Data Model) - được gọi tắt là mô hình phân cấp được đưa ra vào những năm 60, trong mô hình dữ liệu này dữ liệu được tổ chức thành cấu trúc cây, trong đó các nút (node) của cây biểu diễn các bản ghi, giữa các bản ghi liên kết với nhau theo mối quan hệ cha con:

- Một cha có nhiều con.
- Một con chỉ có một cha.

Ưu điểm:

- Thể hiện dễ dàng quan hệ 1-n.
- Việc phân chia dữ liệu dễ thể hiện, đảm bảo an toàn dữ liệu
- Tính độc lập của chương trình và các dữ liệu được đảm bảo

Nhược điểm:

- Không thể hiện được mối quan hệ n-n
- Trong một hệ thống phân cấp, dữ liệu được tổ chức như trên dễ đến khó sửa đổi dữ liệu.
- Lặp lại dữ liệu, lãng phí bộ nhớ và tốn nhiều công sức tạo lập.

- *Mô hình dữ liệu mạng*:

Mô hình dữ liệu mạng (Network Data Model) được gọi tắt là mô hình mạng (Network Model) là mô hình dữ liệu được biểu diễn bởi một đồ thị có hướng. Trong mô hình mạng người ta dùng hai yếu tố là bản ghi và liên kết. Khái niệm bản ghi giống như mô hình phân cấp, liên kết là tập các con trỏ vật lý thiết lập quan hệ chủ sở hữu giữa tập bản ghi này với tập bản ghi khác. So sánh hai mô hình ta thấy bản ghi "đơn hàng" liên kết với bản ghi " số lượng" bản ghi "số lượng" cũng có liên kết với bản ghi " mặt hàng" và bản ghi "số lượng " là thành viên của hai bản ghi chủ khác nhau.

Mô hình mạng người ta đã khắc phục được việc dư thừa dữ liệu của mô hình phân cấp. Tuy vậy cấu trúc hệ thống phức tạp ngoài nội dung thông tin,

mỗi bản ghi còn có thêm thông tin nữa là địa chỉ để truy nhập tới bản ghi thành viên. Với mỗi liên kết phải có nhãn để xác định liên kết.

Ưu điểm:

- Dễ thể hiện mối liên kết n-n
- Kiểu truy cập dữ liệu mềm dẻo hơn kiểu phân cấp

Nhược điểm:

- Việc sửa đổi số liệu khó khăn.
- Với những lập trình viên, việc thiết kế CSDL khó.

- *Mô hình quan hệ:*

Mô hình cơ sở dữ liệu Quan hệ (gọi tắt là mô hình Quan hệ) do E.F Codd đề xuất năm 1971. Mô hình này bao gồm:

- Một hệ thống các ký hiệu để mô tả dữ liệu dưới dạng dòng và cột như quan hệ, bộ, thuộc tính, khóa chính, khoá ngoại, ...

- Một tập hợp các phép toán thao tác trên dữ liệu như phép toán tập hợp, phép toán quan hệ.

Vì tính chất chặt chẽ của toán học về lí thuyết tập hợp nên mô hình này đã mô tả dữ liệu một cách rõ ràng, uyển chuyển và trở thành rất thông dụng.

Ngày nay hầu hết các HQTCSDL đều tổ chức dữ liệu theo mô hình dữ liệu quan hệ.

Thí dụ 1.11:

Xét một hệ thông tin phân phối hàng, hệ này quản lý hoạt động bán hàng cho khách. Các kiểu thực thể chính của hệ thống bao gồm:

Kiểu thực thể **Khách Hàng** gồm các thuộc tính: Mã khách hàng (MaKH), Tên khách hàng (TenKH), tuổi (Tuoi), Địa chỉ khách hàng (DiaChi)

Kiểu thực thể **Hàng Hoá** gồm các thuộc tính: Mã hàng hoá (MaHang), Tên hàng hoá (TenHang), Giá (Gia), Màu sắc của mặt hàng (Mau), Đơn vị tính (DVT)

Kiểu thực thể **Bán Hàng** gồm các thuộc tính: MaKH, MaHang, số lượng (SoLuong)

Ứng với mỗi kiểu thực thể ta có một bảng dữ liệu sau:

Bảng Khách Hàng

Bảng 1.1: Khách hàng

Mã Khách	Tên Khách	Tuổi	Địa Chỉ
KH1	A	20	Hà nội
KH2	B	21	Hà tây
KH3	C	19	Thái Nguyên

Bảng Hàng Hoá

Bảng 1.2: Hàng hoá

Mã Hàng	Tên Hàng	Giá	ĐVT	Màu
MH1	Bóng rổ	500	Quả	Vàng
MH2	Bóng đá	150	Quả	Đỏ
MH3	Bóng chuyền	100	Quả	Xanh

Bảng Bán Hàng

Bảng 1.3 Hàng bán

Mã Khách	Mã Hàng	Số Lượng
KH1	MH2	50
KH1	MH3	40
KH2	MH3	60
KH2	MH1	90
KH3	MH 3	80

Một cơ sở dữ liệu theo mô hình quan hệ thực chất là một tập các bảng mà:

- Mỗi bảng gọi là một quan hệ/ kiểu thực thể/ tệp.
- Mỗi hàng gọi là một bộ/ thực thể/ bản ghi.

- Mỗi cột gọi là một thuộc tính/ trường.

1.6.4 Đánh giá các mô hình

-Về cách biểu diễn dữ liệu:

- Mô hình 1: Tập các cây quá lớn nếu dữ liệu nhiều và phức tạp
- Mô hình 2: Tập các đồ thị có hướng phức tạp
- Mô hình 3: Tập các bảng dễ quan sát vì các bảng độc lập với nhau

-Thao tác trên các mô hình:

+ Thao tác bổ sung thêm một bản ghi

- Mô hình 1, 2: Không thể bổ sung thêm các bản ghi thành viên mà không có bản ghi chủ. Ví dụ thêm mặt hàng 5 thì phải thuộc đơn hàng nào?.
- Môhình 3: Có thể bổ sung vào bảng "hàng hoá" dễ dàng

+ Xoá một bản ghi

- Mô hình 1: Phải xoá toàn bộ cây mà nó là gốc, sửa lại cây mà nó là nhánh
- Mô hình 2: Phải xoá các bản ghi thành viên của nó, sửa lại dây chuyền mà nó là thành viên
- Mô hình 3: Đơn giản ta chỉ xoá dòng có bản ghi đó và các bản ghi ở bảng có liên kết với nó

+ Tìm kiếm bản ghi

- Mô hình 1: Tìm trên các nhánh của cây
- Mô hình 2 : Tìm trên toàn bộ dây chuyền
- Mô hình 3: Tìm trên các bảng

Như vậy so sánh các thao tác ta thấy với mô hình quan hệ đơn giản thuận tiện hơn so với các mô hình trên.

BÀI TẬP VÀ CÂU HỎI CHƯƠNG 1

1. Định nghĩa cơ sở dữ liệu.
2. Nêu các thành phần của một hệ cơ sở dữ liệu.
3. Nêu kiến trúc của một hệ cơ sở dữ liệu.
4. Phân loại các hệ cơ sở dữ liệu

5. Nêu ưu điểm của việc thiết kế một hệ cơ sở dữ liệu.

6. Nêu tính độc lập dữ liệu.

7. Trình bày khái niệm về hệ quản trị cơ sở dữ liệu? Các hệ quản trị cơ sở dữ liệu hiện nay đang được sử dụng.

8. Nêu các chức năng và các thành phần của một hệ quản trị cơ sở dữ liệu.

9. Thế nào là mô hình dữ liệu ? Các mô hình dữ liệu.

Chương 2

MÔ HÌNH DỮ LIỆU QUAN HỆ

2.1. Thuộc tính

-Thuộc tính (Attribute): Thuộc tính là dữ liệu dùng để mô tả một đặc trưng của thực thể. (Các thuộc tính đơn thường ký hiệu là các chữ cái A,B,C,.... Tập thuộc tính thường ký hiệu là các chữ cái X, Y, Z…). Các thuộc tính được phân biệt qua tên gọi và phải thuộc 1 kiểu dữ liệu nhất định (kiểu dữ liệu là kiểu đơn). Tên nên đặt sát với ý nghĩa của nó, mang tính gợi nhớ và không nên quá dài.

-Miền thuộc tính: Là tập hợp các thuộc tính của thực thể, các thực thể thường có rất nhiều thuộc tính, tuy vậy để quản lý ta chỉ cần quản lý một số thuộc tính cần thiết cho thông tin về thực thể.

-Miền trị của thuộc tính (Domain): Là một tập hợp các giá trị của thuộc tính, ký hiệu là DOM(A_i) với i=1,…,n.

Ví dụ 2.1: Thuộc tính GIOITINH có miền trị là DOM(GIOITINH) = {nam, nữ}

2.2. Quan hệ

Định nghĩa: Gọi U = {A_1, A_2, A_3, ... A_n} là tập hữu hạn của các thuộc tính, mỗi thuộc tính A_i với i=1,...,n có miền giá trị tương ứng là DOM(A_i). Quan hệ R xác định trên tập thuộc tính U là tập con của tích Đề – Các.

$$R(U) \subseteq DOM(A_1) \times DOM(A_2) \times ... \times DOM(A_n)$$

Ký hiệu quan hệ R xác định trên tập thuộc tính U là R(U) hoặc R(A_1, A_2, ..., A_n).

Hay có thể viết dưới dạng sau:

R (U) =	A_1	A_2	...	A_n	
	a^1_1	a^1_2	...	a^1_n	
	a^2_1	a^2_2	...	a^2_n	m bộ
	...	...	...	...	
	a^m_1	a^m_2	...	a^m_n	

n thuộc tính

Trong đó: $A_1, A_2,...A_n$: Gọi là miền thuộc tính của quan hệ R

$DOM(A_1) = \{a^1_1, a^2_1, ... a^m_1\}$: Gọi là miền trị của thuộc tính A_1

n: Gọi là bậc của quan hệ R

m: Gọi là lực lượng của quan hệ R

Ta thấy so với một bảng thì:

- Mỗi quan hệ tương ứng với một bảng dữ liệu (là một tệp dữ liệu)
- Mỗi thuộc tính tương ứng vơi một cột dữ liệu trong bảng (là một trường)
- Mỗi bộ tương ứng với một hàng của bảng dữ liệu (là một bản ghi)

Ví dụ 2.2 :

Cho tập thuộc tính U gồm có các thuộc tính: TÊN, Giới tính và tuổi Ta có miền trị của chúng như sau:

DOM(Tên) = {Mai, Trung, Hoa, Anh}

DOM(Giới tính) = {Nam, Nữ}

DOM(Tuổi) = {15,16,17}

Từ đó, ta xây dựng được quan hệ **học sinh** là một tập con Tích Đề các của miền trị các thuộc tính trên như sau:

Bảng 2.1: Chứa thông tin về học sinh

Tên	Giới Tính	Tuổi
Mai	Nữ	15
Anh	Nam	16
Trung	Nam	15
Mai	Nữ	16
Anh	Nữ	15
Trung	Nam	17

2.3. Khoá của một quan hệ

Cho quan hệ R xác định trên tập thuộc tính U, K là một tập thuộc tính $K \subseteq U$. Gọi K là khoá của quan hệ R nếu với $\forall$ bộ $t_i, t_j \in R$; $t_i \neq t_j$ thì $t_i[K] \neq t_j[K]$ (tức

là giá trị trên K của một bộ nào đó khác giá trị trên K của mọi bộ còn lại, hay nói cách khác bộ đó là xác định duy nhất).

Xét thêm rằng nếu với $\forall$ $t_i, t_j \in R; i \neq j$ sao cho $t_i[K] = t_j[K]$ thì $t_i \equiv t_j$ thì K là khóa của quan hệ R

Ví dụ 2.3 : Xét quan hệ R có dạng sau

R	A	B	C	D
t_1=	a_1	b_1	c_1	d_1
t_2=	a_2	b_2	c_2	d_2
t_3=	a_3	b_3	c_2	d_2
t_4=	a_1	b_2	c_2	d_2

Xét thấy: K_1=U, K_2=ABC, K_3=AB, … là các khoá của quan hệ R.

Nhưng X=BC không phải là tập khoá của quan hệ R vì $t_2.X = t_4.X$ nhưng $t_2 \neq t_4$

2.4 Các phép toán của đại số quan hệ

2.4.1 Quan hệ khả hợp

Hai quan hệ R, S gọi là khả hợp nếu chúng có cùng bậc (số các thuộc tính bằng nhau) và miền trị của thuộc tính thứ i của quan hệ này bằng miền trị của thuộc tính thứ i trong quan hệ kia.

Ví dụ 2.4 :

Cho hai quan hệ $R = \{ A_1, A_2, ... A_n\}$ và $S = \{ A'_1, A'_2, ...A'_n\}$

nếu thoả mãn $DOM(A_i) = DOM(A'_i)$, với i=1,...,n thì R và S gọi là hai quan hệ khả hợp.

Chú ý: Nếu hai quan hệ có cùng bậc và tên thuộc tính thứ i trong quan hệ này khác tên với thuộc tính thứ i trong quan hệ kia nhưng chúng có cùng miền trị thì hai quan hệ này cũng là hai quan hệ khả hợp.

2.4.2 Các phép toán

(1) Phép hợp

Hợp của hai quan hệ R và S khả hợp là tập các bộ thuộc R hoặc thuộc S.

Ký hiệu phép hợp là $R \cup S$.

Biểu diễn hình thức phép hợp có dạng:

$$R \cup S = \{ t \mid t \in R \text{ hoặc } t \in S \}$$

Ví dụ 2.5: Cho hai quan hệ R và S có dạng sau:

R	A	B	C
	a_1	b_1	c_1
	a_1	b_2	c_2

S	A	B	C
	a_1	b_1	c_1
	a_2	b_2	c_2

Ta có :

$R \cup S =$	A	B	C
	a_1	b_1	c_1
	a_1	b_2	c_2
	a_2	b_2	c_2

Ví dụ 2.6: Cho hai quan hệ R và S có dạng sau:

R	A	B	C
	a_1	2	c_1
	a_1	3	c_2
	a_2	4	c_2

S	A'	B'	C
	a_1	2	c_1
	a_2	4	c_2
	a_3	5	c_3

Ta có :

$R \cup S =$	A	B	C
	a_1	2	c_1
	a_1	3	c_2
	a_2	4	c_2
	a_3	5	c_3

(2) Phép giao

Giao của hai quan hệ R và S khả hợp là tập các bộ thuộc cả quan hệ R và S. Ký hiệu phép giao là $R \cap S$

Biểu diễn hình thức phép giao có dạng:

$$R \cap S = \{ t \mid t \in R \text{ và } t \in S \}$$

Ví dụ 2.7: Cho hai quan hệ R và S có dạng sau :

R	A	B	C
	a_1	b_1	c_1
	a_1	b_2	c_2
	a_2	b_2	c_2

S	A	B	C
	a_1	b_1	c_1
	a_2	b_2	c_2

Ta có :

R ∩ S =	A	B	C
	a_1	b_1	c_1
	a_2	b_2	c_2

Ví dụ 2.8: Cho hai quan hệ R và S có dạng sau :

R	A	B	C
	a_1	2	c_1
	a_1	3	c_2
	a_2	4	c_2

S	A'	B'	C'
	a_1	2	c_1
	a_2	4	c_2
	a_3	5	c_3

Ta có :

R ∩ S =	A	B	C
	a_1	2	c_1
	a_2	4	c_2

(3) Phép trừ

Hiệu của hai quan hệ R và S khả hợp là tập các bộ thuộc R nhưng không thuộc S. Ký hiệu phép từ là R - S .

Biểu diễn hình thức phép trừ có dạng:

$$R - S = \{ t \mid t \in R \text{ và } t \notin S \}$$

Ví dụ 2.9: Cho hai quan hệ R và S có dạng sau :

R	A	B	C
	a_1	b_1	c_1
	a_1	b_2	c_2

S	A	B	C
	a_1	b_1	c_1
	a_2	b_2	c_2

a_2	b_2	c_2

Ta có :

R - S =	**A**	**B**	**C**
	a_1	b_2	c_2

Ví dụ 2.10: Cho hai quan hệ R và S có dạng sau :

R	**A**	**B**	**C**
	a_1	2	c_1
	a_2	4	c_2

S	**A'**	**B'**	**C'**
	a_1	2	c_1
	a_2	4	c_2
	a_3	5	c_3
	a_1	3	c_2

Ta có :

S - R =	**A'**	**B'**	**C'**
	a_3	5	c_3
	a_1	3	c_2

Chú ý: Ta thấy R - S = R - (R ∩ S)

(4) Phép chiếu

Phép chiếu của quan hệ R trên tập thuộc tính X (X⊆ U), ký hiệu $\Pi_X(R)$ là một tập các bộ, được xây dung bằng cách loại bỏ đi từ các bộ t trong quan hệ R những thuộc tính không nằm trong X.

Để thuận tiện cho việc biểu diễn hình thức phép chiếu, quy ước một số ký hiệu như sau: Gọi t là một bộ thuộc R, A là một thuộc tính (A⊆ U), t[A] là giá trị của bộ t tại thuộc tính A. Giả sử X⊆ U với X={$B_1, B_2,\ldots, B_m$}khi đó t[X] =(t[B_1], t[B_2], …, t[B_m]). Vậy ta có Π_X (R) = { t [X] | t ∈ R}

Ví dụ 2.11: Cho quan hệ R và tập thuộc tính X (với X=AB)

R	**A**	**B**	**C**
	a_1	b_1	c_1
	a_2	b_2	c_2
	a_1	b_2	c_2

Vậy phép chiếu trên tập thuộc tính X của quan hệ R có dạng sau:

Π_X (R) =	**A**	**B**
	a_1	b_1
	a_2	b_2
	a_1	b_2

(5) Phép chọn

-Phép chọn là phép toán lọc ra trong một quan hệ một tập con các bộ thoả mãn các điều kiện của biểu thức chọn F.

-Biểu thức chọn F: là một tổ hợp Boolean của các toán hạng, mỗi toán hạng là một phép so sánh đơn giản giữa hai biến là hai thuộc tính hoặc giữa một biến là một thuộc tính và một hằng, cho giá trị đúng hoặc sai đối với mỗi bộ dữ liệu.

Các phép so sánh: >, >, =, <, < = , < >

Các phép logic là: ∧ (và), ∨ (hoặc), ⌐ (phủ định)

-Biểu diễn hình thức của phép chọn:

$$\sigma_F(R) = \{t / t \in R \text{ và } t(F)=True\}$$

Ví dụ 2.12: Cho quan hệ R có dạng sau:

R	**A**	**B**	**C**
	a_1	b_1	c_1
	a_2	b_2	c_2
	a_3	b_3	c_3

Với biểu thức chọn F: B = "b_1", ta có

σ_F (R) =	**A**	**B**	**C**
	a_1	b_1	c_1

Với biểu thức chọn F: B <> "b_1", ta có

σ_F (R) =	**A**	**B**	**C**
	a_2	b_2	c_2
	a_3	b_3	c_3

Với biểu thức chọn F: (A = "b_2") ^ (C = "c_2")

σ_F (R) =	**A**	**B**	**C**
	a_2	b_2	c_2

(6) Phép tích Đề - các

Cho R là một quan hệ xác định trên tập thuộc tính (A_1, A_2, ... A_n) và quan hệ S xác định trên tập thuộc tính (B_1, B_2,... B_m). Tích Đề - các của R và S là một quan hệ gồm (n+m) thuộc tính và mỗi bộ của quan hệ kết quả có dạng n thành phần đầu là một bộ thuộc R và m thành phần sau là một bộ thuộc S

R x S = { t | t có dạng (a_1, a_2, ... a_n, b_1, b_2, ..., b_m) trong đó {a_1, a_2,..., a_n} ∈ R; {b_1, b_2,..., b_m} ∈ S}

Ví dụ 2.13: Cho hai quan hệ R và S có dạng sau:

R	**A**	**B**	**C**
	a_1	b_1	c_1
	a_1	b_2	c_2

S	**D**	**E**	**F**
	d	e	f
	d'	e'	f'

Ta có kết quả của phép tích Đề các:

R x S =	**A**	**B**	**C**	**D**	**E**	**F**

a_1	b_1	c_1	d	e	f
a_1	b_1	c_1	d'	e'	f'
a_1	b_2	c_2	d	e	f
a_1	b_2	c_2	d'	e'	f'

Ví dụ 2.14: Cho hai quan hệ R và S có dạng sau:

R	A	B
	α	1
	β	2

S	B	C	D
	α	4	d_1
	β	5	d_2
	β	3	d_3
	γ	7	d_4

Ta có phép Tích Đề Các của R và S là:

RxS=	A	R.B	S.B	C	D
	α	1	α	4	d_1
	α	1	β	5	d_2
	α	1	β	3	d_3
	α	1	γ	7	d_4
	β	2	α	4	d_1
	β	2	β	5	d_2
	β	2	β	3	d_3
	β	2	γ	7	d_4

(7) Phép kết nối

Phép kết nối là phép toán với hai quan hệ dựa trên các điều kiện để liên kết với nhau:

-Gọi θ là một trong các phép so sánh: >; <; <=; >=; <>; =

-Biểu thức kết nối có dạng: F = A θ B

Từ đó ta có phép kết nối được xác định như sau:

$$R \underset{F}{\bowtie} S = \{ t(u,v) / u \in R; v \in S \text{ và thoả mãn biểu thức chọn } F \}$$

Ví dụ 2.15: Cho hai quan hệ R và S có dạng sau:

R	**A**	**B**	**C**
	a_1	b_1	1
	a_2	b_2	2
	a_3	b_3	3

S	**E**	D
	1	d_1
	2	d_2

Gọi F là biểu thức kết nối hai quan hệ R và S (F = C ≥ E)

Ta có $R \underset{F}{\bowtie} S$ =

(A	**B**	**C**	**E**	**D)**
a_1	b_1	1	1	d_1
a_2	b_2	2	1	d_1
a_2	b_2	2	2	d_2
a_3	b_3	3	1	d_1
a_3	b_3	3	2	d_2

Chú ý:

-Hai quan hệ muốn kết nối được thì miền thuộc tính kết nối A của quan hệ R phải so sánh được với miền thuộc tính B của quan hệ S

-Nếu T' = R x S mà T = $R \bowtie S$ thì $T \subseteq T'$. Như vậy phép kết nối có thể coi là phép chọn của phép tích Đề các.

-Nếu phép kết nối θ là " = " thì gọi là kết nối bằng. Nếu kết nối bằng qua hai thuộc tính cùng tên và một trong hai thuộc tính được loại bỏ thì gọi là kết nối tự nhiên, ký hiệu "*"

Ví dụ 2.16: Cho hai quan hệ R và S có dạng sau:

R	**A**	**B**	**C**
	a_1	b_1	1
	a_2	b_2	2
	a_3	b_3	3

S	C	D
	1	d_1
	2	d_2

Gọi F là biểu thức kết nối tự nhiên giữa R và S, (F có dạng: R.C=S.C)

Vậy phép kết nối tự nhiên giữa R và S là:

R*S=	A	B	C	D
	a_1	b_1	1	d_1
	a_2	b_2	2	d_2

(8) Phép chia

Cho R là quan hệ xác định trên tập thuộc tính U, với $U=\{A_1,A_2,...,A_n\}$ và quan hệ S xác định trên tập thuộc tính V với $V=\{ B_1,B_2,...,B_m \}$ sao cho $n > m$, $S \neq \emptyset$ và $U \subset V$. Phép chia $R \div S$ là một quan hệ P(M) có dạng sau:

$P(M)=R \div S=\{t.M$ với $t \in R, (t.M) \times S \subseteq R$ và $M=U-V\}$

Ví dụ 2.17: Cho hai quan hệ R và S có dạng sau:

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

Ta có phép chia của R và S là:

R÷ S=	A	B
	a	b
	c	d

Nhận xét:

-Nếu P x S= R thì phép chia không dư

-Nếu P x S $\subset$ R thì phép chia có dư

(9) Một số hàm tiện ích:

1. Hàm SUM(R,A) cho tổng cá giá trị số trong cột A của R

SUM(R,A) = Σ {t.A | t$\in$R}

2. Hàm AVG(R,A)

3. Hàm MAX(R,A)

4. Hàm (MIN(R,A)

2.4.3 Một số ví dụ

Cho cơ sở dữ liệu cung cấp hàng gồm các bảng dữ liệu sau:

Bảng Công Ty (CONGTY) gồm các thuộc tính: Mã công ty (MaCongTy), Tên công ty (TenCongTy), Ngân sách (NganSach), Địa chỉ (DiaChi).

Bảng Hàng Hoá (HANGHOA) gồm các thuộc tính: Mã hàng (MaHang), Tên hàng (TenHang), Mầu sắc (Mau), Đơn vị tính (DonViTinh).

Bảng Cung Cấp hàng (CUNGCAP) gồm các thuộc tính: MaCongTy, MaHang, Số lượng (SoLuong), Đơn giá (DonGia).

Hãy viết biểu thức đại số quan hệ để thực hiện các câu hỏi sau:

- Cho biết danh sách các mặt hàng màu đỏ

$$\sigma_{Mau = \text{“ Đỏ”}}(HANGHOA)$$

- Cho biết mã các công ty cung cấp mặt hàng H1

$$\Pi_{MaCongTy}(\sigma_{MaHang = \text{“H1”}}(CUNGCAP))$$

- Cho biết tên các công ty cung cấp mặt hàng H1

$$\Pi_{TenCongTy}((\sigma_{MaHang = \text{”H1”}}(CUNGCAP)) *_{MaCongTy} CONGTY)$$

- Cho biết những công ty cung cấp cả hai mặt hàng H1 và H2

$$\Pi_{TenCongTy}[(CONGTY *_{MaCongTy} (\sigma_{MaHang=\text{”H1”}}(CUNGCAP)) \cap \cap ((CONGTY *_{MaCongTy} (\sigma_{MaHang=\text{”H2”}}(CUNGCAP)]$$

Câu hỏi tối ưu hơn:

$\Pi_{TenCongTy}$ {CONGTY * [$\Pi_{MaCongTy}$ ($\sigma_{MaHang="H1"}$(CUNGCAP)) $\cap$

MaCongTy $\cap$ $\Pi_{MaCongTy}$ ($\sigma_{MaHang="H2"}$(CUNGCAP))]}

- Cho biết tên các công ty cung cấp ít nhất một mặt hàng màu đỏ

  $\Pi_{TenCongTy}$(CONGTY * CUNGCAP * ($\sigma_{màu="Đỏ"}$(HANGHOA))

  MaCongTy MaCongTy

- Cho biết tên những công ty cung cấp tất cả các mặt hàng

$\Pi_{TenCongTy}$ { CONGTY * [$\Pi_{MaCongTy, ,MaHang}$ CUNGCAP) $\div \Pi_{MaHang}$(HANGHOA)]

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 2

1. Định nghĩa quan hệ. Cho ví dụ minh hoạ.
2. Định nghĩa khoá của một quan hệ. Cho ví dụ minh hoạ.
3. Trình bày các phép toán của đại số quan hệ. Cho các ví dụ minh hoạ.
4. Cho các quan hệ R, S và P có dạng sau:

R	A	B	C
	a_1	b_1	c_1
	a_2	b_2	c_2
	a_3	b_3	c_3

S	D	E	F
	d_1	e_1	f_1
	d_2	e_2	f_2
	d_3	e_3	f_3

P	A	B	D
	a_1	b_1	d_1
	a_2	b_2	d_2
	a_1	b_3	d_3
	a_2	b_1	d_2
	a_2	b_2	d_1
	a_3	b_3	d_3
	a_2	b_3	d_3

Hãy tính giá trị của các biểu thức đại số quan hệ sau:

a) $\Pi_{(AB)}$(P)

b) R*$\Pi_{(AB)}$(P)

c) σ_F(R*$\Pi_{(AB)}$(P)) với F có dạng D='d_1'

a. d) R * S * T

b. e) Π_{CE}($\sigma_{C='c2' \wedge F='f2'}$(R x S))

5. Cho các quan hệ R, S và P có dạng sau:

R	A	B	C

S	D	E

P	A	B	D	H

a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3

d_1	e_1
d_2	e_2
d_3	e_3

a_1	b_1	d_1	h_1
a_2	b_2	d_1	h_2
a_2	b_2	d_2	h_3
a_2	b_2	d_3	h_4

Hãy tính kết quả của các biểu thức đại số quan hệ sau:

a) S x P

1. b) $\sigma_{S.D=P.D}$ (S x P)
2. c) R*($\sigma_{S.D=P.D}$ (S x P))
3. d) R*S*P

6. Cho quan hệ R, S và P có dạng sau:

R	**A**	**B**	**C**
	a_1	b_1	c_1
	a_2	b_2	c_2

S	**D**	**E**
	1	e_1
	2	e_2
	3	e_3

P	**B**	**D**
	b_1	1
	b_2	2
	b_2	2

Hãy tính kết quả của các phép toán đại số quan hệ sau:

- a) S*P
- b) R*S*P
- c) $\sigma_{D=2}$ (S* P)
- d) R * ($\sigma_{D=2}$ (S* P))

7. Cho các quan hệ R, S và Q có dạng sau:

R

A	**B**	**C**
a_2	b_2	c_1
a_1	b_2	c_1
a_2	b_3	c_2

S

B	**C**	**D**
b_2	C_1	d_1
b_2	C_1	d_2
b_1	C_2	d_3

Q

D	**E**	**F**
d_1	e_1	f_1
d_2	e_1	f_2
d_2	e_2	f_3

Hãy tính kết quả của các phép toán đại số quan hệ sau:

- $\delta_{A=a2}$(R*S)
- Π_{ABC}(R*S*Q)
- Π_{BC}(R) $\cup$ Π_{BC}(S)

e. R x S

f. R * $\delta_{B = b1}$(S)

g. Π_B($\delta_{B=b2}$(R*S))

- $\Pi_{BC}(R) \cap \Pi_{BC}(S)$ h. $\delta_{D=d1}(\Pi_{BCD}(S*Q))$

8. Cho các quan hệ LOPHOC, SINHVIEN, DIEMTHI và MONHOC lần lượt như sau:

+Bảng 2.2: Quan hệ LOPHOC

MALOP	**TENLOP**
L01	ĐHCQK2C
L02	ĐHCQK2C
L03	ĐHCQK2B
L04	ĐHCQK2B

+ Bảng 2.3 Quan hệ SINHVIEN

MASV	**HOTEN**	**GIOITINH**	**DIACHI**	**MALOP**
CQK21001	Lê Hồng Vân	1	Thái Nguyên	L01
CQK21002	Nguyễn Văn Hà	1	Bắc giang	L01
CQK21003	Hoàng Thị Gấm	0	Hà Nội	L02
CQK21004	Lê Thị Thao	0	Thái Nguyên	L02

+Bảng 2.4: Quan hệ MONHOC

MAMH	**TENMH**	SOTINCHI
M01	Pascal	3
M02	Ngôn ngữ C	3
M03	Cơ sở dữ liệu	2

+ Bảng 2.5: Quan hệ DIEMTHI

MASV	MAMH	KYHOC	DIEMLAN1	DIEMLAN2
CQK21001	M01	1	4	7
CQK21001	M02	3	8	
CQK21001	M03	3	4	6
CQK21003	M01	1	8	
CQK21004	M02	3	2	6

Hãy viết các biểu thức đại số quan hệ để thực hiện các yêu cầu sau:

c. Cho biết mã sinh viên, họ tên, giới tính của các sinh viên có địa chỉ tại Thái Nguyên?

d. Cho biết mã sinh viên, họ tên, địa chỉ của các sinh viên có giới tính bằng 1?

e. Cho biết mã và tên của các môn học có số tín chỉ bằng3?

f. Cho biết mã của các môn học có số tín chỉ<=2?

g. Cho biết mã sinh viên, họ tên, điểm thi lần 1 của các sinh viên đã học môn học có mã là ‘M01’ trong học kỳ 1?

h. Cho biết mã sinh viên, họ tên, điểm thi lần 1 của các sinh viên đã học môn học có tên môn là ‘cơ sở dữ liệu’?

i. Cho biết mã của các sinh viên đã tích luỹ được tất cả các môn học có ?

j. Cho biết mã của các môn học chưa có sinh viên nào đăng ký học?

k. Cho biết mã, họ tên, mã lớp của các sinh viên phải thi lại môn học có mã môn là ‘M01’?

l. Cho biết mã, họ tên, mã lớp của các sinh viên phải học lại môn học có tên môn là ‘Ngôn ngữ C’?

m. Cho biết mã, họ tên, mã lớp của các sinh viên phải thi lại môn học có tên môn là ‘Pascal’?

n. Cho biết mã sinh viên, mã môn học, điểm thi lần 1 của các sinh viên học trong học kỳ 3?

Chương 3

LÝ THUYẾT THẾT KẾ CƠ SỞ DỮ LIỆU

3.1. Giới thiệu

3.1.1. Vấn đề thiết kế cơ sở dữ liệu

Một cơ sở dữ liệu quan hệ gồm tập các quan hệ. Muốn xây dựng một cơ sở dữ liệu quan hệ cần xác định trong cơ sở dữ liệu đó có những quan hệ gì, mỗi quan hệ có những thuộc tính nào, sự liên kết giữa các quan hệ như thế nào?...

Từ cơ sở phân tích chúng ta mới xây dựng nên sơ đồ thực thể liên kết, xác định các quan hệ và các liên kết cần thiết, chỉnh sửa chuẩn hoá các quan hệ trong hệ thống cơ sở dữ liệu

Bước cuối cùng là nhập dữ liệu theo dõi bảo trì cập nhật, hoàn thiện các quan hệ, các liên kết... trong hệ thống theo yêu cầu của người dùng

3.1.2 Bài toán ví dụ

Giả sử một cửa hàng bán buôn/lẻ các nhân viên mở sổ theo dõi việc bán hàng hàng ngày là một bảng (quan hệ) như sau:

Bảng 3.1: Sổ theo dõi việc bán hàng

Ngày	Số HĐ	Mã KH	Tên KH	Địa chỉ	Mã hàng	Tên hàng	Đơn giá	SL	Thành tiền
15/01/09	HD01	1	Anh	TN	A1	Xe đạp	800	3	2.400
15/01/09	HD01	1	Anh	TN	A2	Xe máy	1.200	2	2.400
17/01/09	HD02	2	Hùng	BG	A2	Xe máy	1.200	1	1.200
18/01/09	HD03	3	Hương	TB	A1	Xe đạp	800	2	1.600
18/01/09	HD03	3	Hương	TB	A2	Xe máy	1.200	4	4.800

Nhận xét: Quan hệ trên được thiết kế chưa tối ưu vì tồn tại một số dị thường về dữ liệu, cụ thể như:

-Dư thừa dữ liệu (Redundancy): Thông tin về khách hàng và hàng hoá bị lặp lại nhiều lần. Nếu khách hàng có mã 1 mua 15 mặt hàng thì thông tin về khách hàng này bị lặp lại 15 lần, tương tự đối với mặt hàng nếu mặt hàng có mã A1, nếu có 2000 khách hàng mua thì thông tin về mặt hàng đó cũng lặp lại 2000 lần

-Không nhất quán (Inconsistency): Là hệ quả của dư thừa dữ liệu. Giả sử sửa bản ghi thứ nhất, tên khách hàng được chữa thành An thì dữ liệu này lại không nhất quán với bản ghi thứ 2 (vẫn có tên là Anh).

-Dị thường khi thêm bộ (Insertion anomalies): Nếu muốn thêm thông tin về một mặt hàng mới nhập (chưa bán cho bất kỳ khách nào) vào quan hệ thì không được vì khoá chính của quan hệ trên gồm 2 thuộc tính Số hoá đơn, Mã hàng.

-Dị thường khi xoá bộ (Deletion anomalies): Giả sử muốn xoá thông tin về mặt hàng có mã là A1 thì ta phải rò tất cả các dòng trong bảng có liên quan đến mặt hàng này để xoá, ngược lại thông tin về mặt hàng đó vẫn tồn tại.

Qua phân tích trên, chúng ta nên tìm cách tách quan hệ trên thành các quan hệ nhỏ hơn.

Vậy ta tách quan hệ trên thành 4 quan hệ sau:

Bảng 3.2: Chứa thông tin về hàng hoá

Mã hàng	Tên hàng	Đơn giá
A1	Xe đạp	800000
A2	Xe máy	12000000

Bảng 3.3 Chứa thông tin về khách hàng

Mã khách	Tên khách	Địa chỉ
1	Anh	TN
2	Hùng	BG
3	Hương	TB

Bảng 3.4: Chứa thông tin về hoá đơn bán hàng

Số hoá đơn	Ngày	Mã khách
HD01	15/01/09	1
HD02	17/01/09	2
HD03	18/01/09	2

Bảng 3.5 : Chứa thông tin về chi tiết hoá đơn bán hàng

Số hoá đơn	Mã hàng	Số lượng
HD01	A1	3
HD01	A2	2
HD02	A2	1
HD03	A1	2
HD03	A2	4

Với cách tổ chức này ta thấy:

- Cơ sở dữ liệu gồm 4 bảng.
- Trong mỗi quan hệ không có sự dư thừa dữ liệu.

3.1.3. Kết luận

Cách tổ chức dữ liệu thứ hai (tách thành 4 quan hệ) tốt hơn thuận lợi hơn cho việc áp dụng máy tính vào xử lý, khắc phục những dị thường về dữ liệu khi cập nhật, sửa chữa dữ liệu như:

-Dư thừa dữ liệu

-Không nhất quán về dữ liệu....

Cơ sở để tách các quan hệ dựa trên sự phụ thuộc giữa các thuộc tính (gọi là phụ thuộc hàm) nghĩa là từ thuộc tính này có thể suy ra thuộc tính kia:

Ví dụ 3.1: Từ mã hàng ta có thể suy ra tên hàng

Mã hàng là "A1" thì "tên hàng" phải là xe đạp

Mã hàng là "A2" thì "tên hàng" phải là xe máy

Việc tách các quan hệ thành các quan hệ con ta gọi là phép chuẩn hoá

3.2. Sơ đồ quan hệ

3.2.1 Phụ thuộc hàm (*Functional Dependencies*)

Cho quan hệ R(U); X, Y là 2 tập thuộc tính ($X,Y \subseteq U$) và một PTH f: $X \rightarrow Y$ Ta nói quan hệ R thoả PTH f và viết R(f) nếu với mọi 2 bộ bất kỳ $t_i, t_j \in R$ giống nhau trên X thì chúng cũng giống nhau trên Y. Hay ta viết:

$R(X \rightarrow Y) \Leftrightarrow (\forall u,v \in R)$: $u.X=v.X \Rightarrow u.Y = v.Y$.

Trong đó u, v là hai bộ bất kỳ thuộc quan hệ R.

Nếu $f:X \rightarrow Y$ là một phụ thuộc hàm xác định trên R(U) thì ta nói rằng tập thuộc tính Y phụ thuộc hàm vào tập thuộc tính X, (hay tập thuộc tính X xác định hàm tập thuộc tính Y.

Nếu Y không phụ thuộc hàm vào X ta có thể viết $X! \rightarrow Y$

Ví dụ 3.2: Cho bảng (quan hệ) SINH VIÊN sau:

Bảng 3.6: Chứa thông tin về sinh viên

Mã SinhViên	HọTên	GiớiTính	Ngày Sinh	Quê Quán
SV01	Lan	Nữ	14/07/86	Hà Nội
SV02	Mai	Nữ	02/05/87	Thái Nguyên
SV04	Anh	Nam	12/05/85	Nam Định
SV05	Hoa	Nữ	20/01/86	Hà Nội

- Ký hiệu một phụ thuộc hàm là f. Ký hiệu một tập phụ thuộc hàm là F. Ví dụ trong bảng 3.6 ta có các phụ thuộc hàm sau:

-Tên sinh viên phụ thuộc vào mã sinh viên (MãSinhViên $\rightarrow$HọTên)

- Quê quán phụ thuộc hàm vào mã sinh viên (MãSinhViên $\rightarrow$QuêQuán)

- Giới tính phụ thuộc hàm vào mã sinh viên (MãSinhViên $\rightarrow$GiớiTính)

- Ngày Sinh phụ thuộc hàm vào mã sinh viên (MãSinhViên $\rightarrow$NgàySinh)

Vậy ta có tập phụ thuộc hàm:

F={MãSinhViên$\rightarrow$HọTên; MãSinhViên$\rightarrow$QuêQuán;
MãSinhViên$\rightarrow$GiớiTính; Mã Sinh Viên $\rightarrow$Ngày Sinh}

Ghi chú:

- Phụ thuộc hàm là cơ sở cho việc chuẩn hoá lược đồ quan hệ.
- Phụ thuộc hàm là những ràng buộc dữ liệu được suy ra từ ý nghĩa và các mối liên quan giữa các thuộc tính.

3.2.2. Lược đồ quan hệ (sơ đồ quan hệ)

Lược đồ quan hệ r là một cặp gồm hai thành phần (U,F) trong đó U là tập hữu hạn các thuộc tính, F là tập các phụ thuộc hàm xác định trên U.

Ký hiệu là: r(U,F)

Ví dụ 3.3: Cho lược đồ quan hệ r(U,F), với U = {A,B,C,D,E}

và F = {A→BC, B →D, AD →E}

Ghi chú:

- Thể hiện của lược đồ quan hệ là quan hệ (bảng dữ liệu)
- Quan hệ luôn xác định trên một lược đồ quan hệ
- Thể hiện của một lược đồ quan hệ có thể khác nhau tại mỗi thời điểm
- Một lược đồ quan hệ có thể tương đương với một tập lược đồ quan hệ nhỏ hơn nhưng có cấu trúc tốt hơn trong việc áp dụng các thao tác dữ liệu.

3.3 Hệ tiên đề cho tập phụ thuộc hàm

3.3.1. Đặt vấn đề

Ta thấy với các bài toán quản lý khác nhau thì ta phải làm việc với các loại dữ liệu khác nhau, như vậy sẽ không có một phương pháp tổng quát cho mọi loại dữ liệu. Hay nói cách khác sẽ không có một lý thuyết mà có thể áp dụng cho mọi cơ sở dữ liệu. Điều đó dẫn đến bài toán tổ chức cơ sở dữ liệu chỉ là một bài toán thủ công không thể áp dụng các công cụ toán học và quá trình xử lý trên máy tính được.

Từ đó người ta tìm một giải pháp sao cho có thể khái quát hoá các cơ sở dữ liệu bằng mô hình toán học và có thể áp dụng được các công cụ toán học. Trong cơ sở dữ liệu khái quát đó, các thuật toán xử lý không phụ thuộc vào ý nghĩa của các thuộc tính cụ thể mà chỉ phụ thuộc vào các ràng buộc đã xác định qua tập thuộc tính và tập phụ thuộc hàm.

Ví dụ 3.4: Ta có lược đồ quan hệ r(U, F) với U là tập hữu hạn các thuộc tính U = {A, B, C}, F là tập các PTH F = {A →BC}

Ta có thể coi A là số báo danh; B là tên; C là tuổi

Cũng có thể coi A là tên hàng; B đơn giá; C là khối lượng

Dù tên cụ thể của A, B, C là gì thì tập U và F cũng vẫn đúng không phụ thuộc vào tên cụ thể của các thuộc tính.

Từ vấn đề trên Armstrong đã nghiên cứu và đưa ra mô hình bài toán khái quát với các tiên đề áp dụng cho mọi cơ sở dữ liệu

3.3.2. Hệ tiên đề Armstrong

Cho lược đồ quan hệ r(U,F) với U={A_1, A_2, …, A_n} là tập các thuộc tính và F là tập PTH. Giả sử X, Y, Z ∈ U, ta có hệ tiên đề Armstrong sau:

1. Tiên đề phản xạ

Nếu Y ⊆ X thì X →Y (Mọi tập con của X thì đều phụ thuộc hàm vào X)

2. Tiên đề tăng trưởng

Nếu X →Y và Z ∈ U thì XZ →YZ

3. Tiên đề bắc cầu

Nếu X →Y và Y →Z thì X →Z

Từ các tiên đề ta có các tính chất trên sơ đồ quan hệ r(U,F); X,Y,Z,W ⊆ U

1. Tính phản xạ chặt

Nếu X →X

2. Tính tựa bắc cầu:

Nếu X →Y và YZ →W thì XZ →W

3. Tính mở rộng vế trái và thu hẹp vế phải

Nếu X →Y thì XZ →Y\W

4. Tính cộng đầy đủ

Nếu X →Y và Z →W thì XZ →YW

5. Tính mở rộng vế trái

Nếu X →Y thì XZ →Y

6. Tính cộng ở vế phải (Luật hợp)

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

7. Tính bộ phận ở vế phải (Luật tách)

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

8. Tính tích luỹ

Nếu X $\rightarrow$Y Z, Z $\rightarrow$W thì X $\rightarrow$YZW

Khi giải quyết các bài toán ta có thể áp dụng các tiên đề Amstrong và các tính chất trên.

3.3.3. Bài toán áp dụng

Cho lược đồ quan hệ R(U,F) với

U={ A, B, C}

F = { AB $\rightarrow$C; C $\rightarrow$A}. Chứng minh BC $\rightarrow$ABC

Giải:

Từ C $\rightarrow$A (gt)

Theo tiên đề tăng trưởng thêm vào hai vế B ta có: BC $\rightarrow$AB (1)

Từ AB $\rightarrow$C (gt)

Thêm AB vào hai vế ta có: AB $\rightarrow$ABC (2)

Từ (1) và (2) theo tiên đề bắc cầu ta có:

BC $\rightarrow$ABC đó là điều phải chứng minh

3.3.4. Kiểm tra tính đúng đắn của hệ tiên đề Amstrong

Giả sử có bảng DS cán bộ: MãCB, TênCB, MãLương, BậcLương

Trong đó: MãCB $\rightarrow$TênCB, MãLương, BậcLương

MãLương $\rightarrow$BậcLương

Mô hình hoá bằng các thuộc tính sau:

Cho lược đồ quan hệ R(U,F). Trong đó

U = {A,B,C,D}

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

a). Kiểm tra tiên đề 1

Nếu đặt X = AB rõ ràng A $\subseteq$ AB

Với hai bộ bất kỳ t_1, t_2 ta đều có

Nếu $t_1.[AB] = t_2.[AB]$

Thì $t_1.[A] = t_2.[A]$

Hiển nhiên ta thấy $AB \rightarrow A$

b) Kiểm tra tiên đề 2

Đặt X = AB và XC = ABC

Đặt Y = D và YC = DC

Với hai bộ bất kỳ t_1, t_2 ta thấy

Nếu $t_1.[ABC]= t_2.[ABC]$

Thì $t_1.[DC] = t_2.[DC]$

Như vậy tiên đề thứ hai là đúng đắn

c). Kiểm tra tiên đề 3

Theo tiên đề 3 ta thấy $A \rightarrow C$; $C \rightarrow D$ thì có thể suy ra $A \rightarrow D$

Với hai bộ bất kỳ t_1, t_2

Nếu $t_1.[A] = t_2.[A]$

Thì $t_1.[D] = t_2.[D]$

Vậy tiên đề này hoàn toàn đúng

3.4 Bao đóng của tập phụ thuộc hàm

Cho lược đồ quan hệ r(U,F), Trong đó U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm. X,Y là các tập thuộc tính ($X,Y \subseteq U$).

Nói rằng phụ thuộc hàm $X \rightarrow Y$ được suy dẫn logíc từ F nếu quan hệ R xác định trên r(U,F) thoả mãn tất cả các phụ thuộc hàm của F thì cũng thoả mãn phụ thuộc hàm $X \rightarrow Y$.

Bao đóng của tập phụ thuộc hàm F (kí hiệu là F^+) là tập tất cả các phụ thuộc hàm được suy dẫn logíc là F

$F^+ = \{f:X \rightarrow Y | X,Y \in U \ F \vdash f\}$

Nếu có $F = F^+$ thì F là họ đầy đủ của các phụ thuộc hàm.

Ví dụ 3.5: Cho lược đồ quan hệ r(U,F), với U = { A,C,B}và F = { $A \rightarrow B$, $B \rightarrow C$} ta có thể suy ra $A \rightarrow C$. Rõ ràng phụ thuộc hàm $A \rightarrow C$ được suy diễn ra từ F.

Ta có F^+ = { A $\rightarrow$B, B $\rightarrow$C, A $\rightarrow$C}

3.5. Phép tách một quan hệ

3.5.1. Định nghĩa

Cho lược đồ quan hệ r xác định trên tập thuộc tính U và F là tập các phụ thuộc hàm. Phép tách lược đồ quan hệ r(U,F) là việc thay thế lược đồ quan hệ r(U,F) bằng các tập lược đồ $r_1(U_1)$, $r_2(U_2)$, ..., $r_m(U_m)$, sao cho U = $U_1 \cup U_2 \cup ... \cup U_m$ Trong đó $U_i \subseteq U$ và $r_i(U_i) = \Pi_{Ui}(R)$ với i=1,..., m

Ký hiệu phép tách của r(U,F) là ρ. Vậy ρ = {$r_1(U_1)$, $r_2(U_2)$, ..., $r_m(U_m)$}

Nói rằng ρ là phép tách – kết nối không mất mát thông tin đối với F nếu với mỗi quan hệ R xác định trên r(U,F) thì R = $\Pi_{U1}(R)$* $\Pi_{U2}(R)$* $\Pi_{U3}(R)$*...* $\Pi_{Um}(R)$.

Định lý 3.1: Cho lược đồ quan hệ r(U,F) nếu ρ= {$R_1(U_1)$, $R_2(U_2)$} là một phép tách của r thì ρ là phép tách không mất mát thông tin đối với F khi và chỉ khi:

$$U_1 \cap U_2 \rightarrow U_1 \setminus U_2$$

Hoặc $U_1 \cap U_2 \rightarrow U_2 \setminus U_1$

Chứng minh:

Ta phải chứng minh hai vấn đề:

-$U_1 \cap U_2 \neq \phi$

-Và R(U) = R(U_1) * R(U_2) *theo tính chất trên*

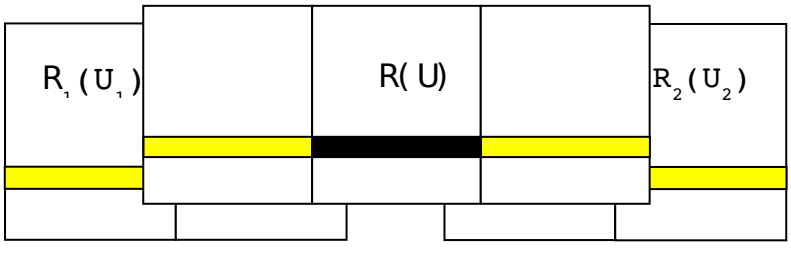
♦ Giả sử xét bảng dữ liệu sau:

Tên	Phách	Điểm
Nam	01	7
Bắc	02	6
Nam	03	4

U = {tên, phách, điểm}

Nếu tách U_1 = {tên}; U_2 = {phách, điểm} Thì $U_1 \cap U_2 = \phi$

Rõ ràng ta thấy dữ liệu không còn chính xác. Minh hoạ bằng bảng sau ta thấy:



♦ Giả sử chọn bộ t nào đó thuộc R. Khi tách thành R_1, R_2 ta được t_1, t_2

Ta thấy $t = t_1 * t_2$ hay $R \subseteq R_1 * R_2$

Mặt khác $\forall t_1 \in R_1$; và $\forall t_2 \in R_2$ ta có:

$$t_1[U_1 \cap U_2] = t_2[U_1 \cap U_2]$$

Theo tính chất phép toán kết nối tự nhiên ta có:

$$t_1 * t_2 = t \quad \text{Hay } R_1 * R_2 \subseteq R$$

Như vậy ta có $R_1 * R_2 = R$

Định lý được chứng minh

Nhận xét: Nếu ta tách một lần được hai quan hệ , tách hai lần được 3 quan hệ vậy muốn tách *m* quan hệ phải tách (*m-1*) lần.

3.5.3. Kiểm tra phép tách không mất mát thông tin

Cho lược đồ quan hệ r(U,F), Trong đó, tập các thuộc tính U ={A_1,A_2, …, A_n} và tập các phụ thuộc hàm F; phép tách ρ. Hãy kiểm tra phép tách ρ:

$$\rho = (r_1, r_2, ...r_m) \text{ có mất mát thông tin không?}$$

Thuật toán 1:

Bước 1: Lập một bảng gồm có n cột, m hàng. Cột thứ j ứng với thuộc tính A_j hàng thứ i ứng với lược đồ r_i. Tại hàng i cột j điền ký hiệu a_j nếu $A_j \in r_i$; ngược lại điền ký hiệu b_{ij}

Bước 2: Đồng nhất giá trị trên bảng

Áp dụng quy trình thay thế đuổi trên bảng:

Xét các phụ thuộc hàm từ F : Giả sử xét PTH $X \rightarrow Y \in F$. Ta xét các hàng có giá trị bằng nhau tại thuộc tính X thì làm bằng giá trị tại thuộc tính Y giữa các hàng đó.

Chú ý: Khi làm bằng giá trị trên Y, nếu một trong các giá trị là a_j thì ưu tiên làm bằng về ký hiệu a_j ngược lại làm bằng chúng bằng một trong các ký hiệu b_{ij}.

Tiếp tục áp dụng các phụ thuộc hàm có trong F (kể cả việc lập lại các phụ thuộc hàm đã được áp dụng) cho tới khi không áp dụng được nữa hay trên bảng kết quả đã xuất hiện ít nhất một hàng có đủ các ký hiệu (a_1, a_2, a_3, … a_n) thì dừng và chuyển sang **bước 3**.

Bước 3: Xét bảng kết quả nếu xuất hiện ít nhất một hàng gồm các ký hiệu a_1, a_2, a_3, … a_n thì ta kết luận phép tách ρ là không mất mát thông tin (bảo toàn thông tin). Ngược lại phép tách ρ không bảo toàn thông tin (mất mát thông tin).

Ví dụ 3.6:

Cho quan hệ: HOCSINH (SBD, TEN, DTOAN, DTIN)

Với các phụ thuộc hàm: SBD $\rightarrow$TEN; SBD $\rightarrow$DTOAN, DTIN

Tách thành hai quan hệ:

HS1(SBD, TEN)

HS2(SBD, DTOSN,DTIN)

+ Lập bảng kiểm tra như sau:

	SBD	TEN	DTOAN	DTIN
HS1	a_1	a_2	b_{13}	b_{14}
HS2	a_1	b_{22}	a_3	a_4

+ Làm bằng các giá trị

Ta thấy dòng 2 tại thuộc tính TEN có giá tri là a_2 và b_{22} mà SBD của hai dòng này có giá trị là a_2. Vậy theo phụ thuộc hàm SBD $\rightarrow$TEN nên ta thay giá trị b_{22} của thuộc tính TEN tại dòng 2 là a_2

Ta có bảng:

	SBD	TEN	DTOAN	DTIN
HS1	a_1	a_2	b_{13}	b_{14}
HS2	a_1	a_2	a_3	a_4

Vậy bảng có dòng 2 toàn là giá trị aj (j = 1..4), nên phép tách trên là không mất mát thông tin

3.6 Chuẩn hoá lược đồ quan hệ

Khi thiết kế một lược đồ quan hệ phải tuân theo một số nguyên tắc để khi thao tác trên cơ sở dữ liệu không dẫn đến sự dị thường vê dữ liệu. Công việc thiết kế dữ liệu theo một dạng chuẩn nào đó gọi là chuẩn hoá dữ liệu.

3.6.1. Các dạng chuẩn trong lược đồ quan hệ

Do việc cập nhật dữ liệu (qua phép chèn, loại bỏ và sửa đổi) gây nên những dị thường, cho nên các quan hệ nhất thiết phải được biến đổi thành các dạng phù hợp, quá trình đó gọi là quá trình chuẩn hoá.

Quan hệ được chuẩn hoá là quan hệ trong đó mỗi miền của một thuộc tính chỉ chứa những giá trị nguyên tố, tức là không phân nhỏ được nữa và do đó mỗi giá trị trong quan hệ cũng là nguyên tố.

Quan hệ có chứa các miền giá trị là không nguyên tố gọi là quan hệ chưa chuẩn hoá.

Chuẩn hoá dữ liệu là quá trình phân rã lược đồ quan hệ chưa chuẩn hoá (có dạng chuẩn thấp) thành các lược đồ quan hệ nhỏ hơn nhưng ở dạng chuẩn cao hơn (có cấu trúc tốt hơn) và không làm mất mát thông tin.

Có các dạng chuẩn sau: 1NF, 2NF, 3NF, BCNF

3.6.2. Một số định nghĩa

a) Thuộc tính khoá

Cho lược đồ quan hệ r(U) với tập thuộc tính U, $A_i \in U$; A gọi là thuộc tính khoá của R nếu tồn tại $K \subseteq U$,

Nếu $A \in K$ mà K là khoá thì A là thuộc tính khoá (A có mặt trong ít nhất một tập khoá của r(U,F)).

Ngược lại A không có mặt trong bất kỳ tập khoá nào của r(U,F) thì A là thuộc tính không khoá.

b) Phụ thuộc hàm đầy đủ

Cho R(U) $X,Y \subseteq U$, Y gọi là phụ thuộc hàm đầy đủ vào X, Nếu $X \rightarrow Y$ và $\forall A \in X; (X - \{A\}) \not\rightarrow Y$

Phụ thuộc hàm đầy đủ ký hiệu là $X \overset{+}{\rightarrow} Y$

c) Phụ thuộc hàm bắc cầu

Cho R(U) $X,Y \subseteq U$ Y gọi là phụ thuộc hàm bắc cầu vào X

Nếu $\exists Z \subseteq U$: $Y-Z \neq \emptyset$, $X \rightarrow Z$, $Z \not\rightarrow X$, $Z \rightarrow Y$, $Y \not\rightarrow X$

Phụ thuộc hàm bắc cầu ký hiệu là $X\%\rightarrow Y$

3.6.3. Dạng chuẩn 1NF (1st Normal Form)

Lược đồ quan hệ r gọi là ở dạng chuẩn một (1NF) nếu toàn bộ các miền trị có mặt trong quan hệ R đều chỉ chứa các giá trị nguyên tố.

Ví dụ 3.7: Cho thông tin của bảng đăng ký học của sinh viên:

Bảng 3.7: Bảng đăng ký học của sinh viên

Mã sinh viên	Tên SV	Ngày sinh	Kỳ học	Đăng ký học		
				Mã môn	Tên môn	Số TC
SV01	Hùng	09/12/1990	1	M01	CSDL	2
				M02	Tin ĐC	3
SV02	Trường	07/09/1990	1	M02	Tin ĐC	3
				M03	Tiếng Anh	3

Quan hệ này không phải là dạng chuẩn 1NF vì giá trị trong thuộc tính **Đăng ký học** không phải là nguyên tố.

3.6.4 Dạng chuẩn 2NF (2nd Normal Form)

Lược đồ quan hệ r gọi là dạng chuẩn hai (2NF) nếu r ở dạng chuẩn 1NF và mọi thuộc tính không khoá của r đều phụ thuộc hàm đầy đủ vào khoá.

3.6.5 Dạng chuẩn 3NF (3rd Normal Form)

Lược đồ quan hệ r gọi là dạng chuẩn 3NF nếu đã ở dạng chuẩn 2 và mọi thuộc tính không khoá của r không phụ thuộc hàm bắc cầu vào khoá .

Ví dụ 3.8 :Xét lược đồ quan hệ: r(SAIP); Trong đó $F=\{SI \rightarrow P; S \rightarrow A\}$

Ta thấy r là dạng chuẩn 1

Xét dạng chuẩn 2: Ta có $S \rightarrow A$

$$\text{và} \quad SI \rightarrow S \Rightarrow SI \rightarrow A$$

Thuộc tính A không phụ thuộc đầy đủ vào khoá của r là SI, như vậy r không phải là 2 NF

Ví dụ 3.9:

Xét lược đồ quan hệ: r(SIDM); F={SI $\rightarrow$D; SD $\rightarrow$M}

-Ta thấy r là dạng chuẩn 2

-Xét dạng chuẩn 3:

SI $\rightarrow$D $\Rightarrow$SI $\rightarrow$SD; (theo luật tăng trưởng); Mà SD $\rightarrow$M $\Rightarrow$SI $\rightarrow$M

Vậy M phụ thuộc bắc cầu vào SI nên r không là 3 NF

3.6.6 Dạng chuẩn BCNF (Boye - Code)

Lược đồ quan hệ r gọi là dạng chuẩn BCNF nếu X $\rightarrow$A thoả trên r, nếu A không thuộc X và X là khoá của r.

Ghi chú:

- Các lược đồ quan hệ ở dạng chuẩn 1NF, 2NF vẫn tồn tại các dị thường về dữ liệu.
- Trong một cơ sở dữ liệu tốt, các quan hệ phải được chuẩn hoá về 3NF hoặc BCNF.
- Một lược đồ quan hệ r ở dạng chuẩn BCNF thì r là 3NF

3.7. Các thuật toán

3.7.1 Bao đóng của tập thuộc tính

Cho lược đồ quan hệ r(U,F), với U={A_1, A_2,…, A_n} và F là tập phụ thuộc hàm, X là một tập thuộc tính (X$\subset$U). Bao đóng của tập thuộc tính X đối với tập phụ thuộc hàm F (ký hiệu là X^+) là tập các thuộc tính có thể suy dẫn logíc từ X qua các phụ thuộc hàm có trong F.

$$X^+ = \{ A \mid A \subseteq U; X \rightarrow A \in F^+\}$$

Thuật toán 2: Tìm bao đóng của tập thuộc tính

Cho lược đồ quan hệ r(U,F). Trong đó U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm. Tìm bao đóng của X (X^+)

Tính liên tiếp các tập X_0, X_1, X_2,…theo phương pháp sau:

Bước 0: Đặt $X_0 = X$

Bước i: Lần lượt xét các phụ thuộc hàm của F

Tính X_i ={ $X_{i-1} \cup \{A\}$/ nếu $\exists Y \rightarrow Z \in F$; $A \in Z$ và $A \notin X^{i-1}$; $Y \subseteq X^{i-1}$; loại $Y \rightarrow Z$ khỏi F}

Vì $X_1 \subseteq X_2 \subseteq \ldots \subseteq U$ nên $\exists j$ sao cho $X_j = X_{j-1}$ (tập X không tăng nữa)

Đặt $X^+ = X_j$; Gọi X^+ là bao đóng của X

Mô tả thuật toán bằng ngôn ngữ giả Pascal:

```
Proc Closure;
Input: r=(U,F); Tập thuộc tính X ⊆ U
Output: Y = X+ = {A ∈ U | X → A ∈ F+}
Begin
Y:= X
Repeat
        Z:=Y;
        For each f  L →R in F do
                if L ⊆ Y then
                Y:=Y ∪ R;
                endif;
        endfor;
Until Y=Z;
return Y;
End;
```

Ví dụ 3.9:

Cho lược đồ quan hệ r(U,F) với U= {A,B,C,D,E,G,H}

và F = { A →BC; C →B; D →EH; AD →G}. Tính $(AD)^+$?

Bài giải:

Đặt $X_1 = (AD)$

Chọn các phụ thuộc hàm có vế trái là A,D,AD có A →BC nên

$X_2 = X_1 \cup B = AD \cup B = ABD$

$X_3 = X_2 \cup C = ADB \cup C = ABCD$

Vì D →EH nên

$X_4 = X_3 \cup E = ADBC \cup E = ABCDE$

$X_5 = X_4 \cup H = ADBCE \cup H = ABCDEH$

Vì AD →G nên

$X_6 = X_5 \cup G = ABCDEH \cup G = ABCDEHG$

$X_7 = X_6 = ABCDEHG$

Kết luận: $(AD)^+ = ABCDEFG$

Bài toán thành viên: Cho lược đồ quan hệ r(U,F) với U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm, X, Y là hai tập thuộc tính ($X,Y \subset U$). Hỏi rằng $X \rightarrow Y$ có là thành viên của F hay không? (có nghĩa là $X \rightarrow Y \in F^+$ hay không?)

Để trả lời cho câu hỏi này ta có thể tính F^+ rồi xác định xem $X \rightarrow Y$ có thuộc F^+ hay không. Nhưng việc tính F^+ đòi hỏi thời gian và công sức. Tuy nhiên, thay vì tính F^+ chúng ta có thể sử dụng định lý sau:

Định lý 3.2: $X \rightarrow Y \in F^+$ Khi và chỉ khi $Y \subseteq X^+$

3.7.2. Phủ của tập các phụ thuộc hàm

Cho hai tập phụ thuộc hàm F và G cùng xác định trên tập thuộc tính U. Nói rằng F và G là tương đương nếu $F^+ = G^+$. Nếu F và G tương đương thì ta có thế nói F là một phủ của G (hoặc G là một phủ của F)

Ký hiệu: $F \equiv G$

Phủ không dư thừa: Cho tập phụ thuộc hàm F xác định trên tập thuộc tính U. F được gọi là phủ không dư thừa nếu:

Không tồn tại phụ thuộc hàm $X \rightarrow Y \in G$ mà $(G - \{ X \rightarrow Y\}) \equiv G$

Ví dụ 3.10: Cho lược đồ quan hệ r(U,F) với $F = \{ A \rightarrow B ; B \rightarrow C ; A \rightarrow C\}$

Ta thấy $A \rightarrow C$ là thừa vì $F - \{A \rightarrow C\}$ tương đương với F. Vậy F là phủ dư thừa.

Phủ tối thiểu: Gọi tập các phụ thuộc hàm F là tối thiểu nếu thoả mãn ba điều kiện:

(1) Mọi phụ thuộc hàm thuộc F đều có dạng: $\{X_i \rightarrow A_i \mid i = 1..m\}$ (nói cách khác về phải mỗi phụ thuộc hàm thuộc F chỉ có một thuộc tính).

(2) F là phủ không dư thừa :

Có nghĩa là không tồn tại PTH $X \rightarrow A \in F$ mà $F^+ = (F - \{ X \rightarrow A\})^+$

(3) F không dư thừa thuộc tính nào ở vế trái, nói cách khác không tồn tại một phụ thuộc hàm $X \rightarrow A \in F$; $Z \subset X$ mà : $F^+ = (F - \{X \rightarrow A \} \cup \{Z \rightarrow A\})^+$

Định lý 3.3: Mối phụ thuộc hàm F đều tương đương với một phủ tối thiểu F'.

Thuật toán 3: Cho lược đồ quan hệ r(U,F) với U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm. Tìm phủ tối thiểu của F.

Bước 1: Tách các PTH sao cho vế phải của mỗi PTH chỉ có một thuộc tính.

Giả sử xét phụ thuộc hàm $X \rightarrow Y$, với $Y = A_1A_2A_3...A_n$ Ta có thể tách thành các phụ thuộc hàm sau:

$X \rightarrow A_1$

$X \rightarrow A_2$

...

$X \rightarrow A_m$

Kết quả ta được F^1 tương đương với F

Bước 2: Loại bỏ các phụ thuộc hàm (PTH) dư thừa.

Gải sử có F_i có dạng $X_j \rightarrow A_j$ | j = 1,2.. m

Đặt $F_0 = F^1$

$$F_i = \begin{cases} F_{i-1} \setminus \{X_j \rightarrow A_j\} \text{ nếu } F_{i-1} \setminus \{X_j \rightarrow A_j\} \text{ tương đương với } F_{i-1} \\ F_{i-1} \text{ nếu ngược lại} \end{cases}$$

Sau m lần ta được $F_m = F_{m-1}$

Đặt $F^2 = F_m$ tương đương với F^1.

Bước 3: Loại bỏ các thuộc tính dư thừa bên trái của mỗi phụ thuộc hàm

Sau bước 2 có $F^2 = \{X_i \rightarrow A_j$ | với i =1..n và X_i có dạng $X_i = A_1, A_2, ... A_n$

-Đặt $X_0 = X_i$

$$X_j = \begin{cases} X_{j-1} \setminus \{A_j\} \text{ nếu } \{F_2 \setminus (X_{i-1} \rightarrow A_i) \cup (X_{i-1} \setminus A_j) \rightarrow A_i\} \text{ tương đương với } F^2 \\ X_{j-1} \text{ nếu ngược lại} \end{cases}$$

Lặp lại quy tắc trên n lần thì ta xét xong phụ thuộc hàm $X_i \rightarrow A_j$ (Có nghĩa là đã loại bỏ tất cả các thuộc tính dư thừa bên trái trong phụ thuộc hàm trên).

Sau bước này ta được F^3 tương đương với F^2. F^3 là phủ tối thiểu của F

Ví dụ 3.11:

Cho lược đồ quan hệ r(U,F) với U={A,B,C,D,E,G,H,L}

và F={A→BC; C→B; D→EL; ADC→G}. Tìm phủ tối thiểu của F?

Bước 1: Tách các phụ thuộc hàm:

Ta có F_1={A→C;C→B;A→B;D→E;D→L;ADC→G}

Bước 2: Loại bỏ các phụ thuộc hàm dư thừa :

$A \rightarrow C$ không dư thừa vì $C \notin A^+ = AB$

$C \rightarrow B$ không dư thừa vì $B \notin C^+ = C$

$A \rightarrow B$ dư thừa vì có $B \in A^+ = ABC$

$D \rightarrow E$ không dư thừa vì có $E \notin D^+ = D$

$D \rightarrow L$ không dư thừa vì có $L \notin D^+ = D$

$ADC \rightarrow G$ không dư thừa vi có $G \notin (ADC)^+ = (ABCDEL)$

Ta có $F_2 = \{C \rightarrow B; A \rightarrow C; D \rightarrow E; D \rightarrow L; ADC \rightarrow G\}$

Bước 3: Loại bỏ các thuộc tính dư thừa ở vế trái:

Vì $A \rightarrow C$ nên phụ thuộc hàm $ADC \rightarrow G$ thừa thuộc tính C nên ta có $AD \rightarrow G$

Ta có $F_3 = \{C \rightarrow B; A \rightarrow C; D \rightarrow E; D \rightarrow F; AD \rightarrow G\}$

Kết luận: Phủ tối thiểu của F là : $F_{tt} = F_3 = \{C \rightarrow B; A \rightarrow C; D \rightarrow E; D \rightarrow F; AD \rightarrow G\}$

Thuật toán 4: Cho hai tập phụ thuộc hàm F và G xác định trên tập thuộc tính U. Với F và G có dạng sau : $F = \{X_i \rightarrow Y_i\} \mid i = 1..m$; $G = \{X_j \rightarrow Y_j\} \mid j = 1..n$. Hãy kiểm tra F và G có tương đương với nhau hay không ?

Bước 1: Với $\forall i = 1..m$ kiểm tra xem $X_i \rightarrow Y_i \in F$ có thuộc G^+ hay không. Theo bài toán thành viên ta kiểm tra xem có thoả $Y_i \subseteq X_J^+$ nếu thoả thì $F^+ \subseteq G^+$

Bước 2: Với $\forall j = 1..n$ kiểm tra xem $X_j \rightarrow Y_j \in G$ có thuộc F^+ không. Theo bài toán thành viên ta kiểm tra xem có thoả $Y_j \subseteq X_i^+$ nếu thoả thì $G^+ \subseteq F^+$

Nếu thoả cả hai điều kiện trên thì $G^+ = F^+$ và ta nói F và G tương đương, ngược lại trong quá trình kiểm tra nếu tồn tại một phụ thuộc hàm không thoả mãn thì F và G là không tương tương nhau.

3.7.4. Khoá tối thiểu của sơ đồ quan hệ.

Cho lược đồ quan hệ $r(U,F)$, $K \subseteq U$. K được gọi là khoá tối thiểu của lược đồ quan hệ nếu:

(1) $K^+ = U$

(2) $\forall A \in K; (K - \{A\})^+ \neq U$

Hai điều kiện trên tương ứng với:

(3) $K \rightarrow U$

(4) $\forall A \in K; (K - \{A\})^+ ! \rightarrow U$

Chú ý:

- K chỉ thoả mãn điều kiện (1) thì K gọi là siêu khoá.

- Trong một số tài liệu thuật ngữ *khoá* được dùng theo nghĩa *siêu khoá* và thuật ngữ *khoá tối thiểu* dùng theo nghĩa *khoá*

- Một sơ đồ quan hệ có ít nhất một tập khoá. Goi M là giao của các khoá:

$$M = \cup (L \backslash R)$$

- L\R là thuộc tính chỉ xuất hiện ở vế trái và không xuất hiện ở vế phải của các phụ thuộc hàm có trong F

- Nếu $M^+ = U$ thì r có một khoá duy nhất ngược lại $M^+ \neq U$ thì r có nhiều hơn một tập khoá.

Thuật toán 5: Tìm một khoá tối thiểu của lược đồ quan hệ

Cho lược đồ quan hệ r (U, F). với $U=\{A_1, A_2, ...A_n\}$ và F là tập phụ thuộc hàm. Tìm khoá tối thiểu của r(U,F)

Bước 1: Đặt $K_0 = U$

Bước i: Tính $K_i = \begin{cases} K_{i-1} \setminus A_j \text{ nếu } (K_{i-1} \setminus A_j)^+ = U, j=1,2\ldots n \\ K_{i-1} \text{ nếu ngược lại} \end{cases}$

Lặp lại bước i n lần

Kết luận: Khoá tối thiểu của r(U,F) là K_i.

Mô tả thuật toán bằng ngôn ngữ giả PASCAl

```
Proc Key;
Input: Tập thuộc tính U; Tập F
Output: K ⊆ U thoả điều kiện
                (1) K+ = U
                (2) ∀A ∈ K; (K –{A})+ ≠ U
Begin
        K:=U;
For each attribute A in U do
        if A ∈ (K- A)+ then
        K:=K-A;
        endif;
endfor;
```

```
return K;
End;
```

Ví dụ 3.12: Cho lược đồ quan hệ r(U,F) có:

U = {A,B,C,D,E,L,G,H}

F = { A $\rightarrow$BC; C $\rightarrow$B; D $\rightarrow$EL; ADC $\rightarrow$G}. Tìm một khoá cho r(U,F)?

Bài giải:

Bước 0: Đặt K_0=U={A,B,C,D,E,L,G,H}

Bước 1: K_1=K_0 \A = {A,B,C,D,E,L,G,H} vì $(K_0 \backslash A)^+ \neq U$

Bước 2: K_2 =K_1 \B={A,C,D,E,L,G,H} vì $(K_1 \backslash B)^+ = U$

Bước 3: K_3 =K_2\C ={A,D,E,L,G,H} vì $(K_2 \backslash C)^+ = U$

Bước 4: K_4 =K_3 ={A,D,E,L,G,H} vì $(K_3 \backslash D)^+ \neq U$

Bước 5: K_5 = K_4 \E ={A,D,L,G,H} vì $(K_4 \backslash E)^+ = U$

Bước 6: K_6 = K_5\L ={A,D,G,H} vì $(K_5 \backslash L)^+ = U$

Bước 7: K_7 = K_6\G ={A,D,H} vì $(K_6 \backslash G)^+ = U$

Bước 8: K_8 =K_7={A,D,H} vì $(K_7 \backslash H)^+ \neq U$

Vậy khóa là K ={A,D,H}

Thuật toán 6: Tìm tất cả các khoá của lược đồ quan hệ

Cho lược đồ quan hệ r(U,F) với U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm. Tìm tất cả các khoá cho r(U,F)?

Phương pháp: Thực hiện theo các bước sau:

Bước 1: Tạo các tập thuộc tính nguồn (TN) và tập thuộc tính trung gian (TG).

TN- Chứa các thuộc tính chỉ xuất hiện ở vế trái và không xuất hiện ở về phải của các phụ thuộc hàm có trong F.

TG- Chứa các thuộc tính vừa xuất hiện ở vế trái vừa xuất hiên ở vế phải của các phụ thuộc hàm có trong F.

Bước 2:

Nếu TG=$\emptyset$ thì r(U,F) chỉ có một khoá K

K=TN

kết thúc thuật toán

Ngược lại

Qua bước 3

Bước 3: Tìm tất cả các tập con X_i của tập trung gian TG.

Bước 4: Tìm các siêu khoá S_i bằng cách với $\forall X_i$

Nếu $(TN \cup X_i)^+ = U$ thì $S_i = TN \cup X_i$

Bước 5: Tìm khoá bằng cách loại các khoá không tối thiểu

Gọi S là tập siêu khoá xác định được ở bước 4, $S=(S_1, S_2, \ldots, S_n)$

Nếu $S_i \subset S_j$ thì loại S_j ra khỏi tập siêu khoá S

S còn lại chính là tập khoá cần tìm.

Ví dụ 3.13:

Cho lược đồ quan hệ r(U,F) với U={CDEKGH} và F={CK → H, C → D, E → C, E → G CK → E}. Tìm tất cả các khoá cho r(U,F) ?

Bước 1 : Xác định các tập thuộc tính TN, TG với TN={K} và TG={CE}

Từ bước 2 đến bước 5: Minh hoạ qua bảng dữ liệu sau

Gọi X_i là tập con của TG

X_i	$TN \cup X_i$	$(TN \cup X_i)^+$	Siêu khoá	Khoá
ϕ	K	K		
C	KC	U	KC	K

				C
E	KE	U	KE	KE
EC	KCE	U	KCE	

Kết luận : Các khoá của r(U,F) là: KC và KE

Ví dụ 3.14 :

Cho r(U,F) với U=(A, B, C, D, E, I).

Và F={ACD → EBI; CE → AD}. Tìm tất cả các khoá cho r(U,F)?

Bài giải:

TN={ C }; TG={ ADE }

X_1	(TN $\cup X_1$)	$(TN \cup X_1)^+$	Siêu khóa	Khóa
θ	C	C		
A	AC	AC		
D	CD	CD		
AD	ACD	ABCDEI	ACD	ACD
E	CE	ABCDEI	CE	CE
AE	ACE	ABCDEI	ACE	
DE	CDE	ABCDEI	CDE	
ADE	ADCE	ABCDEI	ACDE	

Kết luận : Tất cả các khoá của r(U,F) là : CE và ACD

3.7.5 Các bước chuẩn hoá một quan hệ đến 3NF

Thuật toán 7: Chuẩn hoá một quan hệ thành dạng 3NF

Cho lược đồ quan hệ r(U,F) với U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm. Hãy chuẩn hoá r(U,F) về dạng 3NF và phép tách ρ là không mất mát thông tin

$\rho = \{r_1(U_1), r_2(U2)... r_n(Un)\}$, sao cho $r_i(Ui)$ là dạng chuẩn 3NF

Bước 1: Tìm khoá của r

Bước 2: Sử dụng thuật toán 2 tìm phủ tối thiểu của F

Bước 3: Xác định các lược đồ con

Mỗi phụ thuộc hàm thuộc F' tương đương với một lược đồ con.

Giả sử xét $Y \rightarrow A_i \in F_{\text{tốithiểu}}$ ta có lược đồ $r_j(U_j)$, với $U_j = YA_i$ khoá của $r_j(U_j)$ là Y với $j \subseteq (1,\ldots,n)$ và $Y \subseteq U$

Lưu ý:

Nếu $\exists\ X_i \rightarrow A_{i1}$; $X_i \rightarrow A_{i2}$; ...; $X_i \rightarrow A_{il}$ thì $U_i = (X_iA_1A_2...A_i)$ và ta có lược đồ quan hệ $r_i(U_i)$ với khoá là X_i, $i \subseteq (1,,n)$ và $X_i \subseteq U$.

Kết quả ta có các lược đồ: $\{r1(U_1), r_2(U2)... r_n(Un)\}$

Bước 4: Kết luận phép chuẩn hoá

Nếu tồn tại ít nhất một thuộc tính khoá không có mặt trong các lược đồ $\{r1(U_1), r_2(U2)... r_n(Un)\}$ thì kết luận phép chuẩn hoá r(U,F) về 3NF là

$\rho = \{r_0(U_0), r1(U_1), r_2(U2)... r_n(Un)\}$ với $r_0(U_0)=K$

Ngược lại phép chuẩn hoá r(U,F) về 3NF là $\rho = \{r1(U_1), r_2(U2)... r_n(Un)\}$

Ví dụ 3.15:

Cho lược đồ quan hệ r(U,F) với U = {A,B,C,D,E,L,G,H}

và $F = \{ A \rightarrow BC; C \rightarrow B; D \rightarrow EL; ADC \rightarrow G\}$

Chuẩn hoá r thành dạng 3NF

Bước 1: Tìm khoá tối thiểu:

$K_0 = U = \{A,B,C,D,E,G,H,L\}$ dùng thuật toán 4 loại bỏ dần ta có

K = ADH

Bước 2:

Tìm phủ tối thiểu

2.1 Tách các phụ thuộc hàm

$F = \{ A \rightarrow C; C \rightarrow B; A \rightarrow B; D \rightarrow E; D \rightarrow L; ADC \rightarrow G\}$

2.2 Loại bỏ các phụ thuộc hàm dư thừa: Có $A \rightarrow B$ thừa vì có $A \rightarrow C$ và $C \rightarrow B$

Ta có $F = \{C \rightarrow B; A \rightarrow C; D \rightarrow E; D \rightarrow L; ADC \rightarrow G\}$

2.3 Bỏ các thuộc tính thừa ở vế trái

Vì $A \rightarrow C$ nên phụ thuộc hàm $ADC \rightarrow G$ thừa thuộc tính C nên ta có: $AD \rightarrow G$

$\Rightarrow F = \{C \rightarrow B; A \rightarrow C; D \rightarrow E; D \rightarrow L; AD \rightarrow G\}$

Bước 3: Ta có các r_i như sau:

$A \rightarrow C \Rightarrow R_1(U_1) = (AC)$ khoá $K_1 = \{A\}$

$C \rightarrow B \Rightarrow R_2(U_2) = (CB)$ khoá $K_2 = \{B\}$

$D \rightarrow E; D \rightarrow L; \Rightarrow R_3(U_3) = (DEL)$ khoá $K_3 = \{D\}$

$AD \rightarrow G \Rightarrow R_4(U_4) = (ADG)$ khoá $K_4 = \{AD\}$

Bước 4: Kết quả

$\rho = \{r_0(ADH), r_1(AC), r_2(CB), r_3(DEL), r_4(ADG)\}$

Thoả mãn $R_i(U_i)$ là 3NF

Thuật toán 8: Chuẩn hoá một lược đồ quan hệ về dạng chuẩn Boye -Code

Cho lược đồ quan hệ r(U,F) với U là tập hữu hạn các thuộc tính và F là tập phụ thuộc hàm. Hãy chuẩn hoá r(U,F) về dạng chuẩn Boye - Code.

Bước 1: Goi ρ là phép chuẩn hoá r(U,F) về BCNF, $\rho = \{r(U,F)\}$

Bước i: Nếu s là một lược đồ thuộc ρ, s chưa ở BCNF chọn $X \rightarrow A$ là một phụ thuộc hàm thoả trên S mà X không phải là siêu khoá của S và $A \notin X$ thì tách S thành hai lược đồ quan hệ:

$s_i = X \cup Y = s_i$ (XY) Khoá là X

$s_{i+1} = (U \setminus Y)$, Xác định lại khoá và tập phụ thuộc hàm cho s_{i+1}.

Quá trình tiếp tục cho tới khi tất cả các lược đồ đều ở BCNF

Kết quả được phép tách:

$\rho = \{r_1(U_1), r_2(U_2),...,r_m(U_n)\}$ với mỗi r_i là quan hệ ở dạng BCNF

Ví dụ 3.16: Cho lược đồ quan hệ r(U,F)

$U = \{C, T, H, N, S, G\}$

$F = \{ CS \rightarrow G ; C \rightarrow T; HT \rightarrow N; HS \rightarrow N; HN \rightarrow C\}$. Hãy chuẩn hoá r(U,F) về dạng chuẩn BCNF?

Khoá của r là HS

Bước 1: r(U,F) không thoả mãn BCNF

Xét $CS \rightarrow G$ vì CS không là siêu khoá của r nên tách:

$r_1 = (CSG)$; $r_2 = (CTHNS)$ và $F = \{C \rightarrow T; HT \rightarrow N; HS \rightarrow N; HN \rightarrow C\}$, khoá của r_2 là HS. Trong đó r_1 thoả mãn BCNF, r_2 không thoả mãn BCNF.

Bước 2:

Xét r2

Có C $\rightarrow$T mà C không là khoá của r2 nên tách:

r_{21} = (CT); r_{22} = (CHNS) và F= { HC $\rightarrow$N ; HS $\rightarrow$N; HN $\rightarrow$C; }, khoá của r_{22} là HS. Trong đó r_{21} thoả mãn là BCNF, r_{22} không thoả mãn BCNF.

Bước 3:

Xét r22:

Có HN $\rightarrow$C mà CH không là khoá nên tách:

r_{221} = (CHN) ; r_{222} = (CHS); r_{221} thoả mãn CBNF; r_{222} thoả mãn BCNF

Thuật toán dừng vì $\forall$ r_i thoả mãn BCNF.

Cuối cùng ta có:

ρ = {r_1(CSG); r_{21}(CT); r_{221}(CHN); r_{222}(CHS)}

Trong đó mỗi r_i là BCNF.

3.8 Phụ thuộc đa trị

3.8.1 Khái niệm

Ta thấy dữ liệu có mối quan hệ với nhau đó là phụ thuộc hàm. Tuy vậy cũng có trường hợp quan hệ đó không có sự phụ thuộc hàm. ánh xạ trên các thuộc tính không phải là đơn trị mà có nhiều giá trị. Mối quan hệ đó gọi là phụ thuộc đa trị (Multivalued Defendency-MVD)

Ví dụ: Quan hệ KHDH (kế hoạch dạy học)

Giáo viên	Môn	Lớp
A	M2	K1
A	M1	K2
A	M2	K2
A	M1	K1

Như vậy với một giáo viên ta chưa hẳn đã xác định được dạy lớp nào, môn gì cụ thể

3.8.2 Định nghĩa

Cho R là một lược đồ quan hệ X, Y là 2 tập con của R. Z = R – XY. Quan hệ r(R) gọi là phụ thuộc đa trị nếu với bất kỳ 2 bộ t1, t2 $\in$ r, với t1[X] = t2[X] tồn tại một bộ t3 $\in$ r sao cho:

t3[X] = t1[X]; t3[Y] = t1[Y]; t3[Z] = t2[Z];

Ký hiệu phụ thuộc đa trị

$X \rightarrow\!\!\!\!\rightarrow Y$

Ta nói X xác định đa trị Y; hay Y phụ thuộc đa trị vào X

Ví dụ: Xét quan hệ KHDH trên là phụ thuộc đa trị

Nhận xét:

-Xét mô hình trên ta còn có: t4[Z] = t1[Z]; t2[Y] = t4[Y]

-Nếu Y = ϕ thì $X \rightarrow\!\!\!\!\rightarrow \phi$ đúng với mọi quan hệ

-Nếu X = ϕ thì $\phi \rightarrow\!\!\!\!\rightarrow Y$ đúng khi Y độc lập với các thuộc tính khác trong r

3.8.3 Hệ tiên đề:

(1) Tiên đề bù: $X \rightarrow\!\!\!\!\rightarrow Y \Rightarrow Y \rightarrow\!\!\!\!\rightarrow U \backslash Y \backslash X$

(2) Tiên đề tưng trưởng: $X \rightarrow\!\!\!\!\rightarrow Y$; $V \in W \Rightarrow WX \rightarrow\!\!\!\!\rightarrow VY$

(3) Tiên để bắc cầu: $X \rightarrow\!\!\!\!\rightarrow Y$ và $Y \rightarrow\!\!\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\!\!\rightarrow Z \backslash Y$

(4) Tiên đề về quan hệ phụ thuộc đơn trị và đa trị: $X \rightarrow Y$ thì $X \rightarrow\!\!\!\!\rightarrow Y$

4. Nếu $X \rightarrow\!\!\!\!\rightarrow Y$, $Z \subseteq Y$, $W \cap Y = \phi$, $W \rightarrow Z$ thì $X \rightarrow Z$

3.8.4 Các luật suy diễn của phụ thuộc đa trị

Luật hợp:

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

Luật tựa bắc cầu

Nếu $X \rightarrow\!\!\!\!\rightarrow Y$ và $WY \rightarrow\!\!\!\!\rightarrow Z$ thì $WX \rightarrow\!\!\!\!\rightarrow Z \backslash WX$

Luật tựa bắc cầu hỗn hợp

Nếu $X \rightarrow\!\!\!\!\rightarrow Y$ và $XY \rightarrow\!\!\!\!\rightarrow Z$ thì $X \rightarrow\!\!\!\!\rightarrow Z \backslash Y$

Luật tách

Nếu $X \rightarrow\!\!\!\!\rightarrow Y$ và $X \rightarrow\!\!\!\!\rightarrow Z$ thì $X \rightarrow\!\!\!\!\rightarrow Y \backslash Z$; $X \rightarrow\!\!\!\!\rightarrow Z \backslash Y$; $X \rightarrow\!\!\!\!\rightarrow Y \cap Z$

Định lý: Cho r(U) có phép tách $\rho = \{r_1(U1), r_2(U2)\}$ là phép tách hai không mất mát thông tin khi và chỉ khi:

$U1 \cap U2 \rightarrow\!\!\!\!\rightarrow U1 \setminus U2$

$U1 \cap U2 \rightarrow\!\!\!\!\rightarrow U2 \setminus U1$

3.8.5 Dạng chuẩn 4 NF

-Một phụ thuộc hàm đa trị $X \rightarrow\!\!\!\!\rightarrow Y$ gọi là sơ cấp nếu với X, $Y \neq \phi$ $X \cup Y \neq U$ mà $\forall X' < X \Rightarrow X' \not\rightarrow\!\!\!\!\rightarrow Y$

Quan hệ r gọi là dạng chuẩn 4 nếu mọi phụ thuộc đa trị sơ cấp đều được xác định bởi khoá chính

Ví dụ: Quan hệ KHDH trên

Khoá chính: GML

Tách: KHDH1: (GM) có G $\rightarrow\rightarrow$M

KHDH2: (GL) có G $\rightarrow\rightarrow$L

CÂU HỎI VÀ BÀI TẬP CHƯƠNG 3

1. Định nghĩa phụ thuộc hàm và các khái niệm liên quan.
2. Định nghĩa lược đồ quan hệ và cho ví dụ minh họa
3. Phát biểu tiên đề Armstrong và các hệ quả.
4. Định nghĩa bao đóng của một tập thuộc tính.
5. Định nghĩa phủ của một tập phụ thuộc hàm. Phủ tối thiểu.
6. Định nghĩa phép tách một lược đồ quan hệ.
7. Nêu các dạng chuẩn 1NF, 2NF, 3NF, BCNF và cho ví dụ minh hoạ.
8. Cho lược đồ quan hệ r(U,F). Tập thuộc tính U = {ABDEGIH}. Tập phụ thuộc hàm: F ={AB$\rightarrow$E, AG$\rightarrow$I, BE$\rightarrow$I, E$\rightarrow$G, GI$\rightarrow$H}. Chứng minh: AB$\rightarrow$GH.
9. Cho lược đồ quan hệ r(U,F) với U={ABCDEGH); F ={AB$\rightarrow$C, C$\rightarrow$D, CD$\rightarrow$E, CE$\rightarrow$GH, G$\rightarrow$A}.Chứng minh : AB$\rightarrow$E ; AB$\rightarrow$G
10. Cho sơ đồ quan hệ r(U,F) với U={ABCDEGH}; F={A$\rightarrow$D, AB $\rightarrow$DE, CE$\rightarrow$G, E$\rightarrow$H}. Hãy tính $(AB)^+$
11. Cho sơ đồ quan hệ r(U,F) với U={ABCDEG); F={A$\rightarrow$D, AB $\rightarrow$E, BG$\rightarrow$E, CD$\rightarrow$G,E$\rightarrow$C}. Hãy tính $(AB)^+$
12. Cho sơ đồ quan hệ r(U,F); U={ABCDEH}; F={BC$\rightarrow$E, D$\rightarrow$A, C$\rightarrow$A, AE$\rightarrow$D, BE$\rightarrow$CH}. Tìm một khoá tối thiểu cho r(U,F)
13. Cho sơ đồ quan hệ r(U,F), với U={DBIOQS}; F={S$\rightarrow$D, I$\rightarrow$B, IS$\rightarrow$Q, B$\rightarrow$O}. Hãy chuẩn hoá r(U,F) về dạng chuẩn 3NF.
14. Cho sơ đồ quan hệ r(U,F); U=(ABCDEGH);F={ABC$\rightarrow$D, AB$\rightarrow$E, BC$\rightarrow$DC,C$\rightarrow$DE, CE$\rightarrow$H, DC$\rightarrow$G, CH$\rightarrow$G, AD$\rightarrow$H}. Hãy chuẩn hoá r(U,F) về dạng chuẩn 3NF.
15. Cho lược đồ quan hệ r(U,F). U= {C#, I,D,B,K,E,L}; F= { C#$\rightarrow$IBKE, D $\rightarrow$B, K $\rightarrow$E}. Hãy chuẩn hoá r(U,F) về dạng chuẩn 3NF.

16. Cho sơ đồ quan hệ r(U,F), U={ABCD}; F={$D \to B, C \to A, B \to ACD$}. Hãy xác định dạng chuẩn cao nhất của r(U)? Giải thích.

17. Kiểm tra tính kết nối không mất mát thông tin của phép tách r(ABCDE) thành các lược đồ quan hệ sau: r1=AD; r2=AB; r3=BE; r4=CDE; r5 = AE. Với tập phụ thuộc hàm: F={$A \to C$, $B \to C$; $A \to B$; $DE \to C$; $CE \to A$}.

18. Cho lược đồ quan hệ r(U,F); U={ABCDEG}; F ={$AB \to C$; $C \to B$; $ABD \to E$; $G \to A$}.Chuẩn hoá r thành dạng BCNF.

19. Kiểm tra tính không mất mát thông tin của phép tách r(ABCDEG) thành $\rho(r)$ = (BC,AC,ADBE,ADBF).

Với tập phụ thuộc hàm F={$AB \to C$; $C \to B$; $ABD \to E$; $G \to A$}

20. Cho lược đồ quan hệ r(U,F) với

U=(ABCDE)

F={$AB \to C$; $AD \to B$; $B \to D$}.

a) Tìm giao của các khoá?

b) Tìm tất cả các khoá cho r(U,F)?

21. Cho lược đồ quan hệ r(U,F)

U=(ABCDEGH)

F={ $B \to AC$, $DH \to AE$, $AC \to BE$, $E \to H$, $A \to D$, $G \to E$}.

a) Tìm một khoá cho r(U,F).

b) Chứng minh rằng: $CG \to EH \in F^+$

c) Các tập thuộc tính CGH và ABCG có phải là khoá của r hay không?

22. Cho lược đồ quan hệ r(U,F):

U = {A, B, C, D, E, G}

F= {$AB \to C, D \to EG, BE \to C, BC \to D, CG \to BD, ACD \to B, CE \to AG$}.

a). Tìm tất cả các khoá của r(U,F)

b). Tìm giao của khoá

Tìm phủ tối thiểu của F

Chuẩn hoá r(U,F) về dạng chuẩn 3NF

Chương 4

NGÔN NGỮ ĐỊNH NGHĨA VÀ THAO TÁC DỮ LIỆU

4.1 Ngôn ngữ đại số quan hệ

4.1.1 Khái niệm

Là ngôn ngữ dựa trên các phép toán của đại số quan hệ mà ta đã đề cập tới trong Chương 2. Mỗi câu hỏi được biểu diễn bằng một tập các phép toán nào đó.

4.1.2 Các câu lệnh của ngôn ngữ đại số quan hệ

5. ***Phép hợp***

   <quan hệ 1> UNION < quan hệ 2>

6. ***Phép giao***

   <quan hệ 1 > INTERSECT < quan hệ 2>

7. ***Phép trừ***

   <quan hệ 1 > MINUS < quan hệ 2>

8. ***Phép tích Đề- các***

   <quan hệ 1 > TIMES < quan hệ 2>

9. ***Phép chọn***

   SELECT <quan hệ> WHERE < điều kiện>

10. ***Phép chiếu***

    PROJECT <quan hệ> OVER < danh sách thuộc tính>

11. ***Phép kết nối***

    JOIN <quan hệ 1> AND <quan hệ 2> [OVER <danh sách thuộc tính>] [WHERE < danh sách thuộc tính>]

12. ***Phép chia***

    DIVIDE <quan hệ 1> BY <quan hệ 2> OVER <danh sách thuộc tính> [AND <danh sách thuộc tính>]

13. ***Đưa ra kết quả***

    GIVING <kết quả>

4.1.3 Ví dụ minh hoạ

-Bổ xung vào quan hệ CONGTY một công ty nữa

Congty UNION {“CT4”, “Hồng Hà”,1200000, “Nam định”}

GIVING Congty

-Xóa tên công ty CT5

Congty MINUS {“CT5”, , , }

GIVING CongTy

-Sửa địa chỉ của công ty Hồng Hà thành Hà nội, thực chất là xoá bộ cũ thay bộ mới với nội dung mới

Congty MINUS (“CT4”, , ,)

GIVING Tgian

Tgian UNION (“CT4”, , “Hà nội”)

GIVING CongTy

***Chú ý:** Lệnh này cần đề phòng mất dữ liệu*

-Tìm kiếm thông tin về công ty CT1

SELECT CongTy WHERE MaCongTy = “ CT1”

GIVING CongTy

4.1.4 Biểu diễn một số câu hỏi

14. Đưa ra danh sách các mặt hàng màu đỏ

    SELECT Hanghoa WHERE Mau = “Đỏ”

    GIVING Ketqua

15. Cho biết mã các công ty cung cấp mặt hàng H1

    SELECT CungCap WHERE MaHang = “H1” GIVING Tgian

    PROJECT Tgian OVER MaCongTy GIVING Ketqua

Hoặc:

PROJECT (SELECT CungCap WHERE MaHang= ”H1”) OVER MaCongTy

GIVING Ketqua

16. Cho biết tên công ty cung cấp mặt hàng H1

    SELECT Cungcap WHERE MaHang = “H1” GIVING Tgian1

    JOIN Tgian1 AND Congty OVER MaCongTy GIVING Tgian2

    PROJECT Tgian2 OVER TenCongTy GIVING Ketqua

17. Cho biết tên công ty cung cấp cả hai mặt hàng H1 và H2

    SELECT CungCap WHERE MaHang= ”H1” GIVING Tgian1

    PROJECT Tgian1 OVER MaCongTy GIVING Tgian2

SELECT CungCap WHERE MaHang= ”H2” GIVING Tgian1’
PROJECT Tgian1’ OVER MaCongTy GIVING Tgian2’
Tgian2 INTERSECT Tgian2’ GIVING Tgian
JOIN Tgian AND Congty OVER MaCongTy GIVING Tgian’
PROJECT Tgian’ OVER TenCongTy GIVING Ketqua

4.2 Ngôn ngữ SQL (Structure Query Language)

4.2.1 Giới thiệu

SQL được phát triển từ ngôn ngữ SEQUEL-2, thử nghiệm và cài đặt tại Trung tâm nghiên cứu của hãng IBM ở San Joes (California-US), cho hệ thống Quản trị cơ sở dữ liệu lớn điển hình là System - R, SQL vừa đóng vai trò là một ngôn ngữ có thể thao tác độc lập của người sử dụng đầu cuối, đồng thời lại có khả năng là một ngôn ngữ con được nhúng trong ngôn ngữ chủ.

SQL là ngôn ngữ phi thủ tục, dễ sử dụng. Do vậy hiện nay rất nhiều hệ quản trị cơ sở dữ liệu sử dụng ngôn ngữ SQL như ACCESS, ORACLE, DB2...

SQL cho phép tạo lập và thực hiện các thao tác truy xuất trên cơ sở dữ liệu một cách rất dễ dàng.

4.2.2 Nhóm lệnh tạo lập cơ sở dữ liệu

a) Lệnh tạo bảng

CREATE TABLE <tên_bảng>
(<tên_cột> <kiểu_dữ_liệu> [<ràng_buộc_dữ_liệu>] [, ...*n*]

[, < ràng_buộc_bảng >])

Trong đó:

+ tên_bảng, tên_cột: do người sử dụng tự định nghĩa tuân theo quy tắc đặt tên

- Tên cột bắt đầu bằng chữ hoa, sao cho ngắn gọn chính xác và đầy đủ.
- Không nên đặt tên bảng và tên cột có khoảng trắng.
- Không nên đặt tên bảng và tên cột trùng với các từ khóa.

+ Kiểu_dữ_liệu: Chọn một kiểu dữ liệu nào phù hợp với dữ liệu người dùng sẽ nhập vào. Bảng 4.1 cho thấy các kiểu dữ liệu được quy định:

Bảng 4.1 Các kiểu dữ liệu

Tên kiểu dữ liệu	Phạm vi biểu diễn	Kích thước
bigint	Từ 2^63 (-9223372036854775808) đến 2^63-1 (9223372036854775807)	Kiểu số nguyên, 8 byte
int	Từ -2^31 (-2,147,483,648) đến 2^31 - 1 (2,147,483,647)	Kiểu số nguyên. 4 byte
smallint	Từ 2^15 (-32,768) đến 2^15 - 1 (32,767)	Kiểu số nguyên 2 byte
tinyint	Từ 0 đến 255	Kiểu số nguyên 1 byte
bit	Biểu diễn giá trị 0 hoặc 1	Kiểu số nguyên
decimal(n,p)	Từ (-10^38 +1) đến (10^38 –1)	Kiểu số thực tĩnh
numeric(n,p)	Từ (-10^38 +1) đến (10^38 –1)	Kiểu số thực tĩnh
money	Từ -2^63 (-922,337,203,685,477.5808) đến 2^63 - 1 (+922,337,203,685,477.5807),	Kiểu dữ liệu tiển tệ
smallmoney	Từ -214,748.3648 đến +214,748.3647	Kiểu dữ liệu tiển tệ
float	Từ (-1.79E + 308) đến (1.79E + 308)	Kiểu số thực động
real	Từ (-3.40E + 38) đến (3.40E + 38)	Kiểu số thực động
datetime	Từ (January 1- 1753), đến (December 31- 9999)	Kiểu dữ liệu ngày giờ
smalldatetime	Từ (January 1, 1900), đến (June 6, 2079)	Kiểu dữ liệu ngày giờ
char(n)	Kiểu ký tự có độ dài cố định, n_{max}=8000 ký tự	Kiểu ký tự
varchar(n)	Kiểu ký tự có độ dài thay đổi, n_{max}=8000 ký tự	Kiểu ký tự

text	Kiểu ký tự có độ dài tối đa là 2^31 - 1 (2,147,483,647) ký tự	Kiểu ký tự

+ Ràng _buộc_dữ_liệu: Là một số quy định dụng để kiểm tra dữ liệu khi thực hiện các thao tác nhập hoặc cập nhật dữ liệu, có các loại ràng buộc sau:

- Null (mặc định): Chấp nhận giá trị rỗng
- Not Null: Không chấp nhận giá trị rỗng
- Unique: giá trị nhập vào phải duy nhất.
- Primary key: Ràng buộc khoá chính dùng để xác định duy nhất một đối tượng nên giá trị của chúng phải duy nhất, không chấp nhận giá trị Null.

+ Ràng_buộc_bảng: gồm 2 loại là ràng buộc khoá chính và ràng buộc khoá ngoài

Ràng buộc khóa chính được định nghĩa theo cú pháp sau:

CONSTRAINT <tên_ràng_buộc> PRIMARY KEY (danh_sách_thuộc_tính_khoá)

Ràng buộc khóa ngoài: Dùng để kiểm tra sự tương quan về dữ liệu giữa khoá chính và khoá ngoài ở hai bảng dữ liệu có mối quan hệ với nhau, được định nghĩa theo cú pháp sau:

CONSTRAINT <tên_ràng_buộc> FOREIGN KEY (<thuộc_tính_khoá_ngoài>)
REFERENCES <tên_bảng_liên_kết> (<thuộc_tính_liên_kết>)

Ví dụ 4.1: Cho cơ sở dữ liệu quản lý **Thực tập** gồm các bảng dữ liệu sau:

+ Bảng **LopHoc** chứa thông tin về các lớp học (4.2)

MaLop	TenLop
L01	K7A-CNTT
L02	K7B-CNTT

Trong đó: MaLop là Mã lớp; TenLop là Tên lớp

+ Bảng **SinhVien** chứa danh sách sinh viên (4.3):

MaSV	HotenSV	NS	GT	Diachi	MaLop
08K7A1	Nguyễn Văn Dũng	09/11/1989	1	Hà Nội	L01

08K7A2	Lê Ngọc Dương	01/09/1989	1	Bắc Giang	L01
08K7A3	Trần Thị Hồng	17/06/1989	0	Bắc Kạn	L02

Trong đó: MaSV là "Mã số sinh viên"; HoTenSV là "Họ tên sinh viên"; NS là "Ngày sinh"; DiaChi là "Địa chỉ" của sinh viên; GT là "Giới tính";

+ Bảng **DeTai** chứa danh sách các đề tài thực tập (4.4):

MaDT	TenDT	GVHD	KP
DT01	Pháp triển ứng dụng WEB	Nguyễn Hồng An	2000000
DT02	Xây dựng thư viện điện tử	Trần Văn Lâm	2500000
DT03	Truy vấn dữ liệu Multimedia	Ngô Hải Long	3000000

Trong đó: MaDT là "Mã đề tài"; TenDT là "Tên đề tài"; GVHD là "Họ tên giáo viên hướng dẫn đề tài"; KP là "Kinh phí".

+ Bảng **SV_DeTai** chứa thông tin về tình hình thực tập của sinh viên (4.5):

MaSV	MaDT	NTT	KQ
08K7A1	DT01	Công ty phần mền CNN	8
08K7A2	DT02	Khoa CNTT	7
08K7A3	DT03	Công ty phần mềm Thanh Niên	9
08K7A1	DT02	Trung tâm học liệu TN	8

Trong đó: MaDT là "Mã đề tài"; NTT là "Nơi sinh viên đến thực tập"; KQ là "Kết quả thực tập"

Yêu cầu: Hãy tạo cấu trúc ba bảng dữ liệu trên.

Câu lệnh tạo bảng của SQL:

```
Create table LopHoc(
      MaLop   char(6)  Primary Key,
      TenLop  Varchar(35) Not Null)
Create table SinhVien(
      MaSV    Varchar(30)  Primary Key,
      HoTenSV  Varchar(35) Not Null,
      NS      Datetime     Not Null,
```

```
	GT		Bit		Not Null
	DiaChi	Varchar(100) Not Null,
	MaLop	Char(6),
	Constraint MaLop_FK Foreign key (MaLop) References
	LopHoc(MaLop))
Create table DeTai(
	MaDT	Char(9)		Primary key,
	TenDT	Varchar(100) Not Null,
	GVHD	Varchar(30)	Not Null,
	KP	SmallMoney)
Create table SV_DeTai(
	MaDT	Char(9)		Primary key,
	MaSV	Char(10),
	NTT	Varchar(100)	Not Null,
	KQ	Numeric(5,2)	Not Null,
	Constraint MaDT_FK Foreign key (MaDT) References
	DeTai(MaDT),
	Constraint MaSV_FK Foreign key (MaSV) References
	SinhVien(MaSV))
```

b) Thêm một cột

ALTER TABLE <tên _bảng>

ADD <tên_cột > <kiểu_dữ_liệu> [<ràng_buộc_cột>]

Ví dụ 4.2: Thêm vào bảng SinhVien cột số điện thoại

ALTER TABLE SinhVien

ADD SoDT char(10)

c) Xoá cột

ALTER TABLE <tên _bảng>

DROP COLUMN <tên_cột>

Ví dụ 4.3: Câu lệnh sau sẽ xoá cột số điện thoại trong bảng Sinh viên

ALTER TABLE SinhVien

DROP COLUMN SoDT

d) Xoá ràng buộc

ALTER TABLE <tên_bảng>

DROP CONSTRAINT <tên_ràng_buộc>

Ví dụ 4.4: Câu lệnh sau sẽ xoá một ràng buộc khoá ngoài trong bảng **SV_DETAI**

ALTER TABLE **SV_DETAI** DROP CONSTRAINT MaSV_FK

4.2.3. Nhãm lÖnh thao t¸c d÷ liÖu

4.2.3.1 Lệnh truy vấn dữ liệu - SELECT

Việc truy cập và lấy các thông tin từ database được SQL cho phép thực hiện qua câu lệnh SELECT. Câu lệnh SELECT có phạm vi ứng dụng rất rộng, có thể truy cập dữ liệu từ một bảng (table), hay từ nhiều bảng.

Các từ khóa SELECT, FROM, WHERE được sử dụng để tạo nên một câu lệnh SELECT đơn giản nhất

Cú pháp tổng quát có dạng sau:

```
SELECT [ ALL | DISTINCT ] <danh_sách_chọn>
FROM  <danh_sách_bảng>
[ WHERE <điều_kiện >]
[ GROUP BY <danh_sách_cột> ]
[ HAVING <điều_kiện_nhóm> ]
[ ORDER BY <tên_cột> [ ASC | DESC ] [,..n]]
```

a) Mệnh đề **SELECT**

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các thuộc tính, các biểu thức xuất hiện trong bảng kết quả của câu truy vấn. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

S1: Chọn tất cả các cột trong bảng

Khi cần hiển thị tất cả các trường trong bảng, sử dụng ký tự * trong danh sách chọn thay vì phải liệt kê tất cả các cột. Trong trường hợp này, các cột được hiển thị trong bảng kết quả của câu truy vấn sẽ tuân theo thứ tự mà chúng xuất hiện trong bảng cấu trúc.

Ví dụ 4.7: Câu lệnh sau sẽ liệt kê danh sách các sinh viên

```
SELECT * FROM  SinhVien
```

Kết quả:

MaSV	HotenSV	NS	GT	Diachi	MaLop
08K7A1	Nguyễn Văn Dũng	09/11/1989	1	Hà Nội	L01
08K7A2	Lê Ngọc Dương	01/09/1989	1	Bắc Giang	L01
08K7A3	Trần Thị Hồng	17/06/1989	0	Bắc Kạn	L02

S2: Liệt kê tên cột trong danh sách chọn

Trong trường hợp cần chỉ định cụ thể các cột cần hiển thị trong kết quả truy vấn, ta chỉ định danh sách các tên cột trong danh sách chọn. Thứ tự của các cột trong kết quả truy vấn tuân theo thứ tự của các cột trong danh sách chọn.

Ví dụ 4.8: Liệt kê danh sách sinh viên gồm các thuộc tính sau: MaSV, HoTenSV, DiaChi

Câu lệnh SELECT MaSV, HoTenSV, DiaChi FROM SinhVien

Kết quả:

MaSV	HotenSV	Diachi
08K7A1	Nguyễn Văn Dũng	Hà Nội
08K7A2	Lê Ngọc Dương	Bắc Giang
08K7A3	Trần Thị Hồng	Bắc Kạn

Chú ý: Nếu truy vấn được thực hiện trên nhiều bảng và các bảng có các thuộc tính trùng tên thì tên của những thuộc tính này nếu xuất hiện trong câu truy vấn phải được viết dưới dạng:

<Tên_bảng>.<Tên_thuộc tính>

Ví dụ 4.9: Liệt kê mã, họ và tên, các sinh viên đã tham gia ít nhất một lần thực tập

```
SELECT  SV.MaSV, HoTenSV
FROM  SinhVien SV, SV_DeTai TT
```

WHERE

MaSV	HotenSV
08K7A1	Nguyễn Văn Dũng
08K7A2	Lê Ngọc Dương
08K7A3	Trần Thị Hồng

SV.MaSV=TT.MaSV

Kết quả:

S3: Hiển thị với việc thay đổi tiêu đề các cột

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các thuộc tính tương ứng trong bảng. Tuy nhiên, để tiêu đề trở thành thân thiện hơn, ta có thể đổi lại tên tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó là một chuỗi kỹ tự đặt trong dấu nháy kép " ".

<tên_thuộc tính > AS <tiêu_đề_cột>

Ví dụ 4.10: Cho biết mã và tên của các đề tài thực tập.

SELECT MaDT AS "Mã đề tài", TenDT AS "Tên đề tài"

FROM DeTai

Kết quả:

Mã đề tài	Tên đề tài
DT01	Pháp triển ứng dụng WEB
DT02	Xây dựng thư viện điện tử
DT03	Truy vấn dữ liệu Multimedia

S4: Hằng và biểu thức trong danh sách chọn

Ngoài danh sách thuộc tính, trong danh sách chọn của câu lệnh SELECT còn có thể sử dụng các biểu thức. Mỗi biểu thức trong danh sách chọn trở thành một cột trong kết quả truy vấn.

S5: Loại bỏ các bản ghi trùng nhau

Trong kết quả của truy vấn có thể xuất hiện các dòng dữ liệu trùng nhau. Để loại bớt các dòng này, ta chỉ định thêm từ khoá DISTINCT ngay sau từ khoá SELECT.

Ví dụ 4.11: Cho biết thông tin về mã của các sinh viên đã tham gia thực tập.

```
SELECT DISTINCT MaSV  FROM SV_DeTai
```

Kết quả (không tồn tại các bản ghi trùng nhau):

MaSV
08K7A1
08K7A2
08K7A3

Nếu ta thực hiện câu lệnh không có tuỳ chọn DISTINCT như sau:

```
SELECT  MaSV  FROM SV_DeTai
```

Thì kết quả tồn tại cả các bản ghi trùng nhau:

b) Mệnh đề **FROM**

Mệnh đề FROM nhằm chỉ định các bảng cần truy xuất dữ liệu có liên quan đến câu truy vấn. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn. Tên của các bảng và các khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ 4.5: Câu lệnh dưới đây hiển thị mã và tên của các sinh viên.

SELECT MaSV, HoTenSV

FROM SinhVien

Kết quả:

MaSV	HotenSV
08K7A1	Nguyễn Văn Dũng
08K7A2	Lê Ngọc Dương
08K7A3	Trần Thị Hồng

Chú ý: Ta có thể sử dụng các bí danh cho các bảng hay khung nhìn trong câu lệnh truy vấn với cú pháp sau: <Tên_bảng> <Tên_bí_danh>.

Ví dụ 4.6: Câu lệnh sau gán bí danh là SV cho bảng SinhVien

SELECT * FROM SinhVien SV

c) Mệnh đề điều kiện **WHERE**

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một

biểu thức logíc và những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

Ví dụ 4.12: Câu lệnh dưới đây hiển thị mã số của các sinh viên đã thực tập đề tài có mã ‘DT02’

```
SELECT MaSV

FROM    SV_DeTai

WHERE MaDT='DT02'
```

Kết quả:

MaSV
08K7A1
08K7A2

Trong mệnh đề WHERE thường sử dụng

- Các toán tử kết hợp điều kiện (AND, OR)
- Các toán tử so sánh.
- Toán tử phạm vi và toán tử tập hợp
- Các giá trị NULL

S1: Các toán tử so sánh

Toán tử	Ý nghĩa
= (Equals)	Ngang bằng
> (Greater Than)	Lớn hơn
< (Less Than)	Nhỏ hơn
>= (Greater Than or Equal To)	Lớn hơn hoặc bằng
<= (Less Than or Equal To)	Nhỏ hơn hoặc bằng
<> (Not Equal To)	Không bằng
!= (Not Equal To)	Không bằng
!< (Not Less Than)	Không nhỏ hơn
!> (Not Greater Than)	Không lớn hơn

S2: To¸n tö ph¹m vi (Range Operator):

[NOT] BETWEEN a AND b

Toán tử này dùng để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử [NOT] BETWEEN như sau:

Cách sử dụng	Ý nghĩa
giá_tri BETWEEN a AND b	a ≤ giá_trị ≤ b
giá_tri NOT BETWEEN a AND b	(giá_trị < a) AND (giá_trị) > b

Câu lệnh dưới đây cho biết danh sách các đề tài có kinh phí nằm trong khoảng từ 2,5 đến 3 triệu đồng

SELECT *

FROM DeTai

WHERE KP Between 2500000 And 3000000

Kết quả:

MaDT	TenDT	GVHD	KP
DT02	Xây dựng thư viện điện tử	Trần Văn Lâm	2500000
DT03	Truy vấn dữ liệu Multimedia	Ngô Hải Long	3000000

Câu lệnh dưới đây cho biết danh sách các đề tài có kinh phí không nằm trong khoảng từ 2,5 đến 3 triệu đồng

SELECT *

FROM DeTai

WHERE KP NOT Between 2500000 And 3000000

Kết quả:

MaDT	TenDT	GVHD	KP
DT01	Pháp triển ứng dụng WEB	Nguyễn Hồng An	2000000

S3: Toán tử tập hợp

IN và NOT IN

Toán tử IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu SELECT là một danh sách các giá trị. Sau IN (hoặc NOT IN) có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ 4.12: Để biết danh sách các đề tài có kinh phí bằng 2 hoặc 3 triệu đồng thay vì sử dụng câu lệnh:

SELECT * FROM DeTai

WHERE KP =2000000 OR KP=3000000

Ta có thể sử dụng câu lệnh

SELECT * FROM DeTai WHERE KP IN (2000000,3000000)

Kết quả:

MaDT	**TenDT**	**GVHD**	**KP**
DT01	Pháp triển ứng dụng WEB	Nguyễn Hồng An	2000000
DT03	Truy vấn dữ liệu Multimedia	Ngô Hải Long	3000000

S4: Toán tử LIKE và các ký tự đại diện

Toán tử LIKE (hoặc NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. Chúng thường kết hợp với các ký tự đại diện sau đây:

Dấu phần trăm (%): Chỉ một chuỗi các ký tự bất kỳ.

Dấu gạch dưới (_): Chỉ một ký tự bất kỳ

Ví dụ 4.13: Cho biết danh sách các sinh viên có họ tên bắt đầu là ký tự 'L'

```
SELECT *  FROM SinhVien
WHERE  HoTenSV  Like  'L%'
```

Kết quả:

MaSV	HotenSV	NS	GT	Diachi	MaLop
08K7A2	Lê Ngọc Dương	01/09/1989	1	Bắc Giang	L01

Ví dụ 4.14: Cho biết mã, tên, địa chỉ của các sinh viên có họ tên kết thúc bằng chữ ‘ng’

```
SELECT *  FROM SinhVien
WHERE  HoTenSV  Like  '%ng'
```

Kết quả:

MaSV	HotenSV	Diachi
08K7A1	Nguyễn Văn Dũng	Hà Nội
08K7A2	Lê Ngọc Dương	Bắc Giang
08K7A3	Trần Thị Hồng	Bắc Kạn

S5: Giá trị NULL

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không ta sử dụng cách viết:

WHERE tên_cột IS NULL hoặc WHERE tên_cột IS NOT NULL

d) Sắp xếp kết quả truy vấn

Mặc định các dòng dữ liệu trong kết quả của câu truy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục). Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT.

Sau ORDER BY là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng dần.

Ví dụ 4.15: Câu lệnh dưới đây hiển thị danh sách các đề tài và sắp xếp theo chiều giảm dần kinh phí.

```
SELECT * FROM DeTai
ORDER BY KP DESC
```

Kết quả:

MaDT	TenDT	GVHD	KP
DT03	Truy vấn dữ liệu Multimedia	Ngô Hải Long	3000000
DT02	Xây dựng thư viện điện tử	Trần Văn Lâm	2500000
DT01	Pháp triển ứng dụng WEB	Nguyễn Hồng An	2000000

Chú ý: Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo chiều từ trái qua phải.

Ví dụ 4.16: Liệt kê danh sách sinh viên và sắp xếp theo tên sinh viên theo Alphabet (tăng dần), nếu trùng tên thì sắp theo giới tính.

```
SELECT * FROM SinhVien
ORDER BY  HoTenSV, GT
```

e) Phép kết nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép kết nối

Để thực hiện được một phép nối, cần phải xác định được những yếu tố sau:

- Những cột nào cần hiển thị trong kết quả truy vấn.
- Những bảng nào có tham gia vào truy vấn.
- Điều kiện để thực hiện phép nối giữa các bảng dữ liệu là gì?

Trong các yếu tố kể trên, việc xác định chính xác điều kiện để thực hiện phép nối giữa các bảng đóng vai trò quan trọng nhất. Trong đa số các trường hợp, điều kiện của phép nối được xác định nhờ vào mối quan hệ giữa các bảng

cần phải truy xuất dữ liệu. Thông thường, đó là điều kiện bằng nhau giữa khoá chính và khoá ngoài của hai bảng có quan hệ với nhau.

Ví dụ 4.17: Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: Mã sinh viên, họ tên, mã lớp và tên lớp

```
SELECT MaSV, HoTenSV, LopHoc.MaLop, TenLop
FROM SinhVien , LopHoc
WHERE SinhVien.MaLop = LopHoc.MaLop
```

Kết quả:

MaSV	HotenSV	MaLop	TenLop
08K7A1	Nguyễn Văn Dũng	L01	K7A-CNTT
08K7A2	Lê Ngọc Dương	L01	K7A-CNTT
08K7A3	Trần Thị Hồng	L02	K7B-CNTT

Trong câu lệnh trên, các bảng tham gia vào truy vấn bao gồm: SinhVien và LopHoc. Điều kiện để thực hiện phép kết nối giữa hai bảng là điều kiện sau: SinhVien.MaLop = LopHoc.MaLop

Chú ý: - Tên của một số cột nào đó trong các bảng có tham gia vào truy vấn. Nếu tên cột trong các bảng trùng tên nhau thì tên cột phải được viết dưới dạng: ***Tên_bảng.tên_cột***

- Dấu sao (*) được sử dụng trong danh sách chọn khi cần hiển thị tất cả các cột của các bảng tham gia truy vấn.

-Trong trường hợp cần hiển thị tất cả các cột của một bảng nào đó, ta sử dụng cách viết: ***tên_bảng.****

Ví dụ 4.18 : Liệt kê danh sách các sinh viên đã tham gia thực tập đề tại có mã số là ‘DT02’

```
SELECT SinhVien.*
FROM SinhVien , SV_DeTai
WHERE SinhVien.MaSV = SV_DeTai.MaSV AND MaDT=’DT02’
```

Kết quả

MaSV	**HotenSV**	**NS**	**GT**	**Diachi**	**MaLop**
08K7A1	Nguyễn Văn Dũng	09/11/1989	1	Hà Nội	L01

08K7A2	Lê Ngọc Dương	01/09/1989	1	Bắc Giang	L01

f) Thống kê dữ liệu với **GROUP BY**

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình…

Các hàm nhóm được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm nhóm dưới đây:

Hàm nhóm	Chức năng
SUM(Tên_thuộc_tính|biểu_thức)	Tính tổng các giá trị
AVG(Tên_thuộc_tính|biểu_thức)	Tính trung bình của các giá trị
COUNT(Tên_thuộc_tính|biểu_thức)	Đếm số các giá trị trong biểu thức
COUNT(*)	Đếm số các dòng được chọn
MAX(Tên_thuộc_tính|biểu_thức)	Tính giá trị lớn nhất
MIN(Tên_thuộc_tính|biểu_thức)	Tính giá trị nhỏ nhất

Trong đó:

Hàm SUM, AVG chỉ làm việc với các biểu thức số

Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.

Hàm COUNT(*) không bỏ qua các giá trị NULL

S1: Thống kê trên toàn bộ dữ liệu

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm nhóm trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bất kỳ một tên cột hay biểu thức nào ngoài các hàm nhóm.

Ví dụ 4.19 : Để tính trung bình kinh phí của tất cả các đề tài ta sử dụng câu lệnh như sau:

```
SELECT AVG(KP) AS TBKP  FROM  DeTai
```

Kết quả:

TBKP
2500000

S2: Thống kê dữ liệu trên các nhóm

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu, ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu thành các nhóm riêng biệt. Các hàm nhóm được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo từng nhóm dữ liệu.

Ví dụ 4.20: Câu lệnh dưới đây cho biết sĩ số sinh viên của mỗi lớp

```
SELECT LopHoc.MaLop, TenLop, COUNT(MaSV) AS SiSo
FROM   LopHoc, SinhVien
WHERE  LopHoc.MaLop = SinhVien. MaLop
GROUP BY LopHoc.MaLop, TenLop
```

Kết quả:

MaLop	TenLop	SiSo
L01	K7A-CNTT	2
L02	K7B-CNTT	1

Chú ý:

Biểu thức nào điều khiển việc phân nhóm dữ liệu thì các biểu thức đó phải được liệt kê sau mệnh đề GROUP BY.

Trong trường hợp danh sách chọn của câu lệnh SELECT có các hàm nhóm và những biểu thức hoặc thuộc tính không phải là đối số của các hàm nhóm thì những biểu thức hoặc những thuộc tính này phải được liệt kê đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Ví dụ 4.21: Dưới đây là một câu lệnh sai do thiếu trường TenLop sau mệnh đề GROUP BY

```
SELECT   LopHoc.MaLop, TenLop, COUNT(MaSV) AS SiSo
FROM     LopHoc, SinhVien
WHERE    LopHoc.MaLop = SinhVien. MaLop
GROUP BY LopHoc.MaLop
```

g) Mệnh đề điều kiện nhóm **HAVING**

Mệnh đề HAVING là mệnh điều kiện tác động trên các nhóm dữ liệu. Mệnh đề HAVING luôn sử dụng kết hợp với mệnh đề GROUP BY.

Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được phép sử dụng các hàm nhóm trong khi HAVING lại cho phép sử dụng các hàm nhóm trong điều kiện của mình.

Ví dụ 4.22: Cho biết các lớp có sĩ số >=50 học sinh

```
SELECT      LopHoc.MaLop, TenLop, COUNT(*) AS SiSo
FROM        LopHoc, SinhVien
WHERE       LopHoc.MaLop = SinhVien. MaLop
GROUP BY  LopHoc.MaLop, TenLop
HAVING    COUNT(*)>=50
```

h) Truy vấn con (Subquery)

Truy vấn con là một câu lệnh SELECT được lồng vào bên trong một câu lệnh SELECT, INSERT, UPDATE, DELETE hoặc bên trong một truy vấn con khác. Loại truy vấn này được sử dụng để biểu diễn cho những truy vấn trong đó điều kiện của câu truy vấn dữ liệu này cần phải sử dụng đến kết quả của một truy vấn khác.

Ví dụ 4.23 :Đưa ra danh sách các sinh viên đã thực hiện đề tài có mã là ‘DT02’

```
SELECT  SV.*
FROM SinhVien SV, SV_DeTai   TT
WHERE  SV.MaSV=TT.MaSV  AND   MaDT=’DT02’
```

Hoặc có thể viết như sau:

```
SELECT *
FROM SinhVien
WHERE MaSV   IN    (SELECT MaSV
```

```
                    FROM SV_DeTai
                WHERE MaDT = 'DT02')
```

Ví dụ 4.24: Cho biết các sinh viên đã thực tập đề tài có tên đề tài là ‘Truy vấn dữ liệu Multimedia’

```
SELECT *
FROM SinhVien
WHERE MaSV  IN (SELECT MaSV
                FROM SV_DeTai
                WHERE MaDT=(SELECT MaDT
                            FROM DeTai
                            WHERE
                                    TenDT= ’Truy vấn dữ liệu
```

Multimedia’))

Ví du 4.25: Câu lệnh sau đây cho biết danh sách các đề tài chưa có sinh viên thực tập.

```
SELECT *
FROM  DeTai
WHERE MaDT  NOT  IN      (SELECT  DICTINCT  MaDT
                          FROM SV_DeTai)
```

Ví du 4.26: Cho biết danh sách các đề tài có kinh phí lớn nhất?

```
SELECT  *
FROM  DeTai
WHERE  KP = (SELECT Max(KP) FROM  SV_DeTai)
```

4.2.3.2 Nhập dữ liệu.

Dữ liệu trong các bảng được thể hiện dưới các dòng (bản nghi). Để nhập thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT với cú pháp sau:

```
INSERT INTO <tên_bảng>[(<danh_ sách_tên_cột>)]
VALUES (<danh_sách_các_ giá_ trị>)
```

Trong câu lệnh INSERT, danh_sách_tên_cột ngay sau tên bảng không cần thiết phải chỉ định. Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo đúng thứ tự của các trường như khi bảng được định nghĩa.

Ví dụ 4.27: Câu lệnh dưới đây sẽ bổ sung thêm một lớp học vào bảng lớp

```
INSERT INTO  LopHoc
```

VALUES (‘L03’, ‘K7C-KT’)

Ngược lại <danh_ sách_tên_cột> được chỉ định sau tên bảng thì <danh_sách_các_ giá_ trị> phải tương ứng với thứ tự của các cột có trong danh sách.

Ví dụ:

INSERT INTO LopHoc(TenLop)
VALUES (‘K7D-ĐTVT’, ‘L04’)

Ghi chú: Trong trường hợp chỉ nhập giá trị cho một số cột trong bảng, ta phải chỉ định danh sách các cột cần nhập dữ liệu ngay sau tên bảng. Khi đó các cột không được nhập dữ liệu sẽ nhận giá trị mặc định (nếu có) hoặc nhận giá trị NULL (nếu cột cho phép nhận giá trị NULL). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

4.2.3.3. Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh này có cú pháp như sau:

UPDATE <tên_bảng>
SET <tên_cột_1>=<biểu_thức_1> [, …., <tên_cột_k> = <biểu_thức_k>]
[WHERE <điều_kiện>]

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định danh sách tên cột và biểu thức tương ứng sau từ khoá SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng để chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nêu không chỉ định, phạm vi tác động của câu lệnh là toàn bộ các dòng trong bảng)

Ví dụ 4.28: Cập nhật lại địa chỉ của sinh viên có mã là 08K7A1 thành Thái Nguyên

UPDATE SinhVien
SET DiaChi = 'Thái Nguyên' WHERE MaSV ='08K7A1'.

Ví dụ 4.29: Cập nhhật lại kết quả thực tập của các sinh viên đã thực tập đề tài có tên đề tài là ‘Truy vấn dữ liệu Multimedia’.

```
UPDATE  SV_DeTai
SET KQ = 10  WHERE MaDT=(SELECT MaDT
                         FROM DeTai
                         WHERE
                         TenDT='Truy vấn dữ liệu Multimedia').
```

4.2.3.4 Xo¸ d÷ liÖu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE . Cú pháp của câu lệnh như sau:

```
DELETE FROM <tên_bảng>
[WHERE <điều_kiện>]
```

Trong đó, tên của bảng cần xoá dữ liệu được chỉ định sau DELETE FROM. Mệnh đề WHERE trong câu lệnh được sử dụng để chỉ định điều kiện đối với các dòng dữ liệu cần xoá. Nếu câu lệnh DELETE không có mệnh đề WHERE thì toàn bộ các dòng trong bảng đều bị xoá. Mệnh đề FROM chỉ định danh sách các bảng có dữ liệu liên quan đến việc xoá dữ liệu.

+Ví dụ 4.30: Câu lệnh sau sẽ xoá khỏi bảng SinhVien những sinh viên có địa chỉ ở Hà nội

```
DELETE FROM SinhVien
WHERE DiaChi = 'Hà nội'
```

Ví dụ 4.31: Xoá khỏi bảng sinh viên những sinh viên chưa từng tham gia bất kỳ một đề tài thực tập nào?

```
DELETE FROM SinhVien
WHERE MaSV  NOT IN (SELECT  DISTINCT MaSV
                    FROM SV_DeTai )
```

+ Xoá tất cả các thông tin về tình hình thực tập của sinh viên

```
DELETE FROM  SV_DeTai
```

BÀI TẬP CÂU HỎI

1. Cho CSDL gồm 2 quan hệ

CC(MSNCC,TEN_CC,DCCC) và MH(MSNCC,MSMH)

Trong đó:

MSNCC: Mã số người cung cấp.

TEN_CC: Tên người cung cấp

DCCC: Địa chỉ cung cấp

MSMH: Mã số mặt hàng

Hãy cho mỗi quan hệ 5 bộ dữ liệu

Hãy biểu diễn các yêu cầu sau đây bằng ngôn ngữ SQL.

a. Tìm mã số người đã cung cấp

Q1: ít nhất một mặt hàng

Q2: không cung cấp mặt hàng nào

Q3: cung cấp mặt hàng có MSMH là 15

Q4: cung cấp ít nhất một mặt hàng nhưng không có mặt hàng có mã số là 15

b. Mặt hàng có mã số là 12, 13, 15 dược cung cấp bởi các nhà cung cấp địa chỉ nào?

c. Lập danh sách gồm các cột MSNCC, TEN_CC,MSMH) từ cơ sở dữ liệu trên

2. Cho CSDL gồm các quan hệ sau:

DAIHOC(TENTRUONG,HIEUTRUONG,DIACHI)

KHOA(TENTRUONG,MSKHOA,TENKHOA,SOSINHVIEN)

SINHVIEN(TENTRUONG,MSKHOA,MSSV,TENSV,DIACHISV)

trong đó:

SOSINHVIEN: số lượnglà sinh viên

MSKHOA: mã số khoa

TENSV: tên sinh viên

DIACHISV: địa chỉ của sinh viên (hiểu là quê quán)

Hãy cho mỗi quan hệ 5 bộ dữ liệu

Biểu diễn các câu hỏi sau đây bằng ngôn ngữ SQL

a/ Trường Đại học nào có khoa TINHOC

b/ Tổng số sinh viên học ở tất cả các trường đại học.

c/ Sinh viên nào học tại quê nhà (giả sử lấy tên tỉnh, thành phố)

d/ Khoa nào của trường có số sinh viên cao nhất?

e/ Cho biết tên hiệu trưởng của các trường có khoa TINHOC

3. Cho CSDL với các quan hệ:

NHANVIEN(MSNV,TENNHANVIEN,MSCOQUAN,CONGVIEC, THUTRUONG,LUONG)

COQUAN(MSCOQUAN,TENCOQUAN,DIACHI)

Hãy cho mỗi quan hệ 5 bộ dữ liệu

Biểu diễn bằng ngôn ngữ SQL, và đại số quan hệ yêu cầu sau đây:

Q1: Tìm tên những nhân viên ở cơ quan có mã số là 50

Q2: Tìm mã số tất cả các cơ quan từ quan hệ NHANVIEN

Q3: Tìm tên các nhân viên cơ quan có mã số là 15, 20, 25

Q4: Tìm tên những người làm việc ở Đồ Sơn

4. Cho cơ sở dữ liệu quản lý dự án gồm các bảng dữ liệu sau:

+ Bảng NHANVIEN chứa danh sách các nhân viên gồm các thuộc tính sau:

Tên Thuộc tính	Giải thích
MaNV	Mã nhân viên
Hoten	Họ tên nhân viên
Ngaysinh	Ngày sinh
GT	Giới tính

+ Bảng DU_AN chứa thông tin về các dự án gồm có các thuộc tính sau:

Tên Thuộc tính	Giải thích
MaDA	Mã dự án
TenDA	Tên dự án
NganSach	Ngân sách

+ Bảng THAMGIA ghi danh sách sinh viên đăng ký tham gia dự án

Tên Thuộc tính	Giải thích
MaDA	Mã dự án
MaNV	Mã nhân viên
TGBD	Thời gian bắt đầu
TGKT	Thời gian kết thúc

Biểu diễn bằng ngôn ngữ SQL, và đại số quan hệ các yêu cầu sau đây:

a/ Đưa ra danh sách các nhân viên ?

b/ Đưa ra danh sách các nhân viên có giới tính bằng 1 ?

c/ Đưa ra danh sách các dự án có ngân sách lớn nhất ?
d/ Cho biết mỗi nhân viên đã tham gia tổng số bao nhiêu dự án ?
e/ Cho biết mỗi dự án có tổng số bao nhiêu nhân viên tham gia ?
f/ Cho biết mã và tên của các dự án có số tổng số nhân viên tham gia >=10 người ?
g/ Cho biết mã và tên của các nhân viên đã tham gia dự án có tên dự án là 'Dự án nước sạch nông thôn' ?
h/ Đưa ra mã và tên các dự án mà nhân viên có mã là NV01 đã tham gia ?
i/ Đưa ra danh sách các nhân viên chưa tham gia bất kỳ dự án nào ?
k/ Cho biết thông tin về các nhân viên đã tham gia ít nhất một dự án ?
m/ Đưa ra danh sách các dự án có sự sắp xếp giảm dần theo ngân sách ?
5. Cho cơ sở dữ liệu gồm các bảng dữ liệu sau:

+ Bảng NSX (nước sản xuất)

Tên Thuộc tính	Giải thích
MaNSX	Mã nhà sản xuất
TenNSX	Tên nhà sản xuất

+ Bảng SANPHAM (sản phẩm)

Tên Thuộc tính	Giải thích
MaSP	Mã sản phẩm
TenSP	Tên sản phẩm
DVT	Đơn vị tính
NgaySX	Ngày sản xuất
SoLuong	Số lượng
ChungLoai	Chủng loại
MaNSX	Mã nhà sản xuất

Biểu diễn bằng ngôn ngữ SQL, và đại số quan hệ các yêu cầu sau đây:

a) Cho biết mã và tên của nhà sản xuất đã sản xuất sản phẩm có tên là 'Máy lọc nước'?

b) Cho biết mã, tên, số lượng, ngày sản xuất của các sản phẩm do nhà sản xuất có mã là 'N01' đã sản xuất?

c) Hãy tổng hợp thông tin về từng loại sản phẩm của mỗi nhà sản xuất(gồm các thuộc tính sau: MaSP, TenSP, DVT, TongSoLuong)?

d) Cho biết mỗi nhà sản xuất đã sản xuất tổng số bao nhiêu loại sản phẩm?

e) Cho biết danh sách các sản phẩm do nhà sản xuất có tên là ‘Panasonic’ đã sản xuất?

6. Cho cơ sở dữ liệu quản lý điểm gồm các bảng sau:

+ Bảng LopHoc gồm các thuộc tính

Tên Thuộc tính	Giải thích
MaLop	Mã lớp
TenLop	Tên lớp

+Bảng SinhVien gồm các thuộc tính

Tên Thuộc tính	Giải thích
MaSV	Mã sinh viên
HoTen	Họ và tên sinh viên
NS	Ngày sinh
GT	Giới tính
DC	Địa chỉ
MaLop	Mã lớp

+ Bảng MonHoc gồm các thuộc tính

Tên Thuộc tính	Giải thích
MaMon	Mã môn
TenMon	Tên môn
TC	Số tín chỉ

+Bảng Diem gồm các thuộc tính

Tên Thuộc tính	Giải thích
MaSV	Mã sinh viên
MaMon	Mã môn
Ky	Kỳ thi
DiemLan1	Điểm thi lần thi
DiemLan2	Điểm thi lần 2

Hãy biểu diễn các yêu cầu sau đây bằng ngôn ngữ SQL:

a) Cho biết danh sách các sinh viên có giới tính bằng 1 ?

b) Cho biết danh sách các sinh viên có địa chỉ ở Thái Nguyên?

c) Cho biết mã và tên và điểm thi lần 1 của các sinh viên đã học môn học có mã là MH01?

d) Cho biết mã và tên và điểm thi lần 1 của các sinh viên đã học môn học có tên môn là Cơ sở dữ liệu?

e) Cho biết danh sách các sinh viên phải thi lại môn học có tên môn là Cơ sở dữ liệu?

f) Cho biết sinh viên có mã là SV01 phải thi lại những môn học nào?

g) Cho biết điểm cao thi cao nhất của môn học Hệ quản trị cơ sở dữ liệu là bao nhiêu?

h) Cho biết những sinh viên đạt điểm thi cao nhất của môn Cơ sở dữ liệu?

Chương 5

TỐI ƯU HOÁ CÂU HỎI

Nói chung các ngông ngữ bậc cao, đòi hỏi thực hiện trong máy tính đều rất tốn kém thời gian. Do vậy trước khi thực hiện các câu hỏi thuộc các ngôn ngữ đó cần thiết phải biến đổi hợp lý để giảm thời gian tính toán. Việc làm đó tạm thời gọi là "tối ưu hoá ".

Trong chương này chủ yếu trình bày một vài phương pháp tối ưu hoá các biểu thức quan hệ, đặc biệt là xử lý biểu thức có liên quan tới phép kết nối và phép tích Đề - các. Sau đó sẽ trình bày chi tiết một phương pháp tối ưu ho một lớp phổ cập của các biểu thức quan hệ.

5.1 Các chiến lược tối ưu

1- Thức hiện phép chọn sớm nhất như có thể

Biến đổi câu hỏi để đưa phép chọn vào thực hiện trước nhằm làm giảm bớt kích cỡ của kết quả trung gian và do vậy chi phí phải trả giá cho việc truy nhập bộ nhớ thứ cấp cũng như lưu trữ của bộ nhớ chính sẽ nhỏ đi.

2-Tổ hợp những phép chọn với phép tích Đề-các thành phép kết nối

Như đã biết, phép kết nối, đặc biệt là phép kết nối bằng được thực hiện "rẻ" hơn là thực hiện các phép tích Đề-các trên cùng các quan hệ. Nếu kết quả của tích Đề-các R x S là đối số của phép chọn và phép chọn liên quan tới các phép so sánh giữa các thuộc tính của R và S thì rõ ràng phép tích Đề-các là phép kết nối.

3-Tổ hợp dãy các phép tính một ngôi thành một

Một dãy các phép một ngôi như phép chọn hoặc phép chiếu mà kết quả của chúng phụ thuộc vào các bộ của một quan hệ độc lập thì có thể nhóm các phép đó lại.

4-Tìm các biểu thức con chung trong một biểu thức

$F = F1 \times F2 \cup F1 \times F3 \Rightarrow F = F1 \times (F2 \cup F3)$

5-Xử lý độc lập các tệp trước khi xử lý chung CSDL

Có thể sắp xếp và thiết lập các tệp chỉ số cho từng quan hệ độc lập trước khi xử lý

6- Lựa chọn thứ tự thực hiện các phép toán.

Một khi cần chọn trình tự thực hiện các phép tính trong biểu thức hoặc chọn một trong hai đối số của một phép hai ngôi cần tính toán xem chi phí thực hiện các phép tính đó (thường là số phép tính, thời gian, dung tích bộ nhớ theo một tỷ lệ giữa kích cỡ các quan hệ…). Từ đó sẽ có được các chi phí phải trả cho các cách khác nhau để thực hiện các câu hỏi

5.2 Các phép biến đổi tương đương

1-Phép giao hoán

Nếu R_1 và R_2 là hai quan hệ, F là điều kiện trên các thuộc tính của R_1 và R_2 thì

$R_1 \underset{F}{\bowtie} R_2 \equiv R_2 \underset{F}{\bowtie} R_1$; $R_1 * R_2 \equiv R_2 * R_1$; $R_1 \times R_2 \equiv R_2 \times R_1$

2- Phép kết hợp

Nếu R_1, R_2 và R_3 là các quan hệ, F_1 và F_2 là các biểu thức điều kiện thì

$(R_1 \times R_2) \times R_3 \equiv R_1 \times (R_2 \times R_3)$;

$(R_1 \underset{F_1}{\bowtie} R_2) \underset{F_2}{\bowtie} R_3 \equiv R_1 \underset{F_1}{\bowtie} (R_2 \underset{F_2}{\bowtie} R_3)$

3- Xử lý dãy các phép toán chọn

$\sigma_{F1}(\sigma_{F2}(\sigma_{F3}...(\sigma_{F1n}(r))) \equiv \sigma_{F1 \wedge F2 \wedge F3 \wedge ... \wedge Fn}(r)$

4- Xử lý dãy các phép toán chiếu

Nếu có: $A_1A_2...A_n \subseteq B_1B_2...B_n$ Thì $\Pi_{A1A2...An}(\Pi_{B1B2....Bn}(R)) \equiv \Pi_{A1A2...An}(R)$

5- Giao hoán phép chọn và phép chiếu

$\sigma_F(\Pi_{A1A2...An}(R)) \equiv \Pi_{A1A2...An}(\sigma_F(R))$

6-Giao hoán giữa phép chọn và phép tích Đề -các

-Nếu các điều kiện chọn chỉ liên quan đến R1

$$\sigma_F(R_1 \times R_2) \equiv \sigma_{F1}(R_1) \times (R_2)$$

-Nếu $F = F_1 \wedge F_2$ trong đó F_1 chỉ liên quan đến R_1; F_2 chỉ liên quan đến R_2.

$$\sigma_F(R_1 \times R_2) \equiv \sigma_{F1}(R_1) \times \sigma_{F2}(R_2)$$

-Nếu F_1 chỉ liên quan đến R_1, F_2 liên quan đến cả R_1 và R_2

$$\sigma_F(R_1 \times R_2) \equiv \sigma_{F2}(\sigma_{F1}(R_1) \times R_2)$$

7- Giao hoán giữa phép chọn và một phép hợp

$$\sigma_F(R_1 \cup R_2) \equiv \sigma_F(R_1) \cup \sigma_F(R_2)$$

8- Giao hoán giữa một phép chọn và một phép trừ

$$\sigma_F(R_1 - R_2) \equiv \sigma_F(R_1) - \sigma_F(R_2)$$

9- Giao hoán giữa một phép chiếu và phép tích Đề các

Nếu các thuộc tính $A_1, A_2, ..., A_n \in R1$; $B_1, B_2, ..., B_n \in R_2$

Ta có: $\Pi_{A1A2...An,B1B2....Bn}(R_1 \times R_2) \equiv \Pi_{A1A2...An,}(R_1) \times \Pi_{B1B2...Bn}(R_2)$

10-Giao hoán giữa một phép chiếu và một phép hợp

$\Pi_{A1A2...An}(R_1 \cup R_2) \equiv \Pi_{A1A2...An,}(R_1) \cup \Pi_{A1A2....An}(R_2)$

Ví dụ: Cho cơ sở dữ liệu gồm các bảng dữ liệu sau

Bảng Công Ty (CONGTY) gồm các thuộc tính: Mã công ty (MaCongTy), Tên công ty (TenCongTy), Ngân sách (NganSach), Địa chỉ (DiaChi).

Bảng Hàng Hoá (HANGHOA) gồm các thuộc tính: Mã hàng (MaHang), Tên hàng (TenHang), Mầu sắc (Mau), Đơn vị tính (DonViTinh).

Bảng Cung Cấp hàng (CUNGCAP) gồm các thuộc tính: MaCongTy, MaHang, Số lượng (SoLuong), Đơn giá (DonGia).

Yêu cầu: Cho biết tên công ty cung cấp mặt hàng màu đỏ

1/ Dùng đại số quan hệ

$\Pi_{TenCongTy}(\sigma_{Mau =”đỏ”}(\sigma_{CongTy.MaCongTy = CungCap.MaCongTy}$

$(\sigma_{CungCap.MaHang=HangHoa.MaHang}$ (CONGTY x HANGHOA x CUNGCAP))))

-Đẩy phép chọn lên trước

$\Pi_{TenCongTy}(\sigma_{Mau =”đỏ”}(\sigma_{CongTy.MaCongTy = CungCap.MaCongTy}$

(CongTy x $\sigma_{CungCap.MaHang=HangHoa.MaHang}$(HangHoa x CungCap)))

-Chuyển phép chọn và phép tích Đề các thành phép kết nối

$\Pi_{TenCongTy}(\sigma_{Mau =”đỏ”}(\sigma_{CongTy.MaCongTy = CungCap.MaCongTy}$(CongTy x (HangHoa * CungCap)))

MaHang

-Chuyển phép chọn và phép tích Đề các thành phép kết nối

$\Pi_{TenCongTy}(\sigma_{Mau =”đỏ”}$(CongTy *(HangHoa * CungCap))

MaCongTy MaHang

-Đẩy phép chọn lên trước

$\Pi_{TenCongTy}$(CongTy * ($\sigma_{Mau =”đỏ”}$HangHoa) * CungCap))

MaHang MaCongTy

2/ Dùng SQL

-Không tối ưu:

```
SELECT TenCongTy
FROM Congty, HangHoa,CungCap
WHERE (CongTy.MaCongTy = CungCap.MaCongTy)  AND
      (CungCap.MaHang = HangHoa.MaHang)  AND (HangHoa. Mau = “Đỏ”)
```

- Tối ưu:

```
SELECT TenCongTy
FROM CongTy
WHERE  MaCongTy  IN ( SELECT MaCongTy
                      FROM CungCap
                      WHERE  MaHang  IN ( SELECT MaHang
                                          FROM HangHoa
                                          WHERE Mau = “Đỏ”)))
```

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Cơ sở dữ liệu

Giảng viên: ThS. Nguyễn Thị Kim Phụng
Email: phungntk@uit.edu.vn

Nội dung

1. Đại số quan hệ
2. Ngôn ngữ truy vấn SQL
3. Ràng buộc toàn vẹn

1. Đại số quan hệ

1. ĐẠI SỐ QUAN HỆ

- Là một mô hình toán học dựa trên lý thuyết tập hợp
- Đối tượng xử lý là các quan hệ trong cơ sở dữ liệu quan hệ
- Cho phép sử dụng các phép toán rút trích dữ liệu từ các quan hệ
- Tối ưu hóa quá trình rút trích dữ liệu
- Gồm có:
  - Các phép toán đại số quan hệ
  - Biểu thức đại số quan hệ

1. ĐSQH - Các phép toán ĐSQH, biểu thức ĐSQH

- Có năm phép toán cơ bản:
  - **Chọn** (σ) Chọn ra các dòng (bộ) trong quan hệ thỏa điều kiện chọn.
  - **Chiếu** (π) Chọn ra một số cột.
  - **Tích Descartes** ($\times$) Kết hai quan hệ lại với nhau.
  - **Trừ** (-) Chứa các bộ của quan hệ 1 nhưng không nằm trong quan hệ 2.
  - **Hội** ($\cup$) Chứa các bộ của quan hệ 1 và các bộ của quan hệ 2.
- Các phép toán khác:
  - **Giao ($\cap$), kết ($\bowtie$), chia (/ hay $\div$), đổi tên** ($\leftarrow$): là các phép toán không cơ bản (được suy từ 5 phép toán trên, trừ phép đổi tên).
- Biểu thức đại số quan hệ:
  - Là một biểu thức gồm các phép toán ĐSQH.
  - Biểu thức ĐSQH được xem như một quan hệ (không có tên)
  - Kết quả thực hiện các phép toán trên cũng là các quan hệ, do đó có thể kết hợp giữa các phép toán này để tạo nên các quan hệ mới!

1. ĐSQH - Phép chọn σ

Câu hỏi 1: Cho biết các nhân viên nam ?

- Biểu diễn cách 1 : **Cú pháp : $\sigma_{(\text{Điều kiện 1} \wedge \text{điều kiện 2} \wedge \ldots.)}$(Quan hệ)**

Câu hỏi 1: $\sigma_{\text{Phai='Nam'}}$(NhanVien)

- Ngoài ra, có thể biểu diễn cách 2:

Cú pháp : (Quan hệ: điều kiện chọn)

Câu hỏi 1: (NhanVien: Phai='Nam')

NHANVIEN

MANV	HOTEN	NTNS	PHAI
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam
NV002	Trần Đông Anh	01/08/1981	Nữ
NV003	Lý Phước Mẫn	02/04/1969	Nam

Kết quả phép chọn

NHANVIEN

MANV	HOTEN	NTNS	PHAI
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam
NV003	Lý Phước Mẫn	02/04/1969	Nam

1. ĐSQH - Phép chọn σ

Câu hỏi 2: Cho biết các nhân viên nam sinh sau năm 1975 ?

- Biểu diễn cách 1 :

Câu hỏi 2: $\sigma_{(Phai='Nam' \wedge Year(NTNS)>1975)}(NhanVien)$

- Biểu diễn cách 2:

Câu hỏi 2: $(NhanVien: Phai='Nam' \wedge Year(NTNS)>1975)$

NHANVIEN			
MANV	HOTEN	NTNS	PHAI
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam
NV002	Trần Đông Anh	01/08/1981	Nữ
NV003	Lý Phước Mẫn	02/04/1969	Nam

→ Kết quả phép chọn

NHANVIEN			
MANV	HOTEN	NTNS	PHAI

(không có bộ nào thỏa)

1. ĐSQH - Phép chiếu

Câu hỏi 3: Cho biết họ tên nhân viên và giới tính ?

- Biểu diễn cách 1 : **Cú pháp : $\pi_{\text{Cột1, cột2, cột 3, ...}}$ (Quan hệ)**

  Câu hỏi 3 : $\pi_{\text{HOTEN, PHAI}}$ (NhanVien)

- Ngoài ra, có thể biểu diễn cách 2:

  Cú pháp : Quan hệ [cột1,cột2,cột3,…]

  Câu hỏi 3: NhanVien [HoTen, Phai]

NHANVIEN

MANV	HOTEN	NTNS	PHAI
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam
NV002	Trần Đông Anh	01/08/1981	Nữ
NV003	Lý Phước Mẫn	02/04/1969	Nam

→ Kết quả phép chiếu

NHANVIEN

HOTEN	PHAI
Nguyễn Tấn Đạt	Nam
Trần Đông Anh	Nữ
Lý Phước Mẫn	Nam

1. ĐSQH - Phép chiếu

Câu hỏi 4: Cho biết họ tên và ngày tháng năm sinh của các nhân viên nam?

- Biểu diễn cách 1:

Bước 1: $Q \leftarrow \sigma_{(Phai='Nam')}(NhanVien)$

Bước 2: $\pi_{HOTEN, NTNS}(Q)$

Kết quả phép chọn (**còn gọi là biểu thức ĐSQH**) được đổi tên thành quan hệ Q

- Biểu diễn cách 2:

Câu hỏi 4: **(NhanVien: Phai='Nam') [HoTen, NTNS]**

NHANVIEN

MANV	HOTEN	NTNS	PHAI
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam
NV002	Trần Đông Anh	01/08/1981	Nữ
NV003	Lý Phước Mẫn	02/04/1969	Nam

Kết quả phép chiếu

NHANVIEN

HOTEN	NTNS
Nguyễn Tấn Đạt	10/12/1970
Lý Phước Mẫn	02/04/1969

1. ĐSQH - Phép tích Descartes ╳

Câu hỏi 5: Tính tích Descartes giữa 2 quan hệ nhân viên và phòng ban

Cú pháp : Quan-hệ-1 ╳ Quan-hệ-2 ╳ …Quan-hệ-k

Câu hỏi 5 được viết lại: NHANVIEN ╳ PHONGBAN

NHANVIEN

MANV	HOTEN	NTNS	PHAI	PHONG
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam	NC
NV002	Trần Đông Anh	01/08/1981	Nữ	DH
NV003	Lý Phước Mẫn	02/04/1969	Nam	NC

PHONGBAN

MAPH	TENPH	TRPH
NC	Nghiên cứu	NV001
DH	Điều hành	NV002

NHANVIEN X PHONGBAN

MANV	HOTEN	NTNS	PHAI	PHONG	MAPH	TENPH	TRPH
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam	NC	NC	Nghiên cứu	NV001
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam	NC	DH	Điều hành	NV002
NV002	Trần Đông Anh	01/08/1981	Nữ	DH	NC	Nghiên cứu	NV001
NV002	Trần Đông Anh	01/08/1981	Nữ	DH	DH	Điều hành	NV002
NV003	Lý Phước Mẫn	02/04/1969	Nam	NC	NC	Nghiên cứu	NV001
NV003	Lý Phước Mẫn	02/04/1969	Nam	NC	DH	Điều hành	NV002

1. ĐSQH - Phép kết ⋈ (Theta-Join)

Câu hỏi 6: Cho biết mã nhân viên, họ tên và tên phòng mà n/v trực thuộc.

-**Đặt vấn đề**: trở lại ví dụ 5, ta thấy nếu thực hiện phép tích Decartes NHANVIEN X PHONGBAN thì mỗi nhân viên đều thuộc 2 phòng (vì có tổng cộng là 2 phòng ban, nếu có 3, 4,…phòng ban thì số dòng cho một nhân viên trong NHANVIEN X PHONGBAN sẽ là 3, 4,..dòng.

- Thực tế mỗi nhân viên chỉ thuộc duy nhất 1 phòng ban do ràng buộc khóa ngoại (PHONG), do đó để lấy được giá trị MAPH đúng của mỗi nhân viên ➔ phải có điều kiện chọn:

NHANVIEN.PHONG = PHONGBAN.MAPH

biểu diễn phép chọn theo cách 2

((NHANVIEN X PHONGBAN) : NHANVIEN.PHONG=PHONGBAN.MAPH)

MANV	HOTEN	NTNS	PHAI	PHONG	MAPH	TENPH	TRPH
NV001	Nguyễn Tấn Đạt	10/12/1970	Nam	NC	NC	Nghiên cứu	NV001
NV002	Trần Đông Anh	01/08/1981	Nữ	DH	DH	Điều hành	NV002
NV003	Lý Phước Mẫn	02/04/1969	Nam	NC	NC	Nghiên cứu	NV001

1. ĐSQH - Phép kết ⋈ (Theta-Join)

- Cách 1: $\sigma_{\text{NHANVIEN.PHONG=PHONGBAN.MAPH}}$ (**NHANVIEN X PHONGBAN**)

- Cách 2:

(NHANVIEN ⨯ PHONGBAN): (NHANVIEN.PHONG=PHONGBAN.MAPH)

* Phép kết được định nghĩa là phép tích Decartes và có điều kiện chọn liên quan đến các thuộc tính giữa 2 quan hệ, cú pháp :

Quan-hệ-1 ⋈ Quan-hệ-2

Điều kiện kết

(Phép kết với đk tổng quát được gọi là θ-kết, θ có thể là ≠ , =, >, <, >=, <=. Nếu đk kết là phép so sánh = thì gọi là kết bằng)

➡ **Câu hỏi 6 viết lại cách 1:** ⋈

➡ **Câu hỏi 6 viết lại cách 2:**

⋈

1. ĐSQH - kết bằng, kết tự nhiên

Kết bằng:

Kết tự nhiên:

Nếu PHONG trong NHANVIEN được đổi thành MAPH thì ta bỏ đi 1 cột MAPH thay vì phải để MAPH=MAPH, lúc này gọi là phép kết tự nhiên (natural-join)

Hoặc viết cách khác: NHANVIEN * PHONGBAN

1. ĐSQH - Phép kết ⋈

Câu hỏi 7: Tìm họ tên các trưởng phòng của từng phòng ?

⋈

Câu hỏi 8: Cho lược đồ CSDL như sau:

TAIXE (MaTX, HoTen, NgaySinh, GioiTinh, DiaChi)

CHUYENDI (SoCD, MaXe, MaTX, NgayDi, NgayVe, ChieuDai, SoNguoi)

Cho biết họ tên tài xế, ngày đi, ngày về của những chuyến đi có chiều dài >=300km, chở từ12 người trở lên trong mỗi chuyến?

Cách 1: **Q** ← σ **(CHUYENDI)**

⋈

Kết quả:

Cách 2:

⋈

1. ĐSQH - 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 tài xế và số chuyến đi, mã xe mà tài xế đó lái (nếu có)

1. ĐSQH – left outer join (lấy hết tất cả bộ của quan hệ bên trái)

- TAIXE $\overset{matx}{\bowtie}$ CHUYENDI

Matx	Hoten	SoCD	Matx	Maxe
TX01	Huynh Trong Tao	CD01	TX01	8659
TX01	Huynh Trong Tao	CD03	TX01	8659
TX02	Nguyen Sang	CD02	TX02	7715
TX03	Le Phuoc Long	CD04	TX03	4573
TX04	Nguyen Anh Tuan	Null	Null	Null

{

Bộ của quan hệ TAIXE được thêm
Vào dù không phù hợp với kết quả
của quan hệ CHUYENDI

TAIXE

MaTX	Hoten
TX01	Huynh Trong Tao
TX02	Nguyen Sang
TX03	Le Phuoc Long
TX04	Nguyen Anh Tuan

CHUYENDI

SoCD	MaTX	MaXe
CD01	TX01	8659
CD02	TX02	7715
CD03	TX01	8659
CD04	TX03	4573

Tương tự right outer join và full outer join (lấy cả 2)

1. ĐSQH - Phép trừ, phép hội, phép giao tập hợp

- Tất cả các phép toán này đều cần hai quan hệ đầu vào tương thích khả hợp, nghĩa là chúng phải thoả:
  - Cùng số thuộc tính. Ví dụ: R và S đều có 2 thuộc tính.
  - Các thuộc tính `tương ứng ' có cùng kiểu.

R	
HONV	**TENNV**
Vuong	Quyen
Nguyen	Tung

S	
HONV	**TENNV**
Le	Nhan
Vuong	Quyen
Bui	Vu

Phép trừ: R - S

Phép hội: R ∪ S

Phép giao: R ∩ S

NHANVIEN (MaNV, HoTen, Phai, Luong,NTNS, Ma_NQL, MaPH)

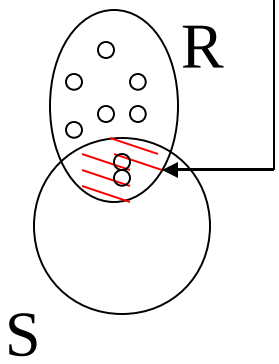
PHANCONG (MaNV, MaDA, ThoiGian)

1. ĐSQH - Phép trừ, phép hội, phép giao tập hợp

Phép trừ: $Q = R - S = \{ t/ t \in R \wedge t \notin S\}$

Phép hội: $Q = R \cup S = \{ t/ t \in R \vee t \in S\}$

Phép giao: $Q = R \cap S = R - (R - S) = \{ t/t \in R \wedge t \in S\}$



R	
HONV	TENNV
Vuong	Quyen
Nguyen	Tung

S	
HONV	TENNV
Le	Nhan
Vuong	Quyen
Bui	Vu

Kết quả phép trừ Q ={Nguyen Tung}

Kết quả phép hội Q ={Vuong Quyen, Nguyen Tung, Le Nhan, Bui Vu}

Kết quả phép giao Q ={Vuong Quyen}

Lưu ý **: Phép hội và phép giao có tính chất giao hoán**

1. ĐSQH - Phép trừ, phép hội, phép giao tập hợp

Câu hỏi 9: Cho biết nhân viên không làm việc ? **(Phép trừ)**

Cách 1:

Cách 2: **(NHANVIEN[MANV]) – (PHANCONG[MANV])**

Câu hỏi 10: Cho biết nhân viên được phân công tham gia đề án có mã số 'TH01' hoặc đề án có mã số 'TH02'? **(Phép hội)**

((PHANCONG: MADA='TH01')[MANV]) ∪ ((PHANCONG : MADA='TH02')[MANV])

Câu hỏi 11: Cho biết nhân viên được phân công tham gia cả 2 đề án 'TH01' và đề án 'TH02'? **(Phép giao)**

((PHANCONG : MADA='TH01')[MANV]) ∩ ((PHANCONG : MADA='TH02')[MANV])

1. ĐSQH - Phép chia tập hợp (/ hay ÷)

- Phép chia (R ÷ S) cần hai quan hệ đầu vào R, S thoả:
  - Tập thuộc tính của R là tập cha của tập thuộc tính S.
    Ví dụ: R có m thuộc tính, S có n thuộc tính : $n \subseteq m$

◆**Định nghĩa**:

R và S là hai quan hệ, R^+ và S^+ lần lượt là tập thuộc tính của R và S. Điều kiện $S^+ \neq \varnothing$ là ***tập con không bằng*** của R^+. Q là kết quả phép chia giữa R và S, $Q^+ = R^+ - S^+$

$$Q = R \div S = \{t / \forall s \in S, \ (t,s) \in R\}$$

$$T_1 \leftarrow \pi_{R^+ - S^+}(R)$$
$$T_2 \leftarrow \pi_{R^+ - S^+}((S \times T_1) - R)$$
$$T \leftarrow T_1 - T_2$$

1. ĐSQH - Phép chia tập hợp (/ hay ÷)

R=PHANCONG

MANV	MADA
001	TH001
001	TH002
002	TH001
002	TH002
002	DT001
003	TH001

S=DEAN

MADA
TH001
TH002
DT001

Kết quả Q

Q= PHANCONG/DEAN

MANV
002

Cho biết nhân viên làm việc cho tất cả các đề án ? (được phân công tham gia tất cả các đề án)

Hoặc viết Q= PHANCONG ÷DEAN

1. ĐSQH - Phép chia tập hợp (/ hay ÷)

R=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

S=MONHOC

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

Mahv
HV01
HV03

Q=KETQUA/MONHOC

$$KETQUA \leftarrow KETQUATHI[Mahv, Mamh]$$

$$MONHOC \leftarrow MONHOC[Mamh]$$

*** Viết cách khác**

KETQUATHI[Mahv,Mamh] /MONHOC[Mamh]

1. ĐSQH – Hàm tính toán trên 1 nhóm và tính toán trên nhiều nhóm (gom nhóm – group by)

- Các hàm tính toán gồm 5 hàm: avg(giá-trị), min(giá-trị), max(giá-trị), sum(giá-trị), count(giá-trị).

- Phép toán gom nhóm: (Group by)

$$G_1, G_2, \ldots, G_n \, \Im_{F_1(A_1), F_2(A_2), \ldots, F_n(A_n)} (E)$$

  - E là biểu thức đại số quan hệ
  - G_i là thuộc tính gom nhóm (nếu không có G_i nào=> không chia nhóm (1 nhóm), ngược lại (nhiều nhóm) => hàm F sẽ tính toán trên từng nhóm nhỏ được chia bởi tập thuộc tính này)
  - F_i là hàm tính toán
  - A_i là tên thuộc tính

1. ĐSQH – Hàm tính toán trên 1 nhóm và tính toán trên nhiều nhóm (gom nhóm – group by)

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

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

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

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

1. Ngôn ngữ truy vấn SQL

2. NGÔN NGỮ TRUY VẤN SQL

- Là ngôn ngữ chuẩn, có cấu trúc dùng để truy vấn và thao tác trên CSDL quan hệ.
- Câu truy vấn tổng quát:

SELECT [DISTINCT] danh_sách_cột | hàm

FROM danh sách các quan hệ (hay bảng, table)

[WHERE điều_kiện]

[GROUP BY danh_sách_cột_gom_nhóm]

[HAVING điều_kiện_trên_nhóm]

[ORDER BY cột1 ASC | DESC, cột2 ASC | DESC,…]

2. SQL

- Toán tử so sánh:
  - =,>,<,>=,<=,<>
  - BETWEEN
  - IS NULL, IS NOT NULL
  - LIKE (%,_)
  - IN, NOT IN
  - EXISTS, NOT EXISTS
  - SOME, ALL, ANY
- Toán tử logic: AND, OR.
- Các phép toán: +, - ,* , /
- Các hàm xử lý ngày (DAY( )), tháng (MONTH( )), năm (YEAR( ))

2. SQL

- 5 hàm: COUNT( ), SUM( ), MAX( ), MIN( ), AVG( )
- Phân loại câu SELECT: SELECT đơn giản, SELECT có mệnh đề ORDER BY, SELECT lồng (câu SELECT lồng câu SELECT khác), SELECT gom nhóm (GROUP BY), SELECT gom nhóm (GROUP BY)có điều kiện HAVING.

Bài tập: Cho lược đồ CSDL "quản lý đề án công ty" như sau

NHANVIEN (MaNV, HoTen, Phai, Luong,NTNS, Ma_NQL, MaPH)

PHONGBAN (MaPH, TenPH, TRPH)

DEAN (MaDA, TenDA, Phong, NamThucHien)

PHANCONG (MaNV, MaDA, ThoiGian)

NHANVIEN

MANV	HOTEN	NTNS	PHAI	MA_NQL	MaPH	LUONG
001	Vuong Ngoc Quyen	22/10/1957	Nu		QL	3.000.000
002	Nguyen Thanh Tung	09/01/1955	Nam	001	NC	2.500.000
003	Le Thi Nhan	18/12/1960	Nu	001	DH	2.500.000
004	Dinh Ba Tien	09/01/1968	Nam	002	NC	2.200.000
005	Bui Thuy Vu	19/07/1972	Nam	003	DH	2.200.000
006	Nguyen Manh Hung	15/09/1973	Nam	002	NC	2.000.000
007	Tran Thanh Tam	31/07/1975	Nu	002	NC	2.200.000
008	Tran Hong Minh	04/07/1976	Nu	004	NC	1.800.000

DEAN

MADA	TENDA	PHONG	NamThucHien
TH001	Tin hoc hoa 1	NC	2002
TH002	Tin hoc hoa 2	NC	2003
DT001	Dao tao 1	DH	2004
DT002	Dao tao 2	DH	2004

PHONGBAN

MAPH	TENPH	TRPH
QL	Quan Ly	001
DH	Dieu Hanh	003
NC	Nghien Cuu	002

PHANCONG

MANV	MADA	THOIGIAN
001	TH001	30,0
001	TH002	12,5
002	TH001	10,0
002	TH002	10,0
002	DT001	10,0
002	DT002	10,0
003	TH001	37,5
004	DT001	22,5
004	DT002	10,0
006	DT001	30,5
007	TH001	20,0
007	TH002	10,0
008	DT002	12,5

2. SQL – BETWEEN, ORDER BY, IS NULL

Câu hỏi 13: Sử dụng =,>,>=,… Danh sách các nhân viên sinh trong khoảng từ năm 1978 đến 1983?

➡ Select MaNV, HoTen From NhanVien
where Year(NTNS)>=1978 AND Year(NTNS)<=1983

Câu hỏi 14: Sử dụng BETWEEN, ORDER BY. Danh sách các nhân viên sinh trong khoảng từ năm 1978 đến 1983? Sắp xếp theo mức lương giảm dần.

➡ Select * From NhanVien where Year(NTNS) BETWEEN 1978 and 1983 ORDER BY Luong DESC

Câu hỏi 15: Sử dụng IS NULL. Cho biết những nhân viên không có người quản lý trực tiếp? (không chịu sự quản lý trực tiếp của người nào)

➡ Select MaNV, HoTen, NTNS, Ma_NQL from NhanVien where Ma_NQL is Null

2. SQL - SO SÁNH IN & NOT IN

Câu hỏi 16: Sử dụng Is Not Null. Cho biết những nhân viên có người quản lý trực tiếp?Thông tin hiển thị gồm: mã nhân viên, họ tên, mã người quản lý.

➡ Select MaNV, HoTen, Ma_NQL from NhanVien
where Ma_NQL is not Null

Câu hỏi 17: Sử dụng IN (so sánh với một tập hợp giá trị cụ thể). Cho biết họ tên nhân viên thuộc phòng ‘NC’ hoặc phòng ‘DH’?

➡ Select DISTINCT Hoten From NhanVien where MaPH in (‘NC’,’DH’)

Câu hỏi 18: Sử dụng IN (so sánh với một tập hợp giá trị chọn từ câu SELECT khác). Cho biết họ tên nhân viên thuộc phòng ‘NC’ hoặc phòng ‘DH’?

➡ Select Hoten from NhanVien where MaPH in (Select MaPH from PHONGBAN where MaPH=‘NC’ OR MaPH=‘DH’)

2. SQL – SO SÁNH IN & NOT IN

Câu hỏi 19 (tt): Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên đã tham gia đề án?

➡ Select MaNV, HoTen, NTNS from NhanVien
where MaNV in (Select MaNv From PhanCong)

Câu hỏi 20: Sử dụng NOT IN. Cho biết mã số, họ tên, ngày tháng năm sinh của những nhân viên không tham gia đề án nào?
Gợi ý cho mệnh đề NOT IN: thực hiện câu truy vấn "tìm nhân viên có tham gia đề án (dựa vào bảng PhanCong)", sau đó lấy phần bù.

➡ Select MaNV, HoTen, NTNS from NhanVien
where MaNV not in (Select MaNv From PhanCong)

Câu hỏi 21 (tt): Cho biết tên phòng ban không chủ trì các đề án triển khai năm 2005? *Gợi ý*: thực hiện câu truy vấn "tìm phòng ban chủ trì các đề án triển khai năm 2005", sau đó lấy phần bù.

➡ Select TenPH from PhongBan where MaPH not in (Select DISTINCT Phong from DEAN where NamThucHien=2005)

2. SQL – SO SÁNH LIKE

Câu hỏi 22: so sánh chuỗi = chuỗi. Liệt kê mã nhân viên, ngày tháng năm sinh, mức lương của nhân viên có tên "Nguyễn Tường Linh"?

```
Select MaNV, NTNS, Luong from NhanVien
where HoTen = 'Nguyễn Tường Linh'
```

Câu hỏi 23: Sử dụng LIKE (%: thay thế 1 chuỗi ký tự). Tìm những nhân viên có họ Nguyễn.

```
Select MaNV, HoTen from NhanVien where HoTen like 'Nguyễn %'
```

Câu hỏi 24 (tt): Tìm những nhân viên có tên Lan.

```
Select MaNV, HoTen from NhanVien where HoTen like '% Lan'
```

Câu hỏi 25 (tt): Tìm những nhân viên có tên lót là "Văn".

```
Select MaNV, HoTen from NhanVien where HoTen like '% Văn %'
```

Câu hỏi 26: Sử dụng LIKE (_: thay thế 1 ký tự bất kỳ). Tìm những nhân viên tên có tên 'Nguyễn La_' (ví dụ Lam, Lan)

```
Select MaNV, HoTen from NhanVien where HoTen like 'Nguyễn La_'
```

2. SQL – HÀM COUNT,SUM,MAX,MIN,AVG

a) Sử dụng các hàm COUNT, SUM, MIN, MAX, AVG trên 1 nhóm lớn (trên toàn bộ quan hệ):

- Câu hỏi 27: Tính số nhân viên của công ty.

```
Select COUNT(MaNV) as SoNV from NhanVien
```

- Câu hỏi 28: Tính số lượng nhân viên quản lý trực tiếp nhân viên khác.

```
Select COUNT (DISTINCT Ma_NQL) from NhanVien
```

- Câu hỏi 29: Tìm mức lương lớn nhất, mức lương trung bình, tổng lương của công ty.

```
Select MAX(Luong), AVG(Luong), SUM(Luong) from NhanVien
```

- Câu hỏi 30: Cho biết nhân viên có mức lương lớn nhất.

```
Select HoTen from NhanVien
Where Luong = (Select MAX(Luong) from NhanVien )
```

2. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 31: Cho biết nhân viên có mức lương trên mức lương trung bình của công ty.

```
Select HoTen from NhanVien where Luong > (Select
AVG(Luong) from NhanVien )
```

b) Sử dụng các hàm COUNT, SUM, MIN, MAX, AVG trên từng nhóm nhỏ: mệnh đề GROUP BY

– Chia các dòng thành các nhóm nhỏ dựa trên tập thuộc tính chia nhóm.

– Thực hiện các phép toán trên nhóm như: Count (thực hiện phép đếm), Sum (tính tổng), Min(lấy giá trị nhỏ nhất), Max(lấy giá trị lớn nhất), AVG (lấy giá trị trung bình).

2. SQL – MỆNH ĐỀ GROUP BY

Quan hệ NV

Q	S
a	10
a	2
b	9
b	5
c	10
c	8
c	6
c	4
c	10
d	16
d	18
d	50

nhóm

Chia các dòng thành các nhóm dựa trên tập thuộc tính chia nhóm

Q	Count(S)
a	2
b	2
c	5
d	3

Tương tự cho các hàm SUM, MIN, MAX, AVG

Các thuộc tính GROUP BY: Q

Câu SQL:

```
Select Q, count(S)
From NV
Group by Q
```

2. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 32: Cho biết số lượng nhân viên theo từng phái?

Do cột phái có 2 giá trị “nam” và “nữ”, trường hợp này ta chia bảng NhanVien thành 2 nhóm nhỏ. Thuộc tính chia nhóm là thuộc tính “Phai”.

➡ Select Phai, count(Manv) as SoNV from NhanVien
Group by Phai

Câu hỏi 33: Cho biết số lượng nhân viên theo từng phòng?

Do cột MaPH có 3 giá trị “NC” và “DH” và “QL”, trường hợp này ta chia bảng nhân viên thành 3 nhóm nhỏ. Thuộc tính chia nhóm là thuộc tính “MaPH”.

➡ Select MaPH, count(Manv) from NhanVien Group by MaPH

Tương tự: cho biết tổng lương của mỗi phòng, cho biết mức lương thấp nhất của từng phòng, mức lương cao nhất, mức lương trung bình của từng phòng

2. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 34: Cho biết tên phòng và số lượng nhân viên theo từng phòng?

Giống câu 29 nhưng bổ sung thêm bảng PhongBan để lấy tên phòng. Thuộc tính chia nhóm là (TenPH) thay cho MaPH.

➡ Select TenPH, count(Manv) as SoLuongNV
From NhanVien n, PhongBan p Where n.MaPh=p.MaPH
Group by TenPH

Câu hỏi 35: Với mỗi phòng, cho biết số lượng nhân viên theo từng phái?

Do cột MaPH có 3 giá trị "NC" và "DH" và "QL", mỗi phòng chia nhỏ theo từng phái: 2 nhóm "Nam" và "Nữ", trường hợp này ta chia bảng nhân viên thành 6 nhóm nhỏ. Như vậy, tập thuộc tính chia nhóm cho câu truy vấn là (Phong, Phai).

➡ Select MaPH, Phai, count(Manv) from NhanVien
Group by Phong, Phai

2. SQL – MỆNH ĐỀ GROUP BY

Câu hỏi 36: Đếm số đề án của từng nhân viên tham gia?

- Do cột MaNV có 7 giá trị "NV001",…"NV008" (không có nhân viên "005"), trường hợp này ta chia bảng PhanCong thành 7 nhóm nhỏ. Với mỗi nhóm nhỏ (MaNV), ta đếm số đề án (count(MADA)) tham gia. Thuộc tính chia nhóm là thuộc tính "MaNV".
- Tương tự: tính tổng số giờ làm việc của mỗi nhân viên (SUM), thời gian làm việc thấp nhất của mỗi nhân viên (MIN), thời gian làm việc lớn nhất của mỗi nhân viên (MAX), thời gian làm việc trung bình,…

➡ Select MaNV, count(MaDA) as SoDATG From PhanCong
Group by MaNV

Câu hỏi 37: Cho biết mã, tên nhân viên và số đề án mà n/v đã tham gia?

➡ Select n.MaNV, HoTen, count(MaDA) as SoDATG
From PhanCong pc, NhanVien n where pc.manv=n.manv
Group by MaNV, HoTen

2. SQL – MỆNH ĐỀ HAVING

- Lọc kết quả theo điều kiện, sau khi đã gom nhóm
- Điều kiện của HAVING là điều kiện về các hàm tính toán trên nhóm (Count, Sum, Min, Max, AVG) và các thuộc tính trong danh sách GROUP BY.

Câu hỏi 38: Cho biết những nhân viên tham gia từ 2 đề án trở lên?

```
Select MaNV, count(MaDA)  as SoDATG From  PhanCong
Group by MaNV
Having count(MaDA) >=2
```

Câu hỏi 39: Cho biết mã phòng ban có trên 4 nhân viên?

```
Select MaPH, count(Manv) from NhanVien Group by MaPH
Having count(Manv)>4
```

1. Ràng buộc toàn vẹn

3. RÀNG BUỘC TOÀN VẸN

- RBTV có bối cảnh trên một quan hệ
  - Ràng buộc miền giá trị
  - Ràng buộc liên bộ
  - Ràng buộc liên thuộc tính
- RBTV có bối cảnh trên nhiều quan hệ
  - Ràng buộc liên thuộc tính liên quan hệ
  - Ràng buộc khóa ngoại (tham chiếu)
  - Ràng buộc liên bộ liên quan hệ
  - Ràng buộc do thuộc tính tổng hợp (Count, Sum)

3. RBTV – CÁC ĐẶC TRƯNG

Các đặc trưng của 1 RBTV:

- Nội dung : phát biểu bằng ngôn ngữ hình thức (phép tính quan hệ, đại số quan hệ, mã giả,…)
- Bối cảnh: là những quan hệ có khả năng làm cho RBTV bị vi phạm.
- Tầm ảnh hưởng: là bảng 2 chiều, xác định các thao tác ảnh hưởng (+) và thao tác không ảnh hưởng (-) lên các quan hệ nằm trong bối cảnh.

3. RBTV – BẢNG TẦM ẢNH HƯỞNG

Bảng tầm ảnh hưởng của RBTV có dạng như sau:

	Thêm	Xóa	Sửa
Quan hệ 1	+	+	- (*)
……			
Quan hệ n	-	-	+(A)

Ký hiệu + : Có thể gây ra vi phạm RBTV
Ký hiệu - : Không thể gây ra vi phạm RBTV
Ký hiệu +(A) : Có thể gây ra vi phạm RBTV khi thao tác trên thuộc tính A
Ký hiệu –(*) : Không thể gây ra vi phạm RBTV do thao tác không thực hiện được

3. RBTV – TRÊN BỐI CẢNH LÀ 1 QUAN HỆ

3.1. Ràng buộc toàn vẹn miền giá trị

- Xét lược đồ quan hệ
  - **NHANVIEN** (MANV, HONV, TENLOT, TENNV, NGSINH, PHAI, DCHI, MA_NQL, PHONG, MLUONG)

Câu hỏi 40: Phái của nhân viên chỉ có thể là ‘Nam’ hoặc ‘Nữ’

- Nội dung:

  $\forall \forall n \in$ NHANVIEN: n.PHAI IN {‘Nam’,’Nữ’}
- Bối cảnh: quan hệ NHANVIEN
- Bảng tầm ảnh hưởng (TAH):

	Thêm	Xóa	Sửa
NHANVIEN	+(PHAI)	-	+(PHAI)

3. RBTV – TRÊN BỐI CẢNH LÀ 1 QUAN HỆ

3.2. Ràng buộc toàn vẹn liên thuộc tính: ràng buộc giữa các thuộc tính trong cùng một quan hệ.

Xét lược đồ quan hệ

DEAN (MADA, TENDA, DDIEM_DA, PHONG, NGBD_DK, NGKT_DK)

Câu hỏi 41: Với mọi đề án, ngày bắt đầu dự kiến (NGBD_DK) phải nhỏ hơn ngày kết thúc dự kiến (NGKT_DK)

Nội dung:

$\forall d \in$ DEAN, d.NGBD_DK <= d.NGKT_DK

3. RBTV – TRÊN BỐI CẢNH LÀ 1 QUAN HỆ

- Bối cảnh: quan hệ DEAN
- Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
DEAN	+ (NGBD_DK, NGKT_DK)	-	+(NGBD_DK, NGKT_DK)

3.3. Ràng buộc toàn vẹn liên bộ: ràng buộc giữa các bộ giá trị trong cùng một quan hệ.

Cho lược đồ quan hệ:

NHANVIEN(MaNV, HoTen, HESO, MucLuong)

Câu hỏi 42: các nhân viên có cùng hệ số lương thì có cùng mức lương.

3. RBTV – TRÊN BỐI CẢNH LÀ 1 QUAN HỆ

- Nội dung:
  - $\forall n1,n2 \in$ NHANVIEN: n1.HESO=n2.HESO thì (n1.MUCLUONG = n2.MUCLUONG)
- Bối cảnh: quan hệ NHANVIEN
- Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
NHANVIEN	+ (HESO, MucLuong)	-	+(HESO, MucLuong)

3. RBTV – BỐI CẢNH NHIỀU QUAN HỆ

3.4. Ràng buộc toàn vẹn tham chiếu

- RBTV tham chiếu còn gọi là ràng buộc phụ thuộc tồn tại hay ràng buộc khóa ngoại.

- Xét lược đồ quan hệ

  PHONGBAN (MAPH, TENPH, TRPH, NGNC)

  NHANVIEN (MANV, HOTEN, NTNS, PHAI, MA_NQL, MAPH, LUONG)

  Câu hỏi 43: Mỗi trưởng phòng phải là một nhân viên trong công ty.

  – Nội dung:

$$\forall p \in \text{PHONGBAN}, \exists n \in \text{NHANVIEN}:$$
$$p.TRPH = n.MANV$$

Hay: PHONGBAN[TRPH] $\subseteq$ NHANVIEN[MANV])

3. RBTV – BỐI CẢNH NHIỀU QUAN HỆ

–Bối cảnh: **NHANVIEN, PHONGBAN**

–Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
PHONGBAN	**+(TRPH)**	**-**	**+(TRPH)**
NHANVIEN	**-**	**+**	**- (*)**

3.5. Ràng buộc toàn vẹn liên thuộc tính liên quan hệ

Xét các lược đồ quan hệ:

DATHANG(MADH, MAKH, NGAYDH)

GIAOHANG(MAGH, MADH, NGAYGH)

3. RBTV – BỐI CẢNH NHIỀU QUAN HỆ

Câu hỏi 44: Ngày giao hàng không được trước ngày đặt hàng

- Nội dung:

$\forall g \in$ GIAO_HANG,
$\exists !d \in$ DAT_HANG:d.MADH= g.MADH $\wedge$d.NGAYDH >= g.NGAYGH

- Bối cảnh: DATHANG, GIAOHANG
- Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
DATHANG	-	-	+ **(ngaydh)**
GIAOHANG	+**(ngaygh)**	-	+ **(ngaygh)**

3. RBTV – BỐI CẢNH NHIỀU QUAN HỆ

3.6. Ràng buộc toàn vẹn liên bộ, liên quan hệ

- RBTV liên bộ, liên quan hệ là điều kiện giữa các bộ trên nhiều quan hệ khác nhau.
- Xét các lược đồ quan hệ
  - **PHONGBAN** (MAPH, TENPH, TRPH, NGNC)
  - **DIADIEM_PHG** (MAPH, DIADIEM)

Câu hỏi 45: Mỗi phòng ban phải có ít nhất một địa điểm phòng

- Nội dung
  - Mỗi phòng ban phải có ít nhất một địa điểm phòng

$\forall \forall p \in$ PHONGBAN, $\exists d \in$ DIADIEM_PHG:

p.MAPH = d.MAPH

3. RBTV – BỐI CẢNH NHIỀU QUAN HỆ

- Bối cảnh: PHONGBAN, DIADIEM_PHG
- Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
PHONGBAN	+	-	-
DIADIEM_PHG	-	+	+ **(MAPH)**

3.7. Ràng buộc toàn vẹn do thuộc tính tổng hợp

PXUAT(SOPHIEU, NGAY, TONGTRIGIA)
CTIET_PX(SOPHIEU, MAHANG, SL, DG)

Câu hỏi 46: Tổng trị giá của 1 phiếu xuất phải bằng tổng trị giá các chi tiết xuất.

3. RBTV – BỐI CẢNH NHIỀU QUAN HỆ

Nội dung

$\forall$ $\forall px \in PXUAT$,

$$px.TONGTRIGIA = \Sigma_{(ct \in CTIET\_PX \wedge ct.SOPHIEU = px.SOPHIEU)} (ct.SL*ct.DG)$$

- Bối cảnh: PXUAT,CTIET_PX
- Bảng tầm ảnh hưởng:

	Thêm	Xóa	Sửa
PXUAT	**-(*)**	**-**	**+ (tongtrigia)**
CTIET_PX	**+(sl,dg)**	**+**	**+ (sl,dg)**

-(*) Ở thời điểm thêm một bộ vào PXUAT, giá trị bộ đó tại TONGTRIGIA là trống.

GIẢI BÀI TẬP

NHẬP MÔN CƠ SỞ DỮ LIỆU

Chương 1. CÁC KHÁI NIỆM CƠ BẢN

1.1. KHÁI NIỆM CƠ SỞ DỮ LIỆU TRONG TIN HỌC

1.1.1. Các mốc lịch sử phát triển Tin học

1936, 1944, 1950, 1954, 1958, 1966, 1968, 1971, 1980,

1990: Tin học —> CNTT

+ Thời gian đầu tiên, khi mới có ngành tin học, nó chỉ có các môn học cơ bản sau: Thuật toán, Lập chương trình máy tính, ngôn ngữ lập trình, …

Chưa có môn học riêng về Cơ sở dữ liệu (CSDL).

+ *Thực tế đơn giản:* Dữ liệu chưa nhiều

- Chưa có môn học riêng về CSDL, vì có thể giải quyết bài toán thực tế một cách đơn giản như sau.

Để quản lý học sinh trong một trường học, người ta chỉ cần tạo lập một bảng danh sách các học sinh (gồm các cột: Họ tên, Ngày sinh, Địa chỉ, …), sau đó ghi vào tệp (File).

- Rõ ràng làm như vậy có thể dẫn tới dư thừa dữ liệu, tốn bộ nhớ, hậu quả tiếp theo là tìm kiếm thông tin sẽ chậm hay không chính xác.

+ *Thực tế phức tạp:* Dữ liệu rất nhiều

- Dữ liệu ngày một nhiều, nếu ghi nhớ chúng không "ngăn lắp", không theo một "trật tự" nhất định, thì rất khó tìm kiếm thông tin, và tốn bộ nhớ.
- Môn học mới cần có, nhằm hướng dẫn cách thức ghi nhớ dữ liệu và phương pháp khai thác dữ liệu một cách hiệu quả. Đó chính là môn CSDL.

1.1.2. Các chuyên ngành trong CNTT

Cách 1: **Hai chuyên ngành:**

Tin học lý thuyết: Thuật toán, CSDL, lập trình, …

Tin học ứng dụng:

Bài toán KH-KT, Bài toán trong công tác quản lý, Bài toán trong hoạt động kinh tế,

Cách 2: **Năm chuyên ngành:**

Khoa học máy tính, Máy tính phần cứng, Mạng máy tính và truyền thông,

Công nghệ phần mềm, Hệ thống thông tin.

1.1.3. Khái niệm Dữ liệu, Cơ sở dữ liệu, Hệ Cơ sở dữ liệu

1/. Khái niệm Dữ liệu

- Dữ liệu (data) có thể hiểu đơn giản là số liệu như họ tên, địa chỉ, số điện thoại của một học sinh hay một khách hàng, ...
- Dữ liệu phức tạp hơn có thể là hình ảnh, âm thanh, dữ liệu đa phương tiện (Multimedia), …

2/. Khái niệm Cơ sở dữ liệu

- Cơ sở dữ liệu (Database: CSDL) có thể hiểu đơn giản là một tập hợp các dữ liệu có liên quan, được lưu trữ trong bộ nhớ theo một cấu trúc nhất định, đã được xác định trước.
- Trong một hệ thống thông tin, CSDL thực chất là một kho chứa dữ liệu.

Ví dụ:

Để quản lý học sinh trong một trường học, có 2 cách tạo lập danh sách các học sinh.

+ Cách 1 (Khi chưa có môn học CSDL):

Người ta chỉ cần tạo lập một bảng danh sách các học sinh (gồm các cột: Họ tên, Ngày sinh, Địa chỉ, Ngành học, Lớp học, …), sau đó ghi vào tệp (File).

+ Cách 2 (Khi đã có môn học CSDL):

Người ta không chỉ tạo ra một bảng danh sách các học sinh, mà tạo ra nhiều bảng dữ liệu liên quan, một CSDL có thể có nhiều bảng dữ liệu, ví dụ:

- Một bảng dữ liệu chính gồm các cột: Họ tên, Ngày sinh, địa chỉ, Mã ngành học, Mã lớp học, …
- Một bảng dữ liệu phụ gồm các cột: Mã ngành học, tên ngành học.
- Một bảng dữ liệu phụ gồm các cột: Mã lớp học, tên lớp học.

Với cách thức tạo lập CSDL như trên sẽ tránh dư thừa dữ liệu, tốn ít bộ nhớ, tốc độ tìm kiếm thông tin sẽ nhanh hơn, …

- Trong bảng dữ liệu chính, thay vì phải ghi tên ngành học, hơi dài: tốn bộ nhớ, Người ta chỉ ghi Mã ngành học: tốn ít bộ nhớ, mặt khác tìm kiếm sẽ nhanh hơn !

3/. Khái niệm Hệ Cơ sở dữ liệu

+ Hệ Cơ sở dữ liệu (CSDL) bao gồm các thành phần sau:

- Cơ sở dữ liệu các thông tin (Kho thông tin).

- Các chương trình thực hiện quản lý CSDL: Cập nhật và khai thác CSDL. (Quản lý Kho thông tin).

1.1.4. Khái niệm Hệ quản trị Cơ sở dữ liệu

Hệ quản trị Cơ sở dữ liệu

+ Hệ quản trị Cơ sở dữ liệu (DataBase Management System: DBMS) là một Hệ chương trình trợ giúp quá trình tạo lập Hệ CSDL và quản lý CSDL.

+ Hệ quản trị Cơ sở dữ liệu có ba thành phần chính:

- Bộ công cụ hỗ trợ tạo lập Cơ sở dữ liệu.

- Bộ công cụ hỗ trợ quản lý Cơ sở dữ liệu (cập nhật, khai thác CSDL).

- Ngôn ngữ lập trình để tạo lập các chương trình quản lý CSDL (cập nhật, khai thác).

Ví dụ:

+ Hệ QT CSDL Foxpro gồm có:

- Bộ công cụ hỗ trợ tạo lập Cơ sở dữ liệu.

- Bộ công cụ hỗ trợ quản lý Cơ sở dữ liệu (cập nhật, khai thác CSDL).

- Ngôn ngữ lập trình Foxpro để tạo lập các chương trình quản lý CSDL.

+ Hệ QT CSDL Oracle

+ Hệ QT CSDL SQL Server

Ví dụ về CSDL

KHACH_HANG

MSKH	TÊNKH	TP
S1	An	HCM
S2	Hoà	HN
S3	Bình	NT
S4	Trang	NT

VAN_CHUYEN

TP	PVC
HCM	01
HN	02
NT	03

MAT_HANG

MSMH	TÊNMH	ĐG
P1	Táo	650
P2	Cam	500
P3	Chanh	450

DAT_HANG

MSKH	MSMH	SL
S1	P1	300
S1	P2	200
S1	P3	400
S2	P1	100
S2	P3	300
S3	P2	200
S4	P2	210

Định nghĩa CSDL: chỉ định cấu trúc mỗi "bảng", bao gồm các phần tử dữ liệu và kiểu dữ liệu tương ứng.

Xây dựng CSDL: Đưa dữ liệu vào các "bảng" KHACHHANG, VANCHUYEN, MATHANG, DATHANG.

Xử lý CSDL: Thực hiện các truy vấn và các phép cập nhật, chẳng hạn: "Khách hàng có tên là An đặt những mặt hàng nào", "Tên những khách hàng đã đặt mặt hàng Cam", "Tính thành tiền"…

1.1.5. Khái niệm Hệ thống thông tin

Để xây dựng được một Hệ thống thông tốt, cần phải hiểu rõ cả 5 chuyên ngành trong CNTT

1.2. CÁC MÔ HÌNH CƠ SỞ DỮ LIỆU

1.2.1. Phân loại tổng quan

1/. Mô hình CSDL bậc thấp **(Mức cụ thể - Mức Vật lý)**

- Mô hình này chỉ quan tâm tới cách thức biểu diễn dữ liệu cụ thể (của CSDL) trong bộ nhớ của máy tính.

Tức là chỉ quan tâm tới việc các dữ liệu của CSDL được lưu trữ trong bộ nhớ của máy tính như thế nào ?

- Mô hình này có ý nghĩa nhiều với các chuyên gia máy tính, nhưng ít có ý nghĩa với người dùng CSDL.

Ví dụ:

2/. Mô hình CSDL bậc cao **(Mức Quan niệm - Logic)**

- Mô hình này quan tâm đến các đối tượng được biểu diễn trong CSDL, ít quan tâm tới cách thức biểu diễn dữ liệu cụ thể trong bộ nhớ của máy tính.
- Mô hình này có ý nghĩa nhiều với với người dùng CSDL, nhưng ít có ý nghĩa.với các chuyên gia máy tính

Ví dụ:

- Mô hình CSDL dạng quan hệ thực thể (Entity Relationship Model)
- Mô hình CSDL hướng đối tượng (Object Oriented Model)

3/. Mô hình CSDL thể hiện **(Mức Logic - Cụ thể)**

- Mô hình CSDL "thể hiện" nằm giữa hai mô hình trên.
- Mô hình này có ý nghĩa với cả chuyên gia máy tính, và với người dùng CSDL.

Ví dụ:

- Mô hình CSDL dạng phân cấp, Mô hình CSDL dạng đồ thị (mạng), Mô hình CSDL dạng quan hệ.

1.2.2. Phân loại cụ thể

Chương 2. MÔ HÌNH CƠ SỞ DỮ LIỆU DẠNG QUAN HỆ

2.1. CÁC KHÁI NIỆM TRONG MÔ HÌNH CSDL QUAN HỆ

Mô hình CSDL quan hệ được Codd đề nghị năm 1970.

2.1.1. Miền, thuộc tính, quan hệ

1/. Khái niệm Miền:

+ Miền (domain) là một tập hợp (các giá trị hoặc các đối tượng) D.

Mỗi miền có một tên, mô tả, kiểu dữ liệu và khuôn dạng.

2/. Quan hệ:

+ Tích Decac:

Gọi $D_1, D_2, \ldots, D_n$ là **n** miền, Tích Decac của **n** miền trên là $D_1 \times D_2 \times \ldots \times D_n$.

+ Quan hệ là tập con của Tích Decac. Tức là Quan hệ **r** $\subseteq D_1 \times D_2 \times \ldots \times D_n$.

3/. Bảng:

+ Bảng là một quan hệ hữu hạn, được biểu diễn thành hàng và cột.

Giá trị trong mỗi cột thuộc về một miền D_i nào đó.

Mỗi hàng là một phần tử của quan hệ **r** .

Ví dụ

Tên miền	M_HOTEN	M_SOĐT
Mô tả	Tập các họ tên người VN	Tập các số điện thoại tại VN
Kiểu dữ liệu	Xâu các ký tự	Xâu các chữ số
Khuôn dạng		(ddd)dddddd

HOTEN	**CMND**	**ĐT_NHA**	**Địa chỉ**	**ĐT_CQ**	**TUOI**
Lê Chí Phèo	220877654	(056)789543	Hà nội	(08)9876548	30
Trần Kim Nở	345267656	(088)765890	Hải phòng	(058)876984	25
Lý Bá Kiến	123123456	(058)908756	Hà nội	(058)888888	50

4/. Thuộc tính:

+ Thuộc tính (Attribute) là một lớp dữ liệu mô tả hành vi, tính chất phát sinh trong CSDL, nghĩa là nó chỉ dựa vào tính chất của lớp dữ liệu này.

Mỗi thuộc tính chỉ có các giá trị trong một miền (domain) của thuộc tính.

Một mục dữ liệu (item) trong thuộc tính là một giá trị trong miền thuộc tính này.

Một thuộc tính là dạng kết nối (joined) nếu nó được định nghĩa từ một vài các thuộc tính khác; do đó domain của nó là tập con của tích Đề các các domain của các thuộc tính này.

Ký hiệu:

- Gọi c là giá trị của thuộc tính C.

Nếu C được tạo thành từ các thuộc tính C_1, C_2,...,C_n, khi đó ta ký hiệu c.C_1 và c(C_1) chỉ giá trị c đối với thuộc tính C_1.

5/. Lược đồ quan hệ: Ký hiệu **R**(A_1, A_2, …, A_n)

Là tập thuộc tính **R** = {A_1, A_2, …, A_n}, mỗi thuộc tính A_i có miền giá trị D_i.

+ Lược đồ quan hệ để mô tả một đối tượng hoặc một loại quan hệ giữa các đối tượng.

+ Bậc của lược đồ quan hệ là số lượng thuộc tính trong lược đồ quan hệ.

Ví dụ:

GV(HOTEN, CMND, ĐT_NHA, ĐC, ĐT_CQ, TUOI)

GV là tên lược đồ quan hệ, có bậc là 6.

HOTEN là một thuộc tính, có miền giá trị DOM(TEN) = M_HOTEN.

ĐT_NHA, ĐT_CQ là các thuộc tính, có miền giá trị DOM(ĐT_NHA) = DOM(ĐT_CQ) = M_SĐT (Miền Số ĐT).

6/. Quan hệ

+ Một quan hệ (Relation) **r** của lược đồ quan hệ **R**(A_1, A_2, …, A_n), ký hiệu là **r**(**R**).

Quan hệ **r** là một tập hữu hạn các bộ (dòng, bản ghi, record) của **R**.

Trong một quan hệ không có hai bộ giống nhau.

+ Một bộ (n-tuple) của **R** là một phần tử của tính Đề các của các domain tương ứng với n thuộc tính của R.

+ Một thực thể (entity) r của R là một bộ của R thoả mãn vị từ ||R||(r)=true.

Chú ý: Thực tế một bộ của tích Đề các có thể hay không là một thực thể của quan hệ R.

Ví dụ: Quan hệ **r** của lược đồ quan hệ **GV**

HOTEN	CMND	ĐT_NHA	ĐC	ĐT_CQ	TUOI
Lê Chí Phèo	220877654	(056)789543	Hà nội	(08)9876548	30
Trần Kim Nở	345267656	(088)765890	Hải phòng	(058)876984	25
Lý Bá Kiến	123123456	(058)908756	Hà nội	(058)888888	50

Các ký hiệu trong mô hình CSDL quan hệ

Lược đồ quan hệ R bậc n: $\mathbf{R(A_1, A_2, \ldots, A_n)}$

Tập thuộc tính của R: $\mathbf{R = \{A_1, A_2, \ldots, A_n\} = R^+}$

Bộ **t** của quan hệ **r**(**R**): $\mathbf{t = <v_1, v_2, \ldots, v_n>}$, trong đó $\mathbf{v_i}$ là giá trị của thuộc tính $\mathbf{A_i}$

t[A_i] (t.A_i , **t**(A_i)): chỉ giá trị của thuộc tính A_i trên bộ **t**.

t[A_u, A_w, …, A_z]: chỉ các giá trị của các thuộc tính A_u, A_w, …, A_z trên bộ **t**.

2.1.2. Khóa của lược đồ quan hệ

1/. *Siêu khoá:*

Tập thuộc tính khác rỗng SK [redacted], được gọi là ***siêu khóa***, nếu

[redacted], [redacted], t_2[redacted], t_1[redacted] t_2[redacted] t_1[SK] [redacted] t_2[SK]

Nhận xét: Mỗi lược đồ quan hệ đều có tối thiểu một siêu khóa.

2/. *Khóa:*

Tập thuộc tính khác rỗng SK [redacted], được gọi là ***khoá***, nếu thỏa mãn đồng thời hai điều kiện: (Tóm lại: Khóa là siêu khóa "nhỏ nhất")

+ K là một siêu khóa của lược đồ quan hệ R.

+ [redacted]' [redacted], K' [redacted], K' không phải là siêu khoá của R.

Chú ý:

- Mọi quan hệ đều có một siêu khóa "tầm thường", đó là tập tất cả các thuộc tính của quan hệ này.

- Khóa là siêu khóa "nhỏ nhất"

Khóa là tập thuộc tính nhỏ nhất, nhờ nó có thể phân biệt các bản ghi với nhau.

Giá trị khóa dùng để nhận biết một bộ trong một quan hệ.

- Khoá được xác định dựa vào ý nghĩa các thuộc tính trong một Lược đồ quan hệ.

- Lược đồ quan hệ có thể có nhiều khoá (gọi là khóa dự tuyển – Candidate key).

Một trong các khóa đó được chỉ định làm khóa chính (primary key) của quan hệ.

Khóa chính thường được chọn là khóa tối thiểu.

Ví dụ: GIẢNG_KHÓA(MÔN, GVIÊN, HKỲ, LỚP, PHÒNG, CA, THỨ)

Tân từ: Mỗi giáo viên (GVIÊN), vào một học kỳ (HKỲ), dạy môn học (MÔN) cho lớp (LỚP), tại phòng (PHÒNG), vào ca giảng (CA) của một thứ trong tuần (THỨ).

[redacted]khoá: {HKỲ, PHÒNG, CA, THỨ}, {MÔN, LỚP}, {GVIÊN, HKỲ, CA, THỨ}

+ Khi cài đặt một quan hệ thành một bảng (Table), cần chọn một khoá làm cơ sở để nhận biết các bộ. Khóa được chọn này gọi là ***khóa chính*** (primary key) [redacted]c thuộc tính khóa chính phải khác trống (khác null).

Thường chọn khóa có số thuộc tính ít hơn làm khóa chính.

Qui ước: các thuộc tính khóa chính được gạch dưới.

VD: GIẢNG_KHÓA(<u>MÔN</u>, GVIÊN, HKỲ, <u>LỚP</u>, PHÒNG, CA, THỨ)

2.1.3. Lược đồ CSDL quan hệ và các ràng buộc toàn ven (RBTV)

Lược đồ CSDL quan hệ = {lược đồ quan hệ} + {Ràng buộc toàn vẹn}

Thể hiện CSDL quan hệ = {Thể hiện quan hệ}

trong đó r_i là thể hiện của R_i thoả mãn các ràng buộc trong tập các ràng buộc toàn vẹn.

Ràng buộc toàn vẹn (RBTV) trên 1 CSDL quan hệ

Ràng buộc toàn vẹn (RBTV, integrity constraint): là những qui tắc, điều kiện, ràng buộc cần được thoả mãn cho mọi thể hiện CSDL quan hệ.

Ràng buộc về khóa (key constraint): 2 bộ khác nhau trong cùng một quan hệ *phải có giá trị tại khoá khác nhau.*

Ràng buộc tham chiếu (referential constraint): Một bộ trong một quan hệ, nếu tham chiếu đến một bộ khác trong một quan hệ khác thì *bộ được tham chiếu phải tồn tại trước.*

Ràng buộc tham chiếu còn gọi là ràng buộc khóa ngoại.

Ngoài ra, còn có một số RBTV về ngữ nghĩa khác.

Khoá ngoại (foreign key)

Xét 2 lược đồ quan hệ R_1 và R_2, FK là 1 tập thuộc tính khác rỗng của R_1. FK được gọi là *khóa ngoại* của R_1 (tham chiếu tới R_2) nếu thỏa mãn 2 điều kiện sau:

Các thuộc tính trong FK phải có cùng miền trị với các thuộc tính khoá chính PK của R_2.

Giá trị tại FK của một bộ t_1 [redacted]$_1$,hoặc bằng giá trị tại PK của một t_2 [redacted]$_2$, hoặc bằng giá trị trống (null). Trường hợp đầu, ta nói t_1 tham chiếu tới bộ t_2.

VD: MAMH là khoá ngoại của ĐATHANG tham chiếu đến MATHANG

Chú ý:

Trong 1 lược đồ quan hệ, một thuộc tính có thể vừa tham gia vào khoá chính, vừa tham gia vào khóa ngoại.

Khoá ngoại có thể tham chiếu đến khóa chính của cùng một lược đồ quan hệ.

VD: NHANVIEN(MaNV, HoTen, MaNguoiPhuTrach)

Có thể có nhiều khóa ngoại tham chiếu đến cùng một khóa chính.

Nên khai báo khoá ngoại (ràng buộc tham chiếu) nếu hệ QTCSDL cho phép.

Ví dụ : **CSDL "CÔNG TY"**

NHANVIEN

Mã-NV	Họ tên	Ngày sinh	Địa chỉ	Mã-DV	Lương

ĐƠN_VỊ

Mã-DV	Tên Đơn vị	Trưởng Đơn vị	Địa điểm ĐV

DỰ_ÁN

Mã-DA	Tên Dự án	Địa điểm DA

PHÂN_CÔNGVIỆC

Mã-NV	**Mã-DA**	Thời gian làm việc

2.2. CÁC PHÉP TÍNH TRONG MÔ HÌNH CSDL QUAN HỆ

2.2.1. Các phép toán cập nhật trên một quan hệ

+ Các phép tính cập nhật trên một quan hệ: Xem, Xen, Xoá, Sửa.

Khi sử dụng các phép toán này, cần đảm bảo các ràng buộc toàn vẹn không bị vi phạm.

+ Các phép tính quan hệ (chiếu, chọn).

2.2.1.1. Phép tính cập nhật

Xem, Xen, Xoá, Sửa.

2.2.1.2. Phép chiếu, phép chọn

1/. Phép chiếu

+ Cho lược đồ quan hệ **R** = {A_1, A_2, …, A_n}, quan hệ **r**, **X** là tập con của R (**X** [redacted] **R**), ta gọi **X** là lược đồ con của **R**.

+ Ta xét quan hệ con của quan hệ **r** chỉ trên tập thuộc tính của **X**, đó là hình chiếu của **r** trên **X**.

Quan hệ **r** chiếu lên **X** là một quan hệ trên lược đồ quan hệ **X** ký hiệu là **r.X**.

Tương tự các phần tử **r.X** được ký hiệu là **t.X** là hình chiếu của **t** lên **X**.

r.X = {**t.X**, **t** [redacted] **r**}.

Phép chiếu được ký hiệu:

[redacted]<ds_thuộc_tính>**(<Tên_quan_hệ>)**

Trong đó:

[redacted] ký hiệu phép chiếu.

<ds_thuộc_tính>: danh sách các thuộc tính của quan hệ <tên_quan_hệ>

<Tên_quan_hệ>: chỉ quan hệ được chọn.

+ Kết quả thu được từ phép chiếu là một quan hệ, có danh sách thuộc tính như trong <ds_thuộc_tính>, với cùng thứ tự.

Chú ý:

+ Nếu <ds_thuộc_tính> chỉ có các thuộc tính không khóa, thì có thể có những bộ trùng lặp sau khi chiếu, phép chiếu ngầm bỏ đi các bộ lặp, do đó kết quả là một quan hệ hợp lệ.

+ Nếu <ds1> [redacted] <ds2> thì [redacted]<ds1>([redacted]<ds2>**(R))** = [redacted]<ds1>**(R)**.

+ Phép chiếu không có tính giao hoán.

Ví dụ:

Cho lược đồ quan hệ **R** ={A, B, C}, lược đồ quan hệ con của **R** là X={A, B}

Phép chiếu <ds_thuộc_tính>**(<Tên_quan_hệ>)** = $_{B}$ **(r)** hay **r.X (A B)**:

r (A	B	C)
a1	b1	c1
a2	b2	c1
a2	b2	c2

r.X (A B) hay $_{B}$ **(r)**

a1	b1
a2	b2

2/. Phép chọu

Phép chọn dùng để trích chọn ***1 tập con*** của quan hệ.

Các bộ được trích chọn phải thỏa mãn *điều kiện chọn*.

Phép chọn được ký hiệu:

<dk_chọn>**(<Tên_quan_hệ>)**

Trong đó:

: ký hiệu phép chọn.

<Tên_quan_hệ>: chỉ quan hệ được chọn.

+ Kết quả thu được từ phép chọn là một quan hệ, có cùng danh sách thuộc tính được chỉ ra trong <Tên_quan_hệ>, nhưng chỉ gồm những bộ thỏa mãn điều kiện chọn.

+ Điều kiện chọn được hình thành từ các mệnh đề có dạng:

<tên_thuộc_tính> <phép_so_sánh> <giá_trị_hằng>

<tên_thuộc_tính> <phép_so_sánh> <tên_thuộc_tính>

<tên_thuộc_tính> là tên thuộc tính của <Tên_quan_hệ>, phép so sánh thường là:

=, >, <,

Các mệnh đề có thể được nối lại nhờ vào các phép

Ví dụ:

Cho lược đồ quan hệ Sinh viên **R**, tìm sinh viên có ít nhất một điểm < 5, tức là xác định  **(R)**

Ten	NS	diem1	diem2
Nam	1978	5	6
Lan	1979	4	6
Hoa	1978	4	3

Ten	NS	diem1	diem2
Hoa	1978	4	3

2.2.2. Các phép toán cập nhật trên nhiều quan hệ

+ Các phép tính trên nhiều quan hệ (như trên tập hợp): hội, giao, trừ, tích Decac, …

+ Các phép tính trên nhiều quan hệ: kết nối quan hệ, phân tách quan hệ, …)

2.2.2.1. Các phép tính như trên tập hợp

+ Khả hợp: (Union compatibility)

Hai lược đồ quan hệ $R(A_1, A_2, …, A_n)$ và $S(B_1, B_2, …, B_n)$ đgl ***khả hợp*** nếu cùng bậc n (cùng số thuộc tính) và có $DOM(A_i) = DOM(B_i)$, với 1 n.

+ Để thực hiện các phép toán trên nhiều quan hệ, điều kiện các quan hệ phải ***khả hợp***.

VD:

SINHVIEN_LOP1

Tên SV	**Địa chỉ**
Trần Kim Nở	Hà nội
Lê Chí Phèo	Hải phòng
Lý Bá Kiến	Hà nội

SINHVIEN_LOP2

Tên SV	**Địa chỉ**
Trương Văn Cam	Sài gòn
Lã Kim Oanh	Hải phòng
Vũ Xuân Trường	Thái bình
Lê Chí Phèo	Hải phòng

1/. Phép hội của R và S, ký hiệu là **R S**, là một quan hệ gồm các bộ thuộc R hoặc thuộc S, hoặc thuộc cả hai quan hệ, các bộ trùng lắp thì loại bỏ.

2/. Phép giao của R và S, ký hiệu là **R S,** là một quan hệ gồm các bộ thuộc đồng thời R và S.

3/. Phép trừ của R và S, ký hiệu **R – S,** là một quan hệ gồm các bộ thuộc R và không thuộc S.

Ví dụ:

SINHVIEN_LOP1 [redacted] SINHVIEN_LOP2

Tên SV	**Địa chỉ**
Trần Kim Nở	Hà nội
Lê Chí Phèo	Hải phòng
Lý Bá Kiến	Hà nội
Trương Văn Cam	Sài gòn
Lã Kim Oanh	Hải phòng
Vũ Xuân Trường	Thái bình

SINHVIEN_LOP1 [redacted] SINHVIEN_LOP2

Tên SV	**Địa chỉ**
Lê Chí Phèo	Hải phòng

SINHVIEN_LOP1 - SINHVIEN_LOP2

Tên SV	**Địa chỉ**
Trần Kim Nở	Hà nội
Lý Bá Kiến	Hà nội

Các tính chất:

Giao hoán: R [redacted] = S [redacted], R [redacted] = S [redacted]

Kết hợp: R [redacted] S [redacted]) = (R [redacted]) [redacted],

R [redacted] S [redacted]) = (R [redacted]) [redacted]

2.2.2.2. Các phép tính kết nối, phân tách quan hệ

1/. Tích Decac (Descartes)

Cho **R**(A_1, A_2, …, A_n) và **S**(B_1, B_2, …, B_m), tích ***Decac*** giữa hai quan hệ R và S, ký hiệu là **R x S,** là quan hệ có n + m thuộc tính.

Q(A_1, A_2, …, A_n, B_1, B_2, …, B_m)

Trong đó 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 có u . v bộ.

Ví dụ:

R	A	B
	a1	b1
	a2	b2

S	C	D
	c1	d1
	c2	d2
	c3	d3

RxS	A	B	C	D
	a1	b1	c1	d1
	a1	b1	c2	d2
	a1	b1	c3	d3
	a2	b2	c1	d1
	a2	b2	c2	d2
	a2	b2	c3	d3

2/. Phép chia

Cho lược đồ quan hệ **R** = {A_1, .., A_n} và **S** là lược đồ con của R (**S** [redacted]),

giả sử **r** và **s** là hai quan hệ trên **R** và **S** tương ứng.

Phép chia của quan hệ r cho s, ký hiệu: **r ÷ s**, kết quả là quan hệ trên lược đồ **R – S** gồm các bộ t, sao cho tồn tại bộ u [redacted] thì bộ <**t, u**> [redacted] **r**.

r ÷ s = {**t** : [redacted] **s** và <**t, u**> [redacted] **r**}

Chú ý: S phải thực sự là tập con của R, (**S** [redacted]).

Ví dụ:

r (A B C D)

a b c d

a b e f

b c e f

e d c d

e d e f

a b d e

s (C D)

c d

e f

r ÷ s (A B)

a b

e d

b c

3/. Phép nối tự nhiên (Join)

Cho hai lược đồ: R_1 và R_2, r_1, r_2 là hai quan hệ tương ứng trên R_1, R_2.

Phép kết nối (tự nhiên) của r_1 và r_2 ký hiệu: $\mathbf{r_1}$ |><| $\mathbf{r_2}$ là quan hệ trên lược đồ $R_1 \cup R_2$ gồm các phần tử **t** mà chiếu lên R_1 là phần tử thuộc r_1, còn chiếu lên R_2 là phần tử thuộc r_2.

$\mathbf{r_1}$ |><| $\mathbf{r_2}$ = {t : $t.R_1 \in r_1$ và $t.R_2 \in r_2$ }

Trong trường hợp hai tập thuộc tính như nhau thì r_1 |><| r_2 = $r_1 * r_2$.

Trong trường hợp hai tập là tách biệt nhau thì r_1 |><| r_2 = $r_1 \times r_2$.

Ví dụ:

r_1 (A	B	C)
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_1	c_1

r_2 (C	D)
c_1	d_1
c_2	d_2

r_1 |><| r_2 (A	B	C	D)
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_2	b_1	c_1	d_1

4/. Phép kết nối theo phép tính θ

Cho r và s là hai quan hệ tương ứng trên hai lược đồ quan hệ R và S rời nhau ($R \cap S = \emptyset$)

Phép kết nối theo phép tính θ của quan hệ r và s, ký hiệu $\mathbf{r}|><|_{i\,\theta\,j}\ \mathbf{s}$ là một quan hệ trên lược đồ $R \in$ gồm những bộ thuộc tích Đề các của r và s sao cho thành phần thứ i của quan hệ r thỏa mãn phép toán θ với thành phần thứ j của quan hệ s.

Ví dụ: Trong trường hợp này θ là quan hệ <, i=2 và j=1.

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

s	D	E
	3	1
	6	2

r |><| **C ≤ D** s

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

5/. Các phép toán quan hệ bổ sung

Hầu hết các hệ QT CSDL đều bổ sung thêm một số phép toán sau:

AVERAGE : tính giá trị trung bình

MAX : tính giá trị lớn nhất

MIN : tính giá trị bé nhất

SUM : tính tổng cộng

COUNT : đếm.

Cú pháp: **<các thuộc tính phân nhóm>*F*<d/s hàm> (<quan hệ>)**

Ví dụ:

Với mỗi phòng ban, tìm số lượng nhân viên và mục lương trung bình.

R(SOPHG, SONV, LUONGTB) [redacted]$_{\text{G}}$***F***$_{\text{COUNT MANV, AVERAGE LUONG}}$(NHANVIEN)

CHƯƠNG 3. LÝ THUYẾT PHỤ THUỘC HÀM

3.1. Các nguyên tắc thiết kế lược đồ quan hệ

Khi chúng ta nhóm các thuộc tính để tạo nên một lược đồ quan hệ, ta giả thiết rằng có một ý nghĩa nào đó gắn với các thuộc tính. Ý nghĩa này, còn gọi là *ngữ nghĩa*, chỉ ra việc hiểu các giá trị thuộc tính lưu giữ trong các bộ của một quan hệ như thế nào. Nói cách khác, các giá trị thuộc tính trong một bộ liên hệ với nhau như thế nào. Nếu việc thiết kế khái niệm được làm một cách cẩn thận, sau đó là một chuyển đổi sang các quan hệ thì hầu hết ngữ nghĩa đã được giải thích và thiết kế kết quả có một ý nghĩa rõ ràng. Nói chung, việc giải thích ngữ nghĩa của quan hệ càng dễ dàng thì việc thiết kế lược đồ quan hệ càng tốt. Một ví dụ về thiết kế lược đồ quan hệ tốt là lược đồ cơ sở dữ liệu "CÔNG TY". Trong lược đồ đó, các thuộc tính đều có ý nghĩa rõ ràng, không có tính mập mờ. Nguyên tắc sau sẽ hỗ trợ cho việc thiết kế lược đồ quan hệ.

Nguyên tắc 1: Thiết kế một lược đồ quan hệ sao cho dễ giải thích ý nghĩa của nó. Đừng tổ hợp các thuộc tính từ nhiều kiểu thực thể và kiểu liên kết vào một quan hệ đơn. Một cách trực quan, nếu một lược đồ quan hệ tương ứng với một kiểu thực thể hoặc một kiểu liên kết thì ý nghĩa trở nên rõ ràng. Ngược lại, một quan hệ tương ứng với một hỗn hợp các thực thể và liên kết thì ý nghĩa trở nên không rõ ràng.

3.1.2 Thông tin dư thừa trong các bộ và sự dị thường cập nhật

Một mục tiêu của thiết kế lược đồ là làm tối thiểu không gian lưu trữ các quan hệ cơ sở. Các thuộc tính được nhóm vào trong các lược đồ quan hệ có một ảnh hưởng đáng kể đến không gian lưu trữ. Nếu cùng một thông tin được lưu giữ nhiều lần trong cơ sở dữ liệu thì ta gọi đó là dư thừa thông tin và điều đó sẽ làm lãng phí không gian nhớ. Ví dụ, giả sử ta có bảng cơ sở sau đây:

HÀNGHÓA_KHO

MãsốHH	**TênHH**	**Mô Tả Hàng**	**Ngàysản xuất**	**MãsốKho**	**TênKho**	**Ghichu**
Mh01	Ốc vít	Loại 3 phân	12/02/79	5	Kho số5	Trữ sản phẩm

Mh02	Bulong	Loại lớn	14/02/66	5	Kho số 5	Trữ sản phẩm
Mh03	Kìm	Khâu bao	05/08/79	4	Vật liệu	Trữ vật liệu
Mh04	Dao	Loại lớn	26/06/52	4	Vật liệu	Trữ vật liệu
Mh05	Kéo	Cắt bao	14/08/73	5	Kho số 5	Trữ sản phẩm
Mh06	Đinh	8 phân	26/03/83	5	Kho số 5	Trữ sản phẩm
Mh07	Dây gai	Xây dựng	15/03/80	4	Vật liệu	Trữ vật liệu
Mh08	Găng tay	Công nghiệp	02/05/47	1	Thiết bị	Các thiết bị điện

Ở đây có sự dư thừa thông tin. Nếu một kho lưu trữ nhiều sản phẩm thì thông tin về KHO (Mã số kho, Tên kho, Ghi chú) được lưu giữ nhiều lần trong bảng. So với việc dùng hai bảng HÀNG HÓA và KHO riêng rẽ không làm lãng phí không gian nhớ.

Ngoài việc lãng phí không gian nhớ nó còn dẫn đến một vấn đề nghiêm trọng là sự dị thường cập nhật. Dị thường cập nhật bao gồm : Dị thường Chèn, dị thường Xoá, dị thường Sửa đổi. Những dị thường cập nhật này sẽ đưa vào cơ sở dữ liệu những thông tin "lạ" và làm cho cơ sở dữ liệu mất tính đúng đắn.

Dị thường Chèn: Gây ra khó khăn khi chèn các bộ giá trị vào bảng hoặc dẫn đến vi phạm ràng buộc.

Ví dụ: Để chèn một bộ giá trị cho một *mặt hàng* mới vào bảng HÀNGHÓA_KHO ngoài các thông tin về *hàng hóa*, ta phải đưa vào các thông tin về *kho* mà sản phẩm đó được lưu trữ hoặc các giá trị null (nếu hàng hóa đó không lưu trữ trong kho nào cả). Các thông tin về *kho* phải được đưa vào một cách đúng đắn, phù hợp với các thông tin của *kho* đó trong các bộ khác.Trong lúc đó, với việc sử dụng 2 quan hệ HÀNG HÓA và KHO chúng ta không phải lo lắng gì, vì các thông tin về một *kho* chỉ được lưu trữ một lần.

Rất khó chèn một *kho* mới vào quan hệ HÀNG HÓA_KHO nếu kho đó không có sản phẩm nào lưu trữ. Cách giải quyết duy nhất là điền các giá trị null vào các thuộc tính của hàng hóa. Điều đó làm nảy sinh vấn đề về ràng buộc bởi vì MãsốHH là khóa chính của quan hệ.

Dị thường Xóa: Gây ra việc mất thông tin khi xóa.

Ví dụ, khi ta xóa một bộ giá trị trong bảng HÀNG HÓA - KHO. Nếu hàng hóa tương ứng với bộ giá trị đó là sản phẩm cuối cùng lưu trong kho thì phép xóa sẽ kéo theo việc làm mất thông tin về kho.

Dị thường Sửa đổi: Gây ra việc sửa đổi hàng loạt khi ta muốn sửa đổi một giá trị trong một bộ nào đó.

Dựa trên các dị thường ở trên, chúng ta có thể phát biểu nguyên tắc sau:

Nguyên tắc 2: Thiết kế các lược đồ quan hệ cơ sở sao cho không sinh ra những dị thường cập nhật trong các quan hệ. Nếu có xuất hiện những dị thường cập nhật thì phải ghi chép lại một cách rõ ràng và phải đảm bảo rằng các chương trình cập nhật dữ liệu sẽ thực hiện một cách đúng đắn.

3.1.3. Các giá trị không xác định trong các bộ

Trong một số thiết kế lược đồ, chúng ta có thể nhóm nhiều thuộc tính với nhau vào một quan hệ "béo". Nếu nhiều thuộc tính không thích hợp cho mọi bộ trong một quan hệ, chúng ta sẽ kết thúc với nhiều giá trị null trong các bộ đó. Điều đó có thể làm tăng không gian ở mức lưu trữ và có thể dẫn đến vấn đề về hiểu ý nghĩa của các thuộc tính. Việc chỉ ra các phép nối ở mức lô gic cũng sẽ gặp khó khăn. Một vấn đề nữa với các giá trị null là các hàm nhóm như COUNT, SUM không áp dụng được đối với chúng. Hơn nữa, các giá trị null có thể nhiều cách giải thích, chẳng hạn như thuộc tính không áp dụng được cho bộ này, giá trị của thuộc tính cho bộ này là không có hoặc giá trị cho thuộc tính là có nhưng vắng mặt. Tóm lại, các giá trị null có nhiều ý nghĩa khác nhau.

Nguyên tắc 3: Tránh càng xa càng tốt việc đặt vào trong các quan hệ cơ sở những thuộc tính mà các giá trị của chúng thường xuyên là null. Nếu không thể tránh được các giá trị null thì phải đảm bảo rằng chúng chỉ áp dụng trong các trường hợp đặc biệt và không áp dụng cho một số lớn các bộ trong quan hệ.

3.1.4 Sinh ra các bộ giả

Nhiều khi chúng ta đưa vào cơ sở dữ liệu những quan hệ không đúng, việc áp dụng các phép toán (nhất là các phép nối) sẽ sinh ra các bộ giá trị không đúng, gọi là các bộ "giả".

Ví dụ, xét hai lược đồ quan hệ:

HH_KHO (Tên, Kho)

HH_SP(Mã sốSP, Mã sốDA, Sốlượng, TênDA, Kho)

HH_KHO	**Tên**	**Kho**
	Đinh	Kho số 2
	Ốc vít	Kho số 1
	Kéo	Kho số 4
	Dao	Kho số 2

HH_SP	**MãsốSP**	**MãsốDA**	**Sốlượng**	**TênDA**	**Kho**
	SP001	1	32	DA01	Kho số 2
	SP001	2	7	DA02	Kho số 1
	SP016	3	40	DA03	Kho số 4
	SP018	1	20	DA01	Kho số 2

Bây giờ ta nối tự nhiên hai quan hệ trên với nhau, ta có quan hệ

	MãsốSP	**MãsốDA**	**Số lượng**	**TênDA**	**Địađiêm**	**Tên**
	SP001	1	32	DA01	Kho số 2	Đinh
*	SP001	1	32	DA01	Kho số 2	Dao
	SP001	2	7	DA02	Kho số 1	Ốc vít
	SP016	3	40	DA03	Kho số 4	Kéo
*	SP018	1	20	DA01	Kho số 2	Dao
	SP018	1	20	DA01	Kho số 2	Đinh

Ta thấy các dòng đánh dấu * là các bộ " giả". Đấy là các bộ giá trị không có trên thực tế.

Nguyên tắc 4: 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". Đừng có các quan hệ chứa các thuộc tính nối khác với các tổ hợp khoá chính-khoá ngoài. Nếu không tránh được những quan hệ như vậy thì đừng nối chúng trên các thuộc tính đó, bởi vì các phép nối có thể sinh ra các bộ "giả".

3.2. Các phụ thuộc hàm

Khái niệm cơ bản nhất trong thiết kế lược đồ quan hệ là khái niệm phụ thuộc hàm. Trong phần này chúng ta sẽ định nghĩa hình thức khái niệm này và cách sử dụng nó để định nghĩa các dạng chuẩn cho các lược đồ quan hệ

3.2.1. Định nghĩa phụ thuộc hàm (functional dependency - FD)

Một phụ thuộc hàm là một ràng buộc giữa hai nhóm thuộc tính của một cơ sở dữ liệu. Giả sử rằng lược đồ cơ sở dữ liệu của ta có n thuộc tính A_1, A_2, …, A_n và hãy nghĩ rằng toàn bộ cơ sở dữ liệu được mô tả bằng một lược đồ quan hệ chung R(U), U ={A_1, A_2, …., A_n}. Giả sử X và Y là hai tập con của R.

Một phụ thuộc hàm, ký hiệu là **X** [redacted] **Y**, giữa hai tập thuộc tính X và Y chỉ ra một ràng buộc trên các bộ có thể có tạo nên một trạng thái quan hệ r của R.

Ràng buộc đó là: với hai bộ $\mathbf{t_1}$ và $\mathbf{t_2}$ bất kỳ trong r, nếu có $\mathbf{t_1}$[**X**] = $\mathbf{t_2}$[**X**] thì cũng phải có $\mathbf{t_1}$[**Y**] = $\mathbf{t_2}$[**Y**].

Nếu có X [redacted], chúng ta cũng nói rằng X xác định hàm Y hoặc Y phụ thuộc hàm vào X. Tập thuộc tính X được gọi là vế trái của FD, tập thuộc tính Y được gọi là vế phải của FD. Như vậy, X xác định hàm Y trong lược đồ quan hệ R khi và chỉ khi nếu hai bộ của r(R) bằng nhau trên các giá trị của X thì chúng nhất thiết phải bằng nhau trên các giá trị của Y.

Chú ý rằng nếu X [redacted] thì không thể nói gì về Y [redacted]

Một phụ thuộc hàm là một tính chất ngữ nghĩa của các thuộc tính. Những người thiết kế cơ sở dữ liệu sẽ dùng hiểu biết của họ về ý nghĩa của các thuộc tính của R để chỉ ra các phụ thuộc hàm có thể có trên mọi trạng thái quan hệ của r(R) của R. Khi ngữ nghĩa của hai tập thuộc tính trong R chỉ ra rằng có thể có một phụ thuộc hàm, chúng ta sẽ đặc tả phụ thuộc hàm như một ràng buộc. Các trạng thái quan hệ r(R) thoả mãn các

ràng buộc phụ thuộc hàm được gọi là các trạng thái hợp pháp của R, bởi vì chúng tuân theo các ràng buộc phụ thuộc hàm. Như vậy, việc sử dụng chủ yếu của các phụ thuộc hàm là dùng để mô tả một lược đồ quan hệ R bằng việc chỉ ra các ràng buộc trên các thuộc tính phải thoả mãn ở mọi thời điểm. Một phụ thuộc hàm là một tính chất của lược đồ quan hệ R chứ không phải là tính chất của một trạng thái hợp pháp r của R. Vì vậy, một phụ thuộc hàm không thể được phát hiện một cách tự động từ một trạng thái r mà phải do một người hiểu biết ngữ nghĩa của các thuộc tính xác định một cách rõ ràng. Ví dụ, ta có quan hệ sau

DẠY	**Giáoviên**	**Mônhọc**	**Tàiliệu**
	Hồng Tuyến	Pttk hệ thống	Lý thuyết CSDL q hệ
	Hồng Tuyến	Otomat&NNHT	Toán rời rạc
	Đặng Hải	Lý thuyết đồ thị	Toán rời rạc
	Lê Duy	Toán A3	Toán cao cấp

Mới nhìn qua, chúng ta có thể nói có một phụ thuộc hàm Tàiliệu [redacted]ônhọc, tuy nhiên chúng ta không thể khẳng định được vì điều đó chỉ đúng với trạng thái quan hệ này, biết đâu trong trạng thái quan hệ khác có thể có hai môn học khác nhau sử dụng cùng một tài liệu tham khảo, ví dụ trên ta thấy hai môn *Otomat &NNHT* và *lý thuyết đồ thị* sử dụng cùng một tài liệu tham khảo đó là *Toán rời rạc*. Với một trạng thái cụ thể, chúng ta chỉ có thể khẳng định là không có một phụ thuộc hàm giữa nhóm thuộc tính này và nhóm thuộc tính khác. Để làm điều đó chúng ta chỉ cần đưa ra một phản ví dụ. Chẳng hạn, ở trong quan hệ trên chúng ta có thể khẳng định rằng không có phụ thuộc hàm giữa Giáoviên và Mônhọc bằng cách chỉ ra ví dụ là *Hồng Tuyến* dạy hai môn học " Pttk hệ thống" và "Otomat&NNHT" vậy Giáo viên không thể xác định duy nhất Môn học.

Để biểu diễn các phụ thuộc hàm trong một lược đồ quan hệ, chúng ta sử dụng khái niệm sơ đồ phụ thuộc hàm. Mỗi FD được biểu diễn bằng một đường nằm ngang. Các thuộc tính ở vế trái của FD được nối với đường biểu diễn FD bằng các đường thẳng đứng, các thuộc tính ở vế phải của FD được nối với đường biểu diễn FD bằng mũi tên chỉ đến các thuộc tính

Ví dụ 1: Ta có lược đồ quan hệ sau:

MUAHANG(Mãhàng, Mãkhách, Tênhàng, Tênkhách, Sốlượng)

Với các phụ thuộc hàm:

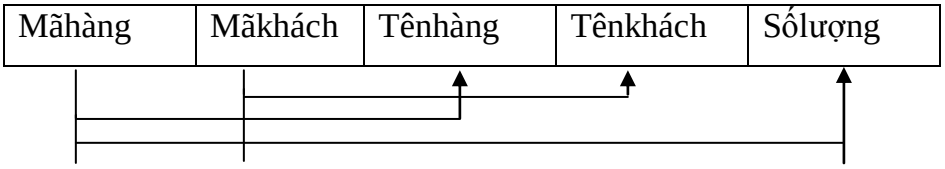
Mãhàng [redacted]ênhàng

Mãkhách [redacted]ênkhách

Mãhàng, Mãkhách [redacted]ốlượng

có sơ đồ phụ thuộc hàm như sau:

MUAHANG



Ví dụ 2: quan hệ ĐIEM(MaSV, TenSV, Ngaysinh, MaMH, TenMH, DVHT, Diem) Có phụ thuộc hàm:

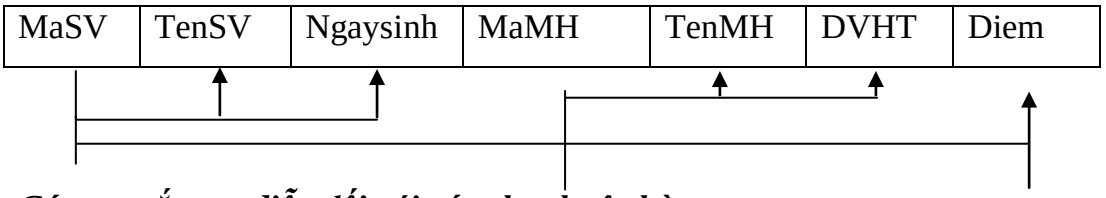
MaSV→TenSV, Ngaysinh

MaMH →TenMH, DVHT

MaSV, MaMH → Diem.

có sơ đồ phụ thuộc hàm như sau:

DIEM



3.2.2. Các quy tắc suy diễn đối với các phụ thuộc hàm

Chúng ta ký hiệu F là tập các phụ thuộc hàm được xác định trên một lược đồ quan hệ R(U). X và Y là hai tập con của U. Một phụ thuộc hàm X [redacted], được gọi là suy diễn được từ một tập các phụ thuộc hàm F được xác định trên R nếu X[redacted] đúng trong mỗi trạng thái quan hệ r là mở rộng hợp pháp của R; nghĩa là mỗi khi r làm thoả mãn mọi phụ thuộc hàm trong F, X[redacted] cũng đúng trong r. Ta sử dụng ký hiệu F |= X[redacted] để ký hiệu phụ thuộc hàm X[redacted] được suy diễn từ tập các phụ thuộc hàm F. Để xác định một cách suy diễn các phụ thuộc hàm có hệ thống, chúng ta phải phát hiện một tập hợp các quy tắc suy diễn. Tập quy tắc này sẽ được sử dụng để suy diễn các phụ thuộc hàm mới từ một tập các phụ thuộc hàm cho trước. Có 6 quy tắc suy diễn đối với các phụ thuộc hàm:

QT1 (quy tắc phản xạ) : Nếu X [redacted] thì X [redacted]

QT2 (quy tắc tăng) : { X [redacted] } |= XZ [redacted]

QT3 (quy tắc bắc cầu) : { X [redacted], Y [redacted] } |= X [redacted]

QT4 (quy tắc chiếu) : { X [redacted] } |= X [redacted] và X [redacted]

QT5 (quy tắc hợp) : { X [redacted], X [redacted] } |= X [redacted]

QT6 (quy tắc tựa bắc cầu): { X [redacted], WY [redacted] } |= WX [redacted]

Quy tắc phản xạ phát biểu rằng một tập hợp các thuộc tính luôn luôn xác định chính nó hoặc một tập con bất kỳ của nó. Vì QT1 tạo ra các phụ thuộc luôn luôn đúng, những phụ thuộc như vậy được gọi là *tầm thường*. Một cách hình thức, một phụ thuộc hàm X [redacted] là tầm thường nếu X [redacted]; ngược lại, nó được gọi là *không tầm thường*. Quy tắc tăng (QT2)nói rằng việc thêm cùng một tập thuộc tính vào cả hai vế của một phụ thuộc hàm sẽ tạo ra một phụ thuộc hàm đúng đắn. Theo QT3, các phụ thuộc hàm là bắc cầu. Quy tắc chiếu (QT4) nói rằng chúng ta có thể bỏ bớt các thuộc tính ra khỏi vế phải của phụ thuộc hàm. Việc áp dụng nhiều lần quy tắc này có thể tách phụ thuộc hàm X [redacted] $A_1, A_2, \ldots, A_n$} thành một tập hợp phụ thuộc hàm { X [redacted] $_1$, X [redacted] $_2$, . . ., X [redacted] $_n$}. Quy tắc hợp (QT5) cho phép chúng ta làm ngược lại; ta có thể gộp các phụ thuộc hàm { X [redacted] $_1$, X [redacted], . . ., X [redacted] $_n$}thành một phụ thuộc hàm đơn X [redacted] $A_1, A_2, \ldots, A_n$}.

Có thể chứng minh các quy tắc suy diễn ở trên một cách trực tiếp hoặc bằng phản chứng dựa trên định nghĩa của phụ thuộc hàm. Để chứng minh phản chứng, ta giả thiết một quy tắc là không đúng và chỉ ra rằng điều đó là không thể. Sau đây là chứng minh các quy tắc.

<u>Quy tắc 1:</u>

Giả sử rằng X [redacted] và hai bộ t_1 và t_2 trong một thể hiện quan hệ r của R sao cho $t_1[X] = t_2[X]$. Khi đó $t_1[Y] = t_2[Y]$ bởi vì X [redacted]; như vậy X [redacted] phải xẩy ra trong r.

<u>Quy tắc 2</u> (chứng minh phản chứng):

Giả sử rằng X [redacted] đúng trong một thể hiện quan hệ r của R nhưng XZ [redacted] không đúng. Khi đó phải có hai bộ t_1 và t_2 trong r sao cho

(1) $t_1[X] = t_2[X]$,

(2) $t_1[Y] = t_2[Y]$,

(3) $t_1[XZ] = t_2[XZ]$ và

(4) $t_1[YZ]$ [redacted] $t_2[YZ]$. Điều đó là không thể bởi vì từ (1) và (3) chúng ta suy ra

(5) $t_1[Z] = t_2[Z]$, và từ (2) và (5) ta suy ra $t_1[YZ] = t_2[YZ]$, mâu thuẫn với (4).

Quy tắc 3: Giả sử ta có X[redacted] và Y[redacted]. Khi đó với mọi bộ t_1 và t_2 trong r, $t_1[X] = t_2[X]$ kéo theo $t_1[Y] = t_2[Y]$ (vì X[redacted]), và $t_1[Y] = t_2[Y]$ kéo theo $t_1[Z] = t_2[Z]$ vì (Y[redacted]). Như vậy, với mọi bộ t_1 và t_2 trong r, $t_1[X] = t_2[X]$ kéo theo $t_1[Z] = t_2[Z]$ hay là X[redacted].

Chúng ta có thể chứng minh các quy tắc từ QT4 đên QT6 theo phương pháp trên. Tuy nhiên ta có thể lợi dụng các quy tắc đã được chứng minh là đúng để chứng minh chúng. Sau đây ta chứng minh theo cách đấy.

Quy tắc 4:

1. X[redacted]Z (cho trước)
2. YZ [redacted] (sử dụng QT1 và YZ [redacted])
3. X[redacted] (sử dụng QT3 trên 1. và 2.)

Quy tắc 5:

1. X[redacted] (cho trước)
2. X[redacted] (cho trước)
3. X[redacted]X (sử dụng QT2 trên 1. bằng cách thêm vào cả hai vế X, và XX=X)
4. YX[redacted]Z (sử dụng QT2 trên 2. bằng cách thêm vào cả hai vế Y)
5. X[redacted]Z (sử dụng QT3 trên 3. và 4.)

Quy tắc 6:

1. X[redacted](cho trước)
2. WY[redacted](cho trước)
3. WX[redacted]Y (sử dụng QT2 trên 1. bằng cách thêm vào c hai vế W)
4. WX[redacted] (sử dụng QT3 trên 3. và 2.)

Từ chứng minh ở trên, chúng ta thấy rằng chỉ cần 3 qui tắc QT1, QT2, QT3 là đủ, các quy tắc sau có thể suy diễn trực tiếp từ 3 qui tắc đó. Các quy tắc từ QT1 đến QT3 được gọi là các *quy tắc suy diễn Amstrong.*

3.2.3. Bao đóng của tập phụ thuộc hàm và bao đóng của tập thuộc tính dưới một tập phụ thuộc hàm.

Thông thường, những người thiết kế cơ sở dữ liệu đầu tiên chỉ ra một tập các phụ thuộc hàm dễ xác định được nhờ ngữ nghĩa của các thuộc tính của R. Sau đó ta sử dụng các quy tắc Amstrong để suy diễn các phụ thuộc hàm bổ sung. Cho trước một tập phụ thuộc hàm F, tập hợp tất cả các phụ thuộc hàm suy ra được từ F bằng cách sử dụng các quy tắc suy diễn được gọi là bao đóng của tập F và được ký hiệu là F^{+}.

Ví dụ: F = { X [redacted]; Y [redacted]}

F^{+} = { F [redacted]X [redacted], X [redacted]T } }

Một cách có hệ thống, để xác định tất cả các phụ thuộc hàm bổ sung, đầu tiên hãy xác định mỗi tập thuộc tính X xuất hiện ở vế trái của một phụ thuộc hàm nào đấy trong F và sau đó xác định tập hợp tất cả các thuộc tính phụ thuộc hàm vào X. Như vậy, với mỗi tập thuộc tính X, chúng ta xác định tập X^{+} các thuộc tính phụ thuộc hàm vào X dựa trên F. X^{+} được gọi là bao đóng của X dưới F và được định nghĩa là:

X^{+} = {A [redacted] | X [redacted]}.

Theo định nghĩa X^{+} chúng ta có bổ đề sau:

Bổ đề 3. 1: X[redacted] được suy diễn từ tập phụ thuộc hàm F bằng các quy tắc suy diễn Amstrong khi và chỉ khi Y [redacted].

Thật vậy, giả sử X[redacted] được suy diễn từ tập phụ thuộc hàm F bằng các quy tắc suy diễn Amstrong và Y = $A_1A_2…A_m$ với A_1, A_2, …A_m là các thuộc tính. Như vậy, theo quy tắc chiếu ta có X [redacted]$_1$, X [redacted]$_2$, …., X [redacted]$_m$. Theo định nghĩa X^{+}, A_i [redacted] với i = 1, 2, …, m. Như vậy, Y [redacted].

Ngược lại, giả sử Y [redacted] và Y = $A_1A_2…A_m$. Theo định nghĩa X^{+} ta có X[redacted]i với i = i, . . m. Theo quy tắc hợp, ta có X [redacted].

Để xác định X^{+} chúng ta sử dụng thuật toán sau:

<u>Thuật toán 3. 1</u> (xác định X^{+}, bao đóng của X dựa trên F)

X^{+} = X;

Repeat

Old X^{+} = X^{+} ;

với mỗi phụ thuộc hàm Y [redacted] trong F thực hiện

nếu X^{+} [redacted] thì X^{+} = X^{+} [redacted];

until (X^+ = Old X^+);

Ví dụ : Xét lược đồ quan hệ

R={MaSV, TenSV, Ngaysinh, MaMH, TenMH, DVHT, Diem}

Có tập phụ thuộc hàm:

F={MaSV→{TenSV, Ngaysinh}, MaMH →{TenMH, DVHT}, {MaSV, MaMH} → Diem}

Xác định {MaSV, MaMH}$^+$.

Áp dụng thuật toán 1. 1 ta có:

{MaSV}$^+$ = {MaSV, TenSV, NgaySinh}

{MaMH }$^+$ = {MaMH, TênMH, DVHT}

{MaSV, MaMH}$^+$ = {MaSV, TenSV, Ngaysinh, MaMH, TenMH, DVHT, Diem}.

Định lý 3. 1: Thuật toán 3.1 tính X^+ là đúng.

Giả sử ta có X [redacted]. Theo thuật toán ở trên, Y sẽ được thêm vào tập X^+, như vậy Y [redacted]$^+$.

Ngược lại, giả sử Y [redacted]$^+$. Theo cách xây dựng X^+ ta có X [redacted]$^+$. Theo quy tắc phản xạ, ta có X^+[redacted]. Vậy X [redacted]

Định lý 3. 2: Hệ quy tắc suy diễn Amstrong là đúng và đầy đủ.

Chúng ta đã chứng minh tính đúng đắn của các quy tắc QT1, QT2, QT3 ở trên. Bây giờ ta chỉ cần chứng minh các quy tắc đó là đầy đủ, nghĩa là nếu X[redacted]không suy diễn lô gic được từ F bằng hệ suy diễn Amstrong thì X[redacted]không thoả mãn trên quan hệ r(R).

Để làm điều đó, giả sử rằng X[redacted] không suy diễn được từ F bằng hệ suy diễn Amstrong, ta sẽ xây dựng một quan hệ r sao cho các phụ thuộc hàm của F là thoả mãn trên r nhưng X [redacted]không thoả mãn trên r. Quan hệ r được xây dựng như sau: r chỉ gồm hai bộ giá trị t_1 và t_2, trong đó các thuộc tính trong t_1 đều có giá trị 1, trong t_2 chỉ có các thuộc tính thuộc X^+ là có giá trị 1 còn các thuộc tính còn lại có giá trị 0.

t1 : 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

t2: 1 1 1 1 1 1 1 0 0 0 0 0 0 0

Ta chứng tỏ rằng mọi phụ thuộc hàm của F đều thoả mãn trên r. Thật vậy, giả sử có phụ thuộc hàm W[redacted] của F không thoả mãn trên r. Như vậy W[redacted]$^+$ vì nếu

không sẽ vi phạm tính bằng nhau của W trên hai bộ t1 và t2. Hơn nữa V không thể là tập con của X^+ bởi vì nếu V là tập con của X^+ thì W sẽ thoả mãn trên r. Vậy phải có ít nhất một thuộc tính A của V là không thuộc X^+. Theo bổ đề 1. 1, nếu W X^+ thì X . Do W nên X . Do A là một thuộc tính của V nên X , hay A thuộc X^+. Điều đó là vô lý bởi vì A không thuộc X^+. Như vậy, mọi phụ thuộc hàm của F là thoả mãn trên r.

Bây giờ ta chứng tỏ rằng X không thoả mãn trên r. Thật vậy, giả sử ngược lại X thoả mãn trên r. Như vậy cả X và Y đều phải thuộc X^+ vì nếu không sẽ vi phạm sự bằng nhau trên các bộ t1 và t2 của X và Y. Nhưng nếu Y thuộc X^+ thì X sẽ suy diễn được từ F theo bổ đề 3. 1. Điều đó mâu thuẫn với giả thiết X không suy diễn được từ F. Vậy X không thể thoả mãn trên r. Định lý được chứng minh.

3.2.4. Bao đóng và khóa

Để ý rằng nếu X^+ là tập tất cả các thuộc tính của quan hệ thì có nghĩa là X xác định hàm các thuộc tính còn lại, hay nói cách khác X là một siêu khóa. Chúng ta có thể kiểm tra xem một tập thuộc tính X có phải là khóa của một quan hệ bằng cách trước tiên xem X^+ có chứa tất cả các thuộc tính của quan hệ hay không sau đó kiểm tra không có một tập con S nào được lập từ X bằng cách loại bỏ một thuộc tính của X thỏa mãn S^+ chứa tất cả các thuộc tính của quan hệ (nghĩa là X là siêu khóa tối thiểu). Ví dụ:

Xét lược đồ quan hệ R(A, B, C, D, E, F) và tập phụ thuộc hàm

F = {AB ; A D; B }

Ta có $\{A, B\}^+ = \{A, B, C, D, E, F\}$, $A^+ = \{A, C, D\}$, $B^+ = \{B, E\}$, vậy AB là khóa của quan hệ.

Thuật toán 3. 2 Tìm một khóa K của R(U) dựa trên tập F các phụ thuộc hàm.

1) Đặt K := U;

2) Với mỗi thuộc tính A trong K, lặp lại các bước sau:

- tính $(K-A)^+$ đối với F ;

- Nếu $(K-A)^+$ chứa tất cả các thuộc tính trong U thì đặt K := K-{A } ;

Thuật toán 3. 2 cho phép chúng ta xác định được các khoá của một quan hệ xuất phát từ một siêu khoá ban đầu là tập tất cả các thuộc tính của quan hệ. Có thể thấy rằng việc tính khoá như vậy rất mất thời gian bởi vì nếu quan hệ có n thuộc tính thì nó có 2^n

tập con. Nếu khoá của quan hệ chỉ có ít thuộc tính thì số lần tính các bao đóng để kiểm tra là rất lớn. Trên thực tế, người ta tìm khoá của quan hệ dựa trên nhận xét sau: Nếu quan hệ có khóa thì các thuộc tính khóa của quan hệ phải là các tập con của tập hợp các thuộc tính ở vế trái các phụ thuộc hàm trong F. Vì vậy, để tìm được các khóa nhanh hơn, trước tiên chúng ta tính L_F là hợp của các thuộc tính ở các vế trái của các phụ thuộc hàm trong F, sau đó đi tính bao đóng của tất cả các tập con của L_F. Nếu bao đóng của tập con nào chứa tất cả các thuộc tính của R thì tập đó là một siêu khóa. Để kiểm tra nó là một khóa ta thực hiện như bước 2) của thuật toán trên.

3.2.5. Tính tương đương của các tập phụ thuộc hàm

Trong phần này chúng ta thảo luận về sự tương đương của hai tập phụ thuộc hàm. Một tập hợp các phụ thuộc hàm E được phủ bởi một tập các phụ thuộc hàm F - hoặc F phủ E - nếu mỗi một phụ thuộc hàm trong E đều ở trong F^+, điều đó có nghĩa là mỗi phụ thuộc hàm trong E có thể suy diễn được từ F. Hai tập phụ thuộc hàm E và F là tương đương nếu $E^+ = F^+$. Như vậy tương đương có nghĩa là mỗi phụ thuộc hàm trong E có thể suy diễn được từ F và mỗi phụ thuộc hàm trong F có thể suy diễn được từ E.

Cho hai tập phụ thuộc hàm E và F. Để chứng minh hai tập phụ thuộc hàm này tương đương, ta phải chứng minh các phụ thuộc hàm của E đều suy ra được từ F và ngược lại các phụ thuộc hàm của F đều suy ra được từ E. Để chứng minh phụ thuộc hàm X[redacted]suy ra được từ tập phụ thuộc hàm F chúng ta có thể thực hiện theo hai cách:

a. Áp dụng các quy tắc suy diễn để biến đổi các phụ thuộc hàm trong F cho đến khi nhận được X[redacted].

b. Áp dụng bổ đề 3. 1, tính X^+ (bao đóng của tập thuộc tính ở vế trái). Nếu X^+ [redacted] thì X [redacted]suy ra được từ F

Ví dụ : Xét hai tập phụ thuộc hàm

F = {A [redacted] AC [redacted], E[redacted]D, E [redacted]}

E = { A [redacted]D, E [redacted]H }

+ Ta chứng minh hai tập phụ thuộc hàm này là tương đương theo cách a.

Chứng minh E phủ F:

E = { A [redacted]D, E [redacted]H }

={A [redacted], A [redacted], E [redacted], E [redacted]} (áp dụng QT4 – chiếu)

= {A [redacted], A[redacted], E [redacted], E[redacted], E [redacted] (áp dụng QT3, bắc cầu)

= {A [redacted], A [redacted], E [redacted]D, E [redacted]} (áp dụng QT5 – hợp)

= { AC [redacted], A [redacted], E [redacted]D, E [redacted]} (áp dụng QT2)

= { AC [redacted], E [redacted]D, E [redacted], A[redacted]} (áp dụng QT3)

Chứng minh F phủ E:

F= {A [redacted] AC [redacted], E[redacted]D, E [redacted]}

= {A[redacted], A [redacted], E [redacted], E [redacted], E [redacted]} (QT4, QT6)

= {A [redacted]D, E [redacted]H} (vì E [redacted], A [redacted] lên có thể bỏ E [redacted])

+ Ta chứng minh hai tập phụ thuộc hàm này là tương đương theo cách b.

Chứng minh F phủ E:

Tìm bao đóng của các vế trái của các phụ thuộc hàm trong E theo F. Áp dụng thuật toán 3. 1 ở trên, ta có

$\{A\}^+$ = { A, C, D };

$\{E\}^+$ = {E, A, D, H},

ta thấy các bao đóng này chứa các vế phải tương ứng. Từ đó suy ra F phủ E.

C*hứng minh E phủ F* :

Tìm bao đóng của các vế trái của các phụ thuộc hàm trong F theo E. Ta có

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

{AC}+ = { A, C, D},

{E}+ = { E, A, H}, ta thấy các bao đóng này chứa các vế phải tương ứng. Từ đó suy ra E phủ F.

Như vậy, E tương đương với F.

3.2.6. Các tập phụ thuộc hàm tối thiểu

Một tập phụ thuộc hàm là *tối thiểu* nếu nó thoả mãn các điều kiện sau đây:

1. Vế phải của các phụ thuộc hàm trong F chỉ có một thuộc tính

2. Chúng ta không thể thay thế bất kỳ một phụ thuộc hàm X [redacted] trong F bằng phụ thuộc hàm Y[redacted] trong đó Y là tập con đúng của X mà vẫn còn là một tập phụ thuộc hàm tương đương với F.

3. Chúng ta không thể bỏ đi bất kỳ phụ thuộc hàm nào ra khỏi F mà vẫn có một tập phụ thuộc hàm tương đương với F

Chúng ta có thể nghĩ về tập tối thiểu các phụ thuộc hàm như là một tập hợp ở dạng chuẩn không có sự dư thừa. Điều kiện 1 đảm bảo rằng mỗi phụ thuộc hàm là ở dạng chính tắc với một thuộc tính ở vế phải. Điều kiện 2 và 3 đảm bảo rằng không có sự dư thừa trong các phụ thuộc hoặc do có các thuộc tính dư thừa ở vế trái của phụ thuộc, hoặc do có một phụ thuộc có thể được suy diễn từ các phụ thuộc khác ở trong F.

Một *phủ tối thiểu* của một tập phụ thuộc hàm F là một tập tối thiểu các phụ thuộc hàm F_{min} tương đương với F. Thường có rất nhiều các phủ tối thiểu cho một tập các phụ thuộc hàm. Chúng ta luôn luôn có thể tìm được ít nhất là một phủ tối thiểu G cho một tập các phụ thuộc hàm F bất kỳ theo thuật toán 3. 3 sau đây:

Thuật toán 3. 3 (Tìm phủ tối thiểu G cho F).

1. Đặt G := F;

2. Thay thế mỗi phụ thuộc hàm X [redacted] A_1, A_2, . . ., A_n} trong G bằng n phụ thuộc hàm X [redacted] $_1$, X [redacted] $_2$, …, X [redacted] $_n$

3. Với mỗi phụ thuộc hàm X [redacted] trong G và với mỗi thuộc tính B là một phần tử của X

nếu (G - (X [redacted])) [redacted] (X [redacted] B}) [redacted]) là tương đương với G

thì thay thế X [redacted] bằng (X-{B}) [redacted] ở trong G

4. Với mỗi phụ thuộc hàm X [redacted] còn lại trong G

nếu (G [redacted] X [redacted] }) là tương đương với G thì loại bỏ X [redacted] ra khỏi G.

Ví dụ áp dụng : Tìm phủ tối thiểu G cho tập phụ thuộc hàm:

F= { A [redacted] C, B [redacted] C, C [redacted] B }

Bước 1: G = { A [redacted] C, B [redacted] C, C [redacted] B }

Bước 2: G= {A [redacted] A [redacted] B [redacted] B [redacted] C [redacted] C [redacted]

Bước 3: Do các phụ thuộc hàm trong G đều có vế trái gồm một thuộc tính nên G vẫn giữ nguyên.

Bước 4: Loại bỏ các phụ thuộc hàm thừa:

1) Do A [redacted] và B [redacted] nên A [redacted] là thừa. Do C [redacted] và B [redacted] nên C [redacted] là thừa. Bỏ những phụ thuộc hàm thừa đi, ta có {A [redacted] B [redacted] B [redacted] C [redacted] là một phủ tối thiểu

2) Do A[redacted] và B[redacted] nên A[redacted] là thừa. Do có B[redacted] và C[redacted] nên B[redacted] là thừa. Do có C[redacted] và A[redacted] nên C[redacted] là thừa. Bỏ những phụ thuộc hàm thừa đi, ta nhận được một phủ tối thiểu khác là {A[redacted] B[redacted] C[redacted].

CHƯƠNG 4. THIẾT KẾ CƠ SỞ DỮ LIỆU QUAN HỆ

Sau khi đã nghiên cứu các phụ thuộc hàm và một số tính chất của chúng, bây giờ chúng ta sẽ sử dụng chúng như thông tin về ngữ nghĩa của các lược đồ quan hệ. Ta giả sử rằng mỗi một quan hệ được cho trước một tập các phụ thuộc hàm và mỗi quan hệ có một khoá chính. Trong phần này chúng ta sẽ nghiên cứu các dạng chuẩn, quá trình chuẩn hoá các lược đồ quan hệ và thiết kế cơ sở dữ liệu quan hệ.

Có hai cách chính để thiết kế cơ sở dữ liệu quan hệ:

Cách thứ nhất là *thiết kế trên-xuống* (top-down design). Đây là cách hay được sử dụng nhất trong thiết kế ứng dụng cơ sở dữ liệu thương mại. Nó bao gồm việc thiết kế một lược đồ quan niệm trong một mô hình dữ liệu bậc cao, chẳng hạn như mô hình E-R, sau đó ánh xạ lược đồ quan niệm vào một tập quan hệ sử dụng các thủ tục ánh xạ. Mỗi một quan hệ được phân tích dựa trên các phụ thuộc hàm và các khóa chính được chỉ định bằng cách áp dụng các thủ tục chuẩn hóa để loại bỏ các phụ thuộc hàm bộ phận và các phụ thuộc hàm bắc cầu. Việc phân tích các phụ thuộc không mong muốn cũng có thể được thực hiện trong quá trình thiết kế quan niệm bằng cách phân tích các phụ thuộc hàm giữa các thuộc tính bên trong các kiểu thực thể và các kiểu liên kết để ngăn ngừa sự cần thiết có sự chuẩn hóa phụ thêm sau khi việc ánh xạ được thực hiện.

Cách thứ hai là *thiết kế dưới-lên* (bottom-up design), một kỹ thuật tiếp cận và nhìn nhận việc thiết kế lược đồ cơ sở dữ liệu quan hệ một cách chặt chẽ trên cơ sở các phụ thuộc hàm được chỉ ra trên các thuộc tính của cơ sở dữ liệu. Sau khi người thiết kế chỉ ra các phụ thuộc, người ta áp dụng một thuật toán chuẩn hóa để tổng hợp các lược đồ quan hệ. Mỗi một lược đồ quan hệ riêng rẽ ở dạng chuẩn 3NF hoặc BCNF hoặc ở dạng chuẩn cao hơn.

Trước tiên, chúng ta trình bày cách tiếp cận thứ hai. Chúng ta sẽ định nghĩa các dạng chuẩn một cách tổng quát, sau đó trình bày các thuật toán chuẩn hóa và các kiểu phụ thuộc khác. Chúng ta cũng sẽ trình bày chi tiết hơn về hai tính chất cần có là tách không mất mát và tách bảo toàn phụ thuộc. Các thuật toán chuẩn hóa thường bắt đầu bằng việc tổng hợp một lược đồ quan hệ rất lớn, gọi là *quan hệ phổ quát*

(universal relation), chứa tất cả các thuộc tính của cơ sở dữ liệu. Sau đó chúng ta thực hiện lặp đi lặp lại việc tách (decomposition) dựa trên các phụ thuộc hàm và các phụ thuộc khác do người thiết kế cơ sở dữ liệu chỉ ra cho đến khi không còn tách được nữa hoặc không muốn tách nữa.

Tiếp theo, chúng ta trình bày về cách tiếp cận thứ nhất để thiết kế cơ sở dữ liệu. Đi từ các lược đồ khái niệm đến lược đồ vật lý.

4. 1. Nhập môn về chuẩn hoá

Quá trình chuẩn hoá (do Codd đề nghị 1972) là xét một lược đồ quan hệ và thực hiện một loạt các kiểm tra để xác nhận nó có thoả mãn một dạng chuẩn nào đó hay không. Quá trình này được thực hiện theo phương pháp trên xuống bằng việc đánh giá mỗi quan hệ với tiêu chuẩn của các dạng chuẩn và tách các quan hệ nếu cần. Quá trình này có thể xem như là việc thiết kế quan hệ bằng phân tích. Lúc đầu, Codd đề nghị ba dạng chuẩn gọi là dạng chuẩn 1, dạng chuẩn 2 và dạng chuẩn 3. Một định nghĩa mạnh hơn cho dạng chuẩn 3 gọi là dạng chuẩn Boyce-Codd do Boyce và Codd đề nghị sau đó. Tất cả các dạng chuẩn này dựa trên các phụ thuộc hàm giữa các thuộc tính của một quan hệ. Sau đó, dạng chuẩn 4 (4NF) và dạng chuẩn 5 (5NF) được đề nghị dựa trên các phụ thuộc hàm đa trị và các phụ thuộc hàm nối.

Chuẩn hoá dữ liệu có thể được xem như một quá trình phân tích các lược đồ quan hệ cho trước dựa trên các phụ thuộc hàm và các khoá chính của chúng để đạt đến các tính chất mong muốn :

(1) Cực tiểu sự dư thừa và

(2) Cực tiểu các phép cập nhật bất thường.

Các lược đồ quan hệ không thoả mãn các kiểm tra dạng chuẩn sẽ được tách ra thành các lược đồ quan hệ nhỏ hơn thoả mãn các kiểm tra và có các tính chất mong muốn. Như vậy, thủ tục chuẩn hoá cung cấp cho người thiết kế cơ sở dữ liệu:

a. 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.

b. Một loạt các kiểm tra dạng chuẩn có thể thực hiện trên các lược đồ quan hệ riêng rẽ sao cho cơ sở dữ liêu quan hệ có thể được chuẩn hoá đến một mức cần thiết.

Dạng chuẩn của một quan hệ liên quan đến điều kiện dạng chuẩn cao nhất mà nó thoả mãn. Các dạng chuẩn khi được xem xét độc lập với các sự kiện khác không

đảm bảo một thiết kế cơ sở dữ liệu tốt. Nói chung, việc xác minh riêng biệt từng lược đồ quan hệ ở dạng chuẩn này dạng chuẩn nọ là chưa đủ. Tốt hơn là quá trình chuẩn hoá thông qua phép tách phải khẳng định một vài tính chất hỗ trợ mà tất cả các lược đồ quan hệ phải có. Chúng gồm hai tính chất sau:

- Tính chất nối không mất mát (hoặc nối không phụ thêm), đảm bảo rằng vấn đề tạo ra các bộ giả không xuất hiện đối với các lược đồ quan hệ được tạo ra sau khi tách.
- Tính chất bảo toàn 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.

Tính chất nối không mất mát là rất quan trọng, phải đạt được bằng mọi cách, còn tính chất bảo toàn phụ thuộc thì cũng rất mong muốn nhưng đôi khi có thể hy sinh.

4.2. Định nghĩa tổng quát các dạng chuẩn.

Nói chung, chúng ta muốn thiết kế các lược đồ của chúng ta sao cho chúng không còn các phụ thuộc bộ phận và các phụ thuộc bắc cầu bởi vì các kiểu phụ thuộc này gây ra các sửa đổi bất thường. Trong phần này chúng ta sẽ đưa ra các định nghĩa về các dạng chuẩn tổng quát hơn, có tính đến tất cả các khóa dự tuyển. Cụ thể, *thuộc tính khóa* được định nghĩa lại *là một bộ phận của một khóa dự tuyển*. Các phụ thuộc hàm bộ phận, đầy đủ, bắc cầu bây giờ sẽ được định nghĩa đối với tất cả các khóa dự tuyển của quan hệ.

4.2.1. Định nghĩa dạng chuẩn 1 (First Normal Form - 1NF)

Định nghĩa: Một lược đồ quan hệ R là ở dạng chuẩn 1 (1NF) nếu miền giá trị của các thuộc tính của nó chỉ chứa các *giá trị nguyên tử* (đơn, không phân chia được) và giá trị của một thuộc tính bất kỳ trong một bộ giá trị phải là một giá trị đơn thuộc miền giá trị của thuộc tính đó.

Ví dụ: Thực thể: Hóa đơn bán hàng

SoHD	Ngay	HoTen	DiaChi	MatHang	SoLuong	DonGia
HDB01	01/02/2006	Nguyen A	100 HQV	Chuot	01	90000
				B.Phim	01	70000

- MatHang, SoLuong, DonGia: Là những thuộc tính không đơn trị.
- Khả năng lưu trữ là khó khăn, xử lý phức tạp
- Giải pháp: Tách bảng thành nhiều bảng con

HÓAĐƠN:

SoHD	Ngay	HoTen	DiaChi
HDB01	01/02/2006	Nguyen A	100 HQV

CHITIẾTHÓAĐƠN

SoHD	MatHang	SoLuong	DonGia
HDB01	Chuot	01	90000
HDB01	B.Phim	01	70000

4.2.2. Định nghĩa dạng chuẩn 2 (Second Normal Form - 2NF)

Định nghĩa: Một lược đồ quan hệ R là ở dạng chuẩn 2 (2NF) nếu mỗi thuộc tính không khóa A trong R không phụ thuộc bộ phận vào một khóa bất kỳ của R.

Ví dụ: Xét lược đồ quan hệ

R={A, B, C, D, E, F}

Với các phụ thuộc hàm:

A [redacted]CDEF;

BC [redacted]DEF;

B [redacted]

D[redacted]

Lược đồ trên có hai khóa dự tuyển là A và BC. Ta chọn A làm khóa chính. Do có phụ thuộc hàm B [redacted] nên F phụ thuộc bộ phận vào khóa B, C, lược đồ vi phạm chuẩn 2NF. (Chú ý rằng, trong định nghĩa dạng chuẩn dựa trên khóa chính, lược đồ này không vi phạm 2NF).

Ví dụ về CHITIẾTBÁNHÀNG có phụ thuộc hàm {SoHD, MaH} [redacted] SoLuong, DonGia}, {MaH} [redacted] TenHang, DonVi}. Lược đồ có khóa chính là {SoHD, MaH}. Do có TenHang, Donvi là thuộc tính không khóa phụ thuộc một phần vào thuộc tính khóa.

SoHD	MaH	TenHang	SoLuong	DonGia	DonVi
HDB01	HH01	B. Phím	10	100000	Cái

Tách thành hai bảng: CHITIẾTBÁN và HÀNGHÓA

CHI TIẾT BÁN

SoHD	MaH	SoLuong	DonGia
HDB01	HH01	10	100000

HÀNG HÓA

MaH	TenHang	DonVi
HH01	B. Phím	Cái

4.2.3. Định nghĩa dạng chuẩn 3 (Third Normal Form - 3NF)

Định nghĩa: Một lược đồ quan hệ R là ở dạng chuẩn 3 (3NF) nếu khi một phụ thuộc hàm X [redacted] thỏa mãn trong R, thì

1) Hoặc X là một siêu khóa của R
2) Hoặc A là một thuộc tính khóa của R

Ví dụ : Xét lược đồ quan hệ R ở ví dụ trên. Giả sử nó được tách thành hai lược đồ

R1 = { A, B, C, D, E }

R2 = { B, F}.

Do có phụ thuộc hàm D [redacted] trong đó D không phải thuộc tính khóa, E cũng không phải là thuộc tính khóa, nên R1 vi phạm chuẩn 3NF. Do đó R1 sẽ tách thành hai quan hệ sau: R11={A,B,C,D}, R12={D,E}.

4.2.4. Định nghĩa dạng chuẩn Boyce - Codd (Boyce - Codd Normal Form – BCNF)

Định nghĩa: Một lược đồ quan hệ là ở dạng chuẩn Boyce-Codd (BCNF) nếu khi một phụ thuộc hàm X [redacted] thỏa mãn trong R thì X là một siêu khóa của R.

Ví dụ: Xét lược đồ R = {A, B, C, D} có A là khóa chính và {B, C} là khóa dự tuyển. Nếu có tồn tại một phụ thuộc hàm D [redacted] thì lược đồ này vi phạm BCNF vì B là một thuộc tính khóa. (Chú ý rằng trong trường hợp định nghĩa dạng chuẩn dựa trên khóa chính, lược đồ này không vi phạm BCNF).

4. 3 . Phép tách các lược đồ quan hệ

Tách quan hệ: Các thuật toán thiết kế cơ sở dữ liệu quan hệ được trình bày trong phần này bắt đầu từ một lược đồ quan hệ phổ quát R = {A_1, A_2, …, A_n} chứa tất cả các thuộc tính của cơ sở dữ liệu. Với giả thiết quan hệ phổ quát, tên của mỗi thuộc tính là duy nhất. Tập hợp F các phụ thuộc hàm thỏa mãn trên các thuộc tính của R do những người thiết kế cơ sở dữ liệu chỉ ra sẽ được các thuật toán sử dụng. Sử dụng các phụ thuộc hàm, các thuật toán sẽ tách lược đồ quan hệ phổ quát R thành một tập hợp các lược đồ quan hệ D = {R_1, R_2, …, R_m}, tập hợp đó sẽ là lược đồ cơ sở dữ liệu quan hệ. D được gọi là một phép tách của R. Như vậy:

Phép tách một lược đồ quan hệ R = {A_1, A_2, …, A_n} là thay thế nó bằng một tập các lược đồ con D = {R_1, R_2, …, R_m}, trong đó các R_i với i=1, 2, . . m là các tập con của R và R_1 [redacted] R_m = R.

4.3.2.1. Phép tách và sự bảo toàn phụ thuộc

Việc mỗi phụ thuộc hàm X [redacted] trong F hoặc được xuất hiện trực tiếp trong một trong các lược đồ quan hệ R_i trong phép tách D hoặc có thể được suy diễn từ các phụ thuộc hàm có trong R_i là rất có lợi. Ta gọi đó là *điều kiện bảo toàn phụ thuộc*. Chúng ta muốn bảo toàn phụ thuộc bởi vì mỗi phụ thuộc trong F biểu thị một ràng buộc trong cơ sở dữ liệu. Bây giờ chúng ta định nghĩa các khái niệm này một cách hình thức.

Cho trước một tập hợp các phụ thuộc F trên R, *phép chiếu của F trên R_i*, ký hiệu là [redacted](F) trong đó R_i là một tập con của R, là một tập hợp các phụ thuộc hàm X[redacted] trong F^+ sao cho các thuộc tính trong X [redacted] đều được chứa trong R_i. Như vậy, phép chiếu của F trên mỗi lược đồ quan hệ R_i trong phép tách D là tập hợp các phụ thuộc hàm trong F^+, bao đóng của F, sao cho các thuộc tính ở vế trái và vế phải của chúng đều ở trong R_i. Ta nói rằng phép tách D = {R1, R2, …, Rm} của R bảo toàn phụ thuộc đối với F nếu hợp của các phép chiếu của F trên mỗi R_i trong D là tương đương với F. Điều đó có nghĩa là

$((\pi_{R_1}(F))\bowtie\pi_{R_2}(F))\bowtie\ldots\bowtie\pi_{R_m}(F))^+ = F^+$

Nếu một phép tách là không bảo toàn phụ thuộc, một vài phụ thuộc sẽ bị mất trong phép tách. Để kiểm tra xem một phụ thuộc hàm X→B, trong đó X là tập thuộc tính thuộc về R_i, B là một thuộc tính thuộc R_i có thỏa mãn trong R_i hay không ta làm như sau: Trước hết tính X^+, sau đó với mỗi thuộc tính B sao cho:

1. B là một thuộc tính của R_i
2. B là ở trong X^+
3. B không ở trong X

Khi đó phụ thuộc hàm X→B thỏa mãn trong R_i.

Một ví dụ về phép tách không bảo toàn phụ thuộc.

Xét lược đồ quan hệ R = { A, B, C, D},

với các phụ thuộc hàm A→CD ; BC→A ; D→B

Lược đồ này có hai khóa dự tuyển là A và BC. Giả sử nó được tách thành

R1 = { D, B}, lược đồ này chứa phụ thuộc hàm D→B

R2 = { A, C, D }, lược đồ này chứa phụ thuộc hàm A→D.

Rõ ràng sau khi tách, phụ thuộc hàm BC→A bị mất.

Định lý 4. 1: Luôn luôn tìm được một phép tách bảo toàn phụ thuộc D đối với F sao cho mỗi quan hệ R_i trong D là ở 3NF. Phép tách D đợc thực hiện theo thuật toán sau đây:

Thuật toán 4. 1: Tạo một phép tách bảo toàn phụ thuộc D = {R_1, R_2, …, R_m} của một quan hệ phổ quát R dựa trên một tập phụ thuộc hàm F sao cho mỗi R_i trong D là ở 3NF. Thuật toán này chỉ đảm bảo tính chất bảo toàn phụ thuộc, không đảm bảo tính chất nối không mất mát.

Input: Một quan hệ vũ trụ R và một tập phụ thuộc hàm F trên các thuộc tính của R

1) Tìm phủ tối thiểu G của F
2) Với mỗi vế trái X của một phụ thuộc hàm xuất hiện trong G, hãy tạo một lược đồ trong D với các thuộc tính { X→A_1}→A_2}→. . .→A_k}} trong đó X→A_1, X→A_2, …, X→A_k chỉ là các phụ thuộc hàm trong G với X là vế trái (X là khóa của quan hệ này).

3) Đặt các thuộc tính còn lại (những thuộc tính chưa được đặt vào quan hệ nào) vào một quan hệ đơn để đảm bảo tính chất bảo toàn thuộc tính.

Ví dụ áp dụng:

Xét lược đồ : R = { A, B, C, D}, với các phụ thuộc hàm

F = {A CD ; BC A ; D ;

Lược đồ này có hai khóa dự tuyển là A và BC.

Ta thực hiện thuật toán như sau:

Bước 1) Trước tiên ta tìm G là phủ tối thiểu của F. Theo thuật toán tìm phủ tối thiểu, đầu tiên ta làm cho các vế phải trong G chỉ chứa một thuộc tính, ta có:

G = { A ; A ; A ; BC ; BC ; D }

Sau đó ta bỏ đi các phụ thuộc hàm thừa (là các phụ thuộc hàm có thể suy diễn được từ các phụ thuộc hàm khác).

Ta thấy A là thừa vì có A D

BC là thừa vì BC và A .

Vậy G còn lại là G = {A ; A ; BC ; D }.

Bước 2) Ghép các phụ thuộc hàm có cùng vế trái vào lược đồ con. Lược đồ R sẽ được tách thành: R_1(A, C, D); R2(B, C, A); R3(D, B) với các khóa chính được gạch dưới.

Định lý 4. 2. Thuật toán tách lược đồ R = {A_1, A_2, …, A_n} thành tập các lược đồ con R_i, (i = 1, 2, . . m) ở 3NF và bảo toàn phụ thuộc.

Chứng minh:

Rõ ràng rằng tất cả các phụ thuộc hàm trong G đều được thuật toán bảo toàn bởi vì mỗi phụ thuộc xuất hiện trong một trong các quan hệ của phép tách D. Bởi vì G tương đương với F, tất cả các phụ thuộc của F cũng được bảo toàn hoặc trực tiếp bằng thuật toán hoặc được suy diễn từ những phụ thuộc hàm trong các quan hệ kết quả, như vậy tính chất bảo toàn phụ thuộc được đảm bảo. Do G là phủ tối thiểu nên trong G không có những phụ thuộc hàm bộ phận và phụ thuộc hàm bắc cầu, do vậy các phụ thuộc hàm trong G đều là phụ thuộc hàm trực tiếp, điều đó có nghĩa là các R_i đều ở 3NF

4.3.2.2 Phép tách không mất mát thông tin (Lossless join)

Phép tách D phải có một tính chất nữa là tách không mất mát (hoặc tính chất nối không phụ thêm), nó đảm bảo rằng không có các bộ giả được tạo ra khi áp dụng một phép nối tự nhiên vào các quan hệ trong phép tách.

Một cách hình thức, ta nói rằng một phép tách $D = \{R_1, R_2, \ldots, R_m\}$ của R có tính chất không mất mát thông tin đối với một tập hợp phụ thuộc hàm F trên R nếu với mỗi trạng thái quan hệ r của R thỏa mãn F thì

$$(\pi_1(r) * \pi_2(r) * \ldots\ldots * \pi_m(r)) = r$$

trong đó * là phép nối tự nhiên của các quan hệ trong D, $\pi_i(r)$ là phép chiếu của r trên R_i.

Nếu một phép tách không có tính chất không mất mát thông tin thì chúng ta có thể nhận được các bộ phụ thêm (các bộ giả) sau khi áp dụng các phép chiếu và nối tự nhiên. Nghĩa của từ mất mát ở đây là mất mát thông tin chứ không phải mất các bộ giá trị. Vì vậy, với tính chất này ta nên gọi chính xác hơn là tính chất nối không phụ thêm.

Chúng ta có thuật toán để kiểm tra một phép tách có tính chất nối không mất mát thông tin hay không như sau:

Thuật toán 4.2: Kiểm tra tính chất nối không mất mát

Input: Một quan hệ vũ trụ $R(A_1, A_2, \ldots A_n)$, một phép tách $D = \{R_1, R_2, \ldots, R_m\}$ của R và một tập F các phụ thuộc hàm.

1) Tạo một ma trận S có m hàng, n cột. Mỗi cột của ma trận ứng với một thuộc tính, mỗi hàng ứng với mỗi quan hệ R_i

2) Đặt S(i, j) = 1 nếu thuộc tính A_j thuộc về quan hệ R_i và bằng 0 trong trường hợp ngược lại

3) Lặp lại vòng lặp sau đây cho đến khi nào việc thực hiện vòng lặp không làm thay đổi S: Với mỗi phụ thuộc hàm X $\rightarrow$ trong F, xác định các hàng trong S có các ký hiệu 1 như nhau trong các cột ứng với các thuộc tính trong X. Nếu có một hàng trong số đó chứa 1 trong các cột ứng với thuộc tính Y thì hãy làm cho các cột tương ứng của các hàng khác cũng chứa 1.

4) Nếu có một hàng chứa toàn ký hiệu "1" thì phép tách có tính chất nối không mất mát; ngược lại phép tách không có tính chất đó.

Định lý 4.3. Thuật toán 4.2 là đúng.

Chứng minh:

Cho trước một quan hệ R được tách thành một số quan hệ R_1, R_2, . ., R_m. Thuật toán 4.2 bắt đầu bằng việc tạo ra một trạng thái quan hệ r trong ma trận S. Hàng i trong S biểu diễn một bộ t_i (tương ứng với quan hệ R_i). Hàng này có các ký hiệu "1" trong các cột tương ứng với các thuộc tính của R_i và các ký hiệu "0" trong các cột còn lại. Như vậy, $t_i[R_i]$ bao gồm các giá trị 1. Sau đó thuật toán biến đổi các hàng của ma trận này (trong vòng lặp của bước 3) sao cho chúng biểu diễn các bộ thỏa mãn tất cả các phụ thuộc hàm trong F. Ở cuối vòng lặp áp dụng các phụ thuộc hàm, hai hàng bất kỳ trong S – chúng biểu diễn hai bộ trong r – có các giá trị giống nhau đối với các thuộc tính của X ở vế trái của phụ thuộc hàm X[redacted] trong F sẽ cũng có các giá trị giống nhau đối với các thuộc tính của vế phải Y. Có thể thấy rằng sau khi áp dụng vòng lặp của bước 3, nếu một hàng bất kỳ trong S kết thúc với toàn ký hiệu "1" thì điều đó có nghĩa là [redacted](r) * [redacted](r)*……* [redacted](r)) = r hay là D có tính chất tách không mất mát đối với F. Mặt khác, nếu không có hàng nào kết thúc bằng tất cả ký hiệu "1" thì D không thỏa mãn tính chất tách không mất mát. Trong trường hợp sau, trạng thái quan hệ r được biểu diễn bằng S ở cuối thuật toán sẽ là một ví dụ về một trạng thái quan hệ r của R thỏa mãn các phụ thuộc trong F nhưng không thỏa mãn điều kiện tách không mất mát. Như vậy, quan hệ này được dùng như một phản ví dụ chứng minh rằng D không có tính chất tách không mất mát đối với F. Chú ý rằng các ký hiệu "1" và "0" không có ý nghĩa đặc biệt gì ở cuối thuật toán.

Ví dụ áp dụng 1:

R = (MãsốNV, TênNV, MãsốDA, TênDA, ĐịađiểmDA, Sốgiờ)

R1= (TênNV, ĐịađiểmDA)

R2 = (MãsốNV, MãsốDA, Sốgiờ, TênDA, ĐịađiểmDA)

F= {MãsốNV[redacted]ênNV, MãsốDA [redacted]TênDA, ĐịađiểmDA}, {MãsốNV, MãsốDA}[redacted]ốgiờ }

	MãsốNV	TênNV	Mã sốDA	TênDA	ĐịađiểmDA	Sốgiờ
R1		1			1	
R2	1		1	1	1	1

Xét lần lượt phụ thuộc hàm MãsốNV[redacted]nNV, MãsốDA[redacted]ênDA, ĐịađiểmDA}, {MãsốNV, Mã sốDA}[redacted]ốgiờ. Ta thấy không có trường hợp nào các thuộc tính tương ứng với các vế trái đều có giá trị bằng 1, vì vậy ta không thể làm gì

để biến đổi ma trận. Ma trận không chứa một hàng gồm toàn ký hiệu "1". Phép tách là mất mát.

Ví dụ áp dụng 2

R = (MãsốNV, TênNV, MãsốDA, TênDA, ĐịađiểmDA, Sốgiờ)

R1= (MãsốNV, TênNV)

R2 = { MãsốDA, TênDA, ĐịađiểmDA }

R3 = (MãsốNV, MãsốDA, Sốgiờ)

F= {MãsốNV [redacted]ênNV, MãsốDA [redacted]TênDA, ĐịađiểmDA}, {MãsốNV, Mã sốDA} [redacted]ốgiờ }

	MãsốNV	TênNV	Mã sốDA	TênDA	ĐịađiểmDA	Sốgiờ
R1	1	1	0	0	0	0
R2	0	0	1	1	1	0
R3	1	0	1	0	0	1

(Giá trị ban đầu của ma trận S)

Xét phụ thuộc hàm MãsốNV [redacted]ênNV. Ta thấy hàng thứ 1 và hàng thứ 3 của S có chứa 1 tại MãsốNV, trong đó hàng thứ 1 chứa 1 tại TênNV nên ta sửa lại giá trị của TênNV ở hàng thứ 3 thành 1.

Xét phụ thuộc hàm MãsốDA [redacted]ênDA, ĐịađiểmDA}, ta thấy hàng thứ 2 và hàng thứ 3 của S chứa 1 tại MãsốDA trong đó hàng thứ hai chứa 1 tại TênDA và ĐịađiểmDA nên ta sửa giá trị của các thuộc tính đó tại hàng thứ 3 thành 1.

MãsốNV	TênNV	Mã sốDA	TênDA	ĐịađiểmDA	Sốgiờ
1	1	0	0	0	0
0	0	1	1	1	0
1	a	1	a	a	1

(Ma trận S sau khi áp dụng hai phụ thuộc hàm đầu tiên dòng cuối cùng chứa toàn ký hiệu "a"). Ma trận chứa một hàng gồm toàn ký hiệu 1. Phép tách này là không mất mát.

Hình 4. 1. Thuật toán kiểm tra nối không mất mát

Thuật toán 4.2 cho phép chúng ta kiểm tra xem một phép tách D cụ thể có tuân theo tính chất nối không mất mát hay không. Câu hỏi tiếp theo là liệu có một thuật toán tách một lược đồ quan hệ phổ quát $R = \{ A_1, A_2, \ldots, A_n \}$ bằng một phép tách $D = \{R_1, R_2, \ldots, R_m\}$ sao cho mỗi R_i là ở BCNF và phép tách D có tính chất nối không mất mát đối với F hay không?. Câu trả lời là có. Trước khi trình bày thuật toán, ta xem một số tính chất của các phép tách nối không mất mát nói chung:

Tính chất 1: Một phép tách $D = \{R_1, R_2\}$ của R có tính chất không mất mát thông tin đối với một tập phụ thuộc hàm F trên R khi và chỉ khi

- Hoặc phụ thuộc hàm $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ ở trong F^+
- Hoặc phụ thuộc hàm $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ ở trong F^+

Với tính chất này, chúng ta có thể kiểm tra lại các phép tách chuẩn hóa trong 4.2 và sẽ thấy rằng các phép tách đó là thỏa mãn tính chất nối không mất mát.

Tính chất 2 : Nếu một phép tách $D = \{R_1, R_2, \ldots, R_m\}$ của R có tính chất nối không mất mát đối với một tập phụ thuộc hàm F trên R và nếu một phép tách $D_1 = \{Q_1, Q_2, \ldots, Q_k\}$ của R_i có tính chất nối không mất mát đối với phép chiếu của F trên Ri thì phép tách $D2 = \{ R_1, R_2, \ldots, R_{i-1}, Q_1, Q_2, \ldots, Q_k, R_{i+1}, \ldots, R_m\}$ của R có tính chất nối không mất mát đối với F.

Tính chất này nói rằng nếu một phép tách D đã có tính chất nối không mất mát đối với một tập F và chúng ta tiếp tục tách một trong các quan hệ Ri trong D thành phép tách khác D_1 (l = 1, 2, . . k) có tính chất nối không mất mát đối với $\pi_i(F)$ thì việc thay R_i trong D bằng D_1 (l = 1, 2, . . k) cũng tạo ra một phép tách có tính chất nối không mất mát đối với F.

Thuật toán 4.3 sau đây sử dụng hai tính chất trên để tạo ra một phép tách D = $\{R_1, R_2, \ldots, R_m\}$ của một quan hệ vũ trụ R dựa trên một tập các phụ thuộc hàm F sao cho mỗi R_i là BCNF.

Thuật toán 4.3. Tách quan hệ thành các quan hệ BCNF với tính chất nối không mất mát

Input: Một quan hệ vũ trụ R và một tập hợp các phụ thuộc hàm F trên các thuộc tính của R

1. Đặt D := {R} ;
2. Khi có một lược đồ quan hệ Q trong D không phải ở BCNF, thực hiện vòng lặp: Với mỗi một lược đồ quan hệ Q trong D không ở BCNF hãy tìm một phụ thuộc hàm X[redacted] trong Q vi phạm BCNF và thay thế Q trong D bằng hai lược đồ quan hệ (Q-Y) và (X[redacted]). Quá trình lặp dừng khi không còn quan hệ nào trong D vi phạm BCNF.

Mỗi lần đi vào vòng lặp trong thuật toán 4.3, chúng ta tách một quan hệ Q không phải BCNF thành hai lược đồ quan hệ. Theo các tính chất 1 và 2, phép tách D có tính chất nối không mất mát. Kết thúc thuật toán, tất cả các quan hệ trong D sẽ ở BCNF.

Trong bước 2 của thuật toán 4.3, cần xác định xem một lược đồ quan hệ Q có ở BCNF hay không. Một phương pháp để làm điều đó là kiểm tra. Với mỗi phụ thuộc hàm X [redacted] trong Q, ta tính X^+. Nếu X^+ không chứa tất cả các thuộc tính trong Q thì X [redacted] vi phạm BCNF bởi vì X không phải là một siêu khóa.

Một kỹ thuật nữa dựa trên quan sát rằng khi một lược đồ quan hệ Q vi phạm BCNF thì có tồn tại một cặp thuộc tính A, B trong Q sao cho { Q- {A, B}}[redacted]. Bằng việc tính bao đóng $\{Q - \{A, B\}\}^+$ cho mỗi cặp thuộc tính {A, B} của Q và kiểm tra xem bao đóng có chứa A (hoặc B) hay không, chúng ta có thể xác định được Q có ở BCNF hay không.

Ví dụ áp dụng: Xét lược đồ quan hệ

R = { A, B, C, D, E, F)

Với các phụ thuộc hàm:

A [redacted]CDEF,

BC [redacted] DEF,

B[redacted]

D[redacted],

D[redacted]

Lược đồ quan hệ này có hai khóa dự tuyển là A và BC.

Ta có B [redacted] vi phạm BCNF vì B không phải là siêu khóa, R được tách thành

R1(B, F) với phụ thuộc hàm B[redacted]

R2(A, B, C, D, E) với các phụ thuộc hàm A[redacted]CDE, BC[redacted]DE, D[redacted], D[redacted] B

Do D[redacted] vi phạm BCNF (D là một thuộc tính không khóa), R2 được tách thành:

R21(D, E) với phụ thuộc hàm D [redacted]

R22(ABCD) với các phụ thuộc hàm A [redacted]CD, BC[redacted]D, D[redacted]

Do D [redacted] vi phạm BCNF (D không phải là thuộc tính khóa), R22 được tách thành

R221(D, B)

R222(A, C, D) với phụ thuộc hàm A[redacted]D (phụ thuộc hàm BC [redacted]D bị mất)

Tóm lại, ta có phép tách D = {R1, R21, R221, R222}. Phép tách này có tính chất nối không mất thông tin nhưng không bảo toàn phụ thuộc. Nếu chúng ta muốn có một phép tách có tính chất nối không mất mát và bảo toàn phụ thuộc thì ta phải hài lòng với các lược đồ quan hệ ở dạng 3NF. Thuật toán sau đây là cải tiến của thuật toán 4.3, tạo ra một phép tách thỏa mãn :

- Bảo toàn phụ thuộc
- Có tính chất nối không mất mát
- Mỗi lược đồ quan hệ kết quả là ở dạng 3NF.

Thuật toán 4.4 Thuật toán tổng hợp quan hệ với tính chất bảo toàn phụ thuộc và không mất mát

Input: Một quan hệ vũ trụ R và một tập các phụ thuộc hàm F trên các thuộc tính của R

1) Tìm phủ tối thiểu G cho F

2) Với mỗi vế trái X của một phụ thuộc hàm xuất hiện trong G hãy tạo ra một lược đồ quan hệ trong D với các thuộc tính {XA_1}A_2}A_k}}, trong đó X, X, …, X$_k$ chỉ là các phụ thuộc hàm ở trong G với X là vế trái (X là khóa của quan hệ này)
3) Nếu không có lược đồ quan hệ nào trong D chứa một khóa của R thì hãy tạo ra thêm một lược đồ quan hệ trong D chứa các thuộc tính tạo nên một khóa của R.

Không phải lúc nào cũng có khả năng tìm được một phép tách thành các lược đồ quan hệ bảo toàn phụ thuộc và mỗi lược đồ trong phép tách là ở BCNF. Các lược đồ quan hệ trong phép tách theo thuật toán ở trên thường là 3NF. Để có các lược đồ BCNF, chúng ta có thể kiểm tra các lược đồ quan hệ 3NF trong phép tách một cách riêng rẽ để xem nó có thỏa mãn BCNF không. Nếu có lược đồ quan hệ R_i không ở BCNF thì ta có thể tách tiếp hoặc để nguyên nó là 3NF.

4. 4. Các phụ thuộc hàm đa trị và dạng chuẩn 4 (4NF - Fourth Normal Form)

Trong phần này chúng ta thảo luận khái niệm phụ thuộc hàm đa trị và định nghĩa dạng chuẩn 4. Các phụ thuộc đa trị hệ quả của dạng chuẩn 1 không cho phép một thuộc tính của một bộ có một tập giá trị (nghĩa là các thuộc tính đa trị). Nếu chúng ta có hai hoặc nhiều hơn các thuộc tính đôc lập và đa trị trong cùng một lược đồ quan hệ thì chúng ta phải lặp lại mỗi một giá trị của một trong các thuộc tính với mỗi giá trị của thuộc tính khác để giữ cho trạng thái quan hệ nhất quán và duy trì tính độc lập giữa các thuộc tính. Ràng buộc đó được chỉ ra bằng một phụ thuộc đa trị.

4.4.1. Định nghĩa phụ thuộc đa trị

Giả thiết có một lược đồ quan hệ R, X và Y là hai tập con của R. Một phụ thuộc đa trị, ký hiệu là X, chỉ ra ràng buộc sau đây trên một trạng thái quan hệ bất kỳ của R: Nếu hai bộ t_1 và t_2 tồn tại trong R sao cho $t_1[X] = t_2[X]$ thì hai bộ t_3 và t_4 cũng tồn tại trong R với các tính chất sau:

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$ và $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$ và $t_4[Z] = t_1[Z]$ với Z = (R- (X))

Khi X thỏa mãn, ta nói rằng X đa xác định Y. Bởi vì tính đối xứng trong định nghĩa, khi X thỏa mãn trong R, X cũng thỏa mãn trong R. Như vậy X kéo theo X và vì thế đôi khi nó được viết là X Z

Định nghĩa hình thức chỉ ra rằng, cho trước một giá trị cụ thể của X, tập hợp các giá trị của Y được xác định bởi giá trị này của X là được xác định hoàn toàn bởi một mình X và không phụ thuộc vào các giá trị của các thuộc tính còn lại Z của R. Như vậy, mỗi khi hai bộ tồn tại có các giá trị khác nhau của Y nhưng cùng một giá trị X thì các giá trị này của Y phải được lặp lại trong các bộ riêng rẽ với mỗi giá trị khác nhau của Z có mặt với cùng giá trị của X. Điều đó tương ứng một cách không hình thức với Y là một thuộc tính đa trị của các thực thể được biểu diễn bằng các bộ trong R.

Ví dụ về phụ thuộc đa trị:

NHÂNVIÊN	TênNV	TênDA	TênconNV
	Nam	DA01	Lan
	Nam	DA02	Hoa
	Nam	DA01	Hoa
	Nam	DA02	Lan

Trong bảng trên có hai phụ thuộc đa trị là TênNV ênDA, TênNV TênconNV

Một phụ thuộc đa trị X được gọi phụ thuộc đa trị tầm thường nếu

a) Y là một tập con của X
b) Hoặc X = R

Một phụ thuộc đa trị không thỏa mãn a) hoặc b) được gọi là một phụ thuộc đa trị không tầm thường. Nếu chúng ta có một phụ thuộc đa trị không tầm thường trong một quan hệ, chúng ta có thể phải lặp các giá trị một cách dư thừa trong các bộ. Trong quan hệ NHÂNVIÊN ở ví dụ trên, các giá trị 'DA01', 'DA02' của TênDA được lặp lại với mỗi giá trị của TênconNV (một cách đối xứng, các giá trị 'Lan', 'Hoa' được lặp lại với mỗi giá trị của TênDA). Rõ ràng ta không mong muốn có sự dư thừa đó. Tuy nhiên, lược đồ quan hệ trên là ở BCNF bởi vì không có phụ thuộc hàm nào thỏa mãn trong quan hệ đó. Vì vậy, chúng ta phải định nghĩa một dạng chuẩn thứ tư mạnh hơn BCNF và ngăn cấm các lược đồ quan hệ như quan hệ NHÂNVIÊN.

4.4.2. Các quy tắc suy diễn đối với các phụ thuộc hàm và phụ thuộc đa trị

Các quy tắc từ qt1 đến qt8 sau đây tạo nên một tập hợp đúng đắn và đầy đủ cho việc suy diễn các phụ thuộc hàm và phụ thuộc đa trị từ một tập các phụ thuộc cho trước. Giả thiết rằng tất cả các thuộc tính được chứa trong một lược đồ quan hệ "vũ trụ" R = {A_1, A_2, …, A_n} và X, Y, Z, W là các tập con của R. (FD ký hiệu phụ thuộc hàm, MVD ký hiệu phụ thuộc đa trị)

Qt1) (quy tắc phản xạ cho FD): Nếu X [redacted] thì X [redacted]

Qt2) (quy tắc tăng cho FD): { X [redacted] |= XZ [redacted]Z

Qt3) (Quy tắc bắc cầu cho FD): { X [redacted], Y [redacted] } |= X[redacted]

Qt4) (quy tắc bù cho MVD): {X [redacted]} |= {X [redacted]R-(X [redacted]))}

Qt5) (quy tắc tăng cho MVD): Nếu X [redacted]và W [redacted] thì WX [redacted]Z

Qt6) (quy tắc bắc cầu cho MVD): { X [redacted], Y [redacted] } |= X [redacted] – Y)

Qt7) (quy tắc tái tạo cho FD và MVD): {X [redacted] |= X [redacted]

Qt8) (quy tắc liên hợp cho FD và MVD): Nếu X [redacted]và có tồn tại W với các tính chất

a) W [redacted] = ∅,

b) W [redacted]và

c) Y [redacted]

thì X [redacted]

Qt1 đến Qt3 là các quy tắc suy diễn Amstrong đối với các phụ thuộc hàm. Qt4 đến Qt6 là các quy tắc suy diễn chỉ liên quan đến các phụ thuộc đa trị. Qt7 và Qt8 liên kết các phụ thuộc hàm và các phụ thuộc đa trị. Đặc biệt, Qt7 nói rằng một phụ thuộc hàm là một trường hợp đặc biệt của một phụ thuộc đa trị. Điều đó có nghĩa là mỗi phụ thuộc hàm cũng là một phụ thuộc đa trị bởi vì nó thỏa mãn định nghĩa hình thức của phụ thuộc đa trị. Về cơ bản, một phụ thuộc hàm X [redacted]là một phụ thuộc đa trị X [redacted]với một hạn chế phụ rằng có nhiều nhất là một giá trị của Y được kết hợp với mỗi giá trị của X. Cho trước một tập hợp các phụ thuộc hàm và phụ thuộc đa trị chỉ ra trên R = { A1, A2, …, An}, chúng ta có thể sử dụng các quy tắc từ qt1 đến qt8 để suy ra tập hợp đầy đủ các phụ thuộc (hàm và đa trị) F^+ đúng trong mọi trạng thái quan hệ r của R thỏa mãn F. Chúng ta lại gọi F^+ là bao đóng của F.

4.4.3. Dạng chuẩn 4

Định nghĩa dạng chuẩn 4: Một lược đồ quan hệ R là ở dạng chuẩn 4 (4NF) đối với một tập hợp các phụ thuộc F (gồm các phụ thuộc hàm và phụ thuộc đa trị) nếu với mỗi phụ thuộc đa trị không tầm thường X[redacted]trong F^+, X là một siêu khóa của R.

Như vậy, một lược đồ quan hệ vi phạm 4NF nếu nó chứa các phụ thuộc hàm đa trị không mong muốn. Ví dụ, lược đồ quan hệ NHÂNVIÊN ở ví dụ trên là vi phạm 4NF bởi vì trong các phụ thuộc hàm đa trị TênNV[redacted]nDA và TênNV[redacted] TênconNV, TênNV không phải là một siêu khóa.

Giả sử chúng ta tách bảng NHÂNVIÊN thành hai bảng như sau:

NV_DA	TênNV	TênDA
	Nam	DA01
	Nam	DA02

NV_CON	TênNV	TênconNV
	Nam	Lan
	Nam	Hoa

Hai bảng này là ở 4NF bởi vì các phụ thuộc đa trị TênNV[redacted]nDA và TênNV [redacted]ênconNV là các phụ thuộc đa trị tầm thường. Trong hai bảng này không có các phụ thuộc đa trị không tầm thường cũng như không có các phụ thuộc hàm.

4.4.4. Tách có tính chất nối không mất mát thành các quan hệ chuẩn 4NF

Khi chúng ta tách một lược đồ quan hệ R thành $R_1 = (X$[redacted]) và $R_2 = (R-Y)$ dựa trên phụ thuộc hàm đa trị X[redacted]đúng trong R, phép tách có tính chất nối không mất mát. Đó cũng là điều kiện cần và đủ cho một phép tách một lược đồ thành hai lược đồ có tính chất nối không mất mát. Ta có tính chất sau:

Tính chất 1': Các lược đồ quan hệ R_1 và R_2 tạo thành một phép tách có tính chất nối không mất mát của R khi và chỉ khi (R_1[redacted]$_2$)[redacted]$_1 - R_2$) (hoặc (R_1[redacted]$_2$) [redacted] ($R_1 - R_2$)).

Áp dụng tính chất trên chúng ta có thuật toán tạo một phép tách có tính chất nối không mất mát thành các lược đồ quan hệ ở dạng 4NF.

Thuật toán 4.5 Tách quan hệ thành các quan hệ 4NF với tính chất nối không mất mát.

Input: Một quan hệ vũ trụ R, một tập phụ thuộc hàm và phụ thuộc đa trị F

1. Đặt D := {R} ;
2. Khi có một lược đồ quan hệ Q trong D không ở 4NF, thực hiện

{Chọn một lược đồ quan hệ Q trong D không ở 4NF;

Tìm một phụ thuộc đa trị không tầm thường X[redacted]trong Q vi phạm 4NF;

Thay thế Q trong D bằng hai lược đồ quan hệ (Q – Y) và (X [redacted]); }.

Ví dụ áp dụng 1:

Xét lược đồ NHÂNVIÊN(TênNV, TênDA, TênconNV). Ta có phụ thuộc hàm đa trị TênNV[redacted]nDA trong đó TênNV không phải là một siêu khóa, vậy nó vi phạm 4NF. Ta tách thành NV_DA(TênNV, TênDA), NV_CON(TênNV, TênconNV).

Ví dụ áp dụng 2:

Cho quan hệ SẢNXUẤT như sau:

SẢNXUẤT	Phânxưởng	Nhânviên	Sảnphẩm
	Phân xưởng 1	Hoàng	Bu lông
	Phân xưởng 1	Yến	Đinh
	Phân xưởng 1	Hoàng	Đinh
	Phân xưởng 1	Yến	Bu lông
	Phân xưởng 2	Minh	Ốc vít
	Phân xưởng 2	Hải	Kìm điện
	Phân xưởng 2	Minh	Kìm điện
	Phân xưởng 2	Hải	Ốc vít

Trong bảng trên có hai phụ thuộc đa trị là Phânxưởng[redacted]hânviên, Phânxưởng[redacted]nphẩm. Quan hệ SẢNXUẤT vi phạm 4NF bởi vì trong các phụ

thuộc hàm đa trị có Phânxưởng không phải là một siêu khóa. Chúng ta tách quan hệ SẢNXUẤT thành hai quan hệ PX_NV(), PX_SP():

NV_DA	Phânxưởng	Nhânviên
	Phân xưởng 1	Hoàng
	Phân xưởng 1	Yến
	Phân xưởng 2	Minh
	Phân xưởng 2	Hải

NV_CON	Phânxưởng	Sảnphẩm
	Phân xưởng 1	Bu lông
	Phân xưởng 1	Đinh
	Phân xưởng 2	Ốc vít
	Phân xưởng 2	Kìm điện

4. 5. Các phụ thuộc nối và dạng chuẩn 5 (Fifth Normal Form - 5NF)

Như chúng ta đã thấy, các tính chất 1 và tính chất 1' cho điều kiện để một lược đồ quan hệ R được tách thành hai lược đồ quan hệ R1 và R2 và phép tách có tính chất nối không mất mát. Tuy nhiên, trong một số trường hợp, có thể không có phép tách có tính chất nối không mất mát của R thành hai lược đồ quan hệ nhưng có thể có phép tách có tính chất nối không mất mát thành nhiều hơn hai quan hệ. Hơn nữa, có thể không có phụ thuộc hàm nào trong R các chuẩn cho đến BCNF và có thể không có phụ thuộc đa trị nào có trong R vi phạm 4NF. Khi đó chúng ta phải sử dụng đến một phụ thuộc khác gọi là phụ thuộc nối và nếu có phụ thuộc nối thì thực hiện một phép tách đa chiều thành dạng chuẩn 5 (5NF).

Một *phụ thuộc nối,* ký hiệu là JD(R_1, R_2, …, R_n) trên lược đồ quan hệ R chỉ ra một ràng buộc trên các trạng thái r của R. Ràng buộc đó tuyên bố rằng mỗi trạng thái hợp pháp r của R phải có phép tách có tính chất nối không mất mát thành R1, R2, ..., Rn. Điều đó nghĩa là:

*(■$_1$(r), ■$_2$(r), …■$_n$(r)) = r

Một phụ thuộc nối JD(R_1, R_2, …, R_n) là một phụ thuộc nối tầm thường nếu một trong các lược đồ quan hệ R_i ở trong JD(R_1, R_2, …, R_n) là bằng R.

Định nghĩa dạng chuẩn 5: Một lược đồ quan hệ R là ở dạng chuẩn 5 (5NF) (hoặc dạng chuẩn nối chiếu (PJNF- Project-Join normal form) đối với một tấp F các

phụ thuộc hàm, phụ thuộc đa trị và phụ thuộc nối nếu với mỗi phụ thuộc nối không tầm thường JD(R_1, R_2, …, R_n) trong F^+, mỗi Ri là một siêu khóa của R.

Ví dụ 1: Xét quan hệ CUNGCẤP gồm toàn các thuộc tính khóa

CUNGCẤP	**Tênnhàcungcấp**	**Tênhàng**	**TênDựán**
	Ncc1	Bulong	Dựán1
	Ncc1	Đaiốc	Dựán2
	Ncc2	Bulong	Dựán2
	Ncc3	Đaiốc	Dựán3
	Ncc2	Đinh	Dựán1
	Ncc2	Bulong	Dựán1
	Ncc1	Bulong	Dựán2

Giả thiết rằng ràng buộc phụ thêm sau đây luôn đúng: Khi một nhà cung cấp S cung cấp hàng P *và* một dự án J sử dụng hàng P *và* nhà cung cấp S cung cấp ít nhất là một hàng cho dự án J *thì* nhà cung cấp S cũng sẽ cung cấp hàng P cho dự án J. Ràng buộc này chỉ ra một phụ thuộc nối JD(R_1, R_2, R_3) giữa ba phép chiếu R_1(Tênnhàcungcấp, Tênhàng), R_2(Tênnhàcungcấp, Têndựán), R_3(Tênhàng, TênDựán) của quan hệ CUNGCẤP. Quan hệ CUNGCẤP được tách thành ba quan hệ R_1, R_2, R_3 ở dạng chuẩn 5. Chú ý rằng nếu ta áp dụng phép nối tự nhiên cho từng đôi quan hệ một thì sẽ sinh ra các bộ giả, nhưng nếu áp dụng phép nối tự nhiên cho cả ba quan hệ thì không sinh ra các bộ giả.

R1

Tênnhàcungcấp	Tênhàng
Ncc1	Bulong
Ncc1	Đaiốc
Ncc2	Bulong
Ncc3	Đaiốc
Ncc2	Đinh

R2

Tênnhàcungcấp	Têndựán
Ncc1	Dựán1
Ncc1	Dựán2
Ncc2	Dựán2
Ncc3	Dựán3
Ncc2	Dựán1

R3

Tênhàng	Têndựán
Bulong	Dựán1
Đaiốc	Dựán2
Bulong	Dựán2
Đaiốc	Dựán3
Đinh	Dựán1

Ví dụ 2: Cho quan hệ NGOẠIKHÓA

NGOẠIKHÓA	Họcviên	Hoạtđộng	Câulạcbộ

Minh	Khiêu vũ	Clb1
Minh	Chơi ghi ta	Clb2
Ngọc	Đánh trống	Clb1
Ngọc	Hát	Clb3
Hải	Múa	Clb1
Minh	Diễn kịch	Clb3
Hải	Chơi piano	Clb2

Quan hệ NGOẠIKHÓA có các ràng buộc sau: Họcviên X có hoạt động Y, hoạt động Y thuộc câu lạc bộ Z. Học viên X muốn tham gia hoạt động Y thì phải đăng ký ở câu lạc bộ Z. Các ràng buộc này chỉ ra một phụ thuộc nối JD(R1, R2, R3) giữa 3 phép chiếu R1(Họcviên, hoạtđộng), R2(Họcviên, câulạcbộ), R3(câulạcbộ, hoạtđộng) của quan hệ NGOẠIKHÓA. Quan hệ NGOẠIKHÓA được tách thành ba quan hệ R_1, R_2, R_3 ở dạng chuẩn 5.

Họcviên	hoạtđộng
Minh	Khiêu vũ
Minh	Chơi ghi ta
Ngọc	Đánh trống
Ngọc	Hát
Hải	Múa
Minh	Diễn kịch
Hải	Chơi piano

Họcviên	câulạcbộ
Minh	Clb1
Minh	Clb2
Ngọc	Clb1
Ngọc	Clb3
Hải	Clb1
Minh	Clb3
Hải	Clb2

câulạcbộ	hoạtđộng
Clb1	Khiêu vũ
Clb2	Chơi ghi ta
Clb1	Đánh trống
Clb3	Hát
Clb1	Múa
Clb3	Diễn kịch
Clb2	Chơi piano

Việc phát hiện các phụ thuộc nối trong các cơ sở dữ liệu thực tế với hàng trăm thuộc tính là một điều rất khó khăn. Vì vậy, thực tiễn thiết kế cơ sở dữ liệu hiện nay thường không chú ý đến nó. Nói chung, trong thực tế thiết kế cơ sở dữ liệu, người ta chỉ chuẩn hóa các bảng đến 3NF, BCNF là đủ.

4.6. Mô hình dữ liệu

4.6.1. Khái niệm

Mô hình dữ liệu (Data model) là một tập hợp các khái niệm dùng để diễn tả tập hợp dữ liệu và hành động để thao tác lên dữ liệu

Các khái niệm trong mô hình dữ liệu được xây dựng bởi cơ chế trừu tượng hóa và mô tả bằng ngôn ngữ hay biểu diễn đồ họa.

4.6.2. Các mô hình dữ liệu thông dụng

4.6.2.1. Mô hình phân cấp

Mô hình phân cấp (hierarchy) được xây dựng từ thập kỷ 70. Đó chính là một mạng có nhiều cây. Hình 4.2 ví dụ minh họa mô hình phân cấp.

Hình 4.2.. Minh họa mô hình phân cấp

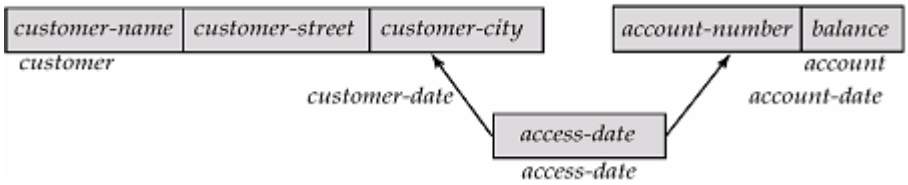
4.6.2.2. Mô hình mạng

Mô hình mạng cũng được xây dựng từ thập kỷ 70. Trong mô hình này, dữ liệu được trình bày dưới dạng một tập các mẫu tin (record). Các mẫu tin và các thuộc tính được biểu diễn bởi kiểu mẫu tin (record type).

```
type  customer = record          type   account = record
      customer-name: string;            account-number: integer;
      customer-street: string;          balance: integer;
      customer-city: string;
end                              end
```

Hình 4.3. Các dạng mẫu tin customer, account.

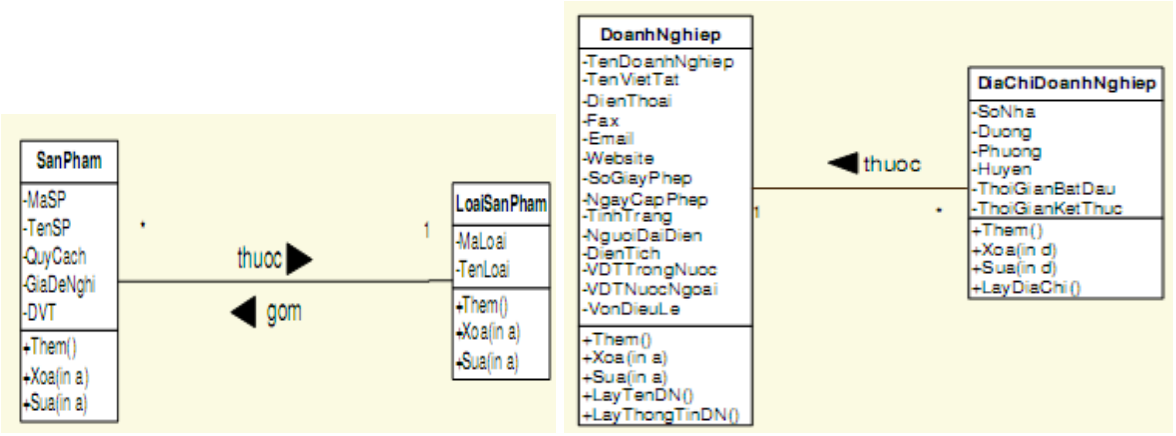
Mối quan hệ giữa các mẫu tin được biểu diễn bằng đường nối (links). Trên đường nối này biểu diễn các mối quan hệ nhiều – nhiều (many – to – many), một – nhiều (one-to-many), một – một (one – to – one).



Hình 4.4. Mối quan hệ của mô hình mạng

4.6.2.3. Mô hình hướng đối tượng

Mô hình hương đối tượng được xây dựng từ giữa thập kỷ 80 – đến nay.



Hình 4.5. Ví dụ về mô hình hướng đối tượng

4.6.3. Chất lượng của mô hình dữ liệu

- Tính diễn đạt: cho phép mô tả một khối lượng lớn đa dạng các khái niệm sao cho có thể biểu diễn toàn diện hơn thế giới thực. Do đó, các mô hình dữ liệu phải giàu về các khái niệm và cả tính diễn đạt.

- Tính đơn giản: mô hình dữ liệu phải đơn giản để cho lược đồ xây dựng bằng mô hình sẽ được những người thiết kế và người sử dụng thông hiểu dễ dàng hơn.

- Tính tốt thiểu: Mô hình sẽ có tính tối thiểu nếu mọi khái niệm trình bày trong mô hình có một ý nghĩa phân biệt khi xem xét trong các mối quan hệ đến mọi khía cạnh khác.

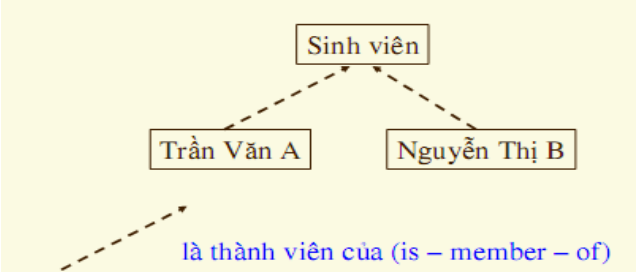
- Tính hình thức: các lược đồ tạo ra bởi các mô hình dữ liệu biểu diễn một đặc tả hình thức dữ liệu. Tính hình thức này đòi hỏi các khái niệm của mô hình sẽ được thể hiện đồng nhất, chính xác và định nghĩa tốt.

- Tính mở rộng: mô hình dữ liệu phải có tính mở cho tương lai.

4.7. Mô hình thực thể kết hợp (Entity Relationship – ER)

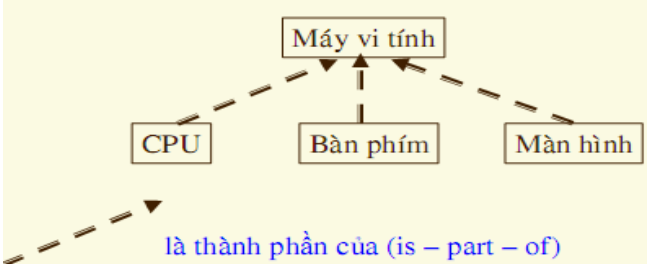
4.7.1. Trừu tượng hóa

- Phân loại:



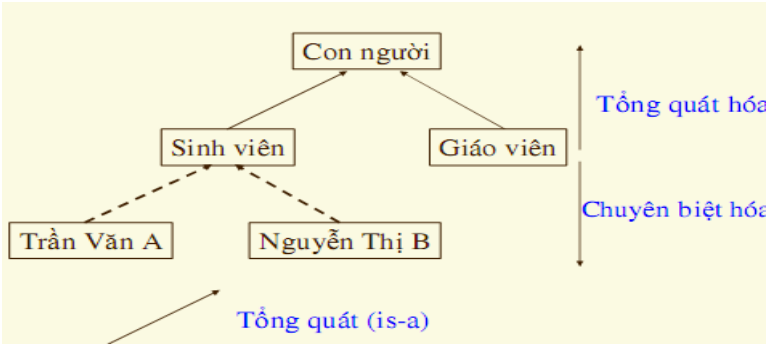
Hình 4.6. Mô hình phân loại

- Kết hợp:



Hình 4.7. Ví dụ minh họa kết hợp

4.7.2. Tổng quát hóa



Hình 4.8. Minh họa tổng quát hóa

4.7.3. Mô hình thực thể kết hợp (ER)

Mô hình thực thể kết hợp:

+ Dễ dùng

+ Hỗ trợ Case tool

+ Dùng để xây dựng mô hình ý niệm

+ Mô hình này ra đời năm 1970 bởi Mr.Chen, tiếp tục được phát triển bởi Teory, Chang và Fry vào năm 1986 và Storey vào năm 1991. Được xem là một chuẩn giúp mô hình hóa dữ liệu.

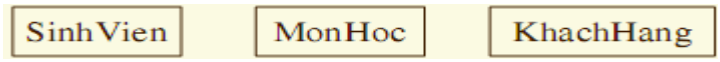
+ Là sự biểu diễn đồ họa các thực thể và các mối liên hệ của chúng trong cấu trúc một database.

4.7.3.1. Các thành phần chính của mô hình ER

a) Thực thể: Là một người, nơi chốn, đối tượng, sự kiện hay một khái niệm trong thế giới thực (có thể là khái niệm trừu tượng) và bắt đầu là danh từ

+ Thực thể mạnh: sự tồn tại độc lập với các kiểu thực thể khác. Thực thể mạnh có thể xác định thực thể yếu qua từ khóa:

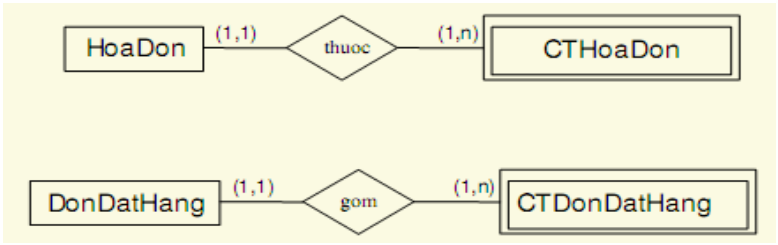
Ví dụ: SinhVien, Monhoc, KhachHang, NhanVien, SanPham, HoaDon



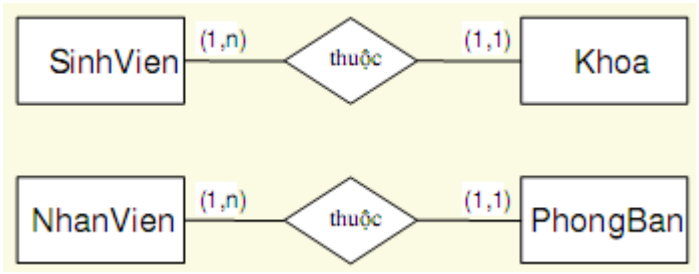
+ Thực thể yếu: Không có đủ các thuộc tính để hình thành nên khóa, tồn tại phụ thuộc vào các thực thể khác. Ký hiệu

- Một tập thực thể yếu có thể có khóa riêng để phân biệt giữa các thực thể yếu có mối liên quan với cùng một thực thể mạnh.
- Khóa của tập thực thể yếu = khóa của tập thuộc tính cha + khóa riêng của tập thực thể yếu

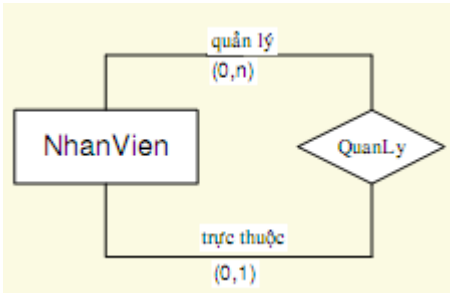
Ví dụ:



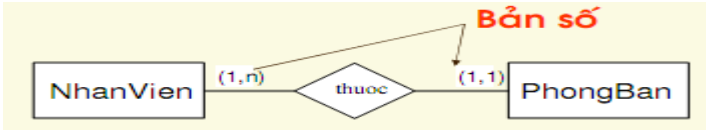
b) Mối kết hợp: Diễn tả mối quan hệ giữa các thực thể. Ký hiệ



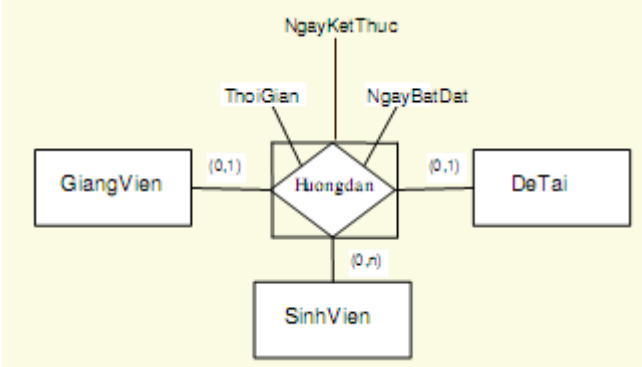
- Mối kết hợp một ngôi (Phản thân)



- Mối kết hợp 2 ngôi (nhị phân)



- Mối kết hợ 3 ngôi (đa phân)

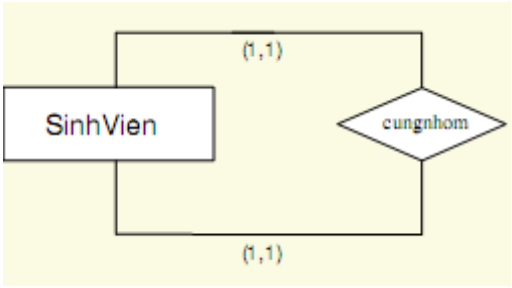


c) Bản số của mối kết hợp

Xét tập quan hệ 2 ngôi giữa hai thực thể A và B:

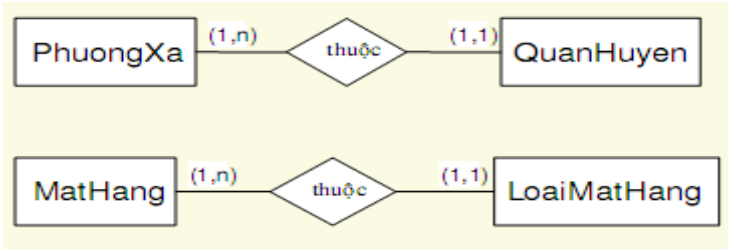
+ Một – một (one-to-one): một thực thể A kết hợp với nhiều nhất một thực thể B và một thực thể B kết hợp với nhiều nhất một thực thể A.

Ví dụ: Một phòng ban có duy nhất 1 trưởng phòng.



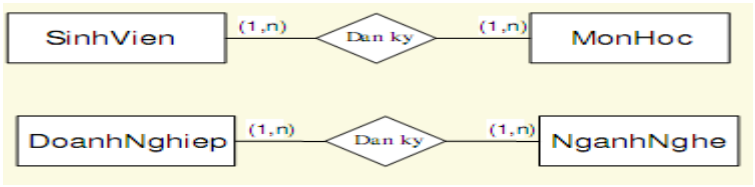
+ Một – nhiều (one to many): Một thực thể A kết hợp với nhiều thực thể trong B và một thực thể B kết hợp với nhiều nhất chỉ một thực thể trong A.

Ví dụ: Một nhân viên chỉ thuộc một phòng ban và một phòng ban có nhiều nhân viên



+ Nhiều – nhiều (many to many): Một thực thể A kết hợp với nhiều thực thể trong B và một thực thể B kết hợp với nhiều thực thể trong A.

Ví dụ: Một sinh viên học nhiều môn học và một môn học có nhiều sinh viên đăng ký học.

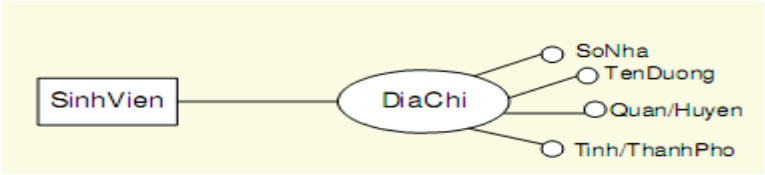


d) Thuộc tính: là đặc tính, tính chất đặc trưng của thực thể hoặc mối kết hợp nhằm để mô tả thực thể hay mối kết hợp.

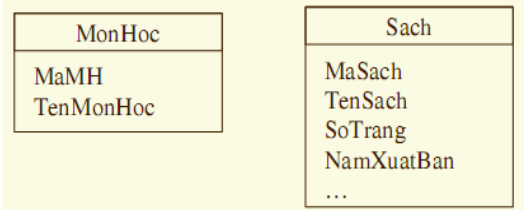
Ví dụ: thực thể SinhVien có các thuộc tính hoten, tuoi, gioitinh, ngaysinh,…

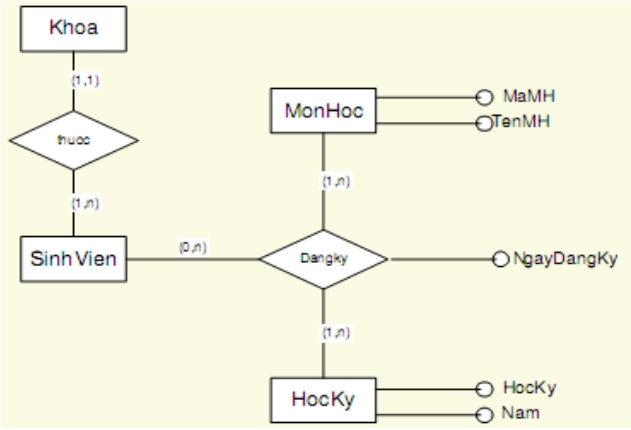
Thuộc tính gồm có các loại:

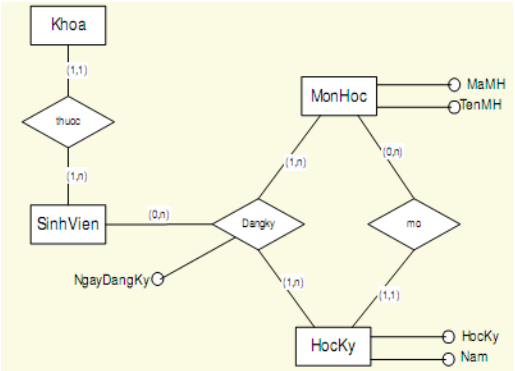
- Thuộc tính đơn trị
- Thuộc tính đa trị
- Thuộc tính kết hợp
- Thuộc tính dẫn xuất

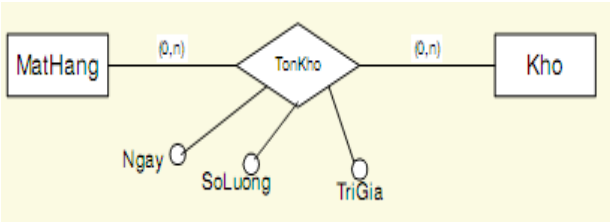


Cách biểu diễn khác









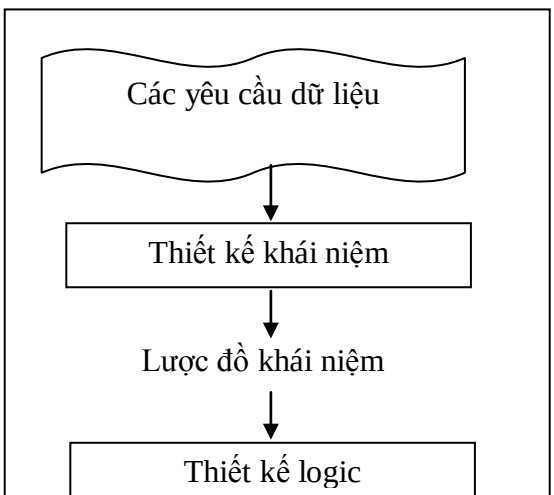
4.7.3.2. Các bước thiết kế mô hình thực thể kết hợp

- Nhận dạng các thực thể.
- Nhận dạng các mối quan hệ.
- Găn kết các thuộc tính vào các thực thể và mối kết hợp.
- Xác định các cấu trúc tiêu biểu (có thể có hoặc không).

4.8. Thiết kế cơ sở dữ liệu

Thiết kế CSDL đòi hỏi một vài quyết định ở nhiều mức khác nhau. Tính phức tạp của công tác này sẽ được quản lý tốt hơn nếu người ta phân rã bài toán thành các bài toán con và giải các bài toán con một cách độc lập bằng cách dùng phương pháp và kỹ thuật đặc biệt. Thiết kế cơ sở dữ liệu được chia ra các thiết khái niệm, logic và vật lý.

Thiết kế cơ sở dữ liệu được thể hiện như tiếp cận chuyển dữ liệu trong việc phát triển hệ thống thông tin bởi lẽ toàn bộ trọng tâm của quá trình thiết kế là ở dữ liệu và các thuộc tính của nó. Với tiếp cận chuyển dữ liệu, trước hết người ta thiết kế cơ sở dữ liệu, rồi đến các ứng dụng sử dụng cơ sở dữ liệu này. Phương pháp này được phát triển sau những năm 70 cùng với việc đề xuất công nghệ cơ sở dữ liệu.



Hình 4.9. Tiếp cận chuyển dữ liệu trong việc thiết kế các hệ thông thông tin

4.8.1. Thiết kế khái niệm

Thiết kế khái niệm bắt đầu từ việc xác định các yêu cầu và kết quả trong lược đồ khái niệm của cơ sở dữ liệu. Lược đồ khái niệm là mô tả mức cao của cấu trúc dữ liệu, độc lập với phần mền quản trị cơ sở dữ liệu cụ thể. Một mô hình khái niệm là ngôn ngữ dùng để mô tả các lược đồ khái niệm. Mục tiêu của thiết kế khái niệm là mô tả nội dung thông tin của cơ sở dữ liệu, chứ không phải là mô tả các cấu trúc lưu trữ do việc quản lý thông tin yêu cầu. Thực tế, thiết kế khái niệm được thực hiện ngay cả khi việc cài đặt, hoàn thiện cuối cùng không sử dụng hệ quản trị cơ sở dữ liệu mà chỉ dùng hệ quản trị tệp và các ngôn ngữ lập trình.

4.8.2. Thiết kế logic

Thiết kế logic bắt đầu từ lược đồ khái niệm và cho ra kết quả là lược đồ logic. Lược đồ logic là mô tả của cấu trúc cơ sở dữ liệu mà hệ quản trị cơ sở dữ liệu xử lý. Một mô hình logic là ngôn ngữ để xác định lược đồ logic; các mô hình logic hay được dùng nhất thuộc về các lớp:

- Mô hình quan hệ,
- Mô hình mạng,
- Mô hình phân cấp,

Mô hình logic phụ thuộc vào lớp của mô hình dữ liệu do hệ quản trị cơ sở dữ liệu dùng, nhưng không phụ thuộc vào một hệ quản trị đặc biệt nào.

Thí dụ: Khi quan tâm đến mô hình quan hệ, người ta tiến hành thiết kế logic theo cùng một cách đối với tất cả các hệ quản trị cơ sở dữ liệu quan hệ bởi vì chúng dùng mô hình quan hệ.

4.8.3. Thiết kế vật lý

Thiết kế vật lý bắt đầu từ lược đồ logic và kết thúc với lược đồ vật lý. Lược đồ vật lý là mô tả cài đặt của cơ sở dữ liệu trên bộ nhớ ngoài: nó mô tả các cấu trúc lưu trữ và các phương pháp truy nhập dùng để truy nhập dữ liệu có hiệu quả. Do đó thiết kế vật lý được sinh ra cho một hệ quản trị cơ sở dữ liệu, và các quyết định trong giai đoạn thiết kế vật lý cũng tác động lại các cấu trúc của lược đồ logic.

Một khi thiết kế cơ sở dữ liệu vật lý đã hoàn tất, các lược đồ vật lý và logic được thể hiện thông qua ngôn ngữ xác định dữ liệu của hệ quản trị cơ sở dữ liệu đích. Các ứng dụng sử dụng cơ sở dữ liệu có thể được hoàn toàn xác định, cài đặt và được thử. Những điều này cho phép một cơ sở dữ liệu dần dần được hình thành.

KẾT LUẬN

Thiết kế cơ sở dữ liệu là một chủ đề quan trọng trong lĩnh vực cơ sở dữ liệu cả về phương diện lý thuyết lẫn thực hành. Kết quả chính của chuyên đề là: Tìm hiểu và nghiên cứu qua tài liệu để hệ thống các vấn đề sau:

1/. Trình bày một số khái niệm cơ bản về CSDL, hệ quản trị CSDL.

2/. Khái niệm mô hình quan hệ: thuộc tính, miền dữ liệu, khóa… và các phép toán đại số quan hệ: kết nối, chiếu, chọn…

3/. Trình bày một số khái niệm cơ bản khái niệm phụ thuộc hàm trong cơ sở dữ liệu quan hệ.

4/. Trình bày các phương pháp chuẩn hóa dữ liệu. Từ đó đưa ra phương pháp thiết kế cơ sở dữ liệu quan hệ.

TÀI LIỆU THAM KHẢO

[1]. Nguyễn Bá Tường, Lý thuyết cơ sở dữ liệu, HVKTQS, 2000

[2]. Nguyễn Bá Tường, Nhập môn cơ sở dữ liệu phân tán, NXB KHKT, 2004

[3]. Bản dịch của Trần Đức Quang, nguyên lý các hệ cơ sở dữ liệu và cơ sở tri thức, NXB thống kê.

[4]. Nguyễn Bá Tường, Cơ sở dữ liệu lý thuyết và thực hành, NXB khoa học và kỹ thuật - 2001

[5]. Đỗ Trung Tuấn, Lý thuyết cơ sở dữ liệu, NXB khoa học và kỹ thuật - 2000

[6]. Lê tiến Vương, Nhập môn cơ sở dữ liệu quan hệ, NXB thống Kê-2000

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

CHƯƠNG I
GIỚI THIỆU
(Introduction)

MỤC ĐÍCH

Chương này trình bày một cái nhìn bao quát về cơ sở dữ liệu (CSDL/DB), về hệ quản trị cơ sở dữ liệu (HQTCSDL/DBMS) và về hệ cơ sở dữ liệu (HCSDL/DBS). Các đòi hỏi khi xây dựng một HQTCSDL đó cũng chính là những chức năng mà một HCSDL cần phải có. Một khái niệm quan trọng là khái niệm giao dịch (Transaction). Các tính chất một giao dịch phải có để đảm bảo một HQTCSDL, được xây dựng trên HCSDL tương ứng, trong suốt quá trình hoạt động sẽ luôn cho một CSDL tin cậy (dữ liệu luôn nhất quán). Quản trị giao dịch nhằm đảm bảo mọi giao dịch trong hệ thống có các tính chất mà một giao dịch phải có. Một điều cần chú ý là trong các tính chất của một giao dịch, *tính chất nhất quán* trước hết phải được đảm bảo bởi người lập trình-người viết ra giao dịch.

YÊU CẦU

Hiểu các khái niệm.

Hiểu các vấn đề đặt ra khi xây dựng một HQTCSDL: thiết kế CSDL, đảm bảo tính nhất quán của CSDL trong suốt cuộc sống của nó, nền tảng phần cứng trên đó một HQTCSDL được xây dựng.

Hiểu cấu trúc hệ thống tổng thể

Hiểu vai trò của các người sử dụng hệ thống.

MỘT SỐ KHÁI NIỆM

- Một cơ sở dữ liệu (CSDL/ DB: DataBase) là một tập hợp các tập tin có liên quan với nhau, được thiết kế nhằm làm giảm thiểu sự lặp lại dữ liệu.
- Một hệ quản trị cơ sở dữ liệu (HQTCSDL/ DBMS: DataBase Management System) là một hệ thống gồm một CSDL và các thao tác trên CSDL đó, được thiết kế trên một nền tảng phần cứng, phần mềm và với một kiến trúc nhất định.
- Một hệ cơ sở dữ liệu (HCSDL/ DBS: DataBase System) là một phần mềm cho phép xây dựng một HQTCSDL.

HỆ CƠ SỞ DỮ LIỆU

Một số điểm bất lợi chính của việc lưu giữ *thông tin có tổ chức* trong hệ thống xử lý file thông thường:

- **Dư thừa dữ liệu và tính không nhất quán** (Data redundancy and inconsistency): Do các file và các trình ứng dụng được tạo ra bởi các người lập trình khác nhau, nên các file có định dạng khác nhau, các chương trình được viết trong các ngôn ngữ lập trình khác nhau, cùng một thông tin có thể được lưu giữ trong các file khác nhau. Tính không thống nhất và dư thừa này sẽ làm *tăng chi phí truy xuất và lưu trữ*, hơn nũa, nó sẽ dẫn đến tính không nhất quán của dữ liệu: *các bản sao của cùng một dữ liệu có thể không nhất quán*.
- **Khó khăn trong việc truy xuất dữ liệu**: Môi trường của hệ thống xử lý file thông thường không cung cấp các công cụ cho phép truy xuất thông tin một cách hiệu quả và thuận lợi.
- **Sự cô lập dữ liệu** (Data isolation): Các giá trị dữ liệu được lưu trữ trong cơ sở dữ liệu phải thoả mãn một số các *ràng buộc về tính nhất quán của dữ liệu (ràng buộc nhất quán/consistency contraints)*. Trong hệ thống xử lý file thông thường, rất khó khăn trong việc thay đổi các chương trình để thoả mãn các yêu cầu thay đổi ràng buộc. Vấn đề trở nên khó khăn hơn khi các ràng buộc liên quan đến các hạng mục dữ liệu nằm trong các file khác nhau.
- **Các vấn đề về tính nguyên tử** (Atomicity problems): Tính nguyên tử của một hoạt động (giao dịch) là: *hoặc nó được hoàn tất trọn vẹn hoặc không có gì cả*. Điều này có nghĩa là một hoạt động (giao dịch) *chỉ làm thay đổi* các dữ liệu bền vững khi nó đã hoàn tất (kết thúc thành công) nếu không, giao dịch không để lại một dấu vết nào trên CSDL. Trong hệ thống xử lý file thông thường khó đảm bảo được tính chất này.
- **Tính bất thường trong truy xuất cạnh tranh**: Một hệ thống cho phép nhiều người sử dụng cập nhật dữ liệu đồng thời, có thể dẫn đến kết quả là dữ liệu không nhất quán. Điều này đòi hỏi một sự giám sát. Hệ thống xử lý file thông thường không cung cấp chức năng này.
- **Vấn đề an toàn** (Security problems): một người sử dụng hệ cơ sở dữ liệu không cần thiết và cũng không có quyền truy xuất tất cả các dữ liệu. Vấn đề này đòi hỏi hệ thống phải đảm bảo được tính phân quyền, chống truy xuất trái phép ...

Các bất lợi nêu trên đã gợi mở sự phát triển các DBMS. Phần sau của giáo trình sẽ đề cập đến các quan niệm và các thuật toán được sử dụng để phát triển một hệ cơ sở dữ liệu nhằm *giải quyết các vấn đề nêu trên*. Một số khái niệm

GÓC NHÌN DỮ LIỆU

Tính hiệu quả của hệ thống đòi hỏi phải thiết kế các cấu trúc dữ liệu phức tạp để biểu diễn dữ liệu trong cơ sở dữ liệu. Các nhà phát triển che dấu sự phức tạp này thông qua các *mức trừu tượng* nhằm đơn giản hóa sự trao đổi của người sử dụng với hệ thống:

- **Mức vật lý (Physical level)**: Mức thấp nhất của sự trừu tượng, mô tả dữ liệu hiện được lưu trữ thế nào. Ở mức này, cấu trúc dữ liệu mức thấp, phức tạp được mô tả chi tiết.
- **Mức luận lý (Logical level)**: Mức kế cao hơn về sự trừu tượng, mô tả dữ liệu gì được lưu trữ trong cơ sở dữ liệu và các mối quan hệ gì giữa các dữ liệu này. Mức logic của sự trừu tượng được dùng bởi các người quản trị cơ sở dữ liệu.
- **Mức view (view level)**: Mức cao nhất của sự trừu tượng, mô tả chỉ một phần của cơ sở dữ liệu toàn thể. Một người sử dụng cơ sở dữ liệu liên quan đến chỉ một bộ phận của cơ sở dữ liệu. Như vậy sự trao đổi của họ với hệ thống được làm đơn giản bởi việc định nghĩa view. Hệ thống có thể cung cấp nhiều mức view đối với cùng một cơ sở dữ liệu.

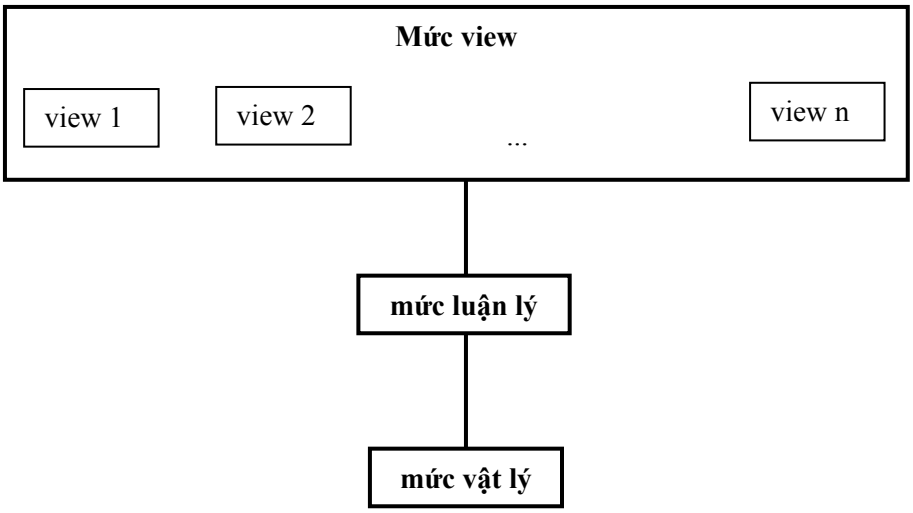


Figure 1

- **Thể hiện và sơ đồ** (instances and schemas): Tập hợp các thông tin được lưu trữ trong cơ sở dữ liệu tại một thời điểm được gọi là một thể hiện (instance) của cơ sở dữ liệu. Thiết kế tổng thể của cơ sở dữ liệu được gọi là sơ đồ (schema).

Một hệ cơ sở dữ liệu có một vài sơ đồ, được phân tương ứng với các mức trừu tượng. ở mức thấp nhất là **sơ đồ vật lý** (physical schema), ở mức trung gian là **sơ đồ luận lý** (logical schema), ở mức cao nhất là **sơ đồ con** (subschema). Nói chung một hệ cơ sở dữ liệu hỗ trợ một sơ đồ vật lý, một sơ đồ luận lý và một vài sơ đồ con.

- Khả năng sửa đổi một định nghĩa ở một mức không ảnh hưởng một định nghĩa sơ đồ ở mức cao hơn được gọi là **sự độc lập dữ liệu** (data independence). Có hai mức độc lập dữ liệu:
  - **Độc lập dữ liệu vật lý** (Physical data independence) là khả năng sửa đổi sơ đồ vật lý không làm cho các chương trình ứng dụng phải viết lại. Các sửa đổi ở mức vật lý là cần thiết để cải thiện hiệu năng.

- **Độc lập dữ liệu luận lý** (Logical data independence) là khả năng sửa đổi sơ đồ luận lý không làm cho các chương trình ứng dụng phải viết lại. Các sửa đổi ở mức luận lý là cần thiết khi cấu trúc luận lý của cơ sở dữ liệu bị thay thế.

MÔ HÌNH DỮ LIỆU

Nằm dưới cấu trúc của một cơ sở dữ liệu là mô hình dữ liệu: một bộ các công cụ quan niệm để mô tả dữ liệu, quan hệ dữ liệu, ngữ nghĩa dữ liệu và các ràng buộc nhất quán. Có ba nhóm mô hình: Các mô hình luận lý dựa trên đối tượng (Object-based logical models), các mô hình luận lý dựa trên mẩu tin (record-based logical models), các mô hình vật lý (physical models).

- **Các mô hình luận lý dựa trên đối tượng** được dùng mô tả dữ liệu ở mức luận lý và mức view. Chúng được đặc trưng bởi việc chúng cung cấp khả năng cấu trúc linh hoạt và cho phép các ràng buộc dữ liệu được xác định một cách tường minh. Dưới đây là một vài mô hình được biết rộng rãi: Mô hình **thực thể - quan hệ** (entity-relationship model), mô hình **hướng đối tượng** (object-oriented model), mô hình **dữ liệu ngữ nghĩa** (semantic data model), mô hình **dữ liệu hàm** (function data model).
- **Các mô hình luận lý dựa trên mẩu tin** được dùng để miêu tả dữ liệu ở mức luận lý hay mức view. Chúng được dùng để xác định cấu trúc luận lý toàn thể của cơ sở dữ liệu và cung cấp sự mô tả mức cao hơn việc thực hiện. Cơ sở dữ liệu được cấu trúc ở dạng mẩu tin định dạng cố định (fixed format record): mỗi mẩu tin xác định một số cố định các trường, mối trường thường có độ dài cố định. Một vài mô hình được biết rộng rãi là: **Mô hình quan hệ, mô hình mạng, mô hình phân cấp**.
- **Mô hình dữ liệu vật lý** được dùng để mô tả dữ liệu ở mức thấp nhất. Hai mô hình dữ liệu vật lý được biết rộng rãi nhất là **mô hình hợp nhất** (unifying model) và **mô hình khung-bộ nhớ** (frame-memory model).

NGÔN NGỮ CƠ SỞ DỮ LIỆU

Một hệ cơ sở dữ liệu cung cấp hai kiểu ngôn ngữ khác nhau: một để xác định sơ đồ cơ sở dữ liệu, một để biểu diễn các vấn tin cơ sở dữ liệu và cập nhật.

- **Ngôn ngữ định nghĩa dữ liệu (Data Definition Language: DDL)** cho phép định nghĩa sơ đồ cơ sở dữ liệu. Kết quả biên dịch các lệnh của DDL là tập hợp các bảng được lưu trữ trong một ***file đặc biệt được gọi là tự điển dữ liệu (data dictionary)*** hay thư mục dữ liệu (***data directory***). Tự điển dữ liệu là một file chứa ***metadata***. File này được tra cứu trước khi dữ liệu hiện hành được đọc hay sửa đổi. Cấu trúc lưu trữ và phương pháp truy cập được sử dụng bởi hệ cơ sở dữ liệu được xác định bởi một tập hợp các định nghĩa trong một kiểu đặc biệt của DDL được gọi là ***ngôn ngữ định nghĩa và lưu trữ dữ liệu (data storage and definition language)***. Kết quả biên dịch của các định nghĩa này là một tập hợp các chỉ thị xác định sự thực hiện chi tiết của các sơ đồ cơ sở dữ liệu (thường được che dấu).
- **Ngôn ngữ thao tác dữ liệu (Data manipulation language: DML)** là ngôn ngữ cho phép người sử dụng truy xuất hoặc thao tác dữ liệu. Có hai kiểu ngôn ngữ thao tác dữ liệu: ***DML thủ tục (procedural DML)*** yêu cầu người sử dụng đặc tả dữ liệu nào cần và làm thế nào để nhận được nó. ***DML không thủ tục (Nonprocedural DML)*** yêu cầu người sử dụng đặc tả dữ liệu nào cần nhưng không cần đặc tả làm thế nào để nhận được nó. Một vấn tin (***query***) là một lệnh yêu cầu tìm lại dữ liệu (***information***

retrieval). Phần ngôn ngữ DML liên quan đến sự tìm lại thông tin được gọi là ngôn ngữ vấn tin (***query language***).

QUẢN TRỊ GIAO DỊCH

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. Đòi hỏi ***toàn thể-hoặc-không*** này được gọi là tính nguyên tử (atomicity). Một yếu tố cần thiết khác là sự thực hiện việc chuyển khoản bảo tồn tính nhất quán của cơ sở dữ liệu: *giá trị của tổng A + B phải được bảo tồn*. Đòi hỏi 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. Đòi hỏi 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 tử 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 mỗi một bảo tồn tính nhất quán của cơ sở dữ liệu.

Đảm bảo tính nguyên tử 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 tử được hoàn thành dễ dàng. Tuy nhiên, do sự hiện diện của các sự cố, một giao dich có thể không hoàn tất thành công sự thực hiện của nó. Nếu tính nguyên tử được đảm bảo, một giao dịch thất bại không gây hiệu quả 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 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 cạnh tranh cập nhật cơ sở dữ liệu, tính nhất quán của dữ liệu có thể không được bảo tồn, ngay cả khi mỗi giao dịch là chính xác. ***Bộ quản trị điều khiển cạnh tranh (concurency-control manager)*** có trách nhiệm điều khiển các trao đổi giữa các giao dịch cạnh tranh để đảm bảo tính thống nhất của CSDL.

QUẢN TRỊ LƯU TRỮ

Các CSDL đòi hỏi một khối lượng lớn không gian lưu trữ, có thể lên đến nhiều terabytes (1 terabyte=10^3 Gigabytes=10^6 Megabytes). Các thông tin phải được lưu trữ trên lưu trữ ngoài (đĩa). Dữ liệu được di chuyển giữa lưu trữ đĩa và bộ nhớ chính khi cần thiết. Do việc di chuyển dữ liệu từ và lên đĩa tương đối chậm so với tốc độ của đơn vị xử lý trung tâm, điều này ép buộc hệ CSDL phải cấu trúc dữ liệu sao cho tối ưu hóa nhu cầu di chuyển dữ liệu giữa đĩa và bộ nhớ chính.

Mục đích của một hệ CSDL là làm đơn giản và dễ dàng việc truy xuất dữ liệu. Người sử dụng hệ thống có thể không cần quan tâm đến chi tiết vật lý của sự thực thi hệ thống. Phần lớn họ chỉ quan đến hiệu năng của hệ thống (thời gian trả lời một câu vấn tin ...).

Bộ quản trị lưu trữ (storage manager) là một module chương trình cung cấp giao diện giữa dữ liệu mức thấp được lưu trữ trong CSDL với các chương trình ứng dụng và các câu vấn tin được đệ trình cho hệ thống. Bộ quản trị lưu trữ có trách nhiệm trao đổi với bộ quản trị file (file manager). Dữ liệu thô được lưu trữ trên đĩa sử dụng hệ thống file (file system), hệ thống này thường được cung cấp bởi hệ điều hành. Bộ quản trị lưu trữ dịch các câu lệnh DML thành các lệnh của hệ thống file mức thấp. Như vậy, *bộ quản trị lưu trữ có nhiệm vụ lưu trữ, tìm lại và cập nhật dữ liệu trong CSDL*.

NHÀ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Một trong các lý do chính đối với việc sử dụng DBMS là có sự điều khiển trung tâm cho cả dữ liệu lẫn các chương trình truy cập các dữ liệu này. Người điều khiển trung tâm trên toàn hệ thống như vậy gọi là nhà quản trị cơ sở dữ liệu (DataBase Administrator - DBA). Các chức năng của DBA như sau:

- ***Định nghĩa sơ đồ***: DBA tạo ra sơ đồ CSDL gốc bằng cách viết một tập các định nghĩa mà nó sẽ được dịch bởi trình biên dịch DDL thành một tập các bảng được lưu trữ thường trực trong tự điển dữ liệu.
- ***Định nghĩa cấu trúc lưu trữ và phương pháp truy xuất***: DBA tạo ra một cấu trúc lưu trữ thích hợp và các phương pháp truy xuất bằng cách viết một tập hợp các định nghĩa mà nó sẽ được dịch bởi trình biên dịch lưu trữ dữ liệu và ngôn ngữ định nghĩa dữ liệu.
- ***Sửa đổi sơ đồ và tổ chức vật lý***
- ***Cấp quyền truy xuất dữ liệu***: Việc cấp các dạng quyền truy cập khác nhau cho phép DBA điều hoà những phần của CSDL mà nhiều người có thể truy xuất. Thông tin về quyền được lưu giữ trong một cấu trúc hệ thống đặc biệt, nó được tham khảo bởi hệ CSDL mỗi khi có sự truy xuất dữ liệu của hệ thống.
- ***Đặc tả ràng buộc toàn vẹn (integrity-contraint)***: Các giá trị dữ liệu được lưu trữ trong CSDL phải thoả mãn một số các ràng buộc nhất quán nhất định. Ví dụ số giờ làm việc của một nhân viên trong một tuần không thể vượt quá một giới hạn 80 giờ chẳng hạn. Một ràng buộc như vậy phải được đặc tả một cách tường minh bởi DBA. Các ràng buộc toàn vẹn được lưu giữ trong một cấu trúc hệ thống đặc biệt được tham khảo bởi hệ CSDL mỗi khi có sự cập nhật dữ liệu.

NGƯỜI SỬ DỤNG CSDL

Mục đích đầu tiên của hệ CSDL là cung cấp một môi trường để tìm lại thông tin và lưu thông tin trong CSDL. Các người sử dụng cơ sở dữ liệu được phân thành bốn nhóm tuỳ theo cách thức họ trao đổi với hệ thống.

- ***Các người lập trình ứng dụng***: Là nhà chuyên môn máy tính người trao đổi với hệ thống thông qua các lời gọi DML được nhúng trong một chương trình được viết trong một ngôn ngữ chủ - *host language* (Pascal, C, Cobol ...). Các chương trình này thường được tham khảo như các chương trình ứng dụng. Vì cú pháp DML thường rất khác với cú pháp của ngôn ngữ chủ, các lời gọi DML thường được bắt đầu bởi một ký tự đặc biệt như vậy mã thích hợp mới có thể được sinh. Một bộ tiền xử lý đặc biệt, được gọi là tiền

biên dịch (precompiler) DML, chuyển các lệnh DML thành các lời gọi thủ tục chuẩn trong ngôn ngữ chủ. Bộ biên dịch ngôn ngữ chủ sẽ sinh mã đối tượng thích hợp. Có những ngôn ngữ lập trình phối hợp cấu trúc điều khiển của các ngôn ngữ giống như Pascal với cấu trúc điều khiển để thao tác đối tượng CSDL. Các ngôn ngữ này (đôi khi được gọi là ngôn ngữ thế hệ thứ tư) thường bao gồm các đặc điểm đặc biệt để làm dễ dàng việc sinh các dạng và hiển thị dữ liệu trên màn hình.

- ***Các người sử dụng thành thạo (Sophisticated users)***: Trao đổi với hệ thống không qua viết trình. Thay vào đó họ đặt ra các yêu cầu của họ trong ngôn ngữ truy vấn CSDL (Database query language). Mỗi câu vấn tin như vậy được đệ trình cho bộ xử lý vấn tin, chức năng của bộ xử lý vấn tin là "dịch" các lệnh DML thành các chỉ thị mà bộ quản trị lưu trữ hiểu. Các nhà phân tích đệ trình các câu vấn tin thăm dò dữ liệu trong cơ sở dữ liệu thuộc vào phạm trù này.
- ***Các người sử dụng chuyên biệt (Specialized users)***: Là các người sử dụng thành thạo, họ viết các ứng dụng CSDL chuyên biệt không nằm trong khung xử lý dữ liệu truyền thống. Trong đó, phải kể đến các hệ thống thiết kế được trợ giúp bởi máy tính (computer-aided design systems), Cơ sở tri thức (knowledge-base) và hệ chuyên gia (expert systems), các hệ thống lưu trữ dữ liệu với kiểu dữ liệu phức tạp (dữ liệu đồ họa, hình ảnh, âm thanh) và các hệ thống mô hình môi trường (environment-modeling systems)
- ***Các người sử dụng ngây thơ (Naive users)***: là các người sử dụng không thành thạo, họ trao đổi với hệ thống bởi cầu dẫn một trong các chương trình ứng dụng thường trực đã được viết sẵn.

CẤU TRÚC HỆ THỐNG TỔNG THỂ

Một hệ CSDL được phân thành các module, mỗi một thực hiện một trách nhiệm trong hệ thống tổng thể. Một số chức năng của hệ CSDL có thể được cung cấp bởi hệ điều hành. Trong hầu hết các trường hợp, hệ điều hành chỉ cung cấp các dịch vụ cơ sở nhất, hệ CSDL phải xây dựng trên cơ sở đó. Như vậy, thiết kế hệ CSDL phải xem xét đến giao diện giữa hệ CSDL và hệ điều hành.

Các thành phần chức năng của hệ CSDL có thể được chia thành các thành phần xử lý vấn tin (query processor components) và các thành phần quản trị lưu trữ (storage manager components).

Các thành phần xử lý vấn tin gồm:

- ***Trình biên dịch DML (DML compiler):*** dịch các lệnh DML trong một ngôn ngữ vấn tin thành các chỉ thị mức thấp mà engine định giá vấn tin (query evaluation engine) có thể hiểu. Hơn nữa, Trình biên dịch DML phải biến đổi một yêu cầu của người sử dụng thành một đích tương đương nhưng ở dạng hiệu quả hơn có nghĩa là tìm một chiến lược tốt để thực hiện câu vấn tin.
- ***Trình tiền biên dịch DML nhúng (Embedded DML Precompiler):*** biến đổi các lệnh DML được nhúng trong một chương trình ứng dụng thành các lời gọi thủ tục chuẩn trong ngôn ngữ chủ. Trình tiền biên dịch phải trao đổi với trình biên dịch DML để sinh mã thích hợp.
- ***Bộ thông dịch DDL (DDL interpreter):*** thông dịch các lệnh DDL và ghi chúng vào một tập hợp các bảng chứa metadata.

- ***Engine định giá vấn tin (Query evaluation engine):*** Thực hiện các chỉ thị mức thấp được sinh ra bởi trình biên dịch DML.

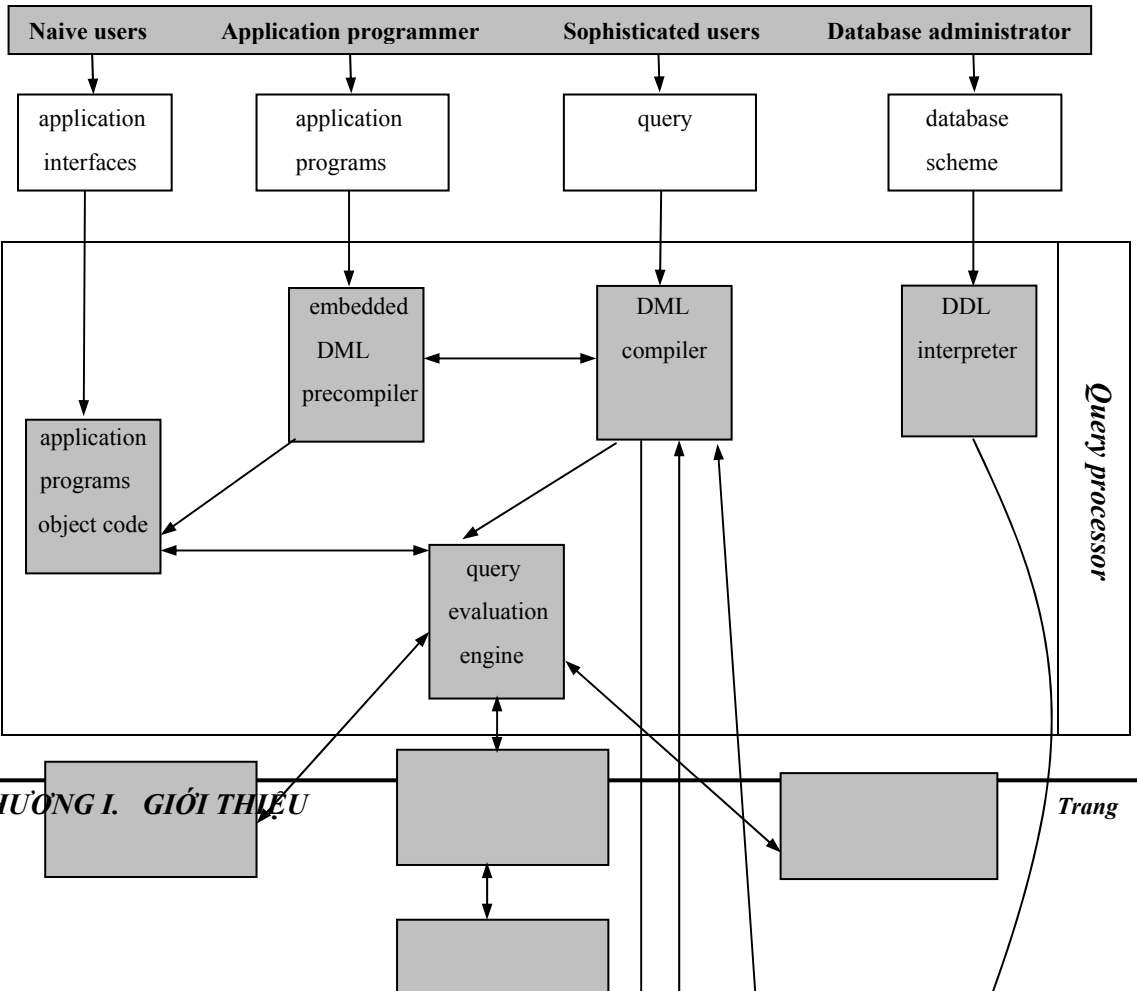
Các thành phần quản trị lưu trữ cung cấp các giao diện giữa dữ liệu mức thấp được lưu trữ trong CSDL và các chương trình ứng dụng, các vấn tin được đệ trình cho hệ thống. Các thành phần quản trị lưu trữ gồm:

- ***Bộ quản trị quyền và tính toàn vẹn (Authorization and integrity manager):*** kiểm tra sự thoả mãn các ràng buộc toàn vẹn và kiểm tra quyền truy xuất dữ liệu của người sử dụng.
- ***Bộ quản trị giao dịch (Transaction manager):*** Đảm bảo rằng CSDL được duy trì trong trạng thái nhất quán cho dù hệ thống có sự cố và đảm bảo rằng các thực hiện giao dịch cạnh tranh tiến triển không xung đột.
- ***Bộ quản trị file (File manager):*** Quản trị cấp phát không gian trên lưu trữ đĩa và các cấu trúc dữ liệu được dùng để biểu diễn thông tin được lưu trữ trên đĩa.
- ***Bộ quản trị bộ đệm (Buffer manager):*** có trách nhiệm đem dữ liệu từ lưu trữ đĩa vào bộ nhớ chính và quyết định dữ liệu nào trữ trong bộ nhớ.

Hơn nữa, một số cấu trúc dữ liệu được cần đến như bộ phận của sự thực thi hệ thống vật lý:

- ***Các file dữ liệu:*** Lưu trữ CSDL
- ***Tự điển dữ liệu (Data Dictionary):*** lưu metadata về cấu trúc CSDL.
- ***Chỉ mục (Indices):*** cung cấp truy xuất nhanh đến các hạng mục dữ liệu chứa các giá trị tìm kiếm.
- ***Dữ liệu thống kê (Statistical data):*** lưu trữ thông tin thống kê về dữ liệu trong cơ sở dữ liệu. Thông tin này được dùng bởi bộ xử lý vấn tin để chọn những phương pháp hiệu quả thực hiện câu vấn tin.

Sơ đồ các thành phần và các nối kết giữa chúng



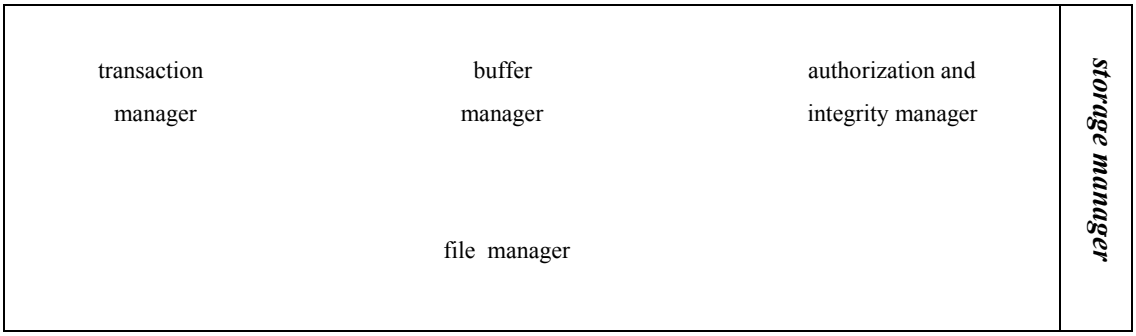


Figure 2

KIẾN TRÚC HỆ CƠ SỞ DỮ LIỆU

Kiến trúc hệ CSDL bị ảnh hưởng nhiều bởi hệ thống máy nền. Các sắc thái của kiến trúc máy như mạng, song song và phân tán được phản ánh trong kiến trúc của hệ CSDL.

- Mạng máy tính cho phép thực hiện một số công việc trên một hệ thống các server, một số công việc trên các hệ thống client. Việc phân chia công việc này dẫn đến sự phát triển hệ CSDL *client-server.*
- Xử lý song song trong một hệ thống máy tính làm tăng tốc độ các hoạt động của hệ CSDL, trả lời các giao dịch nhanh hơn. Các vấn tin được xử lý theo cách khai thác tính song song. Sự cần thiết xử lý vấn tin song song này dẫn tới sự phát triển của hệ CSDL *song song*.
- Dữ liệu phân tán trên các site hoặc trên các bộ phận trong một cơ quan cho phép các dữ liệu thường trú tại nơi chúng được sinh ra nhưng vẫn có thể truy xuất chúng từ các site khác hay các bộ phận khác. Việc lưu nhiều bản sao của CSDL trên các site khác nhau cho phép các tổ chức lớn vẫn có thể tiếp tục hoạt động khi một hay một vài site bị sự cố. Hệ CSDL phân tán được phát triển để quản lý dữ liệu phân tán, trên phương diện địa lý hay quản trị, trải rộng trên nhiều hệ CSDL .

HỆ THỐNG TẬP TRUNG

Các hệ CSDL tập trung chạy trên máy đơn và không trao đổi với các máy khác. Các hệ thống như vậy trải từ các hệ CSDL một người sử dụng chạy trên các máy cá nhân (PC) đến các hệ CSDL hiệu năng cao chạy trên các hệ mainframe. Một hệ máy tính mục đích chung hiện đại gồm một hoặc một vài CPU và một số bộ điều khiển thiết bị được nối với nhau thông qua một bus

chung, cho phép truy xuất đến bộ nhớ chia sẻ. CPU có bộ nhớ cache cục bộ lưu các bản sao của một số phần của bộ nhớ chính nhằm tăng tốc độ truy xuất dữ liệu. Mỗi bộ điều khiển thiết bị phụ trách một kiểu thiết bị xác định. Các CPU và các bộ điều khiển thiết bị có thể thực hiện đồng thời, canh tranh truy cập bộ nhớ. Bộ nhớ cache giúp làm giảm sự tranh chấp truy xuất bộ nhớ. Ta phân biệt hai cách các máy tính được sử dụng: Hệ thống một người dùng và hệ thống nhiều người dùng. Hệ CSDL được thiết kế cho hệ thống một người dùng không hỗ trợ điều khiển cạnh tranh, chức năng phục hồi hoặc là thiếu hoặc chỉ là một sự chép dự phòng đơn giản.

HỆ THỐNG CLIENT-SERVER

Các máy tính cá nhân (PC) ngày càng trở nên mạnh hơn, nhanh hơn, và rẻ hơn. Có sự chuyển dịch trong hệ thống tập trung. Các đầu cuối (terminal) được nối với hệ thống tập trung bây giờ được thế chỗ bởi các máy tính cá nhân. Chức năng giao diện người dùng (user interface) thường được quản lý trực tiếp bởi các hệ thống tập trung nay được quản lý bởi các máy tính cá nhân. Như vậy, các hệ thống tập trung ngày nay hoạt động như các hệ thống server nó làm thoả mãn các đòi hỏi của các client. Chức năng CSDL có thể được chia thành hai phần: phần trước (front-end) và phần sau (back-end). Phần sau quản trị truy xuất cấu trúc, định giá câu vấn tin và tối ưu hoá, điều khiển sự xảy ra đồng thời và phục hồi. Phần trước của hệ CSDL gồm các công cụ như: tạo mẫu (form), các bộ soạn báo cáo (report writer), giao diện đồ hoạ người dùng (graphical user interface). Giao diện giữa phần trước và phần sau thông qua SQL hoặc một chương trình ứng dụng. Các hệ thống server có thể được phân thành các phạm trù : server giao dịch (transaction server), server dữ liệu (data server).

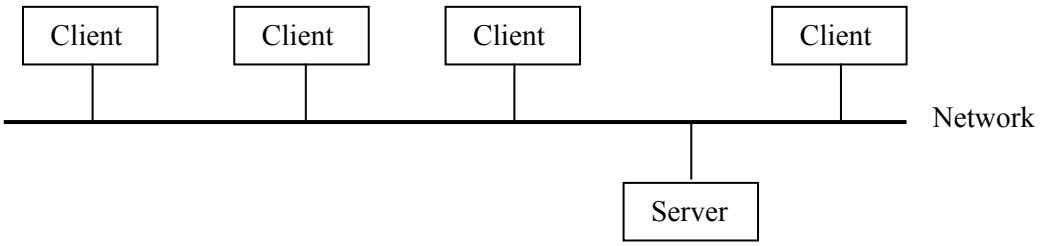


Figure 3

- ***Hệ thống server giao dịch (transaction-server systems)***: còn được gọi là hệ thống server vấn tin (query-server system), cung cấp một giao diện mà các client có thể gửi đến nó các yêu cầu thực hiện một hành động. Để đáp ứng các yêu cầu, hệ thống thực hiện các hành động và gửi lại client các kết quả. Các người sử dụng có thể đặc tả các yêu cầu trong SQL hoặc trong một giao diện trình ứng dụng sử dụng một cơ chế gọi thủ tục xa (remote-procedure-call).
    - ***Các servers giao dịch (Transaction servers):*** Trong các hệ thống tập trung, phần trước (front-end) và phần sau (back-end) được thực hiện trong một hệ thống. Kiến trúc server giao dịch cho phép chia chức năng giữa phần trước và phần sau. Chức năng phần trước được hỗ trợ trên các máy tính cá nhân (PC). Các PC hành động như những khách hàng của các hệ thống server nơi lưu trữ một khối lượng lớn dữ liệu và hỗ trợ các chức năng phần sau. Các clients gửi các giao dịch đến các hệ thống server tại đó các giao dịch được thực hiện và

các kết quả được gửi trả lại cho các clients, người giữ trách nhiệm hiển thị dữ liệu.

ODBC (Open DataBase Connectivity) được phát triển để tạo giao diện giữa các clients và các servers. ODBC là một giao diện trình ứng dụng cho phép các clients sinh ra các lệnh SQL và gửi đến một server tại đó lệnh được thực hiện. Bất kỳ client nào sử dụng giao diện có thể nối với bất kỳ một server nào cung cấp giao diện này.

Các giao diện client-server khác ODBC cũng được sử dụng trong một số hệ thống xử lý giao dịch. Chúng được xác định bởi một giao diện lập trình ứng dụng, sử dụng nó các clients tạo ra các lời gọi thủ tục giao dịch từ xa (transactional remote procedure calls) trên server. Các lời gọi này giống như các lời gọi thủ tục gốc đối với người lập trình nhưng tất cả các lời gọi thủ tục từ xa của một client được bao trong một giao dịch ở server cuối. Như vậy nếu giao dịch bỏ dở, server có thể huỷ bỏ hiệu quả của các lời gọi thủ tục xa riêng lẻ.

- ***Hệ thống server dữ liệu (Data-server systems)***: cho phép các clients trao đổi với các server bằng cách tạo ra các yêu cầu đọc hoặc cập nhật dữ liệu trong các đơn vị như file hoặc trang. Ví dụ, các file-servers cung cấp một giao diện với hệ thống file tại đó các clients có thể tạo, cập nhật, đọc hoặc xoá files. Các servers dữ liệu của cơ sở dữ liệu cung cấp nhiều chức năng hơn; chúng hỗ trợ các đơn vị dữ liệu nhỏ hơn file như trang, bộ (tuple) hoặc đối tượng. Chúng cũng cung cấp phương tiện dễ dàng để lấy chỉ mục (indexing) dữ liệu, phương tiện dễ dàng để tạo giao dịch.
    - ***Các server dữ liệu (Data Servers):*** Các hệ thống server dữ liệu được sử dụng trong các mạng cục bộ, trong đó có một nối kết tốc độ cao giữa các máy clients và máy server, các máy clients có sức mạnh xử lý tương thích với máy server và các công việc phải được thực hiện là tăng cường tính toán. Trong một môi trường như vậy, có thể gửi dữ liệu đến các máy client để thực hiện tất cả các xử lý tại máy clients sau đó gửi dữ liệu trở lại đến máy server. Kiến trúc này đòi hỏi các tính năng back-end đầy đủ tại các clients. Kiến trúc server dữ liệu thường được gặp trong các hệ CSDL hướng đối tượng (Object-Oriented DataBase Systems)

      Gửi trang đối lại với gửi hạng mục (Page shipping versus item shipping): Đơn vị liên lạc dữ liệu có thể là các "hạt thô" (Coarse granularity) như một trang, hay hạt min (fine granularity) như một bộ (tuple)/ đối tượng (object). Ta dùng thuật ngữ hạng mục để chỉ bộ hay đối tượng. Nếu đơn vị liên lạc là một hạng mục sẽ dẫn đến tổng chi phí truyền thông điệp tăng. Đem về hạng mục (fetching item) trước khi nó được yêu cầu, được gọi là đem về trước (Prefetching). Gửi trang có thể được xem như một dạng của đem về trước nếu một trang chứa nhiều hạng mục.

      Chốt (Locking): Các chốt thường được cấp bởi server trên các hạng mục mà nó gửi cho các máy clients. *Khi client giữ một chốt trên một hạng mục dữ liệu, nó có quyền "sử dụng" hạng mục dữ liệu này, hơn nữa trong khoảng thời gian client giữ chốt trên hạng mục dữ liệu không một client nào khác có thể sử dụng hạng mục dữ liệu này*. Bất lợi của gửi trang là các máy client có thể được cấp các chốt "hạt quá thô" -- một chốt trên một trang ẩn chứa các chốt trên tất cả các hạng mục trong trang. Các kỹ thuật nhằm tiết giảm chốt (lock deescalation) được đề nghị, trong đó server có thể yêu cầu các clients truyền trả lại các chốt

trên các hạng mục cấp phát trước. Nếu máy client không cần hạng mục cấp phát trước, nó có thể truyền trả lại các chốt trên hạng mục cho server và các chốt này có thể được cấp phát cho các clients khác.

Trữ dữ liệu (Data caching): Dữ liệu được gửi đến một client với danh nghĩa một giao dịch có thể được trữ ở client, ngay cả khi giao dịch đã hoàn tất, nếu không gian lưu trữ có sẵn. Các giao dịch liên tiếp tại cùng một client có thể dùng dữ liệu được trữ. Tuy nhiên, sự kết dính dữ liệu là một vấn đề cần phải được xem xét: một giao dịch tìm thấy dữ liệu được trữ, nó phải chắc chắn rằng dữ liệu này là "mới nhất" vì các dữ liệu này có thể được cập nhật bởi một client khác sau khi chúng được trữ. Như vậy, vẫn phải trao đổi với server để kiểm tra tính hợp lệ của dữ liệu và để giành được một chốt trên dữ liệu.

Trữ chốt (Lock caching): Các chốt cũng có thể được trữ lại tại máy client. Nếu một hạng mục dữ liệu được tìm thấy trong cache và chốt yêu cầu cho một truy xuất đến hạng mục dữ liệu này cũng tìm thấy trong cache, thì việc truy xuất có thể tiến hành không cần một liên lạc nào với server. Tuy nhiên, server cũng phải lưu lại vết của các chốt được trữ. Nếu một client đòi hỏi một chốt từ server, server phải gọi lại tất cả các chốt xung đột trên cùng hạng mục dữ liệu từ tất cả các máy clients đã trữ các chốt.

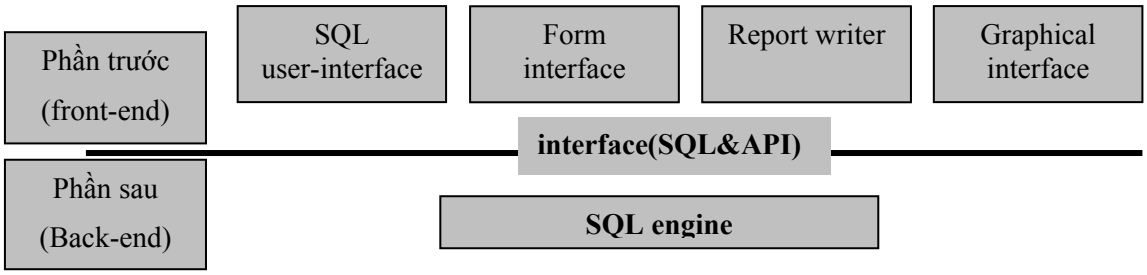


Figure 4

CÁC HỆ SONG SONG (Parallel Systems):

Các hệ song song cải tiến tốc độ xử lý và tốc độ I/O bằng cách sử dụng nhiều CPU và nhiều đĩa song song. Trong xử lý song song, nhiều hoạt động được thực hiện đồng thời. Một máy song song "hạt thô" (coarse-grain) gồm một số nhỏ các bộ xử lý mạnh. Một máy song song đồ sộ (massively parallel) hay "hạt mịn" (fine-grain) sử dụng hàng ngàn bộ xử lý nhỏ hơn. Có hai biện pháp chính để đánh giá hiệu năng của một hệ CSDL. Thứ nhất là năng lực truyền qua (throughput): *số công việc có thể được hoàn tất trong một khoảng thời gian đã cho*. Thứ hai là thời gian đáp ứng (response time): *lượng thời gian cần thiết để hoàn thành một công việc từ lúc nó được đệ trình*. Một hệ thống xử lý một lượng lớn các giao dịch nhỏ có thể cải tiến năng lực truyền qua bởi xử lý song song nhiều giao dịch. Một hệ thống xử lý các giao dịch lớn có thể cải tiến thời gian đáp ứng cũng như năng lực truyền qua bởi thực hiện song song các công việc con (subtask) của mỗi giao dịch.

- **Tăng tốc độ và tăng quy mô (Speedup & Scaleup)**: Tăng tốc độ ám chỉ việc chạy một công việc đã cho trong thời gian ngắn hơn bằng cách tăng bậc song song. Tăng quy mô ám chỉ việc quản lý các công việc lớn bằng cách tăng bậc song song. Chúng ta hãy xét một ứng dụng CSDL chạy trên một hệ thống song song với một số processor và một số đĩa. Giả sử, chúng ta tăng kích cỡ của hệ thống bằng cách tăng số processor, đĩa, và các thành phần khác của hệ thống. Mục đích là xử lý công việc trong thời gian tỷ lệ nghịch với số processor và đĩa được cấp phát.

Giả sử, thời gian thực hiện một công việc trên một máy tính lớn là $\mathbf{T_L}$, thời gian thực hiện cùng công việc này trên máy tính nhỏ là $\mathbf{T_S}$. Tăng tốc độ nhờ song song được định nghĩa là tỷ số $\mathbf{T_S/T_L}$, hệ thống song song được gọi là tăng tốc độ tuyến tính nếu tốc độ tăng là N khi hệ thống lớn có N lần tài nguyên (CPU, đĩa ...) lớn hơn hệ thống nhỏ. Nếu tốc độ tăng nhỏ hơn N, hệ thống được gọi là tăng tốc độ hạ tuyến tính (sublinear).

Tăng quy mô liên quan đến khả năng xử lý các công việc lớn trong trong cùng một lượng thời gian bằng cách cung cấp thêm tài nguyên. Giả sử, Q là một công việc, Q_N là một công việc N lần lớn hơn Q. Giả sử thời gian thực hiện công việc Q trên một máy M_S là T_S và thời gian thực hiện công việc Q_N trên một máy song song M_L, N lần lớn hơn M_S, là T_L. Tăng quy mô được định nghĩa là $\mathbf{T_S/T_L}$. Hệ song song M_L được gọi là tăng quy mô tuyến tính trên công việc Q nếu $\mathbf{T_S = T_L}$. Nếu $T_L > T_S$, hệ thống được gọi là tăng quy mô hạ tuyến tính. Tăng quy mô là một độ đo (metric) quan trọng hơn trong đo lường hiệu quả của các hệ CSDL song song. Đích của song song trong các hệ CSDL là đảm bảo hệ CSDL có thể tiếp tục thực hiện ở một tốc độ chấp nhận được, ngay cả khi kích cỡ của CSDL và số giao dịch tăng lên. Tăng khả năng của hệ thống bằng cách tăng sự song song cung cấp một con đường thuận tiện hơn cho sự phát triển hơn là thay thế một hệ tập trung bởi một máy nhanh hơn. Một số nhân tố ảnh hưởng xấu đến tính hiệu quả của hoạt động song song và có thể làm giảm cả tăng tốc độ và tăng quy mô là:

- *Chi phí khởi động (Startup Costs):* Có một chi phí khởi động kết hợp với sự khởi động một xử lý. Trong một hoạt động song song gồm hàng ngàn xử lý, thời gian khởi động (Startup time) có thể làm lu mờ thời gian xử lý hiện tại, ảnh hưởng bất lợi tới tăng tốc độ.
- *Sự giao thoa (Interference):* Các xử lý thực hiện trong một hệ song song thường truy nhập đến các tài nguyên chia sẻ, một sự giao thoa của mỗi xử lý mới khi nó cạnh tranh với các xử lý đang tồn tại trên các tài nguyên bị chiếm như bus hệ thống, đĩa chia sẻ, thậm chí cả đồng hồ. Hiện tượng này ảnh hưởng đến cả tăng tốc độ lẫn tăng quy mô.
- *Sự lệch (Skew):* Bằng cách chia một công việc thành các bước song song, ta làm giảm kích cỡ của bước trung bình. Tuy nhiên, thời gian phục vụ cho bước chậm nhất sẽ xác định thời gian phục vụ cho toàn bộ công việc. Thường khó có thể chia một công việc thành các phần cùng kích cỡ, như vậy cách mà kích cỡ được phân phối là bị lệch. Ví dụ: một công việc có kích cỡ 100 được chia thành 10 phần và sự phân chia này bị lệch, có thể có một số phần có kích cỡ nhỏ hơn 10 và một số nhiệm vụ có kích cỡ lớn hơn 10, giả sử trong đó có phần kích cỡ 20, độ tăng tốc nhận được bởi chạy các công việc song song chỉ là 5, thay vì 10.

- **Các mạng hợp nhất (Interconnection Network):** Các hệ thống song song gồm một tập hợp các thành phần (Processors, memory và các đĩa) có thể liên lạc với nhau thông qua một mạng hợp nhất. Các ví dụ về các mạng hợp nhất là:
  - *Bus:* Toàn bộ các thành phần hệ thống có thể gửi và nhận dữ liệu qua một bus liên lạc. Các kiến trúc bus làm việc tốt với một số nhỏ các processor. Tuy nhiên, chúng không có cùng quy mô khi tăng sự song song vì bus chỉ điều khiển liên lạc từ chỉ một thành phần tại một thời điểm.

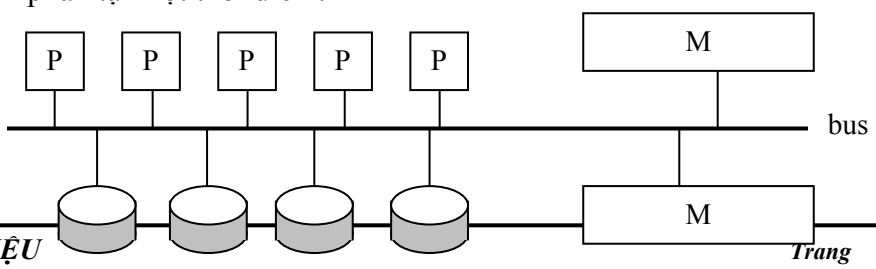
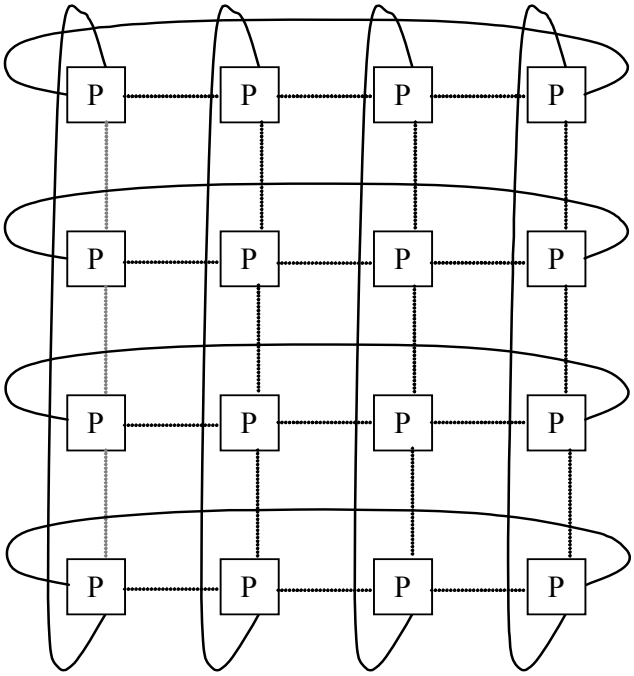


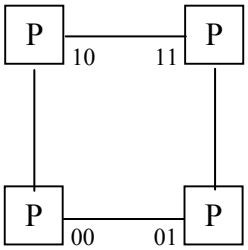
Figure 5

- *Lưới (Mesh):* Các thành phần được sắp xếp như các nút của một lưới, mỗi thành phần được nối với tất cả các thành phần kề với nó trong lưới. Trong một lưới hai chiều mỗi nút được nối với 4 nút kề.

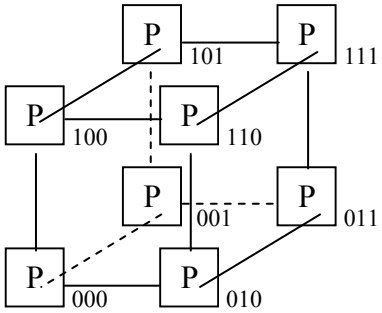


mạng hợp nhất hình lưới

Siêu lập phương một chiều



Siêu lập phương hai chiều



Siêu lập phương ba chiều

Figure 6

- *Siêu lập phương (Hypercube)*: Các thành phần được đánh số theo nhị phân, một thành phần được nối với thành phần khác nếu biểu diễn nhị nhân số của chúng sai khác nhau đúng một bit. Như vậy, mỗi một thành phần trong n thành phần được nối với log(n) thành phần khác. Có thể kiểm nghiệm được rằng trong một sự hợp nhất siêu lập phương một thông điệp từ một thành phần đến một thành phần khác đi quá nhiều nhất log(n) nối kết, còn trong lưới là $\sqrt{n}-1$

- **Các kiến trúc cơ sở dữ liệu song song:** Có một vài mô hình kiến trúc cho các máy song song:

  (Ký hiệu: P = processor, M = Memory, = đĩa)

  - *Bộ nhớ chia sẻ (Shared memory)*: Tất cả các các processor chia sẻ một bộ nhớ chung. Trong mô hình này các processor và các đĩa truy xuất một bộ nhớ chung, thường thông qua một bus hoặc một mạng hợp nhất. Thuận lợi của chia sẻ bộ nhớ là liên lạc giữa các processor là cực kỳ hiệu quả: dữ liệu trong bộ nhớ chia sẻ có thể được truy xuất bởi bất kỳ processor nào mà không phải di chuyển bởi phần mềm. Một processor có thể gửi thông điệp cho một processor khác bằng cách viết vào bộ nhớ chia sẻ, cách liên lạc này nhanh hơn nhiều so với các liên lạc khác. Tuy nhiên, các máy bộ nhớ chia sẻ không thể hỗ trợ nhiều hơn 64 processor vì nếu nhiều hơn, bus hoặc mạng hợp nhất sẽ trở nên dễ bị nghẽn (bottle-neck). Kiến trúc bộ nhớ chia sẻ thường có những cache lớn cho mỗi processor, như vậy việc tham khảo bộ nhớ chia sẻ có thể tránh được mỗi khi có thể.

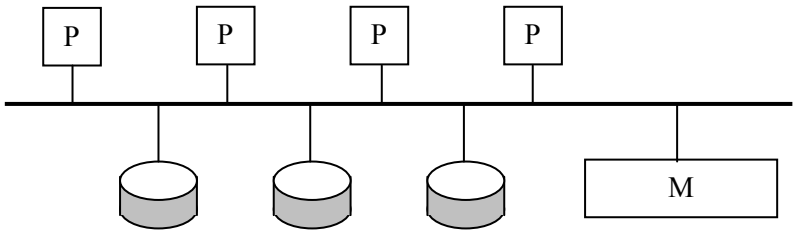


Figure 7

- *Đĩa chia sẻ (Shared disk):* Tất cả các processor chia sẻ đĩa chung. Mô hình này còn được gọi là cụm (cluster). Trong mô hình này tất cả các processor có thể truy xuất trực tiếp đến tất cả các đĩa thông qua một mạng hợp nhất, nhưng mỗi processor có bộ nhớ riêng.

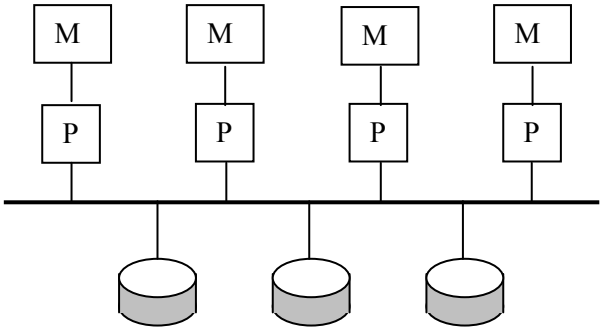


Figure 8

Kiến trúc này có các điểm thuận lợi là: thứ nhất, bus bộ nhớ không bị bottle-neck, thứ hai, cho một phương pháp rẻ để cung cấp một mức độ lượng thứ lỗi--một processor bị hỏng hóc, các processor khác có thể tiếp tục công việc của nó. Ta có thể tạo ra hệ thống con các đĩa tự lượng thứ lỗi bằng cách sử dụng kiến trúc RAID (được trình bày sau này). Vấn đề chính của chia sẻ đĩa là sự hợp nhất các hệ thống con các đĩa trở nên bottle-neck, đặc biệt trong tình huống CSDL truy xuất đĩa nhiều. So sánh với bộ nhớ chia sẻ, chia sẻ đĩa có thể hỗ trợ một số lượng processor lớn hơn, nhưng việc liên lạc giữa các processor chậm hơn.

- *Không chia sẻ (Shared nothing):* Các processor không chia sẻ bộ nhớ chung, cũng không chia sẻ đĩa chung. Trong hệ thống này mỗi nút của máy có một processor, bộ nhớ và một vài đĩa.

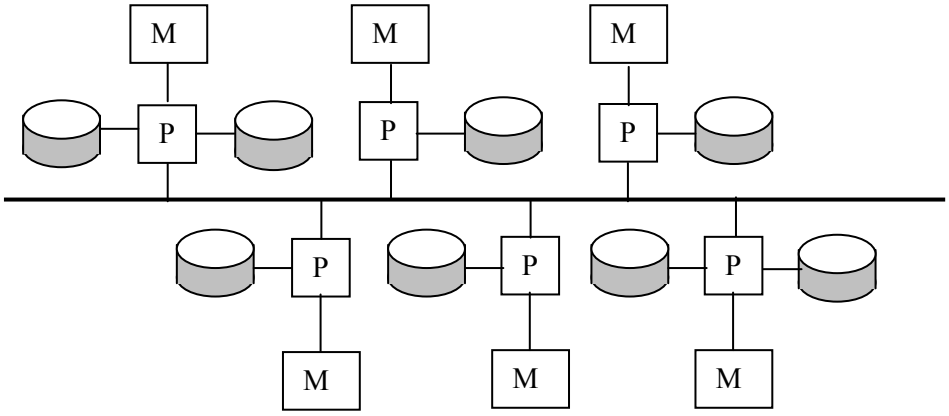


Figure 9

Các processor ở mỗi nút có thể liên lạc với các processor khác qua mạng hợp nhất tốc độ cao. Chức năng của một nút, như server, dữ liệu được chứa trên các đĩa của nó. Mô hình không chia sẻ gì chỉ có vấn đề về việc truy xuất các đĩa không cục bộ và việc truyền các quan hệ kết quả qua mạng. Hơn nữa, đối với các hệ thống không chia sẻ gì, các mạng hợp nhất thường được thiết kế để có thể tăng quy mô, sao cho khả năng truyền của chúng tăng khi các nút mới được thêm vào.

- *Phân cấp (hierarchical):* Mô hình này là một sự lai kiểu của các kiến trúc trước.

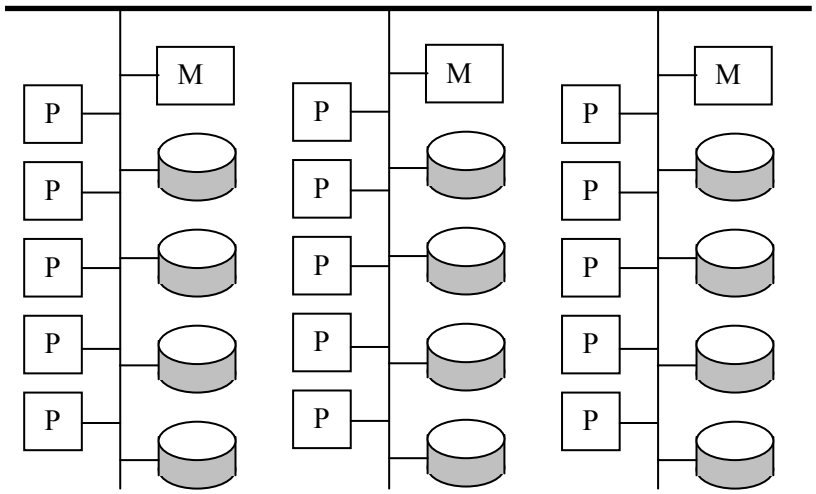


Figure 10

Kiến trúc này tổ hợp các đặc trưng của các kiến trúc chia sẻ bộ nhớ, chia sẻ đĩa và không chia sẻ gì. Ở mức cao nhất, hệ thống bao gồm những nút được nối bởi mạng hợp nhất và không chia sẻ đĩa cũng như bộ nhớ với nút khác. Như vậy, mức cao nhất là kiến trúc không chia sẻ gì. Mỗi nút của hệ thống có thể là hệ thống chia sẻ bộ nhớ với một vài processor. Kế tiếp, mỗi nút có thể là một hệ thống chia sẻ đĩa. Mỗi một hệ thống chia sẻ đĩa lại có thể là một hệ thống chia sẻ bộ nhớ... Như vậy, hệ thống có thể được xây dựng như một sự phân cấp.

CÁC HỆ THỐNG PHÂN TÁN (Distributed Systems):

Trong một hệ thống CSDL phân tán, CSDL được lưu trữ trên một vài máy tính. Các máy tính trong một hệ thống phân tán liên lạc với một máy khác qua nhiều dạng phương tiện liên lạc khác nhau: mạng tốc độ cao, đường điện thoại... Chúng không chia sẻ bộ nhớ cũng như đĩa. Các máy tính trong hệ thống phân tán có thể rất đa dạng về kích cỡ cũng như chức năng: từ các workstation đến các mainframe. Các máy tính trong hệ thống phân tán được tham chiếu bởi một số các tên khác nhau: site , node -- phụ thuộc vào ngữ cảnh mà máy được đề cập. Ta sẽ sử dụng thuật ngữ **site** để nhấn mạnh sự phân tán vật lý của các hệ thống này.

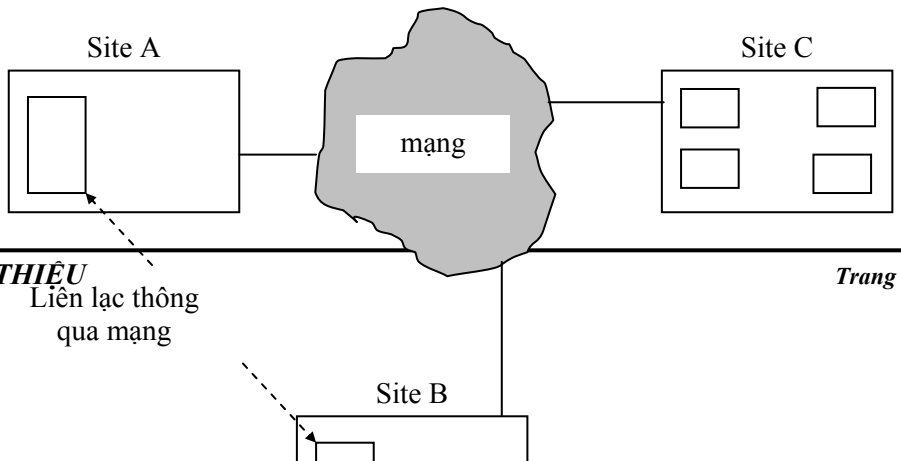


Figure 11

Sự sai khác chính giữa CSDL song song không chia sẻ gì và hệ thống phân tán là CSDL phân tán được tách biệt về mặt địa lý, được quản trị tách biệt và có một sự hợp nhất chậm. Hơn nữa, trong hệ thống phân tán người ta phân biệt giữa các giao dịch cục bộ (local) và toàn thể (global). Giao dịch cục bộ là một giao dịch truy xuất dữ liệu trong một **site** tại đó giao dịch đã được khởi xướng. Giao dịch toàn thể là một giao dịch mà nó hoặc truy xuất dữ liệu trong một site từ một site khác tại đó nó được khởi xướng hoặc truy xuất dữ liệu trong một vài site khác nhau.

BÀI TẬP CHƯƠNG I

I.1 Bốn điểm khác nhau chính giữa một hệ thống xử lý file và một hệ quản trị CSDL là gì ?

I.2 Giải thích sự khác nhau giữa độc lập dữ liệu vật lý và độc lập dữ liệu logic

I.3 Liệt kê năm nhiệm vụ của nhà quản trị CSDL. Đối với mỗi nhiệm vụ, giải tích rõ những vấn đề nảy sinh nếu nhiệm vụ đó không được hoàn thành.

I.4 Năm chức năng chính của nhà quản trị CSDL là gì ?

I.5 Sử dụng một mảng hai chiều kích cỡ n x m như một ví dụ để minh hoạ sự khác nhau:

1. giữa ba mức trừu tượng dữ liệu
2. giữa sơ đồ và thể hiện

I.6 Trong các hệ thống client-server tiêu biểu, máy server thường mạnh hơn máy client rất nhiều: Có bộ xử lý nhanh hơn thậm chí có thể có nhiều bộ xử lý, có bộ nhớ lớn hơn, có dung lượng đĩa lớn hơn. Ta xét một kịch bản trong đó các máy client và các máy server là mạnh như

nhau. Xây dựng một hệ thống client-server theo kịch bản như vậy có ưu nhược điểm gì ? Kịch bản nào phù hợp hơn với kiến trúc server dữ liệu ?

I.7 Giả sử một giao dịch được viết trong C với SQL nhúng, và khoảng 80% thời gian được dùng cho code SQL, 20% còn lại cho code C. Nếu song song được dùng chỉ cho code SQL, Tăng tốc độ có thể đạt tới bao nhiêu ? Giải thích.

I.8 Những nhân tố nào chống lại việc tăng quy mô tuyến tính trong một hệ thống xử lý giao dịch ? Nhân tố nào là quan trọng nhất trong mỗi một kiến trúc sau: bộ nhớ chia sẻ, đĩa chia sẻ, không chia sẻ gì ?

I.9 Xét một mạng dựa trên đường điện thoại quay số tự động, trong đó các site liên lạc theo định kỳ, ví dụ hàng đêm. Các mạng như vậy thường được cấu hình với một site server và nhiều site client. Các site client chỉ nói với server và trao đổi dữ liệu với client khác bởi lưu dữ liệu tại server và lấy dữ liệu được lưu trên server bởi client khác. Ưu, nhược điểm của kiến trúc như vậy là gì ?

CHƯƠNG II

SQL

MỤC ĐÍCH

Giới thiệu một hệ CSDL chuẩn, SQL, các thành phần cơ bản của của nó.

YÊU CẦU

Hiểu các thành phần cơ bản của SQL-92

Hiểu và vận dụng phương pháp "dịch" từ câu vấn tin trong ngôn ngữ tự nhiên sang ngôn ngữ SQL và ngược lại

Hiểu và vận dụng cách thêm (xen), xóa dữ liệu

SQL là ngôn ngữ CSDL quan hệ chuẩn, gốc của nó được gọi là Sequel. SQL là viết tắt của **S**tructured **Q**uery **L**anguage. Có nhiều phiên bản của SQL. Phiên bản được trình bày trong giáo trình này là phiên bản chuẩn SQL-92.

SQL có các phần sau:

- ***Ngôn ngữ định nghĩa dữ liệu (DDL)***. **DDL** của SQL cung cấp các lệnh để định nghĩa các sơ đồ quan hệ, xoá các quan hệ, tạo các chỉ mục, sủa đổi các sơ đồ quan hệ
- ***Ngôn ngữ thao tác dữ liệu tương tác (Interactive DML)*****. IDML** bao gồm một ngôn ngữ dựa trên cả đại số quan hệ lẫn phép tính quan hệ bộ. Nó bao hàm các lệnh xen các bộ, xoá các bộ, sửa đổi các bộ trong CSDL
- ***Ngôn ngữ thao tác dữ liệu nhúng (Embedded DML)*****.** Dạng SQL nhúng được thiết kế cho việc sử dụng bên trong các ngôn ngữ lập trình mục đích chung (genaral-purpose programming languages) như PL/I, Cobol, Pascal, Fortran, C.
- ***Đinh nghĩa view***. **DDL SQL** cũng bao hàm các lệnh để định nghĩa các view.
- ***Cấp quyền (Authorization)*****. DDL SQL** bao hàm cả các lệnh để xác định các quyền truy xuất dến các quan hệ và các view
- ***Tính toàn vẹn (Integrity)*****. DDL SQL** chứa các lệnh để xác định các ràng buộc toàn vẹn mà dữ liệu được lưu trữ trong CSDL phải thoả.
- ***Điều khiển giao dịch*****.** SQL chứa các lệnh để xác định bắt đầu và kết thúc giao dịch, cũng cho phép chốt tường minh dữ liệu để điều khiển cạnh tranh

Các ví dụ minh hoạ cho các câu lệnh SQL được thực hiện trên các sơ đồ quan hệ sau:

- **Branch_schema = (Branch_name, Branch_city, Assets)**: Sơ đồ quan hệ chi nhánh nhà băng gồm các thuộc tính Tên chi nhánh (Branch_name), Thành phố (Branch_city), tài sản (Assets)
- **Customer_schema = (Customer_name, Customer_street, Customer_city)**: Sơ đồ quan hệ Khách hàng gồm các thuộc tính Tên khách hàng (Customer_name), phố (Customer_street), thành phố (Customer_city)
- **Loan_schema = (Branch_name, loan_number, amount)**: Sơ đồ quan hệ cho vay gồm các thuộc tính Tên chi nhánh, số cho vay (Loan_number), số lượng (Amount)
- **Borrower_schema = (Customer_name, loan_number)**: Sơ đồ quan hệ người mượn gồm các thuộc tính Tên khách hàng, số cho vay
- **Account_schema = (Branch_name, account_number, balance)**: Sơ đồ quan hệ tài khoản gồm các thuộc tính Tên chi nhánh, số tài khoản (Account_number), số cân đối (Balance: dư nợ/có)
- **Depositor_schema = (Customer_name, account_number)**: Sơ đồ người gửi gồm các thuộc tính Tên khách hàng, số tài khoản

Cấu trúc cơ sở của một biểu thức SQL gồm ba mệnh đề: **SELECT, FROM và WHERE**

- ♦ Mệnh đề **SELECT** tương ứng với phép chiếu trong đại số quan hệ, nó được sử dụng để liệt kê các thuộc tính mong muốn trong kết quả của một câu vấn tin
- ♦ Mệnh đề **FROM** tương ứng với phép tích Đề các , nó nó liệt kê các quan hệ được quét qua trong sự định trị biểu thức
- ♦ Mệnh đề **WHERE** tương ứng với vị từ chọn lọc, nó gồm một vị từ chứa các thuộc tính của các quan hệ xuất hiện sau FROM

Một câu vấn tin kiểu mẫu có dạng:

SELECT A_1, A_2, ..., A_k
FROM R_1, R_2, ..., R_m
WHERE P

trong đó A_i là các thuộc tính (Attribute), R_j là các quan hệ (Relation) và P là một vị từ (Predicate). Nếu thiếu WHERE vị từ P là TRUE.

Kết quả của một câu vấn tin SQL là một quan hệ.

MỆNH ĐỀ SELECT

Ta tìm hiểu mệnh đề SELECT bằng cách xét một vài ví dụ:

"Tìm kiếm tất cả các tên các chi nhánh trong quan hệ cho vay (loan)":

SELECT Branch_name
FROM Loan;

Kết quả là một quan hệ gồm một thuộc tính Tên chi nhánh (Branch_name)

Nếu muốn quan hệ kết quả không chứa các tên chi nhánh trùng nhau:

SELECT **DISTINCT** Branch_name
FROME Loan;

Từ khoá ALL được sử dụng để xác định tường minh rằng các giá trị trùng không bị xoá và nó là mặc nhiên của mệnh đề SELECT.

Ký tự * được dùng để chỉ tất cả các thuộc tính:

SELECT *
FROM Loan;

Sau mệnh đề SELECT cho phép các biểu thức số học gồm các phép toán +, -, *, / trên các hằng hoặc các thuộc tính:

```
SELECT Branch_name, Loan_number, amount * 100
FROM Loan;
```

MỆNH ĐỀ WHERE

"Tìm tất cả các số cho vay ở chi nhánh tên Perryridge với số lượng vay lớn hơn1200$"

```
SELECT Loan_number
FROM Loan
WHERE Branch_name = 'Perryridge' AND Amount > 1200;
```

SQL sử dụng các phép nối logic: **NOT, AND, OR**. Các toán hạng của các phép nối logic có thể là các biểu thức chứa các toán tử so sánh =, >=, <>, <, <=.

Toán tử so sánh **BETWEEN** được dùng để chỉ các giá trị nằm trong một khoảng:

```
SELECT Loan_number
FROM Loan
WHERE Amount  BETWEEN 50000 AND 100000;
```

≈

```
SELECT  Loan_number
FROM Loan
WHERE Amount >= 50000 AND Amount <= 100000;
```

Ta cũng có thể sử dụng toán tử **NOT BETWEEN**.

MỆNH ĐỀ FROM

"Trong tất cả các khách hàng có vay ngân hàng tìm tên và số cho vay của họ"

```
SELECT DISTINCT Customer_name, Borrower.Loan_number
FROM Borrower, Loan
WHERE Borrower.Loan_number = Loan.Loan_number;
```

SQL sử dụng cách viết <tên quan hệ >.< tên thuộc tính > để che dấu tính lập lờ trong trường hợp tên thuộc tính trong các sơ đồ quan hệ trùng nhau.

"Tìm các tên và số cho vay của tất cả các khách hàng có vay ở chi nhánh Perryridge"

```
SELECT Customer_name, Borrower.Loan_number
FROM Borrower, Loan
WHERE Borrower.Loan_number = Loan.Loan_number AND
        Branch_name ='Perryridge';
```

CÁC PHÉP ĐỔI TÊN

SQL cung cấp một cơ chế đổi tên cả tên quan hệ lẫn tên thuộc tính bằng mệnh đề dạng:

< tên cũ > **AS** < tên mới >

mà nó có thể xuất hiện trong cả mệnh đề SELECT lẫn FROM

```
SELECT DISTINCT Customer_name, Borrower.Loan_number
FROM Borrower, Loan
WHERE Borrower.Loan_number = Loan.Loan_number AND
        Branch_name ='Perryridge';
```

Kết quả của câu vấn tin này là một quan hệ hai thuộc tính: Customer_name, Loan_number

Đổi tên thuộc tính của quan hệ kết quả:

```
SELECT Customer_name, Borrower.Loan_number AS Loan_Id
FROM Borrower, Loan
```

```
        WHERE Borrower.Loan_number = Loan.Loan_number AND
              Branch_name =’Perryridge’;
```

CÁC BIẾN BỘ (Tuple Variables)

Các biến bộ được định nghĩa trong mệnh đề FROM thông qua sử dụng mệnh đề AS:

```
SELECT DISTINCT Customer_name, T.Loan_number
FROM Borrower  AS  T, Loan  AS  S
WHERE T.Loan_number = S.Loan_number AND
      Branch_name =’Perryridge’;
```

“Tìm các tên của tất cả các chi nhánh có tài sản lớn hơn ít nhất một chi nhánh ở Brooklyn“

```
SELECT DISTINCT T.branch_name
FROM  Branch AS T, Banch AS S
WHERE T.assets > S.assets AND S.Branch_City = ‘Brooklyn’
```

SQL92 cho phép sử dụng các viết (v_1, v_2, ..., vn) để ký hiệu một n-bộ với các giá trị v_1, v_2, ..., vn. Các toán tử so sánh có thể được sử dụng trên các n-bộ và theo thứ tự tự điển. Ví dụ (a1, b1) <= (a2, b2) là đúng nếu ($a_1 < b_1$) OR (($a_1 = b_1$) AND ($a_2 < b_2$)).

CÁC PHÉP TOÁN TRÊN CHUỖI

Các phép toán thường được dùng nhất trên các chuỗi là phép đối chiếu mẫu sử dụng toán tử **LIKE**. Ta mô tả các mẫu dùng hai ký tự đặc biệt:

- ký tự phần trăm (%): ký tự % tương xứng với ***chuỗi con*** bất kỳ
- ký tự gạch nối (_): ký tự gạch nối tương xứng với ***ký tự*** bất kỳ.
  - ‘Perry%’ tương xứng với bất kỳ chuỗi nào bắt đầu bởi ‘Perry’
  - ‘%idge%’ tương xứng với bất kỳ chuỗi nào chứa ‘idge’ như chuỗi con
  - ‘___’ tương xứng với chuỗi bất kỳ có đúng ba ký tự
  - ‘___%’ tương xứng với chuỗi bất kỳ có ít nhất ba ký tự

"Tìm tên của tất cả các khách hàng tên phố của họ chứa chuỗi con ‘Main’

```
SELECT Customer_name
FROM Customer
WHERE Customer_street LIKE ‘%Main%’
```

Nếu trong chuỗi mẫu có chứa các ký tự % _ \ , để tránh nhầm lẫn ký tự với"dấu hiệu thay thế", SQL sử dụng cách viết: ký tự escape (\) đứng ngay trước ký tự"đặc biệt". Ví dụ nếu chuỗi mẫu là ab%cd được viết là ‘ab\%cd’, chuỗi mẫu là ab_cde được viết là ‘ab_cde’, chuỗi mẫu là ab\cd được viết là ‘ab\\cd’

SQL cho phép đối chiếu không tương xứng bằng cách sử dụng **NOT LIKE**

SQL cũng cho phép các hàm trên chuỗi: nối hai chuỗi (|), trích ra một chuỗi con, tìm độ dài chuỗi, biến đổi một chuỗi chữ thường sang chuỗi chữ hoa và ngược lại ...

THỨ TỰ TRÌNH BÀY CÁC BỘ (dòng)

Mệnh đề ORDER BY tạo ra sự trình bày các dòng kết quả của một câu vấn tin theo một trình tự. Để liết kê theo thứ tự alphabet tất cả các khách hàng có vay ở chi nhánh Perryridge:

```
SELECT  DISTINCT  Customer_name
FROM Borrower, Loan
WHERE Borrower.Loan_number = Loan.Loan_number AND
            Branch_name = ‘Perryridge’
ORDER BY Customer_name;
```

Mặc nhiên, mệnh đề ORDER BY liệt kê theo thứ tự tăng, tuy nhiên ta có thể làm liệt kê theo thứ tự **giảm**/**tăng** bằng cách chỉ rõ bởi từ khoá **DESC**/ **ASC**

```
SELECT *
FROM Loan
ORDER BY Amount DESC, Loan_number ASC;
```

CÁC PHÉP TOÁN TẬP HỢP

SQL92 có các phép toán **UNION, INTERSECT, EXCEPT** chúng hoạt động giống như các phép toán hợp, giao, hiệu trong đại số quan hệ. Các quan hệ tham gia vào các phép toán này phải tương thích (có cùng tập các thuộc tính).

- Phép toán **UNION**

"tìm kiếm tất cả các khách hàng có vay, có tài khoản hoặc cả hai ở ngân hàng"

```
(SELECT  Customer_name
 FROM Depositor)
UNION
(SELECT Customer_name
 FROM Borrower);
```

Phép toán hợp UNION tự động loại bỏ các bộ trùng, nếu ta muốn giữ lại các bộ trùng ta phải sử dụng **UNION ALL**

```
(SELECT  Customer_name
 FROM Depositor)
UNION ALL
(SELECT Customer_name
 FROM Borrower);
```

- Phép toán **INTERSECT**

"tìm kiếm tất cả các khách hàng có vay và cả một tài khoản tại ngân hàng"

```
(SELECT DISTINCT Customer_name
 FROM Depositor)
INTERSECT
(SELECT DISTINCT Customer_name
 FROM Borrower);
```

Phép toán INTERESCT tự động loại bỏ các bộ trùng, Để giữ lại các bộ trùng ta sử dụng **INTERSECT ALL**

```
(SELECT  Customer_name
 FROM Depositor)
INTERSECT ALL
(SELECT  Customer_name     FROM Borrower);
```

- Phép toán **EXCEPT**

"Tìm kiếm tất cả các khách hàng có tài khoản nhưng không có vay tại ngân hàng"

```
(SELECT Customer_name
 FROM Depositor)
EXCEPT
(SELECT Customer_name
 FROM Borrower);
```

EXCEPT tự động loại bỏ các bộ trùng, nếu muốn giữ lại các bộ trùng phải dùng **EXCEPT ALL**

```
(SELECT Customer_name
 FROM Depositor)
EXCEPT ALL
```

```
(SELECT Customer_name
 FROM Borrower);
```

CÁC HÀM TÍNH GỘP

SQL có các hàm tính gộp (aggregate functions):

- Tính trung bình (Average): **AVG**()
- Tính min : **MIN**()
- Tính max: **MAX**()
- Tính tổng: **SUM**()
- Đếm: **COUNT**()

Đối số của các hàm AVG và SUM phải là **kiểu dữ liệu số**

"Tìm số cân đối tài khoản trung bình tại chi nhánh Perryridge"

```
SELECT AGV(balace)
FROM Account
WHERE Branch_name = 'Perryridge';
```

SQL sử dụng mệnh để **GROUP BY** vào mục đích nhóm các bộ có cùng giá trị trên các thuộc tính nào đó

"Tìm số cân đối tài khoản trung bình tại mỗi chi nhánh ngân hàng"

```
SELECT Branch_name, AVG(balance)
FROM Account
GROUP BY Branch_name;
```

"Tìm số các người gửi tiền đối với mỗi chi nhánh ngân hàng"

```
SELECT Branch_name, COUNT(DISTINCT Customer_name)
FROM Depositor, Account
WHERE Depositor.Account_number = Account.Acount_number
GROUP BY Branch_name
```

Giả sử ta muốn liệt kê các chi nhánh ngân hàng có số cân đối trung bình lớn hơn 1200$. Điều kiện này không áp dụng trên từng bộ, nó áp dụng trên từng nhóm. Để thực hiện được điều này ta sử dụng mệnh đề **HAVING** của SQL

```
SELECT Branch_name, AVG(balance)
FROM Account
GROUP BY Branch_name
HAVING AGV(Balance) > 1200$;
```

Vị từ trong mệnh đề HAVING được áp dụng sau khi tạo nhóm, như vậy hàm AVG có thể được sử dụng

"Tìm số cân đối đối với tất cả các tài khoản"

```
SELECT AVG(Balance)   FROM Account;
```

"Đếm số bộ trong quan hệ Customer"

```
SELECT Count(*)     FROM Customer;
```

SQL không cho phép sử dụng **DISTINCT** với **COUNT(*)**, nhưng cho phép sử dụng DISTINCT với MIN và MAX.

Nếu WHERE và HAVING có trong cùng một câu vấn tin, vị từ sau WHERE được áp dụng trước. Các bộ thoả mãn vị từ WHERE được xếp vào trong nhóm bởi GROUP BY, mệnh đề HAVING (nếu có) khi đó được áp dụng trên mỗi nhóm. Các nhóm không thoả mãn mệnh đề HAVING sẽ bị xoá bỏ.

"Tìm số cân đối trung bình đối với mỗi khách hàng sống ở Harrison và có ít nhất ba tài khoản"

```
SELECT Depositor.Customer_name, AVG(Balance)
```

```
FROM Depositor, Account, Customer
WHERE Depositor.Account_number = Account.Account_number AND
      Depositor.Customer_name = Customer.Customer_name AND
      Customer.city ='Harrison'
GROUP BY Depositor.Customer_name
HAVING COUNT(DISTINT Depositor.Account_number) >= 3;
```

CÁC GIÁ TRỊ NULL

SQL cho phép sử dụng các giá trị null để chỉ sự vắng mặt thông tin tạm thời về giá trị của một thuộc tính. Ta có thể sử dụng từ khoá đặc biệt **null** trong vị từ để thử một giá trị null.

"Tìm tìm tất cả các số vay trong quan hệ Loan với giá trị Amount là null"

```
SELECT Loan_number
FROM Loan
WHERE Amount is null
```

Vị từ **not null** thử các giá trị không rỗng

Sử dụng giá trị null trong các biểu thức số học và các biểu thức so sánh gây ra một số phiền phức. Kết quả của một biểu thức số học là null nếu một giá trị input bất kỳ là null. Kết quả của một biểu thức so sánh chứa một giá trị null có thể được xem là false. SQL92 xử lý kết quả của một phép so sánh như vậy như là một giá trị **unknown**, là một giá trị không là **true** mà cũng không là **false**. SQL92 cũng cho phép thử kết quả của một phép so sánh là unknown hay không. Tuy nhiên, trong hầu khắp các trường hợp, unknown được xử lý hoàn toàn giống như false.

Sự tồn tại của các giá trị null cũng làm phức tạp việc sử lý các toán tử tính gộp. Giả sử một vài bộ trong quan hệ Loan có các giá trị null trên trường Amount. Ta xét câu vấn tin sau:

```
SELECT SUM(Amount)
FROM LOAN
```

Các giá trị được lấy tổng trong câu vấn tin bao hàm cả các trị null. Thay vì tổng là null, SQL chuẩn thực hiện phép tính tổng bằng cách bỏ qua các giá trị input là null.

Nói chung, các hàm tính gộp tuân theo các quy tắc sau khi xử lý các giá trị null: Tất cả các hàm tính gộp ngoại trừ COUNT(*) bỏ qua các giá trị input null. Khi các giá trị nul bị bỏ qua, tập các giá trị input có thể là rõng. COUNT() của một tập rỗng được định nghĩa là 0. Tất cả các hàm tính gộp khác trả lại giá trị null khi áp dụng trên tập hợp input rỗng.

CÁC CÂU VẤN TIN CON LỒNG NHAU (Nested Subqueries)

SQL cung cấp một cơ chế lòng nhau của các câu vấn tin con. Một câu vấn tin con là một biểu thức SELECT-FROM-WHERE được lồng trong một caau vấn tin khác. Các câu vấn tin con thường được sử dụng để thử quan hệ thành viên tập hợp, so sánh tập hợp và bản số tập hợp.

QUAN HỆ THÀNH VIÊN TẬP HỢP (Set relationship)

SQL đưa vào các phép tính quan hệ các phép toán cho phép thử các bộ có thuộc một quan hệ nào đó hay không. Liên từ **IN** thử quan hệ thành viên này. Liên từ NOT IN thử quan hệ không là thành viên.

"Tìm tất cả các khách hàng có cả vay lẫn một tài khoản tại ngân hàng"

Ta đã sử dụng INTERSECTION để viết câu vấn tin này. Ta có thể viết câu vấn tin này bằng các sử dụng IN như sau:

```
SELECT DISTINCT Customer_name
```

```
FROM Borrower
WHERE Customer_name IN (        SELECT Customer_name
                                FROM Depositor)
```

Ví dụ này thử quan hệ thành viên trong một quan hệ một thuộc tính. SQL92 cho phép thử quan hệ thành viên trên một quan hệ bất kỳ.

"Tìm tất cả các khách hàng có cả vay lãn một tài khoản ở chi nhánh Perryridge"

Ta có thể viết câu truy vấn như sau:

```
SELECT DISTINCT Customer_name
FROM Borrower, Loan
WHERE       Borrower. Loan_number = Loan.Loan_number AND
            Branch_name = 'Perryridge'   AND
            (Branch_name. Customer_name IN
                  (SELECT Branch_name, Customer_name
                   FROM Depositor, Account
                   WHERE       Depositor.Account_number   =
                               Account.Account_number )
```

"Tìm tất cả các khách hàng có vay ngân hàng nhưng không có tài khoản tại ngân hàng"

```
SELECT DISTINCT Customer_name
FROM borrower
WHERE Customer_name NOT IN (  SELECT Customer_name
                              FROM Depositor)
```

Các phép toán IN và NOT IN cũng có thể được sử dụng trên các tập hợp liệt kê:

```
SELECT DISTINCT Customer_name
FROM borrower
WHERE Customer_name NOT IN ('Smith', 'Jone')
```

SO SÁNH TẬP HỢP (Set Comparision)

"Tìm tên của tất cả các chi nhánh có tài sản lớn hơn ít nhất một chi nhánh đóng tại Brooklyn"

```
SELECT DISTINCT Branch_name
FROM Branch AS T, Branch AS S
WHERE T.assets > S.assets AND S.branch_city = 'Brooklyn'
```

Ta có thể viết lại câu vấn tin này bằng cách sử dụng mệnh đề"lớn hơn ít nhất một"trong SQL

- **SOME** :

```
SELECT Branch_name
FROM Branch
WHERE Assets > SOME (    SELECT Assets
                         FROM Branch
                         WHERE Branch_city ='Brooklyn')
```

Câu vấn tin con

```
( SELECT Assets
  FROM Branch
  WHERE Branch_city ='Brooklyn')
```

sinh ra tập tất cả các Assets của tất cả các chi nhánh đóng tại Brooklyn. So sánh > SOME trong mệnh đề WHERE nhận giá trị đúng nếu giá trị Assets của bộ được xét lớn hơn ít nhất một trong các giá trị của tập hợp này.

SQL cũng có cho phép các so sánh **< SOME, >= SOME, <= SOME, = SOME, <> SOME**

- **ALL**

"Tìm tất cả các tên của các chi nhánh có tài sản lớn hơn tài sản của bất kỳ chi nhánh nào đóng tại Brooklyn"

```
SELECT Branch_name
FROM Branch
WHERE Assets > ALL (    SELECT Assets
                        FROM Branch
                        WHERE Branch_citty = 'Brooklyn')
```

SQL cũng cho phép các phép so sánh: **< ALL, <= ALL, > ALL, >= ALL, = ALL, <> ALL**.

"Tìm chi nhánh có số cân đối trung bình lớn nhất"

SQL không cho phép hợp thành các hàm tính gộp, như vậy MAX(AVG (...)) là không được phép. Do vậy, ta phải sử dụng câu vấn tin con như sau:

```
SELECT Branch_name
FROM Account
GROUP BY Branch_name
HAVING AVG (Balance) >= ALL ( SELECT AVG (balance)
                              FROM Account
                              GROUP BY Branch_name)
```

THỬ CÁC QUAN HỆ RỖNG

"tìm tất cả các khách hàng có cả vay lẫn tài khoản ở ngân hàng"

```
SELECT Customer_name
FROM Borrower
WHERE  EXISTS ( SELECT *
                FROM Depositor
                WHERE Depositor.Customer_name = Borrower.Customer_name)
```

Cấu trúc **EXISTS** trả lại giá trị true nếu quan hệ kết quả của câu vấn tin con không rỗng. SQL cũng cho phép sử dụng cấu trúc **NOT EXISTS** để kiểm tra tính không rỗng của một quan hệ.

"Tìm tất cả các khách hàng có tài khoản tại mỗi chi nhánh đóng tại Brooklyn"

```
SELECT DISTINCT S.Customer_name
FROM Depositor  AS S
WHERE NOT EXISTS      ( ( SELECT Branch_name
                          FROM Branch
                          WHERE Branch_city = 'Brooklyn')
                        EXCEPT
                          ( SELECT R.branch_name
                            FROM Depositor AS T, Account AS R
                            WHERE    T.Acoount_number = R.Account_number
                            AND  S.Customer_name = T.Customer_name) )
```

THỬ KHÔNG CÓ CÁC BỘ TRÙNG

SQL đưa vào cấu trúc **UNIQUE** để kiểm tra việc có bộ trùng trong quan hệ kết quả của một câu vấn tin con.

"Tìm tất cả khách hàng chỉ có một tài khoản ở chi nhánh Perryridge"

```
SELECT T.Customer_name
FROM Depositor AS T
WHERE UNIQUE ( SELECT   R.Customer_name
               FROM Account, Depositor AS  R
               WHERE T.Customer_name = R.Customer_name AND
                     R.Account_number    =    Account.Acount_number
                     AND   Account.Branch_name = 'Perryridge')
```

Ta có thể thử sự tồn tại của các bộ trùng trong một vấn tin con bằng cách sử dụng cấu trúc **NOT UNIQUE**

"Tìm tất cả các khách hàng có ít nhất hai tài khoản ở chi nhánh Perryridge"

```
SELECT DISTINCT T.Customer_name
FROM  Account, Depositor AS T
WHERE NOT UNIQUE ( SELECT R.Customer_name
                   FROM Account, Depositor AS  R
                   WHERE T.Customer_name=R.Customer_name
                AND   R.Account_number = Account.Account_number
                AND   Account.Branch_name = 'Perryridge')
```

UNIQUE trả lại giá rị false khi và chỉ khi quan hệ có hai bộ trùng nhau. Nếu hai bộ t_1, t_2 có ít nhất một trường null, phép so sánh $t_1 = t_2$ cho kết quả false. Do vậy UNIQUE có thể trả về giá trị true trong khi quan hệ có nhiều bộ trùng nhau nhưng chứa trường giá trị null !

QUAN HỆ DẪN XUẤT

SQL92 cho phép một biểu thức vấn tin con được dùng trong mệnh đề FROM. Nếu biểu thức như vậy được sử dụng, quan hệ kết quả phải được cho một cái tên và các thuộc tính có thể được đặt tên lại (bằng mệnh đề AS)

Ví dụ câu vấn tin con:

```
(SELECT  Branch_name, AVG(Balance)
  FROM Account
  GROUP BY Branch_name)
AS result (Branch_name, Avg_balace)
```

Sinh ra quan hệ gồm tên của tất cả các chi nhánh, và số cân đối trung bình tương ứng. Quan hệ này được đặt tên là result với hai thuộc tính Branch_name và Avg_balance.

"Tìm số cân đối tài sản trung bình của các chi nhánh tại đó số cân đối tài khoản trung bình lớn hơn 1200$"

```
SELECT Branch_name, avg_balance
FROM  ( SELECT  Branch_name, AVG(Balance)
          FROM Account
        GROUP BY Branch_name)
        AS result (Branch_name, Avg_balace)
WHERE avg_balance > 1200
```

VIEWS

Trong SQL, để định nghĩa view ta sử dụng lệnh **CREATE VIEW**. Một view phải có một tên.

CREATE VIEW < tên view > AS < Biểu thức vấn tin >

"Tạo một view gồm các tên chi nhánh, tên của các khách hàng có hoặc một tài khoản hoặc vay ở chi nhánh này"

Giả sử ta muốn đặt tên cho view này là All_customer.

```
CREATE VIEW All_customer        AS
(       SELECT Branch_name, Customer_name
        FROM Depositor, Account
        WHERE Depositor.Account_number = Account.Account_number )
UNION
(       SELECT Branch_name, Customer_name
        FROM Borrower, Loan
        WHERE Borrower.Loan_number = Loan.Loan_number)
```

Tên thuộc tính của một view có thể xác định một cách tường minh như sau:

```
CREATE VIEW Branch_total_loan (Branch_name, Total_loan)      AS
        ( SELECT Branch_name, sum(Amount)
          FROM Loan
          GROUP BY Branch_name)
```

Một view là một quan hệ, nó có thể tham gia vào các câu vấn tin với vai trò của một quan hệ.

```
SELECT Customer_name
FROM All_customer
WHERE Branch_name = 'Perryridge'
```

Một câu vấn tin phức tạp sẽ dễ hiểu hơn, dễ viết hơn nếu ta cấu trúc nó bằng cách phân tích nó thành các view nhỏ hơn và sau đó tổ hợp lại.

Định nghĩa view được giữ trong CSDL đến tận khi một lệnh DROP VIEW < tên view > được gọi.

Trong chuẩn SQL 3 hiện đang được phát triển bao hàm một đề nghị hỗ trợ những view tạm không được lưu trong CSDL.

SỬA ĐỔI CƠ SỞ DỮ LIỆU

DELETE
INSERT
UPDATE

XÓA (Delete)

Ta chỉ có thể xoá nguyên vẹn một bộ trong một quan hệ, không thể xoá các giá trị của các thuộc tính. Biểu thức xoá trong SQL là:

```
DELETE FROM  r
[WHERE  P]
```

Trong đó p là một vị từ và r là một quan hệ.

Lệnh DELETE duyệt qua tất cả các bộ t trong quan hệ r, nếu P(t) là true, DELETE xoá t khỏi r. Nếu không có mệnh đề WHERE, tất cả các bộ trong r bị xoá.

Lệnh DELETE chỉ hoạt động trên một quan hệ.

- DELETE FROM Loan = Xoá tất cả các bộ của quan hệ Loan
- DELETE FROM Depositor WHERE Customer_name = 'Smith'
- DELETE FROM Loan
  WHERE Amount BETWEEN 1300 AND 1500
- DELETE FROM Account
  WHERE Branch_name IN (SELECT Branch_name
  FROM Branch
  WHERE Branch_city = 'Brooklyn')
- DELETE FROM Account
  WHERE Balance < (SELECT AVG(Balance)
  FROM Account)

XEN (Insert)

Để xen dữ liệu vào một quan hệ, ta xác định một bộ cần xen hoặc viết một câu vấn tin kết quả của nó là một tập các bộ cần xen. Các giá trị thuộc tính của bộ cần xen phải thuộc vào miền giá trị của thuộc tính và số thành phần của bộ phải bằng với ngôi của quan hệ.

"Xen vào quan hệ Account một bộ có số tài khoản là A-9732, số cân đối là 1200$ và tài khoản này được mở ở chi nhánh Perryridge"

```
INSERT INTO Account
VALUES ('Perryridge', 'A-9732', 1200);
```

Trong ví dụ này thự tự các giá trị thuộc tính cần xen trùng khớp với thứ tự các thuộc tính trong sơ đồ quan hệ. SQL cho phép chỉ rõ các thuộc tính và các giá trị tương ứng cần xen:

```
INSERT INTO Account (Branch_name, Account_number, Balance)
VALUES ('Perryridge', 'A-9732', 1200);

INSERT INTO Account (Account_number, Balance, Branch_name)
VALUES ('A-9732', 1200, 'Perryridge');
```

"Cấp cho tất cả các khách hàng vay ở chi nhánh Perryridge một tài khoản với số cân đối là 200$ như một quà tặng sử dụng số vay như số tài khoản"

```
INSERT INTO Account
        SELECT Branch_name, Loan_number, 200
        FROM Loan
        WHERE Branch_name = 'Perryridge'
INSERT INTO Depositor
        SELECT Customer_name, Loan_number
        FROM Borrower, Loan
        WHERE Borrower.Loan_number = Loan.Loan_number AND
                Branch_name = 'Perryridge'
```

CẬP NHẬT (Update)

Câu lệnh UPDATE cho phép thay đổi giá trị thuộc tính của các bộ

"Thêm lãi hàng năm vào số cân đối với tỷ lệ lãi suất 5%"

```
UPDATE Account
SET Balance = Balance*1.05
```

Giả sử các tài khoản có số cân đối > 10000$ được hưởng lãi suất 6%, các tài khoản có số cân đối nhỏ hơn hoặc bằng 10000 được hưởng lãi suất 5%

```
UPDATE Account
SET Balance = Balance*1.06
WHERE Balance > 10000
UPDATE Account
SET Balance = Balance*1.05
WHERE Balance <= 10000
```

SQL92 đưa vào cấu trúc CASE như sau:

```
CASE
        WHEN P1 THEN Result1
        WHEN P2 THEN Result2
        ...
        WHEN Pn THEN Resultn
        ELSE  Result0
END
```

trong đó P_i là các vị từ, $Result_i$ là các kết quả trả về của hoạt động CASE tương ứng với vị từ P_i đầu tiên thỏa mãn. Nếu không vị từ P_i nào thỏa mãn CASE trả về $Result_0$.

Với cấu trúc CASE như vậy ta có thể viết lại yêu cầu trên như sau:

```
UPDATE Account
SET Balance = CASE
        WHEN Balance > 10000 THEN Balance*1.06
        ELSE Balance*1.05
END
```

"Trả 5% lãi cho các tài khoản có số cân đối lớn hơn số cân đối trung bình"

```
UPDATE Account
```

```
SET Balance = Balance*1.05
WHERE Balance >  SELECT AVG(Balance)
                 FROM Account
```

CÁC QUAN HỆ NỐI

SQL92 cung cấp nhiều cơ chế cho nối các quan hệ bao hàm nối có điều kiện và nối tự nhiên cũng như các dạng của nối ngoài.

```
Loan INNER JOIN Borrower
ON Loan.Loan_number = Borrower.Loan_number
```

Nối quan hệ Loan và quan hệ Borrower với điều kiên:

```
Loan.Loan_number = Borrower.Loan_number
```

Quan hệ kết quả có các thuộc tính của quan hệ Loan và các thuộc tính của quan hệ Borrower (như vậy thuộc tính Loan_number xuất hiện 2 lần trong quan hệ kết quả).

Để đổi tên quan hệ (kết quả) và các thuộc tính, ta sử dụng mệnh đề **AS**

```
Loan INNER JOIN Borrower
ON Loan.Loan_number = Borrower.Loan_number
AS  LB(Branch, Loan_number, Amount, Cust, Cust_Loan_number)
Loan LEFT OUTER JOIN Borrower
ON Loan.Loan_number = Borrower.Loan_number
```

Phép nối ngoài trái được tính như sau: Đầu tiên tính kết quả của nối trong INNER JOIN. Sau đó đối với mỗi bộ t của quan hệ trái (Loan) không tương xứng với bộ nào trong quan hệ bên phải (borrower) khi đó thêm vào kết quả bộ r gồm các giá trị thuộc tính trái là các giá trị thuộc tính của t, các thuộc tính còn lại (phải) được đặt là null.

```
Loan NATURAL INNER JOIN Borrower
```

Là nối tự nhiên của quan hệ Loan và quan hệ Borrower (thuộc tính trùng tên là Loan_number).

NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU (DDL)

DDL SQL cho phép đặc tả:

- Sơ đồ cho mỗi quan hệ
- Miền giá trị kết hợp với mỗi thuộc tính
- các ràng buộc toàn vẹn
- tập các chỉ mục được duy trì cho mỗi quan hệ
- thông tin về an toàn và quyền cho mỗi quan hệ
- cấu trúc lưu trữ vật lý của mỗi quan hệ trên đĩa

CÁC KIỂU MIỀN TRONG SQL

SQL-92 hỗ trợ nhiều kiểu miền trong đó bao hàm các kiểu sau:

- **char(n) / charater:** chuỗi ký tự dộ dài cố định, với độ dài n được xác định bởi người dùng
- **vachar(n) / character varying (n):** chuỗi ký tự độ dài thay đổi, với độ dài tối đa được xác dịnh bởi người dung là n
- **int / integer:** tập hữu hạn các số nguyên
- **smallint:** tập con của tập các số nguyên **int**
- **numeric(p, d):** số thực dấu chấm tĩnh gồm p chữ số (kể cả dấu) và d trong p chữ số là các chữ số phần thập phân
- **real, double precision:** số thực dấu chấm động và số thực dấu chấm động chính xác kép
- **float(n):** số thực dấu chấm động với độ chính xác được xác định bởi người dùng ít nhất là n chữ số thập phân

- **date:** kiểu năm tháng ngày (YYYY, MM, DD)
- **time:** kiểu thời gian (HH, MM, SS)

SQL-92 cho phép định nghĩa miền với cú pháp:

CREATE DOMAIN < tên miền > < Type >

Ví dụ: CREATE DOMAIN hoten char(30);

Sau khi đã định nghĩa miền với tên hoten ta có thể sử dụng nó để định nghĩa kiểu của các thuộc tính

ĐỊNH NGHĨA SƠ ĐỒ TRONG SQL.

Lệnh **CREATE TABLE** với cú pháp

CREATE TABLE < tên bảng > (
< Thuộc tính 1 > < miền giá trị thuộc tính 1 > ,
...
< Thuộc tính n > < miền giá trị thuộc tính n> ,
< ràng buộc toàn vẹn 1 > ,
...
< ràng buộc toàn vẹn k >)

Các ràng buộc toàn vẹn cho phép bao gồm:

primary key ($A_{i_1}, A_{i_2}, ..., A_{i_m}$ **)**

và

check(P)

Đặc tả **primary key** chỉ ra rằng các thuộc tính $A_{i_1}, A_{i_2}, ..., A_{i_m}$ tạo nên khoá chính của quan hệ. Mệnh đề **check** xác định một vị từ P mà mỗi bộ trong quan hệ phải thoả mãn.

Ví dụ:

```
CREATE TABLE customer (
        customer_name       CHAR(20) not null,
        customer_street     CHAR(30),
        customer_city       CHAR(30),
        PRIMARY KEY(customer_name));
CREATE TABLE branch (
        branch_name         CHAR(15) not null,
        branch_city         CHAR(30),
        assets              INTEGER,
        PRIMARY KEY (branch_name),
        CHECK (assets >= 0));
CREATE TABLE account (
        account_number      CHAR(10) not null,
        branch_name         CHAR(15),
        balance             INTEGER,
        PRIMARY KEY (account_number),
        CHECK       (balance >= 0));
CREATE TABLE depositor (
        customer_name       CHAR(20) not null,
        account_number      CHAR(10) not null,
        PRIMARY KEY (customer_name, account_namber));
```

Giá trị **null** là giá trị hợp lệ cho mọi kiểu trong SQL. Các thuộc tính được khai báo là primary key đòi hỏi phải là ***not null*** và ***duy nhất***. do vậy các khai báo not null trong ví dụ trên là dư (trong SQL-92).

```
CREATE TABLE student (
```

```
name            CHAR(15) not null,
student_ID      CHAR(10) not null,
degree_level    CHAR(15) not null,
PRIMARY KEY (student_ID),
CHECK (degree_level  IN  ('Bachelors', 'Masters', 'Doctorats'));
```

- Xoá một quan hệ khỏi CSDL sử dụng lệnh **Drop table** với cú pháp:
  DROP TABLE < tên bảng >
- Thêm thuộc tính vào bảng đang tồn tại sử dụng lệnh **Alter table** với cú pháp:
  ALTER TABLE < tên bảng > ADD < thuộc tính > < miền giá trị >
- Xoá bỏ một thuộc tính khỏi bảng đang tồn tại sử dụng lệnh **Alter table** với cú pháp:
  ALTER TABLE < Tên bảng > DROP < tên thuộc tính >

SQL NHÚNG (Embedded SQL)

Một ngôn ngữ trong đó các vấn tin SQL được nhúng gọi là ngôn ngữ chủ *(host language)*, cấu trúc SQL cho phép trong ngôn ngữ chủ tạo nên SQL nhúng. Chương trình được viết trong ngôn ngữ chủ có thể sử dụng cú pháp SQL nhúng để truy xuất và cập nhật dữ liệu được lưu trữ trong CSDL.

BÀI TẬP CHƯƠNG II

II.1. Xét CSDL bảo hiểm sau:

person(<u>ss#</u>, name, address): Số bảo hiểm ss# sở hữu bởi người tên name ở địa chỉ address

car(<u>license</u>, <u>year</u>, model): Xe hơi số dăng ký license, sản xuất năm year, nhãn hiệu Model

accident(<u>date</u>, <u>driver</u>, damage_amount): tai nạn xảy ra ngày date, do người lái driver, mức hư hại damage_amount

owns(<u>ss#</u>, <u>license</u>): người mang số bảo hiểm ss# sở hữu chiếc xe mang số đăng ký license

log(<u>license</u>, <u>date</u>, <u>driver</u>): ghi sổ chiếc xe mang số đăng ký license, bị tai nạn ngày do người lái driver

các thuộc tính được gạch dưới là các primary key. Viết trong SQL các câu vấn tin sau:

1. Tìm tổng số người xe của họ gặp tai nạn năm 2001
2. Tìm số các tai nạn trong đó xe của"John"liên quan tới
3. Thêm khách hàng mới: ss# =”A-12345”, name ="David”, address ="35 Chevre Road”, license ="109283”, year =”2002”, model ="FORD LASER"vào CSDL
4. xoá các thông tin lien quan dến xe model "MAZDA"của"John Smith”
5. Thêm thông tin tai nạn cho chiếc xe"TOYOTA"của khách hàng mang số bảo hiểm số"A-84626”

II.2. Xét CSDL nhân viên:

employee (E_name, street, city): Nhân viên có tên E_name, cư trú tại phố street, trong thành phố city

works (E_name, C_name, salary): Nhân viên tên E_name làm việc cho công ty C_name với mức lương salary

copany (C_name, city): Công ty tên C_name đóng tại thành phố city

manages(E_name, M_name): Nhân viên E_name dưới sự quản lý của nhân viên M_name

Viết trong SQL các câu vấn tin sau:

1. Tìm tên của tất cả các nhân viên làm việc cho First Bank
2. Tìm tên và thành phố cư trú của các nhân viên làm việc cho First Bank
3. Tìm tên, phố, thành phố cư trú làm việc cho First Bank hưởng mức lương > 10000$
4. Tìm tất cả các nhân viên trong CSDL sống trong cùng thành phố với công ty mang họ làm việc cho
5. Tìm tất cả các nhân viên sông trong cùng thành phố, cùng phố với người quản lý của họ
6. Tìm trong CSDL các nhân viên không làm việc cho First Bank
7. Tìm trong CSDL, các nhân viên hưởng mức lương cao hơn mọi nhân viên của Small Bank
8. Giả sử một công ty có thể đóng trong một vaì thành phố. Tìm tất cả các công ty đóng trong mỗi thành phố trong đó Small Bank đóng.
9. Tìm tất cả các nhân viên hưởng múc lương cao hơn mức lương trung bình của công ty họ làm việc
10. Tìm công ty có nhiều nhân viên nhất
11. Tìm công ty có tổng số tiền trả lương nhỏ nhất
12. Tìm tất cả các công ty có mức lương trung bình cao hơn mức luong trung bình của công ty First Bank
13. Thay đổi thành phố cư trú của nhân viên"Jones"thành NewTown
14. Nâng lương cho tất cả các nhân viên của First Bank lên 10%
15. nâng lương cho các nhà quản lý của công ty First Bank lên 10%
16. Xoá tất cả các thông tin liên quan tới cong ty Bad Bank

CHƯƠNG III

LƯU TRỮ VÀ CẤU TRÚC TẬP TIN
(Storage and File Structure)

MỤC ĐÍCH

Chương này trình bày các vấn đề liên quan đến vấn đề lưu trữ dữ liệu (trên lưu trữ ngoài, chủ yếu trên đĩa cứng). Việc lưu trữ dữ liệu phải được tổ chức sao cho có thể cất giữ một lượng lớn, có thể rất lớn dữ liệu nhưng quan trọng hơn cả là sự lưu trữ phải cho phép lấy lại dữ liệu cần thiết mau chóng. Các cấu trúc trợ giúp cho truy xuất nhanh dữ liệu được trình bày là: chỉ mục (indice), B^+ cây (B^+-tree), băm (hashing) ... Các thiết bị lưu trữ (đĩa) có thể bị hỏng hóc không lường trước, các kỹ thuật RAID cho ra một giải pháp hiệu quả cho vấn đề này.

YÊU CẦU

Hiểu rõ các đặc điểm của các thiết bị lưu trữ, cách tổ chức lưu trữ, truy xuất đĩa.

Hiểu rõ nguyên lý và kỹ thuật của tổ chức hệ thống đĩa RAID

Hiểu rõ các kỹ thuật tổ chức các mẩu tin trong file

Hiểu rõ các kỹ thuật tổ chức file

Hiểu và vận dụng các kỹ thuật hỗ trợ tìm lại nhanh thông tin: chỉ mục (được sắp, B^+-cây, băm)

KHÁI QUÁT VỀ PHƯƠNG TIỆN LƯU TRỮ VẬT LÝ

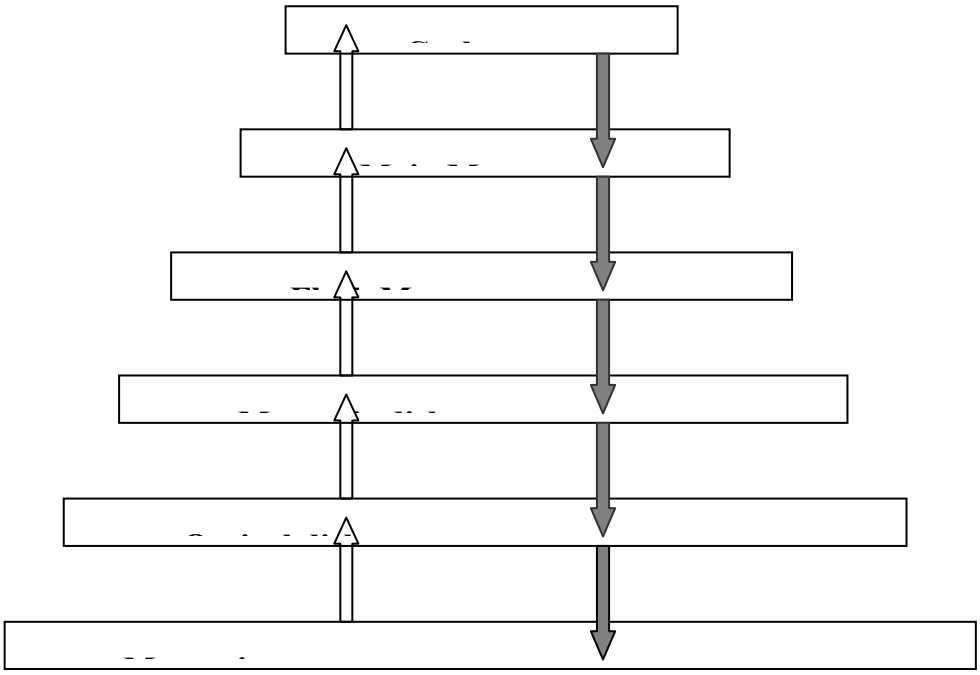
Có một số kiểu lưu trữ dữ liệu trong các hệ thống máy tính. Các phương tiện lưu trữ được phân lớp theo *tốc độ truy xuất, theo giá cả và theo độ tin cậy của phương tiện*. Các phương tiện hiện có là:

- **Cache**: là dạng lưu trữ nhanh nhất và cũng đắt nhất trong các phương tiện lưu trữ. Bộ nhớ cache nhỏ; sự sử dụng nó được quản trị bởi hệ điều hành
- **Bộ nhớ chính (main memory)**: Phương tiện lưu trữ dùng để lưu trữ dữ liệu sẵn sàng được thực hiện. Các chỉ thị máy mục đích chung (general-purpose) hoạt động trên bộ nhớ chính. Mặc dầu bộ nhớ chính có thể chứa nhiều megabytes dữ liệu, nó vẫn là quá nhỏ (và quá đắt giá) để lưu trữ toàn bộ một cơ sở dữ liệu. Nội dung trong bộ nhớ chính thường bị mất khi mất cấp nguồn
- **Bộ nhớ Flash:** Được biết như bộ nhớ chỉ đọc có thể lập trình, có thể xoá (EEPROM: Electrically Erasable Programmable Read-Only Memory), Bộ nhớ Flash khác bộ nhớ chính ở chỗ dữ liệu còn tồn tại trong bộ nhớ flash khi mất cấp nguồn. Đọc dữ liệu từ bộ nhớ flash mất ít hơn 100 ns , nhanh như đọc dữ liệu từ bộ nhớ chính. Tuy nhiên, viết dữ liệu vào bộ nhớ flash phức tạp hơn nhiều. Dữ liệu được viết (một lần mất khoảng 4 đến 10 μs) nhưng không thể viết đè trực tiếp. Để viết đè bộ nhớ đã được viết, ta phải xoá trắng toàn bộ bộ nhớ sau đó mới có thể viết lên nó.
- **Lưu trữ đĩa từ (magnetic-disk): (**ở đây, được hiểu là đĩa cứng) Phương tiện căn bản để lưu trữ dữ liệu trực tuyến, lâu dài. Thường toàn bộ cơ sở dữ liệu được lưu trữ trên đĩa từ. Dữ liệu phải được chuyển từ đĩa vào bộ nhớ chính trước khi được truy nhập. Khi dữ liệu trong bộ nhớ chính này bị sửa đổi, nó phải được viết lên đĩa. Lưu trữ đĩa được xem là truy xuất trực tiếp vì có thể đọc dữ liệu trên đĩa theo một thứ tự bất kỳ. Lưu trữ đĩa vẫn tồn tại khi mất cấp nguồn. Lưu trữ đĩa có thể bị hỏng hóc, tuy không thường xuyên.
- **Lưu trữ quang (Optical storage):** Dạng quen thuộc nhất của đĩa quang học là loại đĩa **CD-ROM** : Compact-Disk Read-Only Memory. Dữ liệu được lưu trữ trên các đĩa quang học được đọc bởi laser. Các đĩa quang học CD-ROM chỉ có thể đọc. Các phiên bản khác của chúng là loại đĩa quang học: viết một lần, đọc nhiều lần (write-once, read-many: **WORM**) cho phép viết dữ liệu lên đĩa một lần, không cho phép xoá và viết lại, và các đĩa có thể viết lại (rewritable) v..v
- **Lưu trữ băng từ (tape storage):** Lưu trữ băng từ thường dùng để backup dữ liệu. Băng từ rẻ hơn đĩa, truy xuất dữ liệu chậm hơn (vì phải truy xuất tuần tự). Băng từ thường có dung lượng rất lớn.

Các phương tiện lưu trữ có thể được tổ chức phân cấp theo tốc độ truy xuất và giá cả. Mức cao nhất là nhanh nhất nhưng cũng là đắt nhất, giảm dần xuống các mức thấp hơn.

Các phương tiện lưu trữ nhanh (*cache, bộ nhớ chính*) được xem như là lưu trữ sơ cấp (primary storage), các thiết bị lưu trữ ở mức thấp hơn như đĩa từ được xem như lưu trữ thứ cấp hay lưu trữ trực tuyến (on-line storage), còn các thiết bị lưu trữ ở mức thấp nhất và gần thấp nhất như đĩa quang học, băng từ kể cả các đĩa mềm được xếp vào lưu trữ tam cấp hay lưu trữ không trực tuyến (off-line).

Bên cạnh vấn đề tốc độ và giá cả, ta còn phải xét đến tính lâu bền của các phương tiện lưu trữ.



Phân cấp thiết bị lưu trữ

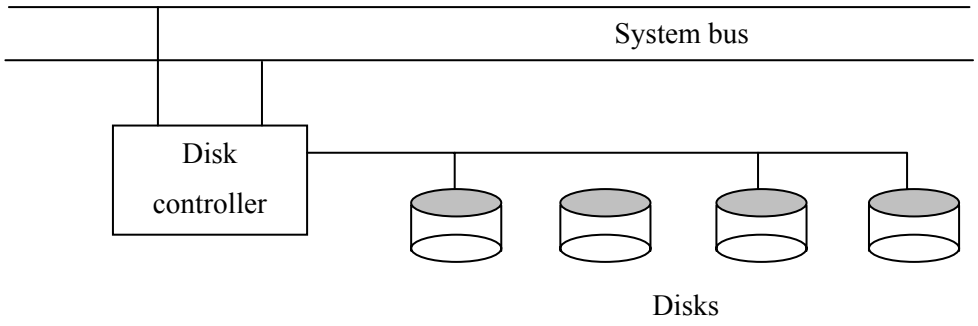
ĐĨA TỪ

ĐẶC TRƯNG VẬT LÝ CỦA ĐĨA

Mỗi tấm đĩa có dạng hình tròn, hai mặt của nó được phủ bởi vật liệu từ tính, thông tin được ghi trên bề mặt đĩa. Đĩa gồm nhiều tấm đĩa. Ta sẽ sử dụng thuật ngữ đĩa để chỉ các đĩa cứng.

Khi đĩa được sử dụng, một động cơ ổ đĩa làm quay nó ở một tốc độ không đổi. Một đầu đọc-viết được định vị trên bề mặt của tấm đĩa. Bề mặt tấm đĩa được chia logic thành các rãnh, mỗi rãnh lại được chia thành các sector, một sector là một đơn vị thông tin nhỏ có thể được đọc, viết lên đĩa. Tuỳ thuộc vào kiểu đĩa, sector thay đổi từ 32 bytes đến 4095 bytes, thông thường là 512 bytes. Có từ 4 đến 32 sectors trên một rãnh, từ 20 đến 1500 rãnh trên một bề mặt. Mỗi bề mặt của một tấm đĩa có một đầu đọc viết, nó có thể chạy dọc theo bán kính đĩa để truy cập đến các rãnh khác nhau. Một đĩa gồm nhiều tấm đĩa, các đầu đọc-viết của tất cả các rãnh được gắn vào một bộ được gọi là cánh tay đĩa, di chuyển cùng nhau. Các tấm đĩa được gắn vào một trục quay. Vì các đầu đọc-viết trên các tấm đĩa di chuyển cùng nhau, nên khi đầu đọc-viết trên một tấm đĩa đang ở rãnh thứ i thì các đầu đọc-viết của các tấm đĩa khác cũng ở rãnh thứ i , do vậy các rãnh thứ i của tất cả các tấm đĩa được gọi là trụ (cylinder) thứ i . Một bộ điều khiển đĩa -- giao diện giữa hệ thống máy tính và phần cứng hiện thời của ổ đĩa. Nó chấp nhận các lệnh mức cao để đọc và viết một sector, và khởi động các hành động như di chuyển cánh tay đĩa đến các rãnh đúng và đọc viết dữ liệu. bộ điều khiển đĩa cũng tham gia vào checksum mỗi sector được viết. Checksum được tính từ dữ liệu được viết lên sector. Khi sector được đọc lại, checksum được tính lại từ dữ liệu được lấy ra và so sánh với checksum đã lưu trữ. Nếu dữ liệu bị sai lạc, checksum được tính sẽ không khớp với checksum đã lưu trữ. Nếu lỗi như vậy xảy ra, bộ điều khiển sẽ lặp lại việc đọc vài lần, nếu lỗi vẫn xảy ra, bộ điều khiển sẽ thông báo việc đọc thất bại. Bộ điều khiển đĩa còn có

chức năng tái ánh xạ các sector xấu: ánh xạ các sector xấu đến một vị trí vật lý khác. Hình dưới bày tỏ các đĩa được nối với một hệ thống máy tính:



Các đĩa được nối với một hệ thống máy tính hoặc một bộ điều khiển đĩa qua một sự hợp nhất tốc độ cao. Hợp nhất hệ thống máy tính nhỏ (Small Computer-System Interconnect: SCSI) thường được sử dụng để nối kết các đĩa với các máy tính cá nhân và workstation. Mainframe và các hệ thống server thường có các bus nhanh hơn và đắt hơn để nối với các đĩa.

Các đầu đọc-viết được giữ sát với bề mặt đĩa như có thể để tăng độ dày đặc (density).

Đĩa đầu cố định (Fixed-head) có một đầu riêng biệt cho mỗi rãnh, sự sắp xếp này cho phép máy tính chuyển từ rãnh này sang rãnh khác mau chóng, không phải di chuyển đầu đọc-viết. Tuy nhiên, cần một số rất lớn đầu đọc-viết, điều này làm nâng giá của thiết bị.

ĐO LƯỜNG HIỆU NĂNG CỦA ĐĨA

Các tiêu chuẩn đo lường chất lượng chính của đĩa là ***dung lượng, thời gian truy xuất, tốc độ truyền dữ liệu và độ tin cậy***.

- ***Thời gian truy xuất (access time)***: là khoảng thời gian từ khi yêu cầu đọc/viết được phát đi đến khi bắt đầu truyền dữ liệu. Để truy xuất dữ liệu trên một sector đã cho của một đĩa, đầu tiên cánh tay đĩa phải di chuyển đến rãnh đúng, sau đó phải chờ sector xuất hiện dưới nó, thời gian để định vị cánh tay được gọi là thời gian tìm kiếm (seek time), nó tỷ lệ với khoảng cách mà cánh tay phải di chuyển, thời gian tìm kiếm nằm trong khoảng 2..30 ms tuỳ thuộc vào rãnh xa hay gần vị trí cánh tay hiện tại.
- ***Thời gian tìm kiếm trung bình (average seek time)***: Thời gian tìm kiếm trung bình là trung bình của thời gian tìm kiếm, được đo luờng trên một dãy các yêu cầu ngẫu nhiên (phân phối đều), và bằng khoảng 1/3 thời gian tìm kiếm trong trường hợp xấu nhất.
- ***Thời gian tiềm ẩn luân chuyển (rotational latency time)***: Thời gian chờ sector được truy xuất xuất hiện dưới đầu đọc/viết. Tốc độ quay của đĩa nằm trong khoảng 60..120 vòng quay trên giây, trung bình cần nửa vòng quay để sector cần thiết nằm dưới đầu đọc/viết. Như vậy, thời gian tiềm ẩn trung bình (average latency time) bằng nửa thời gian quay một vòng đĩa.

Thời gian truy xuất bằng tổng của thời gian tìm kiếm và thời gian tiềm ẩn và nằm trong khoảng 10..40 ms.

- ***Tốc độ truyền dữ liệu:*** là tốc độ dữ liệu có thể được lấy ra từ đĩa hoặc được lưu trữ vào đĩa. Hiện nay tốc này vào khoảng1..5 Mbps

- ***Thời gian trung bình không sự cố (mean time to failure):*** lượng thời gian trung bình hệ thống chạy liên tục không có bất kỳ sự cố nào. Các đĩa hiện nay có thời gian không sự cố trung bình khoảng 30000 .. 800000 giờ nghĩa là khoảng từ 3,4 đến 91 năm.

TỐI ƯU HÓA TRUY XUẤT KHỐI ĐĨA (disk-block)

Yêu cầu I/O đĩa được sinh ra cả bởi hệ thống file lẫn bộ quản trị bộ nhớ ảo trong hầu hết các hệ điều hành. Mỗi yêu cầu xác định địa chỉ trên đĩa được tham khảo, địa chỉ này ở dạng số khối. Một khối là một dãy các sector kề nhau trên một rãnh. Kích cỡ khối trong khoảng 512 bytes đến một vài Kbytes. Dữ liệu được truyền giữa đĩa và bộ nhớ chính theo đơn vị khối. Mức thấp hơn của bộ quản trị hệ thống file sẽ chuyển đổi địa chỉ khối sang số của trụ, của mặt và của sector ở mức phần cứng.

Truy xuất dữ liệu trên đĩa chậm hơn nhiều so với truy xuất dữ liệu trong bộ nhớ chính, do vậy cần thiết một chiến lược nhằm nâng cao tốc độ truy xuất khối đĩa. Dưới đây ta sẽ thảo luận một vài kỹ thuật nhằm vào mục đích đó.

- ***Scheduling:*** Nếu một vài khối của một trụ cần được truyền từ đĩa vào bộ nhớ chính, ta có thể tiết kiệm thời gian truy xuất bởi yêu cầu các khối theo thứ tự mà nó chạy qua dưới đầu đọc/viết. Nếu các khối mong muốn ở trên các trụ khác nhau, ta yêu cầu các khối theo thứ tự sao cho làm tối thiểu sự di chuyển cánh tay đĩa. Các thuật toán scheduling cánh tay đĩa (Disk-arm-scheduling) nhằm lập thứ tự truy xuất các rãnh theo cách làm tăng số truy xuất có thể được xử lý. Một thuật toán thường dùng là thuật toán thang máy (elevator algorithm): Giả sử ban đầu cánh tay di chuyển từ rãnh trong nhất hướng ra phía ngoài đĩa, đối với mỗi rãnh có yêu cầu truy xuất, nó dừng lại, phục vụ yêu cầu đối với rãnh này, sau đó tiếp tục di chuyển ra phía ngoài đến tận khi không có yêu cầu nào chờ các rãnh xa hơn phía ngoài. Tại điểm này, cánh tay đổi hướng, di chuyển vào phía trong, lại dừng lại trên các rãnh được yêu cầu, và cứ như vậy đến tận khi không còn rãnh nào ở trong hơn được yêu cầu, rồi lại đổi hướng .. v .. v .. Bộ điều khiển đĩa thường làm nhiệm vụ sắp xếp lại các yêu cầu đọc để cải tiến hiệu năng.
- ***Tổ chức file:*** Để suy giảm thời gian truy xuất khối, ta có thể tổ chức các khối trên đĩa theo cách tương ứng gần nhất với cách mà dữ liệu được truy xuất. Ví dụ, Nếu ta muốn một file được truy xuất tuần tự, khi đó ta bố trí các khối của file một cách tuần tự trên các trụ kề nhau. Tuy nhiên việc phân bố các khối lưu trữ kề nhau này sẽ bị phá vỡ trong quá trình phát triển của file $\Rightarrow$ file không thể được phân bố trên các khối kề nhau được nữa, hiện tượng này dược gọi là sự phân mảnh (fragmentation). Nhiều hệ điều hành cung cấp tiện ích giúp suy giảm sự phân mảnh này (Defragmentation) nhằm làm tăng hiệu năng truy xuất file.
- ***Các buffers viết không hay thay đổi:*** Vì nội dung của bộ nhớ chính bị mất khi mất nguồn, các thông tin về cơ sở dữ liệu cập nhật phải được ghi lên đĩa nhằm đề phòng sự cố. Hiệu năng của các ứng dụng cập nhật cường độ cao phụ thuộc mạnh vào tốc độ viết đĩa. Ta có thể sử dụng bộ nhớ truy xuất ngẫu nhiên không hay thay đổi (nonvolatile RAM) để nâng tốc độ viết đĩa. Nội dung của nonvolatile RAM không bị mất khi mất nguồn. Một phương pháp chung để thực hiện nonvolatile RAM là sử dụng RAM pin dự phòng (battery-back-up RAM). Khi cơ sở dữ liệu yêu cầu viết một khối lên đĩa, bộ điều khiển dĩa viết khối này lên buffer nonvolatile RAM, và thông báo ngay cho hệ điều hành là việc viết đã thành công. Bộ điều khiển sẽ viết dữ liệu đến đích của nó trên đĩa, mỗi khi đĩa rảnh hoặc buffer nonvolatile RAM đầy. Khi hệ cơ sở dữ liệu yêu cầu một viết khối, nó chỉ chịu một khoảng lặng chờ đợi khi buffer nonvolatile RAM đầy.

- ***Đĩa log (log disk):*** Một cách tiếp cận khác để làm suy giảm tiềm năng viết là sử dụng log-disk: Một đĩa được tận hiến cho việc viết một log tuần tự. Tất cả các truy xuất đến log-disk là tuần tự, nhằm loại bỏ thời gian tìm kiếm, và một vài khối kề có thể được viết một lần, tạo cho viết vào log-disk nhanh hơn viết ngẫu nhiên vài lần. Cũng như trong trường hợp sử dụng nonvolatile RAM, dữ liệu phải được viết vào vị trí hiện thời của chúng trên đĩa, nhưng việc viết này có thể được tiến hành mà hệ cơ sở dữ liệu không cần thiết phải chờ nó hoàn tất. Log-disk có thể được sử dụng để khôi phục dữ liệu. Hệ thống file dựa trên log là một phiên bản của cách tiếp cận log-disk: Dữ liệu không được viết lại lên đích gốc của nó trên đĩa; thay vào đó, hệ thống file lưu vết nơi các khối được viết mới đây nhất trên log-disk, và hoàn lại chúng từ vị trí này. Log-disk được "cô đặc" lại (compacting) theo một định kỳ. Cách tiếp cận này cải tiến hiệu năng viết, song sinh ra sự phân mảnh đối với các file được cập nhật thường xuyên.

RAID

Trong một hệ thống có nhiều đĩa, ta có thể cải tiến tốc độ đọc viết dữ liệu nếu cho chúng hoạt động song song. Mặt khác, hệ thống nhiều đĩa còn giúp tăng độ tin cậy lưu trữ bằng cách lưu trữ dư thừa thông tin trên các đĩa khác nhau, nếu một đĩa có sự cố dữ liệu cũng không bị mất. Một sự đa dạng các kỹ thuật tổ chức đĩa, được gọi là **RAID (Redundant Arrays of Inexpensive Disks)**, được đề nghị nhằm vào vấn đề tăng cường hiệu năng và độ tin cậy.

CẢI TIẾN ĐỘ TIN CẬY THÔNG QUA SỰ DƯ THỪA

Giải pháp cho vấn đề độ tin cậy là đưa vào sự dư thừa: lưu trữ thông tin phụ, bình thường không cần thiết, nhưng nó có thể được sử dụng để tái tạo thông tin bị mất khi gặp sự cố hỏng hóc đĩa, như vậy thời gian trung bình không sự cố tăng lên (xét tổng thể trên hệ thống đĩa).

Đơn giản nhất, là làm bản sao cho mỗi đĩa. Kỹ thuật này được gọi là mirroring hay shadowing. Một đĩa logic khi đó bao gồm hai đĩa vật lý, và mỗi việc viết được thực hiện trên cả hai đĩa. Nếu một đĩa bị hư, dữ liệu có thể được đọc từ đĩa kia. Thời gian trung bình không sự cố của đĩa mirror phụ thuộc vào thời gian trung bình không sự cố của mỗi đĩa và phụ thuộc vào thời gian trung bình được sửa chữa (mean time to repair): thời gian trung bình để một đĩa bị hư được thay thế và phục hồi dữ liệu trên nó.

CẢI TIẾN HIỆU NĂNG THÔNG QUA SONG SONG

Với đĩa mirror, tốc độ đọc có thể tăng lên gấp đôi vì yêu cầu đọc có thể được gửi đến cả hai đĩa. Với nhiều đĩa, ta có thể cải tiến tốc độ truyền bởi phân nhỏ (striping data) dữ liệu qua nhiều đĩa. Dạng đơn giản nhất là tách các bít của một byte qua nhiều đĩa, sự phân nhỏ này được gọi là sự phân nhỏ mức bit (bit-level striping). Ví dụ, ta có một dàn 8 đĩa, ta viết bít thứ i của một byte lên đĩa thứ i . dàn 8 đĩa này có thể được xử lý như một đĩa với các sector 8 lần lớn hơn kích cỡ thông thường, quan trọng hơn là tốc dộ truy xuất tăng lên tám lần. Trong một tổ chức như vậy, mỗi đĩa tham gia vào mỗi truy xuất (đọc/viết), như vậy, số các truy xuất có thể được xử lý trong một giây là tương tự như trên một đĩa, nhưng mỗi truy xuất có thể đọc/viết nhiều dữ liệu hơn tám lần.

Phân nhỏ mức bit có thể được tổng quát cho số đĩa là bội hoặc ước của 8, Ví dụ, ta có một dàn 4 đĩa, ta sẽ phân phối bít thứ i và bít thứ 4+i vào đĩa thứ i. Hơn nữa, sự phân nhỏ không nhất thiết phải ở mức bit của một byte. Ví dụ, trong sự phân nhỏ mức khối, các khối của một file được phân nhỏ qua nhiều đĩa, với n đĩa, khối thứ i có thể được phân phối qua đĩa (i mod n) + 1. Ta cũng

có thể phân nhỏ ở mức byte, sector hoặc các sector của một khối. Hai đích song song trong một hệ thống đĩa là:

1. Nạp nhiều truy xuất nhỏ cân bằng (truy xuất trang) sao cho lượng dữ liệu được nạp trong một đơn vị thời gian của truy xuất như vậy tăng lên.
2. Song song hoá các truy xuất lớn sao cho thời gian trả lời các truy xuất lớn giảm.

CÁC MỨC RAID

Mirroring cung cấp độ tin cậy cao, nhưng đắt giá. Phân nhỏ cung cấp tốc độ truyền dữ liệu cao, nhưng không cải tiến được độ tin cậy. Nhiều sơ đồ cung cấp sự dư thừa với giá thấp bằng cách phối hợp ý tưởng của phân nhỏ với "parity" bit. Các sơ đồ này có sự thoả hiệp *giá-hiệu năng* khác nhau và được phân lớp thành các mức được gọi là các mức RAID.

• **Mức RAID 0 :** Liên quan đến các dàn đĩa với sự phân nhỏ mức khối, nhưng không có một sự dư thừa nào.

• **Mức RAID 1 :** Liên quan đến mirror đĩa

• **Mức RAID 2 :** Cũng được biết dưới cái tên mã sửa lỗi kiểu bộ nhớ (memory-style error-correcting-code : ECC). Hệ thống bộ nhớ thực hiện phát hiện lỗi bằng bit parity. Mỗi byte trong hệ thống bộ nhớ có thể có một bit parity kết hợp với nó. Sơ đồ sửa lỗi lưu hai hoặc nhiều hơn các bit phụ, và có thể dựng lại dữ liệu nếu một bit bị lỗi. ý tưởng của mã sửa lỗi có thể được sử dụng trực tiếp trong dàn đĩa thông qua phân nhỏ byte qua các đĩa. Ví dụ, bít đầu tiên của mỗi byte có thể được lưu trên đĩa 1, bit thứ hai trên đĩa 2, và cứ như vậy, bit thứ 8 trên đĩa 8, các bit sửa lỗi được lưu trên các đĩa thêm vào. Nếu một trong các đĩa bị hư, các bít còn lại của byte và các bit sửa lỗi kết hợp được đọc từ các đĩa khác có thể giúp tái tạo bít bị mất trên đĩa hư, như vậy ta có thể dựng lại dữ liệu. Với một dàn 4 đĩa dữ liệu, RAID mức 2 chỉ cần thêm 3 đĩa để lưu các bit sửa lỗi (các đĩa thêm vào này được gọi là các đĩa overhead), so sánh với RAID mức 1, cần 4 đĩa overhead.

• **Mức RAID 3 :** Còn được gọi là tổ chức parity chen bit (bit-interleaved parity). Bộ điều khiển đĩa có thể phát hiện một sector được đọc đúng hay sai, như vậy có thể sử dụng chỉ một bit parity để sửa lỗi: Nếu một trong các sector bị hư, ta biết chính xác đó là sector nào, Với mỗi bit trong sector này ta có thể hình dung nó là bít 1 hay bit 0 bằng cách tính parity của các bit tương ứng từ các sector trên các đĩa khác. Nếu parity của các bit còn lại bằng với parity được lưu, bit mất sẽ là 0, ngoài ra bit mất là 1. RAID mức 3 tốt như mức 2 nhưng it tốn kém hơn (chỉ cần một đĩa overhead).

• **Mức RAID 4 :** Còn được gọi là tổ chức parity chen khối (Block-interleaved parity), lưu trữ các khối đúng như trong các đĩa chính quy, không phân nhỏ chúng qua các đĩa nhưng lấy một khối parity trên một đĩa riêng biệt đối với các khối tương ứng từ N đĩa khác. Nếu một trong các đĩa bị hư, khối parity có thể được dùng với các khối tương ứng từ các đĩa khác để khôi phục khối của đĩa bị hư.

Một đọc khối chỉ truy xuất một đĩa, cho phép các yêu cầu khác được xử lý bởi các đĩa khác. Như vậy, tốc độ truyền dữ liệu đối với mỗi truy xuất chậm, nhưng nhiều truy xuất đọc có thể được xử lý song song, dẫn đến một tốc độ I/O tổng thể cao hơn. Tốc độ truyền đối với các đọc dữ liệu lớn (nhiều khối) cao do tất cả các đĩa có thể được đọc song song; các viết dữ liệu lớn (nhiều khối) cũng có tốc độ truyền cao vì dữ liệu và parity có thể được viết song song. Tuy nhiên, viết một khối đơn phải truy xuất đĩa trên đó khối được lưu trữ, và đĩa parity (do khối parity cũng phải được cập nhật). Như vậy, viết một khối đơn yêu cầu 4 truy xuất: hai để đọc hai khối cũ, và hai để viết lại hai khối.

- **Mức RAID 5 :** Còn gọi là parity phân bố chen khối (Block-interleaved Distributed Parity), cải tiến của mức 4 bởi phân hoạch dữ liệu và parity giữa toàn bộ N+1 đĩa, thay vì lưu trữ dữ liệu trên N đĩa và parity trên một đĩa riêng biệt như trong RAID 4. Trong RAID 5, tất cả các đĩa có thể tham gia làm thoả mãn các yêu cầu đọc, như vậy sẽ làm tăng tổng số yêu cầu có thể được đặt ra trong một đơn vị thời gian. Đối với mỗi khối, một đĩa lưu trữ parity, các đĩa khác lưu trữ dữ liệu. Ví dụ, với một dàn năm đĩa, parity đối với khối thứ n được lưu trên đĩa (n mod 5)+1. Các khối thứ n của 4 đĩa khác lưu trữ dữ liệu hiện hành của khối đó.

- **Mức RAID 6 :** Còn được gọi là sơ đồ dư thừa P+Q (P+Q redundancy scheme), nó rất giống RAID 5 nhưng lưu trữ thông tin dư thừa phụ để canh chừng nhiều đĩa bị hư. Thay vì sử dụng parity, người ta sử dụng các mã sửa lỗi.

CHỌN MỨC RAID ĐÚNG

Nếu đĩa bị hư, Thời gian tái tạo dữ liệu của nó là đáng kể và thay đổi theo mức RAID được dùng. Sự tái tạo dễ dàng nhất đối với mức RAID 1. Đối với các mức khác, ta phải truy xuất tất cả các đĩa khác trong dàn đĩa để tái tạo dữ liệu trên đĩa bị hư. Hiệu năng tái tạo của một một hệ thống RAID có thể là một nhân tố quan trọng nếu việc cung cấp dữ liệu liên tục được yêu cầu (thường xảy ra trong các hệ CSDL hiệu năng cao hoặc trao đổi). Hơn nữa, hiệu năng tái tạo ảnh hưởng đến thời gian trung bình không sự cố.

Vì RAID mức 2 và 4 được gộp lại bởi RAID mức 3 và 5, Việc lựa chọn mức RAID thu hẹp lại trên các mức RAID còn lại. Mức RAID 0 được dùng trong các ứng dụng hiệu năng cao ở đó việc mất dữ liệu không có gì là trầm trọng cả. RAID mức 1 là thông dụng cho các ứng dụng lưu trữ các log-file trong hệ CSDL. Do mức 1 có overhead cao, mức 3 và 5 thường được ưa thích hơn đối với việc lưu trữ khối lượng dữ liệu lớn. Sự khác nhau giữa mức 3 và mức 5 là tốc độ truyền dữ liệu đối lại với tốc độ I/O tổng thể. Mức 3 được ưa thích hơn nếu truyền dữ liệu cao được yêu cầu, mức 5 được ưa thích hơn nếu việc đọc ngẫu nhiên là quan trọng. Mức 6, tuy hiện nay ít được áp dụng, nhưng nó có độ tin cậy cao hơn mức 5.

MỞ RỘNG

Các quan niệm của RAID được khái quát hoá cho các thiết bị lưu trữ khác, bao hàm các dàn băng, thậm chí đối với quảng bá dữ liệu trên các hệ thống không dây. Khi áp dụng RAID cho dàn băng, cấu trúc RAID cho khả năng khôi phục dữ liệu cả khi một trong các băng bị hư hại. Khi áp dụng đối với quảng bá dữ liệu, một khối dữ liệu được phân thành các đơn vị nhỏ và được quảng bá cùng với một đơn vị parity; nếu một trong các đơn vị này không nhận được, nó có thể được dựng lại từ các đơn vị còn lại.

LƯU TRỮ TAM CẤP (tertiary storage)

ĐĨA QUANG HỌC

CR-ROM có ưu điểm là có khả năng lưu trữ lớn, dễ di chuyển (có thể đưa vào và lấy ra khỏi ổ đĩa như đĩa mềm), hơn nữa giá lại rẻ. Tuy nhiên, so với ổ đĩa cứng, thời gian tìm kiếm của ổ CD-ROM chậm hơn nhiều (khoảng 250ms), tốc độ quay chậm hơn (khoảng 400rpm), từ đó dẫn đến độ trễ cao hơn; tốc độ truyền dữ liệu cũng chậm hơn (khoảng 150Kbytes/s). Gần đây, một định dạng mới của đĩa quang học - Digital video disk (DVD) - được chuẩn hoá, các đĩa này có dung lượng trong khoảng 4,7GBytes đến 17 GBytes. Các đĩa WORM, REWRITABLE cũng trở thành phổ biến. Các WORM jukeboxes là các thiết bị có thể lưu trữ một số lớn các đĩa WORM và có thể nạp tự động các đĩa theo yêu cầu đến một hoặc một vài ổ WORM.

BĂNG TỪ

Băng từ có thể lưu một lượng lớn dữ liệu, tuy nhiên, chậm hơn so với đĩa từ và đĩa quang học. Truy xuất băng buộc phải là truy xuất tuần tự, như vậy nó không thích hợp cho hầu hết các đòi hỏi của lưu trữ thứ cấp. Băng từ được sử dụng chính cho việc backup, cho lưu trữ các thông tin không được sử dụng thường xuyên và như một phương tiện ngoại vi (off-line medium) để truyền thông tin từ một hệ thống đến một hệ thống khác. Thời gian để định vị đoạn băng lưu dữ liệu cần thiết có thể kéo dài đến hàng phút. Jukeboxes băng chứa một lượng lớn băng, với một vài ổ băng và có thể lưu trữ được nhiều TeraBytes (10^{12} Bytes)

TRUY XUẤT LƯU TRỮ

Một cơ sở dữ liệu được ánh xạ vào một số các file khác nhau được duy trì bởi hệ điều hành nền. Các file này lưu trú thường trực trên các đĩa với backup trên băng. Mỗi file được phân hoạch thành các đơn vị lưu trữ độ dài cố định được gọi là khối - đơn vị cho cả cấp phát lưu trữ và truyền dữ liệu.

Một khối có thể chứa một vài hạng mục dữ liệu (data item). Ta giả thiết không một hạng mục dữ liệu nào trải ra trên hai khối. Mục tiêu nổi trội của hệ CSDL là tối thiểu hoá số khối truyền giữa đĩa và bộ nhớ. Một cách để giảm số truy xuất đĩa là giữ nhiều khối như có thể trong bộ nhớ chính. Mục đích là để khi một khối được truy xuất, nó đã nằm sẵn trong bộ nhớ chính và như vậy không cần một truy xuất đĩa nào cả.

Do không thể lưu tất cả các khối trong bộ nhớ chính, ta cần quản trị cấp phát không gian sẵn có trong bộ nhớ chính để lưu trữ các khối. Bộ đệm (Buffer) là một phần của bộ nhớ chính sãn có để lưu trữ bản sao khối đĩa. Luôn có một bản sao trên đĩa cho mỗi khối, song các bản sao trên đĩa của các khối là các phiên bản cũ hơn so với phiên bản trong buffer. Hệ thống con đảm trách cấp phát không gian buffer được gọi là bộ quản trị buffer.

BỘ QUẢN TRỊ BUFFER

Các chương trình trong một hệ CSDL đưa ra các yêu cầu cho bộ quản trị buffer khi chúng cần một khối đĩa. Nếu khối này đã sẵn sàng trong buffer, địa chỉ khối trong bộ nhớ chính được chuyển cho người yêu cầu. Nếu khối chưa có trong buffer, bộ quản trị buffer đầu tiên cấp phát không gian trong buffer cho khối, rút ra một số khối khác, nếu cần thiết, để lấy không gian cho khối mới. Khối được rút ra chỉ được viết lại trên đĩa khi nó có bị sửa đổi kể từ lần được viết lên đĩa gần nhất. Sau đó bộ quản trị buffer đọc khối từ đĩa vào buffer, và chuyển địa chỉ của khối trong bộ nhớ chính cho người yêu cầu. Bộ quản trị buffer không khác gì nhiều so với bộ quản trị bộ nhớ ảo, một điểm khác biệt là kích cỡ của một CSDL có thể rất lớn không đủ chứa toàn bộ trong bộ nhớ chính do vậy bộ quản trị buffer phải sử dụng các kỹ thuật tinh vi hơn các sơ đồ quản trị bộ nhớ ảo kiểu mẫu.

- **Chiến lược thay thế.** Khi không có chỗ trong buffer, một khối phải được xoá khỏi buffer trước khi một khối mới được đọc vào. Thông thường, hệ điều hành sử dụng sơ đồ LRU (Least Recently Used) để viết lên đĩa khối ít được dùng gần đây nhất, xoá bỏ nó khỏi buffer. Cách tiếp cận này có thể được cải tiến đối với ứng dụng CSDL.

- **Khối chốt (pinned blocks).** Để hệ CSDL có thể khôi phục sau sự cố, cần thiết phải hạn chế thời gian khi viết lại lên đĩa một khối. Một khối không cho phép viết lại lên đĩa được gọi là khối chốt.

- **Xuất ra bắt buộc các khối (Forced output of blocks).** Có những tình huống trong đó cần phải viết lại một khối lên đĩa, cho dù không gian buffer mà nó chiếm là không cần đến. Việc

viết này được gọi là *sự xuất ra bắt buộc của một khối*. Lý do ngắn gọn của yêu cầu xuất ra bắt buộc khối là nội dung của bộ nhớ chính bị mất khi có sự cố, ngược lại dữ liệu trên dĩa còn tồn tại sau sự cố.

CÁC ĐỐI SÁCH THAY THẾ BUFFER (Buffer-Replacement Policies).

Mục đích của chiến lược thay thế khối trong buffer là tối thiểu hoá các truy xuất đĩa. Các hệ điều hành thường sử dụng chiến lược LRU để thay thế khối. Tuy nhiên, một hệ CSDL có thể dự đoán mẫu tham khảo tương lai. Yêu cầu của một người sử dụng đối với hệ CSDL bao gồm một số bước. Hệ CSDL có thể xác định trước những khối nào sẽ là cần thiết bằng cách xem xét mỗi một trong các bước được yêu cầu để thực hiện hoạt động được yêu cầu bởi người sử dụng. Như vậy, khác với hệ điều hành, hệ CSDL có thể có thông tin liên quan đến tương lai, chí ít là tương lai gần. Trong nhiều trường hợp, chiến lược thay thế khối tối ưu cho hệ CSDL lại là MRU (Most Recently Used): Khối bị thay thế sẽ là khối mới được dùng gần đây nhất!

Bộ quản trị buffer có thể sử dụng thông tin thống kê liên quan đến xác suất mà một yêu cầu sẽ tham khảo một quan hệ riêng biệt nào đó. Tự điển dữ liệu là một trong những phần được truy xuất thường xuyên nhất của CSDL. Như vậy, bộ quản trị buffer sẽ không nên xoá các khối tự điển dữ liệu khỏi bộ nhớ chính trừ phi các nhân tố khác bức chế làm điều đó. Một chỉ mục (Index) đối với một file được truy xuất thường xuyên hơn chính bản thân file, vậy thì bộ quản trị buffer cũng không nên xoá khối chỉ mục khỏi bộ nhớ chính nếu có sự lựa chọn.

Chiến luợc thay thế khối CSDL lý tưởng cần hiểu biết về các hoạt động CSDL đang được thực hiện. Không một chiến lược đơn lẻ nào được biết nắm bắt được toàn bộ các viễn cảnh có thể. Tuy vậy, một điều đáng ngạc nhiên là phần lớn các hệ CSDL sử dụng LRU bất chấp các khuyết điểm của chiến lược đó.

Chiến lược được sử dụng bởi bộ quản trị buffer để thay thế khối bị ảnh hưởng bởi các nhân tố khác hơn là nhân tố thời gian tại đó khối được tham khảo trở lại. Nếu hệ thống đang xử lý các yêu cầu của một vài người sử dụng cạnh tranh, hệ thống (con) điều khiển cạnh tranh (concurrency-control subsystem) có thể phải làm trễ một số yêu cầu để đảm bảo tính nhất quán của CSDL. Nếu bộ quản trị buffer được cho các thông tin từ hệ thống điều khiển cạnh tranh mà nó nêu rõ những yêu cầu nào đang bị làm trễ, nó có thể sử dụng các thông tin này để thay đổi chiến lược thay thế khối của nó. Đặc biệt, các khối cần thiết bởi các yêu cầu tích cực (active requests) có thể được giữ lại trong buffer, toàn bộ các bất lợi đổ dồn lên các khối cần thiết bởi các yêu cầu bị làm trễ.

Hệ thống (con) khôi phục (crash-recovery subsystem) áp đặt các ràng buộc nghiêm nhặt lên việc thay thế khối. Nếu một khối bị sửa đổi, bộ quản trị buffer không được phép viết lại phiên bản mới của khối trong buffer lên đĩa, vì điều này phá huỷ phiên bản cũ. Thay vào đó, bộ quản trị khối phải tìm kiếm quyền từ hệ thống khôi phục trước khi viết khối. Hệ thống khôi phục có thể đòi hỏi một số khối nhất định khác là xuất bắt buộc (forced output) trước khi cấp quyền cho bộ quản trị buffer để xuất ra khối được yêu cầu.

TỔ CHỨC FILE

Một file được tổ chức logic như một dãy các mẩu tin (record). Các mẩu tin này được ánh xạ lên các khối đĩa. File được cung cấp như một xây dựng cơ sở trong hệ điều hành, như vậy ta sẽ giả thiết sự tồn tại của hệ thống file nền. Ta cần phải xét những phương pháp biểu diễn các mô hình dữ liệu logic trong thuật ngữ file.

Các khối có kích cỡ cố định được xác định bởi tính chất vật lý của đĩa và bởi hệ điều hành, song kích cỡ của mẩu tin lại thay đổi. Trong CSDL quan hệ, các bộ của các quan hệ khác nhau nói chung có kích cỡ khác nhau.

Một tiếp cận để ánh xạ một CSDL đến các file là sử dụng một số file, và lưu trữ các mẩu tin thuộc chỉ một độ dài cố định vào một file đã cho nào đó. Một cách khác là cấu trúc các file sao cho ta có thể điều tiết nhiều độ dài cho các mẩu tin. Các file của các mẩu tin độ dài cố định dễ dàng thực thi hơn file của các mẩu tin độ dài thay đổi.

MẪU TIN ĐỘ DÀI CỐ ĐỊNH (Fixed-Length Records)

Xét một file các mẩu tin account đối với CSDL ngân hàng, mỗi mẩu tin của file này được xác định như sau:

```
type
    depositor = record
        branch_name: char(20);
        account_number: char(10);
        balance:real;
    end
```

Giả sử mỗi một ký tự chiếm 1 byte và mỗi số thực chiếm 8 byte, như vậy mẩu tin account có độ dài 40 bytes. Một cách tiếp cận đơn giản là sử dụng 40 byte đầu tiên cho mẩu tin thứ nhất, 40 byte kế tiếp cho mẩu tin thứ hai, ... Cách tiếp cận đơn giản này nảy sinh những vấn đề sau;

0	Perryridge	A-102	400
1	Round Hill	A-305	350
2	Mianus	A-215	700
3	Downtown	A-101	500
4	Redwood	A-222	700
5	Perryridge	A-201	900
6	Brighton	A-217	750
7	Downtown	A-110	600
8	Perryridge	A-218	700

1. File F chứa các mẩu tin account

0	Perryridge	A-102	400
1	Round Hill	A-305	350
3	Downtown	A-101	500
4	Redwood	A-222	700
5	Perryridge	A-201	900
6	Brighton	A-217	750
7	Downtown	A-110	600
8	Perryridge	A-218	700

2. File F sau khi xóa mẩu tin 2 và di chuyển các mẩu tin sau nó

0	Perryridge	A-102	400
1	Round Hill	A-305	350
8	Perryridge	A-218	700
3	Downtown	A-101	500
4	Redwood	A-222	700
5	Perryridge	A-201	900
6	Brighton	A-217	750
7	Downtown	A-110	600

3. File F sau khi xóa mẩu tin 2 và di chuyển mẩu tin cuối vào vị chí của

header				
0	Perryridge	A-102	400	
1				
2	Mianus	A-215	700	
3	Downtown	A-101	500	
4				
5	Perryridge	A-201	900	
6				
7	Downtown	A-110	600	
8	Perryridge	A-218	700	

1. Khó khăn khi xoá một mẩu tin từ cấu trúc này. Không gian bị chiếm bởi mẩu tin bị xoá phải được lấp đầy với mẩu tin khác của file hoặc ta phải đánh dấu mẩu tin bị xoá.
2. Trừ khi kích cỡ khối là bội của 40, nếu không một số mẩu tin sẽ bắt chéo qua biên khối, có nghĩa là một phần mẩu tin được lưu trong một khối, một phần khác được lưu trong một khối khác. như vậy đòi hỏi phải truy xuất hai khối để đọc/viết một mẩu tin "bắc cầu" đó.

Khi một mẩu tin bị xoá, ta có thể di chuyển mẩu tin kề sau nó vào không gian bị chiếm một cách hình thức bởi mẩu tin bị xoá, rồi mẩu tin kế tiếp vào không gian bị chiếm của mẩu tin vừa được di chuyển, cứ như vậy cho đến khi mỗi mẩu tin đi sau mẩu tin bị xoá được dịch chuyển hướng về đầu. Cách tiếp cận này đòi hỏi phải di chuyển một số lớn các mẩu tin. Một cách tiếp cận khác đơn giản hơn là di chuyển mẩu tin cuối cùng vào không gian bị chiếm bởi mẩu tin bị xoá. Song cách tiếp cận này đòi hỏi phải truy xuất khối bổ xung. Vì hoạt động xen xảy ra thường xuyên hơn hoạt động xoá, ta có thể chấp nhận việc để "ngỏ" không gian bị chiếm bởi mẩu tin bị xoá, và chờ một hoạt động xen đến sau để tái sử dụng không gian đó. Một dấu trên mẩu tin bị xoá là không đủ vì sẽ gây khó khăn cho việc tìm kiếm không gian "tự do" đó khi xen. Như vậy ta cần đưa vào cấu trúc bổ xung. ở đầu của file, ta cấp phát một số byte nhất định làm header của file. Header này sẽ chứa đựng thông tin về file. Header chứa địa chỉ của mẩu tin bị xoá thứ nhất, trong nội dung của mẩu tin này có chứa địa chỉ của mẩu tin bị xoá thứ hai và cứ như vậy. Như vậy, các mẩu tin bị xoá sẽ tạo ra một danh sách liên kết dược gọi là danh sách tự do (free list). Khi xen mẩu tin mới, ta sử dụng con trỏ đầu danh sách được chứa trong header để xác định danh sách, nếu danh sách không rỗng ta xen mẩu tin mới vào vùng được trỏ bởi con trỏ đầu danh sách nếu không ta xen mẩu tin mới vào cuối file.

Xen và xoá đối với file mẩu tin độ dài cố định thực hiện đơn giản vì không gian được giải phóng bởi mẩu tin bị xoá đúng bằng không gian cần thiết để xen một mẩu tin. Đối với file của các mẩu tin độ dài thay đổi vấn đề trở nên phức tạp hơn nhiều.

MẨU TIN ĐỘ DÀI THAY ĐỔI (Variable-Length Records)

Mẩu tin độ dài thay đổi trong CSDL do bởi:

- Việc lưu trữ nhiều kiểu mẩu tin trong một file
- Kiểu mẩu tin cho phép độ dài trường thay đổi
- Kiểu mẩu tin cho phép lặp lại các trường

Có nhiều kỹ thuật để thực hiện mẩu tin độ dài thay đổi. Để minh hoạ ta sẽ xét các biểu diễn khác nhau trên các mẩu tin độ dài thay đổi có định dạng sau:

```
Type   account_list = record
                branch_name: char(20)          ;
                account_info: array[ 1.. ∞ ] of record
                        account_number: char(10);
                        balance: real;
                end;
       end
```

Biểu diễn chuỗi byte (Byte-String Representation)

Một cách đơn giản để thực hiện các mẩu tin độ dài thay đổi là ***gắn một ký hiệu đặc biệt End-of-record (⊥) vào cuối mỗi record***. Khi đó, ta có thể lưu mỗi mẩu tin như một chuỗi byte liên tiếp. Thay vì sử dụng một ký hiệu đặc biệt ở cuối của mỗi mẩu tin, một phiên bản của biểu diễn chuỗi byte ***lưu trữ độ dài mẩu tin ở bắt đầu của mỗi mẩu tin***.

0	Perryridge	A-102	400	A-201	900	A210	700	⊥
1	Round Hill	A-301	350	⊥				
2	Mianus	A-101	800	⊥				
3	Downtown	A-211	500	A-222	600	⊥		
4	Redwood	A-300	650	A-200	1200	A-255	950	⊥
5	Brighton	A-111	750	⊥				

Biểu diễn chuỗi byte của các mẩu tin độ dài thay đổi

Biểu diễn chuỗi byte có các bất lợi sau:

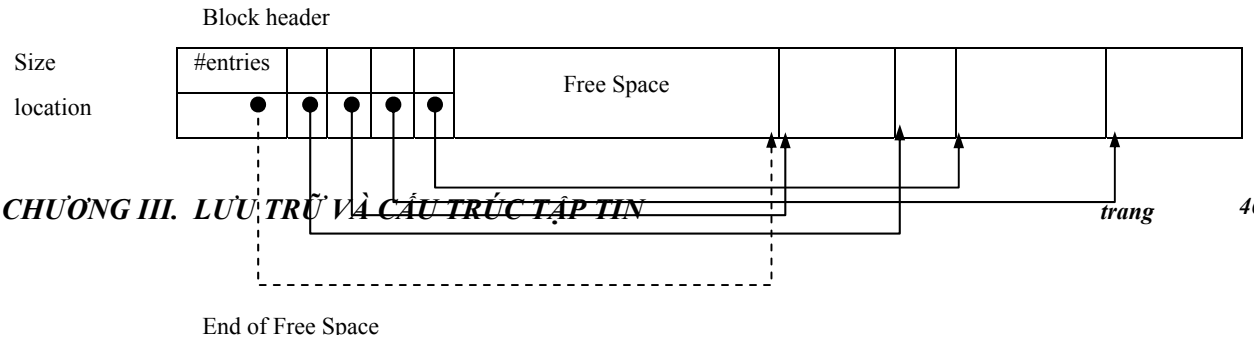
- Khó sử dụng không gian bị chiếm hình thức bởi một mẩu tin bị xoá, điều này dẫn đến một số lớn các mảnh nhỏ của lưu trữ đĩa bị lãng phí.
- Không có không gian cho sự phát triển các mẩu tin. Nếu một mẩu tin độ dài thay đổi dài ra, nó phải được di chuyển và sự di chuyển này là đắt giá nếu mẩu tin bị chốt.

Biểu diễn chuỗi byte không thường được sử dụng để thực hiện mẩu tin độ dài thay đổi, song một dạng sửa đổi của nó được gọi là ***cấu trúc khe-trang*** (slotted-page structure) thường được dùng để tổ chức mẩu tin trong một khối đơn.

Trong cấu trúc slotted-page, có một header ở bắt đầu của mỗi khối, chứa các thông tin sau:

- Số các đầu vào mẩu tin (record entries) trong header
- Điểm cuối không gian tự do (End of Free Space) trong khối
- Một mảng các đầu vào chứa vị trí và kích cỡ của mỗi mẩu tin

Các mẩu tin hiện hành được cấp phát kề nhau trong khối, bắt đàu từ cuối khối, Không gian tự do trong khối là một vùng kề nhau, nằm giữa đầu vào cuối cùng trong mảng header và mẩu tin đàu tiên. Khi một mẩu tin được xen vào, không gian cấp phát cho nó ở cuối của không gian tự do, và đầu vào tương ứng với nó được thêm vào header.



Nếu một mẩu tin bị xoá, không gian bị chiếm bởi nó được giải phóng, đầu vào ứng với nó được đặt là bị xoá (kích cỡ của nó được đặt chẳng hạn là -1). Sau đó, các mẩu tin trong khối trước mẩu tin bị xoá được di chuyển sao cho không gian tự do của khối lại là phần nằm giữa đầu vào cuối cùng của mảng header và mẩu tin đầu tiên. Con trỏ điểm cuối không gian tự do và các con trỏ ứng với mẩu tin bị di chuyển được cập nhật. Sự lớn lên hay nhỏ đi của mẩu tin cũng sử dụng kỹ thuật tương tự (trong trường hợp khối còn không gian cho sự lớn lên của mẩu tin). Cái giá phải trả cho sự di chuyển không quá cao vì các khối có kích cỡ không lớn (thường 4Kbytes).

Biểu diễn độ dài cố định

Một cách khác để thực hiện mẩu tin độ dài thay đổi một cách hiệu quả trong một hệ thống file là sử dụng một hoặc một vài mẩu tin độ dài cố định để biểu diễn một mẩu tin độ dài thay đổi. Hai kỹ thuật thực hiện file của các mẩu tin độ dài thay đổi sử dụng mẩu tin độ dài cố định là:

1. **Không gian dự trữ (reserved space)**. Giả thiết rằng các mẩu tin có độ dài không vượt quá một ngưỡng (độ dài tối đa). Ta có thể sử dụng mảu tin độ dài cố định (có độ dài tối đa), Phần không gian chưa dùng đến được lấp đầy bởi một ký tự đặc biệt: null hoặc End-of-record.
2. **Contrỏ (Pointers)**. Mẩu tin độ dài thay đổi được biểu diễn bởi một danh sách các mẩu tin độ dài cố định, được "móc xích" với nhau bởi các con trỏ.

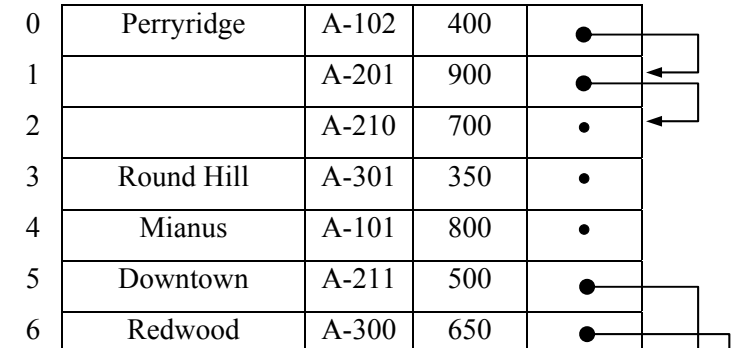
Sự bất lợi của cấu trúc con trỏ là lãng phí không gian trong tất cả các mẩu tin ngoại trừ mẩu tin đầu tiên trong danh sách (mẩu tin đầu tiên cần trường branch_name, các mẩu tin sau trong danh sách không cần thiết có trường này!). Để giải quyết vấn đề này người ta đề nghị phân các khối trong file thành hai loại:

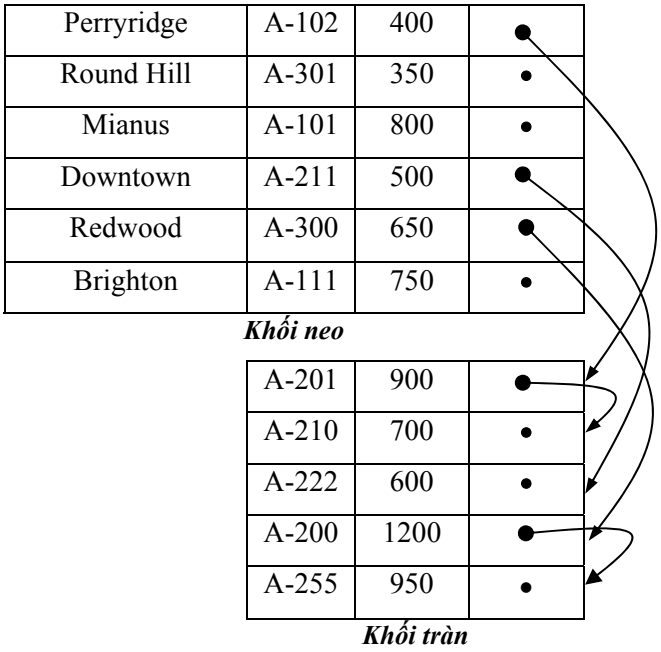
- **Khối neo (Anchor block).** chứa chỉ các mẩu tin đầu tiên trong danh sách
- **Khối tràn (Overflow block).** chứa các mẩu tin còn lại của danh sách

Như vậy, tất cả các mẩu tin trong một khối có cùng độ dài, cho dù file có thể chứa các mẩu tin không cùng độ dài.

0	Perryridge	A-102	400	A-201	900	A210	700	⊥
1	Round Hill	A-301	350	⊥	⊥	⊥	⊥	⊥
2	Mianus	A-101	800	⊥	⊥	⊥	⊥	⊥
3	Downtown	A-211	500	A-222	600	⊥	⊥	⊥
4	Redwood	A-300	650	A-200	1200	A-255	950	⊥
5	Brighton	A-111	750	⊥	⊥	⊥	⊥	⊥

Sử dụng phương pháp không gian dự trữ





Cấu trúc khối neo và khối tràn

TỔ CHỨC CÁC MẪU TIN TRONG FILE

Ta đã xét làm thế nào để biểu diễn các mẩu tin trong một cấu trúc file. Một thể hiện của một quan hệ là một tập hợp các mẩu tin. Đã cho một tập hợp các mẩu tin, vấn đề đặt ra là làm thế nào để tổ chức chúng trong một file. Có một số cách tổ chức sau:

- **Tổ chức file đống (Heap File Organization).** Trong tổ chức này, một mẩu tin bất kỳ có thể được lưu trữ ở bất kỳ nơi nào trong file, ở đó có không gian cho nó. Không có thứ tự nào giữa các mẩu tin. Một file cho một quan hệ.

- **Tổ chức file tuần tự (Sequential File Organization).** Trong tổ chức này, các mẩu tin được lưu trữ *thứ tự tuần tự*, dựa trên ***giá trị của khoá tìm kiếm*** của mỗi mẩu tin.

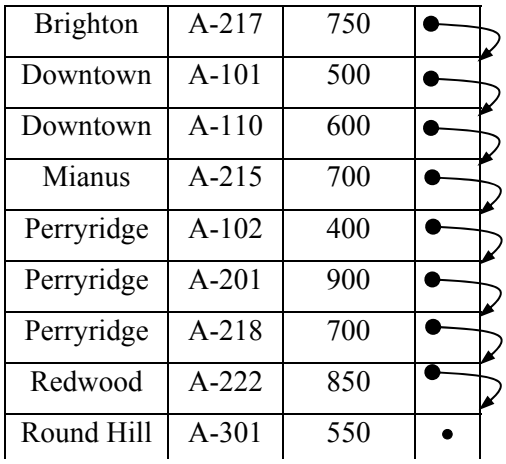
- **Tổ chức file băm (Hashed File Organization).** Trong tổ chức này, có một hàm băm được tính toán trên thuộc tính nào đó của mẩu tin. Kết quả của hàm băm xác định mẩu tin được bố trí trong khối nào trong file. Tổ chức này liên hệ chặt chẽ với cấu trúc chỉ mục.

• **Tổ chức file cụm (Clustering File Organization).** Trong tổ chức này, các mẩu tin của một vài quan hệ khác nhau có thể được lưu trữ trong cùng một file. Các mẩu tin có liên hệ của các quan hệ khác nhau được lưu trữ trên cùng một khối sao cho một hoạt động I/O đem lại các mẩu tin có liên hệ từ tất cả các quan hệ.

TỔ CHỨC FILE TUẦN TỰ

Tổ chức file tuần tự được thiết kế để xử lý hiệu quả các mẩu tin trong thứ tự được sắp dựa trên một khoá tìm kiếm (*search key*) nào đó. Để cho phép tìm lại nhanh chóng các mẩu tin theo thứ tự khoá tìm kiếm, ta "xích" các mẩu tin lại bởi các con trỏ. Con trỏ trong mỗi mẩu tin trỏ tới mẩu tin kế theo thứ tự khoá tìm kiếm. Hơn nữa, để tối ưu hoá số khối truy xuất trong xử lý file tuần tự, ta lưu trữ vật lý các mẩu tin theo thứ tự khoá tìm kiếm hoặc gần với khoá tìm kiếm như có thể.

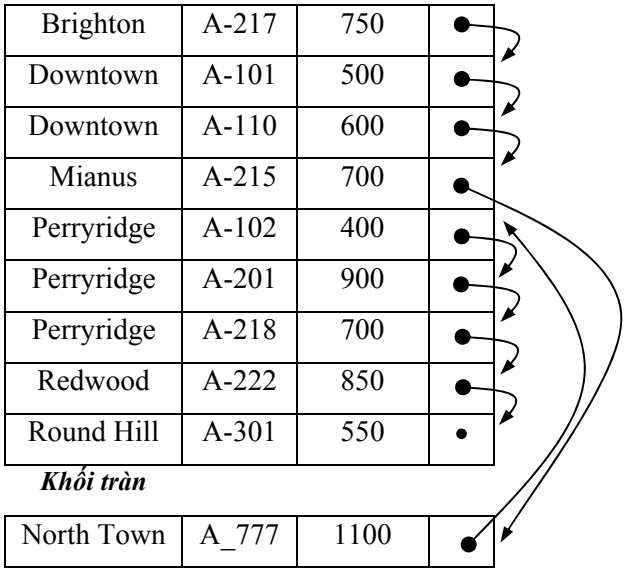
Tổ chức file tuần tự cho phép đọc các mẩu tin theo thứ tự được sắp mà nó có thể hưu dụng cho mục đích trình bày cũng như cho các thuật toán xử lý vấn tin (*query-processing algorithms*).



Brighton	A-217	750	
Downtown	A-101	500	
Downtown	A-110	600	
Mianus	A-215	700	
Perryridge	A-102	400	
Perryridge	A-201	900	
Perryridge	A-218	700	
Redwood	A-222	850	
Round Hill	A-301	550	

Khó khăn gặp phải của tổ chức này là việc duy trì thứ tự tuần tự vật lý của các mẩu tin khi xẩy ra các hoạt động xen, xoá, do cái giá phải trả cho việc di chuyển các mẩu tin khi xen, xoá. Ta có thể quản trị vấn đề xoá bởi dùng dây chuyền các con trỏ như đã trình bày trước đây. Đối với xen, ta có thể áp dụng các quy tắc sau:

1. Định vị mẩu tin trong file mà nó đi trước mẩu tin được xen theo thứ tự khoá tìm kiếm.
2. Nếu có mẩu tin tự do (không gian của mẩu tin bị xoá) trong cùng khối, xen mẩu tin vào khối này. Nếu không, xen mẩu tin mới vào một khối tràn. Trong cả hai trường hợp, điều chỉnh các con trỏ sao cho nó móc xích các mẩu tin theo thứ tự của khoá tìm kiếm.



Brighton	A-217	750	
Downtown	A-101	500	
Downtown	A-110	600	
Mianus	A-215	700	
Perryridge	A-102	400	
Perryridge	A-201	900	
Perryridge	A-218	700	
Redwood	A-222	850	
Round Hill	A-301	550	

Khối tràn

North Town	A_777	1100	

TỔ CHỨC FILE CỤM

Nhiều hệ CSDL quan hệ, mỗi quan hệ được lưu trữ trong một file sao cho có thể lợi dụng được toàn bộ những cái mà hệ thống file của điều hành cung cấp. Thông thường, các bộ của một quan hệ được biểu diễn như các mẩu tin độ dài cố định. Như vậy các quan hệ có thể ánh xạ vào một cấu trúc file. Sự thực hiện đơn giản đó của một hệ CSDL quan hệ rất phù hợp với các hệ CSDL được thiết kế cho các máy tính cá nhân. Trong các hệ thống đó, kích cỡ của CSDL nhỏ. Hơn nữa, trong một số máy tính cá nhân, chủ yếu kích cỡ tổng thể mã đối tượng đối với hệ CSDL là nhỏ. Một cấu trúc file đơn giản làm suy giảm lượng mã cần thiết để thực thi hệ thống.

Cách tiếp cận đơn giản này, để thực hiện CSDL quan hệ, không còn phù hợp khi kích cỡ của CSDL tăng lên. Ta sẽ thấy những điểm lợi về mặt hiệu năng từ việc gán một cách thận trọng các mẩu tin với các khối, và từ việc tổ chức kỹ lưỡng chính bản thân các khối. Như vậy, có vẻ như là một cấu trúc file phức tạp hơn lại có lợi hơn, ngay cả trong trường hợp ta giữ nguyên chiến lược lưu trữ mỗi quan hệ trong một file riêng biệt.

Tuy nhiên, nhiều hệ CSDL quy mô lớn không nhờ cậy trực tiếp vào hệ điều hành nền để quản trị file. Thay vào đó, một file hệ điều hành được cấp phát cho hệ CSDL. Tất cả các quan hệ được lưu trữ trong một file này, và sự quản trị file này thuộc về hệ CSDL. Để thấy những điểm lợi của việc lưu trữ nhiều quan hệ trong cùng một file, ta xét vấn tin SQL sau:

SELECT *account_number, customer_number, customer_treet, customer_city*

FROM *depositor, customer*

WHERE *depositor.customer_name = customer.customername*;

Câu vấn tin này tính một *phép nối* của các quan hệ *depositor* và *customer*. Như vậy, đối với mỗi bộ của *depositor*, hệ thống phải tìm bộ của *customer* có cùng giá trị *customer_name*. Một cách lý tưởng là việc tìm kiếm các mẩu tin này nhờ sự trợ giúp của chỉ mục. Bỏ qua việc tìm kiếm các mẩu tin như thế nào, ta chú ý vào việc truyền từ đĩa vào bộ nhớ. Trong trường hợp xấu nhất, mỗi mẩu tin ở trong một khối khác nhau, điều này buộc ta phải đọc một khối cho một mẩu tin được yêu cầu bởi câu vấn tin. Ta sẽ trình bày một cấu trúc file được thiết kế để thực hiện hiệu quả các câu vấn tin liên quan đến *depositor* ⋈ *customer*. Các bộ *depositor* đối với mỗi *customer_name* được lưu trữ gần bộ *customer* có cùng *customer_name*. Cấu trúc này trộn các bộ của hai quan hệ với nhau, nhưng cho phép xử lý hiệu quả phép nối. Khi một bộ của của quan hệ *customer* được đọc, toàn bộ khối chứa bộ này được đọc từ đĩa vào trong bộ nhớ chính. Do các bộ tương ứng của *depositor* được lưu trữ trên đĩa gần bộ *customer*, khối chứa bộ *customer* chứa các bộ của quan hệ *depositor* cần cho xử lý câu vấn tin. Nếu một *customer* có nhiều *account* đến nỗi các mẩu tin *depositor* không lấp đầy trong một khối, các mẩu tin còn lại xuất hiện trong khối kế cận. Cấu trúc file này, được gọi là ***gom cụm (clustering)***, cho phép ta đọc nhiều mẩu tin được yêu cầu chỉ sử dụng một đọc khối, như vậy ta có thể xử lý câu vấn tin đặc biệt này hiệu quả hơn.

customer_name	customer_street	customer_city
Hays	Main	Brooklyn
Turner	Putnam	Stamford

Quan hệ customer

Hays	Main	Brooklyn
Hays	A-102	
Hays	A-220	
Hays	A-503	
Turner	Putnam	Stamford
Turner	A-305	

Cấu trúc file cụm

Tuy nhiên, cấu trúc gom cụm trên lại tỏ ra không có lợi bằng tổ chức lưu mỗi quan hệ trong một file riêng, đối với một số câu vấn tin, chẳng hạn:

SELECT *

FROM *customer*

Việc xác định khi nào thì gom cụm thường phụ thuộc vào kiểu câu vấn tin mà người thiết kế CSDL nghĩ rằng nó xẩy ra thường xuyên nhất. Sử dụng thận trọng gom cụm có thể cải thiện hiệu năng đáng kể trong việc xử lý câu vấn tin.

LƯU TRỮ TỰ ĐIỂN DỮ LIỆU

Một hệ CSDL cần thiết duy trì *dữ liệu về các quan hệ,* như sơ đồ của các quan hệ. Thông tin này được gọi là ***tự điển dữ liệu (data dictionary)*** hay ***mục lục hệ thống (system catalog).*** Trong các kiểu thông tin mà hệ thống phải lưu trữ là:

- Các tên của các quan hệ
- Các tên của các thuộc tính của mỗi quan hệ
- Các miền (giá trị) và các độ dài của các thuộc tính
- Các tên của các View được định nghĩa trên CSDL và định nghĩa của các view này
- Các ràng buộc toàn vẹn

Nhiều hệ thống còn lưu trữ các thông tin liên quan đến người sử dụng hệ thống:

- Tên của người sử dụng được phép
- Giải trình thông tin về người sử dụng

Các dữ liệu thống kê và mô tả về các quan hệ có thể cũng được lưu trữ:

- Số bộ trong mỗi quan hệ
- Phương pháp lưu trữ được sử dụng cho mỗi quan hệ (cụm hay không)

Các thông tin về mỗi chỉ mục trên mỗi quan hệ cũng cần được lưu trữ :

- Tên của chỉ mục
- Tên của quan hệ được chỉ mục
- Các thuộc tính trên nó chỉ mục được định nghĩa

- Kiểu của chỉ mục được tạo

Toàn bộ các thông tin này trong thực tế bao hàm một CSDL nhỏ. Một số hệ CSDL sử dụng những cấu trúc dữ liệu và mã *mục đích đặc biệt* để lưu trữ các thông tin này. Nói chung, lưu trữ dữ liệu về CSDL trong chính CSDL vẫn được ưa chuộng hơn. Bằng cách sử dụng CSDL để lưu trữ dữ liệu hệ thống, ta đơn giản hoá cấu trúc tổng thể của hệ thống và cho phép sử dụng đầy đủ sức mạnh của CSDL trong việc truy xuất nhanh đến dữ liệu hệ thống.

Sự chọn lựa chính xác biểu diễn dữ liệu hệ thống sử dụng các quan hệ như thế nào là do người thiết kế hệ thống quyết định. Như một ví dụ, ta đề nghị sự biểu diễn sau:

System_catalog_schema = (relation_name, number_of_attributes)

Attribute_schema = (attribute_name, relation_name, domain_type, position, length)

User_schema = (user_name, encrypted_password, group)

Index_schema = (index_name, relation_name, index_type, index_attributes)

View_schema = (view_name, definition)

CHỈ MỤC

Ta xét hoạt động tìm sách trong một thư viện. Ví dụ ta muốn tìm một cuốn sách của một tác giả nào đó. Đầu tiên ta tra trong mục lục tác giả, một tấm thẻ trong mục lục này sẽ chỉ cho ta biết có thể tìm thấy cuốn sách đó ở đâu. Các thẻ trong một mục lục được thư viện sắp xếp thứ tự theo vần chữ cái , như vậy giúp ta có thể tìm đến thẻ cần tìm nhanh chóng không cần phải duyệt qua tất cả các thẻ. Chỉ mục của một file trong các công việc hệ thống rất giống với một mục lục trong một thư viện. Tuy nhiên, chỉ mục được làm như mục lục được mô tả như trên, trong thực tế, sẽ quá lớn để được quản lý một cách hiệu quả. Thay vào đó, người ta sử dụng các kỹ thuật chỉ mục tinh tế hơn. Có hai kiểu chỉ mục:

- **Chỉ mục được sắp (Ordered indices)**. được dựa trên một thứ tự sắp xếp theo các giá trị
- **Chỉ mục băm (Hash indices).** được dựa trên các giá trị được phân phối đều qua các bucket. Bucket mà một giá trị được gán với nó được xác định bởi một hàm, được gọi là hàm băm (hash function)

Đối với cả hai kiểu này, ta sẽ nêu ra một vài kỹ thuật, đáng lưu ý là không kỹ thuật nào là tốt nhất. Mỗi kỹ thuật phù hợp với các ứng dụng CSDL riêng biệt. Mỗi kỹ thuật phải được đánh giá trên cơ sở của các nhân tố sau:

- **Kiểu truy xuất**: Các kiểu truy xuất được hỗ trợ hiệu quả. Các kiểu này bao hàm cả tìm kiếm mẩu tin với một giá trị thuộc tính cụ thể hoặc tìm các mẩu tin với giá trị thuộc tính nằm trong một khoảng xác định.
- **Thời gian truy xuất**: Thời gian để tìm kiếm một hạng mục dữ liệu hay một tập các hạng mục.
- **Thời gian xen**: Thời gian để xen một hạng mục dữ liệu mới. giá trị này bao hàm thời gian để tìm vị trí xen thích hợp và thời gian cập nhật cấu trúc chỉ mục.
- **Thời gian xoá**: Thời gian để xoá một hạng mục dữ liệu. giá trị này bao hàm thời gian tìm kiếm hạng mục cần xoá, thời gian cập nhật cấu trúc chỉ mục.
- **Tổng phí tổn không gian**: Không gian phụ bị chiếm bởi một cấu trúc chỉ mục.

Một file thường đi kèm với một vài chỉ mục. Thuộc tính hoặc tập hợp các thuộc tính được dùng để tìm kiếm mẩu tin trong một file được gọi là ***khoá tìm kiếm***. Chú ý rằng định nghĩa này

khác với định nghĩa khoá sơ cấp (primary key), khoá dự tuyển (candidate key), và siêu khoá (superkey). Như vậy, nếu có một vài chỉ mục trên một file, có một vài khoá tìm kiếm tương ứng.

CHỈ MỤC ĐƯỢC SẮP.

Một chỉ mục lưu trữ các giá trị khoá tìm kiếm trong thứ tự được sắp, và kết hợp với mỗi khoá tìm kiếm, các mẩu tin chứa khoá tìm kiếm này. Các mẩu tin trong file được chỉ mục có thể chính nó cũng được sắp. Một file có thể có một vài chỉ mục trên những khoá tìm kiếm khác nhau. Nếu file chứa các mẩu tin được sắp tuần tự, chỉ mục trên khoá tìm kiếm xác định thứ tự này của file được gọi chỉ mục sơ cấp (primary index). Các chỉ mục sơ cấp cũng được gọi là chỉ mục cụm (clustering index). Khoá tìm kiếm của chỉ mục sơ cấp thường là khoá sơ cấp (khoá chính). Các chỉ mục, khoá tìm kiếm của nó xác định một thứ tự khác với thứ tự của file, được gọi là các chỉ mục thứ cấp (secondary indices) hay các chỉ mục không cụm (nonclustering indices).

Brighton	A-217	750	●
Downtown	A-101	500	●
Downtown	A-110	600	●
Mianus	A-215	700	●
Perryridge	A-102	400	●
Perryridge	A-201	900	●
Perryridge	A-218	700	●
Redwood	A-222	850	●
Round Hill	A-301	550	•

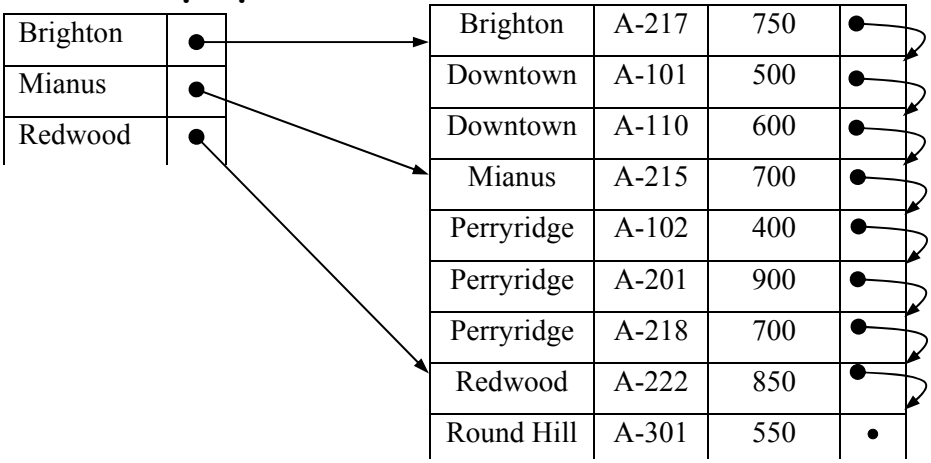
file tuần tự các mẩu tin account

Chỉ mục sơ cấp.

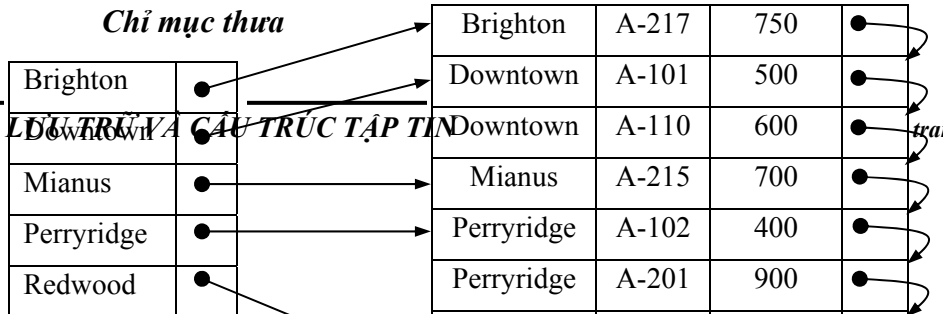
Trong phần này, ta giả thiết rằng tất cả các file được sắp thứ tự tuần tự trên một khoá tìm kiếm nào đó. Các file như vậy, với một *chỉ mục sơ cấp* trên khoá tìm kiếm này, được gọi là file tuần tự chỉ mục (index-sequential files). Chúng biểu diễn một trong các sơ đồ xưa nhất được dùng trong hệ CSDL. Chúng được thiết kế cho các ứng dụng đòi hỏi cả xử lý tuần tự toàn bộ file lẫn truy xuất ngẫu nhiên đến một mẩu tin.

Chỉ mục đặc và chỉ mục thưa (Dense and Sparse Indices)

Chỉ mục đặc



Chỉ mục thưa



Có hai loại chỉ mục được sắp:

- **Chỉ mục đặc**. Mỗi mẩu tin chỉ mục (đầu vào chỉ mục/ index entry) xuất hiện đối với mỗi giá trị khoá tìm kiếm trong file. mẩu tin chỉ mục chứa giá trị khoá tìm kiếm và một con trỏ tới mẩu tin dữ liệu đầu tiên với giá trị khoá tìm kiếm đó.
- **Chỉ mục thưa**. Một mẩu tin chỉ mục được tạo ra chỉ với một số giá trị. Cũng như với chỉ mục đặc, mỗi mẩu tin chỉ mục chứa một giá trị khoá tìm kiếm và một con trỏ tới mẩu tin dữ liệu đầu tiên với giá trị khoá tìm kiếm này. Để định vị một mẩu tin, ta tìm *đầu vào chỉ mục* với giá trị khoá tìm kiếm lớn nhất trong các giá trị khoá tìm kiếm nhỏ hơn hoặc bằng giá trị khoá tìm kiếm đang tìm. Ta bắt đầu từ mẩu tin được trỏ tới bởi đầu vào chỉ mục, và lần theo các con trỏ trong file (dữ liệu) đến tận khi tìm thấy mẩu tin mong muốn.

Ví dụ: Giả sử ta tìm các kiếm mẩu tin đối với chi nhánh Perryridge, sử dụng chỉ mục đặc. Đầu tiên, tìm Perryridge trong chỉ mục (tìm nhị phân!), đi theo con trỏ tương ứng đến mẩu tin dữ liệu (với Branch_name = Perryridge) đầu tiên, xử lý mẩu tin này, sau đó đi theo con trỏ trong mẩu tin này để định vị mẩu tin kế trong thứ tự khoá tìm kiếm, xử lý mẩu tin này, tiếp tục như vậy đến tận khi đạt tới mẩu tin có Branch_name khác với Perryridge.

Đối với chỉ mục thưa, đầu tiên tìm trong chỉ mục, đầu vào có Branch_name lớn nhất trong các đầu vào có Branch_name nhỏ hơn hoặc bằng Perryridge, ta tìm được đầu vào với Mianus, lần theo con trỏ tương ứng đến mẩu tin dữ liệu, đi theo con trỏ trong mẩu tin Mianus để định vị mẩu tin kế trong thứ tự khoá tìm kiếm và cứ như vậy đến tận khi đạt tới mẩu tin dữ liệu Perryridge đầu tiên, sau đó xử lý bắt đầu từ điểm này.

Chỉ mục đặc cho phép tìm kiếm mẩu tin nhanh hơn chỉ mục thưa, song chỉ mục thưa lại đòi hỏi ít không gian hơn chỉ mục đặc. Hơn nữa, chỉ mục thưa yêu cầu một tổn phí duy trì nhỏ hơn đối với các hoạt động xen, xoá.

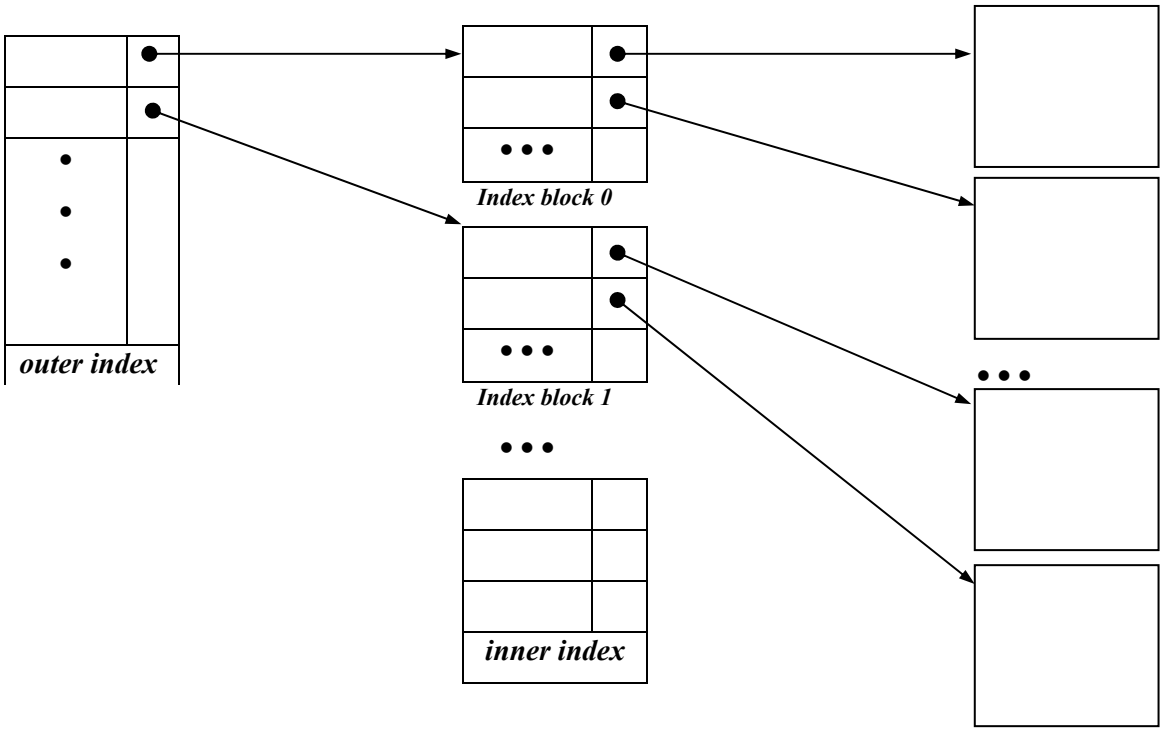
Người thiết kế hệ thống phải cân nhắc sự cân đối giữa thời truy xuất và tổn phí không gian. Một thoả hiệp tốt là có một chỉ mục thưa với một đầu vào chỉ mục cho mỗi khối, vì như vậy cái giá nổi trội trong xử lý một yêu cầu CSDL là thời gian mang một khối từ đĩa vào bộ nhớ chính. Mỗi khi một khối được mang vào, thời gian quét toàn bộ khối là không đáng kể. Sử dụng chỉ mục thưa, ta tìm khối chứa mẩu tin cần tìm. Như vậy, trừ phi mẩu tin nằm trên khối tràn, ta tối thiểu hoá được truy xuất khối, trong khi giữ được kích cỡ của chỉ mục nhỏ như có thể.

Chỉ mục nhiều mức

Chỉ mục có thể rất lớn, ngay cả khi sử dụng chỉ mục thưa, và không thể chứa đủ trong bộ nhớ một lần. Tìm kiếm đầu vào chỉ mục đối với các chỉ mục như vậy đòi hỏi phải đọc vài khối đĩa. Tìm kiếm nhị phân có thể được sử dụng để tìm một đầu vào trên file chỉ mục, song vẫn phải truy xuất khoảng $\lceil \log B \rceil$ khối, với B là số khối đĩa chứa chỉ mục. Nếu B lớn, thời gian truy xuất

này là đáng kể! Hơn nữa nếu sử dụng các khối tràn, tìm kiếm nhị phân không sử dụng được và như vậy việc tìm kiếm phải làm tuần tự. Nó đòi hỏi truy xuất lên đến B khối!!

Để giải quyết vấn đề này, Ta xem file chỉ mục như một file tuần tự và xây dựng chỉ mục thưa cho nó. Để tìm đầu vào chỉ mục, ta tìm kiếm nhị phân trên chỉ mục "ngoài" để được mẩu tin có khoá tìm kiếm lớn nhất trong các mẩu tin có khoá tìm kiếm nhỏ hơn hoặc bằng khoá muốn tìm. Con trỏ tương ứng trỏ tới khối của chỉ mục "trong". Trong khối này, tìm kiếm mẩu tin có khoá tìm kiếm lớn nhất trong các mẩu tin có khoá tìm kiếm nhỏ hơn hoặc bằng khoá muốn tìm, trường con trỏ của mẩu tin này trỏ đến khối chứa mẩu tin cần tìm. Vì chỉ mục ngoài nhỏ, có thể nằm sẵn trong bộ nhớ chính, nên một lần tìm kiếm chỉ cần một truy xuất khối chỉ mục. Ta có thể lặp lại quá trình xây dựng trên nhiều lần khi cần thiết. Chỉ mục với không ít hơn hai mức được gọi là chỉ mục nhiều mức. Với chỉ mục nhiều mức, việc tìm kiếm mẩu tin đòi hỏi truy xuất khối ít hơn đáng kể so với tìm kiếm nhị phân.



Cập nhật chỉ mục

Mỗi khi xen hoặc xoá một mẩu tin, bắt buộc phải cập nhật các chỉ mục kèm với file chứa mẩu tin này. Dưới đây, ta mô tả các thuật toán cập nhật cho các chỉ mục một mức

- **Xoá.** Để xoá một mẩu tin, đầu tiên phải tìm mẩu tin muốn xoá. Nếu mẩu tin bị xoá là mẩu tin đầu tiên trong dây chuyền các mẩu tin được xác định bởi con trỏ của đầu vào chỉ mục trong quá trình tìm kiếm, có hai trường hợp phải xét: nếu mẩu tin bị xoá là mẩu tin duy nhất trong dây chuyền, ta xoá đầu vào trong chỉ mục tương ứng, nếu không, ta thay thế khoá tìm kiếm trong đầu vào chỉ mục bởi khoá tìm kiếm của mẩu tin kề sau mẩu tin bị xoá trong dây chuyền, con trỏ bởi địa chỉ mẩu tin kề sau đó. Trong trường hợp khác, việc xoá mẩu tin không dẫn đến việc điều chỉnh chỉ mục.
- **Xen.** Trước tiên, tìm kiếm dựa trên khoá tìm kiếm của mẩu tin được xen. Nếu là chỉ mục đặc và giá trị khoá tìm kiếm không xuất hiện trong chỉ mục, xen giá trị khoá này và con trỏ tới mẩu tin vào chỉ mục. Nếu là chỉ mục thưa và lưu đầu vào cho mỗi khối, không cần

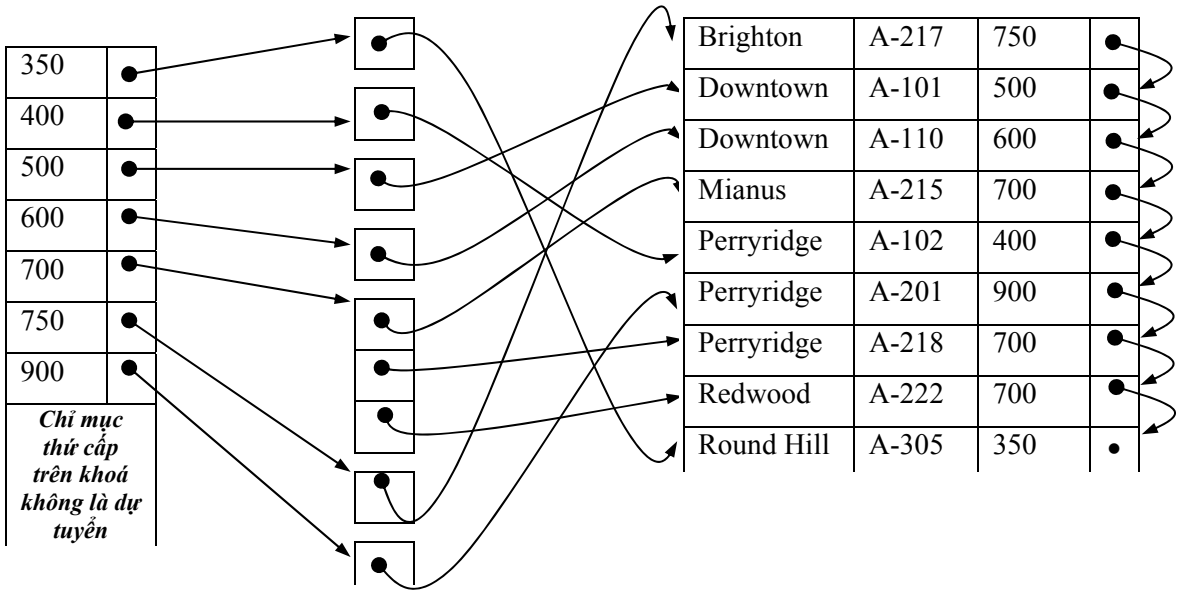
thiết phải thay đổi trừ phi khối mới được tạo ra. Trong trường hợp đó, giá trị khoá tìm kiếm đầu tiên trong khối mới được xen vào chỉ mục.

Giả thuật xen và xoá đối với chỉ mục nhiều mức là một mở rộng đơn giản của các giả thuật vừa được mô tả.

Chỉ mục thứ cấp.

Chỉ mục thứ cấp trên một khoá dự tuyển giống như chỉ mục sơ cấp đặc ngoại trừ các mẩu tin được trỏ đến bởi các giá trị liên tiếp trong chỉ mục không được lưu trữ tuần tự. Nói chung, chỉ mục thứ cấp có thể được cấu trúc khác với chỉ mục sơ cấp. Nếu khoá tìm kiếm của chỉ mục sơ cấp không là khoá dự tuyển, chỉ mục chỉ cần trỏ đến mẩu tin đầu tiên với một giá trị khoá tìm kiếm riêng là đủ (các mẩu tin khác cùng giá trị khoá này có thể tìm lại được nhờ quét tuần tự file).

Nếu khoá tìm kiếm của một chỉ mục thứ cấp không là khoá dự tuyển, việc trỏ tới mẩu tin đầu tiên với giá trị khoá tìm kiếm riêng không đủ, do các mẩu tin trong file không còn được sắp tuần tự theo khoá tìm kiếm của chỉ mục thứ cấp, chúng có thể nằm ở bất kỳ vị trí nào trong file. Bởi vậy, chỉ mục thứ cấp phải chứa tất cả các co trỏ tới mỗi mẩu tin. Ta có thể sử dụng mức phụ gián tiếp để thực hiện chỉ mục thứ cấp trên các khoá tìm kiếm không là khoá dự tuyển. Các con trỏ trong chỉ mục thứ cấp như vậy không trực tiếp trỏ tới mẩu tin mà trỏ tới một bucket chứa các con trỏ tới file.



Chỉ mục thứ cấp phải là đặc, với một đầu vào chỉ mục cho mỗi mẩu tin. Chỉ mục thứ cấp cải thiện hiệu năng các vấn tin sử dụng khoá tìm kiếm không là khóa của chỉ mục sơ cấp, tuy nhiên nó lại đem lại một tổn phí sửa đổi CSDL đáng kể.Việc quyết định các chỉ mục thứ cấp nào là cần thiết dựa trên đánh giá của nhà thiết kế CSDL về tần xuất vấn tin và sửa đổi.

FILE CHỈ MỤC B^+-CÂY (B^+-Tree Index file)

Tổ chức file chỉ mục tuần tự có một nhược điểm chính là làm giảm hiệu năng khi file lớn lên. Để khắc phục nhược điểm đó đòi hỏi phải tổ chức lại file. Cấu trúc chỉ mục B^+-cây là cấu trúc được sử dụng rộng rãi nhất trong các cấu trúc đảm bảo được tính hiệu quả của chúng bất chấp các hoạt động xen, xoá. Chỉ mục B^+-cây là một dạng cây cân bằng (mọi đường dẫn từ gốc đến lá có cùng độ dài). Mỗi nút không là lá có số con nằm trong khoảng giữa $\lceil m/2 \rceil$ và m, trong đó m là một số cố định được gọi là bậc của B^+-cây. Ta thấy rằng cấu trúc B^+-cây cũng đòi hỏi một tổn phí

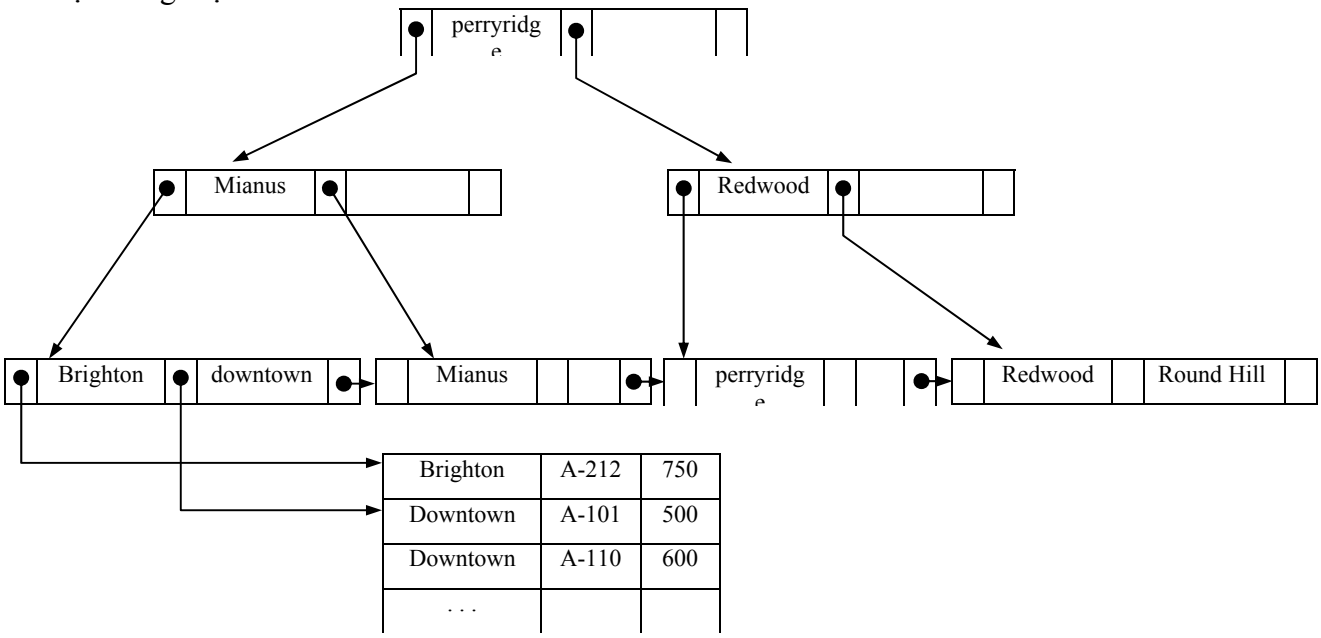
hiệu năng trên xen và xoá cũng như trên không gian. Tuy nhiên, tổn phí này là chấp nhận được ngay cả đối với các file có tần suất sửa đổi cao.

Cấu trúc của B$^+$-cây

Một chỉ mục B$^+$-cây là một chỉ mục nhiều mức, nhưng có cấu trúc khác với file tuần tự chỉ mục nhiều mức (multilevel index-sequential). Một nút tiêu biểu của B$^+$-cây chứa đến n-1 giá trị khoá tìm kiếm. K_1, K_2, ..., K_{n-1}, và n con trỏ P_1, P_2, ..., P_n, các giá trị khoá trong nút được sắp thứ tự: $i < j \Rightarrow K_i < K_j$.

P_1	K_1	P_2	K_2	. . .	P_{n-1}	K_{n-1}	P_n

Trước tiên, ta xét cấu trúc của nút lá. Đối với i = 1, 2, ..., n-1, con trỏ P_i trỏ tới hoặc mẩu tin với giá trị khoá K_i hoặc tới một bucket các con trỏ mà mỗi một trong chúng trỏ tới một mẩu tin với
giá trị khoá K_i. Cấu trúc bucket chỉ được sử dụng trong các trường hợp: hoặc khoá tìm kiếm không là khoá sơ cấp hoặc file không được sắp theo khoá tìm kiếm. Con trỏ P_n được dùng vào mục đích đặc biệt: P_n được dùng để móc xích các nút lá lại theo thứ tự khoá tìm kiếm, điều này cho phép xử lý tuần tự file hiệu quả. Bây giờ ta xem các giá trị khoá tìm kiếm được gắn với một nút lá như thế nào. Mỗi nút nút lá có thể chứa đến n-1 giá trị. Khoảng giá trị mà mỗi nút lá chứa là không chồng chéo. Như vậy, nếu L_i và L_j là hai nút lá với i < j thì mỗi giá trị khoá trong nút L_i nhỏ hơn mọi giá trị khoá trong L_j . Nếu chỉ mục B$^+$-cây là đặc, mỗi giá trị khoá tìm kiếm phải xuất hiện trong một nút lá nào đó.



Các nút không là lá của một B$^+$-cây tạo ra một chỉ mục nhiều mức trên các nút lá. Cấu trúc của các nút không là lá tương tự như cấu trúc nút lá ngoại trừ tất cả các con trỏ đều trỏ đến các nút của cây. Các nút không là lá có thể chứa đến m con trỏ và phải chứa không ít hơn $\lceil m/2 \rceil$ con trỏ ngoại trừ nút gốc. Nút gốc được phép chứa ít nhất 2 con trỏ. Số con trỏ trong một nút được gọi là số nan (fanout) của nút.

Con trỏ P_i của một nút không là lá (chứa p con trỏ, 1 < i < p) trỏ đến một cây con chứa các giá trị khoá tìm kiếm nhỏ hơn K_i và lớn hơn hoặc bằng K_{i-1}. Con trỏ P_1 trỏ đến cây con chứa các giá trị khoá tìm kiếm nhỏ hơn K_1. Con trỏ P_p trỏ tới cây con chứa các khoá tìm kiếm lớn hơn K_{p-1}.

Các vấn tin trên B^+-cây

Ta xét xử lý vấn tin sử dụng B^+-cây như thế nào ? Giả sử ta muốn tìn tất cả các mẩu tin với giá trị khoá tìm kiếm k. Đầu tiên, ta kiểm tra nút gốc, tìm giá trị khoá tìm kiếm nhỏ nhất lớn hơn k, giả sử giá trị khoá đó là K_i. Đi theo con trỏ Pi để di tới một nút khác. Nếu nút có p con trỏ và $k > K_{p-1}$, đi theo con trỏ P_p. Đến một nút tới, lặp lại quá trình tìm kiếm giá trị khoá tìm kiếm nhỏ nhất lớn hơn k và theo con trỏ tương ứng để đi tới một nút khác và tiếp tục như vậy đến khi đạt tới một nút lá. Con trỏ tương ứng trong nút lá hướng ta tới mẩu tin/bucket mong muốn. Số khối truy xuất không vượt quá $\left\lceil \log_{\lceil m/2 \rceil} K \right\rceil$, trong đó K là số giá trị khoá tìm kiếm trong B^+-cây, m là bậc của cây.

Cập nhật trên B^+-cây

- **Xen**. Sử dụng cùng kỹ thuật như tìm kiếm, ta tìm nút lá trong đó giá trị khoá tìm kiếm cần xen sẽ xuất hiện. Nếu khoá tìm kiếm đã xuất hiện rồi trong nút lá, xen mẩu tin vào trong file, thêm con trỏ tới mẩu tin vào trong bucket tương ứng. Nếu khoá tìm kiếm chưa hiện diện trong nút lá, ta xen mẩu tin vào trong file rồi xen giá trị khoá tìm kiếm vào trong nút lá ở vị trí đúng (bảo tồn tính thứ tự), tạo một bucket mới với con trỏ tương ứng. Nếu nút lá không còn chỗ cho giá trị khoá mới, Một khối mới được yêu cầu từ hệ điều hành, các giá trị khoá trong nút lá được tách một nửa cho nút mới, giá trị khoá mới được xen vào vị trí đúng của nó vào một trong hai khối này. Điều này kéo theo việc xen giá trị khoá đầu khối mới và con trỏ tới khối mới vào nút cha. Việc xen cặp giá trị khoá và con trỏ vào nút cha này lại có thể dẫn đến việc tách nút ra làm hai. Quá trình này có thể dẫn đến tận nút gốc. Trong trường hợp nút gốc bị tách làm hai, một nút gốc mới được tạo ra và hai con của nó là hai nút được tách ra từ nút gốc cũ, chiều cao cây tăng lên một.

Procedure Insert(*value* V, *pointer* P)

Tìm nút lá L sẽ chứa giá trị V

Insert_entry(L, V, P)

end procedure

Procedure Insert_entry(*node* L, *value* V, *pointer* P)

If (L có không gian cho (V, P) **then**

Xen (V, P) vào L

else begin /* tách L */

Tạo nút L'

If (L là nút lá) **then begin**

V' là giá trị sao cho $\lceil m/2 \rceil$ giá trị trong các giá trị $L.K_1$, $L.K_2$, ..., $L.K_{m-1}$, V nhỏ hơn V'

n là chỉ số nhỏ nhất sao cho $L.K_n \geq V'$

Di chuyển $L.P_n$, $L.K_n$, ..., $L.P_{m-1}$, $L.K_{n-1}$ sang L'

If (V < V') then xen (V, P) vào trong L else xen (P, V) vào trong L'

end else begin

V' là giá trị sao cho $\lceil m/2 \rceil$ giá trị trong các giá trị $L.K_1$, $L.K_2$, ..., $L.K_{m-1}$, V lớn hơn hoặc bằng V'

n là chỉ số nhỏ nhất sao cho $L.K_n \geq V'$

Thêm Nil, $L.K_n$, $L.P_{n+1}$, $L.K_{n+1}$, ..., $L.P_{m-1}$, $L.K_{m-1}$, $L.P_m$ vào L'

Xoá $L.K_n$, $L.P_{n+1}$, $L.K_{n+1}$, ..., $L.P_{m-1}$, $L.K_{m-1}$, $L.P_m$ khỏi L

```
                    If (V < V') then xen (P, V) vào trong L  else  xen (P, V) vào trong L'
                    xoá (Nil, V') khỏi L'
                end
                If (L không là nút gốc) then Insert_entry(parent(L), V', L')
                else begin
                    Tạo ra nút mới R với các nút con là L và L' với giá trị duy nhất trong nó là V'
                    Tạo R là gốc của cây
                end
                If (L) là một nút lá then begin
                    đặt L'.Pm = L.Pm
                    đặt L.Pm = L'
                end
        end
end procedure
```

- **Xoá**. Sử dụng kỹ thuật tìm kiếm tìm mẩu tin cần xoá, xoá nó khỏi file, xoá giá trị khoá tìm kiếm khỏi nút lá trong B^+-cây nếu không có bucket kết hợp với giá trị khoá tìm kiếm hoặc bucket trở nên rỗng sau khi xoá con trỏ tương ứng trong nó. Việc xoá một giá trị khoá khỏi một nút của B^+-cây có thể dẫn đến nút lá trở nên rỗng, phải trả lại, từ đó nút cha của nó có thể có số con nhỏ hơn ngưỡng cho phép, trong trường hợp đó hoặc phải chuyển một con từ nút anh em của nút cha đó sang nút cha nếu điều đó có thể (nút anh em của nút cha này còn số con ≥ $\lceil m/2 \rceil$ sau khi chuyển đi một con). Nếu không, phải gom nút cha này với một nút anh em của nó, điều này dẫn tới xoá một nút trong khỏi cây, rồi xoá khỏi nút cha của nó một hạng, ... quá trình này có thể dẫn đến tận gốc. Trong trường hợp nút gốc chỉ còn một con sau xoá, cây phải thay nút gốc cũ bởi nút con của nó, nút gốc cũ phải trả lại cho hệ thống, chiều cao cây giảm đi một.

```
Procedure delete(value V, pointer P)
        Tìm nút lá chứa (V, P)
        delete_entry(L, V, P)
end procedure
Procedure delete_entry(node L, value V, pointer P)
        xoá (V, P) khỏi L
        If (L là nút gốc and L chỉ còn lại một con) then
                Lấy con của L làm nút gốc mới của cây, xoá L
        else If (L có quá ít giá trị/ con trỏ) then begin
                L' là anh em kề trái hoặc phải của L
                V' là giá trị ở giữa hai con trỏ L, L' (trong nút parent(L))
                If (các đầu vào của L và L' có thể lấp đầy trong một khối) then begin
                    If (L là nút trước của L') then wsap_variables(L, L')
                    If (L không là lá) then nối V' và tất cả con trỏ, giá trị trong L với L'
                    else  begin nối tất cả các cặp (K, P) trong L với L'; L'.Pp = L.Pp  end
```

```
            delete_entry(parent(L), V', L); xoá nút L
        end
        else begin
            If (L' là nút trước của L) then begin
                If (L không là nút lá) then begin
                    p là chỉ số sao cho L'.P_p là con trỏ cuối trong L'
                    xoá (L'.K_p-1, L'.P_p) khỏi L'
                    xen (L'.P_p, V') như phần tử đầu tiên trong L (right_shift tất cả các phần tử của L)
                    thay thế V' trong parent(L) bởi L'.K_p-1
                end else begin
                    p là chỉ số sao cho L'.P_p là con trỏ cuối trong L'
                    xoá (L'.P_p, L'.K_p) khỏi L'
                    xen (L'.P_p, L'.K_p) như phần tử đầu tiên trong L (right_shift tất cả các phần tử của L)
                    thay thế V' trong parent(L) bởi L'.K_p
                end
        end < đối xứng với trường hợp then >
    end
end procedure
```

Tổ chức file B^+-cây

Trong tổ chức file B^+-cây, các nút lá của cây lưu trữ các mẩu tin, thay cho các con trỏ tới file. Vì mẩu tin thường lớn hơn con trỏ, số tối đa các mẩu tin được lưu trữ trong một khối lá ít hơn số con trỏ trong một nút không lá. Các nút lá vẫn được yêu cầu được lấp đầy ít nhất là một nửa.

Xen và xoá trong tổ chức file B^+-cây tương tự như trong chỉ mục B^+-cây.

Khi B^+-cây được sử dụng để tổ chức file, việc sử dụng không gian là đặc biệt quan trọng, vì không gian bị chiếm bởi mẩu tin là lớn hơn nhiều so với không gian bị chiếm bởi (khoá,con trỏ). Ta có thể cải tiến sự sử sụng không gian trong B^+-cây bằng cách bao hàm nhiều nút anh em hơn khi tái phân phối trong khi tách và trộn. Khi xen, nếu một nút là đầy, ta thử phân phối lại một số đầu vào đến một trong các nút kề để tạo không gian cho đầu vào mới. Nếu việc thử này thất bại, ta mới thực hiện tách nút và phân chia các đầu vào giữa một trong các nút kề và hai nút nhận được do tách nút. Khi xoá, nếu nút chứa ít hơn $\lfloor 2m/3 \rfloor$ đầu vào, ta thử mượn một đầu vào từ một trong hai nút anh em kề. Nếu cả hai đều có đúng $\lfloor 2m/3 \rfloor$ mẩu tin, ta phân phối lại các đầu vào của nút cho hai nút anh em kề và xoá nút thứ 3. Nếu k nút được sử dụng trong tái phân phối (k-1 nút anh em), mỗi nút đảm bảo chứa ít nhất $\lfloor (k-1)m/k \rfloor$ đầu vào. Tuy nhiên, cái giá phải trả cho cập nhật của cách tiếp cận này sẽ cao hơn.

FILE CHỈ MỤC B-CÂY (B-Tree Index Files)

Chỉ mục B-cây tương tự như chỉ mục B^+-cây. Sự khác biệt là ở chỗ B-cây loại bỏ lưu trữ dư thừa các giá trị khoá tìm kiếm. Trong B-cây, các giá trị khoá chỉ xuất hiện một lần. Do các khoá tìm kiếm xuất hiện trong các nút không là lá không xuất hiện ở bất kỳ nơi nào khác nữa trong B-cây, ta phải thêm một trường con trỏ cho mỗi khoá tìm kiếm trong các nút không là lá. Con trỏ thêm vào này trỏ tới hoặc mẩu tin trong file hoặc bucket tương ứng.

Một nút lá B-cây tổng quát có dạng:

P_1	K_1	P_2	K_2	...	P_{m-1}	K_{m-1}	P_m

Một nút không là lá có dạng:

P_1	B_1	K_1	P_2	B_2	K_2		$P_{...}$	$B_{m-...}$	$K_{...}$	P

Các con trỏ P_i là các con trỏ cây và được dùng như trong B^+-cây. Các con trỏ B_i trong các nút không là lá là các con trỏ mẩu tin hoặc con trỏ bucket. Rõ ràng là số giá trị khoá trong nút không lá nhỏ hơn số giá trị trong nút lá. Số nút được truy xuất trong quá trình tìm kiếm trong một B-cây phụ thuộc nơi khoá tìm kiếm được định vị.

Xoá trong một B-cây phức tạp hơn trong một B^+-cây. Xoá một đầu vào xuất hiện ở một nút không là lá kéo theo việc tuyển chọn một giá trị thích hợp trong cây con của nút chứa đầu vào bị xoá. Nếu khoá K_i bị xoá, khoá nhỏ nhất trong cây con được trỏ bởi P_{i+1} phải được di chuyển vào vị trí của K_i. Nếu nút lá còn lại quá ít đầu vào, cần thiết các hoạt động bổ xung.

III.9.4 Định nghĩa chỉ mục trong SQL

Một chỉ mục được tạo ra bởi lệnh **CREATE INDEX** với cú pháp

CREATE INDEX < index-name > **ON** < relation_name > (< attribute-list >)

attribute-list là danh sách các thuộc tính của quan hệ được dùng làm khoá tìm kiếm cho chỉ mục. Nếu muốn khai báo là khoá tìm kiếm là khoá dự tuyển, thêm vào từ khoá UNIQUE:

CREATE UNIQUE INDEX < index-name > **ON** < relation_name > (< attribute-list >)

attribute-list phải tạo thành một khoá dự tuyển, nếu không sẽ có một thông báo lỗi.

Bỏ đi một chỉ mục sử dụng lệnh DROP:

DROP INDEX < index-name >

BĂM (HASHING)

BĂM TĨNH (Static Hashing)

Bất lợi của tổ chức file tuần tự là ta phải truy xuất một cấu trúc chỉ mục để định vị dữ liệu, hoặc phải sử dụng tìm kiếm nhị phân, và kết quả là có nhiều hoạt động I/O. Tổ chức file dựa trên kỹ thuật băm cho phép ta tránh được truy xuất một cấu trúc chỉ mục. Băm cung cung cấp một phương pháp để xây dựng các chỉ mục.

Tổ chức file băm

Trong tổ chức file băm, ta nhận được địa chỉ của khối đĩa chứa một mẩu tin mong muốn bởi tính toán một hàm trên giá trị khoá tìm kiếm của mẩu tin. thuật ngữ bucket được dùng để chỉ một đơn vị lưu trữ. Một bucket kiểu mẫu là một khối đĩa, nhưng có thể được chọn nhỏ hơn hoặc lớn hơn một khối đĩa.

K ký hiệu tập tất cả các giá trị khoá tìm kiếm, B ký hiệu tập tất cả các địa chỉ bucket. Một hàm băm h là một hàm từ K vào B : h: $K \rightarrow \mathrm{B}$

Xen một mẩu tin với giá trị khoá K vào trong file: ta tính h(K). Giá trị của h(K) là địa chỉ của bucket sẽ chứa mẩu tin. Nếu có không gian trong bucket cho mẩu tin, mẩu tin được lưu trữ trong bucket.

Tìm kiếm một mẩu tin theo giá trị khoá K: đầu tiên tính h(K), ta tìm được bucket tương ứng. sau đó tìm trong bucket này mẩu tin với giá trị khoá K mong muốn.

Xoá mẩu tin với giá trị khoá K: tính h(K), tìm trong bucket tương ứng mẩu tin mong muốn, xoá nó khỏi bucket.

Hàm băm

Hàm băm xấu nhất là hàm ánh xạ tất cả các giá trị khoá vào cùng một bucket. Hàm băm lý tưởng là hàm phân phối ***đều*** các giá trị khoá vào các bucket, như vậy mỗi bucket chứa một số lượng mẩu tin như nhau. Ta muốn chọn một hàm băm thoả mãn các tiêu chuẩn sau:

- ***Phân phối đều***: Mỗi bucket được gán cùng một số giá trị khoá tìm kiếm trong tập hợp tất cả các giá trị khoá có thể
- ***Phân phối ngẫu nhiên***: Trong trường hợp trung bình, các bucket được gán một số lượng giá trị khoá tìm kiếm *gần bằng nhau*.

Các hàm băm phải được thiết kế thận trọng. Một hàm băm xấu có thể dẫn đến việc tìm kiếm chiếm một thời gian tỷ lệ với số khoá tìm kiếm trong file.

Điều khiển tràn bucket

Khi xen một mẩu tin, nếu bucket tương ứng còn chỗ, mẩu tin được xen vào bucket, nếu không sẽ xảy ra tràn bucket. Tràn bucket do các nguyên do sau:

- **Các bucket không đủ.** Số các bucket n_B phải thoả mãn $n_B > n_r / f_r$ trong đó n_r là tổng số mẩu tin sẽ lưu trữ, f_r là số mẩu tin có thể lấp đầy trong một bucket.
- **Sự lệch.** Một vài bucket được gán cho một số lượng mẩu tin nhiều hơn các bucket khác, như vậy một bucket có thể tràn trong khi các bucket khác vẫn còn không gian. Tình huống này được gọi là sự lệch bucket. Sự lệch xảy ra do hai nguyên nhân:
  1. Nhiều mẩu tin có cùng khoá tìm kiếm
  2. Hàm băm được chọn phân phối các giá trị khoá không đều

Ta quản lý tràn bucket bằng cách dùng các bucket tràn. Nếu một mẩu tin phải được xen vào bucket B nhưng bucket B đã đầy, khi đó một bucket tràn sẽ được cấp cho B và mẩu tin được xen vào bucket tràn này. Nếu bucket tràn cũng đầy một bucket tràn mới lại được cấp và cứ như vậy. Tất cả các bucket tràn của một bucket được "móc xích" với nhau thành một danh sách liên kết. Việc điều khiển tràn dùng danh sách liên kết như vậy được gọi là dây chuyền tràn. Đối với dây chuyền tràn, thuật toán tìm kiếm thay đổi chú ít: trước tiên ta cũng tính giá trị hàm băm trên khoá tìm kiếm, ta được bucket B, kiểm tra các mẩu tin, trong bucket B và tất cả các bucket tràn tương ứng, có giá trị khoá khớp với giá trị tìm không.

Một cách điều khiển tràn bucket khác là: Khi cần xen một mẩu tin vào một bucket nhưng nó đã đầy, thay vì cấp thêm một bucket tràn, ta sử dụng một hàm băm kế trong một dãy các hàm băm được chọn để tìm bucket khác cho mẩu tin, nếu bucket sau cũng đầy, ta lại sử dụng một hàm băm kế và cứ như vậy... Dãy các hàm băm thường được sử dụng là { $h_i(K) = (h_{i-1}(K) + 1) \bmod n_B$ với $1 \le i \le n_B - 1$ và h_0 là hàm băm cơ sở }.

Dạng cấu trúc băm sử dụng dây chuyền bucket được gọi là băm mở. Dạng sử dụng dãy các hàm băm được gọi là băm đóng. Trong các hệ CSDL, cấu trúc băm đóng thường được ưa dùng hơn.

Chỉ mục băm

Một chỉ mục băm tổ chức các khoá tìm kiếm cùng con trỏ kết hợp vào một cấu trúc file băm như sau: áp dụng một hàm băm trên khoá tìm kiếm để định danh bucket sau đó lưu giá trị khoá và con trỏ kết hợp vào bucket này (hoặc vào các bucket tràn). Chỉ mục băm thường là chỉ mục thứ cấp.

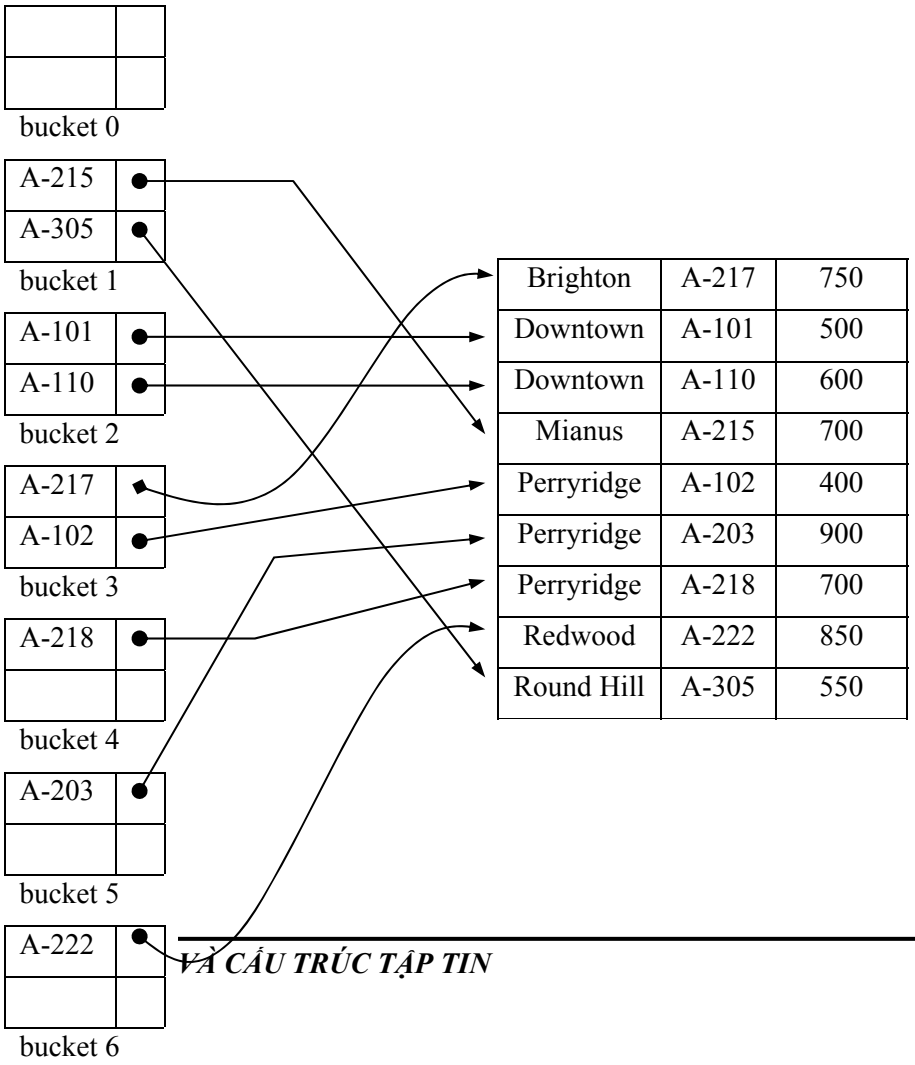
Hàm băm trên số tài khoản được tính theo công thức:

h(Account_number) = (tổng các chữ số trong số tài khoản) mod 7

BĂM ĐỘNG (Dynamic Hashing)

Trong kỹ thuật băm tĩnh (static hashing), tập B các địa chỉ bucket phải là cố định. Các CSDL phát triển lớn lên theo thời gian. Nếu ta sử dụng băm tĩnh cho CSDL, ta có ba lớp lựa chọn:

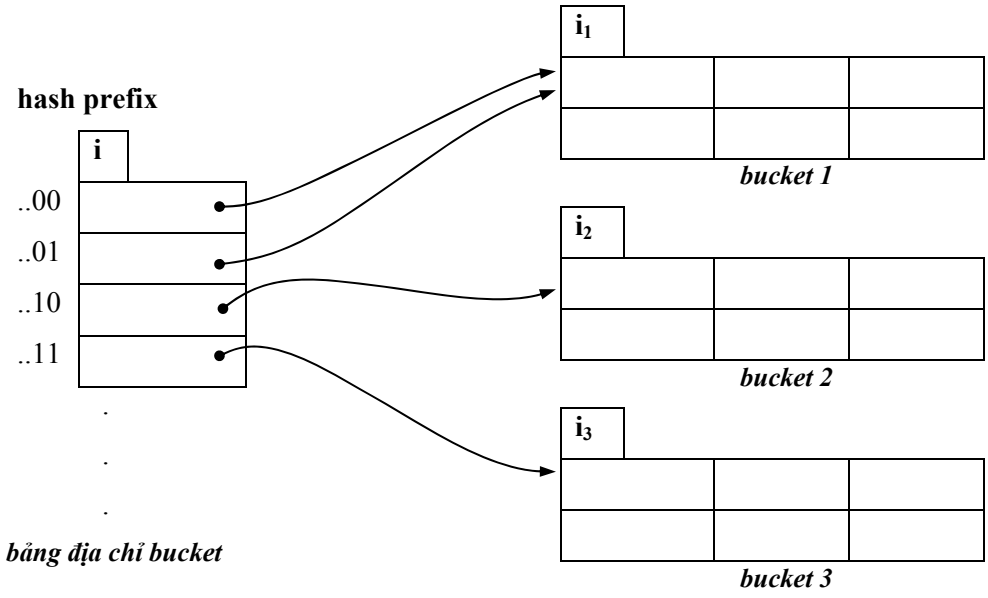
1. Chọn một hàm băm dựa trên kích cỡ file hiện hành. Sự lựa chọn này sẽ dẫn đến sự suy giảm hiệu năng khi CSDL lớn lên.
2. Chọn một hàm băm dựa trên kích cỡ file dự đoán trước cho một thời điểm nào đó trong tương lai. Mặc dù sự suy giảm hiệu năng được cải thiện, một lượng đáng kể không gian có thể bị lãng phí lúc khởi đầu.
3. Tổ chức lại theo chu kỳ cấu trúc băm đáp ứng sự phát triển kích cỡ file. Một sự tổ chức lại như vậy kéo theo việc lựa chọn một hàm băm mới, tính lại hàm băm trên mỗi mẩu tin trong file và sinh ra các gán bucket mới. Tổ chức lại là một hoạt động tốn thời gian. Hơn nữa, nó đòi hỏi cấm truy xuất file trong khi đang tổ chức lại file.



Chỉ mục băm trên khoá tìm kiếm account-number của file account

Kỹ thuật băm động cho phép sửa đổi hàm băm để phù hợp với sự tăng hoặc giảm của CSDL. Một dạng băm động được gọi là băm có thể mở rộng (extendable hashing) được thực hiện như sau: Chọn một hàm băm h với các tính chất đều, ngẫu nhiên và có miền giá trị tương đối rộng, chẳng hạn, là một số nguyên b bit (b thường là 32). Khi khởi đầu ta không sử dụng toàn bộ b bit giá trị băm. Tại một thời điểm, ta chỉ sử dụng i bit $0 \le i \le b$. i bit này được dùng như một độ dời (offset) trong một bảng địa chỉ bucket phụ. giá trị i tăng lên hay giảm xuống tuỳ theo kích cỡ CSDL.

Số i xuất hiện bên trên bảng địa chỉ bucket chỉ ra rằng i bit của giá trị băm h(K) được đòi hỏi để xác định bucket đúng cho K, số này sẽ thay đổi khi kích cỡ file thay đổi. Mặc dù i bit dược đòi hỏi để tìm đầu vào đúng trong bảng địa chỉ bucket, một số đầu vào bảng kề nhau có thể trỏ đến cùng một bucket. Tất cả các như vậy có chung hash prefix chung, nhưng chiều dài của prefix này có thể nhỏ hơn i. Ta kết hợp một số nguyên chỉ độ dài của hash prefix chung này, ta sẽ ký hiệu số nguyên kết hợp với bucket j là i_j. Số các đầu vào bảng địa chỉ bucket trỏ đến bucket j là $2^{(i-i_j)}$.



Cấu trúc băm có thể mở rộng tổng quát

Để định vị bucket chứa giá trị khoá tìm kiếm K , ta lấy i bit cao đầu tiên của h(K), tìm trong đầu vào bảng tương ứng với chuỗi bit này và lần theo con trỏ trong đầu vào bảng này. Để xen một mẩu tin với giá trị khoá tìm kiếm K, tiến hành thủ tục dịnh vị trên, ta được bucket, giả sử là bucket j. Nếu còn cho cho mẩu tin, xen mẩu tin vào trong bucket đó. Nếu không, ta phải tách bucket ra và phân phối lại các mẩu tin hiện có cùng mẩu tin mới. Để tách bucket, đầu tiên ta xác định từ giá trị băm có cần tăng số bit lên hay không.

- Nếu $i = i_j$, chỉ có một đầu vào trong bảng địa chỉ bucket trỏ đến bucket j. ta cần tăng kích cỡ của bảng địa chỉ bucket sao cho ta có thể bao hàm các con trỏ đến hai bucket kết quả

của việc tách bucket j bằng cách xét thêm một bit của giá trị băm. tăng giá trị i lên một, như vậy kích cỡ của bảng địa chỉ bucket tăng lên gấp đôi. Mỗi một đầu vào được thay bởi hai đầu vào, cả hai cùng chứa con trỏ của đầu vào gốc. Bây giờ hai đầu vào trong bảng địa chỉ bucket trỏ tới bucket j. Ta định vị một bucket mới (bucket z), và đặt đầu vào thứ hai trỏ tới bucket mới, đặt i_j và i_z về i, tiếp theo đó mỗi một mẩu tin trong bucket j được băm lại, tuỳ thuộc vào i bit đầu tiên, sẽ hoặc ở lại bucket j hoặc được cấp phát cho bucket mới được tạo.

- Nếu $i > i_j$ khi đó nhiều hơn một đầu vào trong bảng địa chỉ bucket trỏ tới bucket j. như vậy ta có thể tách bucket j mà không cần tăng kích cỡ bảng địa chỉ bucket. Ta cấp phát một bucket mới (bucket z) và đặt i_j và i_z đến giá trị là kết quả của việc thêm 1 vào giá trị i_j gốc. Kế đến, ta điều chỉnh các đầu vào trong bảng địa chỉ bucket trước đây trỏ tới bucket j. Ta để lại nửa đầu các đầu vào, và đặt tất cả các đầu vào còn lại trỏ tới bucket mới tạo (z). Tiếp theo, mỗi mẩu tin trong trong bucket j được băm lại và được cấp phát cho hoặc vào bucket j hoặc bucket z.

Để xoá một mẩu tin với giá trị khoá K, trước tiên ta thực hiện thủ tục định vị, ta tìm được bucket tương ứng, gọi là j, ta xoá cả khoá tìm kiếm trong bucket lẫn mẩu tin mẩu tin trong file. bucket cũng bị xoá, nếu nó trở nên rỗng. Chú ý rằng, tại điểm này, một số bucket có thể được kết hợp lại và kích cỡ của bảng địa chỉ bucket sẽ giảm đi một nửa.

Ưu điểm chính của băm có thể mở rộng là hiệu năng không bị suy giảm khi file tăng kích cỡ, hơn nữa, tổng phí không gian là tối tiểu mặc dù phải thêm vào không gian cho bảng địa chỉ bucket. Một khuyết điểm của băm có thể mở rộng là việc tìm kiếm phải bao hàm một mức gián tiếp: ta phải truy xuất bảng địa chỉ bucket trước khi truy xuất đến bucket. Vì vậy, băm có thể mở rộng là một kỹ thuật rất hấp dẫn.

CHỌN CHỈ MỤC HAY BĂM ?

Ta đã xét qua các sơ đồ: chỉ mục thứ tự, băm. Ta có thể tổ chức file các mẩu tin bởi hoặc sử dụng tổ chức file tuần tự chỉ mục, hoặc sử dụng B^+-cây, hoặc sử dụng băm ... Mỗi sơ đồ có những các ưu điểm trong các tình huống nhất định. Một nhà thực thi hệ CSDL có thể cung cấp nhiều nhiều sơ đồ, để lại việc quyết định sử dụng sơ đồ nào cho nhà thiết kế CSDL. Để có một sự lựa chọn khôn ngoan, nhà thực thi hoặc nhà thiết kế CSDL phải xét các yếu tố sau:

- Cái giá phải trả cho việc tổ chức lại theo định kỳ của chỉ mục hoặc băm có chấp nhận được hay không?
- Tần số tương đối của các hoạt động xen và xoá là bao nhiêu ?
- Có nên tối ưu hoá thời gian truy xuất trung bình trong khi thời gian truy xuất trường hợp xấu nhất tăng lên hay không ?
- Các kiểu vấn tin mà các người sử dụng thích đặt ra là gì ?

CẤU TRÚC LƯU TRỮ CHO CSDL HƯỚNG ĐỐI TƯỢNG

SẮP XẾP CÁC ĐỐI TƯỢNG VÀO FILE

Phần dữ liệu của đối tượng có thể được lưu trữ bởi sử dụng các cấu trúc file được mô tả trước đây với một số thay đổi do đối tượng có kích cỡ không đều, hơn nữa đối tượng có thể rất lớn. Ta có thể thực thi các trường tập hợp ít phần tử bằng cách sử dụng danh sách liên kết, các trường tập hợp nhiều phần tử bởi B-cây hoặc bởi các quan hệ riêng biệt trong cơ sở dữ liệu. Các trường tập hợp cũng có thể bị loại trừ ở mức lưu trữ bởi chuẩn hoá. Các đối tượng cực lớn khó có

thể phân tích thành các thành phần nhỏ hơn có thể được lưu trữ trong một file riêng cho mỗi đối tượng.

THỰC THI ĐỊNH DANH ĐỐI TƯỢNG

Vì đối tượng được nhận biết bởi định danh của đối tượng (OID = objject Identifier), Một hệ lưu trữ đối tượng cần phải có một cơ chế để tìm kiếm một đối tượng được cho bởi một OID. Nếu các OID là logic, có nghĩa là chúng không xác định vị trí của dối tượng, hệ thống lưu trữ phải duy trì một chỉ mục mà nó ánh xạ OID tới vị trí hiện hành của đối tượng. Nếu các OID là vật lý, có nghĩa là chúng mã hoá vị trí của đối tượng, đối tượng có thể dược tìm trực tiếp. Các OID điển hình có ba trường sau:

1. Một volume hoặc định danh file
2. Một định danh trang bên trong volume hoặc file
3. Một offset bên trong trang

Hơn nữa, OID vật lý có thẻ chứa một định danh duy nhất, nó là một số nguyên tách biệt OID với các định danh của các đối tượng khác đã được lưu trữ ở cùng vị trí trước đây và đã bị xoá hoặc dời đi. Định danh duy nhất này cũng được lưu với đối tượng, các định danh trong một OID và đối tượng tương ứng phù hợp. Nếu định danh duy nhất trong một OID vật lý không khớp với với định danh duy nhất trong đối tượng mà OID này trỏ tới, hệ thống phát hiện ra rằng con trỏ là bám và báo một lỗi. Lỗi con trỏ như vậy xảy ra khi OID vật lý tương ứng với đối tượng cũ đã bị xoá do tai nạn. Nếu không gian bị chiếm bởi đối tượng được cấp phát lại, có thể có một đối tượng mới ở vào vị trí này và có thể được định địa chỉ không đúng bởi định danh của đối tượng cũ. Nếu không phát hiện được, sử dụng con trỏ bám có thể gây nên sự sai lạc của một đối tượng mới được lưu ở cùng vị trí. Định danh duy nhất trợ giúp phát hiện lỗi như vậy. Giả sử một đối tượng phải di chuyển sang trang mới do sự lớn lên của đối tượng và trang cũ không có không gian phụ. Khi đó OID vật lý trỏ tới trang cũ bây giờ không còn chứa đối tượng. Thay vì thay đổi OID của đối tượng (điều này kéo theo sự thay đổi mỗi đối tượng trỏ tới đối tượng này) ta để địa chỉ forward ở vị trí cũ. Khi CSDL tìm đối tượng, nó tìm địa chỉ forward thay cho tìm đối tượng và sử dụng địa chỉ forward để tìm đối tượng.

QUẢN TRỊ CÁC CON TRỎ BỀN (persistent pointers)

Ta thực thi các con trỏ bền trong ngôn ngữ lập trình bền (persistent programming language) bằng cách sử dụng các OID. Các con trỏ bền có thể là các OID vật lý hoặc logic. Sự khác nhau quan trọng giữa con trỏ bền và con trỏ trong bộ nhớ là kích thước concủa con trỏ. Con trỏ trong bộ nhớ chỉ cần đủ lớn để định địa chỉ toàn bộ bộ nhớ ảo, hiện tại kích cỡ con trỏ trong bộ nhớ là 4 byte. Con trỏ bền để định địa chỉ toàn bộ dữ liệu trong một CSDL, nên kích cỡ của nó ít nhất là 8 byte.

Pointer Swizzling

Hành động tìm một đối tượng được cho bởi định danh được gọi là *dereferencing*. Đã cho một con trỏ trong bộ nhớ, tìm đối tượng đơn thuần là một sự tham khảo bộ nhớ. Đã cho một con trỏ bền, dereferencing một đối tượng có một bước phụ: phải tìm vị trí hiện hành của đối tượng trong bộ nhớ bởi tìm con trỏ bền trong một bảng. Nếu đối tượng chưa nằm trong bộ nhớ, nó phải được nạp từ đĩa. Ta có thể thực thi bảng tìm kiếm này hoàn toàn hiệu quả bởi sử dụng băm, song tìm kiếm vẫn chậm.

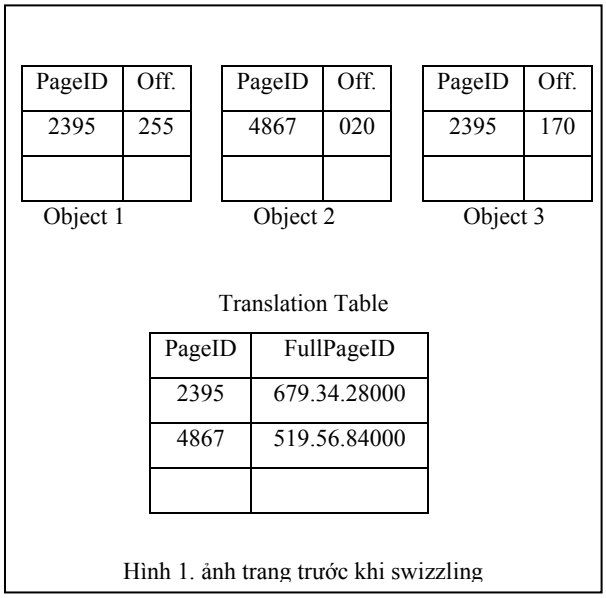
pointer swizling là một phương pháp để giảm cái giá tìm kiếm các đối tượng bền đã hiện diện trong bộ nhớ. ý tưởng là khi một con trỏ bền được dereference, đối tượng được định vị và mang vào trong bộ (nhớ nếu nó chưa có ở đó). Bây giờ một bước phụ được thực hiện: một con trỏ trong bộ nhớ tới đối tượng được lưu vào vị trí của con trỏ bền. Lần kế con trỏ bền tương tự được dereference, vị trí trong bộ nhớ có thể được đọc ra trực tiếp. Trong trường hợp các đối tượng bền phải di chuyển lên đĩa để lấy không gian cho đối tượng bền khác, cần một bước phụ để đảm bảo đối tượng vẫn trong bộ nhớ cũng phải được thực hiện. Khi một đối tượng được viết ra. bất kỳ con trỏ bền nào mà nó chứa và bị swizzling phải được unswizzling như vậy được chuyển đổi về biểu diễn bền của chúng. pointer swizzling

trên poiter dereferenc được mô tả này được gọi là software swizzling. Quan trị buffer sẽ phức tạp hơn nếu pointer swizzling được sử dụng.

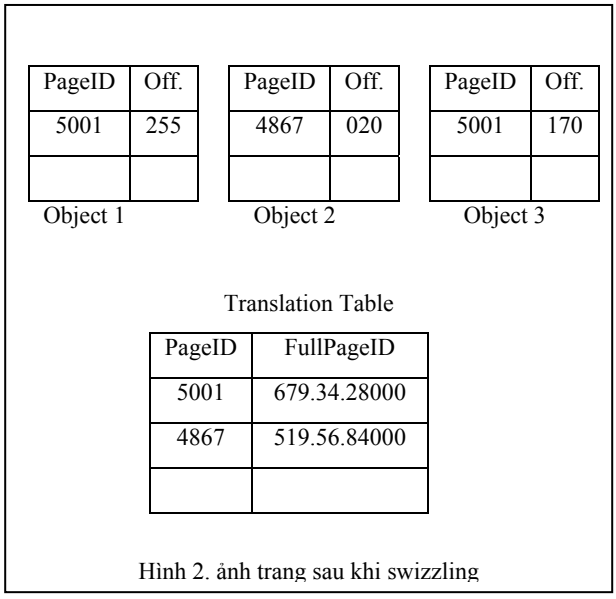
Hardware swizzling

Việc có hai kiểu con trỏ, con trỏ bền (persistent pointer) và con trỏ tạm (transient pointer / con trỏ trong bộ nhớ), là điều khá bất lợi. Người lập trình phải nhớ kiểu con trỏ và có thể phải viết mã chương trình hai lần- một cho các con trỏ bền và một cho con trỏ tạm. Sẽ thuận tiên hơn nếu cả hai kiểu con trỏ này cùng kiểu. Một cách đơn giản để trộn lãn hai con rỏ này là mở rộng chiều dài con trỏ bộ nhớ cho bằng kích cỡ con trỏ bền và sử dụng một bit của phần định danh để phân biệt chúng. Cách làm này sẽ làm tăng chi phí lưu trữ đối với các con trỏ tạm. Ta sẽ mô tả một kỹ thuật được gọi là hardware swizzling nó sử dụng phần cứng quản trị bộ nhớ để giải quyết vấn đề này. Hardware swizzling có hai điểm lợi hơn so với software swizzling: Thứ nhất, nó cho phép lưu trữ các con trỏ bền trong đối tượng trong lượng không gian bằng với lượng không gian con trỏ bộ nhớ đòi hỏi. Thứ hai, nó chuyển đổi trong suốt giữa các con trỏ bền và các con trỏ tạm một cách thông minh và hiệu quả. Phần mềm được viết để giải quyết các con trỏ trong bộ nhớ có thể giải quyết các con trỏ bền mà không cần thay đổi.

hardware swizzling sử dụng sự biểu diễn các con trỏ bền được chứa trong đối tượng trên đĩa như sau: Một con trỏ bền được tách ra thành hai phần, một là định danh trang và một là offset bên trong trang. Định danh trang thường là một con trỏ trực tiếp nhỏ: mỗi trưng có một bảng dịch (translation table) cung cấp một ánh xạ từ các định danh trang ngắn đến các định danh CSDL đầy đủ. Hệ thống phải tìm định danh trang nhỏ trong một con trỏ bền trong bảng dịch để tìm định danh trang đầy đủ. Bảng dịch, trong trường hợp xấu nhất, chỉ lớn bằng số tối đa các con trỏ có thể được chứa trong các đối tượng trong một trang. Với một trang kích thước 4096 byte, con trỏ kích thước 4 byte, số tối đa các con trỏ là 1024. Trong thực tế số tối đa nhỏ hơn con số này rất nhiều. Định danh trang nhỏ chỉ cần đủ số bit để định danh một dòng trong bảng, nếu số dòng tối đa là 1024, chỉ cần 10 bit để định danh trang nhỏ. Bảng dịch cho phép toàn bộ một con trỏ bền lấp đầy một không gian bằng không gian cho một con trỏ trong bộ nhớ.



Hình 1. ảnh trang trước khi swizzling



Hình 2. ảnh trang sau khi swizzling

Trong hình 1, trình bày sơ đồ biểu diễn con trỏ bền, có ba đối tượng trong trang, mỗi một chứa một con trỏ bền. Bảng dịch cho ra ánh xạ giữa định danh trang ngắn và định danh trang CSDL đầy đủ đối với mỗi định danh trang ngắn trong các con trỏ bền này. Định danh trang CSDL được trình bày dưới dạng ***volume.file.offset***. Thông tin phụ được duy trì với mỗi trang sao cho tất cảc các con trỏ bền trong trang có thể tìm thấy. Thông tin được cập nhất khi một đối tượng được tạo ra hay bị xoá khỏi trang. Khi một con trỏ trong bộ nhớ được dereferencing, nếu hệ điều hành phát hiện trang trong không gian địa chỉ ảo được trỏ tới không được cấp phát lưu trữ hoặc có truy xuất được bảo vệ, khi đó một sự vi phạm đoạn được ước đoán là xảy ra. Nhiều hệ điều hành cung cấp một cơ chế xác định một hàm sure được gọi khi vi phạm đoạn xảy ra, một cơ chế cấp phát lưu trữ cho các trang trong không gian địa chỉ ảo, và một tập các quyền truy xuất trang. Đầu tiên, ta xét một con trỏ trong bộ nhớ trỏ tới trang v được khử tham chiếu, khi lưu trữ chưa được cấp phát cho trang này. Một vi phạm đoạn sẽ xảy ra và kết quả là một lời gọi hàm trên hệ CSDL. Hệ CSDL dầu tiên xác định trang CSDL nào đã được cấp phát cho trang bộ nhớ ảo v, gọi định danh trang đầy đủ của trang CSDL là P, nếu không có trang CSDL cấp phát cho v, một lỗi được thông báo., nếu không, hệ CSDL cấp phát không gian lưu trữ cho trang v và nạp trang CSDL P vào trong v. Pointer swizzling bây giờ được làm đối với trang P như sau: Hệ thông tìm tất cả các con trỏ bèn được chứa trong các đối tương trong trang, bằng cách sử dụng thông tin phụ được lưu trữ trong trang. Ta xét một con trỏ như vậy và gọi nó là (p_i, o_i), trong đó p_i là định danh trang ngắn và

o_i là offset trong trang. Giả sử P_i là định danh trang đầy đủ của p_i được tìm thấy trong bảng dịch trong trang P. Nếu trang P_i chưa có một trang bộ nhớ ảo được cấp cho nó, một trang tự do trong không gian địa chỉ ảo sẽ được cấp cho nó. Trang P_i sẽ nằm ở vị trí địa chỉ ảo nàynếu và khi nó được mang vào. Tại điểm này, trang trong không gian địa chỉ ảo không có bất kỳ một lưu trữ nào được cấp cho nó, cả trong bộ nhớ lẫn trên đĩa, đơn tuần chỉ là một khoảng địa chỉ dự trữ cho trang CSDL. Bây giờ giả sử trang bộ nhớ ảo đã được cấp phát cho P_i là v_i . Ta cập nhật con trỏ (p_i, o_i) bởi thay thế p_i bởi v_i , cuối cùng sau khi swizzling tất cả các con trỏ bền trong P, sự khử tham chiếu gây ra vi phạm đoạn được cho phép tiếp tục và sẽ tìm thấy đối tượng đang được tìm kiếm trong bộ nhớ.

Trong hình 2, trình bày trạng thái trang trong hình 1 sau khi trang này được mang vào trong bộ nhớ và các con trỏ trong nó đã được swizzling. ở đây ta giả thiết trang định danh trang CSDL của nó là 679.34.28000 được ánh xạ đến trang 5001 trong bộ nhớ, trong khi trang định danh của nó là 519.56.84000 được ánh xạ đến trang 4867. Tất cả các con trỏ trong đối tượng đã được cập nhật để phản ánh tương ứng mới và bây giờ có thể được dùng như con trỏ trong bộ nhớ. ở cuối của giai đoạn dịch đối với một trang, các đối tượng trong trang thoả mãn một tính chất quan trọng: Tất cả các con trỏ bền được chứa trong đối tượng trong trang được chuyển đổi thành các con trỏ trong bộ nhớ.

CÂU HỎI VÀ BÀI TẬP CHƯƠNG III

III.1 Xét sự sắp xếp các khối dữ liệu và các khối parity trên bốn đĩa sau:

Đĩa 1	Đĩa 2	Đĩa 3	Đĩa 4
B_1	B_2	B_3	B_4
P_1	B_5	B_6	B_7
B_8	P_2	B_9	B_{10}
⋮	⋮	⋮	⋮

Trong đó các B_i biểu diễn các khối dữ liệu, các khối P_i biểu diễn các khối parity. Khối P_i là khối parity đối với các khối dữ liệu B_{4i-3} , B_{4i-2} , B_{4i-1} , B_{4i} . Hãy nêu các vấn đề gặp phải của cách sắp xếp này.

III.2 Một sự mất điện xảy ra trong khi một khối đang được viết sẽ dẫn tới kết quả là khối đó có thể chỉ được viết một phần. Giả sử rằng khối được viết một phần có thể phát hiện được. Một viết khối nguyên tử là hoặc toàn bộ khối được viết hoặc không có gì được viết (không có khối được viết một phần). Hãy đề nghị những sơ đồ để có được các viết khối nguyên tử hiệu quả trên các sơ đồ RAID:

1. Mức 1 (mirroring)
2. Mức 5 (block interleaved, distributed parity)

III.3 Các hệ thống RAID tiêu biểu cho phép thay thế các đĩa hư không cần ngưng truy xuất hệ thống. Như vậy dữ liệu trong đĩa bị hư sẽ phải được tái tạo và viết lên đĩa thay thế trong khi hệ thống vẫn tiếp tục hoạt động. Với mức RAID nào thời lượng giao thoa giữa việc tái tạo và các truy xuất đĩa còn đang chạy là ít nhất ? Giải thích.

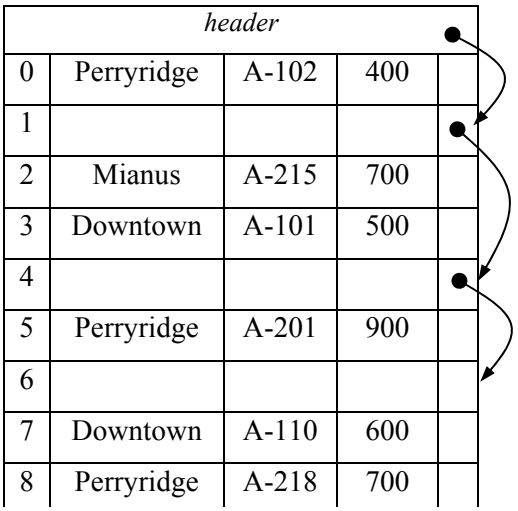
III.4 Xét việc xoá mẩu tin 5 trong file:

0	**Perryridge**	A-102	400
1	Round Hill	A-305	350
8	Perryridge	A-218	700
3	Downtown	A-101	500
4	Redwood	A-222	700
5	Perryridge	A-201	900
6	Brighton	A-217	750
7	Downtown	A-110	600

So sánh các điều hay/dở tương đối của các kỹ thuật xoá sau:

1. Di chuyển mẩu tin 6 đến không gian chị chiếm bởi mẩu tin 5, rồi di chuyển mẩu tin 7 đến chỗ bị chiếm bởi mẩu tin 6.
2. Di chuyển mẩu tin 7 đến chỗ bị chiếm bởi mẩu tin 5
3. Đánh dấu xoá mẩu tin 5.

III.5 Vẽ cấu trúc của file:



Sau mỗi bước sau:

1. Insert(Brighton, A-323, 1600)
2. Xoá mẩu tin 2
3. Insert(Brighton, A-636, 2500)

III.6 Vẽ lại cấu trúc file:

0	Perryridge	A-102	400	A-201	900	A210	700	⊥
1	Round Hill	A-301	350	⊥				
2	Mianus	A-101	800	⊥				
3	Downtown	A-211	500	A-222	600	⊥		
4	Redwood	A-300	650	A-200	1200	A-255	950	⊥
5	Brighton	A-111	750	⊥				

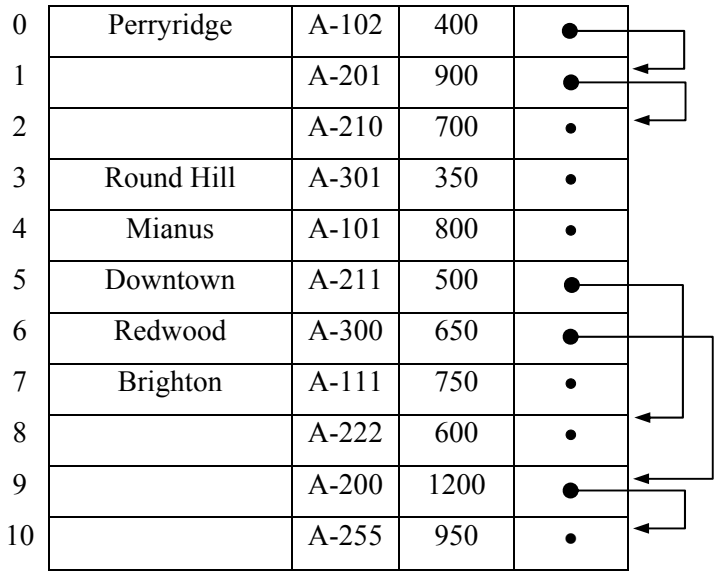
Sau mỗi bước sau:

1. Insert(Mianus, A-101, 2800)
2. Insert(Brighton, A-323, 1600)
3. Delete (Perryridge, A-102, 400)

III.7 Điều gì sẽ xảy ra nếu xen mẩu tin (Perryridge, A-999, 5000) vào file trong **III.6.**

III.8 Vẽ lại cấu trúc file dưới đây sau mỗi bước sau:

1. Insert(Mianus, A-101, 2800
2. Insert(Brighton, A-323, 1600)
3. Delete (Perryridge, A-102, 400)



(• = con trỏ nil)

III.9 Nêu lên một ví dụ, trong đó phương pháp không gian dự trữ để biểu diễn các mẩu tin độ dài thay đổi phù hợp hơn phương pháp con trỏ.

III.10 Nêu lên một ví dụ, trong đó phương pháp con trỏ để biểu diễn các mẩu tin độ dài thay đổi phù hợp hơn phương pháp không gian dự trữ.

III.11 Nếu một khối trở nên rỗng sau khi xoá. Khối này được tái sử dụng vào mục đích gì ?

III.12 Trong tổ chức file tuần tự, tại sao khối tràn được sử dụng thậm chí, tại thời điểm đang xét, chỉ có một mẩu tin tràn ?

III.13 Liệt kê các ưu điểm và nhược điểm của mỗi một trong các chiến lược lưu trữ CSDL quan hệ sau:

1. Lưu trữ mỗi quan hệ trong một file
2. Lưu trữ nhiều quan hệ trong một file

III.14 Nêu một ví dụ biểu thức đại số quan hệ và một chiến lược xử lý vấn tin trong đó:

1. MRU phù hợp hơn LRU
2. LRU phù hợp hơn MRU

III.15 Khi nào sử dụng chỉ mục đặc phù hợp hơn chỉ mục thưa ? Giải thích.

III.16 Nêu các điểm khác nhau giữa chỉ mục sơ cấp và chỉ mục thứ cấp .

III.17 Có thể có hai chỉ mục sơ cấp đối với hai khoá khác nhau trên cùng một quan hệ ? Giải thích.

III.18 Xây dựng một B^+-cây đối với tập các giá trị khoá: (2, 3, 5, 7, 11, 15, 19, 25, 29, 33, 37, 41, 47). Giả thiết ban đầu cây là rỗng và các giá trị được xen theo thứ tự tăng. Xét trong các trường hợp sau:

1. Mỗi nút chứa tối đa 4 con trỏ
2. Mỗi nút chứa tối đa 6 con trỏ
3. Mỗi nút chứa tối đa 8 con trỏ

III.19 Đối với mỗi B^+-cây trong bài tập **III.18** Bày tỏ các bước thực hiện trong các vấn tin sau:

1. Tìm mẩu tin với giá trị khoá tìm kiếm 11
2. Tìm các mẩu tin với giá trị khoá nằm trong khoảng [7..19]

III.20 Đối với mỗi B^+-cây trong bài tập **III.18.** Vẽ cây sau mỗi một trong dãy hoạt động sau:

1. Insert 9
2. Insert 11
3. Insert 11
4. Delete 25
5. Delete 19

III.21 Cùng câu hỏi như trong **III.18** nhưng đối với B-cây

III.22 Nêu và giải thích sự khác nhau giữa băm đóng và băm mở. Nêu các ưu, nhược điểm của mỗi kỹ thuật này.

III.23 Điều gì gây ra sự tràn bucket trong một tổ chức file băm ? Làm gì để giảm sự tràn này ?

III.24 Giả sử ta đang sử dụng băm có thể mở rộng trên một trên một file chứa các mẩu tin với các giá trị khoá tìm kiếm sau:

2, 3, 5, 7, 11, 17, 19, 23, 37, 31, 35, 41, 49, 55

Vẽ cấu trúc băm có thể mở rộng đối với file này nếu hàm băm là **h(x) = x mod 8** và mỗi bucket có thể chứa nhiều nhất được ba mẩu tin.

III.25 Vẽ lại cấu trúc băm có thể mở rộng trong bài tập **III.24** sau mỗi bước sau:

1. Xoá 11
2. Xoá 55
3. Xen 1

4. Xen 15

CHƯƠNG IV

GIAO DỊCH
(Transaction)

MỤC ĐÍCH

Giới thiệu khái niệm giao dịch, các tính chất một giao dịch cần phải có để sự hoạt động của nó trong môi trường có "biến động" vẫn đảm bảo cho CSDL luôn ở trạng thái nhất quán.

Giới thiệu các khái niệm khả tuần tự, khả tuần tự xung đột, khả tuần tự view, khả phục hồi và cascadeless, các thuật toán kiểm thử tính khả tuần tự xung đột và khả tuần tự view

YÊU CẦU

Hiểu khái niệm giao dịch, các tính chất cần phải có của giao dịch và "ai" là người đảm bảo các tính chất đó

Hiểu các khái niệm về khả tuần tự, khả phục hồi và mối tương quan giữa chúng

Hiểu và vận dụng các thuật toán kiểm thử

KHÁI NIỆM

Một giao dịch là một đơn vị thực hiện chương trình truy xuất và có thể cập nhật nhiều hạng 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 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 thành công, 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 nhà băng 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 hạng mục dữ liệu X từ CSDL đến buffer của giao dịch thực hiện hoạt động **READ** này.
- **WRITE(X)**. chuyển hạng 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);

figure IV- 1

Ta xem xét mỗi một trong các yêu cầu 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 bị phá huỷ 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 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 phá huỷ 50$. Tổng A+B không còn được bảo tồn.

  Như vậy, kết quả của sự cố là trạng thái của hệ thống không còn phản ánh trạng thái của thế giới mà CSDL được giả thiết nắm giữ. Ta sẽ gọi trạng thái như vậy là trạng thái không nhất quán. Ta phải đảm bảo rằng tính bất nhất này không xuất hiện trong một hệ CSDL. Chú ý rằng, cho dù thế nào tại một vài thời điểm, hệ thống cũng phải ở trong trạng thái không nhất quán. Ngay cả khi 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 cung cấp, ***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.

- **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 nàyvà 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 giả hiệu năng của hệ thống. Các giải pháp khác cho phép nhiều giao dịch thực hiện cạnh 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 nào đó. Đảm bảo tính cô lập là trách nhiệm của một thành phần của hệ CSDL được gọi là thành phần quản trị cạnh 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 mà giao dịch bị bỏ dở này phải bị huỷ bỏ. Mỗi khi các thay đổi do giao dịch bị bỏ dở bị huỷ bỏ, ta nói rằng giao dịch bị cuộn lại (rolled back). Việc này là trách nhiệm của sơ đồ khôi phục nhằm quản trị các giao dịch bị bỏ dở. Một giao dịch hoàn tất thành công sự thực hiện của nó được gọi là được bàn giao (committed). Một giao dịch được bàn giao (committed), thực hiện các 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 (committed), ta không thể huỷ bỏ các hiệu quả của nó bằng các bỏ dở nó. Cách duy nhất để huỷ bỏ các hiệu quả của một giao dịch được bàn giao (committed) 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.
- **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 di vào trạng thái Committed, tương tự, một giao dịch bị bỏ dở nếu nó đã đi vào 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 hhi 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:

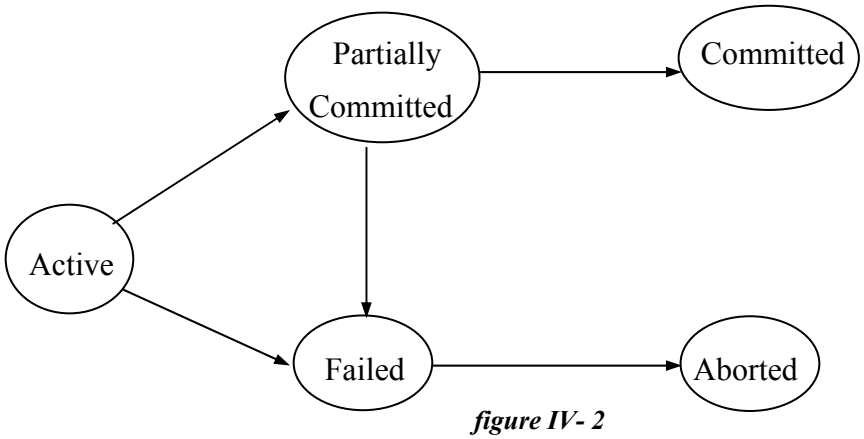


figure IV- 2

Với giả thiết sự cố hệ thống không gây ra sự mất dữ liệu trên đĩa, Một giao dịch đi vào 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***, nhưng chỉ nếu giao dịch bị bở 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.
- ***Giết 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.

Ta phải thận trọng khi thực hiện viết ngoài khả quan sát (observable external **Write** - như viết ra terminal hay máy in). Mỗi khi một viết như vậy xẩy 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 viết như thế xẩy 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 viết ngoài như vậy trong lưu trữ không hay thay đổi và thực hiện các 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 viết ngoài, hệ CSDL sẽ làm các 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 tạo bản sao của CSDL được gọi là

các bản sao bóng (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 bóng (shadow-database), một giao dịch muốn cập nhật CSDL, đầu tiên tạo ra một bản sao đầy đủ của CSDL. Tất cả các cập nhật được làm trên bản sao này, không đụng chạm tới bản gốc (bản sao bóng). 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 (committed) 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 (committed) 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 (committed), 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 hiệu quả nào được phản ánh, 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 hiệu 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 CẠNH 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 đếm xỉa tới sự thực hiện cạnh tranh các giao dịch sẽ cần thêm các công việc phụ. Một phương pháp dễ 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 cạnh 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 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 cạnh 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 cạnh 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 phá huỷ cho dù mỗi giao dịch là đúng. Một giải pháp để giải quyết vấn đề này là sử dụng định thời. Hệ CSDL phải điều khiển sự trao đổi giữa các giao dịch cạnh 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 cạnh tranh (concurrency-control scheme).

Xét hệ thống nhà băng 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 và được xác định như sau:

T_1 : **Read**(A);
A:=A-50;
Write(A);
Read(B);
B:=B+50;
Write(B);

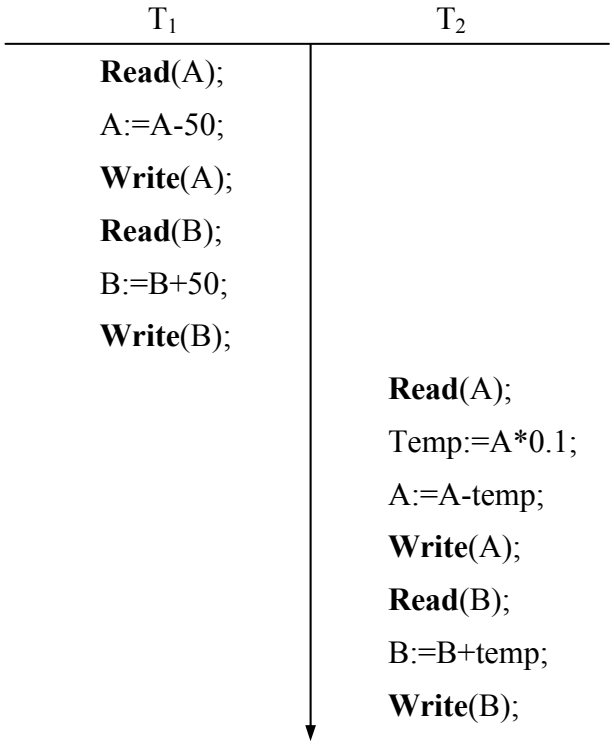
figure IV- 3

Giao dịch T_2 chuyển 10% số dư từ tài khoản A sang tài khoản B, và được xác định như sau:

T_2 : **Read**(A);
Temp:=A*0.1;
A:=A-temp;
Write(A);
Read(B);
B:=B+temp;
Write(B);

figure IV- 4

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 mỗi một tại một thời điểm theo thứ tự T_1 rồi tới T_2. Như vậy, dãy thực hiện này là như hình bên dưới, trong đó dãy các bước chỉ thị ở trong thứ tự thời gian từ đỉnh xuống đáy, các chỉ thị của T_1 nằm ở cột trái còn các chỉ thị của T_2 nằm ở cột phải:

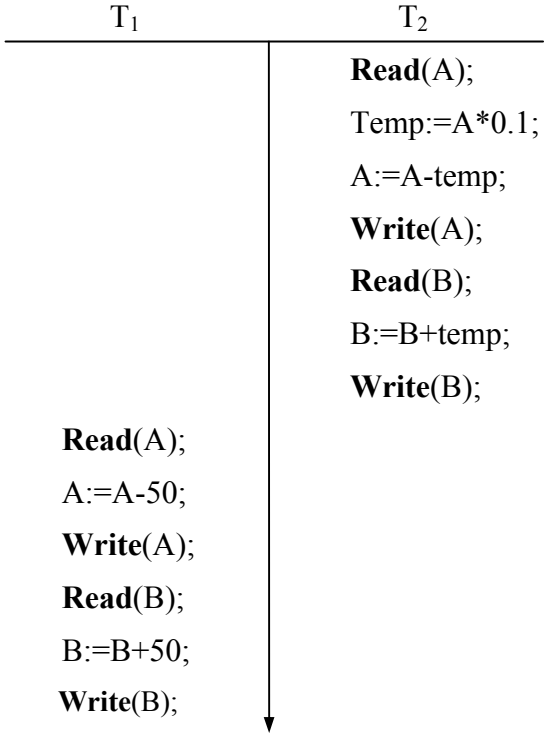


Schedule-1

figure IV- 5

Giá trị sau cùng của các tài khoản A và B, sau khi thực hiện dãy các chỉ thị theo trình tự này là 855$ và 2145$ tương ứng. Như vậy, tổng giá trị của hai tài khoản này (A + B) được bảo tồn sau khi thực hiện cả hai giao dịch.

Tương tự, nếu hai giao dịch được thực hiện mỗi một tại một thời điểm song theo trình tự T_2 rồi đến T_1 , khi đó dãy thực hiện sẽ là:



Schedule-2

figure IV- 6

Và kết quả là các giá trị cuối cùng của tài khoản A và B tương ứng sẽ là 850$ và 2150$.

Các dãy thực hiện vừa được mô tả trên được gọi là các lịch trình (schedules). Chúng biểu diễn trình tự thời gian các chỉ thị được thực hiện trong hệ thống. Một lịch trình 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 lịch trình hợp lệ nào. Các lịch trình schedule-1 và schedule-2 là tuần tự. Mỗi lịch trình 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 lịch trình. Như vậy, đối với một tập n giao dịch, có n! lịch trình tuần tự hợp lệ khác nhau. Khi một số giao dịch được thực hiện đồng thời, lịc trình 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).

Có thể có một vài dãy thực hiện, vì nhiều chỉ thị của các giao dịch có thể đan xen nhau. Nói chung, không thể dự đoán chính xác những chỉ thị nào của một giao dịch sẽ được thực hiện trước khi CPU chuyển cho giao dịch khác. Do vậy, số các lịch trình có thể đối với một tập n giao dịch lớn hơn n! nhiều.

T_1	T_2
Read(A);	
A:=A-50;	
Write(A);	
	Read(A);
	Temp:=A*0.1;
	A:=A-temp;
	Write(A);
Read(B);	
B:=B+50;	
Write(B);	
	Read(B);
	B:=B+temp;
	Write(B);

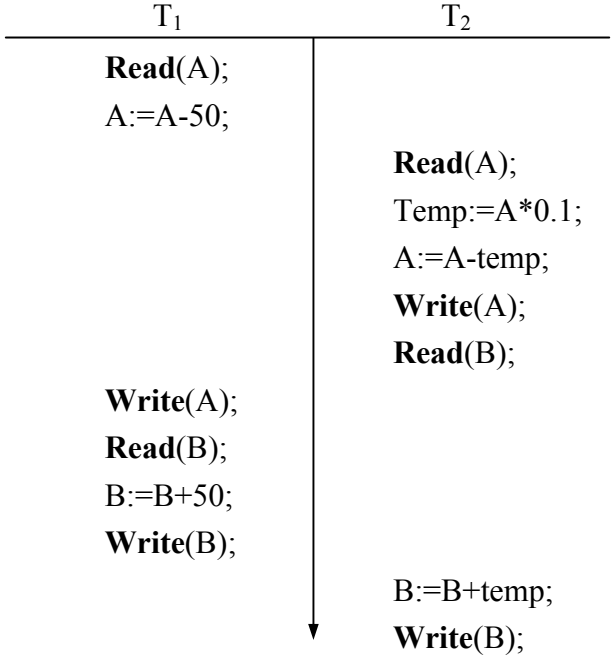
Schedule-3 --- một lịch trình cạnh tranh tương đương schedule-1

figure IV- 7

Không phải tất cả các thực hiện cạnh tranh cho ra một trạng thái đúng. Ví dụ schedule-4 sau cho ta một minh hoạ về nhận định này:

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 cạnh 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 lịch trình được phép thực hiện 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 cạnh tranh (concurrency-control component). Ta có thể đảm bảo sự nhất quán của CSDL với thực hiện cạnh tranh bằng cách nắm chắc rằng một lịch trình được thực hiện có cùng hiệu quả như một lịch trình tuần tự.

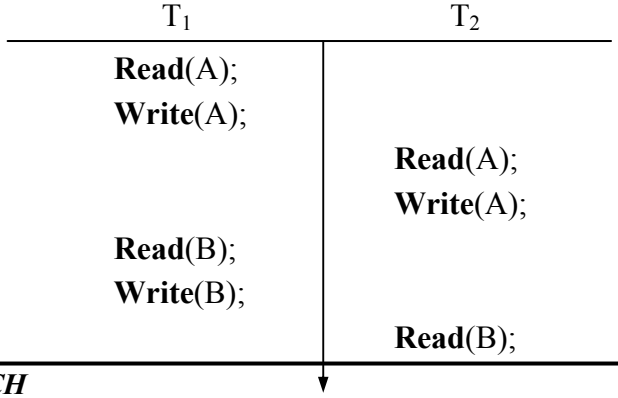


Schedule-4 --- một lịch trình cạnh tranh

figure IV- 8

TÍNH KHẢ TUẦN TỰ (Serializability)

Hệ CSDL phải điều khiển sự thực hiện cạnh 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 lịch trình nào sẽ đảm bảo tính nhất quán và các lịch trình 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 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 hạng 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 hạng 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 lịch trình, như trong biểu diễn với quy ước như vậy của schedule-3 dưới đây:



Write(B);

Schedule-3 (viết dưới dạng thoả thuận)

figure IV- 9

TUẦN TỰ XUNG ĐỘT (Conflict Serializability)

Xét lịch trình 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 hạng 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 lịch trình. Tuy nhiên, nếu I_i và I_j tham khảo cùng một hạng 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): Nếu I_i thực hiện trước I_j, Khi đó T_i không đọc giá trị được viết bởi T_j bởi chỉ thị I_j . Nếu I_j thực hiện trước I_i , T_i sẽ đọc giá trị của Q được viết bởi I_j , như vậy thứ tự của I_i và I_j là quan trọng.
3. I_i = **Write**(Q); I_j = **Read**(Q): Thứ tự của I_i và I_j là quan trọng do cùng lý do trong trường hợp trước.
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 I_i vf I_j sẽ ảnh hưởng trực tiếp đến giá trị cuối của Q trong trạng thái CSDL kết quả (của lịch trình S).

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 hạng mục dữ liệu và có ít nhất một hoạt động là **Write**. Ta xét lịch trình schedule-3 như ví dụ minh hoạ cho các chỉ thị xung đột.

T_1	T_2
Read(A);	
Write(A);	
	Read(A);
	Write(A);
Read(B);	
Write(B);	
	Read(B);
	Write(B);

figure IV- 10

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 hạng mục dữ liệu khác nhau.

I_i và I_j là hai chỉ thị liên tiếp trong lịch trình 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 lịch trình mới S' tương đương với S. 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 lịch trình tương đương – schedule-5 dưới đây

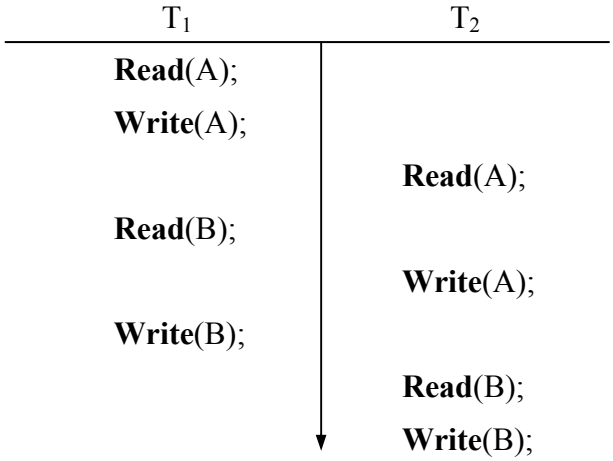
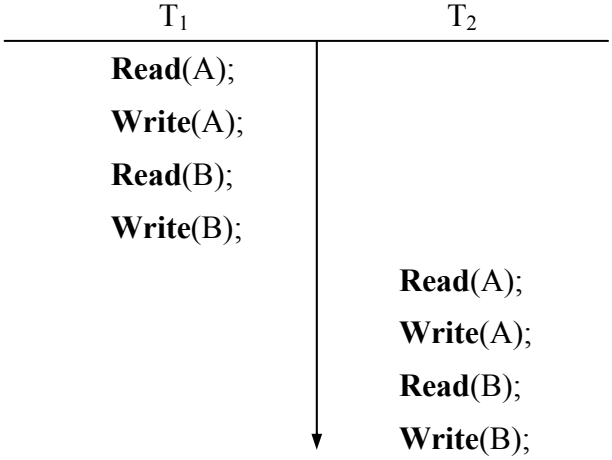


figure IV- 11

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 lịch trình mới (schedule-6 –lịch trình tuần tự) tương đương với lịch trình ban đầu (schedule-3):



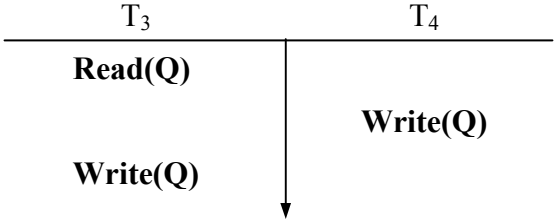
Schedule-6

figure IV- 12

Sự tương đương này cho ta thấy: bất chấp trạng thái hệ thống ban đầu, schedule-3 sẽ sinh ra cùng trạng thái cuối như một lịch trình tuần tự nào đó.

Nếu một lịch trình S có thể biến đổi thành một lịch trình S' bởi một dãy các đổi chỗ các chỉ thị không xung đột, ta nói S và S' là tương đương xung đột (*conflict equivalent*). Trong các schedule đã được nêu ở trên, ta thấy schedule-1 tương đương xung đột với schedule-3.

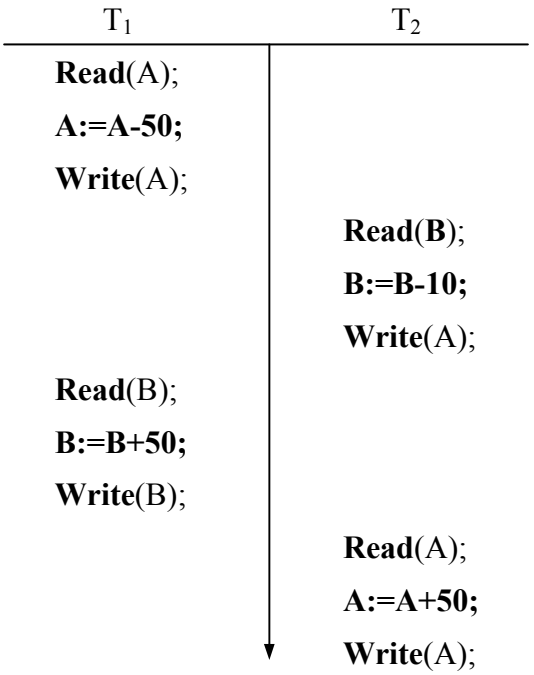
Khái niệm tương đương xung đột dẫn đến khái niệm tuần tự xung đột. Ta nói một lịch trình S là khả tuần tự xung đột (*conflict serializable*) nếu nó tương đương xung đột với một lịch trình tuần tự. Như vậy, schedule-3 là khả tuần tự xung đột. Như một ví dụ, lịch trình schedule-7 dưới đây không tương đương xung đột với một lịch trình tuần tự nào do vậy nó không là khả tuần tự xung đột:



Schedule-7

figure IV- 13

Có thể có hai lịch trình 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 lịch trình schedule-8 như dưới đây, lịch trình này không tương đương xung đột với lịch trình tuần tự $< T_1, T_5 >$ do trong lịch trình schedule-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 lịch schedule-8 hoặc sau khi thực hiện lịch trình tuần tự $<T_1, T_5 >$ là như nhau--- là 960 và 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

figure IV- 14

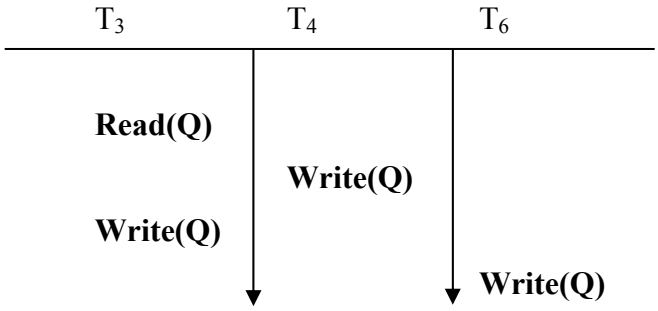
TUẦN TỰ VIEW (View Serializability)

Xét hai lịch trình S và S', trong đó cùng một tập hợp các giao dịch tham gia vào cả hai lịch trình. Các lịch trình S và S' được gọi là tương đương view nếu ba điều kiện sau được thoả mãn:

1. Đối với mỗi hạng mục dữ liệu Q, nếu giao dịch T_i đọc giá trị khởi đầu của Q trong lịch trình S, thì giao dịch T_i phải cũng đọc giá trị khởi đầu của Q trong lịch trình S'.
2. Đối với mỗi hạng mục dữ liệu Q, nếu giao dịch T_i thực hiện **Read(Q)** trong lịch trình 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 hạng mục dữ liệu Q, giao dịch thực hiện hoạt động **Write(Q)** sau cùng trong lịch trình S, phải thực hiện hoạt động **Write(Q)** sau cùng trong lịch trình S'.

Điều kiện 1 và 2 đảm bảo mỗi giao dịch đọc cùng các giá trị trong cả hai lịch trình và do vậy thực hiện cùng tính toán. Điều kiện 3 đi cặp với các điều kiện 1 và 2 đảm bảo cả hai lịch trình cho ra kết quả là trạng thái cuối cùng của hệ thống như nhau. Trong các ví dụ trước, schedule-1 là không tương tương view với lịch trình 2 do, trong schedule-1, giá trị 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 schedule-2. Schedule-1 tương đương view với schedule-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 lịch trình.

Quan niệm tương đương view đưa đến quan niệm tuần tự view. Ta nói lịch trình S là khả tuần tự view (view serializable) nếu nó tương dương view với một lịch trình tuần tự. Ta xét lịch trình sau:



Schedule-9

figure IV- 15

Nó tương đương view với lịch trình tuần tự < T_3, T_4, T_6 > do chỉ thị **Read(Q)** đọc giá trị khởi đầu của Q trong cả hai lịch trình và T_6 thực hiện **Write** sau cùng trong cả hai lịch trình như vậy schedule-9 khả tuần tự view.

Mỗi lịch trình khả tuần tự xung đột là khả tuần tự view, nhưng có những lịch trình khả tuần tự view không khả tuần tự xung đột (ví dụ schedule-9).

Trong schedule-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 mù (blind write). Các Write mù xuất hiện trong bất kỳ lịch trình khả tuần tự view không khả tuần tự xung đột.

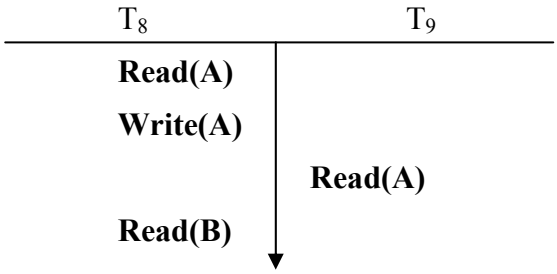
TÍNH KHẢ PHỤC HỒI (Recoverability)

Ta đã nghiên cứu các lịch trình có thể chấp nhận dưới quan điểm sự nhát 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 cạnh tranh.

Nếu giao dịch T_i thất bại vì lý do nào đó, ta cần huỷ bỏ hiệu quả của giao dịch này để đảm bảo tính nguyên tử của giao dịch. Trong hệ thống cho phép thực hiện cạnh 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 lịch trình được phép trong hệ thống.

LỊCH TRÌNH KHẢ PHỤC HỒI (Recoverable Schedule)

Xét lịch trình schedule-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 (commit) ngay sau khi thực hiện chỉ thị **Read(A).** Như vậy T_9 bàn giao trước T_8 . Giả sử T_8 thất bại trước khi bàn giao, vì T_9 vì T_9 đã đọc giá trị của hạng mục giữ liệu A được viết bởi T_8 , ta phải bỏ dở T_9 để đảm bảo tính nguyên tử giao dịch. Song 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 .



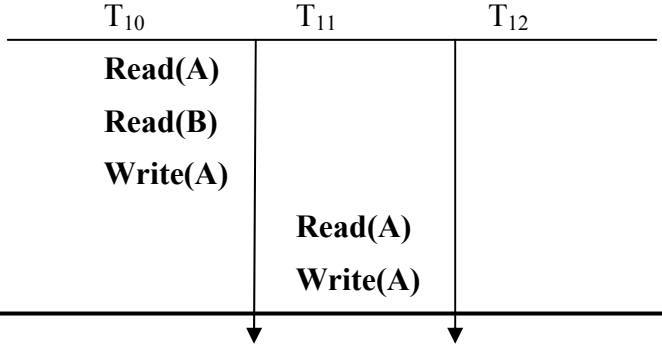
Schedule-10

figure IV- 16

Lịch trình schedule-10 là một ví dụ về lịch trình không phục hồi được và không dược phép. Hầu hết các hệ CSDL đòi hỏi tất cả các lịch trình phải phục hồi được. *Một lịch trình khả phục hồi là lịch trình trong đó, đối với mỗi cặp giao dịch T_i , T_j , nếu T_j đọc hạng 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 .*

LỊCH TRÌNH CASCADELESS (Cascadeless Schedule)

Ngay cả khi lịch trình 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 lich trình schedule-11 sau



Read(A)

Schedule-11

figure IV- 17

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 lịch trình để việc cuộn lại hàng loạt không thể xảy ra. Các lịch trình như vậy được gọi là các lịch trình cascadeless. *Một lịch trình cascadeless là một lịch trình trong đó mỗi cặp giao dịch T_i , T_j nếu T_j đọc một hạng 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 lịch trình cascadeless là khả phục hồi.

THỰC THI CÔ LẬP (Implementation of Isolation)

Có nhiều sơ đồ điều khiển cạnh tranh có thể được sử dụng để đảm bảo các tính chất một lịch trình 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 cạnh tranh, chỉ các lịch trình 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.

Như một ví dụ, ta xét một sơ đồ điều khiển cạnh tranh sau: Một giao dịch tậu một chốt (lock) trên toàn bộ CSDL trước khi nó khởi động và tháo chốt khi nó đã bàn giao. Trong khi giao dịch giữ chốt không giao dịch nào khác được phép tậu chốt và như vậy phải chờ đến tận khi chốt được tháo. Trong đối sách chốt, chỉ một giao dịch được thực hiện tại một thời điểm và như vậy chỉ lịch trình tuần tự được sinh ra. Sơ đồ điều khiển cạnh tranh này cho ra một hiệu năng cạnh tranh nghèo nàn. Ta nói nó cung cấp một bậc cạnh tranh nghèo (poor degree of concurrency).

Mục đích của các sơ đồ điều khiển cạnh tranh là cung cấp một bậc canh tranh cao trong khi vẫn đảm bảo các lịch trình được sinh ra là khả tuần tự xung đột hoặc khả tuần tự view và cascadeless.

ĐỊNH NGHĨA GIAO DỊCH TRONG SQL

Chuẩn SQL đặc tả sự bắt đầu một giao dịch một cánh không tường minh. Các giao dịch được kết thúc bởi một trong hai lệnh SQL sau:

- **Commit work** bàn giao giao dịch hiện hành và bắt đầu một giao dịch mới
- **Rollback work** gây ra sự huỷ bỏ giao dịch hiện hành

Từ khoá **work** là chọn lựa trong cả hai lệnh. Nếu một chương trình kết thúc thiếu cả hai lệnh này, các cập nhật hoặc được bàn giao hoặc bị cuộn lại là các sự thực hiện phụ thuộc.

Chuẩn cũng đặc tả hệ thống phải đảm bảo cả tính khả tuần tự và tính tự do từ việc cuộn lại hàng loạt. Định nghĩa tính khả tuần tự được ding bởi chuẩn là một lịch trình phải có cùng hiệu quả như một lịch trình tuần tự như vậy tính khả tuần tự xung đột và view đều được chấp nhận.

Chuẩn SQL-92 cũng cho phép một giao dịch đặc tả nó có thể được thực hiện theo một cách mà có thể làm cho nó trở nên không khả tuần tự với sự tôn trọng các giao dịch khác. Ví dụ, một giao dịch có thể hoạt động ở mức **Read uncommitted**, cho phép giao dịch đọc các mẩu tin them chí nếu chúng không được bàn giao. Đặc điểm này được cung cấp cho các giao dich dài các

kết quả của chúng không nhất thiết phải chính xác. Ví dụ, thông tin xấp xỉ thường là đủ cho các thống kê được dùng cho tối ưu hoá vấn tin.

Các mức nhất quán được đặc tả trong SQL-92 là:

- **Serializable :** mặc nhiên
- **Repeatable read :** chỉ cho phép đọc các record đã được bàn giao, hơn nữa yêu cầu giữa hai **Read** trên một record bởi một giao dịch không một giao dịch nào khác được phép cập nhật record này. Tuy nhiên, giao dịch có thể không khả tuần tự với sự tôn trọng các giao dịch khác. Ví dụ, khi tìm kiếm các record thoả mãn các điều kiện nào đó, một giao dịch có thể tìm thấy một vài record được xen bởi một giao dịch đã bàn giao,
- **Read committed:** Chỉ cho phép đọc các record đã được bàn giao, nhưng không có yêu cầu thêm trên các **Read** khả lặp. Ví dụ, giữa hai **Read** của một record bởi một giao dịch, các mẩu tin có thể được cập nhật bởi các giao dịch đã bàn giao khác.
- **Read uncommitted:** Cho phép đọc cả các record chưa được bàn giao. Đây là mức nhất quán thấp nhất được phép trong SQL-92.

KIỂM THỬ TÍNH KHẢ TUẦN TỰ

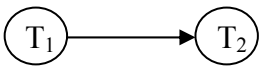
Khi thiết kế các sơ đồ điều khiển cạnh tranh, ta phải chứng tỏ rằng các lịch trình được sinh ra bởi sơ đồ là khả tuần tự. Để làm điều đó, trước tiên ta phải biết làm thế nào để xác định, với một lịch trình cụ thể đã cho, có là khả tuần tự hay không.

KIỂM THỬ TÍNH KHẢ TUẦN TỰ XUNG ĐỘT

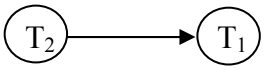
Giả sử S là một lịch trình. Ta xây dựng một đồ thị định hướng, được gọi là đồ thị trình tự (precedence graph), từ S. Đồ thị gồm một cặp (V, E) trong đó V là tập các đỉnh và E là tập các cung. Tập các đỉnh bao gồm tất cả các giao dịch tham gia vào lịch trình. Tập các cung bao gồm tất cả các cung dạng $T_i \rightarrow T_j$ sao cho một trong các điều kiện sau được thoả mãn:

1. T_i thực hiện **Write(Q)** trước T_j thực hiện **Read(Q).**
2. T_i thực hiện **Read(Q)** trước khi T_j thực hiện **Write(Q).**
3. 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 lịch trình tuần tự S' nào tương đương với S, T_i phải xuất hiện trước T_j .

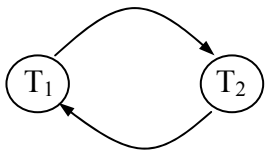
Đồ thị trình tự đối với schedule-1 là: 

vì tất cả các chỉ thị của T_1 được thực hiện trước chỉ thị đầu tiên của T_2

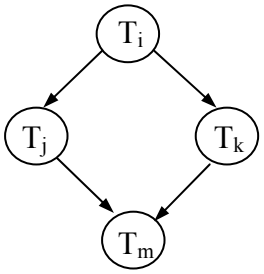
Đồ thị trình tự đối với schedule-2 là: 

vì tất cả các chỉ thị của T_2 được thực hiện trước chỉ thị đầu tiên của T_1

Đồ thị trình tự đối với schedule-4 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 lịch trình S không là khả tuần tự xung đột. Nếu đồ thị không chứa chu trình, khi đó lịch trình S là khả tuần tự xung đột. Thứ tự khả tuần tự có thể nhận được thông qua sắp xếp topo (topological sorting), nó xác định một thứ tự tuyến tính nhất quán với thứ tự bộ phận của đồ thị trình tự. Nói chung, có một vài thứ tự tuyến tính có thể nhận được qua sắp xếp topo. Ví dụ, đồ thị sau:



Có hai thứ tự tuyến tính chấp nhận được là:

figure IV- 18

Như vậy, để kiểm thử tính khả tuần tự xung đột, ta cần xây dựng đồ thị trình tự và gọi thuật toán phát hiện chu trình. Ta nhận được một sơ đồ thực nghiệm để xác định tính khả tuần tự xung đột. Như ví dụ, schedule-1 và schedule-2, đồ thị trình tự của chúng không có chu trình, do vậy chúng là các chu trình khả tuần tự xung đột, trong khi đồ thị trình tự của schedule-4 chứa chu trình do vậy nó không là khả tuần tự xung đột.

KIỂM THỬ TÍNH KHẢ TUẦN TỰ VIEW

Ta có thể sửa đổi phép kiểm thử đồ thị trình tự đối với tính khả tuần tự xung đột dể kiểm thử tính khả tuần tự view. Tuy nhiên, phép kiểm thử này phải trả giá cao về thời gian chạy.

Xét lịch trình schedule-9, nếu ta tuân theo quy tắc trong phép kiểm thử tính khả tuần tự xung đột để tạo đồ thị trình tự, ta nhận được đồ thị sau:

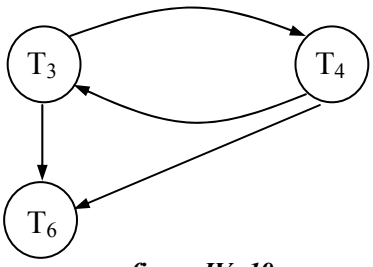


figure IV- 19

Đồ thị này có chu trình, do vậy schedule-9 không là khả tuần tự xung đột. Tuy nhiên, đã đã thấy nó là khả tuần tự view (do nó tương đương với lịch trình tuần tự < T_3, T_4 , T_6 >).

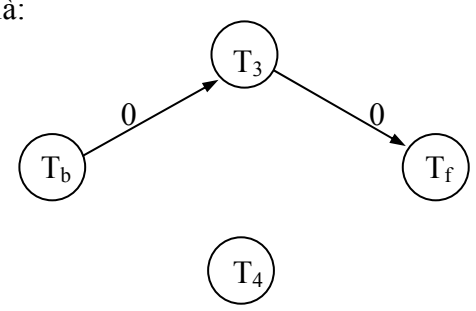
Cung $T_3 \rightarrow T_4$ không được xen vào đồ thị vì các giá trị của hạng mục Q được sản sinh bởi T_3 và T_4 không được dùng bởi bất kỳ giao dịch nào khác và T_6 sản sinh ra giá trị cuối mới của Q. Các chỉ thị **Write(Q)** của T_3 và T_4 được gọi là các **Write vô dụng (Useless Write)**. Điều trên chỉ ra rằng không thể sử dụng đơn thuần sơ đồ đồ thị trình tự để kiểm thử tính khả tuần tự view. Cần thiết phát triển một sơ đồ cho việc quyết định cung nào là cần phải xen vào đồ thị trình tự.

Xét một lịch trình S. Giả sử giao dịch T_j đọc hạng mục dữ liệu Q được viết bởi T_i . Rõ ràng là nếu S là khả tuần tự view, khi đó, trong bất kỳ lịch trình tuần tự S' tương đương với S, T_i phải đi trước T_j . Bây giờ giả sử rằng, trong lịch trình S, giao dịch T_k thực hiện một **Write(Q)**, khi đó, trong lịch trình S', T_k phải hoặc đi trước T_i hoặc đi sau T_j . Nó không thể xuất hiện giữa T_i và T_j vì như vậy T_j không đọc giá trị của Q được viết bởi T_i và như vậy S không tương đương view với S'. Các ràng buộc này không thể biểu diễn được trong thuật ngữ của mô hình đồ thị trình tự đơn giản được nêu lên trước đây. Như trong ví dụ trước, khó khăn nảy sinh ở chỗ ta biết một trong hai cung $T_k \rightarrow T_i$ và $T_j \rightarrow T_k$ phải được xen vào đồ thị nhưng ta chưa tạo được quy tắc để xác định sự lựa chọn thích hợp. Để tạo ra quy tắc này, ta cần mở rộng đồ thị định hướng để bao hàm các cung gán nhãn, ta gọi đồ thị như vậy là đồ thị trình tự gán nhãn (Label precedence graph). Cũng như trước đây, các nút của đồ thị là tất cả các giao dịch tham gia vào lịch trình. Các quy tắc xen cung gán nhãn được diễn giải như sau:

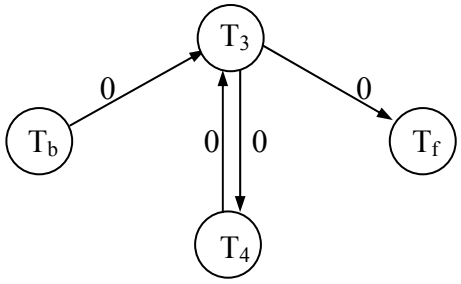
Giả sử S là lịch trình gồm các giao dịch { T_1 , T_2 , ... , T_n }. T_b và T_f là hai giao dịch giả: T_b phát ra **Write(Q)** đối với mỗi Q được truy xuất trong S, T_f phát ra **Read(Q)** đối với mỗi Q được truy xuất trong S. Ta xây dựng lich trình mới S' từ S bằng cách xen T_b ở bắt đầu của S và T_f ở cuối của S. Đồ thị trình tự gán nhãn đố với S' được xây dung dựa trên các quy tắc:

1. Thêm cung $T_i \rightarrow^0 T_j$, nếu T_j đọc giá trị của hạng mục dữ liệu Q được viết bởi T_i
2. Xoá tất cả các cung liên quan tới các giao dịch vô dụng. Một giao dịch T_i được gọi là vô dụng nếu không có con đường nào trong đồ thị trình tự dẫn từ T_i đến T_f .
3. Đối với mỗi hạng mục dữ liệu Q sao cho T_j đọc giá trị của Q được viết 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 $T_j \neq T_f$, khi đó xen cung $T_j \rightarrow^0 T_k$ vào đồ thị trình tự gán nhãn
   b. Nếu $T_i \neq T_b$ và $T_j = T_f$ khi đó xen cung $T_k \rightarrow^0 T_i$ vào đồ thị trình tự gán nhãn
   c. Nếu $T_i \neq T_b$ và $T_j \neq T_f$ khi đó xen cả hai cung $T_k \rightarrow^p T_i$ và $T_j \rightarrow^p T_k$ vào đồ thị trình tự gán nhãn, trong đó p là một số nguyên duy nhất lớn hơn 0 mà chưa được sử dụng trước đó để gán nhãn cung.

Quy tắc 3c phản ánh rằng nếu T_i viết hạng mục dữ liệu được đọc bởi T_j thì một giao dịch T_k viết cùng hạng 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. Như một ví dụ, ta xét schedule-7. Đồ thị trình tự gán nhãn của nó được xây dựng qua các bước 1 và 2 là:



Đồ thị sau cùng của nó là (cung $T_3 \rightarrow T_4$ là kết quả của 3a, cung $T_4 \rightarrow T_3$ là kết quả của 3b) :



Đồ thị trình tự gán nhãn của schedule-9 là:

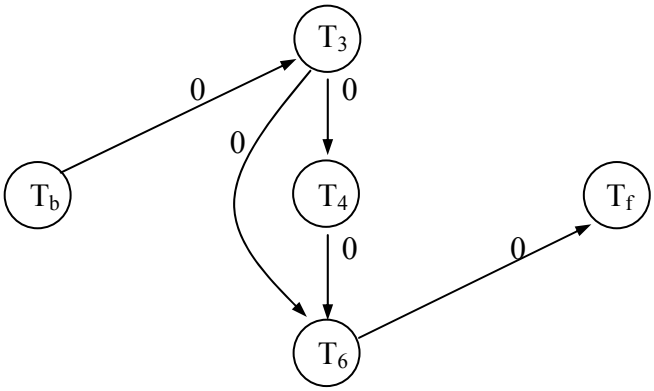
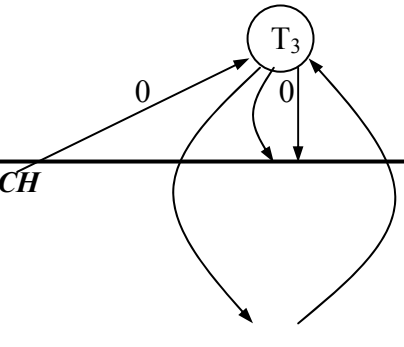


figure IV- 20

Cuối cùng, ta xét lịch trình schedule-10:

T_3	T_3	T_7
Read(Q)		
	Write(Q)	
		Read(Q)
Write(Q)		
		Write(Q)

Đồ thị trình tự gán nhãn của schedule-10 là:



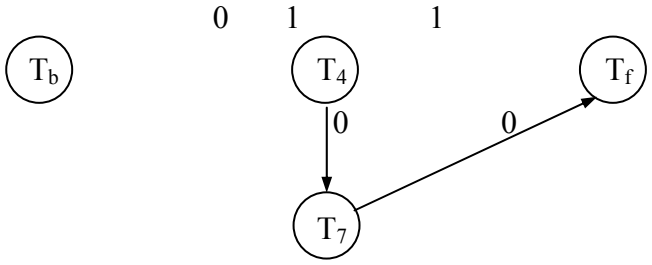


figure IV- 21

Đồ thị trình tự gán nhãn của schedule-7 chứa chu trình tối tiểu : $T_3 \rightarrow T_4 \rightarrow T_3$

Đồ thị trình tự gán nhãn của schedule-10 chứa chu trình tối tiểu: $T_3 \rightarrow T_1 \rightarrow T_3$

Đồ thị trình tự gán nhãn của schedule-9 không chứa chứa chu trình nào.

Nếu đồ thị trình tự gán nhận không chứa chu trình, lịch trình tương ứng là khả tuàn tự view, như vậy schedule-9 là khả tuần tự view. Tuy nhiên, nếu đồ thị chứa chu trình, điều kiện này không kéo theo lịch trình tương ứng không là khả tuần tự view. Đồ thị trình tự gán nhãn của schedule-7 chứa chu trình và lịch trình này không là khả tuần tự view. Bên cạnh đó, lịch trình schedule-10 là khả tuần tự view, nhưng đồ thị trình tự gán nhãn của nó có chứa chu trình. 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 chứa đúng một cung trong mỗi cặp. Nếu một đồ thị nào đó trong các đồ thị này là phi chu trình, khi đó lịch trình 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 vấn đề NP-complet !!!

Ta xét đồ thị schedule-10. nó có đúng một cặp tách biệt. Hai đồ thị triên biệt là:

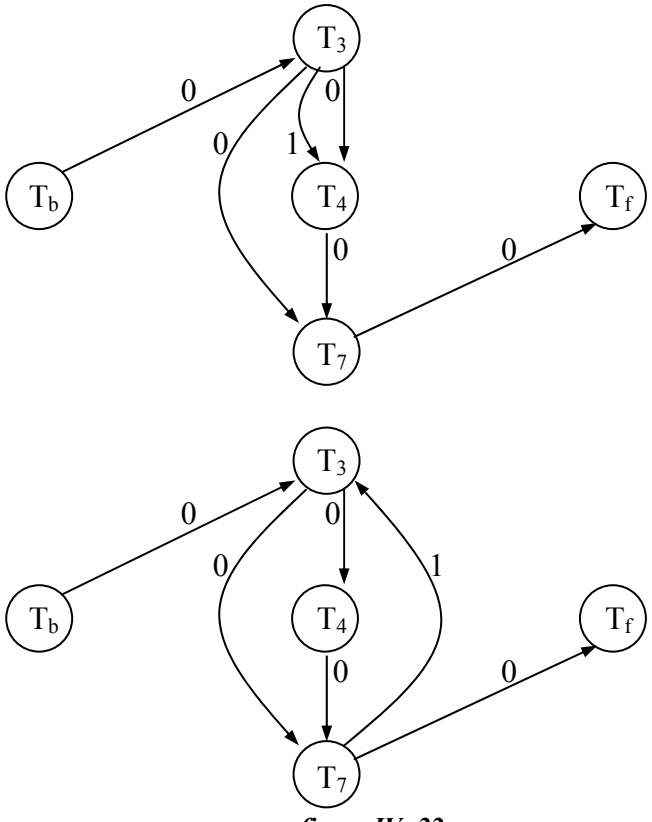


figure IV- 22

Đồ thị thứ nhất không chứa chu trình, do vậy lịch trình là khả tuần tự view.

BÀI TẬP CHƯƠNG IV

IV.1 Liệt kê các tính chất ACID. Giải thích sự hữu ích của mỗi một trong chúng.

IV.2 Trong khi thực hiện, một giao dịch trải qua một vài trạng thái đến tận khi nó bàn giao hoặc bỏ dở. Liệt kê tất cả các dãy trạng thái có thể giao dịch có thể trải qua. Giải thích tại sao mỗi bắc cầu trạng thái có thể xảy ra.

IV.3 Giải thích sự khác biệt giữa lịch trình tuần tự (Serial schedule) và lịch trình khả tuần tự (Serializable schedule).

IV.4 Xét hai giao dịch sau:

T_1 : **Read(A);**
Read(B);
If A=0 then B:=B+1;
Write(B).

T_2 : **Read(B);**
Read(A);
If B=0 then A:=A+1;
Write(A).

Giả thiết yêu cầu nhất quán là A=0 V B=0 với A=B=0 là các giá trị khởi đầu

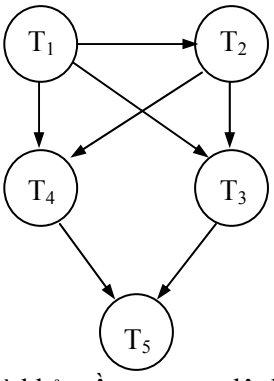
a. Chứng tỏ rằng mỗi sự thự hiện tuần tự bao gồm hai giao dịch này bảo tồn tính nhất quán của CSDL.

b. Nêu một sự thực hiện cạnh tranh của T_1 và T_2 sinh ra một lịch trình không khả tuần tự.

c. Có một sự thực hiện cạnh tranh của T_1 và T_2 sinh ra một lịch trình khả tuần tự không ?

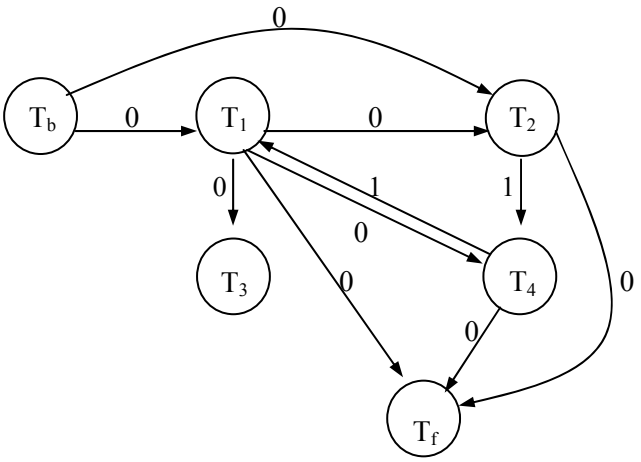
IV.5 Do một lịch trình khả tuần tự xung đột là một lịch trình khả tuần tự view. Tại sao ta lại nhấn mạnh tính khả tuần tự xung đột hơn tính khả tuần tự view?

IV.6 Xét đồ thị trình tự sau:



Lịch trình tương ứng là khả tuần tự xung đột không? Giải thích

IV.7 Xét đồ thị trình tự gán nhãn sau:



Lịch trình tương ứng là khả tuần tự view không? Giải thích.

IV.8 Lịch trình khả phục hồi là gì? Tại sao cần thiết tính khả phục hồi của một lịch trình?

IV.9 Lịch trình cascadeless là gì? Tại sao cần thiết tính cascadeless của lịch trình?

CHƯƠNG V
ĐIỀU KHIỂN CẠNH TRANH
(Concurrency Control)

MỤC ĐÍCH

Một trong các tính chất cơ bản của một giao dịch là *tính cô lập*. Khi một vài giao dịch thực hiện một cách cạnh tranh trong CSDL, tính cô lập có thể không được bảo tồn. Đối với hệ thống, cần phải điều khiển sự trao đổi giữa các giao dịch cạnh tranh; sự điều khiển này được thực hiện thông qua một trong tập hợp đa dạng các cơ chế được gọi là *sơ đồ điều khiển cạnh tranh*.

Các sơ đồ điều khiển cạnh tranh được xét trong chương này được dựa trên tính khả tuần tự. Trong chương này ta cũng xét sự quản trị các giao dịch thực hiện cạnh canh nhưng không xét đến sự cố hỏng hóc.

YÊU CẦU

Hiểu các khái niệm

Hiểu các kỹ thuật điều khiển cạnh tranh:

- Các kỹ thuật dựa trên chốt (lock)
- Các kỹ thuật dựa trên tem thởi gian
- Các kỹ thuật hỗp hợp

Hiểu nguyên lý của các kỹ thuật này

Hiểu các kỹ thuật điều khiển deadlock

V.1. GIAO THỨC DỰA TRÊN CHỐT

Một phương pháp để đảm bảo tính khả tuần tự là yêu cầu việc truy xuất đến hạng mục dữ liệu được tiến hành theo kiểu loại trừ tương hỗ; có nghĩa là trong khi một giao dịch đang truy xuất một hạng mục dữ liệu, không một giao dịch nào khác có thể sửa đổi hạng mục này. Phương pháp chung nhất được dùng để thực thi yêu cầu này là cho phép một giao dịch truy xuất một hạng mục dữ liệu chỉ nếu nó đang giữ chốt trên hạng mục dữ liệu này.

V.1.1. CHỐT (Lock)

Có nhiều phương thức chốt hạng mục dữ liệu. Ta hạn chế việc nghiên cứu trên hai phương thức:

1. **Shared.** Nếu một giao dịch T_i nhận được một chốt ở phương thức shared (ký hiệu là S) trên hạng 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 chốt ở phương thức Exclusive (ký hiệu là X), khi đó T_i có thể cả đọc lẫn viết Q.

Ta yêu cầu là mỗi giao dịch đòi hỏi một chốt ở một phương thức thích hợp trên hạng mục dữ liệu Q, phụ thuộc vào kiểu hoạt động mà nó sẽ thực hiện trên Q. Giả sử một giao dịch T_i đồi hỏi một chốt phương thức A trên hạng mục Q mà trên nó giao dịch T_j ($T_j \neq T_i$) hiện đang giữ một chốt phương thức B. Nếu giao dịch T_i có thể được cấp một chốt trên Q ngay, bất chấp sự hiện diện của chốt phương thức B, khi đó ta nói phương thức A tương thích với phương thức B. Một hàm như vậy có thể được biểu diễn bởi một ma trận. Quan hệ tương thích giữa hai phương thức chốt được 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.

figure V- 1

Các chốt phương thức **shared** có thể được giữ đồng thời trên một hạng mục dữ liệu. Một chốt **exclusive** đến sau phải chờ đến tận khi tất cả các chốt phương thức **shared** đến trước được tháo ra.

Một giao dịch yêu cầu một chốt **shared** trên hạng mục dữ liệu Q bằng cách thực hiện chỉ thị **lock-S(Q)**, yêu cầu một chốt **exclusive** thông qua chỉ thị **lock-X(Q)**. Một hạng mục dữ liệu Q có thể được tháo chốt thông qua chỉ thị **unlock(Q)**.

Để truy xuất một hạng mục dữ liệu, giao dịch T_i đầu tiên phải chốt hạng mục này. Nếu hạng mục này đã bị chốt bởi một giao dịch khác ở phương thức không tương thích, bộ điều khiển cạnh tranh sẽ không cấp chốt cho đến tận khi tất cả các chốt không tương thích bị giữ bởi các giao dịch khác được tháo. Như vậy T_i phải chờ đến tận khi tất cả các chốt không tương thích bị giữ bởi các giao dịch khác được giải phóng.

Giao dịch T_i có thể tháo chốt một hạng mục dữ liệu mà nó đã chốt trước đây. Một giao dịch cần thiết phải giữ một chốt trên một hạng mục dữ liệu chừng nào mà nó còn truy xuất hạng mục này. Hơn nữa, đối với một giao dịch việc tháo chốt ngay sau truy xuất cuối cùng đến hạng 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 đươch xác định như sau:

T_1 : **Lock-X(B);**
Read(B);
B:=B-50;
Write(B);
Unlock(B);
Lock-X(A);
Read(A);
A:=A+50;
Write(A);
Unlock(A);

figure V- 2

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);

figure V- 3

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 cạnh tranh, giả sử theo lịch trình schedule-1, trong trường hợp như vậy giao dịch T_2 sẽ hiển thị giá trị 250$ --- một kết quả không đúng. Lý do của sai lầm này là do giao dịch T_1 đã tháo chốt hạng mục B quá sớm và T_2 đã tham khảo một trạng thái không nhất quán !!!

Lịch trình schedule 1 bày tỏ các hành động được thực hiện bởi các giao dịch cũng như các thời điểm khi các chốt được cấp bởi bộ quản trị điều khiển cạnh tranh. Giao dịch đưa ra một yêu cầu chốt không thể thực hiện hành động kế của mình đến tận khi chốt được cấp bởi bộ quản trị điều khiển cạnh tranh; do đó, chốt phải được cấp trong khoảng thời gian giữa hoạt động yêu cầu chốt và hành động sau của giao dịch. Sau này ta sẽ luôn giả thiết chốt được cấp cho giao dịch ngay trước hành động kế và như vậy ta có thể bỏ qua cột bộ quản trị điều khiển cạnh tranh trong bảng

T_1	T_2	Bộ quản trị điều khiển cạnh 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)
	Read(A)	
	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)		

Schedule-1

figure V- 4

Bây giờ giả sử rằng tháo chốt bị làm trễ đến cuối giao dịch. Giao dịch T_3 tương ứng với T_1 với tháo chốt 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);

figure V- 5

Giao dịch T_4 tương ứng với T_2 với tháo chốt 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);

figure V- 6

Các lịch trình có thể trên T_3 và T_4 không để cho T_4 hiển thị trạng thái không nhất quán.

Tuy nhiên, sử dụng chốt có thể dẫn đến một tình huống không mong đợi. Ta hãy xét lịch trình bộ phận schedule-2 trên T_3 và T_4 sau:

T_3	T_4
Lock-X(B)	

Schedule-2

figure V- 7

Do T_3 giữ một chốt phương thức Exclusive trên B, nên yêu cầu một chốt phương thức shared của T_4 trên B phải chờ đến khi T_3 tháo chốt. Cũng vậy, T_3 yêu cầu một chốt Exclusive trên A trong khi T_4 đang giữ một chốt 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 (khoá chết). Khi tình huống khoá chết xảy ra hệ thống buộc phải cuộn lại một trong các giao dịch. Mỗi khi một giao dịch bị cuộn lại, các hạng mục dữ liệu bị chốt bởi giao dịch phải được tháo chốt 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 chốt hoặc tháo chốt hạng mục dữ liệu ngay khi có thể sau đọc hoặc viết hạng mục, ta có thể rơi vào trạng thái không nhất quán. Mặt khác, nếu ta không tháo chốt một hạng mục dữ liệu trước khi yêu cầu một chốt trên một hạng mục khác, dealock có thể xảy ra. Có các phương pháp tránh dealock trong một số tình huống, tuy nhiên nói chung dealock là khó tránh khi sử dụng chốt nếu ta muốn tránh trạng thái không nhất quán. Dealock 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.

Ta sẽ yêu cầu mỗi giao dịch trong hệ thống tuân theo một tập các quy tắc , được gọi là giao thức chốt (locking protocol), chỉ định khi một giao dịch có thể chốt và tháo chốt mỗi một trong các hạng mục dự liệu. Giao thức chốt hạn chế số các lịch trình có thể. Tập các lịch trình như vậy là một tập con thực sự của tập tất cả các lịch trình khả tuần tự có thể.

Xét { T_0 , T_1 , ..., T_n } một tập các giao dịch tham gia vào lịch trình S. Ta nói T_i đi trước T_j trong S, và được viết là $T_i \rightarrow T_j$, nếu tồn tại một hạng mục dữ liệu Q sao cho T_i giữ chốt phương thức A trên Q , T_j giữ chốt 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 lịch trình tuần tự nào.

Ta nói một lịch trình S là hợp lệ dưới một giao thức chốt nếu S là một lịch trình tuân thủ các quy tắc của giao thức chốt đó. Ta nói rằng một giao thức chốt đảm bảo tính khả tuần tự xung đột nếu và chỉ nếu đối với tất cả các lịch trình hợp lệ, quan hệ $\rightarrow$ kết hợp là phi chu trình.

V.1.2. CẤP CHỐT

Khi một giao dịch yêu cầu một chốt trên một hạng mục dữ liệu ở một phương thức và không có một giao dịch nào khác giữ một chốt trên cùng hạng mục này ở một phương thức xung đột, chốt có thể được cấp. Tuy nhiên, phải thận trọng để tránh kịch bản sau: giả sử T_2 giữ một chốt phương thức shared trên một hạng mục dữ liệu, một giao dịch khác T_1 yêu cầu một chốt phương thức exclusive cũng trên hạng mục này, rõ ràng T_1 phải chờ T_2 tháo chốt. Trong khi đó một giao dịch khác T_3 yêu cầu một chốt phương thức shared, do yêu cầu chốt này tương thích với phương thức chốt được giữ bởi T_1 nên nó được cấp cho T_3. Tại thời điểm T_2 tháo chốt, T_1 vẫn phải chờ sự tháo chốt của T_3, nhưng bây giờ lại có một giao dịch T_4 yêu cầu một chốt phương thức shared và nó lại được cấp do tính tương thích và cứ như vậy, có thể T_1 sẽ không bao giờ được cấp chốt mà nó yêu cầu trên hạng mục dữ liệu. Ta gọi hiện tượng này là bị chết đói (starved).

Để tránh sự chết đói của các giao dịch, việc cấp chốt được tiến hành như sau: Khi một giao dịch T_i yêu cầu một chốt trên một hạng mục dữ liệu Q ở phương thức M, chốt sẽ được cấp nếu các điều kiện sau được thoả mãn:

1. Không có giao dịch khác đang giữ một chốt trên Q ở phương thức xung đột với M
2. Không có một giao dịch nào đang chờ được cấp một chốt trên M và đã đưa ra yêu cầu về chốt trước T_i

V.1.3. GIAO THỨC CHỐT HAI KỲ (Two-phase locking protocol)

Giao thức chốt hai kỳ là một giao thức đảm bảo tính khả tuần tự. Giao thức này yêu cầu mỗi một giao dịch phát ra yêu cầu chốt và tháo chốt thành hai kỳ:

1. **Kỳ xin chốt (Growing phase).** Một giao dịch có thể nhận được các chốt, nhưng có không thể tháo bất kỳ chốt nào
2. **Kỳ tháo chốt (Shrinking phase).** Một giao dịch có thể tháo các chốt nhưng không thể nhận được một chốt mới nào.

Khởi đầu, một giao dịch ở kỳ xin chốt. Giao dịch tậu được nhiều chốt như cần thiết. Mỗi khi giao dịch tháo một chốt, nó đi vào kỳ tháo chốt và nó không thể phát ra bất kỳ một yêu cầu chốt nào nữa. Các giao dich T_3 và T_4 là hai kỳ. Các giao dịch T_1 và T_2 không là hai kỳ. Người ta có thể chứng minh được giao thức chốt hai kỳ đảm bảo tính khả tuần tự xung đột, nhưng không đảm bảo tránh được dealock và việc cuộn lại hàng loạt. Cuộn lại hàng loạt có thể tránh được bởi một sự sửa đổi chốt hai kỳ được gọi là giao thức chốt hai kỳ nghiêm ngặt. Chốt hai kỳ nghiêm

ngặt đòi hỏi thêm tất cả các chốt phương thức exclusive phải được giữ đến tận khi giao dịch bàn giao. Yêu cầu này đảm bảo rằng bất kỳ dữ liệu nào được viết bởi một giao dịch chưa bàn giao bị chốt trong phương thức exclusive đến tận khi giao dịch bàn giao, điều đó ngăn ngừa bất kỳ giao dịch khác đọc dữ liệu này.

Một biến thể khác của chốt hai kỳ là giao thức chốt hai kỳ nghiêm khắc. Nó đòi hỏi tất cả các chốt được giữ đến tận khi giao dịch bàn giao. Hầu hết các hệ CSDL thực hiện chốt hai kỳ nghiêm ngặt hoặc nghiêm khắc.

Một sự tinh chế giao thức chốt hai kỳ cơ sở dựa trên việc cho phép chuyển đổi chốt: nâng cấp một chốt shared sang exclusive và hạ cấp một chốt exclusive thành chốt shared. Chuyển đổi chốt không thể cho phép một cách tuỳ tiện, nâng cấp chỉ được phép diễn ra trong kỳ xin chốt, còn hạ cấp chỉ được diễn ra trong kỳ tháo chốt. Một giao dịch thử nâng cấp một chốt trên một hạng mục dữ liệu Q có thể phải chờ. Giao thức chốt hai kỳ với chuyển đổi chốt cho phép chỉ sinh ra các lịch trình khả tuần tự xung đột. Nếu các chốt exclusive được giữ đến tận khi bàn giao, các lịch trình sẽ là cascadeless.

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($A_1 + A_2$).

figure V- 8

Nếu ta sử dụng giao thức chốt hai kỳ, khi đó T_8 phải chốt A_1 ở phương thức exclusive. Bởi vậy, sự thực hiện cạnh tranh của hai giao dịch rút cuộc trở thành thực hiện tuần tự. Ta thấy rằng T_8 cần một chốt 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 chốt A_1 ở phương thức shared, và đổi chốt này sang phương thức exclusive sau này. Như vậy ta có thể nhận được tính cạnh tranh cao hơn, vì như vậy T_8 và T_9 có thể truy xuất đến A_1 và A_2 đồng thời.

Ta biểu thị sự chuyển đổi từ phương thức shared sang phương thức exclusive bởi upgrade và từ phương thức exclusive sang phương thức shared bởi downgrade. Upgrade chỉ được phép xảy ra trong kỳ xin chốt và downgrade chỉ được phép xảy ra trong kỳ tháo chốt. Lịch trình chưa hoàn tất dưới đây cho ta một minh hoạ về giao thức chốt hai kỳ với chuyển đổi chốt.

Chú ý rằng một giao dịch thử cập nhật một chốt trên một hạng mục dữ liệu Q có thể buộc phải chờ. Việc chờ bắt buộc này xảy ra khi Q đang bị chốt bởi giao dịch khác ở phương thức shared.

Giao thức chốt hai kỳ với chuyển đổi chốt chỉ sinh ra các lịch trình khả tuần tự xung đột, các giao dịch có thể được tuần tự hoá bởi các điểm chốt của chúng. Hơn nữa, nếu các chốt exclusive được giữ đến tận khi kết thúc giao dịch, lịch trình sẽ là cascadeless.

T_8	T_9
Lock-S(A_1)	
	Lock-S(A_2)
Lock-S(A_2)	
	Lock-S(A_2)
Lock-S(A_3)	
…	
	Unlock(A_1)
	Unlock(A_2)
UpGrade(A_1)	

figure V- 9

Ta mô tả một sơ đồ đơn giản nhưng dược sử dụng rộng rãi để sinh tự động các chỉ thị chốt và tháo chốt thích hợp cho một giao dịch: Mỗi khi giao dich T xuất ra một chỉ thị **Read(Q),** hệ thống sẽ xuất ra một chỉ thị **Lock-S(Q)** ngay trước chỉ thị **Read(Q)**. Mỗi khi giao dịch T xuất ra một hoạt động **Write(Q),** hệ thống sẽ kiểm tra xem T đã giữ một chốt shared nào trên Q hay chưa, nếu đã, nó xuất ra một chỉ thị **Upgrade(Q)** ngay trước chỉ thị **Write(Q),** nếu chưa, nó xuất ra chỉ thị **Lock-X(Q)** ngay trước **Write(Q).** Tất cả các chốt giao dịch nhận được sẽ được tháo chốt sau khi giao dịch bàn giao hay bỏ dở.

V.1.4. GIAO THỨC DỰA TRÊN ĐỒ THỊ (Graph-Based Protocol)

Ta đã biết, trong trường hợp thiếu vắng các thông tin liên quan đến cách thức các hạng mục dữ liệu được truy xuất, giao thức chốt hai kỳ là cần và đủ để đảm bảo tính khả tuần tự. Nếu ta muốn phát triển các giao thức không là hai kỳ, ta cần các thông tin bổ xung trên cách thức mỗi giao dịch truy xuất CSDL. Có nhiều mô hình khác nhau về lượng thông tin được cung cấp. Mô hình đơn giản nhất đòi hỏi ta phải biết trước thứ tự trong đó các hạng mục dữ liệu sẽ được truy xuất. Với các thông tin như vậy, có thể xây dựng các giao thức chốt không là hai kỳ nhưng vẫn đảm bảo tính khả tuần tự xung đột.

Để có được hiểu biết trước như vậy, ta áp đặt một thứ tự bộ phận, ký hiệu $\rightarrow$, trên tập tất cả các hạng mục dữ liệu $D = \{ d_1 , d_2 , ..., d_n \}$. Nếu $d_i \rightarrow d_j$, bất kỳ giao dịch nào truy xuất cả d_i và d_j phải truy xuất d_i trước khi truy xuất d_j . Thứ tự bộ phận này cho phép xem D như một đồ thị định hướng phi chu trình, được gọi là đồ thị CSDL (DataBase Graph). Trong phần này, để đơn giản, ta hạn chế chỉ xét các đồ thị là các cây và ta sẽ đưa ra một giao thức đơn giản, được gọi là giao thức cây (tree protocol), giao thức này hạn chế chỉ dùng các chốt exclusive.

Trong giao thức cây, chỉ cho phép chỉ thị chốt **Lock-X**, mỗi giao dịch T có thể chốt một hạng mục dữ liệu nhiều nhất một lần và phải tuân theo các quy tắc sau:

1. Chốt đầu tiên bởi T có thể trên bất kỳ hạng mục dữ liệu nào
2. Sau đó, một hạng mục dữ liệu Q có thể bị chốt bởi T chỉ nếu cha của Q hiện đang bị chốt bởi T
3. Các hạng mục dữ liệu có thể được tháo chốt bất kỳ lúc nào

4. Một hạng mục dữ liệu đã bị chốt và được tháo chốt bởi T, không thể bị T chốt lại lần nữa.

Các lịch trình hợp lệ trong giao thức cây là khả tuần tự xung đột.

Ví dụ: Cây CSDL là:

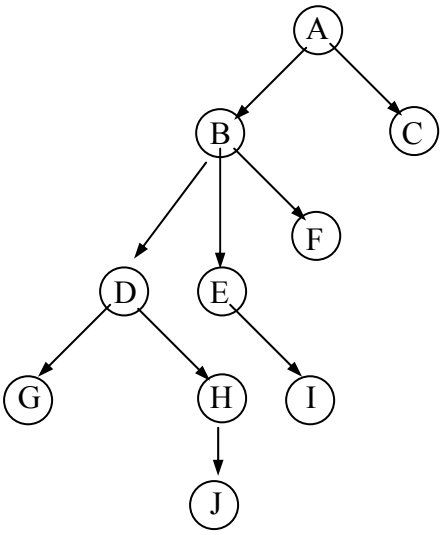


figure V- 10

Chỉ các chỉ thị chốt và tháo chốt của các giao dịch được trình bày:

T_{10} : **Lock-X(B); Lock-X(E); Lock-X(D); Unlock(B); Unlock(E); Lock-X(G); Unlock(D); Unlock(G).**

T_{11} : **Lock-X(D); Lock-X(H); Unlock(D); Unlock(H).**

T_{12} : **Lock-X(B); Lock-X(E); Unlock(B); Unlock(E).**

T_{13} : **Lock-X(D); Lock-X(H); Unlock(D); Unlock(H).**

figure V- 11

Một lịch trình tuân theo giao thức cây chứa tất cả bốn giao dịch trên được cho trong hình bên dưới. Ta nhận thấy, các lịch trình tuân thủ giao thức cây không chỉ là khả tuần tự xung đột mà còn đảm bảo không có dealock. Giao thức cây có mặt thuận lợi so với giao thức hai kỳ là tháo chốt có thể xảy ra sớm hơn. Việc tháo chốt sớm có thể dẫn đến rút ngắn thời gian chờ đợi và tăng tính cạnh tranh. Hơn nữa, do giao thức là không dealock, nên không có cuộn lại. Tuy nhiên giao thức cây có điểm bất lợi là, trong một vài trường hợp, một giao dịch có thể phải chốt những hạng mục dữ liệu mà nó không truy xuất. Chẳng hạn, một giao dịch cần truy xuất các hạng mục dữ liệu A và J trong đồ thị CSDL trên, phải chốt không chỉ A và J mà phải chốt cả các hạng mục B, D, H. Việc chốt bổ xung này có thể gây ra việc tăng tổng phí chốt, tăng thời gian chờ đợi và giảm tính cạnh tranh. Hơn nữa, nếu không biết trước các hạng mục dữ liệu nào sẽ cần thiết phải chốt, các giao dịch sẽ phải chốt gốc của cây mà điều này làm giảm mạnh tính cạnh tranh.

Đối với một tập các giao dịch, có thể có các lịch trình khả tuần tự xung đột không thể nhận được từ việc tuân theo giao thức cây. Có các lịch trình được sinh ra bởi tuân theo giao thức chốt hai kỳ nhưng không thể được sinh ra bởi tuân theo giao thức cây và ngược lại.

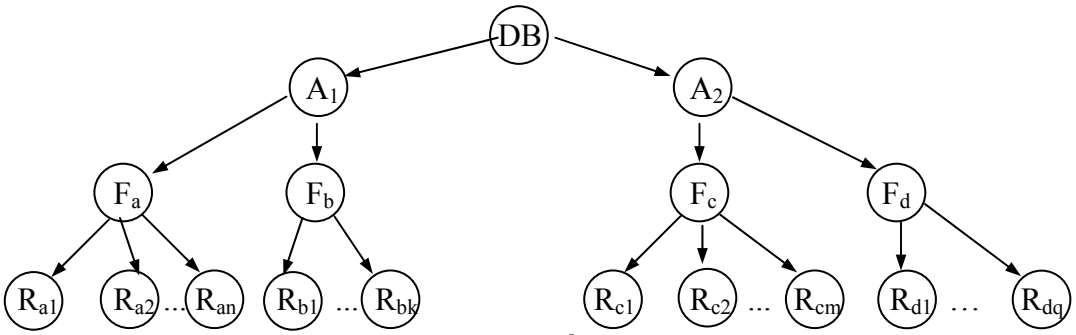
T_{10}	T_{11}	T_{12}	T_{13}
Lock-X(B)			
	Lock-X(D)		
	Lock-X(H)		
	Unlock(D)		
Lock-X(E)			
Lock-X(D)			
Unlock(B)			
Unlock(E)			
		Lock-X(B)	
		Lock-X(E)	
	Unlock(H)		
Lock-X(G)			
Unlock(D)			
			Lock-X(D)
			Lock-X(H)
			Unlock(D)
			Unlock(H)
		Unlock(E)	
		Unlock(B)	
Unlock(G)			

Lịch trình khả tuần tự dưới giao thức cây

figure V- 12

V.1.5. ĐA HẠT (Multiple Granularity)

Trong các sơ đồ điều khiển cạnh tranh được mô tả trước đây, ta đã sử dụng hạng mục dữ liệu như đơn vị trên nó sự đồng bộ hoá được thực hiện. Tuy nhiên, có các hoàn cảnh trong đó việc nhóm một vài hạng mục dữ liệu và xử lý chúng như một đơn vị đồng bộ hoá mang lại nhiều lợi ích. Nừu một giao dich T_i phải truy xuất toàn bộ CSDL và giao thức chốt được sử dụng, khi đó T_i phải chốt mỗi hạng mục dữ liệu trong CSDL. Như vậy việc thực hiện các chốt này sẽ tiêu tốn một thời gian đáng kể. Sẽ hiệu quả hơn nếu T_i chỉ cần một yêu cầu chốt để chốt toàn bộ CSDL. Mặt khác, nếu T_i cần truy xuất chỉ một vài hạng mục dữ liệu, nó không cần thiết phải chốt toàn bộ CSDL vì như vậy sẽ giảm tính cạnh tranh. Như vậy, cái mà ta cần là một cơ chế cho phép hệ thống xác định nhiều mức hạt. Một cơ chế như vậy là cho phép các hạng mục dữ liệu có kích cỡ khác nhau và xác định một sự phân cấp các hạt dữ liệu, trong đó các hạt nhỏ được ẩn náu bên trong các hạt lớn. Sự phân cấp như vậy có thể được biểu diễn đồ thị như một cây. Một nút không là lá của cây đa hạt biểu diễn dữ liệu được kết hợp với con cháu của nó. Như một ví dụ minh hoạ, ta xét cây sau:



Phân cấp hạt

figure V- 13

Nó gồm bốn mức nút. Mức cao nhất là toàn bộ CSDL. Thấp hơn là các nút kiểu vùng: CSDL bao gồm các vùng này. Mỗi vùng lại có các nút kiểu file như các con của nó, mỗi vùng chứa đúng các file này và không file nào nằm trong nhiều hơn một vùng. Cuối cùng, mỗi file có các nút con kiểu mẩu tin, không mẩu tin nào hiện diện trong hơn một file.

Mỗi nút trong cây có thể được chốt một các cá nhân. Như đã làm trong giao thức chốt hai kỳ, ta sẽ sử dụng các phương thức chốt **shared** và **exclusive** . Khi một giao dịch chốt một nút, trong phương thức **shared** hoặc **exclusive**, giao dịch cũng chốt tất cả các nút con cháu của nút này ở cùng phương thức. Ví dụ T_i chốt tường minh file F_b ở phương thức exclusive, nó đã chốt ẩn tất cả các mẩu tin của F_b cũng trong phương thức exclusive.

Giả sử giao dịch T_j muốn chốt mẩu tin R_{b6} của file F_b. vì giao dịch T_i đã chốt tường minh file F_b, mẩu tin R_{b6} cũng bị chốt ẩn. Song làm thế nào để hệ thống biết được T_j có thể chốt R_{b6} hay không: T_j phải duyệt cây từ gốc đến mẩu tin R_{b6} , nếu có một nút bất kỳ tren đường dẫn bị chốt ở phương thức không tương thích, T_j phải chờ. Bây giờ, nếu T_k muốn chốt toàn bộ CSDL, nó phải chốt nút gốc. Tuy nhiên, do T_i hiện đang giữ một chốt trên F_b, một bộ phận của cây, nên T_k sẽ không thành công. Vậy làm thế nào để hệ có thể chốt được nút gốc: Một khả năng là tìm kiếm trên toàn bộ cây, giải pháp này phá huỷ hoàn toàn sơ đồ mục đích của sơ đồ chốt đa hạt. Một giải pháp hiệu quả hơn là đưa vào một lớp mới các phương thức chốt, được gọi là phương thức chốt tăng cường (intension lock mode). Nếu một nút bị chốt ở phương thức tăng cường, chốt tường minh được tiến hành ở mức thấp hơn của cây (hạt min hơn). Chốt tăng cường được được đặt trên tất cả các tổ tiên của một nút trước khi nút đó được chốt tường minh. Như vậy một giao dịch không cần thiết phải tìm kiếm toàn bộ cây để xác định nó có thể chốt một nút thành công hay không. Một giao dịch muốn chốt một nút, chẳng hạn Q, phải duyệt một đường dẫn từ gốc đến Q, trong khi duyệt cây, giao dịch chốt các nút trên đường đi ở phương thức tăng cường.

Có một phương thức tăng cường kết hợp với phương thức **shared** và một với phương thức **exclusive**. Nếu một nút bị chốt ở phương thức tăng cường shared (IS), chốt tường minh được tiến hành ở mức thấp hơn trong cây, nhưng chỉ là một các chốt phương thức shared. Tương tự, nếu một nút bị chốt ở phương thức tăng cường exclusive (IX), chốt tường minh được tiến hành ở mức thấp hơn với các chốt exclusive hoặc shared. Nếu một nút bị chốt ở phương thức shared và phương thức tăng cường exclusive (SIX), cây con có gốc là nút này bị chốt tường minh ở phương thức shared và chốt tường minh được tiến hành ở mức thấp hơn với các chốt exclusive. Hàm tính tương thích đối với các phương thức chốt này được cho bởi ma trận:

	IS	IX	S	SIX	X
IS	True	True	True	True	False
IX	True	True	False	False	False
S	True	False	True	False	False
SIX	True	False	False	False	False
X	False	False	False	False	False

figure V- 14

Giao thức chốt đa hạt dưới đây đảm bảo tính khả tuần tự. Mỗi giao dịch T có thể chốt một nút Q theo các quy tắc sau:

1. Hàm tương thích chốt phải được kiểm chứng
2. Gốc của cây phải được chốt đầu tiên, và có thể được chốt ở bất kỳ phương thức nào
3. Một nút Q có thể được chốt bởi T ở phương thức S hoặc IS chỉ nếu cha của Q hiện đang bị chốt bởi T ở hoặc phương thức IX hoặc phương thức IS.
4. Một nút Q có thể được chốt bởi T ở phương thức X, SIX hoặc IX chỉ nếu cha của Q hiện đang bị chốt ở hoặc phương thức IX hoặc phương thức SIX
5. T có thể chốt một nút chỉ nếu trước đó nó chưa tháo chốt một nút nào.
6. T có thể tháo chốt một nút Q chỉ nếu không con nào của Q hiện đang bị chốt bởi T

Ta thấy rằng giao thức đa hạt yêu cầu các chốt được tậu theo thứ tự Top-Down, được tháo theo thứ tự Bottom-Up.

Ví dụ: Xét cây phân cấp hạt như trên và các giao dịch sau:

- Giả sử giao dịch T_{18} đọc mẩu tin R_{a2} của file F_a. Khi đó T_{18} cần phải chốt DB, vùng A_1 và F_a ở phương thức IS và R_{a2} ở phương thức S: **Lock-IS(DB); Lock-IS(A_1); Lock-IS(F_a); lock-S(R_{a2})**
- Giả sử giao dịch T_{19} sửa đổi mẩu tin R_{a9} trong file F_a , khi đó T_{19} cần phải chốt CSDL, vùng A_1 và file F_a ở phương thức IX và R_{a9} ở phương thức X: **Lock-IX(DB); Lock-IX(A_1); lock-IX(F_a); lock-X(R_{a9})**
- Giả sử giao dịch T_{20} đọc tất cả các mẩu tin của file F_a , khi đó T_{20} cần phải chốt CSDL, và vùng A_1 ở phương thức IS và chốt F_a ở phương thức S: **Lock-IS(DB); Lock-IS(A_1); Lock-S(F_a)**
- Giả sử giao dịch T_{21} đọc toàn bộ CSDL, nó có thể làm điều đó sau khi chốt CSDL ở phương thức S: **Lock-S(DB)**

Chú ý rằng T_{18}, T_{20} và T_{21} có thể truy xuất đồng thời CSDL, giao dịch T_{19} có thể thực hiện cạnh tranh với T_{18} nhưng không với T_{20} hoặc T_{21}

V.2. GIAO THỨC DỰA TRÊN TEM THỜI GIAN (Timestamp-based protocol)

V.2.1. TEM THỜI GIAN (Timestamp)

Ta kết hợp với mỗi giao dịch T_i trong hệ thống một tem thời gian cố định duy nhất, được biểu thị bởi $TS(T_i)$. Tem thời gian này được gán bởi hệ CSDL trước khi giao dịch T_i bắt đầu thực

hiện. Nếu một giao dịch T_i đã được gán tem thời gian $TS(T_i)$ và một giao dịch mới T_j đi vào hệ thống, khi đó $TS(T_i) < TS(T_j)$. Có hai phương pháp đơn giản để thực hiện sơ đồ này:

1. Sử dụng giá trị của đồng hồ hệ thống như tem thời gian: Một tem thời gian của một giao dịch bằng giá trị của đồng hồ khi giao dịch đi vào hệ thống.
2. Sử dụng bộ đếm logic: bộ đếm được tăng lên mỗi khi một tem thời gian đã được gán, tem thời gian của một giao dịch bằng với giá trị của bộ đếm khi giao dich đi vào hệ thống.

Tem thời gian của các giao dịch xác định thứ tự khả tuần tự. Như vậy, *nếu $TS(T_i) < TS(T_j)$, hệ thống phải đảm bảo rằng lịch trình được sinh ra là tương đương với một lịch trình tuần tự trong đó T_i xuất hiện trước T_j* .

Để thực hiện sơ đồ này, ta kết hợp với mỗi hạng mục dữ liệu Q hai giá trị tem thời gian:

- **W-timestamp(Q)** biểu thị tem thời gian lớn nhất của giao dịch bất kỳ đã thực hiện **Write(Q)** thành công
- **R-timestamp(Q)** biểu thị tem thời gian lớn nhất của giao dịch bất kỳ đã thực hiện **Read(Q)** thành công

Các tem thời gian này được cập nhật mỗi khi một **Write** hoặc một **Read** mới được thực hiện.

V.2.2. GIAO THỨC THỨ TỰ TEM THỜI GIAN (Timestamp-Ordering Protocol)

Giao thức thứ tự tem thời gian đảm bảo rằng các **Write** và **Read** xung đột bất kỳ được thực hiện theo thứ tự tem thời gian. Giao thức này hoạt động như sau:

1. Giả sử giao dịch T_i phát ra **Read(Q)**.
   a. Nếu $TS(T_i) <$ W-Timestamp(Q), T_i cần đọc một giá trị của Q đã được viết rồi. Do đó, hoạt động **Read** bị vứt bỏ và T_i bị cuộn lại.
   b. Nếu $TS(T_i) \geq$ W-Timestamp(Q), hoạt động **Read** được thực hiện và R-Timestamp được đặt bằng giá trị lớn nhất trong hai giá trị R-Timestamp và $TS(T_i)$.
2. Giả sử giao dịch T_i phát ra **Write(Q)**.
   a. Nếu $TS(T_i) <$ R-Timestamp(Q), Giá trị của Q mà T_i đang sinh ra được giả thiết là để được dùng cho các giao dịch đi sau nó (theo trình tự thời gian), nhưng nay không cần đến nữa. Do vậy, hoạt động **Write** này bị vứt bỏ và T_i bị cuộn lại
   b. Nếu $TS(T_i) <$ W-Timestamp(Q), T_i đang thử viết một giá trị đã quá hạn của Q, Từ đó, hoạt động **Write** bị vứt bỏ và T_i bị cuộn lại
   c. Ngoài ra, hoạt động **Write** được thực hiện và W-Timestamp(Q) được đặt là $TS(T_i)$

Một giao dịch T_i bị cuộn lại bởi sơ đồ điều khiển cạnh tranh như kết quả của hoạt động **Read** hoặc **Write** đang được phát ra, được gán với một tem thời gian mới và được tái khởi động lại (được xem như một giao dịch mới tham gia vào hệ thống)

Ta xét các giao dịch T_{14} và T_{15} được xác định như dưới đây:

T_{14} : **Read(B);**
Read(A);
Display(A+B);.

T_{15} : **Read(B);**
B:=B-50;
Write(B);
Read(A);
A:=A+50;
Write(A);
Display(A+B).

figure V- 15

Ta giả thiết rằng một giao dịch được gán cho một tem thời gian ngay trước chỉ thị đầu tiên của nó. Như vậy, lịch trình schedule-3 dưới đây có $TS(T_{14}) < TS(T_{15})$, và là một lịch trình hợp lệ dưới giao thức tem thời gian:

T_{14}	T_{15}
Read(B)	
	Read(B)
	B:=B-50
	Write(B)
Read(A)	
	Read(A)
Display(A+B)	
	A:=A+50
	Write(A)

Schedule-3

figure V- 16

Giao thức thứ tự tem thời gian đảm bảo tính khả tuần tự xung đột và không dealock.

V.2.3. QUY TẮC VIẾT THOMAS (Thomas' Write rule)

Một biến thể của giao thức tem thời gian cho phép tính cạnh tranh cao hơn giao thức thứ tự tem thời gian. Trước hết ta xét lịch trình schedule-4 sau:

T_{16}	T_{17}
Read(Q)	
	Write(Q)
Write(Q)	

Schedule-4

figure V- 17

Nếu áp dụng giao thức thứ tự tem thời gian, ta có $TS(T_{16}) < TS(T_{17})$. Hoạt dộng **Read(Q)** của T_{16} và **Write(Q)** của T_{17} thành công, khi T_{16} toan thực hiện hoạt động **Write(Q)** của nó, vì $TS(T16) < TS(T_{17}) =$ W-timestamp(Q), nên **Write(Q)** của T_{16} bị vứt bỏ và giao dịch T_{16} bị cuộn lại. Sự cuộn lại này là không cần thiết. Nhận xét này cho ta một sửa đổi phiên bản giao thức thứ tự tem thời gian:

Các quy tắc giao thức đối với **Read** không thay đổi, các quy tắc đối với **Write** được thay đổi chút ít như sau:

Giả sử giao dịch T_i phát ra **Write(Q).**

1. Nếu $TS(T_i) <$ R-timestamp(Q), Giá trị của Q mà T_i đang sinh ra được giả thiết là để được dùng cho các giao dịch đi sau nó (theo trình tự thời gian), nhưng nay không cần đến nữa. Do vậy, hoạt động **Write** này bị vứt bỏ và T_i bị cuộn lại.
2. Nếu $TS(T_i) <$ W-timestamp(Q), T_i đang thử viết một giá trị lỗi thời của Q. Do vậy, hoạt động **Write** này có thể bị bỏ lơ (không được thực hiện, nhưng T_i không bị cuộn lại).
3. Ngoài ra, hoạt động **Write** được thực hiện và W-timestamp(Q) được đặt là $TS(T_i)$.

Sự sửa đổi đối với giao thức thứ tự tem thời gian này được gọi là quy tắc viết Thomas. Quy tắc viết Thomas cho khả năng sinh các lịch trình khả tuần tự mà các giao thức thức trước đây không thể.

V.2.4. GIAO THỨC DỰA TRÊN TÍNH HỢP LỆ

Trong trường hợp đa số các giao dịch trong hệ thống là các giao dịch chỉ đọc (read-only), tỷ suất xung đột giữa các giao dịch là thấp. Như vậy nhiều giao dịch trong chúng thực hiện thiếu sự giám sát của sơ đồ điều khiển cạnh tranh cũng vẫn giữ cho hệ thống ở trạng thái nhất quán. Hơn nữa, một sơ đồ điều khiển cạnh tranh đưa vào một tổng phí đáng kể (cho thực hiện mã lệnh, thời gian chờ của giao dịch ...). Việc tìm một sơ đồ với tổng phí nhỏ là một mục tiêu. Nhưng khó khăn là ta phải biết trước những giao dịch sẽ bị dính líu vào một xung đột. Để có được các hiểu bét đó, ta cần một sơ đồ để giám sát hệ thống.

Ta giả thiết rằng mỗi giao dịch T_i , trong thời gian sống của nó, thực hiện trong hai hoặc ba kỳ khác nhau, phụ thuộc vào nó là một giao dịch chỉ đọc hay là một giao dịch cập nhật. Các kỳ này, theo thứ tự, là như sau:

1. **Kỳ đọc.** Trong kỳ này, các giá trị của các hạng mục dữ liệu khác nhau được đọc vào các biến cục bộ của T_i . Tất cả các hoạt động **Write** được thực hiện trên các biến cục bộ tạm, không cập nhật CSDL hiện hành.
2. **Kỳ hợp lệ.** Giao dịch T_i thực hiện một phép kiểm thử sự hợp lệ để xác định xem nó có thể sao chép đến CSDL các biến cục bộ tạm chứa các kết quả của các hoạt **Write** mà không vi phạm tính khả tuần tự xung đột hay không.

3. **Kỳ viết.** Nếu T_i thành công trong kỳ hợp lệ, các cập nhật hiện hành được áp dụng vào CSDL, nếu không T_i bị cuộn lại.

Mỗi giao dịch phải trải qua ba kỳ theo thứ tự trên, tuy nhiên, ba kỳ của các giao dịch đang thực hiện cạnh tranh có thể đan xen nhau.

Các kỳ đọc và kỳ viết tự nó đã rõ ràng. Chỉ có kỳ hợp lệ là cần thảo luận thêm. Để thực hiện kiểm thử sự hợp lệ, ta cần biết khi nào các kỳ khác nhau của giao dịch T_i xảy ra. Do vậy, ta sẽ kết hợp ba tem thời gian với giao dịch T_i :

1. **Start(T_i).** Thời gian khi T_i bắt đầu sự thực hiện.
2. **Validation(T_i).** Thời gian khi T_i kết thúc kỳ đọc và khởi động kỳ hợp lệ.
3. **Finish(T_i).** Thời gian khi T_i kết thúc kỳ viết.

Ta xác định thứ tự khả tuần tự bởi kỹ thuật thứ tự tem thời gian sử dụng giá trị tem thời gian **Validation(T_i).** Như vậy, giá trị $TS(T_i) = Validation(T_i)$ và nếu $TS(T_j) < TS(T_k)$ thì bất kỳ lịch trình nào được sinh ra phải tương đương với một lịch trình tuần tự trong đó giao dịch T_i xuất hiện trước giao dịch T_k . Lý dota chọn $Validation(T_i)$ như tem thời gian của T_i , mà không chọn $Start(T_i)$, là vì ta hy vọng thời gian trả lời sẽ nhanh hơn.

Phép kiểm thử hợp lệ đối với T_j đòi hỏi rằng, đối với mỗi giao dịch T_i với $TS(T_i) < TS(T_j)$, một trong các điều kiện sau phải được thoả mãn:

1. **Finish(T_i) < Start(T_j).** Do T_i hoàn tất sự thực hiện của nó trước khi T_j bắt đầu, thứ tự khả tuần tự được duy trì.
2. **Tập các hạng mục dữ liệu được viết bởi T_i không giao với tập các hạng mục dữ liệu được đọc bởi T_j và T_i hoàn tất kỳ viết của nó trước khi T_j bắt đầu kỳ hợp lệ (Start(T_j) < Finish(T_i) < Validation(T_j)).** Điều kiện này đảm bảo rằng các viết của T_i và T_j là không chồng chéo. Do các viết của T_i không ảnh hưởng tới đọc của T_j và do T_j không thể ảnh hưởng tới đọc của T_i , thứ tự khả tuần tự được duy trì.

Lịch trình schedule-5 cho ta một minh hoạ về giao thức dựa trên tính hợp lệ:

T_{14}	T_{15}
Read(B)	
	Read(B)
	B:=B-50
	Read(A)
	A:=A+50
Read(A)	
Xác nhận tính hợp lệ	
Display(A+B)	
	Xác nhận tính hợp lệ
	Write(B)
	Write(A)

figure V- 18

Sơ đồ hợp lệ tự động canh chừng việc cuộn lại hàng loạt, do các **Write** hiện tại xảy ra chỉ sau khi giao dịch phát ra **Write** đã bàn giao.

V.3. CÁC SƠ ĐỒ ĐA PHIÊN BẢN (Multiversion Schemes)

Các sơ đồ điều khiển cạnh tranh được thảo luận trước đây đảm bảo tính khả tuần tự hoặc bởi làm trễ một hoạt động hoặc bỏ dở giao dịc đã phát ra hoạt động đó. Chẳng hạn, một hoạt động **Read** có thể bị làm trễ vì giá trị thích hợp còn chưa được viết hoặc nó có thể bị vứt bỏ vì giá trị mà nó muốn đọc đã bị viết đè rồi. Các khó khăn này có thể được che đi nếu bản sao cũ của mỗi hạng mục dữ liệu được giữ trong một hệ thống.

Trong các hệ CSDL đa phiên bản, mỗi hoạt động **Write(Q)** tạo ra một bản mới của Q. Khi một hoạt động **Read(Q)** được phát ra, hệ thống chọn lựa một trong các phiên bản của Q để đọc. Sơ đồ điều khiển cạnh tranh phải đảm bảo rằng việc chọn lựa này được tiến hành sao cho tính khả tuần tự được đảm bảo. Do lý do hiệu năng, một giao dịch phải có khả năng xác định dễ dàng và mau chóng phiên bản dhạng mục dữ liệu sẽ đọc.

V.3.1. THỨ TỰ TEM THỜI GIAN ĐA PHIÊN BẢN

Kỹ thuật chung được dùng trong các sơ đồ đa phiên bản là tem thời gian. Ta kết hợp với một giao dịch một tem thời gian tĩnh duy nhất, ký hiệu $TS(T_i)$. Tem thời gian này được gán trước khi khi giao dịch bắt đầu sự thực hiện. Mỗi hạng mục dữ liệu Q kết hợp với một dãy $<Q_1, Q_2, ... , Q_m>$ mỗi phiên bản Q_k chứa ba trường dữ liệu:

- **Content** là giá trị của phiên bản Q_i
- **W-timestamp(Q_k)** là tem thời gian của giao dịch đã tạo ra phiên bản Q_k
- **R-timestamp(Q_k)** là tem thời gian lớn nhất của giao dịch đã đọc thành công phiên bản Q_k

Một giao dịch, gọi là T_i , tạo ra phiên bản Q_k của hạng mục dữ liệu Q bằng cách phát ra một hoạt động **Write(Q)**. Trường **Content** của phiên bản chứa giá trị được viết bởi T_i . **W-timestamp** và **R-timestamp** được khởi động là $TS(T_i)$. Giá trị **R-timestamp** được cập nhật mỗi khi một giao dịch T_j đọc nội dung của Q_k và **R-timestamp(Q_k)** < $TS(T_j)$.

Sơ đồ tem thời gian đa phiên bản dưới đây sẽ đảm bảo tính khả tuần tự. Sơ đồ hoạt động như sau: giả sử T_j phát ra một hoạt động **Read(Q)** hoặc **Write(Q).** Q_k ký hiệu phiên bản của Q, tem thời gian viết của nó là tem thời gian viết lớn nhất nhỏ hơn hoặc bằng $TS(T_j)$.

1. Nếu giao dịch T_j phát ra một **Read(Q)**, khi đó giá trị trả lại là nội dung của phiên bản Q_k
2. Nếu T_j phát ra một **Write(Q)** và nếu $TS(T_j)$ < **R-timestamp(Q_k)** khi đó giao dịch T_j bị cuộn lại. Nếu không, nếu $TS(T_j)$ = **W-timestamp(Q_k)** nội dung của Q_k bị viết đè, khác đi một phiên bản mới của Q được tạo.

Các phiên bản không còn được dùng đến nữa bị xoá đi dựa trên quy tắc sau: Giả sử có hai phiên bản Q_i và Q_j của một hạng mục dữ liệu và cả hai phiên bản này cùng có W-timestamp nhỏ hơn tem thời gian của giao dịch già nhất trong hệ thống, khi đó phiên bản già hơn trong hai phiên bản Q_i và Q_j sẽ không còn được dùng nữa và bị xoá đi.

Sơ đồ thứ tự tem thời gian đa phiên bản có tính chất hay đó là một yêu cầu **Read** không bao giờ thất bại và không phải chờ đợi. Trong một hệ thống mà hoạt động **Read** xảy ra nhiều hơn **Write** cái lợi này là đáng kể. Tuy nhiên có vài điều bất lợi của sơ đồ này là: thứ nhất đọc một

hạng mục dữ liệu cũng đòi hỏi cập nhật trường **R-timestamp**, thứ hai là xung đột giữa các giao dịch được giải quyết bằng cuộn lại.

V.3.2. CHỐT HAI KỲ ĐA PHIÊN BẢN

Giao thức chốt hai kỳ đa phiên bản cố gắng tổ hợp những ưu điểm của điều khiển cạnh tranh với các ưu điểm của chốt hai kỳ. Giao thức này phân biệt các giao dịch ***chỉ đọc*** và các giao dịch ***cập nhật***.

Các giao dịch ***cập nhật*** thực hiện chốt hai kỳ nghiêm khắc (các chốt được giữ đến tận khi kết thúc giao dịch). Mỗi hạng mục dữ liệu có một tem thời gian. Tem thời gian trong trường hợp này không là tem thời gian dựa trên đồng hồ thực mà là một bộ đếm, sẽ được gọi là TS-counter.

Các giao dịch ***chỉ đọc*** được gán tem thời gian là giá trị hiện hành của TS-counter trước khi chúng bắt đầu sự thực hiện: chúng tuân theo giao thức thứ tự tem thời gian đa phiên bản để thực hiện đọc. Như vậy, khi một giao dịch chỉ đọc T_i phát ra một **Read(Q),** giá trị trả lại là nội dung của phiên bản mà tem thời gian của nó là tem thời gian lớn nhất nhỏ hơn $TS(T_i)$.

Khi một giao dịch ***cập nhật*** đọc một hạng mục, nó tậu một chốt **shared** trên hạng mục, rồi đọc phiên bản mới nhất của hạng mục (đối với nó). Khi một giao dịch cập nhật muốn viết một hạng mục, đầu tiên nó tậu một chốt exclusive trên hạng mục này, rồi tạo ra một phiên bản mới cho hạng mục. **Write** được thực hiện trên phiên bản mới này và tem thời gian của phiên bản mới được khởi động là $+\infty$.

Khi một giao dịch cập nhật T_i hoàn tất các hoạt động của nó, nó thực hiện xử lý bàn giao như sau: Đầu tiên, T_i đặt tem thời gian trên mỗi phiên bản nó đã tạo là TS-counter+1; sau đó T_i tăng TS-counter lên 1. Chỉ một giao dịch cập nhật được phép thực hiện xử lý bàn giao tại một thời điểm.

Các phiên bản bị xoá cùng kiểu cách với thứ tự tem thời gian đa phiên bản.

V.4. QUẢN TRỊ DEADLOCK

Một hệ thống ở trạnh 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 tạp hợp. Chính xác hơn, tồn tại một tập các giao dịch $\{ T_0 , T_2 , ..., T_n \}$ sao cho T_0 đang chờ một hạng mục dữ liệu được giữ bởi T_1 , T_1 đang chờ một hạng mục dữ liệu đang bị chiếm bởi T_2 , ..., T_{n-1} đang chờ một hạng mục dữ liệu được giữ bởi T_n và T_n đang chờ một hạng mục T_0 đang chiếm. Không một giao dịch nào có thể tiến triển được trong tình huống như vậy. Một cách chữa trị là viện dẫn một hành động tẩy rửa, chẳng hạn 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. Giao 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.

V.4.1. PHÒNG NGỪA DEADLOCK (Deadlock prevention)

Có hai cách tiếp cận phòng ngừa deadlock: Một đảm bảo không có chờ đợi vòng tròn xảy ra bằng cách sắp thứ tự các yêu cầu chốt hoặc đòi hỏi tất cả các chốt được tậu cùng nhau. Một

cách tiếp cận khác gần hơn với khắc phục deadlock và thực hiện cuộn lại thay vì chờ đợi một chốt. Chờ đợi là tiềm ẩn của deadlock.

Sơ đồ đơn giản nhất dưới cách tiếp cận thứ nhất đòi hỏi mỗi giao dịch chốt tất cả các hạng mục dữ liệu trước khi nó bắt đầu thực hiện. Hơn nữa, hoặc tất cả được chốt trong một bước hoặc không hạng mục nào được chốt. Giao thức này có hai bất lợi chính: một là khó dự đoán, trước khi giao dịch bắt đầu, các hạng mục dữ liệu nào cần được chốt, hai là hiệu suất sử dụng hạng mục dữ liệu rất thấp do nhiều hạng mục có thể bị chốt nhưng không được sử dụng trong một thời gian dài.

Sơ đồ phòng ngừa deadlock khác là áp đặt một thứ tự bộ phận trên tất cả các hạng mục dữ liệu và yêu cầu một giao dịch chốt một hạng mục dữ liệu theo thứ tự được xác định bởi thứ tự bộ phận này.

Cách tiếp cận thứ hai để phòng ngừa deadlock là sử dụng ưu tiên và cuộn lại quá trình. Với ưu tiên, một giao dịch T_2 yêu cầu một chốt bị giữ bởi giao dịch T_1 , chốt đã cấp cho T_1 có thể bị lấy lại và cấp chgo T_2 , T_1 bị cuộn lại. Để điều khiển ưu tiên, ta gán một tem thời gian duy nhất cho mỗi giao dịch. Hệ thống sử dụng các tem thời gian này để quyết định một giao dịch phải chờ hay cuộn lại. Việc chốt vẫn được sử dụng để điều khiển cạnh tranh. Nếu một giao dịch bị cuộn lại, nó vẫn giữ tem thời gian cũ của nó khi tái khởi động. Hai sơ đồ phòng ngừa deadlock sử dụng tem 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 hạng mục dữ liệu bị chiếm bởi T_j , T_i được phép chờ chỉ nếu nó có tem 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 hạng mục dữ liệu hiện đang bị giữ bởi T_j , T_i được phép chờ chỉ nếu nó có tem 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 tem thời gian nhỏ nhất. Giao dịch này không thể bị yêu cầu cuộn lại trong cả hai sơ đồ. Do tem thời gian luôn tăng và do các giao dịch không được gán tem thời gian mới khi chúng bị cuộn lại, một giao dịch bị cuộn lại sẽ có tem 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 hạng 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 hạng 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 hạng 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 tậu được hạng 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 hạng mục dữ liệu nó chiếm giữ. Khi T_i khởi động lại, và yêu cầu hạng 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.

V.4.2. 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 chốt phải chờ nhiều nhất một khoảng thời gian xác định. Nếu chốt 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.

V.4.3. PHÁT HIỆN DEADLOCK VÀ KHÔI PHỤC

Nếu một hệ thống không dùng giao 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 xẩy ra hay không. Nếu có, hệ thống phải khôi phục lại từ deadlock, muốn vậy hệ thống phải:

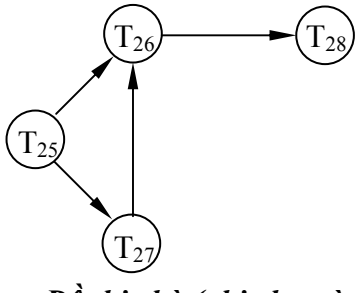
- Duy trì thông tin về sự cấp phát hiện hành các hạng mục dữ liệu cho các giao dịch cũng như các yêu cầu hạng mục dữ liệu chưa được chưa được giải quyết.
- Cung cấp một thuật toán sử dụng các thông tin này để xác định hệ thống đã đi vào trạng thái deadlock chưa.
- Phục hồi từ deadlock khi phát hiện được deadlock đã xảy ra.

V.4.3.1 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 hạng mục dữ liệu nó cần.

Khi giao dịch T_i yêu cầu một hạng mục dữ liệu đang bị giữ bởi giao dịch T_j khi đó cung $T_i \rightarrow T_j$ được xen vào đồ thị. Cạnh này bị xoá đi chỉ khi giao dịch T_j không còn giữ hạng 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)

figure V- 19

Do đồ thị không có chu trình nên hệ thông không trong trạng thái deadlock. Bây giờ, giả sử T_{28} yêu cầu một hạng mục dữ liệu được giữ bởi T_{27} , cung $T_{28} \rightarrow T_{27}$ được xen vào đồ thị, điều này dẫn đến tồn tại một chu trình $T_{26} \rightarrow T_{27} \rightarrow T_{28} \rightarrow T_{26}$ có nghĩa là hệ thống rơi vào tình trạng deadlock và T_{26} , T_{27} , T_{28} bị deadlock.

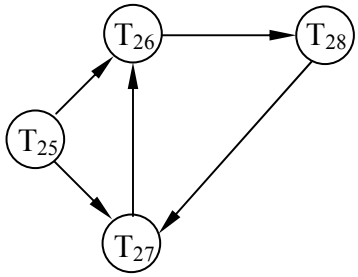


figure V- 20

Vấn đề đặt ra là khi nào thì chạy thủ tục phát hiện ? câu trả lời phụ thuộc hai yêu tố sau:

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 hạng 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.

V.4.3.2 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 hạng mục dữ liệu
    c. Giao dịch cần bao nhiêu hạng 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 xung 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.

V.5. BÀI TẬP CHƯƠNG V

V.1 Chứng tỏ rằng giao thức chốt hai kỳ đảm bảo tính khả tuần tự xung đột và các giao dịch có thể được làm tuần tự tương ứng với các điểm chốt của chúng.

V.2 Xét hai giao dịch sau:

T_{31} : **Read(A);**
Read(B);
If A=0 then B:=B+1;
Write(B);

T_{32} : **Read(B);**
Read(A);
If B=0 then A:=A+1;
Write(A);

Thêm các chỉ thị chốt và tháo chốt vào hai giao dịch T_{31} và T_{32} sao cho chúng tuân theo giao thức chốt hai kỳ. Sự thực hiện các giao dịch này có thể dẫn đến deadlock ?

V.3 Nêu các ưu điểm và nhược điểm của giao thức chốt hai kỳ nghiêm ngặt.

V.4 Nêu các ưu điểm và nhược điểm của giao thức chốt hai kỳ nghiêm khắc.

V.5 Bộ quản trị điều khiển cạnh tranh của một hệ CSDL phải quyết định cấp cho một đòi hỏi chốt hoặc bắt giao dịch yêu cầu phải chờ. Hãy thiết kế một cấu trúc dữ liệu (tiết kiệm không gian) cho phép bộ quản trị điều khiển cạnh tranh cho ra quyết định mau chóng.

V.6 Xét sự mở rộng thành giao thức cây-chốt sau. Nó cho phép cả chốt shared và exclusive:

1. Một giao dịch chỉ đọc chỉ có thể yêu cầu các chốt shared. Một giao dịch cập nhật chỉ có thể yêu cầu các chốt exclusive
2. Mỗi giao dịch phải tuân theo các quy tắc của giao thức cây-chốt. Các giao dịch chỉ đọc đầu tiên có thể chốt bất kỳ hạng mục dữ liệu nào, các giao dịch cập nhật đầu tiên phải chốt gốc.

   Chứng tỏ giao thức đảm bảo tính khả tuần tự và không có deadlock

V.7 Cho ra ví dụ về các lịch trình có thể dưới giao thức cây nhưng không thể dưới giao thức chốt hai kỳ và ngược lại.

V.8 Xét một biến thể của giao thức cây, được gọi là giao thức rừng. CSDL được tổ chức như một rừng. Mỗi giao dịch T phải tuân theo các quy tắc sau:

1. Chốt đầu tiên trong mỗi cây có thể trên bất kỳ hạng mục dữ liệu nào
2. Chốt từ thứ hai trở về sau có thể được yêu cầu chỉ nếu cha của nút được yêu cầu hiện đang bị chốt
3. Các hạng mục dữ liệu có thể được tháo chốt bất kỳ lúc nào
4. Một hạng mục dữ liệu không được chốt lại bởi T sau khi T đã tháo chốt cho nó

   Chứng tỏ giao thức rừng không đảm bảo tính khả tuần tự

V.9 Khi một giao dịch bị cuộn lại dưới thứ tự tem thời gian, nó được gán một tem thời gian mới. Tại sao không đơn giản để nó giữ lại tem thời gian cũ ?

V.10 Trong chốt đa hạt, sự khác nhau giữa chốt tường minh và chốt ẩn là gì ?

V.11 phương thức chốt SIX là hữu dụng trong chốt đa hạt. Phương thức exclusive và shared tăng cường không được sử dụng. Tại sao nó lại vô dụng ?

V.12 Đối với mỗi một trong các giao thức sau, mô tả sắc thái ứng dụng thực tế gợi ý sử dụng giao thức và sắc thái gợi ý không nên sử dụng giao thức:

1. Chốt hai kỳ
2. Chốt hai kỳ với chốt đa hạt
3. Giao thức cây
4. Thứ tự tem thời gian
5. Hợp lệ
6. Thứ tự tem thời gian đa phiên bản
7. Chốt hai kỳ đa phiên bản

V.13 Chứng tỏ rằng có những lịch trình là có thể dưới giao thức chốt hai kỳ nhưng không thể dưới giao thức tem thời gian và ngược lại.

V.14 Với điều kiện nào tránh deadlock ít đắt giá hơn cho phép deadlock rồi phát hiện?

CHƯƠNG VI
HỆ THỐNG PHỤC HỒI
(Recovery system)

MỤC ĐÍCH

Một hệ thống máy tính, cũng giống như các thiết bị cơ - điện khác, luôn có nguy cơ bị hỏng hóc do nhiều nguyên nhân hư đĩa, mất nguồn, lỗi phần mềm v..v... Điều này dẫn đến hậu quả là sự mất thông tin. Vì vậy, hệ quản trị cơ sở dữ liệu phải có các cơ chế đáp ứng lại nguy cơ hệ thống bị hỏng hóc, nhằm đảm bảo ***tính nguyên tử và tính lâu bền*** của các giao dịch. Chương này trình bày các nguyên lý của một hệ thống phục hồi nhằm khôi phục CSDL đến một trạng thái nhất quán trước khi xảy ra sự cố.

YÊU CẦU

Hiểu rõ các sự cố có thể xảy ra trong đời sống của một cơ sở dữ liệu, các nguyên nhân của sự không nhất quán dữ liệu.

Hiểu các kỹ thuật phục hồi, các ưu nhược điểm của mỗi kỹ thuật.

PHÂN LỚP HỎNG HÓC:

Có nhiều kiểu hỏng hóc có thể xảy đến với hệ thống, mỗi một trong chúng cần được ứng xử một cách riêng biệt. Trong chương này ta chỉ xét các kiểu hỏng hóc sau:

- **Hỏng hóc trong giao dịch**: Có hai loại lỗi làm cho giao dịch bị hỏng hóc: **1. Lỗi luận lý**: Giao dịch không thể tiếp tục thực hiện bình thường được nữa do một số điều kiện bên trong không được thoả. ví dụ như: dữ liệu đầu vào không đúng, không tìm thấy dữ liệu, trào dữ liệu hoặc do việc sử dụng tài nguyên vượt hạn định. **2. Lỗi hệ thống:** Hệ thống rơi vào trạng thái không mong muốn ví dụ như trạng thái deadlock.
- **Hệ thống bị hư hỏng:** Có một phần cứng sai chức năng hoặc có một sai sót trong phần mềm cơ sở dữ liệu hay hệ điều hành.
- **Đĩa bị hư hỏng:** Một khối đĩa bị mất nội dung.

Để hệ thống có thể đề ra được chiến lược phục hồi lỗi phù hợp, trước tiên cần phải xác định các loại hỏng hóc trên các thiết bị lưu trữ dữ liệu. Sau đó, cần xác định những hỏng hóc này ảnh hưởng như thế nào đến nội dung cơ sở dữ liệu. Nhiệm vụ quan trọng sau cùng là đề ra các giải pháp nhằm đảm bảo tính nhất quán của cơ sở dữ liệu và tính nguyên tử của giao dịch mỗi khi hỏng hóc đã phát sinh. Các giải pháp này thường được gọi là các giải thuật phục hồi (recovery algorithms).

Các giải thuật phục hồi gồm có hai phần:

1. Các hành động được thực hiện trong suốt quá trình hoạt động bình thường của giao dịch nhằm đảm bảo có đầy đủ thông tin cho việc phục hồi sau này.
2. Các hành động được thực hiện sau khi lỗi phát sinh. Nhằm khôi phục nội dung của cơ sở dữ liệu trở về một trạng thái trước đó, và trạng thái này thoã mãn được các yêu cầu về tính nhất quán của cơ sở dữ liệu, tính bền và tính nguyên tử của giao dịch .

CẤU TRÚC LƯU TRỮ:

Như đã xét trong chương II, các hạng mục dữ liệu khác nhau của cơ sở dữ liệu có thể được lưu trên nhiều phương tiện lưu trữ khác nhau. Để nắm được cách thức đảm bảo tính nguyên tử và tính lâu bền của một giao dịch, cần phải có cái nhìn sâu hơn về các loại thiết bị lưu trữ dữ liệu và cách thức truy xuất chúng.

CÁC LOẠI LƯU TRỮ:

- **Lưu trữ không ổn định (volatile storage):** Thông tin lưu trong thiết bị lưu trữ không ổn định sẽ bị mất khi hệ thống bị hỏng hóc. Ví dụ của thiết bị lưu trữ không ổn định là: bộ nhớ chính, bộ nhớ cache. Sự truy cập đến các thiết bị lưu trữ không ổn định là cực nhanh. Lý do: một là: do tính chất của bộ nhớ cho phép như vậy; hai là: có thể truy xuất trực tiếp các hạng mục dữ liệu chứa trong nó.
- **Lưu trữ ổn định (nonvolatile storage):** Thông tin lưu trữ trong thiết bị lưu trữ ổn định thường không bị mất khi hệ thống bị sự cố. Tuy nhiên, nguy cơ bản thân thiết bị lưu trữ ổn định bị hỏng vẫn có thể xảy ra. Ví dụ của thiết bị lưu trữ ổn định là: đĩa từ và băng từ. Trong hầu hết các hệ cơ sở dữ liệu, thiết bị lưu trữ ổn định thường được dùng là đĩa từ. Các loại thiết bị lưu trữ ổn định khác được dùng để lưu trữ phòng hờ (back up) dữ liệu.

- **Lưu trữ bền (stable storage):** Theo lý thuyết thì thông tin chứa trong thiết bị lưu trữ bền ***không bao giờ*** bị mất khi hệ thống bị hư hỏng. Tuy nhiên, trong thực tế, ta khó lòng tạo ra được một thiết bị đạt được tính chất lý tưởng như vậy. Chỉ có giải pháp tăng cường độ bền mà thôi.

THỰC THI LƯU TRỮ BỀN:

Tiêu chí để thực hiện việc lưu trữ bền là nhân bản thông tin cần thiết trong một vài phương tiện lưu trữ ổn định khác nhau với các phương thức hỏng hóc độc lập và cập nhật các phiên bản thông tin này một cách có tổ chức, sao cho dù có lỗi xuất hiện trong quá trình chuyển dữ liệu, thông tin vẫn không bị hư hại.

- Các hệ thống RAID đảm bảo rằng việc hỏng hóc của một đĩa không gây sự mất dữ liệu. Dạng thức đơn giản và nhanh nhất của RAID là dùng đĩa gương (mirrored disk). Các dạng thức khác giúp tiết kiệm chi phí, nhưng cái giá phải trả là thời gian đọc ghi chậm hơn.
- Tuy nhiên các hệ thống RAID vẫn không đảm bảo được tính an toàn dữ liệu khi gặp phải tai họa như: cháy nổ, lụt lội. Người ta đề nghị một hệ thống lưu trữ mới an toàn hơn hoạt động theo nguyên tắc sau: Sao lưu dữ liệu sang một vài vị trí địa lý khác nhau thông qua mạng máy tính.

Sau đây là cách thức đảm bảo thông tin lưu trữ không bị lỗi trong quá trình đọc ghi dữ liệu:

Việc chuyển một khối dữ liệu giữa bộ nhớ và đĩa có thể dẫn đến kết quả:

- ***Thành công hoàn toàn***: Thông tin được chuyển đến đích an toàn.
- ***Bị lỗi một phần***: Có lỗi xuất hiện trong quá trình chuyển dữ liệu và khối đích chứa thông tin không đúng.
- ***Bị lỗi hoàn toàn***: Lỗi xuất hiện ngay ở giai đoạn đầu của quá trình truyền dữ liệu. Khối đích giữ nguyên như ban đầu.

Nếu có lỗi xuất hiện trong quá trình truyền dữ liệu, hệ thống phải phát hiện được và thực thi thủ tục phục hồi lỗi. Để làm được như vậy, hệ thống phải duy trì hai khối dữ liệu vật lý cho mỗ khối dữ liệu luận lý. (Trong tình huống dùng hệ thống đĩa gương thì hai khối vật lý này ở cùng một địa điểm, trong tình huống dùng hệ thống sao lưu từ xa, hai khối này ở hai địa điểm khác nhau).

Một thao tác ghi dữ liệu được thực thi như sau:

1. Viết thông tin lên khối vật lý thứ nhất.
2. Khi hành động ghi thứ nhất thành công, tiếp tục ghi phần thông tin trên lên khối vật lý thứ hai.
3. Thao tác ghi được coi là thành công khi thao tác ghi thứ hai thành công.

Trong quá trình phục hồi, từng cặp khối vật lý được kiểm tra:

1. Nếu nội dung của cả hai như nhau và không có lỗi có thể phát hiện, khi đó không cần làm gì thêm.
2. Nếu một trong hai khối có lỗi phát hiện được, khi đó thay thế khối bị lỗi bởi nội dung của khối còn lại.
3. Nếu cả hai khối không có lỗi phát hiện được, nhưng nội dung của chúng khác nhau, thay thế khối thứ nhất bởi khối thứ hai.

Yêu cầu phải so sánh từng cặp khối vật lý một đòi hỏi phải mất nhiều thời gian. Người ta có thể cải thiện tình huống này bằng cách lưu vết những thao tác viết khối trong tiến trình thực thi. Khi phục hồi, chỉ những khối nào thao tác ghi ở trong tiến trình thực thi mới cần được đem so sánh. Giao thức để viết ra một khối đến một site xa tương tự như viết khối trong hệ thống đĩa gương..

TRUY CẬP DỮ LIỆU

Như đã xét trong chương II, hệ cơ sở dữ liệu nằm thường trực trên các thiết bị lưu trữ ổn định (thường là đĩa từ) và thường được phân thành các đơn vị lưu trữ kích thước cố định được gọi là khối (blocks). Khối là đơn vị truyền nhận dữ liệu từ/ra đĩa. Một khối có thể chứa vài hạng mục dữ liệu. Ta giả thiết không có hạng mục dữ liệu nào trải ra trên nhiều hơn một khối.

Các giao dịch nhập (input) thông tin từ đĩa vào bộ nhớ chính và xuất (output) thông tin theo chiều ngược lại. Các thao tác nhập/xuất này được thực hiện theo đơn vị khối. Khối nằm trên đĩa được gọi là khối vật lý (physical block), khối được trữ tạm trong bộ nhớ chính được gọi là khối đệm (buffer block). Vùng bộ nhớ tạm chứa các khối dữ liệu được gọi là vùng đệm đĩa (disk buffer).

Việc di chuyển khối giữa đĩa và bộ nhớ được thực hiện thông qua hai thao tác:

1. **Input(B)** chuyển khối vật lý B vào bộ nhớ chính.
2. **Output(B)** chuyển khối đệm B ra đĩa và thay thế cho khối vật lý tương ứng ở đó.

Hình dưới đây sẽ mô phỏng cho hai thao tác này

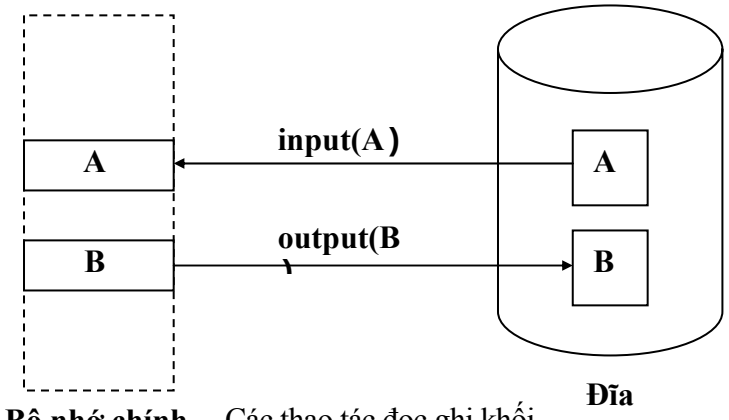


figure VI- 1

Mỗi giao dịch T_i có một vùng làm việc riêng ở đó các bản sao của tất cả các hạng mục dữ liệu được truy xuất và cập nhật được lưu giữ. Vùng làm việc này được tạo ra khi giao dịch khởi động. Nó bị xoá đi khi giao dịch bàn giao (commit) hoặc huỷ bỏ (abort). Mỗi hạng mục dữ liệu ***x*** được trữ trong vùng làm việc của giao dịch T_i sẽ được ký hiệu là x_i. Giao dịch T_i trao đổi với hệ cơ sở dữ liệu bằng cách chuyển dữ liệu đến/ra vùng làm việc của nó sang vùng đệm của hệ thống.

Hai thao tác dùng để chuyển dữ liệu:

1. **read(X)** gán giá trị của hạng mục dữ liệu X cho biến cục bộ x_i. Thao tác này được thực hiện như sau:
   a. Nếu khối B_X chứa X không có trong bộ nhớ chính thì thực hiện thao tác **input(B_X).**
   b. Gán cho x_i giá trị của X trong khối đệm.
2. **write(X)** gán giá trị của biến cục bộ x_i cho hạng mục dữ liệu X trong khối đệm. Thao tác này được thực hiện như sau:
   a. Nếu khối B_X chứa X không có trong bộ nhớ thì thực hiện thao tác **input(B_X).**
   b. Gán giá trị của x_i cho X trong vùng đệm B_X.

Chú ý rằng cả hai thao tác đều có thể đòi hỏi chuyển một khối từ đĩa vào bộ nhớ chính nhưng không yêu cầu chuyển một khối từ bộ nhớ chính ra đĩa.

Đôi khi một khối đệm bị ghi bắt buộc ra đĩa do bộ quản lý vùng đệm cần không gian bộ nhớ cho các mục đích khác hoặc do hệ cơ sở dữ liệu muốn phản ánh những thay đổi trong khối dữ

liệu B trên đĩa. Khi hệ cơ sở dữ liệu thực hiện thao tác **Output(B)** ta nói nó đã xuất bắt buộc khối đệm B ra đĩa.

Khi một giao dịch cần truy xuất hạng mục dữ liệu X lần đầu, nó phải thực hiện **Read(X)**. Khi đó tất cả các cập nhật đối với X được thực hiện trên x_i. Sau khi giao dịch truy xuất X lần cuối, nó thực hiện **Write(X)** để ghi lại sự thay đổi của X trong CSDL.

Không nhất thiết phải thực hiện thao tác **Output(B_X)** ngay sau khi thao tác **write(X)** hoàn thành. Lý do là: khối đệm B_X có thể còn chứa các hạng mục dữ liệu khác đang được truy xuất. Nếu hệ thống bị hư hỏng ngay sau khi thao tác **write(X)** hoàn thành, nhưng trước khi thực hiện thao tác **Output(B_X)**, giá trị mới của X sẽ không bao giờ được ghi ra đĩa, do đó, nó bị mất!

PHỤC HỒI VÀ TÍNH NGUYÊN TỬ:

Trở lại với ví dụ đơn giản về hệ thống ngân hàng: Giao dịch T_i thực hiện việc chuyển $50 từ tài khoản A sang tài khoản B. Giả sử giá trị ban đầu của các tài khoản A và B là $1000 và $2000. Giả sử hệ thống bị hư hỏng trong khi T_i đang thực thi: sau khi thao tác **output(B_A)** được thực hiện và trước khi thực hiện thao tác **output(B_B)** (B_A và B_B là hai khối đệm chứa hai hạng mục A và B). Người ta có thể thực hiện một trong hai giải pháp phục hồi sau:

1. Thực hiện lại T_i. Thủ tục này sẽ dẫn đến kết quả: giá trị của A là $900 thay vì phải là $950. Do đó, hệ thống ở trong trạng thái không nhất quán.
2. Không thực hiện lại T_i . Kết quả: giá trị của A và B tương ứng sẽ là $950 và $2000. Hệ thống cũng trong trạng thái không nhất quán.

Vấn đề phát sinh ở chỗ: T_i thực hiện nhiều thao tác sửa đổi nội dung cơ sở dữ liệu, do đó cần nhiều thao tác xuất dữ liệu ra đĩa, nhưng lỗi phát sinh không cho phép tất cả các thao tác xuất dữ liệu hoàn thành.

Giải pháp nhằm đạt được tính nguyên tử là: trước khi thực hiện các thao tác sửa đổi cơ sở dữ liệu, cần ghi ra các thiết bị lưu trữ bền những thông tin mô tả các sửa đổi này. Cụ thể của giải pháp trên sẽ được trình bày trong các phần V.4, V.5 và V.6.

PHỤC HỒI DỰA TRÊN SỔ GHI LỘ TRÌNH (Log-based recovery)

Một cấu trúc thường được dùng để ghi lại những thay đổi trên cơ sở dữ liệu là sổ ghi lộ trình (log). Log là một dãy các mẩu tin lộ trình (log records). Một thao tác cập nhật trên cơ sở dữ liệu sẽ được ghi nhận bằng một log record. Một log record kiểu mẫu chứa các trường sau:

- **Định danh giao dịch (transaction identifier)**: định danh duy nhất của giao dịch thực hiện hoạt động **write**
- **Định danh hạng mục dữ liệu (Data-item identifier)**: định danh duy nhất của hạng mục dữ liệu được viết (thường là vị trí của hạng mục dữ liệu trên đĩa)
- **Giá trị cũ (Old value)**: giá trị của hạng mục dữ liệu trước khi viết
- **Giá trị mới (New value)**: giá trị hạng mục dữ liệu sẽ có sau khi viết

Có một vài log record đặc biệt mang các ý nghĩa riêng. Bảng sau đây chỉ ra một số loại log record và ý nghĩa của chúng:

LOẠI LOG RECORD	Ý NGHĨA
< T_i **start** >	Giao dịch T_i đã khởi động.
< T_i, X_j, V_1, V_2 >	Giao dịch T_i đã thực hiện thao tác ghi trên hạng mục dữ liệu X_j, X_j có giá trị V_1 trước khi ghi và nhận giá trị V_2 sau khi ghi.
< T_i **commit** >	Giao dịch T_i đã bàn giao.
< T_i **abort** >	Giao dịch Ti đã huỷ bỏ.

Mỗi khi một giao dịch thực hiện một thao tác ghi, trước tiên phải tạo ra một log record cho thao tác ghi đó (trong log file), trước khi giao dịch thay đổi cơ sở dữ liệu. Như vậy, hệ thống có cơ sở để huỷ bỏ (undo) một thay đổi đã được làm trên cơ sở dữ liệu bằng cách sử dụng trường **Old-value** trong log record.

log phải được lưu trong những thiết bị lưu trữ bền. Mỗi một log record mới ngầm định sẽ được thêm vào cuối tập tin log.

CẬP NHẬT TRÌ HOÃN CƠ SỞ DỮ LIỆU (Deferred Database Modification):

Kỹ thuật cập nhật trì hoãn đảm bảo tính nguyên tử của giao dịch bằng cách ghi lại tất cả những sửa đổi cơ sở dữ liệu vào sổ ghi lộ trình (log), nhưng trì hoãn sự thực hiện tất cả các thao tác viết dữ liệu ra đĩa của giao dịch cho đến khi giao dịch bàn giao một phần (partially commits). Nhắc lại rằng: một giao dịch được gọi là bàn giao một phần khi hành động cuối cùng của nó được thực hiện xong. Kỹ thuật cập nhật trì hoãn được trình bày trong phần này giả thiết rằng các giao dịch được thực hiện một cách tuần tự.

Khi giao dịch bàn giao một phần, thông tin trên log kết hợp với giao dịch được sử dụng trong việc viết trì hoãn. Nếu hệ thống có sự cố trước khi giao dịch hoàn thành việc thực hiện của nó hoặc giao dịch bị bỏ dở khi đó thông tin trên log bị bỏ lơ.

Sự thực thi của một giao dịch được tiến triển như sau:

- Trước khi giao dịch T_i bắt đầu thực hiện, một mẫu tin < T_i **start** > được ghi ra sổ lộ trình.
- Trước khi T_i thực hiện thao tác **write(X),** một mẫu tin < T_i, X, V_2 > được ghi ra sổ lộ trình.
- Cuối cùng, khi giao dịch T_i bàn giao một phần, mẫu tin < T_i **commit** > được ghi ra sổ lộ trình.

Khi giao dịch bàn giao một phần, các mẫu tin trong sổ lộ trình kết hợp với giao dịch sẽ được sử dụng để thực hiện việc ghi trì hoãn các hạng mục dữ liệu ra đĩa. Nhà thiết kế phải đảm bảo rằng, trước khi hoạt động ghi hạng mục dữ liệu diễn ra, các mẫu tin log đã được ghi thành công ra các thiết bị lưu trữ bền. Ngoài ra cũng cần để ý: kỹ thuật cập nhật trì hoãn chỉ cần ghi lại giá trị mới của hạng mục dữ liệu (V_2) mà thôi.

Để minh hoạ, ta sử dụng ví dụ hệ thống ngân hàng đơn giản. Gọi T_0 là giao dịch có nhiệm vụ chuyển $50 từ tài khoản A sang tài khoản B, T_1 là giao dịch có nhiệm vụ rút $100 từ tài khoản C. Giả sử giá trị ban đầu của các tài khoản A, B, C là $1000, $2000 và $700. Hành động của T_0 và T_1 được mô tả như sau:

T_0	T_1
read(A)	Read(C)
A:=A-50	C:=C-100
write(A)	write(C)
read(B)	
B:=B+50	
write(B)	

Giả thiết các giao dịch được thực hiện tuần tự: T_0 rồi tới T_1. Một phần của sổ lộ trình ghi lại những thông tin liên quan đến hoạt động của hai giao dịch trên được cho trong bảng dưới đây:

<T_0 start>
<T_0 ,A, 950>
<T_0 ,B, 2050>

<T_0 commit>
<T_1 start>
<T_1 ,C, 600>
<T_1 commit>

figure VI- 2

Sau khi có sự cố xảy ra, hệ thống phục hồi sẽ tham khảo sổ lộ trình để chọn ra những giao dịch nào cần được làm lại (redo). Giao dịch T_i cần được làm lại khi và chỉ khi sổ nhật ký có chứa cả hai mẫu tin <T_i start> và <T_i commit>.

Thủ tục làm lại giao dịch T_i như sau:

- **redo(T_i)** đặt giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T_i. Các giá trị mới sẽ được tìm thấy trong sổ lộ trình (log).

Hoạt động redo phải đồng hiệu lực (idempotent) có nghĩa là việc thực hiện nó nhiều lần tương đương với việc thực hiện nó một lần.

Trở lại ví dụ vừa nêu, ta có bảng mô tả trạng thái của sổ ghi lộ trình và cơ sở dữ liệu như sau:

LOG	CƠ SỞ DỮ LIỆU
<T_0 start>	
<T_0 ,A, 950>	
< T_0 ,B, 2050>	
<T_0 commit>	
	A=950
	B=2050
<T_1 start>	
<T_1 ,C, 600>	
<T_1 commit>	
	C=600

figure VI- 3

Sau đây là một số tình huống mô phỏng:

1. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(B)** của giao dịch T_0 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống khởi động trở lại, sẽ không có hành động "thực hiện lại giao dịch" nào cần phải làm, do không có mẫu tin ghi **commit** nào xuất hiện trong sổ lộ trình. Nghĩa là giá trị của A,B và C vẫn giữ nguyên là $1000, $2000 và $700.
2. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(C)** của giao dịch T_1 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống hoạt động trở lại, thủ tục **redo(T_0)** sẽ được thực hiện do có sự xuất hiện của mẫu tin **<T_0 commit>** trong sổ lộ trình. Sau khi thủ tục này được thực thi, giá trị của A và B sẽ là $950 và $2050.

CẬP NHẬT TỨC THỜI CƠ SỞ DỮ LIỆU (Immediate Database Modification):

Kỹ thuật cập nhật tức thời cho phép các thao tác sửa đổi cơ sở dữ liệu có quyền xuất dữ liệu tức thời ra đĩa trong khi giao dịch vẫn còn ở trong trạng thái hoạt động (active state). Hành động thay đổi nội dung dữ liệu tức thời của các giao dịch đang hoạt động được gọi là "những thay đổi chưa được bàn giao" (uncommitted modifications).

Sự thực thi của một giao dịch được tiến hành như sau:

- Trước khi giao dịch T_i bắt đầu sự thực hiện, một mẫu tin < T_i **start** > được ghi ra sổ lộ trình.

- Trước khi T_i thực hiện thao tác **write(X),** một mẫu tin < T_i, X, V_1, V_2 > được ghi ra sổ lộ trình.
- Cuối cùng, khi giao dịch T_i bàn giao một phần, mẫu tin < T_i **commit** > được ghi ra sổ lộ trình.

Cần phải đảm bảo rằng, trước khi hoạt động ghi hạng mục dữ liệu diễn ra, các mẫu tin log đã được ghi thành công ra các thiết bị lưu trữ bền. Ngoài ra, cũng cần chú ý là mẫu tin log cho hành động **write(X)** của giao dịch T_i, tức là mẫu tin <T_i, X, V_1, V_2> có chứa cả hai giá trị mới (V_2) và cũ (V_1) của hạng mục dữ liệu X.

Trở lại với ví dụ trong phần V.4.1, ta có một phần của sổ lộ trình liên quan đến các hoạt động của T_0 và T_1 như sau:

<T_0 start>
<T_0 , A, 1000, 950>
<T_0 , B, 2000, 2050>
<T_0 commit>
<T_1 start>
<T_1 , C, 700, 600>
<T_1 commit>

figure VI- 4

Bảng mô tả trạng thái của sổ ghi lộ trình và cơ sở dữ liệu như sau:

LOG	CƠ SỞ DỮ LIỆU
<T_0 start>	
<T_0 , A, 1000, 950>	
< T_0 , B, 2000, 2050>	
	A=950
	B=2050
<T_0 commit>	
<T_1 start>	
<T_1 , C, 700, 600>	
	C=600
<T_1 commit>	

figure VI- 5

Kỹ thuật cập nhật tức thời sử dụng hai thủ tục khôi phục sau lỗi:

- **undo(T_i)** đặt lại giá trị cũ cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T_i. Các giá trị cũ sẽ được tìm thấy trong sổ lộ trình (log).
- **redo(T_i)** đặt giá trị mới cho tất cả các hạng mục dữ liệu được cập nhật bởi giao dịch T_i. Các giá trị mới sẽ được tìm thấy trong sổ lộ trình (log).

Sau khi lỗi xuất hiện, hệ thống phục hồi tham khảo sổ ghi để quyết định những giao dịch nào cần được làm lại (redo) và những giao dịch nào cần được huỷ bỏ (undo).

- Giao dịch T_i cần được huỷ bỏ khi sổ ghi chứa mẫu tin <T_i start> nhưng không có mẫu tin <T_i commit>.
- Giao dịch T_i cần được làm lại khi sổ ghi có chứa cả mẫu tin <T_i start> lẫn mẫu tin <T_i commit>.

Sau đây là một số tình huống mô phỏng:

1. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(B)** của giao dịch T_0 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống khởi động trở lại, nó sẽ tìm thấy mẫu tin <T_0 start> trong sổ ghi, nhưng không có mẫu tin <T_0 commit> tương ứng. Do đó giao dịch T_0 cần phải được huỷ bỏ. Nghĩa là thủ tục **undo(T_0)** sẽ được gọi và giá trị của A,B và C vẫn giữ nguyên là \$1000, \$2000 và \$700.
2. Giả sử lỗi hệ thống xảy ra sau khi mẫu tin log cho hành động **write(C)** của giao dịch T_1 vừa được ghi ra thiết bị lưu trữ bền. Khi hệ thống hoạt động trở lại, hai thủ tục **redo(T_0)** và **undo(T_1)** sẽ được thực hiện. Do có sự xuất hiện của các mẫu tin **<T_0 start>**, **<T_0 commit>**, **<T_1 start>** trong sổ lộ trình. Sau khi hai thủ tục này được thực thi, giá trị của A, B, C sẽ là \$950,ì \$2050 và \$700.

ĐIỂM KIỂM SOÁT (Checkpoint):

Khi lỗi hệ thống xuất hiện, hệ thống phục hồi phải tham khảo sổ ghi lộ trình để quyết định những giao dịch nào cần được làm lại và những giao dịch nào cần được huỷ bỏ. Theo nguyên lý thì cần phải tìm kiếm toàn bộ nội dung của sổ ghi để có được quyết định trên.

Hướng tiếp cận trên sẽ gặp phải hai khó khăn lớn:

1. Quá trình tìm kiếm mất nhiều thời gian.
2. Theo các giải thuật vừa nêu, hầu hết các giao dịch cần được làm lại đã ghi những dữ liệu được cập nhật ra cơ sở dữ liệu rồi. Việc làm lại chúng tuy không có hại gì, nhưng lại làm cho tiến trình khôi phục trở nên lâu hơn.

Công cụ "điểm kiểm soát" (checkpoint) được sử dụng để cải thiện hiệu năng của quá trình khôi phục. Trong quá trình hoạt động của mình, hệ thống sẽ duy trì một sổ ghi lộ trình bằng cách sử dụng một trong hai kỹ thuật được giới thiệu trong phần V.4.1 và V.4.2. Ngoài ra, hệ thống còn phải thực hiện một cách chu kỳ các hành động đặt điểm kiểm soát. Hành động này đòi hỏi một dãy các thao tác sau:

1. Xuất ra lưu trữ bền tất cả các mẫu tin ghi nhận lộ trình (log record) đang nằm trong bộ nhớ chính.
2. Xuất ra đĩa tất cả những khối đệm đã được cập nhật.
3. Xuất ra thiết bị lưu trữ bền một log-record <**checkpoint**>

Các giao dịch sẽ không được phép thực hiện bất kỳ thao tác cập nhật dữ liệu nào (ví dụ như ghi các khối đệm, ghi các mẫu tin log) khi hành động đặt điểm kiểm soát đang được thực hiện.

Sự hiện diện của điểm kiểm soát trong sổ ghi cho phép hệ thống tổ chức quá trình phục hồi tốt hơn. Xét một giao dịch T_i đã bàn giao (commit) trước một điểm kiểm soát. Ta có mẫu tin **<T_i commit>** xuất hiện trước mẫu tin **<checkpoint>.** Có nghĩa là tất cả các thay đổi mà T_i đã làm đối với cơ sở dữ liệu phải được thực hiện trước khi người ta đặt điểm kiểm soát trên. Vì vậy, trong giai đoạn phục hồi sau lỗi, người ta không cần phải làm lại (**redo)** giao dịch T_i.

Dựa trên điểm cải tiến này, ta cải tiến lại các kỹ thuật đã được trình bày trong phần V.4.1 và V.4.2 như sau:

1. Sau khi lỗi hệ thống xuất hiện, hệ thống phục hồi sẽ kiểm tra lại sổ lộ trình (log) để tìm ra giao dịch T_i thoả điều kiện: đó là giao dịch gần đây nhất được khởi động trước điểm kiểm soát gần đây nhất. Qui trình tìm T_i như sau: dò ngược trong sổ ghi lộ trình cho đến khi tìm thấy mẫu tin <**checkpoint**> đầu tiên. Từ điểm kiểm soát này, lại tiếp tục dò ngược trong sổ ghi cho đến khi tìm thấy mẫu tin <**T_i start**> đầu tiên. Mẫu tin này chỉ ra giao dịch T_i .
2. Khi đã xác định được giao dịch T_i rồi, các thủ tục **undo** và **redo** chỉ được áp dụng cho giao dịch T_i và các giao dịch diễn ra sau T_i. Chúng ta ký hiệu tập những giao dịch vừa nói là **T**.
3. Với kỹ thuật "Cập nhật tức thời cơ sở dữ liệu", tiến trình phục hồi như sau:

- Với mọi giao dịch $T_k \in \mathbf{T}$ mà không có mẫu tin <**T_k commit**> trong sổ ghi lộ trình, thực thi **undo(T_k)**.
- Với mọi giao dịch $T_k \in T$ mà có mẫu tin <**T_k commit**> trong sổ ghi lộ trình, thực thi **redo(T_k)**.

4. Không cần thực thi thao tác **undo** khi sử dụng kỹ thuật "Cập nhật có trì hoãn cơ sở dữ liệu".

PHÂN TRANG BÓNG (Shadow Paging):

Kỹ thuật "Phân trang bóng" cũng là kỹ thuật cho phép phục hồi sau lỗi, nhưng ý tưởng thực hiện khác với các kỹ thuật dựa trên sổ ghi lộ trình vừa trình bày ở phần trên.
Sau đây là một số khái niệm cần được giải trình:

- ***Trang (page) là gì***? Như đã trình bày ở các phần trước, cơ sở dữ liệu được lưu vào thiết bị lưu trữ không phai thành nhiều khối có kích thước cố định. Người ta gọi những khối này là trang (page).
- ***Bảng trang và ý nghĩa của nó***: Khái niệm trang đã nói được mượn từ lý thuyết về Hệ điều hành. Cách quản lý trang cũng được thừa kế từ đó. Giả sử rằng cơ sở dữ liệu được phân thành n trang và sự phân bố trên đĩa của chúng có thể không theo một thứ tự cụ thể nào cả. Tuy nhiên, phải có cách để tìm ra nhanh và đúng trang thứ i của cơ sở dữ liệu ($1 \le i \le n$). Người ta dùng bảng trang (được mô phỏng như trong hình 5.2) cho mục đích này. Bảng trang có n đầu vào (entry). Mỗi đầu vào ứng với một trang. Một đầu vào chứa một con trỏ, trỏ đến một trang trên đĩa. Đầu vào đầu tiên chỉ đến trang đầu tiên của cơ sở dữ liệu, đầu vào thứ hai chỉ đến trang thứ hai ...

Ý tưởng then chốt của kỹ thuật "Phân trang bóng" là người ta sẽ duy trì hai bảng trang trong suốt chu kỳ sống của giao dịch, một bảng trang gọi là "*bảng trang hiện hành"* (current page table), bảng trang còn lại gọi là "*bảng trang bóng"* (shadow page table). Khi giao dịch khởi động, hai bảng trang này giống nhau. Bảng trang bóng sẽ không thay đổi suốt quá trình hoạt động của giao dịch. Bảng trang hiện hành sẽ bị thay đổi mỗi khi giao dịch thực hiện tác vụ **write.** Tất cả các tác vụ **input** và **output** đều sử dụng bảng trang hiện hành để định vị các trang trong đĩa. Điểm quan trọng khác là nên lưu bảng trang bóng vào thiết bị lưu trữ bền.

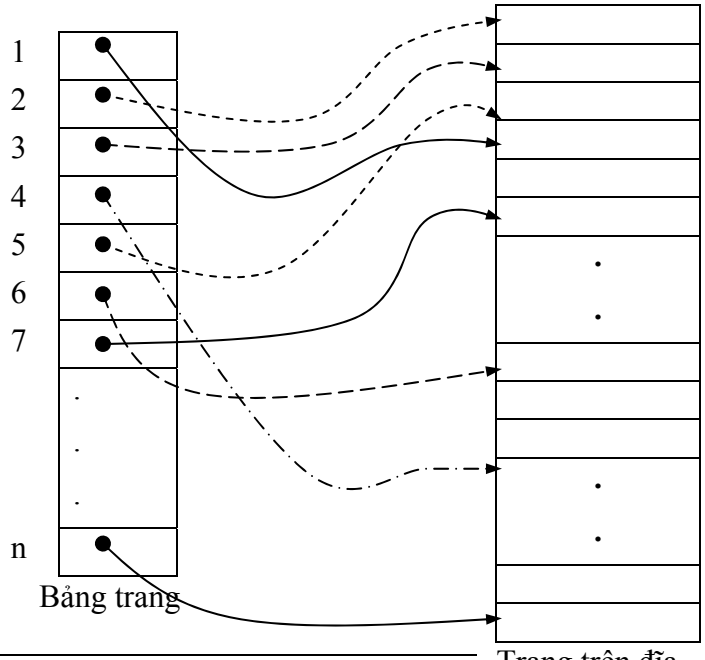


figure VI- 6

Giả sử giao dịch thực hiện tác vụ **write(X)** và hạng mục dữ liệu X được chứa trong trang thứ ***i.*** Tác vụ **write** được thực thi như sau:

1. Nếu trang thứ ***i*** chưa có trong bộ nhớ chính, thực hiện **input(X).**
2. Nếu đây là lệnh ghi được thực hiện ***lần đầu tiên*** trên trang thứ ***i*** bởi giao dịch, sửa đổi bảng trang hiện hành như sau:
   a. Tìm một trang chưa được dùng trên đĩa.
   b. Xoá trang vừa được tìm xong ở bước 2.a khỏi danh sách các khung trang tự do.
   c. Sửa lại bảng trang hiện hành sao cho đầu vào thứ ***i*** trỏ đến trang mới vừa tìm được trong bước 2.a.
3. Gán giá trị x_i cho X trong trang đệm (buffer page).

Để bàn giao một giao dịch, cần làm các bước sau:

1. Đảm bảo rằng tất cả các trang đệm trong bộ nhớ chính đã được giao dịch sửa đổi phải được xuất ra đĩa.
2. Xuất bảng trang hiện hành ra đĩa. chú ý là không được viết đè lên trang bóng
3. Xuất địa chỉ đĩa của bảng trang hiện hành ra vị trí cố định trong thiết bị lưu trữ bền. Vị trí này chính là nơi chứa địa chỉ của bảng trang bóng. Hành động này sẽ ghi đè lên địa chỉ của bảng trang bóng cũ. Như vậy, bảng trang hiện hành sẽ trở thành bảng trang bóng và giao dịch được bàn giao.

Nếu sự cố xảy ra trước khi hoàn thành bước thứ 3, hệ thống sẽ trở về trạng thái trước khi giao dịch được thực hiện. Nếu sự cố xảy ra sau khi bước thứ 3 hoàn thành, hiệu quả của giao dịch được bảo tồn; không cần thực hiện thao tác **redo** nào cả. Ví dụ trong hình 5.3 dưới đây mô phỏng lại trạng thái của các bảng trang hiện hành và bảng trang bóng khi giao dịch thực hiện thao tác ghi lên trang thứ tư của cơ sở dữ liệu có 10 trang.

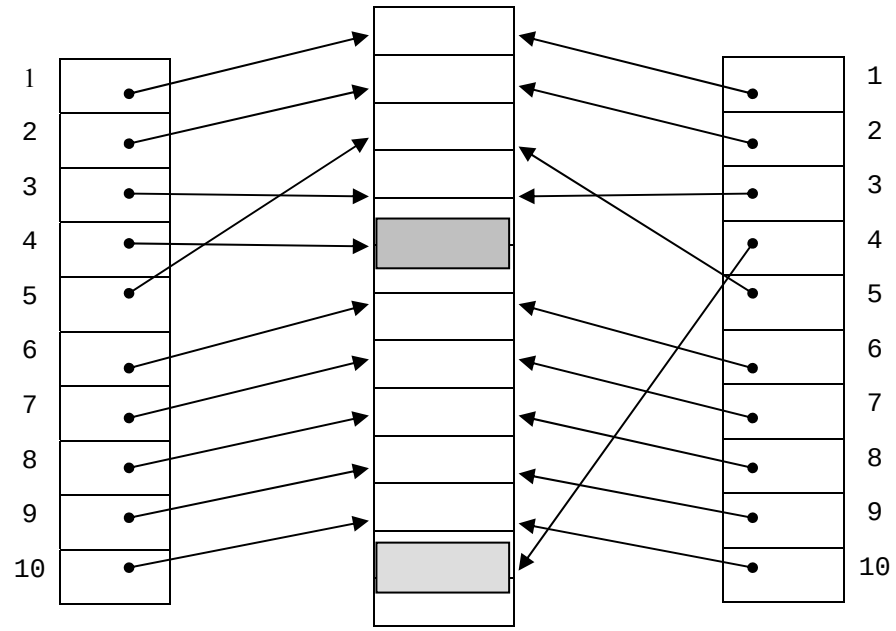


figure VI- 7 Ví dụ về bảng trang bóng và bảng trang hiện hành

Kỹ thuật phân trang bóng có một số điểm lợi hơn so với các kỹ thuật dựa trên sổ ghi:

1. Không mất thời gian ghi ra các log record.
2. Khôi phục sau sự cố nhanh hơn, do không cần các thao tác **undo** hoặc **redo**.

Tuy nhiên kỹ thuật phân trang bóng lại có nhiều nhược điểm:

- **Tổng phí bàn giao.** Xuất nhiều khối ra đĩa: các khối dữ liệu hiện tại, bảng trang hiện hành, địa chỉ của bảng trang hiện hành. Trong kỹ thuật dựa vào sổ ghi, chỉ cần xuất ra các log record, mà thông thường, các log record này vừa đủ chứa trong một khối.
- **Sự phân mảnh dữ liệu**. Trong chương II có trình bày chiến lược gom cụm vật lý các trang dữ liệu có liên quan với nhau. Sự gom cụm này cho phép việc vận chuyển dữ liệu nhanh hơn. Kỹ thuật phân trang bóng lại đổi vị trí của trang khi trang này bị sửa đổi. Điều này dẫn đến tính gom cụm dữ liệu không còn, hoặc phải dùng các giải pháp gom cụm lại rất mất thời gian.
- **Phải thu nhặt rác**. Mỗi khi giao dịch bàn giao, các trang chứa giá trị dữ liệu cũ đã bị sửa đổi bởi giao dịch sẽ trở thành không truy xuất được. Vì chúng không thuộc danh sách các trang tự do nhưng cũng không chứa dữ liệu hữu dụng. Ta gọi chúng là "rác". Cần thiết phải định kỳ tìm kiếm và thêm các trang rác vào trong danh sách các trang tự do. Hành động này được gọi là "thu nhặt rác".
- Ngoài ra, kỹ thuật phân trang bóng sẽ gặp nhiều khó khăn hơn kỹ thuật dựa vào sổ ghi khi cần được tinh chỉnh để đáp ứng cho yêu cầu phục vụ song song cho nhiều giao dịch. Vì những lý do trên, kỹ thuật phân trang bóng không được sử dụng rộng rãi lắm.

PHỤC HỒI VỚI CÁC GIAO DỊCH CẠNH TRANH

Cho dến bây giờ, ta chỉ xét các kỹ thuật phục hồi áp dụng cho các giao dịch được thực thi tuần tự. Bây giờ chúng ta sẽ tìm cách cải tiến kỹ thuật dựa vào sổ ghi nhằm đáp ứng yêu cầu phục vụ đồng thời cho nhiều giao dịch cạnh tranh. Ý tưởng thực hiện là: Không quan tâm đến số lượng các giao dịch cạnh tranh, hệ thống vẫn sử dụng một vùng đệm đĩa và một sổ ghi lộ trình. Các khối đệm được chia sẻ bởi tất cả các giao dịch. Chúng ta sẽ cho phép việc cập nhật tức thời cơ sở dữ liệu và cho phép một khối đệm có nhiều hạng mục dữ liệu được cập nhật bởi một hoặc nhiều giao dịch.

TRAO ĐỔI VỚI ĐIỀU KHIỂN CẠNH TRANH

Cơ chế phục hồi phụ thuộc rất nhiều vào cơ chế điều khiển cạnh tranh được sử dụng. Để cuộn lại một giao dịch thất bại (failed transaction), người ta phải huỷ bỏ (undo) các cập nhật được thực hiện bởi giao dịch. Giả sử giao dịch T_0 phải bị cuộn lại và một hạng mục dữ liệu Q đã bị T_0 thay đổi giá trị và cần phải được đặt lại giá trị cũ. Bằng cách sử dụng kỹ thuật dựa vào sổ ghi lộ trình, ta trả lại giá trị cũ cho Q bằng cách sử dụng một log record. Giả thiết lại có giao dịch thứ hai T_1 cũng vừa cập nhật Q xong, trước khi T_0 bị cuộn lại. Như vậy, sự cập nhật được thực hiện bởi T_1 sẽ bị mất đi nếu T_0 bị cuộn lại.

Biện pháp khắc phục là: nếu một giao dịch T đã cập nhật một hạng mục dữ liệu Q, thì không một giao dịch nào khác có quyền cập nhật lên hạng mục dữ liệu đó trong khi T chưa bàn giao hoặc chưa bị cuộn lại. Chúng ta có thể đảm bảo yêu cầu trên được thoả bằng cách sử dụng kỹ thuật "chốt hai kỳ nghiêm ngặt" (strict two-phase locking).

CUỘN LẠI GIAO DỊCH:

Phương pháp để cuộn lại (rollback) một giao dịch T_i, sử dụng sổ ghi, trong môi trường cạnh tranh như sau:

1. Dò ngược sổ ghi lộ trình để tìm ra các log record có dạng <T_i, Xj, V_1, V_2>.

2. Hạng mục dữ liệu Xj sẽ được trả lại giá trị cũ V_1.
3. Việc dò tìm kết thúc khi tìm thấy mẫu tin <T_i start>.

Việc dò ngược sổ ghi lộ ¹eó một ý nghĩa rất quan trọng, do một giao dịch có thể đã cập nhật một hạng mục dữ liệu nhiều hơn một lần. Một ví dụ: Xét một cặp log records như sau:

<T_i, A, 10, 20>
<T_i, A, 20, 30>

Cặp mẫu tin này thể hiện hai hành động cập nhật hạng mục dữ liệu A của giao dịch T_i. Nếu dò ngược sổ ghi lộ trình, A sẽ được trả về giá trị đúng là 10. Ngược lại, A sẽ nhận giá trị sai là 20.

Nếu kỹ thuật strict two-phase locking được sử dụng để điều khiển cạnh tranh, thì việc trả về giá trị cũ cho một hạng mục dữ liệu sẽ không xoá đi những tác động của các giao dịch khác lên hạng mục dữ liệu này.

CÁC ĐIỂM KIỂM SOÁT

Ở phần V.4.3, người ta đã sử dụng điểm kiểm soát (checkpoint) để làm giảm số lượng các log record mà hệ thống phục hồi phải dò tìm trong sổ ghi trong giai đoạn phục hồi sau lỗi. Nhưng, do đã giả thiết là không có cạnh tranh nên giải pháp V.4.3 chỉ xét đến những giao dịch sau trong quá trình khôi phục lỗi:

- Những giao dịch được khởi động sau điểm kiểm soát gần đây nhất.
- Một giao dịch (nếu có) đang trong trạng thái hoạt động (active) tại thời điểm người ta đặt điểm kiểm soát gần đây nhất.

Tình huống càng phức tạp khi các giao dịch được thực thi cạnh tranh. Có nghĩa là tại thời điểm đặt điểm kiểm soát, có thể có nhiều giao dịch đang ở trong trạng thái hoạt động.

Trong một hệ thống xử lý các giao dịch cạnh tranh, ta yêu cầu rằng: một mẫu tin ghi dấu kiểm soát (checkpoint log record) phải có dạng như sau:

<**checkpoint L**>

Trong đó L là danh sách các giao dịch đang hoạt động tại thời điểm đặt điểm kiểm soát.

Một lần nữa, ta qui ước rằng: khi hành động đặt điểm kiểm soát đang diễn ra, các giao dịch không được phép thực hiện bất kỳ thao tác cập nhật dữ liệu nào cả trên các khối đệm lẫn trên sổ ghi lộ trình.

Tuy nhiên, qui ước trên lại gây phiền toái, bởi vì các giao dịch phải ngừng hoạt động khi đặt điểm kiểm soát. Một kỹ thuật nâng cao giải quyết điểm phiền toái này là "Điểm kiểm soát mờ" (fuzzy checkpoint).

PHỤC HỒI KHỞI ĐỘNG LẠI (Restart Recovery)

Khi hệ thống phục hồi sau lỗi, nó tạo ra hai danh sách: *undo-list* bao gồm các giao dịch cần phải huỷ bỏ và *redo-list* bao gồm danh sách các giao dịch cần được làm lại.

Qui trình tạo lập hai danh sách *redo-list, undo-list* được thực hiện như sau:

1. Đầu tiên, chúng sẽ rỗng.
2. Dò ngược sổ ghi lộ trình, kiểm tra các mẫu tin cho đến khi tìm được mẫu tin <**checkpoint**> đầu tiên:
   - **a.** Với mỗi mẫu tin được tìm thấy theo dạng <**T_i commit**>, ta thêm T_i vào trong *redo-list.*
   - **b.** Với mỗi mẫu tin được tìm thấy theo dạng <**T_i start**>, nếu T_i không thuộc *redo-list* thì thêm T_i vào trong *undo-list.*
   - **c.** Khi tất cả các log record đã được xem xét, ta kiểm tra danh sách **L** trong mẫu tin <**checkpoint L**>. Với mọi giao dịch T_i trong **L**, nếu T_i không thuộc *redo-list* thì thêm T_i vào *undo-list*.

Khi hai danh sách *redo-list, undo-list* được thiết lập xong, tiến trình phục hồi được tiến hành như sau:

1. Dò ngược lại sổ ghi và thực hiện thủ tục **undo** đối với mỗi log record thuộc về giao dịch T_i trong *undo-list.* Các log record của các giao dịch nằm trong danh sách *redo-list* sẽ bị bỏ qua trong giai đoạn này. Việc dò ngược sẽ ngưng khi mẫu tin <**T_i start**> được tìm thấy cho mọi giao dịch T_i thuộc danh sách *undo-list.*
2. Định vị mẫu tin <**checkpoint L**> gần đây nhất trong log.
3. Dò sổ ghi theo chiều xuôi bắt đầu từ mẫu tin <**checkpoint L**> gần đây nhất và thực hiện thủ tục **redo** đối với mỗi log record thuộc về giao dịch T_i nằm trong danh sách *redo-list*. Trong giai đoạn này, bỏ qua các log record của các giao dịch thuộc danh sách *undo-list*.

Việc xử lý ngược ở bước 1 là rất quan trọng, nhằm đảm bảo kết quả trả về của cơ sở dữ liệu là đúng.

Sau khi tất cả các giao dịch trong danh sách *undo-list* bị huỷ bỏ, tất cả các giao dịch trong danh sách *redo-list* sẽ được làm lại. Sau khi tiến trình phục hồi thành công, xử lý giao dịch được tiếp tục.

Việc thực hiện huỷ bỏ các giao dịch trong *undo-list* trước khi làm lại các giao dịch trong *redo-list* có ý nghĩa rất quan trọng. Nếu làm ngược lại, vấn đề sau sẽ phát sinh: Giả sử hạng mục dữ liệu A có giá trị khởi đầu là 10. Giao dịch T_i đổi A thành 20 sau đó T_i bị huỷ bỏ. Sau đó, một giao dịch khác T_j cập nhật A thành 30. Đến đây thì hệ thống bị lỗi ngừng hoạt động. Hiện trạng của sổ ghi tại thời điểm hệ thống bị lỗi như sau:

<T_i, A, 10, 20>
<T_j, A, 10, 30>
<T_j commit>

Nếu thực hiện redo trước, A sẽ được đặt giá trị 30. Sau đó thực hiện undo, A sẽ được đặt giá trị 10, mà giá trị này sai. Giá trị cuối cùng của A phải là 30.

ĐIỂM KIỂM SOÁT MỜ (fuzzy checkpoint):

Kỹ thuật fuzzy checkpoint cho phép các giao dịch được cập nhật dữ liệu trên các khối đệm khi checkpoint-record đã được viết xong nhưng trước thời điểm các khối đệm đã sửa đổi được ghi ra đĩa.

Ý tưởng thực hiện fuzzy checkpoint như sau: Thay vì phải dò ngược sổ ghi để tìm mẫu tin checkpoint, ta sẽ lưu vị trí của mẫu tin checkpoint cuối cùng trong sổ ghi vào một chỗ cố định trong đĩa gọi là ***last_checkpoint***. Tuy nhiên, thông tin này sẽ không được cập nhật khi một mẫu tin checkpoint được ghi ra đĩa. Thay vào đó, trước khi một mẫu tin checkpoint được ghi ra sổ ghi, ta tạo ra một danh sách các khối đệm bị sửa đổi. Thông tin ***last_checkpoint*** được cập nhật chỉ sau khi tất cả các khối đệm bị sửa đổi được ghi ra đĩa.

last_checkpoint chỉ được dùng cho mục đích ***undo***.

BÀI TẬP CHƯƠNG VI

VI.1. Trình bày các điểm khác nhau giữa 3 kiểu lưu trữ: lưu trữ không ổn định, lưu trữ ổn định và lưu trữ bền theo tiêu chuẩn đánh giá là *chi phí cài đặt*.

VI.2. Thực tế, lưu trữ bền không thể thực hiện được. Giải thích tại sao? Hệ cơ sở dữ liệu giải quyết vấn đề này như thế nào?

VI.3. So sánh các kỹ thuật cập nhật tức thời và cập nhật có trì hoãn trong sơ đồ phục hồi dựa vào sổ ghi lộ trình theo các tiêu chuẩn: *tính dễ cài đặt và tổng chi phí thực hiện*.

VI.4. Giả sử rằng kỹ thuật cập nhật tức thời được sử dụng trong hệ thống. Bằng ví dụ, hãy chỉ ra rằng: tình trạng không nhất quán dữ liệu sẽ xảy ra nếu các log record không được ghi ra thiết bị lưu trữ bền trước khi giao dịch bàn giao (commit).

VI.5. Giải thích mục đích của cơ chế điểm kiểm soát (checkpoint). Hành động đặt điểm kiểm soát nên được thực hiện theo chu kỳ bao lâu là hợp lý?

VI.6. Khi hệ thống phục hồi sau lỗi, nó xây dựng 2 danh sách: *undo-list* và *redo-list*. Giải thích tại sao các log record của các giao dịch trong danh sách *undo-list* phải được xử lý theo thứ tự ngược, trong khi những log record trong danh sách *redo-list* lại được xử lý theo chiều xuôi?

VI.7. So sánh sơ đồ phục hồi phân trang bóng và sơ đồ phục hồi bằng sử dụng sổ ghi lộ trình theo tiêu chuẩn: *tính dễ cài đặt và tổng chi phí thực hiện*.

VI.8. Giả sử một cơ sở dữ liệu có 10 khối đĩa liên tiếp (khối 1, 2, 3, ..., 10). Hãy thể hiện trạng thái của buffer và thứ tự vật lý có thể có của các khối sau các thao tác cập nhật sau, giả sử: kỹ thuật phân trang bóng được sử dụng, buffer trong bộ nhớ chỉ đủ chứa 3 khối, chiến lược quản lý buffer là LRU (Least Recently Used)

Đọc khối 3
Đọc khối 7
Đọc khối 5

Đọc khối 3

Đọc khối 1

Sửa đổi khối 1

Đọc khối 10

Sửa khối 5