Mục lục

CHƯƠNG 1. KHÁI QUÁT VỀ CÁC HỆ CƠ SỞ DỮ LIỆU

Mục đích

Khái quát các nguyên lý của hệ cơ sở dữ liệu (CSDL) gồm CSDL, hệ quản trị CSDL, con người và các trang thiết bị lưu trữ xử lý dữ liệu.

Trình bày về quá trình quản lý dữ liệu bao gồm định nghĩa các cấu trúc lưu trữ thông tin, cung cấp các cơ chế cho việc thao tác thông tin, đảm bảo an toàn cho các sự cố và truy cập không được phép, đảm bảo các dị thường dữ liệu khi có nhiều người cùng chia sẻ dữ liệu.

Kiến trúc 3 mức của một hệ CSDL nhằm thể hiện các mức trừu tượng dữ liệu, giúp cho đa số người sử dụng tránh phải quan tâm đến chi tiết về lưu trữ và bảo trì dữ liệu.

Ba mức thiết kế CSDL cùng các sản phẩm tương ứng là các lược đồ ngoài, lược đồ khái niệm, lược đồ trong được trình bày, khái niệm độc lập dữ liệu cũng được đề cập.

Các chức năng và thành phần chủ yếu của hệ quản trị CSDL (DBMS).

Vai trò và chức năng của con người trong mối quan hệ tương tác với hệ CSDL.

Yêu cầu

Hiểu và cho ví dụ minh họa các khái niệm cơ bản của hệ CSDL.

Cho ví dụ minh họa về các chức năng của DBMS (đã học ở các năm dưới) và thấy được vai trò của mình trong tương lai với hệ CSDL.

Cho ví dụ minh họa nêu lên ý nghĩa của kiến trúc 3 mức.

1.1. Cơ sở dữ liệu là gì? Tại sao cần tới các hệ cơ sở dữ liệu?

Trước khi các hệ CSDL ra đời (khoảng đầu những năm 60 của thế kỷ 20), mỗi chương trình ứng dụng đều có một tệp dữ liệu tương ứng và mỗi khi chương trình ứng dụng cần được sửa đổi hoặc mở rộng thì tệp dữ liệu tương ứng cũng phải thay đổi theo. Việc lưu giữ thông tin của một tổ chức trong một hệ xử lý tệp như vậy (thường hệ này được hỗ trợ bởi một hệ điều hành truyền thống) có những nhược điểm chính như sau:

- *Dư thừa dữ liệu và không nhất quán* (cùng một dữ liệu có thể có được lưu trữ trong nhiều tệp khác nhau; khi tiến hành cập nhật có thể bỏ sót và dẫn tới không nhất quán).
- *Khó khăn trong việc truy cập dữ liệu* (các môi trường xử lý tệp truyền thống không cho phép dữ liệu được tìm kiếm theo cách thức thuận tiện và hiệu quả).
- *Sự cô lập của dữ liệu* (dữ liệu nằm rải rác trong nhiều tệp và các tệp có thể có khuôn dạng khác nhau, nên khó viết các chương trình ứng dụng mới để tìm các dữ liệu thích hợp).
- *Các vấn đề toàn vẹn* (khi có thêm những ràng buộc mới, khó thay đổi các chương trình để có thể tuân thủ chúng).
- *Các vấn đề về tính nguyên tố của các giao tác* (với hệ xử lý tệp truyền thống khó có thể đảm bảo được tính chất "hoặc thực hiện hoàn toàn hoặc không thực hiện gì" và khó đưa được hệ thống trở về trạng thái nhất quán trước khi xảy ra sự cố).
- *Các dị thường của truy cập tương tranh* (để tăng tính hiệu quả và trả lời nhanh hơn, nhiều hệ thống cho phép nhiều người dùng cập nhật dữ liệu đồng thời và như vậy có thể dẫn đến dữ liệu không nhất quán).
- *Các vấn đề an toàn*. Thường thì mỗi người dùng của hệ CSDL chỉ được phép truy cập một phần của CSDL và điều đó cũng là một biện pháp giữ cho dữ liệu trong CSDL được an toàn. Còn với hệ xử lý – tệp truyền thống, các chương trình ứng dụng được thêm vào hệ thống an toàn một cách thức không tiên liệu trước nên khó đảm bảo được các ràng buộc an toàn như vậy.

Các hệ CSDL ra đời nhằm giải quyết những vấn đề nêu trên.

Một **cơ sở dữ liệu (CSDL)** *là một tập hợp các dữ liệu có liên quan với nhau được lưu trữ trên các thiết bị nhớ thứ cấp (như băng từ, đĩa từ ...) để đáp ứng nhu cầu khai thác thông tin của nhiều người sử dụng với nhiều mục đích khác nhau.*

Trong cách hiểu trên, trước hết CSDL phải phản ánh thông tin về hoạt động của nhiều sự vật hiện tượng, nghĩa là biểu thị một "góc" của thế giới thực tại, nó phải là một tập hợp các thông tin mang tính hệ thống chứ không thể là một tập

hợp dữ liệu tùy tiện chứa những thông tin rời rạc không có mối quan hệ với nhau. Thông tin lưu trữ trong CSDL được chia sẻ cho nhiều người sử dụng và nhiều ứng dụng khác nhau. Từ đó có thể thấy việc xây dựng và khai thác một CSDL liên quan đến một số vấn đề như đảm bảo tính nhất quán và toàn vẹn dữ liệu, tính bảo mật và quyền khai thác thông tin của người sử dụng, tính an toàn cho dữ liệu khi xảy ra sự cố nào đó…

Phần mềm cho phép người dùng giao tiếp với CSDL, cung cấp một môi trường thuận lợi và hiệu quả để tìm kiếm và lưu trữ thông tin của CSDL được gọi là ***hệ quản trị cơ sở dữ liệu*** *(hệ QTCSDL).*

Người ta thường dùng thuật ngữ ***hệ cơ sở dữ liệu*** để chỉ sự kết hợp của các yếu tố sau:

- Cơ sở dữ liệu

- Hệ QTCSDL để truy cập CSDL đó.

- Con người và trang thiết bị để lưu trữ dữ liệu.

Mục đích chính của một hệ CSDL là cung cấp cho người dùng một cách nhìn trừu tượng về dữ liệu. Điều đó có nghĩa là hệ thống che dấu những chi tiết phức tạp về cách thức dữ liệu được lưu trữ và bảo trì. Chính vì vậy, trong cuộc sống hiện đại ngày nay, việc sử dụng các CSDL trở nên phổ biến, quen thuộc đến nỗi nhiều lúc chúng ta coi đó là điều tự nhiên. Khi đến thư viện tìm mượn sách, nhờ máy tính ít nhất chúng ta có thể biết được thông tin chi tiết về sách của thư viện, thông tin về sách đã có người xếp hàng đặt mượn… Khi chúng ta muốn đặt chỗ cho chuyến bay sắp tới của mình, nhân viên đại lí bán vé hàng không sẽ nhanh chóng cung cấp những thông tin cần thiết giúp chúng ta như một thông tin cập nhật vào tập hợp dữ liệu được lưu trữ. Sự phát triển mạnh mẽ của Internet ở thập kỉ cuối của thế kỷ 20 đã làm số người truy cập và khai thác thông tin trong các cơ sở dữ liệu tăng lên rất nhanh chóng. Với các giao diện Web, người ta có thể đăng kí các khóa học ở một trường đại học, có thể xem số dư trong tài khoản của mình ở một ngân hàng, có thể tìm hiểu chi tiết về một mặt hàng nào đó, … Càng ngày việc truy xuất thông tin trong các cơ sở dữ liệu càng trở thành một bộ phận thiết yếu trong cuộc sống của mỗi người. Vì nhiều người sử dụng CSDL không thuộc giới chuyên tin, những người phát triển hệ thống đã che dấu không

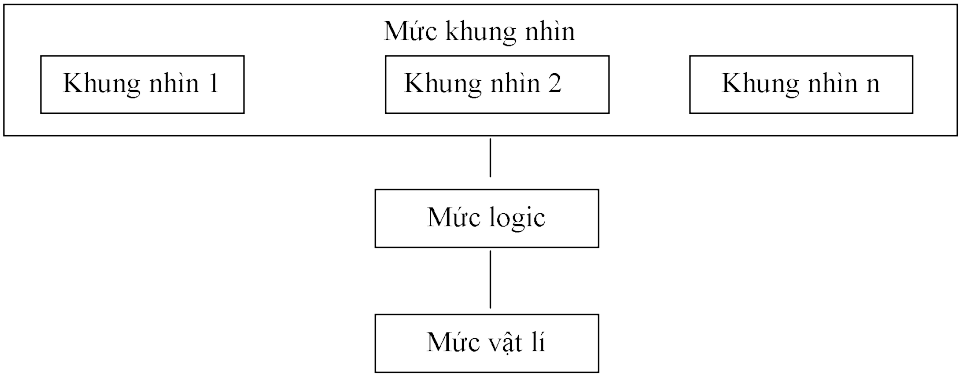
cho người dùng biết sự phức tạp đó thông qua nhiều mức biểu diễn, nhằm làm đơn giản những tương tác của người dùng với hệ thống.

1.2. Kiến trúc ba mức của một hệ CSDL

Theo ANSI-PARC (American National Standard Institue – Planning and Requirements Committee: Viện tiêu chuẩn quốc gia Mỹ - Ủy ban nhu cầu và kế hoạch Mỹ) có 3 mức biểu diễn một CSDL.

Mức vật lí (còn gọi là mức trong). Mức biểu diễn thấp nhất này mô tả dữ liệu được thực sự lưu trữ như thế nào trong CSDL. Đây là mức thể hiện các cài đặt có tính chất vật lí của CSDL để đạt được tối ưu trong các lần thực hiện các thao tác tìm kiếm và lưu trữ, để tận dụng được các vùng nhớ còn trống. Mức vật lý cũng là mức phản ánh các cấu trúc dữ liệu, các tổ chức tệp được dùng cho việc lưu trữ dữ liệu trên các thiết bị nhớ thứ cấp. Điều đó cũng có nghĩa là mức này tiếp xúc với các phương thức truy nhập của hệ điều hành để đặt dữ liệu vào các thiết bị nhớ, xây dựng các tập chỉ mục, truy xuất dữ liệu,… liên quan đến các vấn đề như sự cấp phát vùng nhớ cho dữ liệu và các chỉ mục, các mô tả bản ghi để lưu trữ, các kĩ thuật nén dữ liệu và giải mã dữ liệu.

Mức logic (còn gọi là mức khái niệm). Đây là mức mô tả những dữ liệu nào được lưu trữ CSDL và có những mối quan hệ nào giữa chúng. Nói một cách cụ thể hơn, mức logic biểu diễn các thực thể (trong thế giới thực tại), các thuộc tính và các mối quan hệ giữa các thực thể đó; mức logic cũng cho thấy các ràng buộc trên dữ liệu, các thông tin về ngữ nghĩa của dữ liệu, các thông tin về an ninh và toàn vẹn của dữ liệu. Tuy nhiên mức biểu diễn này chỉ quan tâm đến cái gì được lưu trữ trong CSDL chứ không quan tâm đến cách thức lưu trữ.



Hình 1.1. Ba mức của sự biểu diễn dữ liệu

Mức khung nhìn (còn gọi là mức ngoài). Mức biểu diễn cao nhất này mô tả chỉ một phần của toàn bộ CSDL, phần thích hợp với một người sử dụng nhất định. Mức này gồm một số khung nhìn của những người sử dụng đặt vào CSDL. Mỗi người dùng có thể không quan tâm đến toàn bộ thông tin của hệ CSDL mà chỉ cần một phần thông tin nào đó. Họ có một cách nhìn thế giới thực tại theo cách nhìn gần gũi và phù hợp với họ. Khung nhìn dành cho người sử dụng đó chỉ gồm những thực thể cùng những thuộc tính, những mối quan hệ của những thực thể mà họ quan tâm. Các khung nhìn khác nhau cũng có thể trình bày cùng một dữ liệu nhưng ở những khuôn dạng khác nhau. Ví dụ người sử dụng này có thể nhìn thấy thông tin ngày theo kiểu (họ, tên) còn người sử dụng khác lại ở kiểu (tên, họ). Một số khung nhìn có thể chứa các dữ liệu suy dẫn ra được hay tính toán được, những dữ liệu này vốn không được thực sự lưu trữ trong CSDL nhưng được tạo ra khi cần đến.

Tóm lại, mức khung nhìn là cách cảm nhận của người dùng về dữ liệu, mức vật lý là cách nhìn nhận của hệ QTCSDL và hệ điều hành về dữ liệu. Mức logic nằm giữa mức khung nhìn và mức vật lý, có thể coi đây là cách cảm nhận của toàn thể cộng đồng người dùng về dữ liệu. Tại mức logic tồn tại cả hai ánh xạ đến hai mức còn lại, tạo nên một sự độc lập đối với nhau của hai mức đó.

Có thể thấy *mục đích của kiến trúc ba mức* nêu trên chính là sự tách biệt quan niệm về CSDL của nhiều người sử dụng với những chi tiết biểu diễn về vật lý của CSDL . Điều đó dẫn đến những thuận lợi sau:

Đối với một CSDL, mỗi người dùng có một khung nhìn riêng của mình. Họ có thể thay đổi khung nhìn của họ và sự thay đổi này không làm ảnh hưởng đến những khung nhìn dữ liệu của người dùng khác đang dùng chung CSDL này.

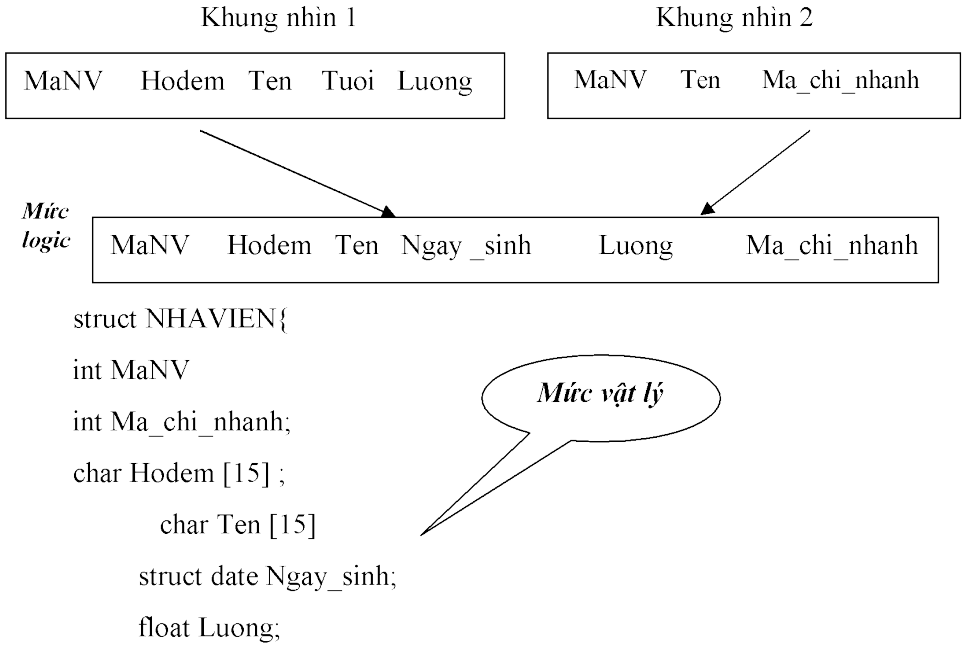
Những tương tác của người dùng với CSDL không phụ thuộc vào những vấn đề chi tiết trong lưu trữ dữ liệu (chẳng hạn như vấn đề chỉ mục hóa hay bảng băm).

Người quản trị CSDL (Database Administrator, thường viết tắt là DBA) có thể thay đổi cấu trúc khái niệm của CSDL mà không làm ảnh hưởng đến tất cả người dùng.

1.3. Lược đồ và thể hiện của CSDL

Toàn bộ mô tả CSDL được gọi là **lược đồ CSDL** (database schema). Tương ứng với ba mức biểu diễn dữ liệu nói trên có ba loại lược đồ. Ở mức cao nhất chúng ta có nhiều lược đồ ngoài (còn gọi là lược đồ con) cho những cách nhìn dữ liệu khác nhau của những người sử dụng khác nhau. Ở mức logic chúng ta có lược đồ logic. Ở mức thấp nhất chúng ta có lược đồ vật lí. Thường thì các hệ CSDL hỗ trợ một lược đồ vật lí, một lược đồ logic và nhiều lược đồ con.

Một điều quan trọng là cần phân biệt mô tả của CSDL (tức là lược đồ CSDL) với bản thân CSDL. Lược đồ được xác định trong quá trình thiết kế CSDL và người ta không muốn nó thay đổi thường xuyên. Trong khi đó bản thân CSDL sẽ thay đổi theo thời gian do dữ liệu được thêm vào, xóa đi hay sửa đổi. Toàn bộ dữ liệu lưu trữ trong CSDL tại một thời điểm nhất định được gọi là một *thể hiện* của CSDL (database instance). Như vậy nhiều thể hiện của CSDL có thể tương ứng với cùng một lược đồ CSDL . Đôi khi lược đồ còn được gọi là *nội hàm* (instension) của CSDL và một thể hiện còn được gọi là một *mở rộng* hay một *trạng thái* (extension hay state) của CSDL .



struct NHANVIEN *next; /*con trỏ đến bản ghi tiếp của tệp NHANVIEN*};

index MaNV; index Ma_chi_nhanh; /*xác định các chỉ mục cho tệp NHANVIEN*/

1.4. Sự độc lập của dữ liệu

Có thể nói một cách khác về mục đích của kiến trúc ba mức của CSDL mà chúng ta vừa nói ở trên (ở mục 2), đó là sự độc lập dữ liệu (data independence), hiểu theo nghĩa các lược đồ ở mức trên không bị ảnh hưởng khi có sự thay đổi các lược đồ ở các mức dưới. Có hai loại độc lập dữ liệu.

Độc lập dữ liệu vật lí là khả năng sửa đổi lược đồ vật lí mà không làm thay đổi lược đồ khái niệm và như vậy cũng không đòi hỏi viết lại các trình ứng dụng. Để tăng tính hiệu quả, nhiều khi cần có những thay đổi ở mức vật lí. Chẳng hạn như sử dụng các tổ chức tệp khác trước, dùng thiết bị nhớ khác, thay đổi các chỉ mục hay thay đổi thuật toán băm.

Độc lập dữ liệu logic là khả năng sửa đổi lược đồ logic mà không làm thay đổi các khung nhìn (các lược đồ ngoài), cũng có nghĩa là không đòi hỏi viết lại các trình ứng dụng. Các sửa đổi ở mức logic là cần thiết mỗi khi cấu trúc logic của CSDL cần phải thay đổi, chẳng hạn cần thêm hay bớt các thực thể nào đó, các thuộc tính hay các mối quan hệ của chúng. Dĩ nhiên những người dùng có chạm đến những thông tin đã thay đổi này sẽ được thông báo về sự thay đổi nhưng điều quan trọng là những người dùng khác sẽ không bị ảnh hưởng gì.

Độc lập dữ liệu logic khó thực hiện hơn độc lập dữ liệu vật lí vì các chương trình ứng dụng phụ thuộc nhiều vào cấu trúc logic của dữ liệu mà chúng truy cập.

Khái niệm độc lập dữ liệu trong nhiều mặt tương tự với khái niệm *kiểu dữ liệu trừu tượng* trong các ngôn ngữ lập trình hiện đại. Cả hai đều che dấu người dùng những chi tiết cài đặt, cho phép người dùng tập trung vào cấu trúc chung hơn là tập trung vào các chi tiết cài đặt ở mức thấp.

1.5. Những cách tiếp cận một CSDL

Trên thực tế, một lược đồ được viết trong ngôn ngữ định nghĩa dữ liệu của một hệ quản trị CSDL cụ thể. Để mô tả các yêu cầu dữ liệu của một tổ chức sao cho mô tả đó dễ hiểu đối với nhiều người sử dụng khác nhau thì ngôn ngữ này lại

ở mức quá thấp. Như vậy cần phải có một mô tả lược đồ ở mức cao hơn, nói cách khác cần phải có một mô hình dữ liệu.

Mô hình dữ liệu là một tập các khái niệm và kí pháp dùng để mô tả dữ liệu, các mối quan hệ của dữ liệu, các ràng buộc trên dữ liệu của một tổ chức.

Như vậy có thể xem như một mô hình dữ liệu có ba thành phần:

1. Phần mô tả cấu trúc của CSDL.
2. Phần mô tả các thao tác, định nghĩa các phép toán được phép trên dữ liệu.
3. Phần mô tả các ràng buộc toàn vẹn để đảm bảo sự chính xác của dữ liệu.

Khi dùng mô hình dữ liệu chúng ta có thể biểu diễn dữ liệu theo một cách dễ hiểu và vì vậy mô hình cũng được sử dụng trong việc thiết kế CSDL.

Đã có nhiều mô hình dữ liệu được đề xuất và có thể chia thành ba nhóm theo các cách tiếp cận mô hình hóa như sau:

+ Mô hình (dữ liệu) logic trên cơ sở đối tượng.

Chẳng hạn mô hình quan hệ thực thể, mô hình hướng đối tượng… Các mô hình thuộc nhóm này thường được dùng trong việc mô tả dữ liệu ở mức logic và khung nhìn. Chúng cung cấp các khả năng cấu trúc rất mềm dẻo và cho phép các ràng buộc được đặc tả tường minh. Chúng ta sẽ bàn bạc kỹ hơn về các mô hình này trong các chương sau.

+ Mô hình (dữ liệu) logic trên cơ sở bản ghi.

Các mô hình này thường được dùng trong việc mô tả dữ liệu ở mức khung nhìn và logic. Chúng được dùng mô tả cấu trúc logic tổng thể của CSDL, đồng thời cung cấp một mức cao hơn của sự cài đặt. Trong các mô hình này CSDL được cấu trúc thành các bản ghi có khuôn dạng cố định gồm một số trường (thuộc tính) có thể thuộc nhiều kiểu dữ liệu.

Các mô hình logic quen thuộc được dùng là: Mô hình quan hệ (relational data model), mô hình mạng, mô hình phân cấp. Các mô hình này cũng sẽ được trình bày kỹ trong chương sau.

+ Mô hình (dữ liệu) vật lí.

Các mô hình logic tập trung vào bản chất logic của biểu diễn dữ liệu, tập trung vào cái được biểu diễn trong CSDL, còn được gọi là các mô hình dữ liệu bậc cao. Các mô hình vật lí tập trung vào những chi tiết cho biết dữ liệu được lưu trữ thế nào, còn được gọi là các mô hình dữ liệu bậc thấp.

Các mô hình dữ liệu vật lý mô tả dữ liệu được lưu trữ thế nào trên máy tính, mô tả cấu trúc bản ghi, thứ tự các bản ghi và con đường truy cập. Hai mô hình dữ liệu vật lý quen dùng là mô hình hợp nhất và mô hình bộ nhớ-khung. Mô hình này thường chỉ phù hợp với các chuyên gia lập trình chứ không cần thiết đối với đa số người sử dụng.

1.6. Hệ quản trị cơ sở dữ liệu (DBMS)

Hệ chương trình được xây dựng để giúp người sử dụng định nghĩa, tạo lập, xử lý, bảo trì các CSDL và cung cấp các truy cập có điều khiển đến các CSDL này gọi là hệ quản trị cơ sở dữ liệu (DataBase Management System-DBMS), viết tắt là DBMS.

DBMS cung cấp cho người sử dụng các ***phương tiện*** sau:

a. Cung cấp 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 từ 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).

b. Các kiểm soát, các điều khiển đối với truy cập vào CSDL

Bao gồm:

+ 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 đảm bảo. Tuy nhiên, do sự hiện diện của các sự cố, một giao dịch có thể không hoàn tất 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=103 Gigabytes=106 Megabytes). Các thông tin phải được lưu trữ trên lưu trữ ngoài (đĩa). Dữ liệu được di chuyển giữa đĩa lưu trữ và bộ nhớ chính khi cần thiết. Do việc di chuyển dữ liệu từ đĩa lên bộ nhớ 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 tâm 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.

Việc phân biệt một DBMS và các phần mềm khác được dựa trên hai ***đặc tính*** sau:

- *Khả năng quản lý một bộ dữ liệu bền.*

- *Khả năng truy cập có hiệu quả một số lượng lớn dữ liệu.*

Ngoài hai chức năng trên được coi như là hai chức năng cơ bản, các DBMS thương mại còn có một số ***chức năng*** sau:

- Hỗ trợ ít nhất một mô hình dữ liệu, hay một sự trừu tượng hoá toán học để thông qua đó người sử dụng có thể quan sát dữ liệu.

- Hỗ trợ cho một ngôn ngữ bậc cao nào đó để cho phép người sử dụng định nghĩa cấu trúc của dữ liệu, truy cập dữ liệu và thao tác dữ liệu.

- Cung cấp một từ điển dữ liệu, đó là các mô tả về dữ liệu được lưu trữ. Dữ liệu trong từ điển dữ liệu được gọi là "siêu dữ liệu" (meta-data), thông thường một từ điển dữ liệu cất giữ tên, kiểu, kích thước các bản ghi, tên của các mối quan hệ, các ràng buộc toàn vẹn trên dữ liệu, tên các người có quyền truy cập, các lược đồ trong, lược đồ khái niệm, lược đồ ngoài và các ánh xạ giữa chúng. Thường các meta-data là các thủ tục lưu trữ do kỹ sư CSDL lập nên cho một CSDL cụ thể.

- Quản lý giao tác (transaction) cung cấp một cơ chế đảm bảo tất cả các cập nhật trong một giao tác được thực hiện hoặc tất cả được hủy bỏ, để đảm bảo tính nhất quán dữ liệu.

- Cung cấp các dịch vụ điều khiển tương tranh nhằm cho phép nhiều người sử dụng truy cập đồng thời, chính xác vào CSDL vào cùng một thời điểm.

- Điều khiển truy cập, đó là khả năng giới hạn việc truy cập vào CSDL của những người sử dụng trái phép và khả năng kiểm tra tính hợp lệ của dữ liệu.

- Có khả năng phục hồi lại dữ liệu khi hệ thống bị hỏng.

- Hỗ trợ truyền thông dữ liệu.

- Bảo đảm tính toàn vẹn dữ liệu, có nghĩa là đảm bảo dữ liệu trong CSDL và những thay đổi dữ liệu phải tuân theo những luật xác định nhằm đảm bảo sự chính xác và nhất quán của dữ liệu được lưu trữ.

Ngoài ra DBMS còn cung cấp các dịch vụ *hỗ trợ tính độc lập dữ liệu* và các *dịch vụ tiện ích.*

1.6.1. Kiến trúc của một hệ quản trị cơ sở dữ liệu

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.

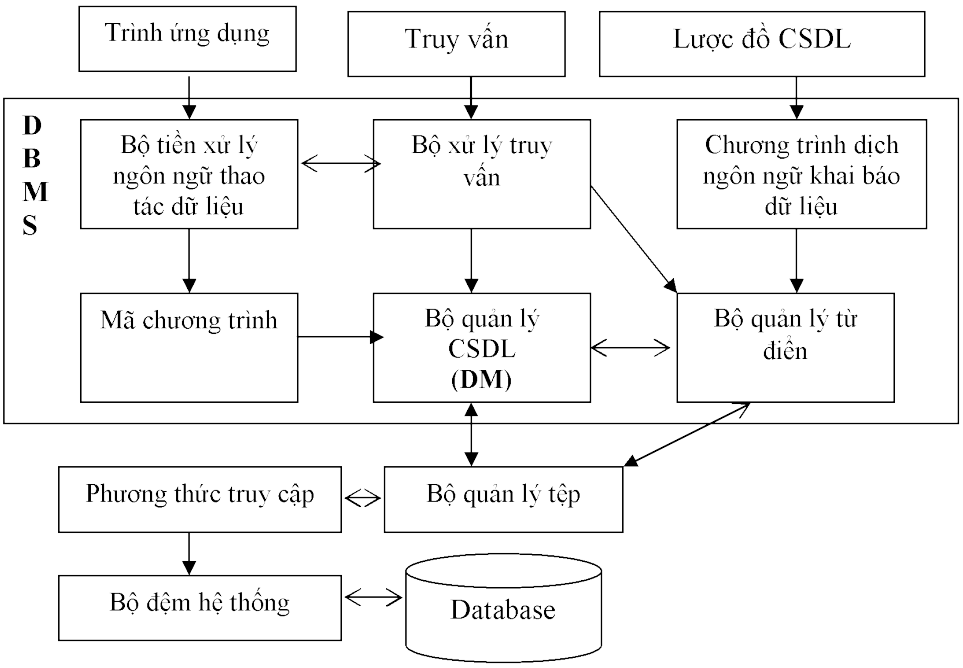
Hơn nữa, một số cấu trúc dữ liệu được cần đến như bộ phận của sự thực thi hệ thống vật lý:

- Các file dữ liệu: Lưu trữ CSDL.

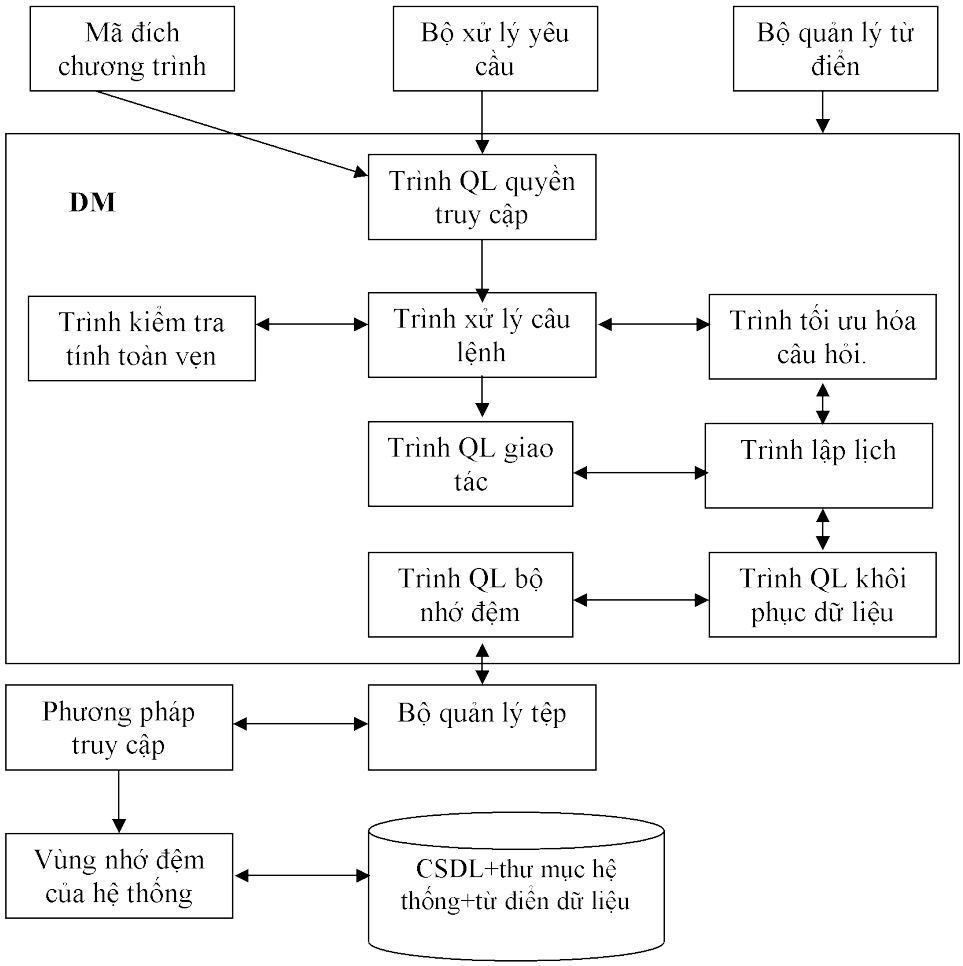
- Tự điển dữ liệu (Data Dictionary): lưu metadata về cấu trúc CSDL. Một đặc trưng cơ bản của giải pháp cơ sở dữ liệu là hệ thống cơ sở dữ liệu không chỉ gồm có bản thân cơ sở dữ liệu mà còn có cả định nghĩa hoặc mô tả đầy đủ về cấu trúc cơ sở dữ liệu và các ràng buộc. Định nghĩa này được lưu trữ trong từ điển hệ thống, nó chứa các thông tin như là cấu trúc của mỗi tệp, kiểu và dạng lưu trữ của từng mục dữ liệu. Các thông tin được lưu giữ trong từ điển gọi là siêu dữ liệu (meta-data) và chúng mô tả cấu trúc của dữ liệu nguyên thủy. Phần mềm hệ quản trị cơ sở dữ liệu và những người sử dụng cơ sở dữ liệu sử dụng từ điển để lấy thông tin về cấu trúc của cơ sở dữ liệu.

- Chỉ mục (Indices): cung cấp truy xuất nhanh đến các hạng mục dữ liệu chứa các giá trị tìm kiếm.

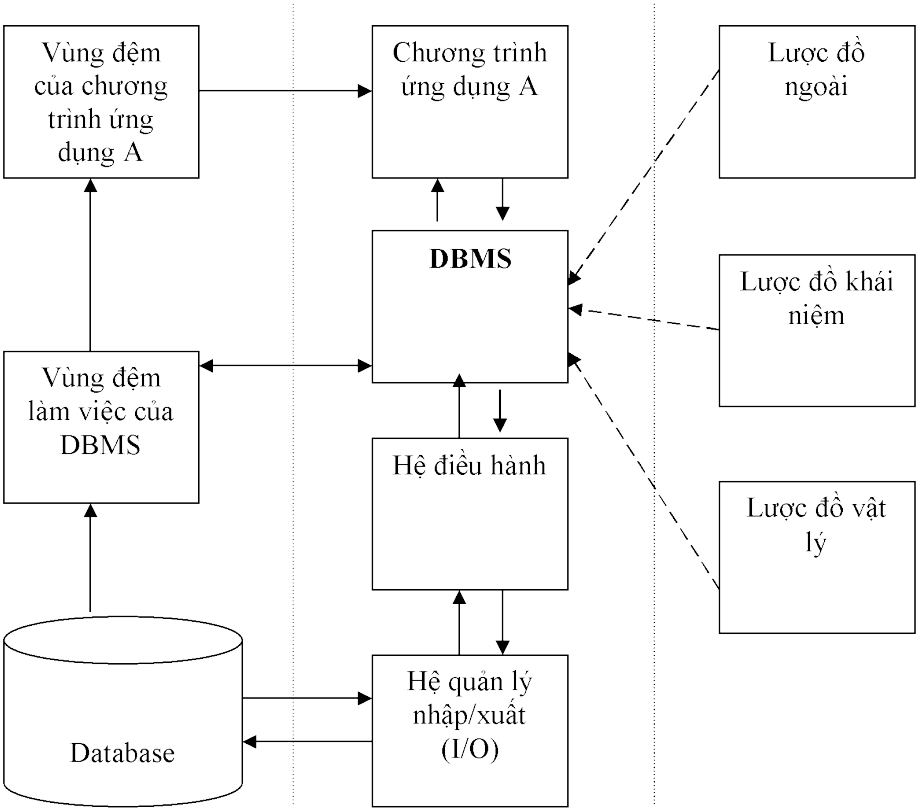
- Dữ liệu thống kê (Statistical data): lưu trữ thông tin thống kê về dữ liệu trong cơ sở dữ liệu. Thông tin này được dùng bởi bộ xử lý vấn tin để chọn những phương pháp hiệu quả thực hiện câu vấn tin.



Hình 1.2. Các thành phần chính của một DBMS



Hình 1.3. Bộ quản lý cơ sở dữ liệu



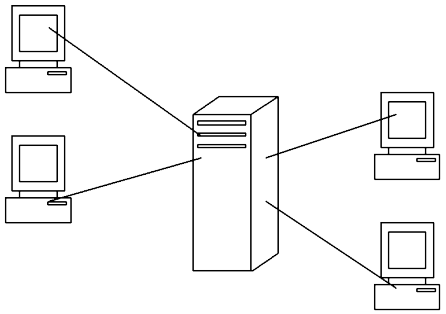
Hình 1.4. Hoạt động một chương trình ứng dụng thông qua một DBMS

1.6.2. Sơ lược về các kiến trúc hệ quản trị CSDL đa người dùng

Kiến trúc truyền thống của một hệ thống đa người dùng có tên gọi là hệ xử lí từ xa (teleprocesssing). Một hệ xử lí từ xa bao gồm một máy tính (một CPU) và một số trạm đầu cuối (terminals) như trong hình 1.5.

Tất cả các xử lí đều được thực hiện trên cùng một máy tính (về phương diện vật lí). Các trạm đầu cuối được nối với máy tính trung tâm, chúng gửi các thông điệp yêu cầu đến chương trình ứng dụng của người dùng (khi sử dụng các dịch vụ của hệ QLCSDL), nhờ vào hệ thống điều khiển truyền thông của hệ điều hành. Các thông điệp gửi trả về cho trạm đầu cuối của người dùng cũng theo con đường đó. Có thể thấy rằng trong kiến trúc này máy tính trung tâm không những phải chạy các trình ứng dụng và hệ QTCSDL mà còn phải thực hiện một lượng

lớn công việc thay cho các trạm đầu cuối. Trong những năm qua con người đã đạt được những thành công lớn trong việc nâng cao khả năng của máy tính cá nhân (PC) và phát triển công nghệ mạng máy tính. Điều này dẫn chúng ta đến khuynh hướng thay thế những máy tính lớn (mainhframe) đắt tiền bởi các mạng máy tính PC đạt hiệu quả không kém với giá cả phải chăng hơn. Khuynh hướng này làm nảy sinh hai loại kiến trúc mới cho hệ thống đa người dùng, đó là kiến trúc file – server và kiến trúc client – server.



Hình 1.5. Kiến trúc hệ xử lý từ xa

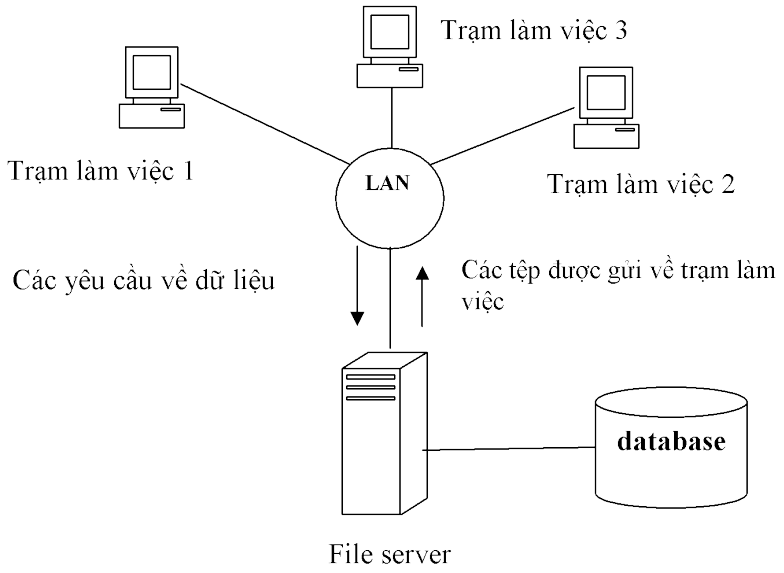
1.6.2.1. Kiến trúc máy chủ - tệp (file – server)

Trong kiến trúc máy chủ - tệp các xử lí không tập trung vào một máy tính trung tâm mà được phân tán trên mạng, thường là mạng cục bộ (LAN). Hình 1.6 giới thiệu kiến trúc này.

File – server lưu giữ các tệp dữ liệu mà các ứng dụng và hệ QTCSDL cần đến. Tuy nhiên các ứng dụng và hệ QTCSDL chạy trên mỗi trạm làm việc (workstation) và yêu cầu các tệp dữ liệu ở file – server khi cần đến. File – server hoạt động đơn giản như một đĩa cứng chứa dữ liệu có thể chia sẻ. Hệ QTCSDL trên mỗi trạm làm việc gửi yêu cầu đến file – server đòi hỏi những dữ liệu lưu trên đĩa mà hệ QTCSDL cần. Như vậy kiến trúc này sẽ có những nhược điểm chính sau đây:

1. Lượng dữ liệu truyền qua lại trên mạng rất nhiều.
2. Mỗi trạm làm việc phải cài đặt một bản sao đầy đủ của hệ QTCSDL.

3. Việc giải quyết các vấn đề tương tranh, khôi phục dữ liệu và bảo đảm tính nhất quán của dữ liệu sẽ phức tạp hơn do nhiều hệ QTCSDL truy cập vào cùng các tệp dữ liệu.



Hình 1.6. Kiến trúc máy chủ - tệp (file server)

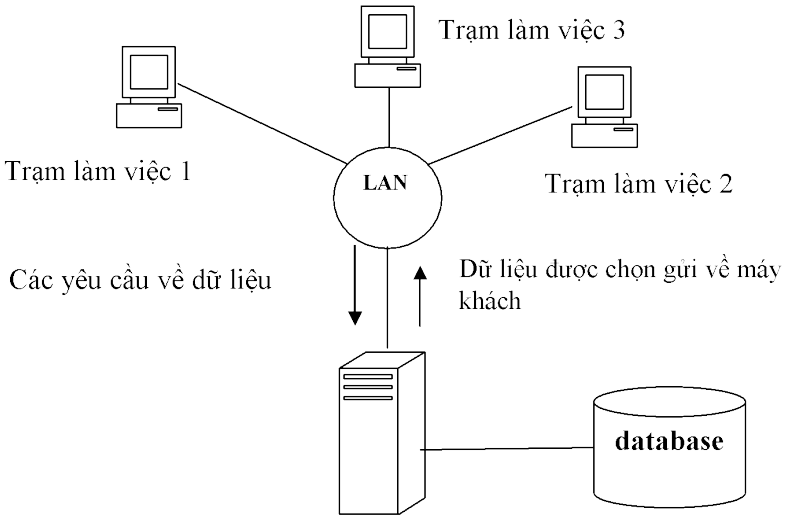
1.6.2.2. Kiến trúc máy khách – máy chủ (client – server)

Kiến trúc máy khách – máy chủ (có thể gọi tắt là kiến trúc khách – chủ) khắc phục được các nhược điểm của hai loại kiến trúc nêu trên. Trong kiến trúc máy khách – máy chủ các bộ phận phần mềm tương tác với nhau tạo nên hệ thống: tiến trình máy khách yêu cầu cung cấp tài nguyên nào đó và tiến trình máy chủ cung cấp tài nguyên đó. Hai tiến trình này không nhất thiết phải nằm trên cùng một máy tính, trên thực tế thường có một tiến trình máy chủ đặt tại một điểm (site) của mạng (cục bộ) còn các tiến trình máy khách đặt tại các điểm khác của mạng. Hình 1.7 minh họa kiến trúc máy khách/chủ.

Trong ngữ cảnh sơ sở dữ liệu, tiến trình client quản trị giao diện người dùng và ứng dụng logic, nó hoạt động như một trạm làm việc cao cấp theo nghĩa trên đó chạy các ứng dụng CSDL. Tiến trình client nhận yêu cầu của người dùng, kiểm tra cú pháp và sinh ra các câu truy vấn trong ngôn ngữ SQL hoặc trong ngôn ngữ CSDL thích hợp khác. Tiếp theo, tiến trình client gửi thông điệp đến server, chờ

nhận trả lời và định dạng dữ liệu trả ra cho người sử dụng đầu cuối. Còn tiến trình server thì tiếp nhận và xử lí các yêu cầu về cơ sở dữ liệu rồi gửi trả kết quả về lại cho client. Tiến trình server bao gồm luôn cả việc kiểm tra quyền truy cập dữ liệu, đảm bảo tính toàn vẹn dữ liệu, bảo trì hệ thống từ điển, thực hiện các truy vấn và các tiến trình cập nhật. Ngoài ra nó còn cung cấp các dịch vụ điều khiển tương tranh và khôi phục dữ liệu. Có thể kể đến một số ưu điểm của kiến trúc loại này như sau:

- Khả năng truy cập rộng rãi đến các CSDL.
- Nâng cao khả năng thực hiện: nếu tiến trình server và các tiến trình client ở trên các máy tính khác nhau thì các CPU khác nhau có thể cùng chạy song song, mỗi CPU thực hiện tiến trình của nó.
- Chi phí cho phần cứng có thể được giảm do chỉ cần server có cấu hình đủ mạnh để lưu trữ và quản trị cơ sở dữ liệu.
- Chi phí cho truyền thông được giảm do một phần trong các thao tác của ứng dụng được giải quyết trên client, truyền thông trên mạng chỉ gồm: yêu cầu về truy cập cơ sở dữ liệu của client gửi đến server và dữ liệu kết quả từ server gửi cho client.
- Nâng cao được khả năng đảm bảo tính nhất quán của dữ liệu. Server có thể kiểm soát được tính toàn vẹn bởi các ràng buộc này được định và kiểm tra chỉ tại đó.
- Kiến trúc này phù hợp với việc xây dựng các hệ thống có tính mở.



Hình 1.7. File Server (với DBMS)

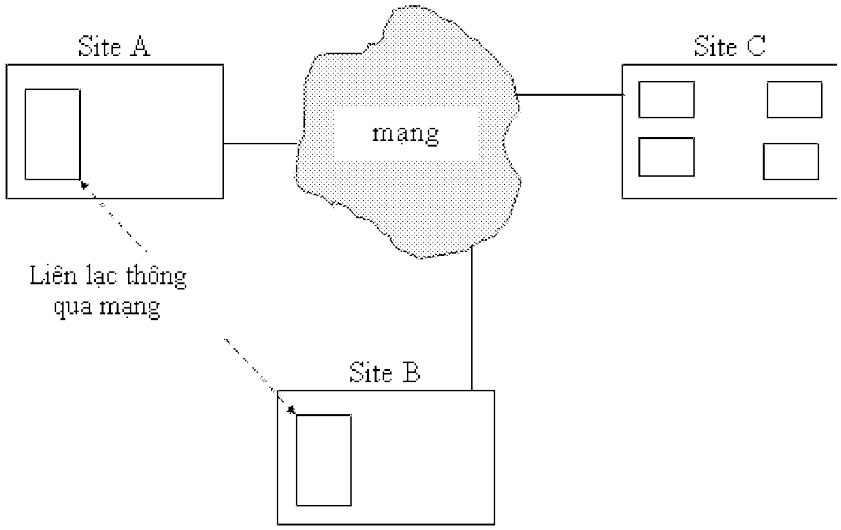
1.6.2.3. Các hệ song song (Parallel Systems)

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.

Cụ thể về quá trình tăng tốc độ và tăng quy mô, các 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, các kiến trúc của hệ thống song song… Có thể xem thêm trong [2].

1.6.2.4. 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.



Hình 1.8. Hệ thống phân tán

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.

1.7. Vai trò của con người trong hệ CSDL

Với các CSDL nhỏ, có thể chỉ có một người tạo lập (định nghĩa) và thao tác trên đó. Nhưng với những CSDL lớn, có thể có sự tham gia của nhiều người vào việc xây dựng, bảo trì và sử dụng nó.

1.7.1. Người quản trị CSDL

Người quản trị CSDL (Database Administrator - DBA) là người có trách nhiệm quản lí các tài nguyên của hệ CSDL các tài nguyên đó là: CSDL, hệ quản trị CSDL và các phần mềm liên quan. Đây là người có vai trò thiết kế và cài đặt CSDL về mặt vật lí, cấp phát các quyền truy cập CSDL, cấp phần mềm và phần cứng theo yêu cầu, duy trì các hoạt động hệ thống đảm bảo thỏa mãn yêu cầu của các trình ứng dụng và của người dùng. Như vậy DBA phải là người có những hiểu biết chi tiết và kĩ năng về mặt kĩ thuật trong lĩnh vực CSDL, hệ quản trị CSDL và môi trường hệ thống.

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.

1.7.2. Người thiết kế CSDL

Trong những dự án thiết kế các CSDL lớn người ta có thể phân biệt hai nhóm người thiết kế: thiết kế CSDL logic và thiết kế CSDL vật lí.

Người thiết kế CSDL logic có trách nhiệm xác định dữ liệu được lưu trữ trong CSDL (các thực thể và các thuộc tính), xác định các mối quan hệ giữa dữ liệu, các ràng buộc trên dữ liệu được lưu trữ. Như vậy người thiết kế CSDL logic phải có hiểu biết thấu suốt và đầy đủ về dữ liệu của tổ chức (thế giới nhỏ mà CSDL phản ánh) và các luật làm việc của tổ chức đó. Để việc thiết kế có hiệu quả, người thiết kế CSDL logic phải giao tiếp với những người dùng CSDL trong tương lai, hiểu được nhu cầu sử dụng của họ để có thể đưa ra một thiết kế phù hợp.

Người thiết kế CSDL vật lí chọn mô hình dữ liệu logic và quyết định nó được thực hiện về mặt lí luận như thế nào. Như vậy việc thiết kế CSDL vật lí liên quan đến những vấn đề như ánh xạ mô hình dữ liệu logic vào tập các bảng và các ràng buộc toàn vẹn; chọn lựa cấu trúc lưu trữ và phương thức truy cập dữ liệu để đạt hiệu quả cao khi thực hiện các thao tác trên CSDL; thiết kế những hệ thống an ninh nào đó cho dữ liệu.

Có thể thấy người thiết kế CSDL logic liên quan đến câu hỏi "cái gì được lưu trữ trong CSDL?", còn người thiết kế CSDL vật lí liên quan đến câu hỏi "lưu trữ như thế nào?".

1.7.3. Người lập trình ứng dụng

Khi CSDL đã được cài đặt thì các chương trình ứng dụng đáp ứng nhu cầu khai thác CSDL của người dùng sẽ được cài đặt. Đây chính là công việc của người viết các chương trình ứng dụng (Application Programmer). Thông thường người lập trình ứng dụng thể hiện các đặc tả của người phân tích thiết kế hệ thống thành chương trình. Mỗi chương trình bao gồm các câu lệnh yêu cầu hệ quản trị CSDL thực hiện một số thao tác trên CSDL (truy xuất dữ liệu, thêm, xóa, sửa đổi dữ liệu). Nhóm người lập trình ứng dụng có thể bao gồm:

- Người phân tích hệ thống.

+ Xác định các yêu cầu của người dùng, đặc biệt là những người sử dụng thường xuyên.

+ Phát triển các đặc tả cho các giao tác được đóng gói.

- Lập trình viên

+ Cài đặt các đặc tả thành các chương trình.

+ Kiểm tra, gỡ lỗi, viết tài liệu và bảo trì.

- Nhóm người hỗ trợ

+ Thiết kế và cài đặt HQT CSDL.

+ Phát triển các công cụ hỗ trợ thiết kế và tăng hiệu năng.

+ Vận hành và duy trì môi trường cho hệ CSDL.

1.7.4. Người sử dụng đầu cuối

Người sử dụng chính là khách hàng của CSDL, bởi CSDL được thiết kế, cài đặt và bảo trì để cung cấp những thông tin họ cần. Theo cách sử dụng CSDL, có thể chia người sử dụng đầu cuối thành các nhóm: người sử dụng đơn giản (còn gọi là người sử dụng ngây thơ) và người sử dụng tinh tế.

Người sử dụng đơn giản thường là những người không có hiểu biết sâu sắc về CSDL, họ truy cập vào CSDL thông qua các chương trình ứng dụng để thực hiện những thao tác đơn giản. Họ yêu cầu thực hiện những thao tác này bằng cách đưa vào những câu lệnh đơn giản hay chọn các mục trên bảng chọn (menu).Nói một cách khác họ sử dụng các giao dịch định sẵn.

Người sử dụng tinh tế là những người hiểu biết về cấu trúc CSDL, về những tiện ích mà hệ quản trị CSDL cung cấp. Họ có thể sử dụng ngôn ngữ truy vấn bậc cao như SQL để thực hiện những thao tác cần thiết. Một số người sử dụng tinh tế thậm chí có thể viết chương trình ứng dụng mãn yêu cầu của họ, có thể liệt ra đó là những người phát triển công cụ và những người bảo trì các phần mềm ứng dụng chạy trên hệ quản trị CSDL.

Tóm tắt chương 1

* Một **cơ sở dữ liệu (CSDL)** là một tập hợp các dữ liệu có liên quan với nhau được lưu trữ trên các thiết bị nhớ thứ cấp (như băng từ, đĩa từ …) để đáp ứng nhu cầu khai thác thông tin của nhiều người sử dụng với nhiều mục đích khác nhau.

* Mục đích của kiến trúc ba mức nêu trên chính là sự tách biệt quan niệm về CSDL của nhiều người sử dụng với những chi tiết biểu diễn về vật lý của CSDL.

* Có thể nói một cách khác về mục đích của kiến trúc ba mức của CSDL là sự độc lập dữ liệu (data independence), hiểu theo nghĩa các lược đồ ở mức trên không bị ảnh hưởng khi có sự thay đổi các lược đồ ở các mức dưới.

* Toàn bộ mô tả CSDL được gọi là **lược đồ CSDL** (database schema). Một điều quan trọng là cần phân biệt mô tả của CSDL (tức là lược đồ CSDL) với bản thân CSDL. Lược đồ được xác định trong quá trình thiết kế CSDL. Toàn bộ dữ liệu lưu trữ trong CSDL tại một thời điểm nhất định được gọi là một *thể hiện* của CSDL (database instance). Lược đồ còn được gọi là *nội hàm* (instension) của CSDL và một thể hiện còn được gọi là một *mở rộng* hay một *trạng thái* (extension hay state) của CSDL.

* Hệ chương trình được xây dựng để giúp người sử dụng định nghĩa, tạo lập, xử lý, bảo trì các CSDL và cung cấp các truy cập có điều khiển đến các CSDL này gọi là hệ quản trị cơ sở dữ liệu (DataBase Management System-DBMS), viết tắt là DBMS.

* Với các CSDL nhỏ, có thể chỉ có một người tạo lập (định nghĩa) và thao tác trên đó. Nhưng với những CSDL lớn, có thể có sự tham gia của nhiều người vào việc xây dựng, bảo trì và sử dụng nó.

Câu hỏi ôn tập và bài tập chương 1

1. Định nghĩa lại khái niệm: dữ liệu, cấu trúc dữ liệu và cơ sở dữ liệu. Nhận xét định nghĩa cơ sở dữ liệu trong tài liệu này với các định nghĩa cơ sở dữ liệu ở một số tài liệu khác.

2. Giải thích sự khác nhau giữa độc lập dữ liệu vật lý và độc lập dữ liệu logic?

3. Liệt kê các chức năng và vai trò nhiệm vụ của người quản trị CSDL. Đối với mỗi nhiệm vụ, giải thích rõ những vấn đề nảy sinh nếu nhiệm vụ đó không được hoàn thành.

4. Cho ví dụ thự tiễn về các khái niệm sau:

Ba mức trừu tượng dữ liệu.

Sơ đồ và thể hiện dữ liệu.

5. Trong quá trình học ở các năm trước, chúng ta đã có xây dựng một số chương trình quản lý tệp tin khi học ngôn ngữ lập trình. Anh chị hãy nhận xét về các chức năng mà hệ thống tệp tin của mình đạt được. Nhận xét ưu khuyết điểm của hệ thống đó.

6. Mục tiêu của các hệ cơ sở dữ liệu? Ví dụ minh hoạ.

7. Khái niệm File có gì khác với khái niệm cơ sở dữ liệu, ví dụ minh họa?

8. Anh chị sẽ đóng vai trò gì trọng hệ cơ sở dữ liệu? Nêu các kỹ năng và kiến thức cần thiết cần có của mình?

9. Có thể viết một tiểu luận nhỏ với những yêu cầu sau:

+ *Với một hệ quản trị cơ sở dữ liệu đã học, anh chị hãy hệ thống hóa các câu lệnh liên quan đến các phương tiện và chức năng của hệ quản trị cơ sở dữ liệu ấy cung cấp, cho nhận xét.*

+ *Tìm hiểu các vấn đề, các bài toán liên quan đến CSDL phân tán, CSDL trong các hệ CSDL đa người dùng.*

CHƯƠNG 2. MÔ HÌNH CƠ SỞ DỮ LIỆU

Mục đích

Trình bày quá trình thiết kế CSDL và một số mô hình dữ liệu khái niệm bậc cao, đặc biệt lưu ý đến mô hình quan hệ thực thể.

Yêu cầu

- Nêu được ý nghĩa của các mô hình dữ liệu.

- Hiểu được các khái niệm và quy tắc cơ bản của mô hình quan hệ thực thể và mô hình quan hệ thực thể mở rộng.

- Thiết kế một số sơ đồ quan hệ quan hệ thực thể trên các bài toán thực tiễn.

- Nắm được sơ lược mối quan hệ giữa các mô hình dữ liệu bậc cao.

2.1. Mô hình dữ liệu khái niệm bậc cao và quá trình thiết kế CSDL

Quá trình thiết kế CSDL được minh họa ở hình 2.1. Bước ***thứ nhất*** là tập hợp các yêu cầu và phân tích. Kết quả của bước này là một tập hợp các yêu cầu của người dùng được ghi ở dạng súc tích. Những đặc tả như vậy càng chi tiết và càng đầy đủ càng tốt.

Bước ***thứ hai*** là thiết kế khái niệm. Ở bước này người thiết kế lựa chọn một mô hình dữ liệu, dùng các khái niệm của mô hình đã chọn để chuyển những đặc tả yêu cầu của người dùng (kết quả của bước thứ nhất) sang thành một lược đồ khái niệm. Lược đồ khái niệm là một mô tả cô đọng về yêu cầu dữ liệu của những người dùng bao gồm: mô tả chi tiết các kiểu dữ liệu, các quan hệ, các ràng buộc.

Các khái niệm do mô hình dữ liệu bậc cao cung cấp được sử dụng trong những mô tả của lược đồ khái niệm. Do những khái niệm đó không chứa các chi tiết cài đặt, chúng thường dễ hiểu và có thể được dùng để giao lưu với những người sử dụng. Lược đồ khái niệm bậc cao được dùng để đảm bảo rằng kết quả của quá trình thiết kế CSDL sẽ đáp ứng được tất cả đòi hỏi của người dùng và đảm bảo rằng những đòi hỏi đó không chứa mâu thuẫn. Với bước thiết kế này, người thiết kế CSDL có thể tập trung vào việc đặc tả các tính chất của dữ liệu mà chưa cần quan tâm đến các chi tiết về lưu trữ.

Một lược đồ khái niệm được thiết kế một cách đầy đủ cũng bao gồm cả những đặc tả yêu cầu về chức năng, đó là những thao tác (hay giao tác) được thực hiện trên dữ liệu. Điều này cũng giúp khẳng định rằng lược đồ khái niệm thỏa mãn những yêu cầu chức năng đã xác định.

Ở bước này cần phải:

– Quan tâm đến nội dung của CSDL.

– Chọn một mô hình dữ liệu và diễn đạt dữ liệu theo mô hình đó.

Các bước cơ bản của nó là:

– *Đầu vào:*

+ Kết quả của giai đoạn phân tích – phác thảo cấu trúc CSDL

+ Có thể là mô hình thực thể kết hợp, sơ đồ lớp đối tượng…, hoặc một cấu trúc phổ quát.

+ Tổng quát về dữ liệu/ thông tin

+ Mối quan hệ, phụ thuộc về ngữ nghĩa giữa chúng

– *Đầu ra:*

+ Lược đồ CSDL mức quan niệm

Nhằm đảm bảo lưu trữ thông tin đầy đủ và không dư thừa.

"Tốt" hơn, "phù hợp" hơn với yêu cầu của môi trường phát triển ứng dụng.

Bước ***thứ ba*** là thiết kế logic (hay còn gọi là ánh xạ mô hình dữ liệu). Ở bước này người thiết kế cài đặt cơ sở dữ liệu bằng một hệ QTCSDL. Hầu hết các QTCSDL dùng một mô hình dữ liệu thể hiện (chẳng hạn mô hình quan hệ hay mô hình hướng đối tượng….), do vậy lược đồ khái niệm được chuyển đổi từ mô hình dữ liệu bậc cao sang mô hình dữ liệu thể hiện. Kết quả của bước này là một lược đồ CSDL dưới dạng một mô hình dữ liệu thể hiện của hệ QTCSDL. Đây là bước trung gian giữa mức quan niệm và mức vật lý, chuẩn bị cho thiết kế mức vật lý, ở bước này ta quan tâm đến các vấn đề sau:

- Nhu cầu khai thác CSDL của ứng dụng như tần suất truy xuất, các con đường truy xuất chính

– Quan tâm đến mô hình CSDL của ứng dụng:

Các bước cơ bản của nó là:

– *Đầu vào*:

+ Lược đồ mức quan niệm

+ Thông tin về nhu cầu khai thác

– *Đầu ra*:

+ Cấu trúc CSDL đã chuyển đổi sang cấu trúc phù hợp với ứng dụng.

+ Đồ thị quan hệ, các con đường truy xuất dữ liệu.

Bước ***cuối cùng*** là thiết kế vật lí. Các đặc điểm về mặt vật lí của CSDL phải được đặc tả ở giai đoạn này, chúng bao gồm các cấu trúc lưu trữ bên trong và kiểu tổ chức tệp cho CSDL.

Cụ thể là: Chọn lựa cách cài đặt CSDL trên một hệ quản trị CSDL cụ thể; Tìm hiểu các đặc trưng và hỗ trợ của hệ quản trị; Quan tâm đến tính hiệu quả và tốc độ xử lý.

Các bước cơ bản là:

– *Đầu vào*:

+ Kết quả của giai đoạn thiết kế mức logic.

+ Thông tin về hệ quản trị.

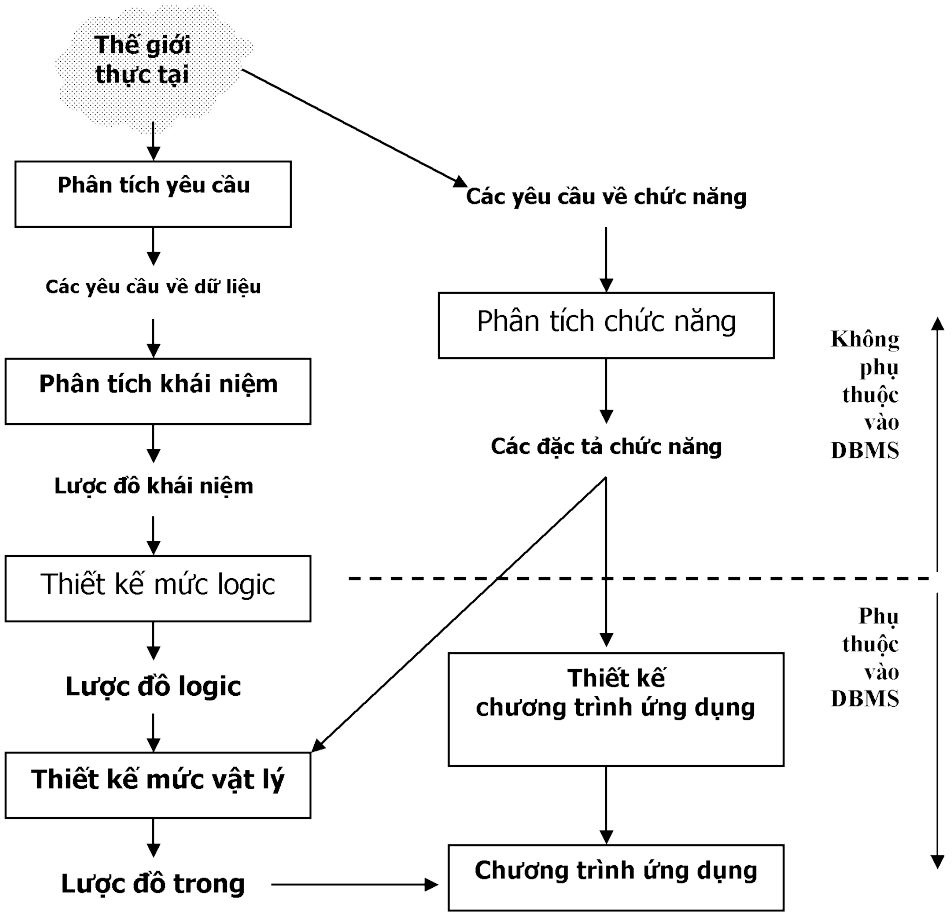
– *Đầu ra*:

+ Lược đồ CSDL hoàn chỉnh, sẵn sàng cài đặt

+ Khai báo khóa chính, khóa ngoại, chỉ mục

+ Xác định một số thông số, tùy chọn của CSDL mà hệ quản trị hỗ trợ.

+ Ràng buộc toàn vẹn và an toàn dữ liệu.



Hình 2.1. Quá trình thiết kế một CSDL trong quá trình thiết kế hệ thống thông tin

2.2. Mô hình quan hệ thực thể (the entity-relationship model)

Mô hình quan hệ thực thể, nhiều khi còn gọi là mô hình thực thể - liên kết (thực thể - kết hợp) hay mô hình thực thể - quan hệ. Mục đích của mô hình quan hệ thực thể là cho phép *mô tả lược đồ khái niệm, trừu tượng hóa cấu trúc* và *các ràng buộc* của một cơ sở dữ liệu mà không cần chú ý đến tính hiệu quả hay thiết kế vật lý như phần lớn ở các mô hình khác. Thông thường người ta thừa nhận rằng sơ đồ quan hệ thực thể đã được xây dựng và sau đó sẽ được chuyển thành lược đồ logic ở một mô hình khác mà trên đó các hệ CSDL thực sự sẽ được xây dựng.

Để dễ theo dõi các ví dụ, ta xét các phân tích yêu cầu của một công ty được nêu như sau: Công ty có nhiều phòng ban. Một phòng có duy nhất một tên, một mã số và một trưởng phòng. Một phòng có thể có nhiều trụ sở làm việc khác nhau. Cần lưu lại thời điểm trưởng phòng nhậm chức.

Công ty thực hiện nhiều dự án. Một dự án có duy nhất một tên, một mã số và một địa điểm triển khai. Một phòng có thể giám sát nhiều dự án.

Công ty có nhiều nhân viên. Một nhân viên có duy nhất một mã số, địa chỉ, mức lương, giới tính, ngày sinh và nhiều sở thích. Một nhân viên chỉ được làm việc trong một phòng. Một nhân viên có thể tham gia nhiều dự án.

Một nhân viên có nhiều thân nhân. Một thân nhân có tên, giới tính, ngày sinh và mối quan hệ với nhân viên. Người sử dụng muốn lưu lại số giờ tham gia dự án của 1 nhân viên trong 1 tuần và thông tin người trưởng phòng của 1 nhân viên.

Hãy xây dựng mô hình dữ liệu nhằm biểu diễn dữ liệu của công ty qua các yêu cầu dữ liệu được phân tích trên.

2.2.1. Các khái niệm trong mô hình quan hệ thực thể

✍ Các thực thể (entity)

Khái niệm thực thể không thể định nghĩa một cách hình thức, nó giống như điểm và đường thẳng trong hình học. Ta hiểu thực thể là một cái gì đó tồn tại và có khả năng phân biệt được với các thực thể khác. Chẳng hạn mỗi một người là một thực thể, mỗi một xe máy là một thực thể...

✍ Tập (kiểu) thực thể

Một nhóm bao gồm các thực thể có dạng tương tự nhau được gọi là một tập thực thể. Khái niệm tập thực thể tương tự như khái niệm lớp (class) còn khái niệm thực thể tương tự khái niệm đối tượng (object) trong lý thuyết lập trình hướng đối tượng.

Ví dụ

- 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

Việc chọn các thực thể như thế nào để nhóm thành một tập thực thể tuỳ thuộc vào từng mục đích khác nhau. Một trong những bước chính trong việc chọn một lược đồ cho thế giới thực là chọn ra các tập thực thể. Để xây dựng tập thực thể cần phải xác định được một tập hữu hạn các đặc tính khác nhau của các thực thể trong tập thực thể đó, các tính chất này được gọi là các thuộc tính.

✍ Thuộc tính và khoá

Các tập thực thể có các đặc tính được gọi là các *thuộc tính* (attribute). Mỗi một thực thể trong tập thực thể được xác định bởi một bộ dữ liệu của các thuộc tính thể hiện thông tin về thực thể đó. Ứng với mỗi một thuộc tính có một tập các giá trị cho thuộc tính đó gọi là *miền giá trị* của thuộc tính đó.

Ví dụ 1 Tập thực thể nhân viên có thể khai báo với các thuộc tính như tên (có miền giá trị là tập chuỗi các ký tự), tuổi (có miền giá trị tập số nguyên dương), lớp (có miền giá trị là tập chuỗi các ký tự).

Có thể phân loại các thuộc tính như sau:

- Thuộc tính đơn (nguyên tử) và thuộc tính gộp. Chẳng hạn thuộc tính giới tính là thuộc tính đơn, còn thuộc tính Họ tên (Họ, Đệm, Tên) là thuộc tính gộp.

- Thuộc tính đơn trị và thuộc tính đa trị. Chẳng hạn thuộc tính Mã số nhân viên là thuộc tính đơn trị, còn thuộc tính Sở thích là thuộc tính đa trị.

- Thuộc tính cơ sở và thuộc tính dẫn xuất. Chẳng hạn Ngày sinh và Tuổi.

Một thuộc tính hay một tập các thuộc tính mà những giá trị của nó xác định duy nhất mỗi thực thể trong một tập các thực thể được gọi là ***khoá (key)*** của tập thực thể đó. Về mặt nguyên tắc, mỗi tập thực thể đều phải có một khoá để đáp ứng được giả thiết là mỗi thực thể đều có khả năng phân biệt được với các thực thể khác. Việc chọn ra khoá của một tập thực thể nào đó có thể khác nhau tùy vào những hoàn cảnh khác nhau.

Lưu ý: Khóa có thể gồm nhiều thuộc tính và một kiểu thực thể có thể có nhiều hơn một khoá.

Ví dụ 2 Với tập thực thể người, nếu chúng ta xét tập thực thể này trên nước Việt Nam thì có thể chọn số chứng minh nhân dân của mỗi người để làm khoá. Nhưng nếu xét tập thực thể này trên nhiều nước thì không có gì để đảm bảo được rằng sẽ không có ít nhất hai người sẽ có cùng số chứng minh. Do đó khoá của mỗi người sẽ là tập hai thuộc tính là số chứng minh và quốc tịch của người đó.

2.2.2. Các mối quan hệ (liên kết)

Có thể định nghĩa phi hình thức về mối quan hệ và kiểu quan hệ như sau:

+ Quan hệ (Relationship) là sự kết hợp của 2 hoặc nhiều thực thể phân biệt theo một ý nghĩa nào đó.

Ví dụ: Nhân viên A tham gia vào Dự án X. Nhân viên A giám sát Nhân viên B.

+ Kiểu quan hệ (Relationship Type) là tập hợp các quan hệ cùng kiểu.

Ví dụ: Kiểu quan hệ THAMGIA, trong đó các thực thể NHANVIEN quan hệ với các thực thể DUAN.

Một kiểu quan hệ có thể có nhiều thuộc tính. Chẳng hạn thuộc tính Giờ của kiểu quan hệ THAMGIA.

Một cách hình thức thì một kiểu quan hệ R giữa n kiểu thực thể $E_1, \ldots, E_n$

$r_i \in R$ quan hệ là một n-bộ $(e_1, \ldots, e_n)$, trong đó $e_j \in E_j$.

Một quan hệ trên $E_1, \ldots, E_n$ là một tập con của tích Cartersian $E_1 \times \ldots \times E_n$.

+ Bậc của kiểu quan hệ: là số lượng kiểu thực thể tham gia vào quan hệ.

- Kiểu quan hệ nhị phân.

- Kiểu quan hệ tam phân hoặc có bậc cao hơn.

- Kiểu quan hệ đệ quy khi cùng một kiểu thực thể tham gia vào quan hệ với vai trò khác nhau. Ví dụ : quan hệ *giám sát* giữa các Nhân viên.

+ Lực lượng tối đa (tối thiểu) của quan hệ là số lượng tối đa (tối thiểu) các quan hệ mà một thực thể có thể tham gia.

+ Ràng buộc về tỉ lệ lực lượng (đối với quan hệ nhị phân) ở đây quan tâm đến tỉ số lực lượng tối đa của 2 kiểu thực thể tham gia vào quan hệ.

Gồm các loại sau :

- Quan hệ Isa.

- Quan hệ một-một.

- Quan hệ nhiều-một.

- Quan hệ nhiều-nhiều.

⇘ Quan hệ Isa

Ta nói tập thực thể A có quan hệ Isa với tập thực thể B nếu B là một tổng quát hoá của A, hay nói cách khác A là một dạng đặc biệt của B. Mục đích chính của việc khai báo mối quan hệ Isa giữa tập thực thể A và B là: A có thể thừa hưởng các thuộc tính của B nhưng có thể thêm một số thuộc tính khác. Tập thực thể A và B lần lượt được gọi là lớp con và lớp cha. Ta có một số đặc trưng của quan hệ Isa sau đây:

- Một thực thể của lớp con cũng là một thực thể của lớp cha.

- Một thực thể của lớp cha có thể là thực thể của nhiều lớp con.

- Một thực thể của lớp con kế thừa mọi thuộc tính và mọi mối quan hệ của thực thể lớp cha.

- Một thực thể của lớp con có thể có các thuộc tính và các mối quan hệ của riêng nó.

Ví dụ 3: Giả sử ta xét tập thực thể B gồm các nhân viên làm việc trong một công ty nào đó và A là tập thực thể gồm những người quản lý các phòng ban trong công ty. Khi đó tập thực thể A sẽ có mối quan hệ Isa với tập thực thể B.

✍ Mối quan hệ nhiều-một.

Một mối quan hệ được gọi là nhiều-một từ tập thực thể E_1 vào tập thực thể E_2 nếu như: mỗi một thực thể trong tập thực thể E_2 được quan hệ với không hay nhiều thực thể trong tập thực thể E_1, nhưng mỗi thực thể trong E_1 chỉ được kết hợp với nhiều nhất một thực thể trong E_2. Do vậy mối quan hệ này là một hàm từ E_1 đến E_2.

Khái niệm mối quan hệ nhiều-một được tổng quát hoá thành các mối quan hệ giữa ba tập thực thể trở lên. Nếu có một mối quan hệ R giữa các tập $E_1,E_2,...,E_k$ và nếu các thực thể cho trước của tất cả các tập trừ E_i có nhiều nhất một thực thể có quan hệ với thực thể E_i thì ta gọi R là mối quan hệ nhiều-một từ $E_1,...,E_{i-1},E_{i+1},...,E_k$ đến E_i.

Ví dụ 4 Gọi A là tập thực thể gồm các nhân viên trong một công ty và B là tập thực thể các phòng ban trong công ty. Khi đó mối quan hệ làm việc giữa A và B là mối quan hệ nhiều-một (do mỗi nhân viên chỉ làm việc trong một phòng ban nhưng mỗi phòng ban lại có thể có nhiều nhân viên làm việc).

✍ Mối quan hệ nhiều-nhiều

Nếu mối thực thể của E_1 có quan hệ với nhiều thực thể của E_2 và mỗi thực thể của E_2 cũng có quan hệ với nhiều thực thể của E_1 thì quan hệ giữa E_1 và E_2 được gọi là mối quan hệ nhiều-nhiều.

Ví dụ 5 Gọi A là tập thực thể gồm những người bán xe máy và B là tập thực thể gồm các loại xe máy. Khi đó mối quan hệ bán xe giữa A và B là mối quan hệ nhiều-nhiều.

+ Ràng buộc về sự tham gia:

Có 2 loại:

- Tham gia bắt buộc.
- Tham gia không bắt buộc.

Một số tài liệu còn định nghĩa khái niệm kiểu thực thể yếu (Weak Entity) đó là các thực thể mà sự tồn tại của nó phụ thuộc vào một kiểu thực thể khác – kiểu thực thể này được gọi là kiểu thực thể mạnh hay kiểu thực thể định danh. Đối với kiểu thực thể yếu, ta có khái niệm *khóa bộ phận* đó là các thuộc tính dùng để định danh (bộ phận) các thực thể yếu. Các thuộc tính này quan hệ với khóa của kiểu

thực thể mạnh (định danh) khác để tạo thành khóa của thực thể yếu. Giữa kiểu thực thể yếu và thực thể định danh có kiểu quan hệ định danh (Identifying Relationship) đó là: sự quan hệ giữa thực thể yếu và thực thể định danh nó.

Ví dụ 6

THANNHAN là thực thể yếu.

NHANVIEN là thực thể định danh của THANNHAN.

PHUTHUOC là quan hệ định danh.

2.3. Sơ đồ mối quan hệ thực thể (Entity Relationship Diagram - ERD)

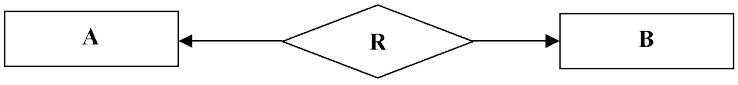
Để mô tả thông tin về các tập thực thể và mối quan hệ giữa các tập thực thể, người ta sử dụng sơ đồ mối quan hệ thực thể. Trong sơ đồ này, ta sử dụng các qui ước sau:

- Các hình chữ nhật dùng để biểu diễn các tập thực thể.

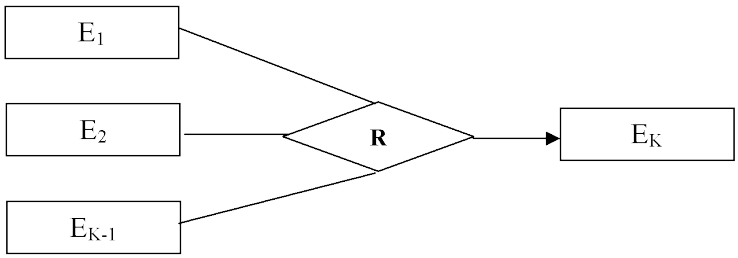
- Các hình oval biểu diễn các thuộc tính. Chúng được liên kết với các tập thực thể bằng các cung không định hướng. Các thuộc tính là thành phần của một khoá của thực thể sẽ được gạch dưới. Một trường hợp đặc biệt là đôi khi chúng ta cũng xác định một tập chỉ có một thuộc tính và gọi tập đó bằng tên thuộc tính của nó. Khi đó tập thực thể sẽ được ký bởi một hình oval chứ không phải là một hình chữ nhật.

- Các hình thoi biểu diễn các mối quan hệ. Chúng được liên kết với các tập thực thể bằng các cung định hướng hay không định hướng theo qui tắc sau đây:

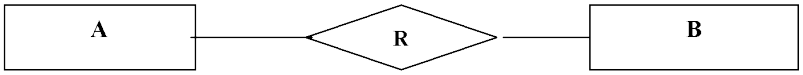
+ Nếu R là mối quan hệ một-một giữa A và B thì vẽ các cung định hướng từ hình thoi nhãn R đến các hình chữ nhật nhãn A và B.



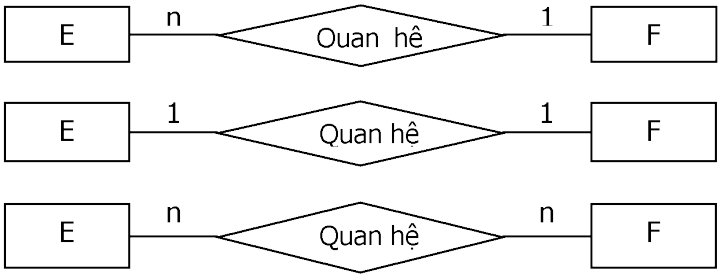
+ Nếu R là mối quan hệ nhiều-một từ tập $E_1,...,E_{k-1}$ đến E_k thì vẽ các cung không định hướng từ hình thoi nhãn R vào các hình chữ nhật nhãn $E_1,...,E_{k-1}$ và vẽ một cung định hướng từ hình thoi đến hình chữ nhật nhãn E_k.



+ Nếu R là mối quan hệ nhiều-nhiều giữa A và B thì vẽ các cung không định hướng từ hình thoi nhãn R vào các hình chữ nhật có nhãn A và B.

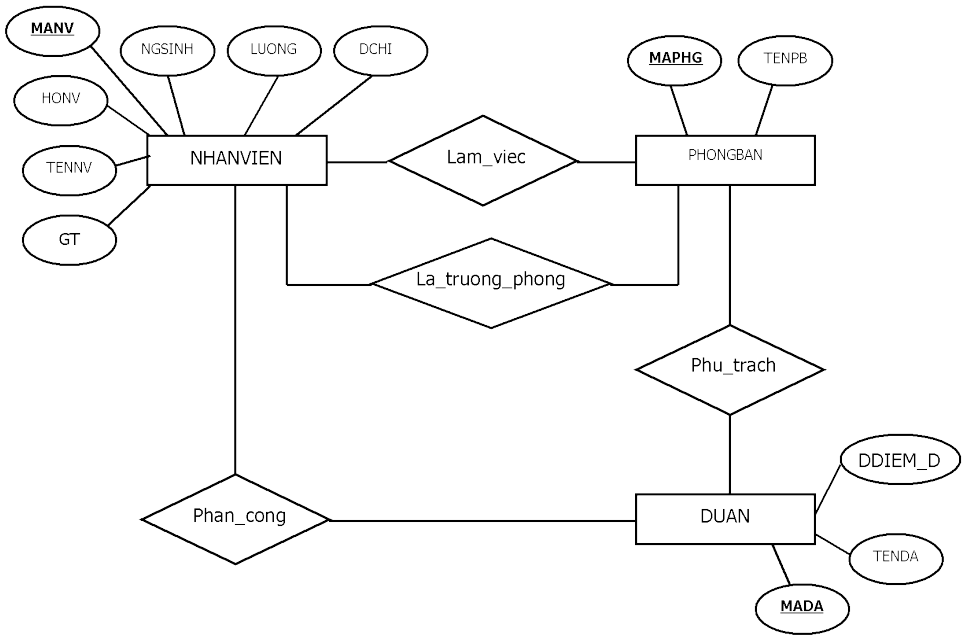


Hoặc có thể ghi chú mối liên hệ lên cạnh và chỉ cần vẽ cung vô hướng giữa hai tập thực thể.



+ Mô tả quan hệ liên kết thực thể yếu (định danh) và kiểu thực thể yếu bằng hình thoi và hình chữ nhật nét đôi.

Ví dụ 6 Sơ đồ sau đây diễn tả thông tin và mối quan hệ giữa các tập thực thể NHANVIEN, PHONGBAN và DEAN. Hai tập thực thể NHANVIEN và PHONGBAN được gắn với nhau bởi mối quan hệ **Lam_viec**, mỗi một nhân viên chỉ thuộc về một phòng nào đó trong khi một phòng có thể có nhiều nhân viên nên đây là mối quan hệ nhiều-một và một nhân viên có thể được phân công tham gia vào một hay nhiều đề án và ngược lại một đề án gồm nhiều nhân viên tham gia, một phòng ban có một nhân viên làm trưởng phòng và được phân công phụ trách một đề án.



Ví dụ 7 Sơ đồ thực thể quan hệ dưới đây mô tả thông tin và các mối quan hệ giữa các tập thực thể trong một siêu thị nhỏ ở thị trấn Yuppie Valley (Yuppie Valley Culinary Boutique: YVCB). (Xem [1]).

Một trong những khía cạnh quan trọng trong việc kinh doanh của YVCB là quan hệ với những người cung cấp hàng. Do đó trong CSDL sẽ có một tập thực thể NGUOI_CC với hai thuộc tính là TEN_CC (tên người cung cấp) và DCHI_CC (địa chỉ người cung cấp). Trong thực tế người cung cấp còn có thể có các thuộc tính khác như số điện thoại nhưng ở đây ta sẽ không đề cập.

Một mặt quan trọng đối với người cung cấp là tập các mặt hàng mà họ cung cấp không thể lưu trữ một cách thuận lợi như là một thuộc tính của NGUOI_CC. Do đó, ta sẽ dùng tập thực thể MAT_HANG với hai thuộc tính là TEN_H (tên mặt hàng) và MA_H (mã của mặt hàng). Giữa người cung cấp và mặt hàng sẽ được nối với nhau bởi mối quan hệ nhiều-nhiều CUNGCAP. Ngoài ra còn có một tập thực thể thứ ba là GIA (giá mặt hàng) cũng liên quan đến mối quan hệ này. Trong sơ đồ, ta biểu diễn tập thực thể GIA bởi một hình oval thay vì hình chữ nhật. Như vậy dựa vào mối quan hệ này, ta sẽ xác định được người cung cấp cung cấp mặt hàng nào đó cho siêu thị với một giá nào đó.

YVCB được tổ chức thành các gian hàng, mỗi một gian hàng có một người quản lý và một số nhân viên. Tập thực thể GIAN_H có các thuộc tính là TEN_GH (tên gian hàng) và MA_GH (mã gian hàng). Mỗi gian hàng sẽ bán một số mặt hàng nào đó và mỗi mặt hàng chỉ được bán trong một gian hàng nhất định. Do đó giữa tập thực thể MAT_HANG và GIAN_H sẽ tồn tại mối quan hệ nhiều-một BAN.

Các nhân viên được biểu diễn bởi tập thực thể NHANVIEN với các thuộc tính TEN_NV và LUONG. Tập thực thể này có mối quan hệ nhiều-một LAMVIEC với tập thực thể GIAN_H. Những người quản lý các gian hàng được mô tả bởi tập thực thể NGUOI_QL và tập thực thể này có mối quan hệ một-một QUANLY với tập thực thể GIAN_H và có mối quan hệ Isa với tập thực thể NHANVIEN.

Một tập thực thể quan trọng khác của YVCB KHACH_H (khách hàng) với các thuộc tính là TEN_KH (tên của khách hàng), DCHI_KH (địa chỉ khách hàng) và TAIKHOAN (tài khoản khách hàng). Mỗi khách hàng đặt hàng thông qua đơn đặt hàng (DON_DH có các thuộc tính SO_DON và NGAY_DH) và mối quan hệ DAT_H giữa DON_DH và KHACH_H là mối quan hệ nhiều-một. Tuy nhiên, nội dung thực sự của DON_DH được thể hiện thông qua mối quan hệ B_GOM giữa DON_DH, MAT_HANG và SOLUONG (mỗi đơn đặt hàng bao gồm một số mặt hàng nào đó với một số lương cụ thể nào đó), mối quan hệ này là mối quan hệ nhiều-một từ DON_DH và MAT_HANG vào SOLUONG.

Mô hình CSDL sau đây tổ chức hệ thông tin cần thiết để quản lý sự buôn bán của siêu thị YVCB (Yuppi Valley Culinary Bontique). Trong sơ đồ quan hệ thực thể của siêu thị này gồm có các tập thực thể sau:

NV là tập thực thể các nhân viên trong siêu thị. Để đơn giản cho việc thiết kế chúng ta giả thiết rằng mỗi thực thể của tập thực thể này có các thuộc tính: TNV (tên nhân viên) , LG (lương nhân viên), trong đó, TNV có thể lựa chọn làm khoá.

NQL (người quản lý) là tập thực thể các trưởng của các gian hàng trong siêu thị. Tập thực thể này là một bộ phận của tập thực thể NV, vì người quản lý cũng là nhân viên của siêu thị.

GH (gian hàng) là tập thực thể các gian hàng trong siêu thị. Tập thực thể này có các thuộc tính SGH (số hiệu của gian hàng), TGH (tên gian hàng). SGH có thể được lựa cọn làm khoá.

MH (mặt hàng) là tập thực thể các mặt hàng được bày bán trong siêu thị. Tập thực thể này có các thuộc tính TMH (tên mặt hàng), SMH (số hiệu của mặt hàng). SMH có thể lựa chọn làm khoá của MH.

NCC (người cung cấp) là tập thực thể các nhà cung cấp hàng cho siêu thị. Tập thực thể này có các thuộc tính TCC (tên người cung cấp hàng), DCC (địa chỉ của người cung cấp hàng). TCC có thể được lựa chọn để làm khoá cho NCC.

DDH (đơn đặt hàng) là tập thực thể các đơn được đặt để mua hàng hoá trong siêu thị. Tập thực thể này có các thuộc tính SD (số hiệu của đơn), NDH (ngày đặt hàng). Trong đó SD có thể chọn làm khoá của DDH.

KH (khách hàng) là tập thực thể các khách hàng mua hàng ở siêu thị. Tập thực thể này có các thuộc tính TKH (tên khách hàng), DKH (địa chỉ khách hàng), TC (cân đối thu chi trong tài khoản của khách hàng). TKH có thể được chọn làm khoá của KH.

GIA (giá) là tập thực thể giá các mặt hàng. Tập thực thể này chỉ gồm một thuộc tính, nó cũng là thuộc tính khoá.

SL (số lượng) là tập thực thể số lượng hàng. Tập thực thể này chỉ gồm một thuộc tính. Thuộc tính này là khoá của tập thực thể SL.

Giữa các tập thực thể này chúng ta có thể thiết lập các mối quan hệ sau:

LV (làm việc) là mối quan hệ giữa tập thực thể NV và GH. Ta nói rằng thực thể a của tập thực thể NV có mối quan hệ LV với thực thể x của tập thực thể GH nếu nhân viên a làm việc trong gian hàng x. Đây là mối quan hệ nhiều - một (có thể có nhiều nhân viên làm việc trong cùng một gian hàng nhưng không có nhân viên nào làm việc tại hai gian hàng trong cùng một thời gian).

QL (quản lý) là mối quan hệ giữa tập thực thể NQLvà GH. Chúng ta nói rằng thực thể a của tập thực thể NQL có mối quan hệ QL với thực thể x của tập thực thể GH nếu nhân viên a là thủ trưởng (quản lý) gian hàng x. Đây là mối quan hệ một - một (một gian hàng chỉ có một trưởng gian hàng và một trưởng gian hàng chỉ có thể quản lý một gian hàng duy nhất).

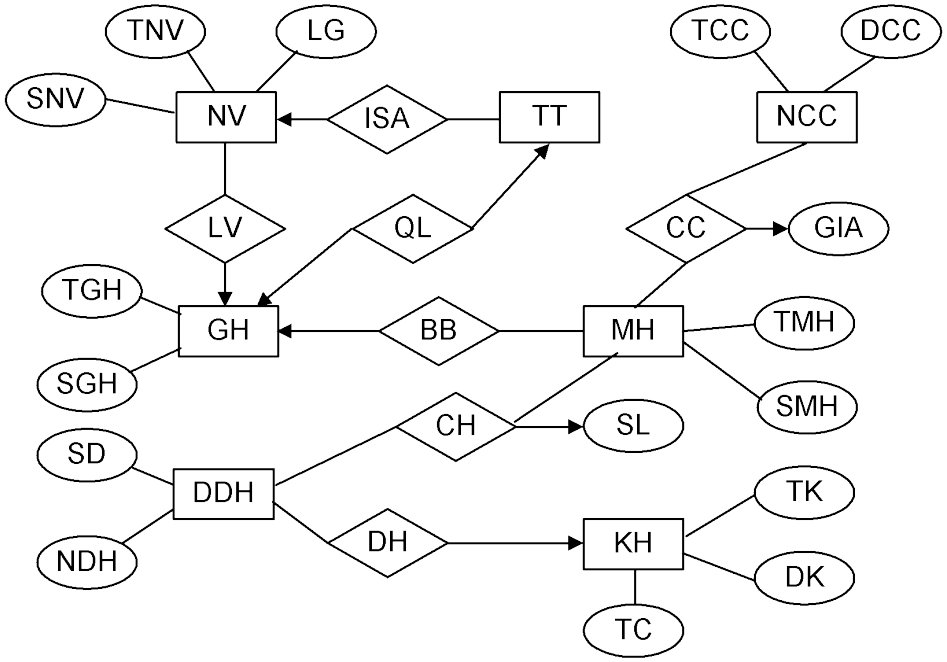
BB (bày bán) là mối quan hệ giữa tập thực thể MH và GH. Chúng ta nói rằng thực thể a của tập thực thể MH có quan hệ BB với thực thể x của tập thực thể GH nếu mặt hàng a được bày bán ở gian hàng x. Đây là mối quan hệ nhiều - một (mỗi mặt hàng chỉ được bày bán trong một gian hàng nhất định và mỗi gian hàng có thể bày bán nhiều mặt hàng khác nhau).

CC (cung cấp) là mối quan hệ giữa các tập thực thể NCC, MH và GIA. Chúng ta nói rằng thực thể a của tập thực thể NCC và thực thể b của tập thực thể MH có quan hệ CC với thực thể x của tập thực thể GIA nếu người cung cấp a cung cấp mặt hàng b với giá x. Đây là mối quan hệ nhiều - một từ NCC và MH vaò GIA (một người cung cấp cung cấp một mặt hàng nào đó chỉ với một giá duy nhất).

DH (đặt hàng) là mối quan hệ giữa tập thực thể DDH và KH. Chúng ta nói rằng thực thể a của tập thực thể DDH có quan hệ DH với thực thể x của tập thực thể KH nếu đơn hàng a được đặt bởi khách hàng x. Đây là mối quan hệ nhiều - một (một khách hàng có thể đặt nhiều đơn đặt hàng khác nhau và mỗi đơn chỉ được đặt bởi một khách hàng).

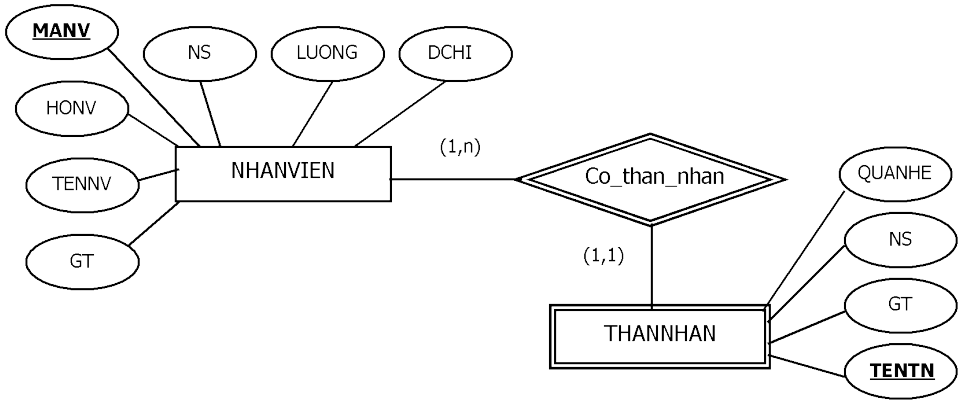
CH (chứa) là mối quan hệ giữa tập thực thể DDH, MH và SL. Chúng ta nói rằng thực thể a của tập thực thể DDH và thực thể b của tập thực thể MH có mối quan hệ CH với thực thể n của tập thực thể SL nếu đơn đặt hàng a chứa mặt hàng b với số lượng n. Đây là mối quan hệ nhiều-một từ DDH và MH vào SL (mỗi đơn đặt hàng có hiện diện một mặt hàng nào đó với một số lượng duy nhất).

Ngoài ra, dễ dàng nhận thấy rằng giữa tập thực thể NV và tập thực thể NQL có một mối quan hệ hiển nhiên là quan hệ ISA. Quan hệ này cho thấy rằng mỗi người quản lý cũng là một nhân viên của siêu thị. Đây là mối quan hệ nhiều - một từ NQL vào tập thực thể NV.



Hình 2.2. Sơ đồ mối quan hệ thực thể của YVCB

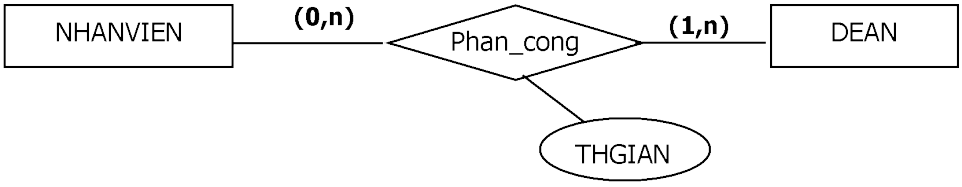
Dưới đây là sơ đồ quan hệ thực thể thể hiện mối quan hệ giữa nhân viên và thân nhân của họ qua quan hệ có thân nhân.



2.3.1. Thuộc tính trên mối quan hệ

Thuộc tính trên mối quan hệ nhằm 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.

Ví dụ:



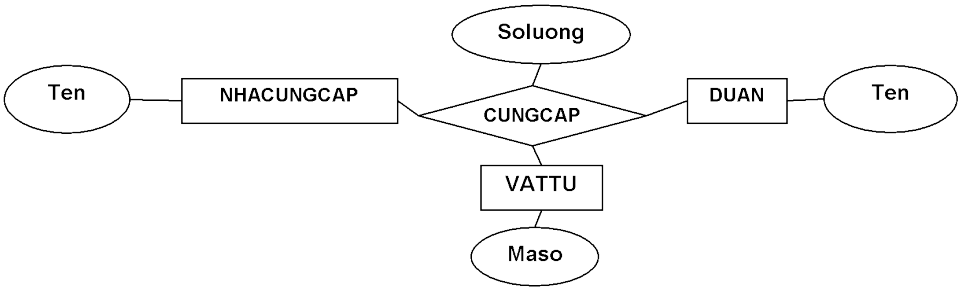
Thuộc tính THGIAN là một thuộc tính trên mối quan hệ Phan_cong.

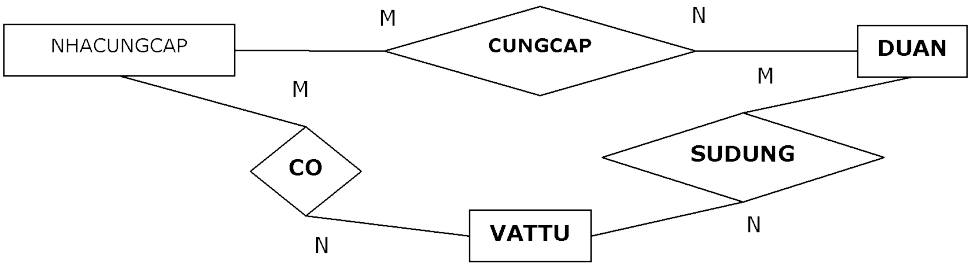
2.3.2. Ràng buộc tham gia

Xét lại ví dụ 6 ở trên ta thấy phòng ban nào cũng cần có trưởng phòng và một nhân viên phải làm việc trong một phòng ban nào đó. Ràng buộc quan hệ này được gọi là ràng buộc *tham gia toàn bộ*. Trong sơ đồ ER ta sử dụng vạch đôi để chỉ mối quan hệ này. Ngược lại không phải nhân viên nào cũng là trưởng phòng, ràng buộc quan hệ này được gọi là ràng buộc *tham gia bộ phận*.

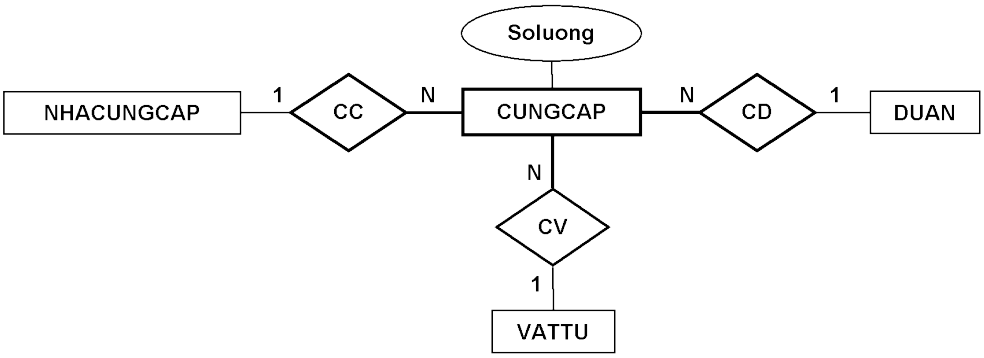
2.3.3. Mối quan hệ tam phân

Ngoài mối quan hệ nhị phân ta con gặp mối quan hệ tam phân, đó là mối quan hệ 3 ngôi giữa 3 tập thực thể. Trong hình vẽ dưới đây thì quan hệ cung cấp (cungcap) là quan hệ tam phân, cần chuyển thành các quan hệ nhị phân để tiện trong quá trình chuyển đổi sang mô hình logic.

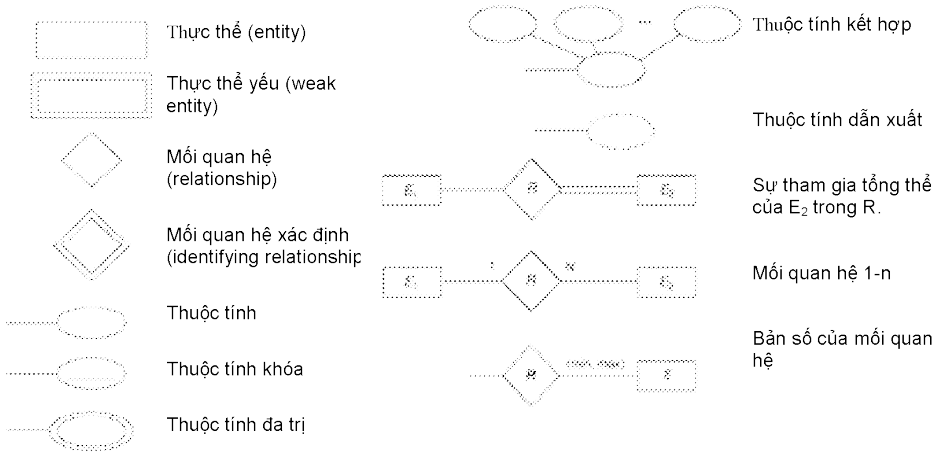




Trong sơ đồ trên quan hệ tam phân ***cung cấp*** được chuyển đổi thành quan hệ nhị phân nhờ thêm vào hai quan hệ ***có*** và ***sử dụng***. Hoặc có thể chuyển đổi bằng cách thêm vào tập thực thể mới CUNG CAP.



Có thể hệ thống lại các ký hiệu của sơ đồ quan hệ thực thể qua hình dưới đây:



Hình 2.3. Các ký hiệu của sơ đồ quan hệ thực thể

2.4. Mô hình quan hệ thực thể mở rộng (Enhanced Entity Relationship Model – EER)

Các khái niệm cơ bản của mô hình ER không đủ để biểu diễn một số các ứng dụng phức tạp, vì vậy cần phải thêm vào sơ đồ này một số khái niệm để tăng khả năng mô tả đối tượng rõ ràng và chính xác hơn. Mô hình quan hệ thực thể mở rộng EER là sự mở rộng của sơ đồ ER bằng cách thêm vào một số các khái niệm trừu tượng (abstraction) và thể hiện các ràng buộc rõ ràng hơn.

Một số khái niệm được bổ sung, trong đó có một số khái niệm tương tự như trong lập trình hướng đối ượng (OOP) đó là: lớp, kiểu quan hệ lớp cha/ lớp con, tính thừa kế, chuyên biệt, tổng quát hóa, phân cấp.

Lớp cha: là loại thực thể bao gồm một số các thực thể riêng biệt được thể hiện trong mô hình dữ liệu.

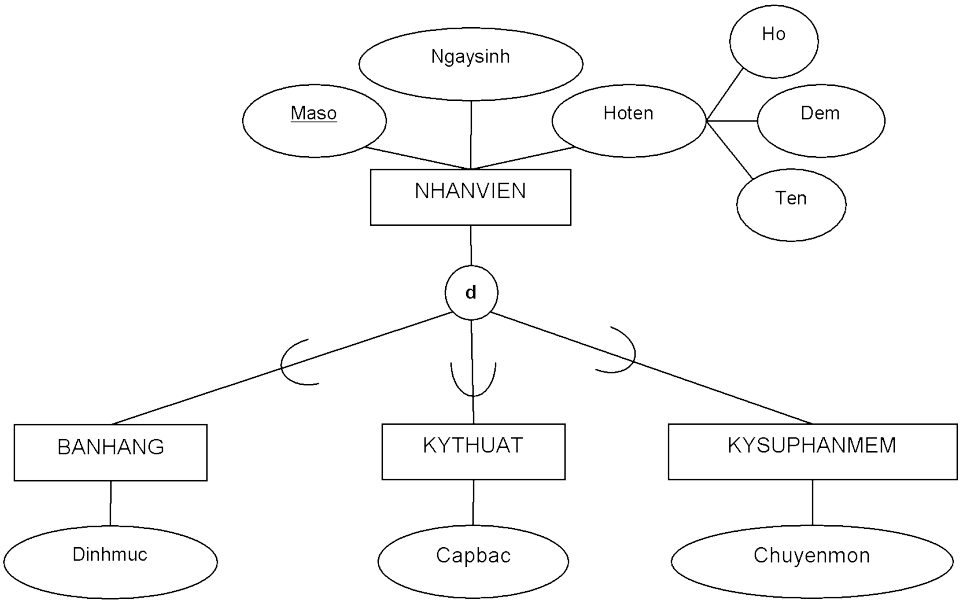
Lớp con: là các thực thể có vai trò riêng biệt nhưng là một thành viên của lớp cha.

Ví dụ: Lớp con: Quản lý, kế toán, thư ký…. và Lớp cha: Nhân viên.

Mối quan hệ giữa lớp con và lớp cha là quan hệ isa đã nói ở trên. Một thực thể của lớp con cũng là một thực thể của lớp cha. Một thực thể của lớp cha có thể là thực thể của nhiều lớp con. Một thực thể của lớp con kế thừa mọi thuộc tính và

mọi mối quan hệ của thực thể lớp cha. Một thực thể của lớp con có thể có các thuộc tính và các mối quan hệ của riêng nó. kiểu quan hệ lớpcha/lớpcon được biểu diễn bằng một đường nối có thêm một ký hiệu tập con "⊂" ở giữa đường nối. Các lớp con trong một chuyên biệt được nối với một vòng tròn và vòng tròn được nối với lớp cha.

Nhờ có quan hệ này mà chúng ta tránh mô tả các định nghĩa trùng lặp nhau, đồng thời thêm thông tin về ngữ nghĩa vào trong thiết kế.



Hình 2.4. Quan hệ thừa kế cha/con

2.4.1. Chuyên biệt hóa (CBH) và tổng quát hóa (TQH).

Chuyên biệt hóa là quá trình xây dựng các lớp con của một kiểu thực thể. Có thể có nhiều sự CBH của cùng một kiểu thực thể.

Ví dụ: Từ thực thể nhân viên nhờ vào quá trình chuyên biệt hóa ta có thể rút ra các tập thực thể sau: nhân viên bán hàng, nhân viên kỹ thuật…

Các bước thực hiện CBH:

- Định nghĩa các lớp con của một kiểu thực thể.
- Thiết lập các thuộc tính riêng của lớp con.

- Thiết lập các kiểu quan hệ riêng của lớp con.

Tổng quát hóa là quá trình xây dựng một kiểu thực thể tổng quát từ các kiểu thực thể đã có. Có thể nói đây là quá trình ngược với chuyên biệt hóa. Các thao tác chuyên biệt hóa, tổng quát hóa là các quá trình cơ bản của việc trừu tượng hóa dữ liệu nhằm nhận diện các tập thực thể có trong bài toán.

Cần lưu ý một số ràng buộc sau khi tiến hành CBH và TQH, đó là:

- Tính tách rời của các lớp con – các lớp con là rời nhau.

- Tính đầy đủ của lớp cha – lớp cha bao chứa các lớp con.

Lớp con kế thừa các thuộc tính và các kiểu quan hệ của các lớp tổ tiên.

2.4.2. Các loại ràng buộc trên sự chuyên biệt và tổng quát hóa

Ràng buộc rời rạc (disjointness constraint) mô tả quan hệ giữa lớp cha và các lớp con phải độc lập hoàn toàn (một thực thể là thành viên của chỉ một lớp con được chuyên biệt hóa). Trong sơ đồ EER, ràng buộc rời rạc được ký hiệu bởi chữ d (disjoint) nằm trong vòng tròn.

Ràng buộc chồng chéo (Overlaping constraint) cho biết quan hệ giữa lớp cha và các thực thể ở lớp con là không tách rời được (một thực thể có thể là thành viên của nhiều lớp con theo sự chuyên biệt hóa). Trong sơ đồ EER, ràng buộc này được mô tả bởi chữ o (overlap) bên trong vòng tròn.

Ràng buộc đầy đủ (completeness constraint) bao gồm:

Ràng buộc toàn bộ (total) cho biết tất cả các thực thể trong lớp cha phải là thành viên của ít nhất một lớp con nào đó trong chuyên biệt. Ràng buộc này được thể hiện bằng một đường nét đôi nối giữa lớp cha và vòng tròn chuyên biệt.

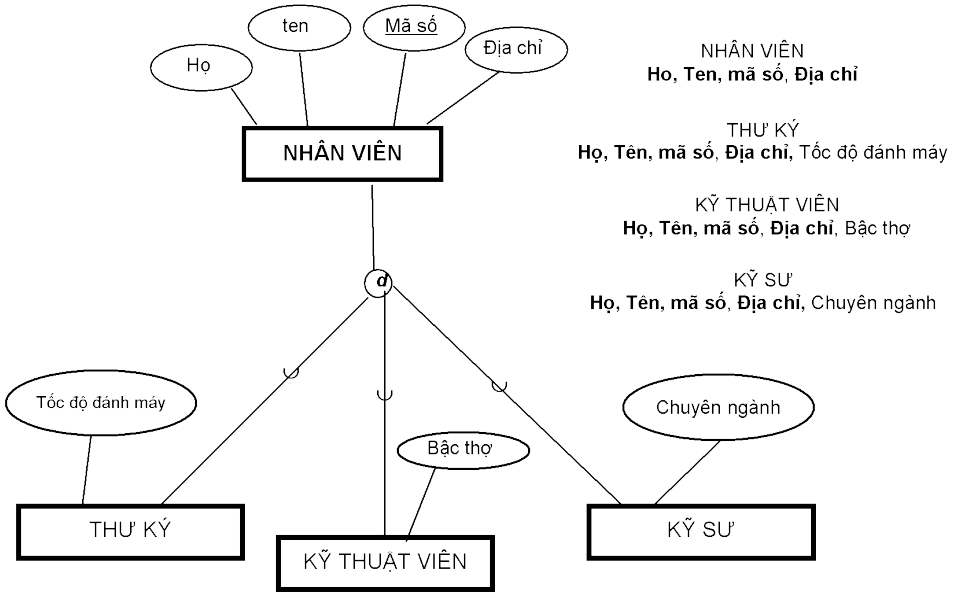
Ràng buộc từng phần (partial) cho phép một thực thể ở lớp cha không thuộc bất kỳ một lớp con nào trong chuyên biệt. Ràng buộc này được thể hiện bằng đường nét đơn.

2.4.3. Chuyên biệt (tổng quát) phân cấp và lưới

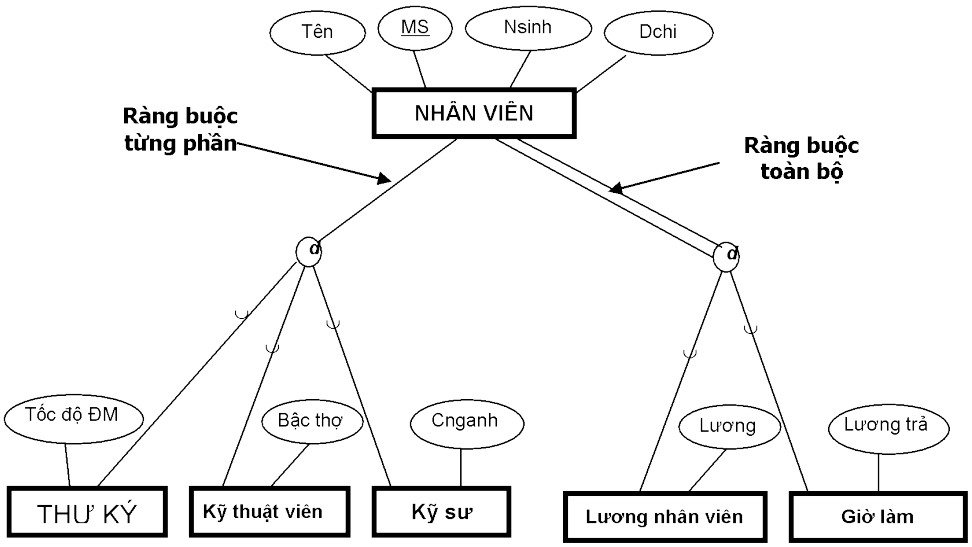
Một lớp con có thể có lớp con của chính nó bao gồm 2 loại:

Phân cấp (hierarchy) là ràng buộc trong đó tất cả các lớp con chỉ tham gia vào 1 quan hệ lớp cha/con (thừa kế đơn ánh)

Lưới (Lattice) là ràng buộc trong đó lớp con có thể tham gia vào nhiều hơn 1 quan hệ cha/con (thừa kế bội). Trong loại chuyên biệt này lớp con không chỉ kế thừa thuộc tính của lớp cha mà còn kế thừa thuộc tính của lớp cha của lớp cha nó.

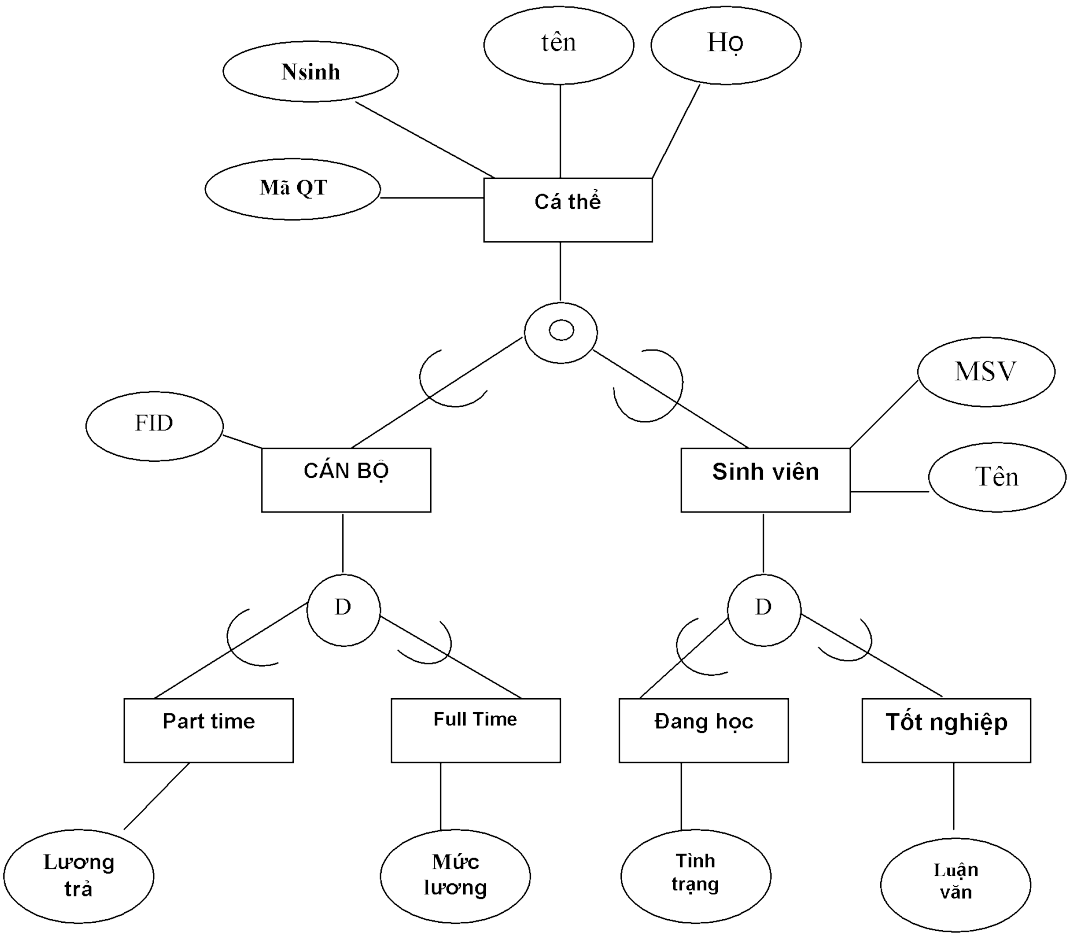


Hình 2.5. Chuyên biệt hóa và tổng quát hóa

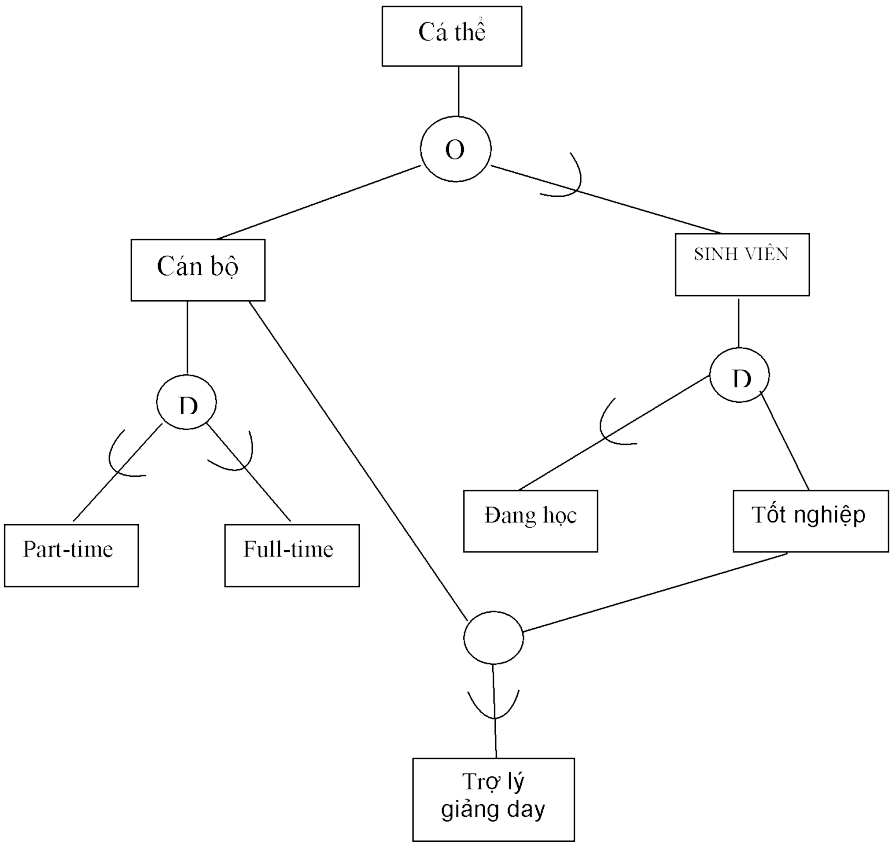


Hình 2.6. Quan hệ cha/con với ràng buộc rời rạc, ràng buộc toàn bộ và ràng buộc bộ phận

Trong hình vẽ dưới đây thì quan hệ giữa CÁ THỂ và CÁN BỘ cùng SINH VIÊN là quan hệ chồng chéo.



Hình 2.7. Chuyên biệt phân cấp



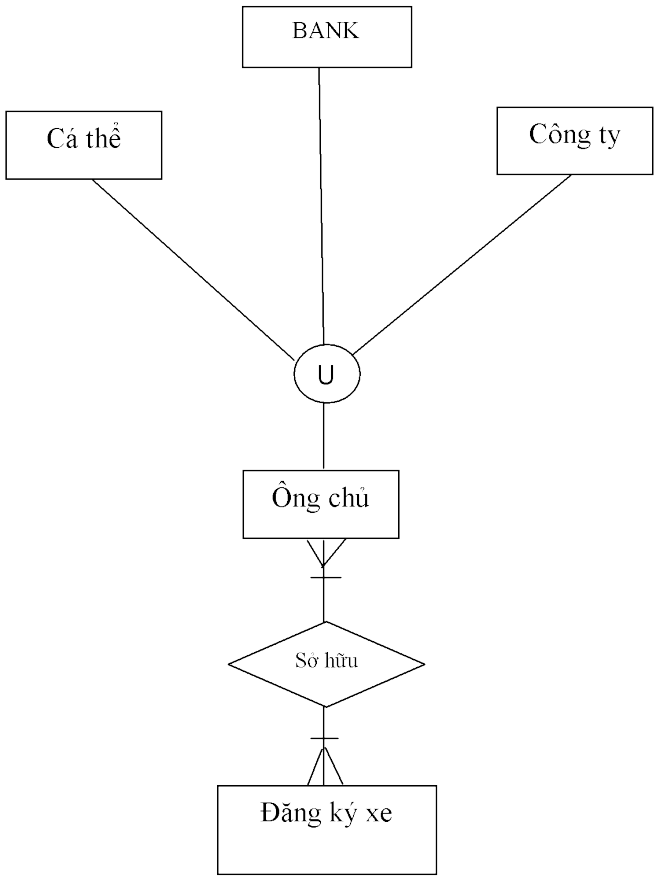
Hình 2.8. Chuyên biệt lưới

2.4.4. Kiểu hợp - phạm trù

Trong chuyên biệt lưới, một lớp con có thể thừa kế từ nhiều lớp cha, thừa hưởng tất cả các thuộc tính của lớp cha (đa thừa kế). Vấn đề là làm thế nào để mô tả quan hệ lớp con/cha của 1 lớp con với một tập các lớp cha?

Ta gọi lớp con đa thừa kế là lớp con chia sẻ (shared subclass) viết tắt là ssc, mỗi ssc có thể tham gia vào nhiều mối quan hệ cha/con với mỗi quan hệ có một lớp cha của ssc. Trong nhiều trường hợp cần mô hình hóa các lớp cha của một ssc lại thành một lớp chung ta gọi là một phạm trù (category) hay một kiểu hợp (union type).

Ví dụ: Trong cơ sở dữ liệu lưu trữ việc đăng ký xe ô tô thì chủ xe có thể là tư nhân, một ngân hàng hay một công ty. Phạm trù Ông chủ (Owner) là một hợp nhất của 3 lớp cha Cá thể (Person), Bank và Công ty (Company) của lớp con đăng ký xe (Registered Vehicle).



Trong hình vẽ trên ta dùng ký hiệu U trong vòng tròn để chỉ sự hợp nhất 3 lớp cha thành phạm trù Ông chủ (Owner).

Một phạm trù có thể là phạm trù tổng hợp (Total) hay phạm trù bộ phận (Partial). Một phạm trù tổng hợp là hợp nhất của tất cả các lớp cha trong khi phạm trù bộ phận thì có thể là hợp nhất một số lớp mà thôi. Trong sơ đồ EER tương tự ở trên ta dùng vạch đôi để chỉ phạm trù tổng hợp

Có thể tóm lược một số nguyên tắc thiết kế sơ đồ ERD:

+ *Phân tích yêu cầu*:

Phân tích yêu cầu là bước quan trọng của vòng đời CSDL. Người thiết kế phải khảo sát, phỏng vấn..., nhằm xác định CSDL đáp ứng được gì và cần cái gì. Mục tiêu cơ bản của bước này là:

• Xác định rõ yêu cầu của từng bộ phận, khách quan trung thực. Phân loại thực thể, thuộc tính.

• Mô tả thông tin về các đối tượng và xác định mối quan hệ giữa các đối tượng cần thiết kế.

• Xác định các loại giao dịch trên CSDL. Tương tác giữa các giao dich.

• Xác định các ràng buộc toàn vẹn, tính bảo mật để áp đặt lên CSDL.

• Xác định phần cứng hệ thống, điều kiện cài đặt.

• Tài liệu khảo sát.

+ *Các bước thiết kế*:

- (1) Xác định tập thực thể.
- (2) Xác định mối quan hệ.
- (3) Xác định thuộc tính và gắn thuộc tính cho tập thực thể và mối quan hệ.
- (4) Quyết định miền giá trị cho thuộc tính.
- (5) Quyết định thuộc tính khóa.
- (6) Quyết định (min, max) cho mối quan hệ.

+ *Các nguyên tắc thiết kế*:

- Khảo sát kỹ qui trình nghiệp vụ của người dùng và ứng dụng.
- Giao tiếp với nhiều nhóm người dùng.
- Sử dụng các thuật ngữ của thế giới thực.
- Chính xác, đơn giản, dễ hiểu và tránh trùng lắp.

+ *Phương pháp tiếp cận*:

- Nhận dạng các thành phần cơ bản của lược đồ
  - Các kiểu thực thể, kiểu quan hệ và các thuộc tính.

- Các thuộc tính khóa, tỉ lệ lực lượng và ràng buộc tham gia.
- Các kiểu thực thể yếu.
- Các chuyên biệt hóa và tổng quát hóa.

- 2 phương pháp tiếp cận
  - Thiết kế trực tiếp lược đồ tổng thể.
  - Thiết kế lược đồ tổng thể bằng cách tích hợp các lược đồ con.

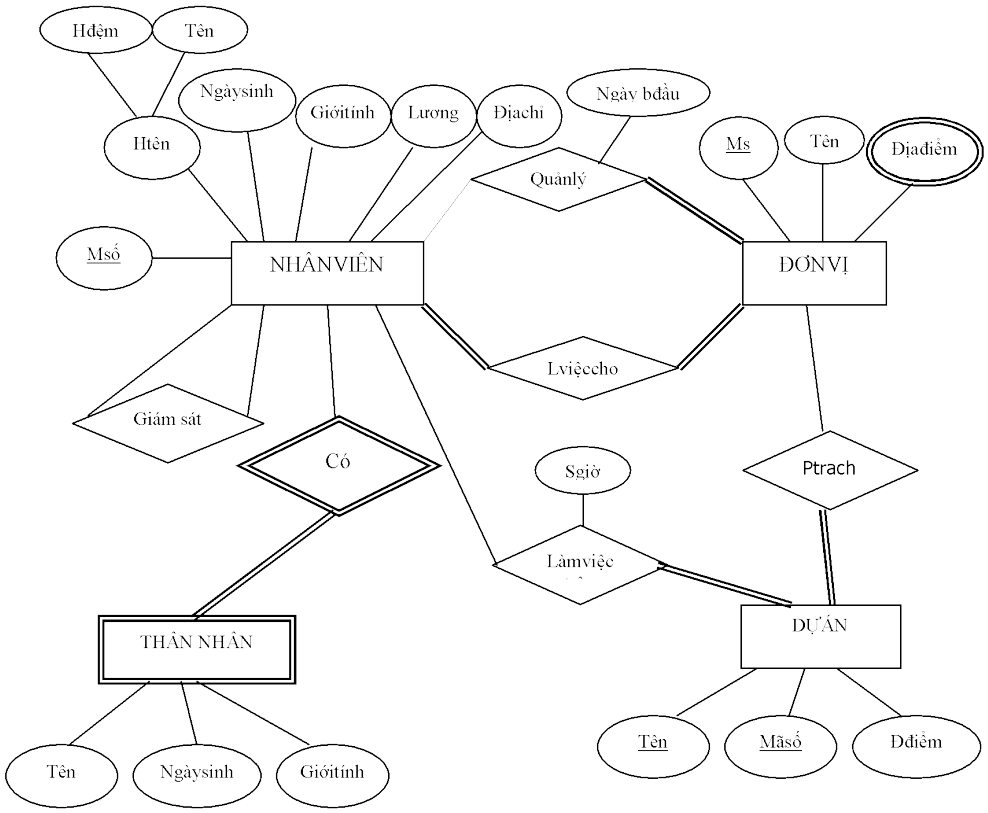
+ *Tinh chỉnh lược đồ*:

Để tinh chỉnh lược đồ ERD sau khi thiết kế, cần lưu ý một số điểm sau.

- Kiểu thực thể hay thuộc tính. Ví dụ Tên của Khoa có thể là thuộc tính của các thực thể sau: Sinh viên, Giảng viên, Môn học… có thể được tách ra thành một thực thể KHOA.

- Kiểu quan hệ hay thuộc tính.
- Sắp xếp các thuộc tính.
- CBH hay TQH.
- Quan hệ nhị phân hay quan hệ n-phân. Nên tìm cách để chuyển các quan hệ n- phân về thành các quan hệ nhị phân.

Dưới đây là sơ đồ ER ở mức đơn giản cho bài toán quản lý công ty đã nói ở đầu chương.



Hình 2.9. Sơ đồ ER ở mức đơn giản cho bài toán quản lý công ty

2.5. Mô hình dữ liệu mạng (network data model)

Mô hình dữ liệu mạng là mô hình thực thể quan hệ trong đó các mối quan hệ là nhị phân nhiều-một. Điều này cho phép chúng ta sử dụng một một mô hình đồ thị có hướng đơn giản cho dữ liệu. Tại vị trí của các tập thực thể, mô hình đưa ra kiểu mẫu tin logic. Một kiểu mẫu tin logic là một tên đặt cho một tập các bản ghi, được gọi là bản ghi logic (logical record). Các bản ghi logic được cấu tạo bởi các trường chứa các giá trị cơ bản như số nguyên, chuỗi ký tự. Tập các tên của các trường và các kiểu của nó thiết lập nên kiểu dạng bản ghi logic.

2.5.1. Nhận dạng bản ghi

Giữa các thuật ngữ sử dụng cho mô hình mạng và các thuật ngữ sử dụng cho mô hình quan hệ có sự tương đồng với nhau, được mô tả sau đây:

Khuôn dạng bản ghi logic: Lược đồ quan hệ

Bản ghi logic : Bộ

Kiểu bản ghi logic : Tên quan hệ

Tuy nhiên có một sự khác biệt quan trọng giữa các bộ của các quan hệ và các bản ghi của kiểu bản ghi. Trong mô hình quan hệ hướng giá trị, các bộ chẳng qua là các giá trị của các thành phần. Hai bộ có cùng giá trị cho cùng các thuộc tính là những bộ giống nhau. Ngược lại, mô hình mạng là mô hình hướng đối tượng, ít nhất theo nghĩa là nó hỗ trợ nhận dạng đối tượng. Các bản ghi của mô hình mạng có thể xem như là có một khoá vô hình (invisitable key) mà thực chất là địa chỉ của bản ghi, tức là đặc tính nhận dạng đối tượng của nó. Dấu hiệu nhận dạng duy nhất này làm cho các bản ghi khác nhau, cho dù chúng có cùng các giá trị trong các trường tương ứng.

2.5.2. Các đường nối (links)

Thay vì nói đến các mối quan hệ hai ngôi nhiều một, chúng ta sẽ đề cập đến các đường nối trong mô hình mạng. Mạng ở đây là một đồ thị có hướng, nó thực sự là một dạng đơn giản hoá của sơ đồ thực thể quan hệ nhằm biểu diễn các bản ghi và các đường nối của chúng. Các nút tương ứng với các kiểu bản ghi. Nếu có một đường nối giữa hai kiểu bản ghi T1 và T2, và đường nối là nhiều-một từ T1 đến T2, thì ta vẽ một cung từ nút T1 đến nút T2 và ta nói đường nối là từ T1 đến

T2. Các nút và các cung được gán nhãn bởi tên của các kiểu bản ghi và đường nối của chúng.

2.5.3. Biểu diễn các tập thực thể trong mô hình mạng

Các tập thực thể được biểu diễn trực tiếp bởi các kiểu bản ghi logic, các thuộc tính của một tập thực thể trở thành các trường của khuôn dạng bản ghi logic. Trường hợp đặc biệt duy nhất là khi một tập thực thể E tạo khóa của nó bằng các trường của tập thực thể F mà E có quan hệ thông qua mối quan hệ R. Ta không cần đặt những trường này của F vào trong khuôn dạng bản ghi của E bởi vì các bản ghi của E không cần được phân biệt bởi các giá trị trường của chúng. Hơn nữa, chúng sẽ được phân biệt bởi các con trỏ vật lý đặt trong các bản ghi của E để biểu diễn mối quan hệ R, và những con trỏ này sẽ chỉ từ một bản e của kiểu E đến bản ghi tương ứng của kiểu F giữ giá trị khoá cho e.

Mặt khác, khi mối quan hệ là isa và tập con không có các trường mà tập cha không có. Lấy ví dụ mối quan hệ giữa NGUOI_QL và NHANVIEN trong YVCB, ta sẽ loại bỏ kiểu bản ghi cho tập con (tức là NGUOI_QL) và để mối quan hệ giữa NGUOI_QL và các tập thực thể khác (ngoại trừ NHANVIEN) được biểu diễn trong mô hình mạng bằng đường nối liên quan đến NHANVIEN. Bản thân mối quan hệ Isa có thể được biểu diễn bởi một trường một bit cho biết một nhân viên có phải là một người quản lý không. Một lựa chọn khác là biểu diễn ngầm Isa, chỉ các bản ghi NHANVIEN cho người quản lý sẽ tham gia trong mối quan hệ liên quan đến tập các người quản lý (chẳng hạn QUANLY).

2.5.4. Biểu diễn các mối quan hệ

Mặc dù chỉ có những mối quan hệ nhiều-một hay một-một là có khả năng thể hiện trực tiếp bằng các đường nối, ta có thể sử dụng phương pháp sau đây để thể hiện mối quan hệ bất kỳ. Giả sử chúng ta có mối quan hệ R giữa các tập thực thể E_1, E_2,..., E_k. Khi đó, với các thực thể đại diện cho mối quan hệ R, ta sẽ tạo mới một kiểu bản ghi logic T biểu diễn cho các k-bộ (e_1, e_2,...,e_k) của chúng. Khuôn dạng của kiểu bản ghi này có thể rỗng. Tuy nhiên, đôi khi cần phải bổ sung các trường mang thông tin (information-carrying fields) khuôn dạng của kiểu bản ghi mới T. Trong mọi trường hợp, chúng ta tạo các đường nối L_1, L_2,...,L_k. Đường nối L_i sẽ đi từ kiểu bản ghi T đến kiểu bản ghi của tập thực thể E_i (ta còn

gọi là E_i). Mục đích của việc này là bản ghi của kiểu T cho $(e_1,e_2,...,e_k)$ được nối đến bản ghi của kiểu E_i cho e_i, và vì thế mỗi đường nối là nhiều-một.

Một trường hợp đặc biệt, nếu một mối quan hệ là nhiều-một từ $E_1,E_2,...,E_{k-1}$ vào E_k và tập thực thể E_k không xuất hiện trong mối quan hệ nào khác thì ta sử dụng kiểu bản ghi T thay cho E_k và các thuộc tính của E_k sẽ được lưu trữ trong T. Chẳng hạn trong YVCB, mối quan hệ CUNGCAP là nhiều-một từ NGUOI_CC và MAT_HANG vào GIA, và GIA không có mối quan hệ với bất cứ tập thực thể nào khác nữa. Do đó có thể tạo kiểu T chứa GIA như là một trường và nó có các đường nối đến NGUOI_CC và MAT_HANG.

Ví dụ: Sau đây ta sẽ biến đổi sơ đồ thực thể quan hệ của YVCB sang mô hình mạng. Ta sẽ sử dụng các kiểu bản ghi logic để biểu diễn các tập thực thể trong sơ đồ (ở đây tập thực thể NGUOI_QL sẽ được thể hiện bởi kiểu bản ghi logic của tập thực thể cha, tức là NHANVIEN). Do đó ta có các khuôn dạng bản ghi logic sau:

NHANVIEN(TEN_NV, LUONG)

NGUOI_CC(TEN_CC, DCHI_CC)

MAT_HANG(TEN_H, MA_H)

GIAN_H(TEN_GH, MA_GH)

DON_DH(SODON, NGAY_DH)

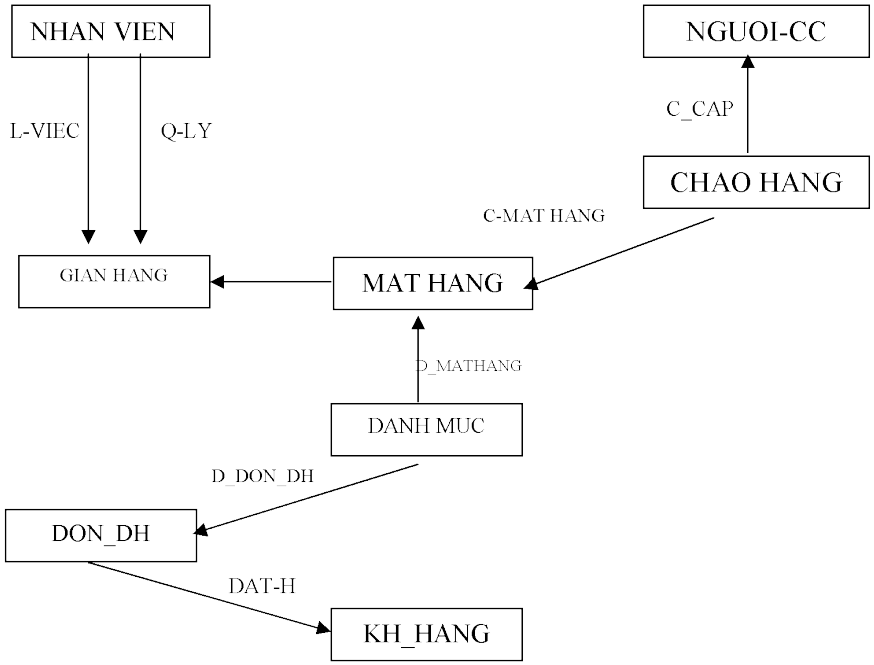
KHACH_H(TEN_KH, DCHI_KH, TAIKHOAN)

Trong YVCB, các mối quan hệ CUNGCAP và BGOM không phải là mối quan hệ hai ngôi, nhiều-một nên ta cần thêm hai kiểu bản ghi sau:

CHAOHANG(GIA)

DANHMUC(SOLUONG)

để thể hiện cho các mối quan hệ này. Mô hình mạng của YVCB khi đó sẽ được biểu diễn như sau:



Hình 2.10. Mô hình mạng của YVCB

2.6. Mô hình dữ liệu phân cấp

Mô hình phân cấp đơn giản chỉ là mô hình mạng có cấu trúc như là một rừng. Trong mô hình phân cấp, các đường nối chỉ theo hướng từ nút cha tới nút con. Những thuật ngữ sử dụng trong mô hình mạng như kiểu bản ghi logic cũng được sử dụng trong mô hình phân cấp. Bất cứ một sơ đồ thực thể quan hệ nào có thể được diễn tả trong mô hình quan hệ và mô hình mạng, nó cũng có thể được diễn tả trong một mô hình phân cấp.

2.6.1. Chuyển đổi mô hình mạng thành mô hình phân cấp

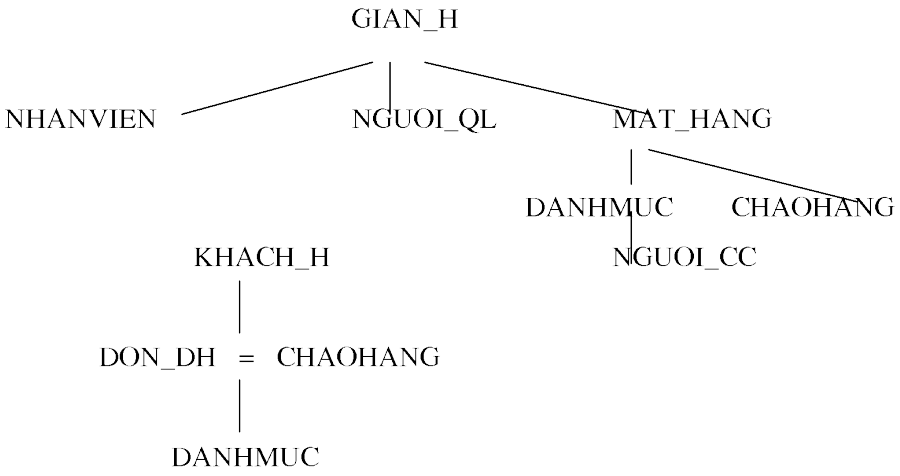
Trong phần này, chúng ta sẽ tìm hiểu về cách thiết kế mô hình phân cấp bằng cách tách một mô hình mạng đơn giản thành một hay nhiều cây. Trong mô hình phân cấp mọi đường nối đi từ nút con đến nút cha, do đó ta phải bắt đầu tại một nút càng nhiều đường nối vào nó càng tốt và lấy nó làm gốc của một cây. Ta sẽ gắn vào cây đó tất cả các nút có thể gắn được (chú ý các đường nối phải chỉ

vào nút cha). Khi không chọn được nút nào nữa, ta sẽ lặp lại với nút chưa được gắn làm gốc và gắn vào nó càng nhiều nút nếu có thể. Như vậy trong một rừng, mỗi một nút có thể xuất hiện một hay nhiều lần.

Thuật toán chuyển đổi từ mô hình mạng thành mô hình phân cấp được mô tả như sau:

```
Procedure    BUILD(n);
  Chọn(n):=true;
  For   ”mỗi đường nối từ nút m nào đó đến n”   do
     Begin
        ”Tạo m là một nút con của n”;
        If    not chọn(m)  then  BUILD(m);
     End;
End;
/* Chương trình chính */
For  ’’tất cả mọi nút k”  do chọn(k):=false;
While  ’’còn tồn tại nút chưa được chọn”  do
    Begin
          ”Lấy ra một nút n chưa được chọn”;
         BUILD(n);
   End;
```

Ví dụ: Dưới đây là mô hình phân cấp được xây dựng từ mô hình mạng của YVCB ở trên.



2.6.2. Bản ghi CSDL (database record)

Một mô hình phân cấp ở mức khái niệm bao gồm tập hợp các cây, trong đó các nút của nó là các bản ghi; mỗi một cây được gọi là một bản ghi CSDL. Một bản ghi CSDL tương ứng với một cây của lược đồ CSDL, và bản ghi gốc (root record) của một bản ghi CSDL tương ứng với một thực thể của kiểu bản ghi gốc. Nếu T là một nút của lược đồ và S là một trong những con của nó thì mỗi một bản ghi của kiểu T trong bản ghi CSDL có không hoặc nhiều bản ghi con của kiểu S.

2.6.3. Một số vấn đề thường gặp trong mô hình phân cấp

- Như đã đề cập ở trên, trong mô hình phân cấp một nút có thể xuất hiện nhiều lần trong rừng. Điều này dẫn đến một số vấn đề sau:

Lãng phí không gian do dữ liệu bị lặp lại nhiều lần trong bản ghi.

Sự thiếu nhất quán có thể xảy ra khi chúng ta cập nhật dữ liệu.

- Trong mô hình mạng, ta có thể đi trên các đường nối theo hai cách:hoặc là từ một nút nào đó đi đến nút sở hữu nó hoặc là đi đến nút được nó sở hữu. Trong khi đó các đường nối trong mô hình phân cấp được coi là chỉ đi theo một cách: từ nút cha tới nút con.

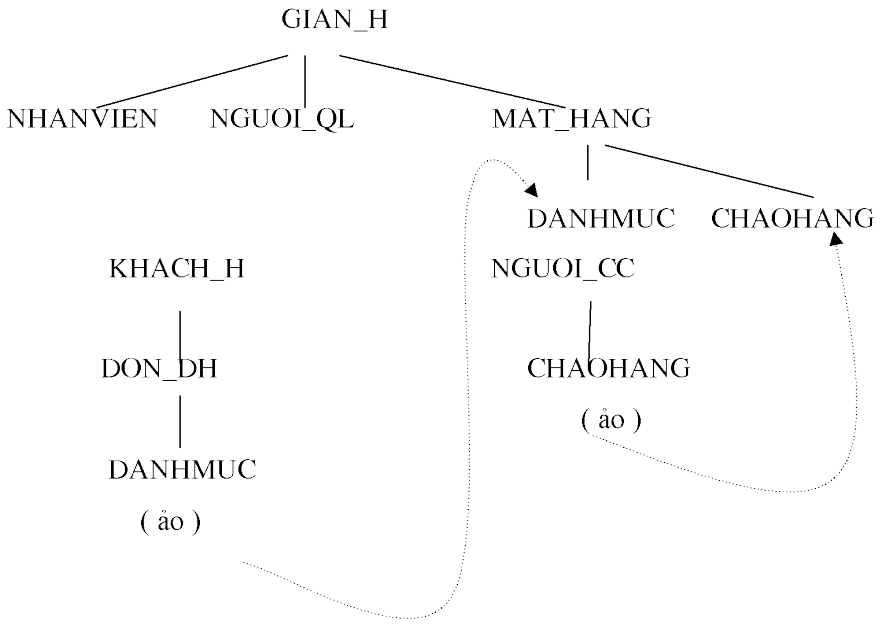
2.6.4. Các kiểu bản ghi ảo (virtual record record)

Ta có thể giải quyết các vấn đề được đề cập ở trên theo cùng cách. Trong mỗi lược đồ, ta đòi hỏi mỗi kiểu bản ghi phải chỉ xuất hiện một lần. Nếu như ta bỏ

sung vào một nút cho một kiểu bản ghi mà nút đó đã có trong rừng thì ta sẽ đặt vào tại đó một bản ghi ảo của kiểu đó. Trong một thể hiện, thay vì một bản ghi vật lý, ta cho nó là một con trỏ chỉ đến vị trí của bản ghi vật lý trong CSDL. Khi đó thủ tục BUILD đã đề cập ở trên phải được thay đổi như sau:

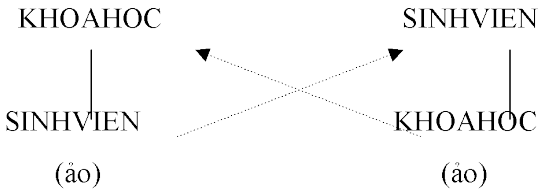
```
Procedure    BUILD(n);
 Chọn(n):=true;
 For   "mỗi đường nối từ nút m nào đó đến n"   do
       If   not chọn(m)  then
          Begin
              "Tạo m là một nút con của n";
                BUILD(m);
          End
        Else  " Tạo nút ảo m là một nút con của n"
End;
```

Theo phương pháp này, mô hình phân cấp của YVCB như ở trên sẽ trở thành như sau:



Các kiểu bản ghi ảo còn giải quyết được vấn đề về việc di chuyển trên các đường nối theo cả hai hướng. Nếu chúng ta có một mối quan hệ nhiều một từ kiểu bản ghi R đến kiểu bản ghi S, ta có thể tạo R là con của S và sau đó tạo S ảo là con của R. Nếu ta có mối quan hệ nhiều nhiều giữa R và S, ta không thể tạoể là con của S cũng như S là con của R, tuy nhiên ta có thể cho R và S tại vị trí tự nhiên của chúng trong rừng và sau đó tạo con của mỗi nút là phiên bản ảo của nút kia.

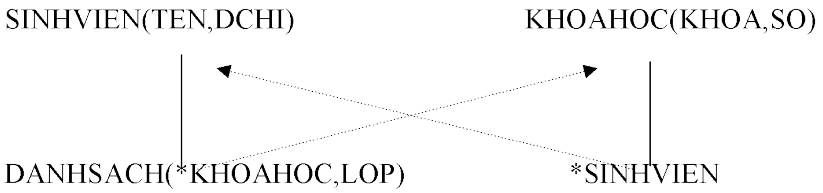
Ví dụ: Giả sử chúng ta có mối quan hệ nhiều-nhiều giữa KHOAHOC và SINHVIEN. Khi đó thay vì tạo một kiểu bản ghi mới làm trung gian giữa KHOAHOC và SINHVIEN, trong mô hình phân cấp ta có thể tạo một lược đồ với hai cây như sau:



2.6.5. Các kiểu bản ghi kết hợp (combined record type)

Để giải quyết vấn đề liên quan đến việc duyệt nhanh qua các đường dẫn tuỳ ý mà nhà thiết kế lược đồ CSDL tin là sẽ xảy ra trong thực tế, ta thường cần các bản ghi kết hợp chứa một số trường thông thường để lưu trữ dữ liệu và các trường khác là các con trỏ chỉ đến các kiểu bản ghi khác. Cũng như mọi kiểu bản ghi ảo, nếu chúng ta đang ở một bản ghi r có chứa một trường thuộc kiểu "T ảo", ta có thể đi theo con trỏ đó như một bản ghi của kiểu T là một con của r.

Ví dụ: Giả sử chúng ta muốn lưu trữ lớp trong bản ghi danh sách trong việc sắp xếp các sinh viên và các khoá học. Ta có thể sửa đổi lược đồ trong ví dụ vừa nêu trên thông qua việc thay thế KHOAHOC "ảo" con của SINHVIEN bằng một bản ghi kết hợp có một trường KHOAHOC ảo cũng như trường một trường LOP. Hình dưới đây là lược đồ có được theo cách này (trong đó ta sử dụng *T tiêu biểu cho một bản ghi ảo của T)



Tóm tắt chương 2

* Các bước cơ bản và sản phẩm cơ bản của thiết kế cơ sở dữ liệu là gì?

* Mục đích của mô hình quan hệ thực thể là cho phép *mô tả lược đồ khái niệm, trừu tượng hóa cấu trúc* và *các ràng buộc* của một cơ sở dữ liệu mà không cần chú ý đến tính hiệu quả hay thiết kế vật lý như phần lớn ở các mô hình khác.

* Để mô tả thông tin về các tập thực thể và mối quan hệ giữa các tập thực thể, người ta sử dụng sơ đồ mối quan hệ thực thể.

* Mô hình dữ liệu mạng là mô hình thực thể quan hệ trong đó các mối quan hệ là nhị phân nhiều-một.

* Mô hình phân cấp đơn giản chỉ là mô hình mạng có cấu trúc như là một rừng. Trong mô hình phân cấp, các đường nối chỉ theo hướng từ nút cha tới nút con.

Câu hỏi ôn tập chương 2

1- Hãy nói về vai trò của mô hình dữ liệu bậc cao trong quá trình thiết kế cơ sở dữ liệu.

2- Liệt kê các trường hợp cần phải sử dụng giá trị null trong giá trị của thuộc tính.

3- Định nghĩa các thuật ngữ sau: thực thể, thuộc tính, giá trị thuộc tính, thể hiện quan hệ, thuộc tính phức hợp, thuộc tính đa trị, thuộc tính suy diễn được, thuộc tính phức tạp, thuộc tính khoá, miền giá trị.

4- Kiểu thực thể là gì? Tập thực thể là gì? Giải thích sự khác nhau giữa một thực thể, một kiểu thực thể và một tập thực thể.

5- Giải thích sự khác nhau giữa một thuộc tính và một tập giá trị.

6 - Kiểu quan hệ là gì? Giải thích sự khác nhau giữa một thể hiện quan hệ, một tập quan hệ và một kiểu quan hệ.

7- Vai trò tham gia là gì? Khi nào cần phải sử dụng các tên vai trò trong mô tả các kiểu quan hệ.

8- Mô tả cách chỉ ra các ràng buộc cấu trúc trên các kiểu quan hệ.

9- Với điều kiện nào một thuộc tính của một kiểu quan hệ cấp 2 có thể chuyển thành một thuộc tính của một trong các kiểu thực thể tham gia vào kiểu quan hệ.

10- Khi chúng ta nghĩ đến các quan hệ như là các thuộc tính, các tập giá trị của các thuộc tính đó là gì?

11- Kiểu quan hệ đệ quy là gì? Cho một số ví dụ về các kiểu quan hệ đệ quy.

12- Khi nào khái niệm kiểu thực thể yếu được dùng trong mô hìn hoá cơ sở dữ liệu? Định nghĩa các thuật ngữ: kiểu thực thể chủ, kiểu thực thể yếu, khoá bộ phận, kiểu quan hệ xác định.

13- Trình bày các khái niệm lớp, lớp con, chuyên biệt hoá, tổng quát hoá. Trong hoàn cảnh nào ta cần tách một lớp thành các lớp con.

14- Trình bày cách biểu diễn đồ hoạ của các mô hình ER.

15- Nêu ý nghĩa của sơ đồ ER và EER (Các loại sơ đồ dùng để làm gì? do ai vẽ và vẽ cho ai?).

Bài tập chương 2

1) Tìm một số ví dụ về thực thể yếu.

2) *Vẽ mô hình ER thể hiện các mối quan hệ sau, xác định lực lượng tham gia vào quan hệ, loại quan hệ (bắt buộc hay tùy chọn) cho mỗi quan hệ dưới đây:*

   Vợ với chồng.

   Sinh viên và bảng kết quả thi.

   Con cái và cha mẹ.

   Cầu thủ và đội bóng.

   Sinh viên và khóa học.

3) *Vẽ mô hình quan hệ thực thể cho mỗi yêu cầu sau:*

Trường ĐHSP Huế cần quản lý các sinh viên. Mỗi kỳ thi trường cần in ra danh sách sinh viên được sắp xếp theo tên. Về sinh viên cần lưu: Mã sv, Họ và tên sv, ngày sinh, địa chỉ, khoa và khoá học (ví dụ khoá 8 hoặc khoá 9).

Công ty ABC cần lưu trữ thông tin để hàng năm vào ngày 1 – 6 in ra danh sách các cháu dưới 15 tuổi (con của nhân viên công ty) được nhận quà gồm các thông tin: Họ tên bố/ mẹ, Họ tên con, Tuổi, Số tiền, Ký nhận. Một nhân viên có thể có nhiều con, không có trường hợp cả hai vợ chồng cùng làm trong công ty.

Công ty cần quản lý việc phân công nhân sự vào các dự án. Một nhân viên có thể tham gia vào một hay nhiều dự án, hoặc không tham gia vào dự án nào cả. Một nhân viên không thể tham gia 2 dự án cùng thời gian. Mỗi dự án phải có ít nhất một nhân viên tham gia. Nhân viên cần lưu: Mã NV, Tên NV, Địa chỉ. Mỗi dự án có: Mã dự án, Tên dự án, Ngày bắt đầu và ngày kết thúc.

Sở địa chính quốc gia cần xây dựng CSDL để quản lý hộ khẩu theo chủ hộ. Một thành phố có nhiều quận. Một quận có nhiều phường. Một phường có nhiều hộ. Không có hai hộ ở cùng một địa chỉ.

Một siêu thị có nhu cầu lưu trữ các thông tin về các mặt hàng được bán trong siêu thị. Siêu thị kinh doanh nhiều ngành hàng như thực phẩm, may mặc, đồ gia dụng... Mỗi ngành hàng có nhiều loại hàng như thực phẩm thì có các loại hàng như rau quả, mỳ ăn liền hoặc bánh kẹo... và mỗi loại hàng có nhiều mặt hàng như loại hàng bánh kẹo có các mặt hàng như bánh ngọt Chocopie, kẹo hoa quả Quảng Ngãi...

Một công ty có nhu cầu quản lý việc bán hàng. Mỗi khi khách mua hàng muốn thanh toán tiền hàng, nhân viên bán hàng sẽ lập một hóa đơn thanh toán có dạng sau:

Công ty TNHH Alpha-Beta

HOÁ ĐƠN THANH TOÁN

Số hóa đơn: HD00124 Ngày mua: 09/12/2010

Tên khách hàng: Nguyễn Tèo Địa chỉ: 176, Trương Gia Mô, Huế

Mã hàng	Tên hàng	ĐVT	Số lượng	Đơn giá	Thành tiền
MH001	Máy in Cannon	chiếc	01	2560000	2560000
MH002	Chuột quang	chiếc	02	152000	304000

4) *Với mỗi khẳng định dưới đây, xác định 2 tập thực thể và 1 quan hệ, chỉ rõ lực lượng, sự tồn tại của mỗi quan hệ đó, vẽ mô hình ER. Chú ý: Nếu khẳng định không đủ xác định các yêu cầu trên, chỉ ra giả thiết để yêu cầu đó được rõ nghĩa.*

1. Một phòng ban tuyển dụng nhiều nhân viên, một nhân viên được tuyển bởi nhiều nhất một phòng ban.
2. Một nhà quản lý quản lý nhiều nhất một phòng ban, một phòng ban được quản lý bởi nhiều nhất một nhà quản lý.
3. Một tác giả có thể viết nhiều quyển sách, một quyển sách có thể được viết bởi nhiều tác giả.
4. Một đội bóng bao gồm nhiều cầu thủ, mỗi cầu thủ chơi trong duy nhất một đội bóng.
5. Một giảng viên dạy nhiều nhất một khóa học, một khóa học được giảng bởi chính xác một giảng viên.
6. Một chuyến bay kết nối 2 sân bay, mỗi sân bay được sử dụng cho nhiều chuyến bay.
7. Một đơn mua hàng có thể có nhiều sản phẩm, một sản phẩm có thể xuất hiện trong nhiều đơn mua hàng.
8. Mỗi khách hàng có thể yêu cầu nhiều đơn đặt hàng, một đơn đặt hàng chỉ thuộc về duy nhất một khách hàng.

5) Dựa vào các phân tích sơ bộ dưới đây, hãy lập mô hình quan hệ thực thể (gồm loại thực thể, mối quan hệ, bản số, thuộc tính của loại thực thể, khoá của loại thực thể) cho mỗi bài toán quản lý sau:

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).

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, ….

6) Thiết kế cơ sở dữ liệu cho một bảo tàng nghệ thuật với các yêu cầu sau.

• Bảo tàng có một bộ sưu tập các tác phẩm nghệ thuật (TPNT). Mỗi TPNT có một mã số duy nhất (MaTP), một tác giả (Tacgia), một năm sáng tác (Nam) nếu có, một chủ đề (Chude) và một lời diễn giải (Chugiai). Các tác phẩm nghệ thuật được phân loại theo nhiều cách được mô tả dưới đây.

• Các TPNT được phân loại dựa trên loại hình sáng tác. Có 3 loại hình sáng tác chính sau: hội họa (HOIHOA), điêu khắc (DIEUKHAC) và tạc tượng (TACTUONG). Ngoài ra còn các loại hình khác (KHAC).

• Loại hình HOIHOA được thể hiện bởi chất liệu (Chatlieu) như sơn dầu, màu nước,… , vật liệu (Vatlieu) như giấy, vải, gỗ… và trường phái (Truongphai) như hiện đại, ấn tượng,…

• Loại hình DIEUKHAC và TACTUONG được thể hiện bởi vật liệu (Vatlieu) như gỗ, đá…, chiều cao (Cao), khối lượng (Khoiluong) và phong cách (Phongcach).

• Tác phẩm nghệ thuật thuộc loại hình KHAC được thể hiện bởi thể loại (Theloai) như ảnh chụp,… và phong cách (Phongcach).

• Các TPNT còn được phân thành loại sở hữu riêng (SOHUU) là tài sản của bảo tàng được mô tả bởi ngày sở hữu tác phẩm (NgaySohuu), tình trạng (Tinhtrang) đang được trưng bày hay lưu trong kho, trị giá (Trigia); hoặc loại đi mượn (MUON) là tác phẩm được mượn từ bộ sưu tập khác (BoSuuTap), ngày mượn (NgayMuon), ngày trả (NgayTra).

• Các TPNT cũng có thông tin mô tả về quốc gia xuất xứ (Xuatxu), mô tả thời đại (Thoidai) chẳng hạn phục hưng, hiện đại, cổ đại …

• Thông tin về tác giả (TACGIA) nếu có như tên (Ten), ngày sinh (Ngaysinh), ngày mất (Ngaymat), quốc tịch (Quoctich), thời đại (Thoidai), chuyên môn (Chuyenmon) và diễn giải (Chugiai). Giả sử tên của các TACGIA là duy nhất.

• Các cuộc triển lãm (TRIENLAM) được xác định bởi tên (Ten), ngày mở cửa (NgayMo) và ngày đóng cửa (NgayDong). Các TRIENLAM liên quan đến tất cả các tác phẩm nghệ thuật được trưng bày trong cuộc triển lãm đó.

• Thông tin về các bộ sưu tập khác (BOSUUTAP) mà bảo tàng có liên hệ để mượn tác phẩm được mô tả bởi tên duy nhất (Ten), hình thức sưu tập (Hinhthuc) chẳng hạn bảo tàng, cá nhân,… , diễn giải (Chugiai), địa chỉ (Diachi), số điện thoại (Dienthoai) và người giao dịch (Doitac).

Xây dựng sơ đồ thực thể - quan hệ mở rộng (EER). Trình bày các giả định của bạn và giải thích các lựa chọn trong việc thiết kế EER của bạn.

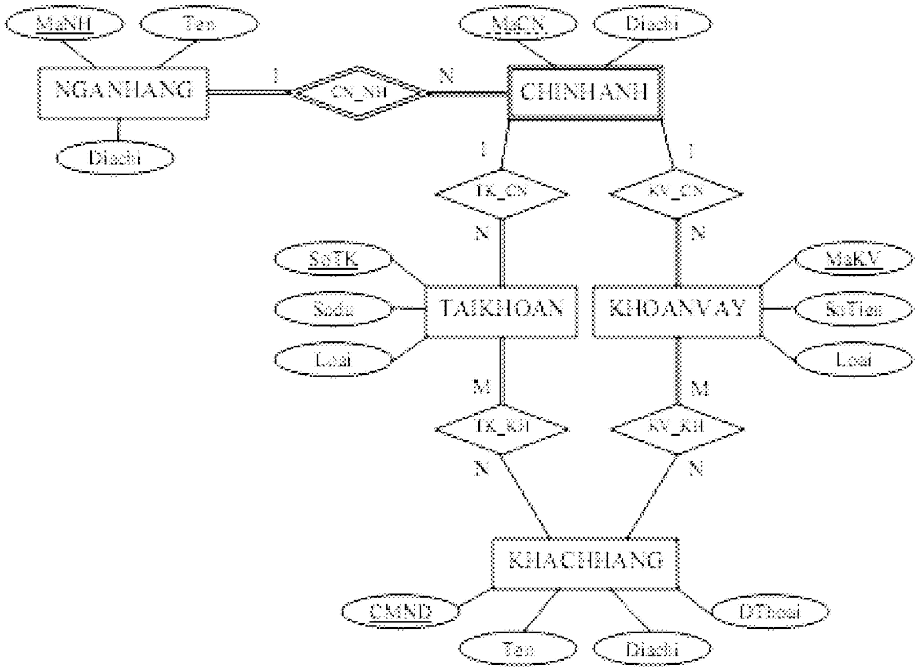
7) Xét sơ đồ ER biểu diễn một phần cơ sở dữ liệu của một ngân hàng ở dưới đây, xét xem.

• Có các kiểu thực thể yếu trong lược đồ hay không? Nếu có cho biết tên, khóa bộ phận và quan hệ định danh của các kiểu thực thể yếu đó.

• Cho biết tên tất cả các kiểu quan hệ và xác định ràng buộc (min, max) của mỗi kiểu thực thể tham gia vào các kiểu quan hệ.

• Giả sử mỗi khách hàng có ít nhất một tài khoản nhưng chỉ có nhiều nhất hai khoản vay tại cùng một thời điểm và một chi nhánh ngân hàng không thể có

nhiều hơn 1000 khoản vay. Điều này được thể hiện bằng ràng buộc (min, max) như thế nào?



8) Xét lược đồ trong bài 7 và giả sử rằng cần phải lưu các loại tài khoản khác nhau như tài khoản tiết kiệm (TK_TIETKIEM), tài khoản vãng lai (TK_ VANGLAI), ... và các khoản vay khác nhau như khoản vay mua nhà (KV_ NHA), khoản vay mua ôtô (KV_OTO), ... Giả sử muốn lưu lại các giao dịch (GIAODICH) trên mỗi tài khoản như rút tiền, gởi tiền, kiểm tra, ... và các thanh toán (THANHTOAN) của mỗi khoản vay; mỗi giao dịch và thanh toán có thông tin gồm số tiền, thời điểm thực hiện. Sử dụng EER và các khái niệm chuyên biệt hóa và tổng quát hóa để bổ sung lược đồ.

Chương 3. MÔ HÌNH CƠ SỞ DỮ LIỆU QUAN HỆ

Mục đích

Trình bày một loại mô hình cơ sở dữ liệu ở mức logic – mô hình cơ sở dữ liệu quan hệ.

Yêu cầu

Sinh viên nắm vững các khái niệm và các phép toán cơ bản của mô hình dữ liệu quan hệ.

Các quy tắc chuyển đổi từ sơ đồ ERD sang lược đồ quan hệ.

Các phép toán đại số quan hệ.

Vận dụng trong việc mô tả dữ liệu.

3.1. Mô hình dữ liệu quan hệ

Từ những năm 70 của thế kỷ 20 mô hình dữ liệu quan hệ đã được đưa ra lần đầu tiên trong bài báo của E. F. Codd *"A Relation Model for Large Shared Data Banks"*, Communications of ACM, 6/1970. Mô hình cung cấp một cấu trúc dữ liệu đơn giản và đồng bộ dựa trên khái niệm ***quan hệ,*** có nền tảng lý thuyết vững chắc là lý thuyết tập hợp. Hơn nữa do tính trực quan, kiến trúc đơn giản, ngoài ra người ta cũng đã chứng minh sự tương đương và cung cấp những phép biến đổi giữa mô hình quan hệ với các mô hình mạng và phân cấp. Mô hình này là cơ sở của các Hệ quản trị CSDL thương mại như Oracle, DB2, SQL Server… Vì vậy mô hình dữ liệu quan hệ được xem là *mô hình dữ liệu logic* thông dụng hiện nay, mặc dù ngày nay đang có nhiều mô hình dữ liệu tiên tiến được nghiên cứu triển khai nhưng vẫn chưa được thương mại hóa rộng rãi, đồng thời mô hình dữ liệu quan hệ vẫn được xem là một nền tảng cho các mô hình dữ liệu tiên tiến vì những lý do trên.

Cũng nên lưu ý rằng, các khái niệm hình thức của các mô hình dữ liệu logic còn là cơ sở toán học cho ngôn ngữ truy vấn dữ liệu và các vấn đề có liên quan.

3.1.1. Nhìn nhận quan hệ trên quan điểm lý thuyết tập hợp

3.1.1.1. Quan hệ, thuộc tính, bộ

Định nghĩa 3.1. Cho tập hữu hạn $U = \{A_1, A_2, \ldots, A_n\}$ khác rỗng ($n \geq 1$). Các phần tử của U được gọi là *thuộc tính*. Ứng với mỗi thuộc tính $A_i \in U$, $i=1,2,\ldots,n$ có một tập chứa ít nhất hai phần tử $dom(A_i)$ được gọi là miền trị của thuộc tính A_i. Gọi D là hợp của các $dom(A_i)$, $i=1,2,\ldots,n$. Một *quan hệ* R với các thuộc tính $U = \{A_1, A_2, \ldots, A_n\}$, ký hiệu là R(U), là một tập các ánh xạ $t: U \rightarrow D$ sao cho với mỗi $A_i \in U$ ta có $t(A_i) \in dom(A_i)$. Mỗi ánh xạ được gọi là một *bộ* của quan hệ R.

Có thể xem một quan hệ là một tập con của tập tích Descartes $dom(A_1) x dom(A_2) x \ldots x dom(A_n)$.

Mỗi quan hệ R(U) có hình ảnh là một *bảng*, mỗi *cột* ứng với một thuộc tính, mỗi *dòng* là một bộ.

Ta ký hiệu t[U] là một bộ trên tập thuộc tính U.

Một quan hệ rỗng, ký hiệu $\varnothing$, là quan hệ không chứa bộ nào.

Vì mỗi quan hệ là một tập các bộ nên trong quan hệ không có hai bộ trùng lặp, nhưng thứ tự các bộ trong quan hệ là không quan trọng.

Cần lưu ý thứ tự các thuộc tính trong tích Descartes nói trên trong thực tiễn là không bắt buộc, mặc dù ta biết tích Descartes không có tính giao hoán.

3.1.1.2. Các ký hiệu và một số quy ước

Nếu chúng ta đặt tên cho một quan hệ là R và lược đồ quan hệ của nó có các thuộc tính $A_1, A_2, \ldots, A_k$ thì ta sẽ biểu diễn lược đồ quan hệ như sau: $R(A_1, A_2, \ldots, A_k)$

Theo truyền thống của lý thuyết cơ sở dữ liệu chúng ta chấp nhận các quy định sau đây:

Các thuộc tính được ký hiệu bằng các chữ *LATIN HOA* đầu bảng chữ *A*, *B*, *C*,...

Tập thuộc tính được ký hiệu bằng các chữ *LATIN HOA* cuối bảng chữ *X*, *Y*, *Z*,...

Các phần tử trong một tập thường được liệt kê như một xâu ký tự, không có các ký hiệu biểu diễn tập, chẳng hạn ta viết $X=ABC$ thay vì viết $X=\{A,B,C\}$. XY biểu diễn hợp của hai tập *X* và *Y*, $X \cup Y$. Phép trừ hai tập *X* và *Y* được ký hiệu là $X \setminus Y$, hoặc $X-Y$.

Một phân hoạch của tập M (thành các tập con rời nhau và có hợp là M), $X_1, X_2, ..., X_m$ được ký hiệu là

M=$X_1 | X_2 | ... | X_m$ với ý nghĩa:

$M = X_1 \cup X_2 \cup ... \cup X_m$ và $X_i \cap X_j = \emptyset$, $1 \le i,j \le m$, $i \ne j$

Các bộ được biểu diễn bằng các chữ *Latin thường* có thể kèm chỉ số *t, u, v, t_1 ...*

Với mỗi bộ *t* trong quan hệ $R(U)$ và mỗi tập con các thuộc tính $X \subseteq U$ ta ký hiệu $t[X]$ hoặc $t.X$ là hạn chế của bộ *t* trên tập thuộc tính *X*.

Ta chấp nhận quy ước tự nhiên là miền trị của mọi thuộc tính chứa ít nhất hai phần tử. Trong trường hợp một miền trị của thuộc tính chỉ chứa một giá trị duy nhất thì ta có thể loại bỏ cột tương ứng của thuộc tính đó trong quan hệ.

Ta chấp nhận quy ước sau đây: Mọi cặp bộ t và v trong mọi quan hệ giống nhau trên miền rỗng các thuộc tính, $t.\emptyset = v.\emptyset$.

Ví dụ:

Xét quan hệ NHANVIEN cho thông tin về các nhân viên của một đơn vị bao gồm tên nhân viên (TNV), ngày sinh (NGAYSINH), chuyên môn (CM), số điện thoại (SDT) như sau:

NHANVIEN

TNV	NGAYSINH	CM	SDT
Lê Vũ	02/07/68	Chương trình dịch	827369
Lê Công Tuấn Anh	09/07/66	Cơ sở dữ liệu	823073
Lê Thanh Tấn	12/08/68	Mạng máy tính	529753
Trần Thúy An	04/12/74	Hệ thống thông tin	824237
Lê Hữu Hòa	09/11/75	Mạng máy tính	824243
Nguyễn Ngọc Thủy Tiên	15/08/76	Mạng Máy Tính	523957
Vũ Thị Giáng Hương	05/12/67	Trí tuệ nhân tạo	521321

Như vậy lược đồ quan hệ NHANVIEN có 4 thuộc tính TNV, NGAYSINH, CM, SDT và do đó mỗi bộ của quan hệ có 4 thành phần: tên nhân viên, ngày sinh, chuyên môn, số điện thoại.

Miền NGAYSINH là tập các giá trị ngày tháng sao cho các bộ trong quan hệ còn là nhân viên.

Miền SDT là tập các giá trị là các số nguyên có 6 hay 7 chữ số thõa mãn đó là các số điện thoại.

Rõ ràng các bộ được ghi trong bảng nhân viên phải thỏa quan hệ họ là các nhân viên.

3.1.2. Lược đồ quan hệ

Một lược đồ quan hệ R là một cặp có thứ tự R=<U,F>, trong đó U là tập hữu hạn các thuộc tính và F là tập các điều kiện giữa các thuộc tính (F còn gọi là tập các ràng buộc toàn vẹn).

Một lược đồ quan hệ được sử dụng để mô tả về cấu trúc và các ràng buộc toàn vẹn của một quan hệ, đó là nội dung của CSDL và các tính chất bất biến tương đối của một quan hệ. Trong khi đó quan hệ là một thể hiện của CSDL, là dữ liệu có trong một bảng và có thể thay đổi theo thời gian, nhưng phải thỏa mãn các ràng buộc của lược đồ.

Một lược đồ CSDL trong mô hình dữ liệu quan hệ bao gồm một hay nhiều lược đồ quan hệ.

3.1.3. Ràng buộc toàn vẹn

Ràng buộc toàn vẹn (RBTV) (Integrity Constraint) là những qui tắc, điều kiện, ràng buộc cần được thỏa mãn cho mọi thể thiện của CSDL quan hệ. RBTV được mô tả khi định nghĩa lược đồ quan hệ. RBTV được kiểm tra khi các quan hệ có thay đổi.

3.1.4. Siêu khóa (Super key)

Gọi SK là một tập con khác rỗng các thuộc tính của R, SK là siêu khóa khi:

$$\forall r, \forall t_1, t_2 \in r, t_1 \neq t_2 \Rightarrow t_1[SK] \neq t_2[SK]$$

Nhận xét:

Siêu khóa là tập các thuộc tính dùng để xác định tính duy nhất của mỗi bộ trong quan hệ.

Mọi lược đồ quan hệ có tối thiểu một siêu khóa.

3.1.5. Khóa

Gọi K là một tập con khác rỗng các thuộc tính của R

K là khóa nếu thỏa đồng thời 2 điều kiện :

- *K là một siêu khóa của R*
- $\forall K' \subset K, K' \neq K \Rightarrow K'$ *không phải là siêu khóa của R.*

Nhận xét:

Giá trị của khóa dùng để nhận biết một bộ trong quan hệ.

Khóa là một đặc trưng của lược đồ quan hệ, không phụ thuộc vào thể thiện quan hệ.

Khóa được xây dựng dựa vào ý nghĩa của một số thuộc tính trong quan hệ.

Lược đồ quan hệ có thể có nhiều khóa, việc chọn một khoá và xem nó như một khoá duy nhất rất có ích; chẳng hạn rất nhiều cấu trúc lưu trữ vật lý cần có một khoá duy nhất. Do đó, thuật ngữ *khoá chính* (primary key) sẽ được sử dụng để chỉ đến khoá được chọn ra từ một tập các khoá gọi là *khoá ứng viên* (candidate key).

Lưu ý: *Các thuộc tính khóa chính phải có giá trị khác null ?*

Ví dụ:

Xét quan hệ

NHANVIEN(MANV, TENNV, HONV, NGSINH, DCHI, PHAI, LUONG, PHONG). Có 2 khóa là K_1=MANV hay K_2= {HONV, TENNV, NGSINH}. Khi cài đặt quan hệ thành bảng (table) ta nên chọn 1 khóa làm cơ sở để nhận biết các bộ. Khóa K_1=MANV được chọn gọi là khóa chính.

3.1.6. Tham chiếu và khái niệm khóa ngoại (Foreign key)

Một bộ trong quan hệ R, tại thuộc tính A nếu nhận một giá trị từ một thuộc tính B của quan hệ S, ta gọi R tham chiếu S.

Ví dụ: Xét quan hệ Sinh viên(MSV,HOTEN, DIEM, MAKHOA) và quan hệ Khoa(MAKHOA, TENKHOA, SDT) khi đó thuộc tính MAKHOA trong quan hệ sinh viên được tham chiếu từ thuộc tính MAKHOA trong quan hệ Khoa.

Xét 2 lược đồ R và S, gọi FK là tập thuộc tính khác rỗng của R, FK là ***khóa ngoại (Foreign Key)*** của R khi:

- Các thuộc tính trong FK phải có cùng miền giá trị với các thuộc tính khóa chính của S

- Giá trị tại FK của một bộ $t_1 \in R$

+ *Hoặc bằng giá trị tại khóa chính của một bộ $t_2 \in S$*

+ *Hoặc bằng giá trị rỗng.*

Trong ví dụ trên thì MAKHOA là khóa ngoại của quan hệ Sinh viên.

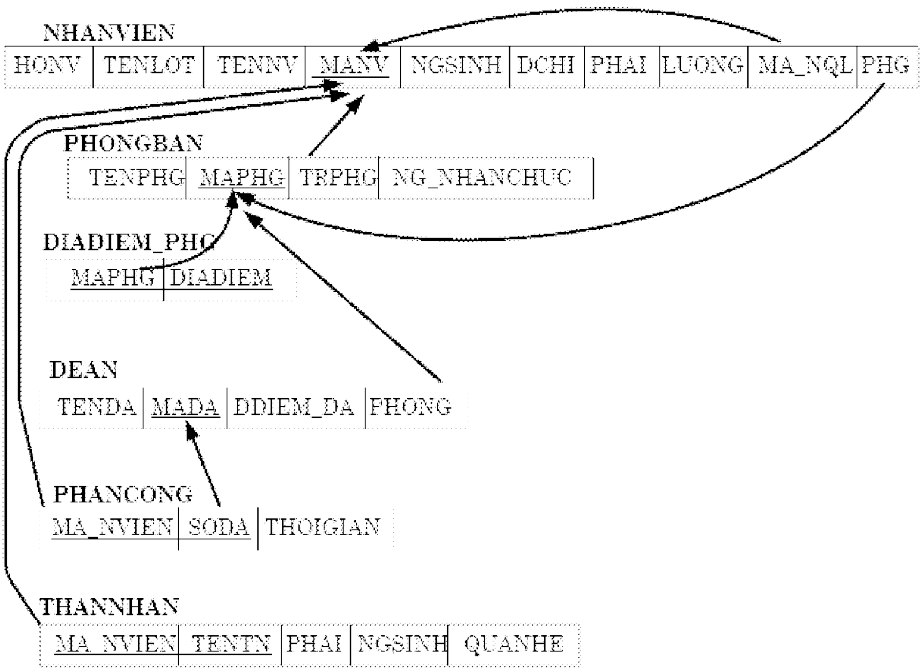
Nhận xét:

- Trong một lược đồ quan hệ, một thuộc tính vừa có thể tham gia vào khóa chính, vừa tham gia vào khóa ngoại

- Khóa ngoại có thể tham chiếu đến khóa chính trên cùng 1 lược đồ quan hệ

- Có thể có nhiều khóa ngoại tham chiếu đến cùng một khóa chính

- Ràng buộc tham chiếu = Ràng buộc khóa ngoại.



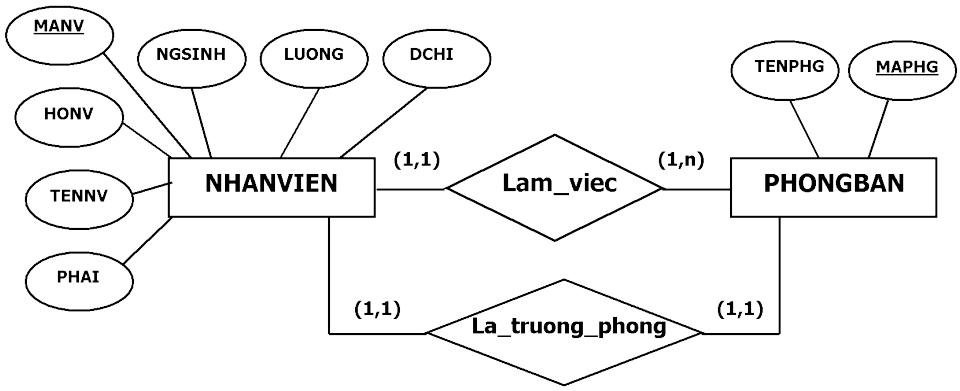
3.2. Chuyển đổi từ sơ đồ thực thể - quan hệ (ERD) sang lược đồ quan hệ

Nếu xem mô hình ER là mô hình dữ liệu ngoài (khái niệm) và mô hình dữ liệu quan hệ là mô hình logic, ta có một số qui tắc chuyển đổi từ sơ đồ thực thể - quan hệ (ERD) sang lược đồ quan hệ, nhằm chuyển đổi một cách "*hợp lý*" các dữ liệu ở mức khái niệm sang mức logic để tổ chức lưu trữ và xử lý.

Qui tắc 1: Biến đổi một tập thực thể thành một quan hệ.

Đối với tập thực thể thông thường (regular entity type): khóa của quan hệ là khóa của tập thực thể.

Lưu ý : Thuộc tính của quan hệ là thuộc tính của tập thực thể ; Quan hệ chỉ chứa các thuộc tính thành phần của thuộc tính phức hợp; Quan hệ không chứa các thuộc tính đa trị.



Được chuyển thành các lược đồ tương ứng:

PHONGBAN(TENPHG, MAPHG)

NHANVIEN(MANV, TENNV, HONV, NGSINH, DCHI, PHAI, LUONG)

Qui tắc 2: Biến đổi thuộc tính đa trị thành một quan hệ.

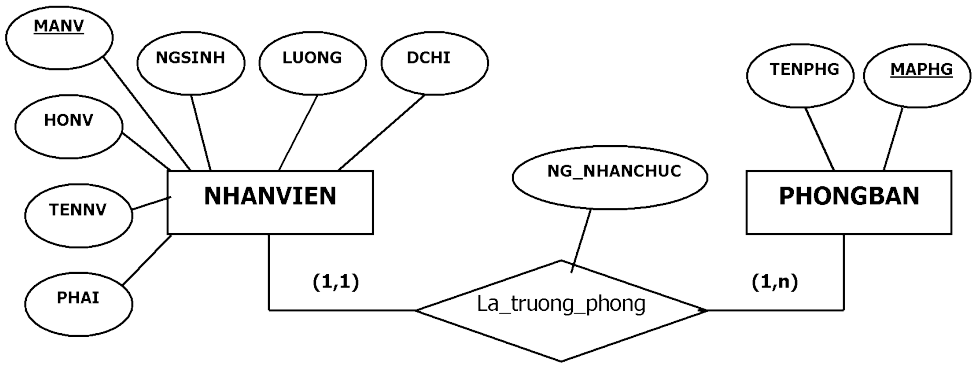
Quan hệ chứa khóa của tập thực thể và thuộc tính đa trị.

Khóa của quan hệ gồm khóa của tập thực thể và thuộc tính đa trị.

Ví dụ: Xét mô hình thực thể quản lý CÔNG TY với kiểu thực thể ĐƠNVỊ. Kiểu thực thể này có thuộc tính ĐịađiểmĐV là một thuộc tính đa trị (có thể nhận một lúc nhiều giá trị - vì một đơn vị có thể có nhiều địa điểm đặt khác nhau). Khi chuyển thành mô hình quan hệ nó sẽ được chuyển thành một quan hệ có khoá chính là MãsốĐV, Địa điểm và có thể có thêm một số thuộc tính khác lưu thông tin về địa điểm.

Qui tắc 3: Biểu diễn mối quan hệ 1-ngôi (đệ qui) hoặc 2-ngôi có quan hệ một-một.

Đặt khóa của kiểu thực thể bên phía bắt buộc và các thuộc tính của mối quan hệ vào quan hệ của kiểu thực thể bên phía tùy chọn.

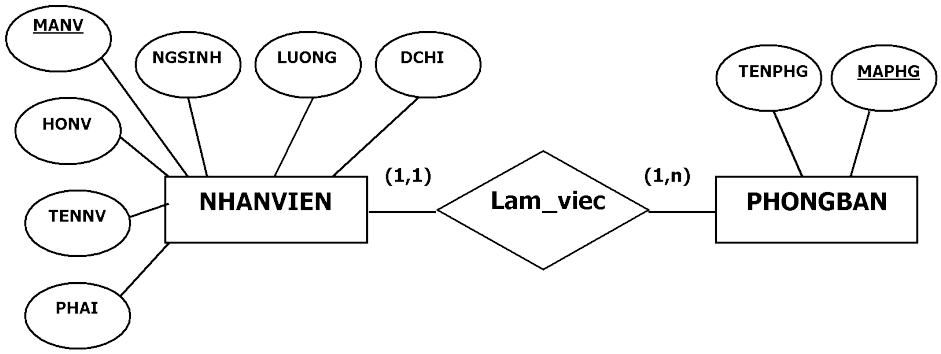


Khi chuyển đổi về các lược đồ quan hệ thì quan hệ PHONGBAN cần được lưu ý.

PHONGBAN(<u>MAPHG</u>, TENPHG, **MANV, NG_NHANCHUC**)

<u>Qui tắc 4</u>: Biểu diễn mối quan hệ 1-ngôi hoặc 2-ngôi có quan hệ một-nhiều.

Đặt khóa của kiểu thực thể bên phía một và các thuộc tính của mối quan hệ vào quan hệ của kiểu thực thể bên phía nhiều.



Khi đó ta chuyển MAPHG vào lược đồ NHANVIEN lam khoa ngoại của lược đồ này..

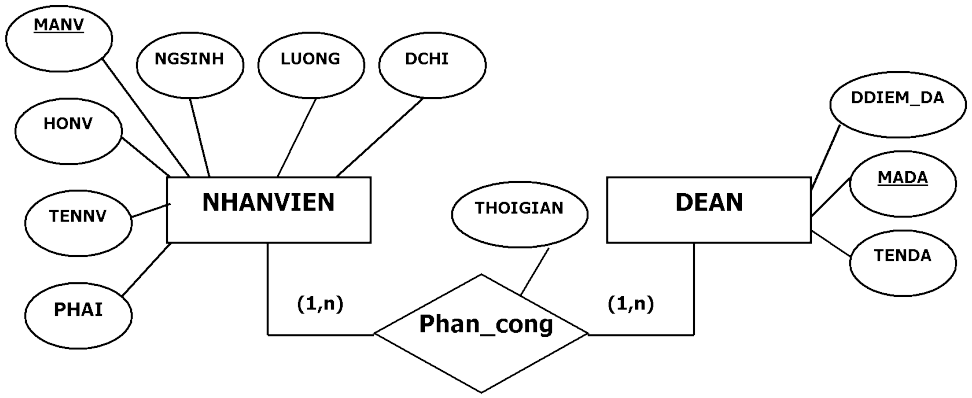
NHANVIEN(<u>MANV</u>, TENNV, HONV, NGSINH, DCHI, PHAI, LUONG, ***MAPHG***)

<u>Qui tắc 5</u>: Biến đổi mối quan hệ 1-ngôi hoặc 2-ngôi có quan hệ nhiều-nhiều thành một quan hệ.

Quan hệ chứa các khóa của các kiểu thực thể tham gia vào mối quan hệ.

Khóa của quan hệ gồm cả hai khóa của hai kiểu thực thể.

Thuộc tính của quan hệ là thuộc tính của mối quan hệ.



Quan hệ Phan_cong được chuyển thành lược đồ quan hệ:

PHANCONG(MANV, MADA, THOIGIAN)

Qui tắc 6: Biến đổi mối quan hệ tam phân (3-ngôi) thành một quan hệ.

Quan hệ chứa ba khóa của ba kiểu thực thể tham gia vào mối quan hệ.

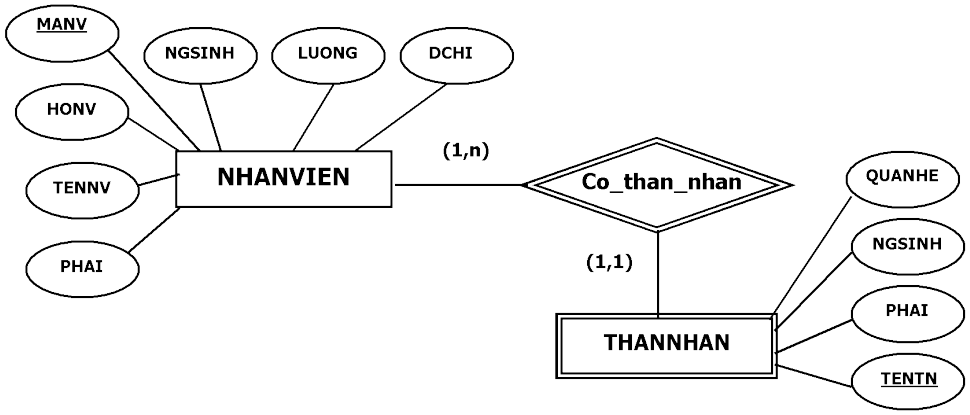
Mối quan hệ có bao nhiêu kiểu thực thể bên phía một thì quan hệ có bấy nhiêu khóa: đối với một kiểu thực thể bên phía một thì khóa của quan hệ gồm cả hai khóa của hai kiểu thực thể còn lại. Nếu không có kiểu thực thể bên phía một thì khóa của quan hệ bao gồm cả ba khóa của ba kiểu thực thể.

Thuộc tính của quan hệ là thuộc tính của mối quan hệ.

Ví dụ: Giả sử chúng ta có kiểu quan hệ ĐẠILÝ <cung cấp> VẬTTƯ cho DỰÁN. Đây là một kiểu quan hệ ba ngôi. Giả sử rằng kiểu thực thể ĐẠILÝ có thuộc tính khoá là MãsốĐL, kiểu thực thể VẬTTƯ có thuộc tính khoá là MãsốVT, kiểu thực thể DỰÁN có thuộc tính khoá là MãsốDA còn kiểu quan hệ <cung cấp> có thuộc tính là Sốlượng để lưu số lượng vật tư mà một đai lý cung cấp cho môt dự án. Khi đó kiểu quan hệ <cung cấp> sẽ được chuyển thành một quan hệ có tên là CUNGCẤP với các thuộc tính MãsốĐL, MãsốVT , MãsốDA, Sốlượng và khoá chính gồm ba thuộc tính MãsốĐL, MãsốVT, MãsốDA.

Qui tắc 7: Thực thể yếu

Chuyển thành một quan hệ có cùng tên với thực thể yếu, thêm vào thuộc tính khóa của quan hệ liên quan.



Ta có lược đồ sau tương ứng với thực thể yếu THANNHAN trong sơ đồ trên.

THANNHAN(**MANV**, TENTN, PHAI, NGSINH, QUANHE)

3.3. Khóa chung và bộ khuyết

- Khi hai quan hệ có cùng các khóa dự bị chung chúng ta có thể hợp các thuộc tính của hai lược đồ quan hệ và thay hai lược đồ này bằng một lược đồ mà tập thuộc tính là hợp của hai tập thuộc tính. Lợi ích của việc làm này là chúng ta tiết kiệm được không gian nhớ cần để lặp lại các giá trị khoá trong hai quan hệ. Hơn nữa, nhiều khi việc trả lời một số yêu cầu sẽ nhanh hơn nếu xử lý trên quan hệ đã hợp nhất này.

Với việc hợp nhất hai lược đồ, các quan hệ trên các lược đồ thành phần cũng có thể được hợp nhất để được một quan hệ trên lược đồ mới theo nguyên tắc các bộ có giá trị trùng nhau trên các thuộc tính khoá chung được nối lại với nhau.

Ví dụ. Các lược đồ quan hệ GH và QL có chung khoá TGH. Chúng ta ghép hai lược đồ này thành một lược đồ mới QLGH với 3 thuộc tính TGH, SGH, TQL (ở đây thuộc tính TNV được đổi NQL).

GH

TGH	SGH
Quần áo	001
Điện tử	003
Giày dép	005
Văn hoá phẩm	012

QL

TNV	TGH
Nguyễn Văn An	Giày dép
Trần Thu Trang	Điện tử
Lê Thanh Toàn	Quần áo
Hà Văn Trung	Văn hoá phẩm

QLGH

TGH	SGH	NQL
Quần áo	001	Lê Thanh Toàn
Điện tử	003	Trần Thu Trang
Giày dép	005	Nguyễn Văn An
Văn hoá phẩm	012	Hà Văn Trung

Trong việc hợp nhất lược đồ quan hệ GH với lược đồ QL chúng ta ngầm định rằng các quan hệ trên hai lược đồ này có cùng tập hợp các gian hàng. Trong thực tế điều này đôi lúc không phù hợp. Chẳng hạn, có thể có gian hàng thực phẩm, số 009 tạm thời chưa có trưởng gian hàng. Trong một số trường hợp khi hợp nhất 2 quan hệ trong lược đồ mới sẽ có một số bộ bị khuyết đi một số giá trị ứng với một số thuộc tính. Để khắc phục hiện tượng này các hệ QTCSDL quan hệ cho phép khai báo null ở các vị trí khuyết, với điều kiện là hai trường có giá trị null không thể so sánh với nhau, các bộ như thế ta gọi là các ***bộ khuyết***.

Nhận xét: Chúng ta có suy nghĩ gì về *tính đúng đắn* của các qui tắc chuyển đổi trên và thế nào là chuyển đổi từ sơ đồ ERD sang lược đồ quan hệ một cách "*hợp lý*".

3.4. Các phép toán trên mô hình dữ liệu quan hệ

Trong phần trước, ta đã tìm hiểu về biểu diễn toán học của một quan hệ, nó được coi như là hình thức nền tảng cho mô hình dữ liệu quan hệ. Phần này trình bày các phép toán được dùng kết hợp với mô hình đó. Có hai cách biểu diễn khác nhau được sử dụng cho việc diễn tả các phép toán trên quan hệ.

- Biểu diễn đại số, được gọi là *đại số quan hệ* (relational algebra), trong đó các câu hỏi được diễn tả bằng cách áp dụng các các phép toán đặc biệt vào các quan hệ.

- Biểu diễn logic, được gọi là *phép tính quan hệ* (relational caculus), trong đó các câu hỏi được diễn tả bằng các công thức logic mà các bộ trong câu trả lời phải thoả mãn.

Cả hai phương pháp này đều có khả năng diễn đạt như nhau, tức là một câu hỏi nếu như được biểu diễn bằng đại số quan hệ thì cũng có khả năng biểu diễn bằng phép tính quan hệ và ngược lại. Các phép tính quan hệ thường được sử dụng trong mô hình dữ liệu logic, trong đó các quan hệ được biểu diễn bằng cách sử dụng các vị từ cấp một.

Lưu ý: Các phép toán đại số quan hệ hay phép tính quan hệ là cơ sở toán học cho ngôn ngữ truy vấn dữ liệu SQL cũng như các vấn đề khác có liên quan như tối ưu hóa câu truy vấn …

Trong phần này chúng ta sẽ tìm hiểu về các phép toán đại số quan hệ.

Có 2 loại xử lý dữ liệu:

+ Làm thay đổi dữ liệu (cập nhật) như thêm mới, xóa và sửa.

+ Không làm thay đổi dữ liệu (rút trích), đó là các thao tác truy vấn (query).

Cách xử lý theo phương pháp đại số quan hệ nhằm biểu diễn câu truy vấn dữ liệu dưới dạng biểu thức, trong khi phương pháp biểu diễn logic sử dụng phép tính quan hệ (Relational Calculus) nhằm biểu diễn kết quả của câu truy vấn.

3.4.1. Các phép toán trên tập hợp

Xem quan hệ là tập hợp các bộ, ta có các phép toán trên tập hợp:

- Phép hội $R \cup S$
- Phép giao $R \cap S$
- Phép trừ $R - S$

Để định nghĩa các phép toán này trước hết ta xét khái niệm sau:

Định nghĩa 3.2. Tính khả hợp (Union Compatibility)

Hai lược đồ quan hệ $R(A_1, A_2, \ldots, A_n)$ và $S(B_1, B_2, \ldots, B_n)$ là khả hợp nếu:

- Cùng bậc n (cùng có n thuộc tính)
- $DOM(Ai)=DOM(Bi)$, $1 \leq i \leq n$

Kết quả của $\cup$, $\cap$, và – là một quan hệ có cùng tên thuộc tính với quan hệ đầu tiên R. Dưới đây ta xét 2 quan hệ R và S khả hợp.

3.4.1.1. Phép hợp

Ký hiệu: $R \cup S$

$R \cup S$ 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: Cho quan hệ r =

A	B	C
a1	b1	c1
a2	b1	c2
a2	b2	c1

và quan hệ s =

A	B	C
a1	b1	c1
a2	b2	c2

Khi đó, ta có quan hệ $r \cup s$ =

A	B	C
a1	b1	c1
a2	b1	c2
a2	b2	c1
a2	b2	c2

3.4.1.2. Phép giao

Ký hiệu: $R \cap S$

$R \cap S$ 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: Với r và s là 2 quan hệ như ở ví dụ phần trên, thì :

$r \cap s$ =

A	B	C
a1	b1	c1

3.4.1.3. Phép hiệu

Ký hiệu: R – S

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: Với r và s trong ví dụ phần trên thì:

r - s =	A	B	C
	a2	b1	c2
	a2	b2	c1

Một số tính chất của các phép toán tập hợp

Giao hoán

$$R \cup S = S \cup R$$

$$R \cap S = S \cap R$$

Kết hợp

$$R \cup (S \cup T) = (R \cup S) \cup T$$

$$R \cap (S \cap T) = (R \cap S) \cap T$$

3.4.2. Các phép toán nhằm rút trích một phần của quan hệ

Gồm:

Phép chọn σ (selection)

Phép chiếu π (projection)

3.4.2.1. Phép chọn

Được dùng để lấy ra các bộ của quan hệ R. Các bộ được chọn phải thỏa mãn điều kiện chọn F. Ký hiệu: $\sigma_F(R)$

$$\sigma_F(r) = \{t / t \in r \text{ và } F(t) = True\}$$

F 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>

Các <phép so sánh> gồm $<$, $>$, $\leq$, $\geq$, $\neq$, $=$

Các mệnh đề được nối lại nhờ các phép $\wedge$, $\vee$, $\neg$

Kết quả trả về của phép toán là một quan hệ mới thỏa:

- Có cùng danh sách thuộc tính với R
- Có số bộ luôn ít hơn hoặc bằng số bộ của R.

Ví dụ: r =

A	B	C	D
a1	b1	1	2
a1	b1	2	4
a3	b3	5	7
a2	b1	6	4

Với F = (D<5).

Thì σ_F (r) =

A	B	C	D
a1	b1	1	2
a1	b1	2	4
a2	b1	6	4

3.4.2.2. Phép chiếu

Được dùng để lấy ra một vài cột của quan hệ R.

Ký hiệu: $\pi_{A1, A2, ..., Ak}(R)$.

Với X= $A_1, A_2, \ldots, A_k$, $\prod_X(R) = \{t[X]/ t \in R\}$

Kết quả trả về của phép toán 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	D
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c2	d2
a2	b2	c3	d3

thì $\prod_{AB}(r)$ =

A	B
a1	b1
a2	b2

$\prod_{AC}(r)$ =

A	B
a1	c1
a2	c2
a2	c3

3.4.3. Các phép toán kết hợp các quan hệ

Gồm: Tích Descartes × , phép Chia và Kết nối (join).

3.4.3.1. Phép tích Descartes (Đề-các)

Cho hai lược đồ quan hệ $Q_1(A_1,A_2,..,A_n)$ và $Q_2(B_1,B_2,..,B_m)$. Giả sử r_1 , r_2 là hai quan hệ trên Q_1, Q_2 tương ứng. Tích Descartes của r_1 và r_2, ký hiệu là: $r_1 \times r_2$ là quan hệ trên lược đồ quan hệ có tập thuộc tính $Q = Q_1 \cup Q_2$.

$$r_1 \times r_2 = \{(t_1,t_2): t_1 \in r_1, t_2 \in r_2\}$$

Ví dụ. Cho r_1 và r_2

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

$r_1 \times r_2$

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

3.4.3.2. Phép chia 2 quan hệ

Cho R(Z) và S(X), Z là tập thuộc tính của R, X là tập thuộc tính của S, $X \subseteq Z$. Phép chia R cho S được ký hiệu: $R \div S$.

$$\mathbf{T = R \div S = \{t/ \forall\ u \in S, (t,u) \in R\}}$$

Kết quả của phép chia là một quan hệ T(Y) thỏa:

- Với Y=Z-X
- Có t là một bộ của T nếu với mọi bộ t[S]∈S, tồn tại bộ t[R]∈R thỏa 2 điều kiện:

- $t_R(Y) = t$
- $t_R(X) = t_S(X)$

Ví dụ

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

3.4.3.3. Phép kết nối

Được dùng để kết hợp các bộ của các quan hệ lại với nhau. Ký hiệu $\bowtie$

Cho $R(A_1, A_2, \ldots, A_n)$ và $(B_1, B_2, \ldots, B_m)$.

Kết quả của phép kết là một quan hệ Q có n + m thuộc tính $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$. Mỗi bộ của Q là tổ hợp của 2 bộ trong R và S, thỏa mãn một số điều kiện kết nào đó có dạng $A_i \theta B_j$, với:

A_i là thuộc tính của R, B_j là thuộc tính của S

A_i và B_j có cùng miền giá trị

θ là một trong các phép so sánh $\neq, =, <, >, \leq, \geq$

Có thể phân loại các phép kết như sau:

Kết theta (theta join) là phép kết có điều kiện. Ký hiệu $R \bowtie_C S$. C gọi là điều kiện kết trên thuộc tính.

Kết bằng (equi join) khi C là điều kiện so sánh bằng.

Kết tự nhiên (natural join). Ký hiệu $R \bowtie S$ hay $R * S$.

Kết quả của phép kết bằng hay kết nối tự nhiên bỏ bớt đi 1 cột với các cột giống nhau.

Ví dụ

Cho hai quan hệ r_1 và r_2:

r_1

A	B	C
6	5	4
7	5	5
4	2	6

r_2

B	C	H
5	4	9
4	6	8
7	5	3

Với điều kiện kết nối C = "B>=H". Ta được kết quả $r_1 \bowtie_C r_2$ là quan hệ sau:

A	B	C	B	C	H
6	5	4	7	5	3
6	5	5	7	5	3

Còn $r_1 * r_2$ là quan hệ sau:

A	B	C	H
6	5	4	9

3.4.4. Một số phép toán khác

3.4.4.1. Phép gán tên

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 $\leftarrow$.

Ví dụ: $S \leftarrow \sigma_P(R)$. Gán kết quả của phép toán $\sigma_P(R)$ thành quan hệ S.

3.4.4.2. Phép gom nhóm

Được dùng để phân chia quan hệ thành nhiều nhóm dựa trên điều kiện gom nhóm nào đó. Ký hiệu: $(G_1, G_2, \ldots, G_n)\, \Im_{F1(A1), F2(A2), \ldots, Fn(An)}(E)$

- E là một biểu thức ĐSQH, thường là một quan hệ.
- $G_1, G_2, \ldots, G_n$ là các thuộc tính gom nhóm
- $F_1, F_2, \ldots, F_n$ là các hàm kết hợp, chẳn hạn AVG, MIN, MAX, SUM, COUNT.

- $A_1, A_2, \ldots, A_n$ là các thuộc tính tính toán trong các hàm F_i (i=1,2,…,n).

Ví dụ: Khi tính số lượng nhân viên và lương trung bình của từng phòng ban từ bảng LUONG(MSNV, TENNV, LUONG, MAPHONGBAN). Ta dùng phép toán sau: (MAPHONGBAN)$\Im_{COUNT(MANV), SUM(LUONG)}$(LUONG).

3.4.4.3. Phép kết nối ngoài

Phép kết nối ngoài nhằm mở rộng phép kết để tránh mất mát thông tin. Phép toán này bao gồm các bước:

- Thực hiện phép kết
- Lấy thêm các bộ không thỏa điều kiện kết

Có 3 hình thức

Mở rộng bên trái $\supset\!\bowtie$

Mở rộng bên phải $\bowtie\!\subset$

Mở rộng 2 bên $\supset\!\bowtie\!\subset$

Ví dụ: Với 2 quan hệ r_1, r_2 cho ở phần phép kết nối, ta có: $r_1 \bowtie\!\subset r_2$ là quan hệ sau:

A	B	C	H
6	5	4	9
7	5	5	Null
4	2	6	Null

3.4.4.4. Phép nửa nối

Cho hai quan hệ r xây dựng trên $R(U_1)$và s trên $S(U_2)$. Phép nửa nối của r và s, ký hiệu là r $\ltimes$ s là một quan hệ trên lược đồ R_1 gồm các bộ của $r * s$ chiếu lên R, tức là:

$$r \ltimes s = \{t: t \in r * s . U_1\}$$

Ví dụ: Giả sử r và s là các quan hệ sau:

r =

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

s =

B	C	D
b	c	d
b	c	e
a	d	b

Khi đó ta có

$r \ltimes s$	A	B	C
	a	b	c
	d	b	c
	c	a	d

Tương tự có thể định nghĩa $r \rtimes s$ như là $r*s$ rồi lấy chiếu trên S

Ta có thể chứng minh hoặc đưa ra các ví dụ để kiểm chứng:

$$r \ltimes s = r * (s.U_1 \cap U_2)$$

$$r \ltimes s \neq r \rtimes s$$

Phép nửa nối rất có ý nghĩa khi tối ưu hoá các biểu thức quan hệ tương ứng với các yêu cầu vấn tin.

Ví dụ: Hãy cho một ví dụ trong thực tiễn có sử dụng đến phép nửa nối.

3.5. Tính đầy đủ của các phép toán

Tập các phép toán σ, π, $\times$, $-$, $\cup$ được gọi là tập đầy đủ các phép toán ĐSQH. Các phép toán khác có thể được biểu diễn qua chúng:

Ví dụ 1:

$$R \cap S = R \cup S - ((R - S) \cup (S - R))$$

$$R \bowtie_C S = \sigma_C(R \times S)$$

Ví dụ 2:

Cho R(Z) và S(X), Z là tập thuộc tính của R, Y là tập thuộc tính của S, $X \subseteq Z$, $Y = Z \backslash X$. Phép chia R cho S, $T = R \div S$ được biểu diễn phép chia thông qua tập đầy đủ các phép toán ĐSQH như sau:

$$Q_1 \leftarrow \pi_Y(R)$$

$$Q_2 \leftarrow Q_1 \times S$$

$$Q_3 \leftarrow \pi_Y(Q_2 - R)$$

$$T \leftarrow Q_1 - Q_3$$

3.6. Đại số quan hệ như là ngôn ngữ hỏi

Cho CSDL về việc cung ứng hàng hoá của các hãng, gồm ba lược đồ quan hệ sau:

S(S#, SNAME, STATUS, CITY): các hãng cung ứng hàng hoá

P(P#, PNAME, COLOR, WEIGHT, CITY): các mặt hàng

SP(S#, P#,QTY): Các mặt hàng đã cung cấp, ở đây

QTY: số lượng đã cung cấp. S#, P# lần lượt là mã của hãng cung ứng có tên là SNAME, mã của mặt hàng có tên PNAME.

- Tìm mã số của những hãng đã cung ứng mặt hàng P2

$$\Pi_{S\#}\,(\sigma_{P\#\,=\,'P2'}(SP))$$

- Tìm mã số của những hãng cung ứng ít nhất một mặt hàng màu đỏ.

$$\Pi_{S\#}\,(\sigma_{color\,=\,'red'}(P*SP))$$

hoặc

$$\Pi_{S\#}\,(\sigma_{color\,=\,'red'}(P))*(SP)$$

Một câu hỏi đặt ra là: *Các phép toán đại số quan hệ được dùng ở đâu, khi nào và cho ai trong quá trình thiết kế cơ sở dữ liệu và xây dựng các chương trình ứng dụng*?

Tóm tắt chương 3

* Nhìn nhận quan hệ trên quan điểm lý thuyết tập hợp.

* Lược đồ quan hệ, Ràng buộc toàn vẹn, Siêu khóa (Super key), Khóa, Tham chiếu và khái niệm khóa ngoại (Foreign key).

* Các phép toán trên mô hình dữ liệu quan hệ.

* Tập các phép toán σ, π, $\times$, $-$, $\cup$ được gọi là tập đầy đủ các phép toán ĐSQH. Các phép toán khác có thể được biểu diễn qua chúng.

Câu hỏi ôn tập chương 3

1) Định nghĩa các thuật ngữ sau: miền, thuộc tính, n-bộ, lược đồ quan hệ, trạng thái quan hệ, cấp của quan hệ, lược đồ cơ sở dữ liệu, trạng thái cơ sở dữ liệu.

2) Vì sao các bộ trong một quan hệ là không có thứ tự?

3) Vì sao không cho phép các bộ trùng lặp trong một quan hệ?

4) Siêu khóa và khóa khác nhau ở chỗ nào?

5) Vì sao phải chỉ định một trong các khóa dự tuyển làm khóa chính?

6) Nêu những đặc trưng làm cho các quan hệ khác với các bảng hoặc các tệp thông thường?

7) Nêu các lý do về việc tồn tại các giá trị không xác định trong các quan hệ?

8) Hãy giải thích về ràng buộc toàn vẹn thực thể và ràng buộc toàn vẹn tham chiếu. Vì sao các ràng buộc này là quan trọng?

9) Định nghĩa khóa ngoài. Khái niệm này dùng để làm gì? Các khóa ngoài đóng vai trò như thế nào trong phép nối?

10) Hãy giải thích các phép toán cập nhật trên các quan hệ và các kiểu ràng buộc toàn vẹn phải được kiểm tra đối với mỗi phép toán cập nhật.

11) Liệt kê các phép toán đại số quan hệ và mục đích của từng phép toán.

12) Khả hợp là gì? Vì sao các phép toán hợp, giao, trừ đòi hỏi các quan hệ tham gia vào phép toán phải khả hợp?

13) Hãy giải thích các kiểu truy vấn cần có việc đặt lại tên các thuộc tính để chỉ ra truy vấn một cách rõ ràng.

14) Hãy nêu các kiểu phép toán nối khác nhau.

15) Phép toán hàm là gì? Nó được dùng vì mục đích nào?

16) Các phép nối ngoài khác với các phép nối trong như thế nào? Phép hợp ngoài khác với phép hợp như thế nào?

Bài tập chương 3

1. Phân tích và trả lời các câu hỏi sau

- Phân biệt các thuật ngữ: thuộc tính – miền; lược đồ quan hệ - quan hệ; siêu khóa – khóa; toàn vẹn thực thể - toàn vẹn tham chiếu; khóa chính – khóa dự tuyển.

- Vì sao trong một quan hệ không có hai bộ giống hệt nhau? Trong một quan hệ giá trị khóa của một bộ là khác Null.

- Cho ví dụ minh họa ý nghĩa của khóa ngoại.

- Cho biết lý do có thể dẫn đến giá trị Null trong quan hệ.

- Cho ví dụ về việc cập nhật dữ liệu có thể gây ra sự vi phạm toàn vẹn thực thể và toàn vẹn tham chiếu.

- Chúng ta có suy nghĩ gì về tính đúng đắn của các qui tắc chuyển đổi trên và thế nào là chuyển đổi từ sơ đồ ERD sang lược đồ quan hệ một cách "hợp lý"?

2. Trong một hệ QTCSDL cụ thể các phép toán đại số quan hệ sau ứng với những lệnh nào?

- Phép chiếu, phép chọn, phép kết nối (trong trường hợp này biểu thức kết nối ứng với biểu thức gì trong lệnh của hệ QTCSDL đó), tích Đêcác, phép hợp, phép hiệu. Rút ra ý nghĩa thực tiễn của các phép toán đại số quan hệ. Tiến hành cài đặt các thủ tục thực hiện các kết quả của các phép toán trên.

3. Chuyển lược đồ trong các bài tập ở chương 2 sang mô hình dữ liệu quan hệ, lưu ý có tính đến hiệu suất truy vấn.

4. Cho các bảng sau:

Student (sname, address, gender, birthyear) – thông tin về sinh viên

Course (cname, dname) – thông tin về khóa học

Result (sname, cname, grade) – thông tin sinh viên đã tham gia các khóa học nào và đạt điểm số là bao nhiêu

Professor (pname, dname, address, gender, birthday) – thông tin về giáo viên.

Sinh viên

Tên	*Địa chỉ*	*Giới tính*	*Năm sinh*
Le Na	Hue	F	1968
Phan Ngoc	Quang Tri	M	1990

Kết quả

Tên	*Lớp*	*Điểm*
Le Na	Toan 1	9
Le Na	Anh 2	8

Tran Son	Hue	M	1992	Le Na	Tin 2	5
Bui Thai	Hue	M	1989	Ha Anh	Toan 2	6
Khanh Ly	Da Nang	F	1990	Ha Anh	Tin 1	8
Ha Anh	Hue	F	1985	Ha Anh	Gtich	4
Ngo Minh	Quang Binh	M	1981	Le Anh	Toan 1	3
Thai Hoa	Vinh	F	1990	Le Anh	Tin 2	5
Le Anh	Hue	M	1985	Le Anh	Anh 2	2
Hoang Anh	Thanh Hoa	F	1982	Thai Hoa	Anh 3	4
				Thai Hoa	Toan 1	7
				Tran Son	Anh 2	8
				Tran Son	Đại số	8

Giảng viên

Họ tên	*Môn học*	*Địa chỉ*	*Giới tính*	*Ngày sinh*
Ha Chau	Tin	Hue	F	1/1/1972
Hoang Nam	Tin	Q. Tri	M	2/2/1973
Tran Son	Tin	Vinh	M	3/3/1974
Vu Nhung	Anh	Hue	F	4/4/1965
Hoang Hoa	Anh	Q. Tri	F	5/5/1972
Le Dung	Toan	Vinh	F	6/6/1964
Nguyen Dat	Tin	D. Nang	M	4/8/1973

Môn học

Lớp	*Môn học*
Toan 1	Tin
Toan 2	Tin
Tin 1	Tin
Tin 2	Tin
Anh 1	Anh
Anh 2	Anh
Anh 3	Anh
G Tích	Toan

a. Hiển thị Tên, địa chỉ và năm sinh của các nữ sinh viên.
b. Hiển thị Tên, địa chỉ của những sinh viên nữ sinh năm 1990.
c. Hiển thị tên những môn do Khoa Tin tổ chức.
d. Hiển thị Tên môn, Tên khoa các môn do Khoa Tin hoặc Khoa Anh tổ chức.
e. Tìm những sinh viên (Tên sv) nhà ở Huế và sinh sau năm 1985.
f. Tìm tên và địa chỉ của các giáo viên khoa Tin và Anh.

g. Hiện kết quả học tập của các sinh viên có điểm Toán 1 >= 8 hoặc môn Đại số >= 8.
h. Tìm những sinh viên có điểm Toán 1 = 9 và điểm Anh 2 = 8 (sử dụng phép chia).
i. Hiện thông tin cá nhân và kết quả học tập của các sinh viên có đăng ký học.
j. Hiện thông tin cá nhân của các sinh viên không đăng ký học.
k. Hiện tên sinh viên, tên môn, điểm và tên khoa của các sinh viên có học các môn do khoa Tin tổ chức.
l. Hiện tên, địa chỉ của những sinh viên có đăng ký học nhưng không học Giải tích và Toán 1.
m. Hiện tên của những sinh viên có cùng địa chỉ với cô Ha Chau và có điểm Toán 1 từ 8 trở lên.
n. Tìm địa chỉ của các sinh viên nữ, sinh năm 1980 hoặc sau đó, có điểm của môn học do khoa Tin tổ chức >= 8.
o. Hiện các sinh viên và giáo viên có cùng địa chỉ theo mẫu sau: tên sinh viên, địa chỉ sinh viên, giới tính sinh viên, tên giảng viên, địa chỉ giảng viên, giới tính giảng viên (sử dụng phép nối).
p. Hiện các sinh viên và giáo viên có cùng địa chỉ theo mẫu sau: tên, địa chỉ, giới tính (sử dụng phép hợp).
q. Hiện tên sinh viên có học các môn mà sinh viên Khanh Ly học (sử dụng phép chia).
r. Hiện tên của sinh viên có điểm Toán 1 và Anh 2 >= 8.
s. Hiện ra tên của các sinh viên có cùng năm sinh.
t. Cho biết thông tin về giáo viên Chuyên ngành Tin mà chưa tham gia giảng dạy.
u. Cho biết Mã sinh viên, Tên sinh viên, Mã môn học và Điểm tất cả các môn của những sinh viên mà có điểm Toán 1 từ 5 trở lên.
v. Xem danh sách môn học thuộc chuyên ngành Tin mà chưa có sinh viên học.

5. Cho lược đồ quan hệ cơ sở dữ liệu gồm thông tin về các nhà cung cấp (S – Supplies), thông tin về các mặt hàng (P – Products) và thông tin về số

lượng các mặt hàng đã được cung ứng bởi các nhà cung cấp nào (SP – Supply Product).

S :

S#	Sname	City	Status
S1	Smith	Paris	20
S2	Jones	London	10
S3	Blake	London	30
S4	Clark	Paris	20
S5	Adams	Athen	30

P :

P#	Pname	Color	Weight	Stock
P1	Nut	Red	17	London
P2	Bolt	Green	12	Paris
P3	Screw	Blue	13	Rome
P4	Screw	Red	17	London
P5	Cam	Blue	12	Paris
P6	Cog	Red	19	London

SP :

S#	P#	Amount
S1	P1	200
S1	P2	300
S2	P3	400
S3	P3	200
S3	P4	500
S4	P6	300
S5	P2	200
S5	P3	250

a. Cho biết số hiệu, tên và tình trạng sản xuất (status) của tất cả các nhà cung cấp ở Paris.

b. Hiển thị mã số và tên của các sản phẩm có số lượng từ 10 đến 15.

c. Hiển thị tên và thành phố của các nhà cung cấp đã không cung ứng sản phẩm có mã P3.

d. Cho biết các nhà cung cấp có trụ sở tại cùng thành phố.

e. Tìm mã số nhà cung cấp mà phân phối ít nhất 250 sản phẩm có màu xanh được lưu kho tại Paris.

f. Những nhà cung cấp mà chưa cung ứng sản phẩm nào có tên là gì?

g. Xem thông tin của các sản phẩm có màu xanh Blue hoặc có trọng lượng không quá 15 (sử dụng phép hợp).

h. Xem số hiệu và tên những nhà cung cấp đóng trụ sở tại London hoặc đã cung ứng sản phẩm có tên Crew.

i. Xem tên những nhà cung cấp mà không cung ứng những mặt hàng do S1 cung ứng (sử dụng phép chia).

j. Hiển thị những sản phẩm màu đỏ mà chưa được nhà sản xuất nào cung ứng.

6. Với các lược đồ trong bài 5, xác định số bộ trả về (yêu cầu liệt kê chi tiết) khi thực hiện các câu đại số quan hệ sau:

a. $\Pi_{Sname,\ City}(\sigma_{Status=20}(S))$

b. $\Pi_{S\#,\ Sname}(\sigma_{Amount<270}\ (S \bowtie_{S.S\#=SP.S\#} SP))$

c. $\Pi_{P\#,\ Pname}(\sigma_{Amount=300\ V\ Color='Green'}(P \bowtie_{P.P\#=SP.P\#} SP))$

d. $\Pi_{P\#,\ Pname,\ Amount}(\sigma_{Color='Red' \wedge Amount>=300}(P \bowtie_{P.P\#=SP.P\#} SP))$

e. $\Pi_{P\#,\ Pname,\ Amount}[(\sigma_{Color='Red'}(P)) \bowtie_{P.P\#=SP.P\#} (\sigma_{Amount>=300}(SP))]$

f. $\Pi_{Pname,\ Stock}[\ \Pi_{S\#}(\sigma_{City='London'}(S)) \bowtie_{S.S\#=SP.S\#} (P \bowtie_{P.P\#=SP.P\#} SP)\ ]$

g. $(\Pi_{S\#}(S) - \Pi_{S\#}(SP)) \rtimes_{S.S\#=SP.S\#} S$

h. $\Pi_{P\#,\ Pname}(\sigma_{Amount=300}(P \bowtie_{P.P\#=P.P\#} SP)) \cup \Pi_{P\#,\ Pname}(\sigma_{Color='Green'}(P \bowtie_{P.P\#=P.P\#} SP))$

i. $S \ltimes [\ \Pi_{S\#}(\sigma_{Color='Red'}(P) \bowtie_{P.P\#=SP.P\#} SP)) \cap \Pi_{S\#}(\sigma_{Color='Blue'}(P) \bowtie_{P.P\#=P.P\#} SP))\ ]$

j. $\Pi_{S\#,\ Sname,\ P\#,\ Pname}(S) \bowtie_{S.City=P.Stock} (\Pi_{P.Stock}(P))$

k. $\rho[A, \Pi_{P\#}(\sigma_{Color='Blue'}(P))\ ]$

$\Pi_{S\#,\ Sname}(S) \bowtie_{S.S\#=B.S\#} \rho(B, \Pi_{S\#}(SP \div A))$

CHƯƠNG 4. NGÔN NGỮ CƠ SỞ DỮ LIỆU

Mục đích

- Trình bày ngôn ngữ cơ sở dữ liệu SQL, các thành phần cơ bản của của nó.

Yêu cầu

- Vận dụng được quá trình chuyển từ câu truy vấn trong ngôn ngữ tự nhiên sang ngôn ngữ SQL và ngược lại.

- Nắm vững ngôn ngữ thao tác và định nghĩa dữ liệu và khai báo một số ràng buộc toàn vẹn cơ bản trên SQL và ngôn ngữ điều khiển dữ liệu của SQL.

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 định nghĩa 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 điều khiển 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.

Ở đây chúng tôi chỉ trình bày cú pháp chung của SQL.

Một số phần trong chương này khi biên soạn được chúng tôi trích dẫn từ tài liệu [7].

4.1. Sơ lược về ngôn ngữ SQL

SQL, viết tắt của *Structured Query Language* (ngôn ngữ hỏi có cấu trúc) là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sử dụng để tương tác với cơ sở dữ liệu quan hệ.

SQL được sử dụng để thực hiện các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

- **Định nghĩa dữ liệu:** SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.
- **Truy xuất và thao tác dữ liệu:** Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.
- **Điều khiển truy cập:** SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu
- **Đảm bảo toàn vẹn dữ liệu:** SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập nhật cũng như các lỗi của hệ thống.

Mặc dù SQL không phải là một ngôn ngữ lập trình như C, C^{++}, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu.

Khác với các ngôn ngữ lập trình quen thuộc như C, C^{++}, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng.

Bản thân SQL không phải là một hệ quản trị cơ sở dữ liệu, nó không thể tồn tại độc lập. SQL thực sự là một phần của hệ quản trị cơ sở dữ liệu, nó xuất hiện trong các hệ quản trị cơ sở dữ liệu với vai trò ngôn ngữ và là công cụ giao tiếp giữa người sử dụng và hệ quản trị cơ sở dữ liệu.

Trong hầu hết các hệ quản trị cơ sở dữ liệu quan hệ, SQL có những vai trò như sau:

- **SQL là ngôn ngữ hỏi có tính tương tác:** Người sử dụng có thể dễ dàng thông qua các trình tiện ích để gởi các yêu cầu dưới dạng các câu lệnh SQL đến cơ sở dữ liệu và nhận kết quả trả về từ cơ sở dữ liệu
- **SQL là ngôn ngữ lập trình cơ sở dữ liệu**: Các lập trình viên có thể

nhúng các câu lệnh SQL vào trong các ngôn ngữ lập trình để xây dựng nên các chương trình ứng dụng giao tiếp với cơ sở dữ liệu

• **SQL là ngôn ngữ quản trị cơ sở dữ liệu:** Thông qua SQL, người quản trị cơ sở dữ liệu có thể quản lý được cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, điều khiển truy cập cơ sở dữ liệu,...

• **SQL là ngôn ngữ cho các hệ thống khách/chủ (client/server):** Trong các hệ thống cơ sở dữ liệu khách/chủ, SQL được sử dụng như là công cụ để giao tiếp giữa các trình ứng dụng phía máy khách với máy chủ cơ sở dữ liệu.

• **SQL là ngôn ngữ truy cập dữ liệu trên Internet:** Cho đến nay, hầu hết các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ để tương tác với dữ liệu trong các cơ sở dữ liệu.

• **SQL là ngôn ngữ cơ sở dữ liệu phân tán:** Đối với các hệ quản trị cơ sở dữ liệu phân tán, mỗi một hệ thống sử dụng SQL để giao tiếp với các hệ thống khác trên mạng, gởi và nhận các yêu cầu truy xuất dữ liệu với nhau.

• **SQL là ngôn ngữ sử dụng cho các cổng giao tiếp cơ sở dữ liệu:** Trong một hệ thống mạng máy tính với nhiều hệ quản trị cơ sở dữ liệu khác nhau, SQL thường được sử dụng như là một chuẩn ngôn ngữ để giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

Trong chương này các thuật ngữ trong CSDL quan hệ như quan hệ, thuộc tính, bộ được thay thế bằng các thuật ngữ như bảng, cột, bản ghi hoặc hàng tương ứng.

Các ví dụ trong chương được dựa trên cơ sở dữ liệu mẫu được mô tả dưới đây, về quản lý sinh viên và điểm thi của sinh viên trong một trường đại học.

Cơ sở dữ liệu bao gồm các bảng sau đây:

• Bảng KHOA lưu trữ dữ liệu về các khoa hiện có ở trong trường.

• Bảng LOP bao gồm dữ liệu về các lớp trong trường.

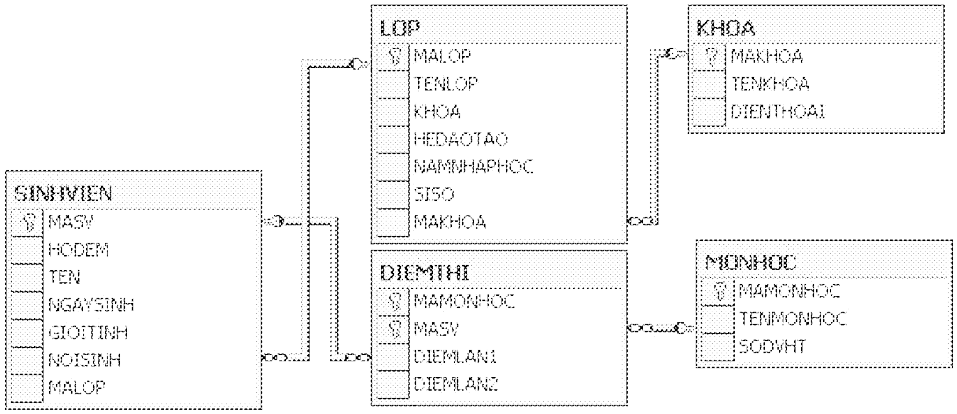
• Bảng SINHVIEN được sử dụng để lưu trữ dữ liệu về các sinh viên trong trường.

• Bảng MONHOC bao gồm các môn học (học phần) được giảng dạy

trong trường

• Bảng DIEMTHI với dữ liệu cho biết điểm thi kết thúc môn học của các sinh viên.

Mối quan hệ giữa các bảng được thể hiện qua sơ đồ dưới đây



Các bảng trong cơ sở dữ liệu, mối quan hệ giữa chúng và một số ràng buộc được cài đặt như sau:

```
CREATE TABLE khoa
(
makhoa    NVARCHAR(5)   NOT NULL
   CONSTRAINT pk_khoa PRIMARY KEY,
tenkhoa    NVARCHAR(50)   NOT NULL,
dienthoai  NVARCHAR(15)    NULL
)

CREATE TABLE lop
(
malop     NVARCHAR(10) NOT NULL
    CONSTRAINT pk_lop  PRIMARY KEY,
tenlop     NVARCHAR(30) NULL ,
```

```
khoa    SMALLINT  NULL ,
hedaotao   NVARCHAR(25) NULL ,
namnhaphoc   INT   NULL ,
siso    INT   NULL ,
makhoa    NVARCHAR(5) NULL
)
CREATE TABLE sinhvien
(
masv    NVARCHAR(10) NOT NULL
   CONSTRAINT pk_sinhvien PRIMARY KEY,
hodem    NVARCHAR(25) NOT NULL ,
ten    NVARCHAR(10) NOT NULL ,
ngaysinh   SMALLDATETIME NULL ,
gioitinh   BIT   NULL ,
noisinh    NVARCHAR(100) NULL ,
malop    NVARCHAR(10) NULL
)
CREATE TABLE  monhoc
(
mamonhoc   NVARCHAR(10) NOT NULL
   CONSTRAINT pk_monhoc PRIMARY KEY,
tenmonhoc   NVARCHAR(50) NOT NULL ,
sodvtc    SMALLINT  NOT NULL
)
CREATE TABLE diemthi
(
mamonhoc  NVARCHAR(10) NOT NULL ,
```

```
 masv    NVARCHAR(10) NOT NULL ,
 diemlan1   NUMERIC(5, 2) NULL ,
 diemlan2   NUMERIC(5, 2) NULL,
CONSTRAINT pk_diemthi PRIMARYKEY(mamonhoc,masv)
 )

 ALTER TABLE  lop
 ADD
  CONSTRAINT fk_lop_khoa
  FOREIGN KEY(makhoa)
  REFERENCES khoa(makhoa)
  ON DELETE CASCADE
  ON UPDATE CASCADE

 ALTER TABLE sinhvien
 ADD
  CONSTRAINT fk_sinhvien_lop
  FOREIGN KEY (malop)
  REFERENCES lop(malop)
  ON DELETE CASCADE
  ON UPDATE CASCADE

 ALTER TABLE diemthi
 ADD
  CONSTRAINT fk_diemthi_monhoc
  FOREIGN KEY (mamonhoc)
  REFERENCES monhoc(mamonhoc)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE,
    CONSTRAINT fk_diemthi_sinhvien
    FOREIGN KEY (masv)
    REFERENCES sinhvien(masv)
    ON DELETE CASCADE
    ON UPDATE CASCADE

  ALTER TABLE monhoc
  ADD
    CONSTRAINT chk_monhoc_sodht
    CHECK(sodvtc>0 and sodvtc<=5)

  ALTER TABLE diemthi
  ADD
    CONSTRAINT chk_diemthi_diemlan1
    CHECK (diemlan1>=0 and diemlan1<=10),
    CONSTRAINT chk_diemthi_diemlan2
CHECK (diemlan2>=0 and diemlan2<=10)
```

4.2. Ngôn ngữ thao tác dữ liệu

Trong phần này, trình bày nhóm các câu lệnh trong SQL được sử dụng cho mục đích truy vấn và thao tác trên dữ liệu. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language) bao gồm các câu lệnh sau:

- SELECT: Sử dụng để truy xuất dữ liệu từ môt hoặc nhiều bảng.
- INSERT: Bổ sung dữ liệu.
- UPDATE: Cập nhật dữ liệu

• DELETE: Xoá dữ liệu

Trong các câu lệnh này, SELECT là câu lệnh tương đối phức tạp và được sử dụng nhiều trong cơ sở dữ liệu. Với câu lệnh này, ta không chỉ thực hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thực hiện được các yêu cầu thống kê dữ liệu phức tạp. Cũng chính vì vậy, phần đầu của chương này sẽ tập trung tương đối nhiều đến câu lệnh SELECT. Các câu lệnh INSERT, UPDATE và DELETE được bàn luận đến ở cuối mục.

4.2.1. Truy xuất dữ liệu với câu lệnh SELECT

Trong SQL các vấn đề khai thác CSDL được chuyển thành các câu hỏi để truy vấn thông tin từ một cơ sở dữ liệu gồm nhiều bảng khác nhau. Kết quả của một câu truy vấn là một bảng mới. Bảng này có các cột được lấy từ các cột hay kết quả của các phép tính trên các cột của cơ sở dữ liệu cần khai thác và các hàng cũng chính là các hàng thích hợp được chọn ra từ cơ sở dữ liệu này. Bảng được kết xuất ra màn hình, máy in hay một file mới.

Câu truy vấn trong SQL là phép ánh xạ được miêu tả như một khối SELECT - FROM – WHERE.

Cú pháp chung của câu lệnh SELECT có dạng:

```
SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn
[INTO tên_bảng_mới]
FROM  danh_sách_bảng/khung_nhìn
[WHERE     điều_kiện]
[GROUP BY  danh_sách_cột]
[HAVING điều_kiện]
[ORDER BY  cột_sắp_xếp]
[COMPUTE  danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Trong đó:

`Select_list` là danh sách các cột cần truy vấn.

`New_table`: tên bảng chứa kết quả truy vấn cần lưu.

`Table-name|VIEW_name`: tên bảng chứa thông tin cần truy vấn.

Chú ý: Các thành phần trong một câu lệnh SELECT phải được sử dụng theo thứ tự nêu trên.

Khối SELECT gồm ba mệnh đề chính

SELECT: Xác định nội dung các cột cần đưa ra kết quả.

FROM: Chỉ ra các bảng mà ta lấy thông tin trên đó.

WHERE: Xác định các bảng ghi thoả yêu cầu chọn lọc để đưa ra kết quả.

Câu lệnh SELECT được sử dụng để tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng được hiển thị dưới dạng bảng, tức là một tập hợp các dòng và các cột (ngoại trừ trường hợp sử dụng câu lệnh SELECT với mệnh đề COMPUTE).

Dưới đây lần lượt giải thích rõ hơn về khối SELECT để khai thác thông tin từ dữ liệu.

4.2.1.1. Từ khoá ALL|DISTINCT

Từ khoá DISTINCT được sử dụng nhằm loại bỏ, chỉ giữa lấy một dòng đại diện nếu có nhiều dòng giống nhau.

Từ khoá ALL (có thể thay thế bằng dấu *) được sử dụng nhằm lấy tất cả các hàng được truy xuất (mặc định)

4.2.1.2. Mệnh đề FROM

Mệnh đề FROM trong câu lệnh SELECT được sử dung nhằm chỉ định các bảng và khung nhìn cần truy xuất dữ liệu. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn, tên của các bảng và khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ 4.2.1: Kết quả của câu lệnh sau đây cho biết mã lớp, tên lớp và hệ đào tạo của các lớp hiện có.

```
SELECT malop,tenlop,hedaotao FROM lop
```

MALOP	TENLOP	HEDAOTAO
C24101	Toán K24	Chính quy
C24102	Tin K24	Chính quy
C24103	Lý K24	Chính quy
C24301	Sinh K24	Chính quy
C25101	Toán K25	Chính quy
C25102	Tin K25	Chính quy
C25103	Lý K25	Chính quy
C25301	Sinh K25	Chính quy
C26101	Toán K26	Chính quy
C26102	Tin K26	Chính quy

4.2.1.2 Danh sách chọn trong câu lệnh SELECT

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức được chỉ định ngay sau từ khoá SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

a. Chọn tất cả các cột trong bảng

Khi cần hiển thị tất cả các trường trong các bảng, ta sử dụng ký tự * trong danh sách chọn thay vì phải liệt kê danh sách tất cả các cột. Trong trường hợp này, các cột được hiển thị trong kết quả truy vấn sẽ tuân theo thứ tự mà chúng đã được tạo ra khi bảng được định nghĩa.

Ví dụ 4.2.2: Câu lệnh

```
SELECT * FROM lop
```

cho kết quả như sau:

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHTO1
C24102	Tin K24	24	Chính quy	2000	DHTO2
C24103	Lý K24	24	Chính quy	2000	DHTO3
C24301	Sinh K24	24	Chính quy	2000	DHTO5
C25101	Toán K25	25	Chính quy	2001	DHTO1
C25102	Tin K25	25	Chính quy	2001	DHTO2
C25103	Lý K25	25	Chính quy	2001	DHTO3
C25301	Sinh K25	25	Chính quy	2001	DHTO5
C26101	Toán K26	26	Chính quy	2002	DHTO1
C26102	Tin K26	26	Chính quy	2002	DHTO2

b. 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 trường trong danh sách chọn.

Ví dụ 4.2.3: Câu lệnh

```
SELECT malop,tenlop,namnhaphoc,khoa FROM lop
```

Cho biết mã lớp, tên lớp, năm nhập học và khoá của các lớp và có kết quả như sau:

MALOP	TENLOP	NAMNHAPHOC	KHOA
C24101	Toán K24	2000	24
C24102	Tin K24	2000	24
C24103	Lý K24	2000	24
C24301	Sinh K24	2000	24
C25101	Toán K25	2001	25
C25102	Tin K25	2001	25
C25103	Lý K25	2001	25
C25301	Sinh K25	2001	25
C26101	Toán K26	2002	26
C26102	Tin K26	2002	26

Lưu ý: Nếu truy vấn được thực hiện trên nhiều bảng/khung nhìn và trong các bảng/khung nhìn có các trường trùng tên thì tên của những trường này nếu xuất hiện trong danh sách chọn phải được viết dưới dạng:

```
tên_bảng.tên_trường
```

Ví dụ 4.2.4:

```
SELECT malop, tenlop, lop.makhoa, tenkhoa FROM lop,
khoa WHERE lop.malop = khoa.makhoa
```

c. Thay đổi tiêu đề các cột

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các trường tương ứng trong bảng. Tuy nhiên, để các tiêu đề trở nên thân thiện hơn, ta có thể đổi tên các tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó, ta sử dụng cách viết:

```
tiêu_đề_cột = tên_trường
```

hoặc `tên_trường AS tiêu_đề_cột`

hoặc `tên_trường  tiêu_đề_cột`

Ví dụ 4.2.5: Câu lệnh dưới đây:

```
SELECT 'Mã lớp'= malop,tenlop 'Tên lớp',khoa AS
'Khoá' FROM lop
```

Cho biết mã lớp, tên lớp và khoá học của các lớp trong trường. Kết quả của câu lệnh như sau:

Mã lớp	Tên Lớp	Khoá
C24101	Toán K24	24
C24102	Tin K24	24
C24103	Lý K24	24
C24301	Sinh K24	24
C25101	Toán K25	25
C25102	Tin K25	25
C25103	Lý K25	25
C25301	Sinh K25	25
C26101	Toán K26	26
C26102	Tin K26	26

d. Sử dụng cấu trúc CASE trong danh sách chọn

Cấu trúc CASE được sử dụng trong danh sách chọn nhằm thay đổi kết quả của truy vấn tuỳ thuộc vào các trường hợp khác nhau. Cú pháp như sau:

```
CASE biểu_thức
      WHEN biểu_thức_kiểm_tra THEN kết_quả
     [ ... ]
     [ELSE kết_quả_của_else]
  END
```

hoặc:

```
  CASE
  WHEN điều_kiện THEN kết_quả
       [ ... ]
     [ELSE kết_quả_của_else]
  END
```

Ví dụ 4.2.6: Để hiển thị mã, họ tên và giới tính (nam hoặc nữ) của các sinh viên, ta sử dụng câu lệnh.

```
SELECT masv,hodem,ten,
     CASE gioitinh
           WHEN 1 THEN 'Nam'
           ELSE 'Nữ'
         END AS gioitinh
FROM sinhvien
```

hoặc:

```
SELECT masv,hodem,ten,
     CASE
               WHEN gioitinh=1 THEN 'Nam'
           ELSE 'Nữ'
     END AS gioitinh
FROM sinhvien
```

Kết quả của hai câu lệnh trên đều có dạng như sau

MASV	HODEM	TEN	GIOITINH
0241010001	Ngô Thị Nhật	Anh	Nữ
0241010002	Nguyễn Thị Ngọc	Anh	Nữ
0241010003	Ngô Việt	Bắc	Nam
0241010004	Nguyễn Đình	Bình	Nam
0241010005	Hồ Đăng	Chiến	Nam
0241020001	Nguyễn Tuấn	Anh	Nam
0241020002	Trần Thị Kim	Anh	Nữ
0241020003	Võ Đức	Ấn	Nam
0241020004	Nguyễn Công	Bình	Nam
0241020005	Nguyễn Thanh	Bình	Nam
0241020006	Lê Thị Thanh	Châu	Nữ
0241020007	Bùi Đình	Chiến	Nam
0241020008	Nguyễn Công	Chính	Nam
...	...	...	...

e. Hằng và biểu thức trong danh sách chọn

Ngoài danh sách trường, trong danh sách chọn của câu lệnh SELECT còn có thể sử dụng các biểu thức. Mỗi một biểu thức trong danh sách chọn trở thành một cột trong kết quả truy vấn.

Ví dụ 4.2.7. Câu lệnh dưới đây cho biết tên và số tiết của các môn học

```
SELECT tenmonhoc,sodvtc*15 AS sotiet FROM monhoc
```

TENMONHOC	SOTIET
Hoá đại cương	45
Tin học đại cương	60
Ngôn ngữ C	75
Lý thuyết hệ điều hành	60
Cấu trúc dữ liệu và ...	60
Đại số tuyến tính	60
Giải tích 1	60
Bài tập Đại số	30
Bài tập Giải tích 1	30
Vật lý đại cương	45

4.2.1.3. Chỉ định điều kiện truy vấn dữ liệu

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các

điều kiện đối với việc truy xuất dữ liệu. Biểu thức logic sau mệnh đề WHERE nhằm chỉ ra những dòng dữ liệu nào thoả mãn điều kiện mới được hiển thị trong kết quả truy vấn.

Trong các biểu thức điều kiện, SQL cho phép sử dụng các hàm mẫu MAX, MIN, AVG, SUM, COUNT, các phép toán số hoc, phép toán xử lý chuỗi, toán tử luận lý, so sánh và cả toán tử tập hợp như: UNION, INTERSET, MINUS, CONTAINS, IN và NOT IN...

Trong mệnh đề WHERE thường sử dụng:

- Các toán tử kết hợp điều kiện (AND, OR)
- Các toán tử so sánh
- Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)
- Danh sách
- Kiểm tra khuôn dạng dữ liệu.
- Các giá trị NULL

a. Toán tử số học - toán tử so sánh

Toán tử số học: cộng, trừ, nhân, chia, modulo (a%b hoặc a Mod b).

Các toán tử so sánh:

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ 2.8. câu lệnh sau cho biết tên, địa chỉ và điện thoại của những nhân

viên có hệ số lương nhỏ hơn 1.92:

```
SELECT hoten, diachi, dienthoai  FROM canbo
WHERE  hsluong <1.92
```

b. Giới hạn (BETWEEN và NOT BETWEEN)

Từ khoá BETWEEN và NOT BETWEEN được sử dụng nhằm chỉ định khoảng giá trị cần tìm kiếm đối với câu lệnh SELECT.

Ví dụ 4.2.9. Câu lệnh sau đây cho biết tên và địa chỉ những nhân viên có hệ số lương trong khoảng 1.92 đến 3.11

```
SELECT hoten, diachi  FROM canbo
WHERE hsluong BETWEEN 1.92 and 3.11
```

c. Toán tử tập hợp IN và NOT IN

Từ khoá IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu lệnh SELECT là một danh sách các giá trị. Sau IN hoặc NOT IN có thể là một danh sách các giá trị hoặc một câu lệnh SELECT khác.

Ví dụ 4.2.10. Hiển thị thông tin về các nhân viên có hệ số lương là 1.86, 1.92 hoặc 2.11:

```
SELECT * FROM canbo
WHERE hsluong IN (1.86, 1.92, 2.11)
```

IN và NOT IN rất hay sử dụng khi xem xét các câu truy vấn dưới góc độ là kết quả của các phép toán tập hợp hay đại số quan hệ.

d. Khuôn dạng (LIKE và NOT LIKE)

Từ khoá LIKE (hoặc NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng dữ liệu cần tìm kiếm. Thường kèm với các kí tự đại diện:

%: thay cho một xâu bất kỳ.

_: thay cho một kí tự bất kỳ.

[ ]: kí tự đơn bất kì trong một tập được chỉ định.

[^]: kí tự đơn bất kỳ không nằm trong tập được chỉ định.

Ví dụ 4.2.11. Câu lệnh sau hiển thị thông tin về các nhân viên có tên

HUNG.

```
SELECT * FROM canbo WHERE hoten LIKE  '%HUNG'
```

e. Các giá trị chưa biết (IS NULL và NOT IS NULL)

- Giá trị NULL có thể được nhập vào một cột cho phép chấp nhận giá trị NULL theo một trong 3 cách sau:

- Nếu không có dữ liệu được đưa vào và không có mặc định cho cột. hay kiểu dữ liệu trên cột đó.

- Người sử dụng trực tiếp đưa giá trị NULL vào cột đó.

Một cột có kiểu dữ liệu là kiểu số sẽ chứa giá trị NULL nếu giá trị chỉ định gây ra tràn số.

f. Kết hợp các điều kiện

Chúng ta sử dụng các toán tử logic: AND, OR, NOT, các phép toán logic này dùng để xây dựng các biểu thức điều kiện phức tạp hơn trong mệnh đề WHERE.

Ví dụ 4.2.12. Câu lệnh sau cho biết nhân viên nào có mã nhân viên là CB01001 và điện thoại 821034 và hệ số lương trên 1,92.

```
SELECT hoten, diachi FROM canbo
WHERE macb="CB01001" and dienthoai="821034" and hsluong>1.92
```

g. Sử dụng exits, any, all trong các truy vấn lồng

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 truy vấn dữ liệu cần phải sử dụng đến kết quả của một truy vấn khác.

Khi sử dụng truy vấn con cần lưu ý một số quy tắc sau:

• Trong hầu hết các trường hợp, một truy vấn con thường phải có kết quả là một cột (tức là chỉ có duy nhất một cột trong danh sách chọn).

• Các tên cột xuất hiện trong truy vấn con có thể là các cột của các bảng trong truy vấn ngoài.

• Một truy vấn con thường được sử dụng làm điều kiện trong mệnh đề WHERE hoặc HAVING của một truy vấn khác.

• Nếu truy vấn con trả về đúng một giá trị, nó có thể sử dụng như là một thành phần bên trong một biểu thức (chẳng hạn xuất hiện trong một phép so sánh bằng).

Giả sử ta có cơ sở dữ liệu quản lý các mặt hàng được cung cấp bởi những nhà cung cấp khác nhau, gồm 3 quan hệ sau đây:

S(S#, SNAME, CITY); P(P#, PNAME, COLOR); SP(S#, P#, SLUONG)

Trong đó S#, P# là mã số của hãng và của sản phẩm. SNAME, PNAME là tên của hãng và tên của sản phẩm. S, P, SP lần lượt là các danh sách các hãng cung ứng, các mặt hàng, và danh sách các mặt hàng đã được cung ứng. Một hãng có thể cung ứng nhiều mặt hàng và một mặt hàng có thể do nhiều hãng cung ứng.

Thực hiện các yêu cầu khai thác dữ liệu sau:

Ví dụ 4.2.13. (Sử dụng exits)

+ Tìm những nhà cung cấp cung cấp ít nhất một mặt hàng

```
SELECT * FROM S
WHERE EXITS (SELECT * FROM SP WHERE SP.S# = S.S#)
```

+ Tìm những hãng chưa hoạt động (tức có tên trong S nhưng chưa cung cấp hàng trong SP)

```
SELECT * FROM S
WHERE NOT EXITS (SELECT * FROM SP WHERE SP.S#=S.S#)
```

Có thể sử dụng NOT IN và IN để giải quyết yêu cầu trên.

Kết quả của truy vấn con có thể được sử dụng để thực hiện phép so sánh số học với một biểu thức của truy vấn cha. Trong trường hợp này, truy vấn con được sử dụng dưới dạng:

```
WHERE biểu_thức phép_toán_số_học [ANY|ALL]
          (truy_vấn_con)
```

Trong đó phép toán số học có thể sử dụng bao gồm: =, <>, >, <, >=, <=; Và truy vấn con phải có kết quả bao gồm đúng một cột.

Ví dụ. Câu lệnh dưới đây cho biết danh sách các mặt hàng có số lượng lớn hơn hoặc bằng số lượng của mặt hàng có mã số là P*I-001*

```
SELECT *
FROM SP
WHERE SLUONG>=(SELECT SLUONG
               FROM SP
               WHERE P#='PI-001')
```

Nếu truy vấn con trả về nhiều hơn một giá trị, việc sử dụng phép so sánh như trên sẽ không hợp lệ. Trong trường hợp này, sau phép toán so sánh phải sử dụng thêm lượng từ ALL hoặc ANY. Lượng từ ALL được sử dụng khi cần so sánh giá trị của biểu thức với tất cả các giá trị trả về trong kết quả của truy vấn con; ngược lai, phép so sánh với lượng từ ANY có kết quả đúng khi chỉ cần một giá trị bất kỳ nào đó trong kết quả của truy vấn con thoả mãn điều kiện.

Ví dụ 4.2.14. (Sử dụng ANY)

+ Tìm tên của những mặt hàng có mã số mặt hàng là mặt hàng nào đó mà hãng S1 đã bán.

```
SELECT PNAME FROM P
WHERE P#= ANY (SELECT P# FROM SP WHERE S#='S1')
```

+ Tìm tên những hãng cung cấp ít nhất một mặt hàng màu đỏ

```
SELECT SNAME FROM S
WHERE S# = ANY (
SELECT S# FROM SP, S
WHERE S.S#=SP.S# AND P.P#=SP.P# AND P.COLOR='RED')
```

Ví dụ 4.2.15. (Sử dụng ALL)

+ Tìm mã số những hãng cung cấp số lượng một lần một mặt hàng nào đó lớn hơn hoặc bằng số lượng mỗi lần cung cấp của các hãng.

```
SELECT S#  FROM SP
WHERE SLUONG >= ALL (SELECT SLUONG FROM SP)
```

Câu lệnh trên tương đương với

```
SELECT S# FROM SP
WHERE SLUONG IN(SELECT MAX(SLUONG) FROM SP).
```

Trong các ví dụ trên các câu truy vấn ở sau các mệnh đề IN, NOT IN, EXITS, ANY… được gọi là các truy vấn con.

Ở đây ta có một số lưu ý về sử dụng SQL để cài đặt phép chia nhằm giải quyết một số câu truy vấn có yêu cầu "tất cả".

Phép chia R÷S là tập các giá trị a_i trong R sao cho không có giá trị b_i nào trong S làm cho bộ (a_i, b_i) không tồn tại trong R.

R	**A**	**B**	**C**	**D**	**E**
	α	A	α	a	1
	α	A	γ	a	1
	α	A	γ	b	1
	β	A	γ	a	1
	β	A	γ	b	3
	γ	A	γ	a	1
	γ	A	γ	b	1
	γ	A	β	b	1

S	**D**	**E**
	A	1
	B	1

R÷S	**A**	**B**	**C**
	α	a	γ
	γ	a	γ

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:

"Tìm tên các nhân viên được phân công làm tất cả các đồ án" tương đương với câu truy vấn sau: "Tìm tên các nhân viên mà không có đề án nào là không được phân công làm"

Tập bị chia: PHANCONG(MA_NVIEN, SODA)

Tập chia: DEAN(MADA)

Tập kết quả: KQ(MA_NVIEN)

Kết KQ nối với NHANVIEN để lấy ra TENNV

4.2.1.4. Tạo mới bảng dữ liệu từ kết quả của câu lệnh SELECT

Câu lệnh SELECT ... INTO có tác dụng tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của truy vấn. Bảng mới được tạo ra sẽ có số cột bằng số cột được chỉ định trong danh sách chọn và số dòng sẽ là số dòng kết quả của truy vấn

Ví dụ 4.2.16. Câu lệnh dưới đây truy vấn dữ liệu từ bảng SINHVIEN và tạo một bảng TUOISV bao gồm các trường HODEM, TEN và TUOI

```
    SELECT hodem,ten,YEAR(GETDATE())-YEAR(ngaysinh)
AS tuoi FROM sinhvien INTO tuoisv
```

Lưu ý: Nếu trong danh sách chọn có các biểu thức thì những biểu thức này phải được đặt tiêu đề.

4.2.1.5. 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. Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng.

Ví dụ 4.2.17. Câu lệnh dưới đây hiển thị danh sách các môn học và sắp xếp theo chiều giảm dần của số đơn vị tín chỉ.

```
SELECT * FROM monhoc ORDER BY sodvtc DESC
```

MAMONHOC	TENMONHOC	SODVHT
TI-002	Ngôn ngữ C	5
TI-003	Lý thuyết hệ điều hành	4
TI-001	Tin học đại cương	4
TI-004	Cấu trúc dữ liệu và giải thuật	4
TO-001	Đại số tuyến tính	4
TO-002	Giải tích 1	4
HO-001	Hoá đại cương	3
VL-001	Vật lý đại cương	3
TO-004	Bài tập Giải tích 1	2
TO-003	Bài tập Đại số	2

Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo thứ tự từ trái qua phải.

Ví dụ 4.2.18. Câu lệnh

```
SELECT hodem,ten,gioitinh,
       YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM sinhvien WHERE ten='Bình'
ORDER BY gioitinh,tuoi
```

có kết quả là:

HODEM	TEN	GIOITINH	TUOI
Nguyễn Thị	Bình	0	23
Hoàng Văn	Bình	1	21
Châu Văn Quốc	Bình	1	21
Nguyễn Thanh	Bình	1	22
Nguyễn Đình	Bình	1	22
Nguyễn Công	Bình	1	25

Thay vì chỉ định tên cột sau ORDER BY, ta có thể chỉ định số thứ tự của cột cần được sắp xếp. Câu lệnh ở ví dụ trên có thể được viết lại như sau:

```
SELECT hodem,ten,gioitinh,YEAR(GETDATE())-
```

```
YEAR(ngaysinh) AS tuoi FROM sinhvien WHERE ten='Bình'
      ORDER BY 3, 4
```

4.2.1.6. Phép hợp

Phép hợp được sử dụng trong trường hợp ta cần gộp kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất. SQL cung cấp toán tử UNION để thực hiện phép hợp. Cú pháp như sau:

```
Câu_lệnh_1
UNION [ALL] Câu_lệnh_2
[UNION [ALL] Câu_lệnh_3]
...
[UNION [ALL] Câu_lệnh_n]
 [ORDER BY  cột_sắp_xếp]
[COMPUTE  danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Trong đó

Câu_lệnh_1 có dạng

```
SELECT  danh_sách_cột
[INTO tên_bảng_mới]
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
```

và *Câu_lệnh_i* (i = 2,..,n) có dạng

```
SELECT  danh_sách_cột
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
```

Ví dụ 4.2.19. Giả sử ta có hai bảng Table1 và Table2 lần lượt như sau:

A	B	C
a	1	10
b	2	20
c	3	30
d	4	40
a	5	50
b	6	60

D	E
a	1
b	2
d	3
e	4

câu lệnh

```
SELECT A,B FROM Table1
 UNION
 SELECT D,E FROM table2
```

Cho kết quả như sau:

A	B
a	1
a	5
b	2
b	6
c	3
d	3
d	4
e	4

Nếu trong các truy vấn thành phần của phép hợp xuất hiện những dòng dữ liệu giống nhau thì trong kết quả truy vấn chỉ giữ lại một dòng. Nếu muốn giữ lại các dòng này, ta phải sử dụng thêm từ khoá ALL trong truy vấn thành phần.

Ví dụ 4.2.20. Câu lệnh

```
SELECT A,B FROM Table1
 UNION ALL
 SELECT D,E FROM table2
```

Cho kết quả như sau

A	B
a	1
b	2
c	3
d	4
a	5
b	6
a	1
b	2
d	3
e	4

Cần chú ý các nguyên tắc sau:

• Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.

• Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong bản thân mỗi truy vấn thành phần phải cùng kiểu dữ liệu.

• Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau. Nguyên nhân là do phép hợp so sánh các cột từng cột một theo thứ tự được cho trong mỗi truy vấn.

• Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).

• Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

• Truy vấn thành phần đầu tiên có thể có INTO để tạo mới một bảng từ kết quả của chính phép hợp.

• Mệnh đề ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng ở trong bất kỳ truy vấn thành phần nào.

• Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác

động lên kết quả chung của phép hợp.

• Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.

• Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

4.2.2. Các loại phép nối

4.2.2.1. Phép nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép nối. Một câu lệnh nối kết hợp các dòng dữ liệu trong các bảng khác nhau lại theo một hoặc nhiều điều kiện nào đó và hiển thị chúng trong kết quả truy vấn.

Ví dụ 4.2.21. Xét hai bảng Khoa và Lop nói ở đầu chương, giả sử ta cần biết mã lớp và tên lớp của các lớp thuộc Khoa Tin, ta phải làm như sau:

• Chọn ra dòng trong bảng KHOA có tên khoa là *Khoa Tin*, từ đó xác định được mã khoa (MAKHOA) là *DHT02*.

• Tìm kiếm trong bảng LOP những dòng có giá trị trường MAKHOA là *DHT02* (tức là bằng MAKHOA tương ứng trong bảng KHOA) và đưa những dòng này vào kết quả truy vấn.

Như vậy, để thực hiện được yêu cầu truy vấn dữ liệu trên, ta phải thực hiện phép nối giữa hai bảng KHOA và LOP với điều kiện nối là MAKHOA của KHOA bằng với MAKHOA của LOP. Câu lệnh sẽ được viết như sau:

```
SELECT malop,tenlop
FROM khoa,lop
WHERE khoa.makhoa = lop.makhoa AND tenkhoa='Khoa
Tin'
```

Một câu lệnh nối cũng được bắt đầu với từ khóa SELECT. Các cột được chỉ định tên sau từ khoá SELECT là các cột được hiển thị trong kết quả truy vấn. Việc sử dụng tên các cột trong danh sách chọn có thể là:

• 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 4.2.22. Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: mã sinh viên, họ và tên, mã lớp, tên lớp và tên khoa

```
SELECT masv,hodem,ten,sinhvien.malop,tenlop,tenkhoa
FROM sinhvien,lop,khoa
WHERE sinhvien.malop=lop.malop
AND lop.makhoa = khoa.makhoa
```

Trong câu lệnh trên, các bảng tham gia vào truy vấn bao gồm SINHVIEN, LOP và KHOA. Điều kiện để thực hiện phép nối giữa các bảng bao gồm hai điều kiện: sinhvien.malop = lop.malop và lop.malop = khoa.malop.

4.2.2.2. Phép nối bằng và phép nối tự nhiên

Một *phép nối bằng* (equi-join) là một phép nối trong đó giá trị của các cột được sử dụng để nối được so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia nối đều được đưa ra trong kết quả.

Ví dụ 4.2.23. Câu lệnh dưới đây thực hiện phép nối bằng giữa hai bảng LOP và KHOA

```
SELECT *
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

Một dạng đặc biệt của phép nối bằng được sử dụng nhiều là *phép nối tự nhiên* (natural-join). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khoá ngoài và khoá chính của hai bảng; Trong danh sách chọn của câu lệnh chỉ giữ lại một cột trong hai cột tham gia vào điều kiện của phép nối

Ví dụ 4.2.24. Để thực hiện phép nối tự nhiên, câu lệnh trong ví dụ 2.23

được viết lại như sau

```
SELECT malop,tenlop,khoa,hedaotao,namnhaphoc,
   siso,lop.makhoa,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

hoặc viết dưới dạng ngắn gọn hơn:

```
SELECT lop.*,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

4.2.2.3. Phép tự nối và các bí danh

Ví dụ 4.2.25. Để biết được họ tên và ngày sinh của các sinh viên có cùng ngày sinh với sinh viên *Phạm Thị Quỳnh Anh*, ta phải thực hiện phép tự nối ngay trên chính bảng *sinhvien*. Trong câu lệnh nối, bảng *sinhvien* xuất hiện trong mệnh đề FROM với bí danh là *a* và *b*. Bảng *sinhvien* với bí danh là *a* sử dụng để chọn ra sinh viên có họ tên là *Phạm Thị Quỳnh Anh* và bảng *sinhvien* với bí danh là *b* sử dụng để xác định các sinh viên trùng ngày sinh với sinh viên *Phạm Thị Quỳnh Anh*. Câu lệnh được viết như sau:

```
SELECT b.hodem,b.ten,b.ngaysinh
FROM sinhvien a, sinhvien b
WHERE a.hodem='Phạm Thị Quỳnh' AND a.ten='Anh' AND
      a.ngaysinh=b.ngaysinh AND a.masv<>b.masv
```

SQL đưa ra một cách khác để biểu diễn cho phép nối, trong cách biểu diễn này, điều kiện của phép nối không được chỉ định trong mệnh đề WHERE mà được chỉ định ngay trong mệnh đề FROM của câu lệnh. Cách sử dụng phép nối này cho phép ta biểu diễn phép nối cũng như điều kiện nối được rõ ràng, đặc biệt là trong trường hợp phép nối được thực hiện trên ba bảng trở lên.

4.2.2.4. Phép nối trong

Điều kiện để thực hiện phép nối trong được chỉ định trong mệnh đề FROM

theo cú pháp như sau:

```
     tên_bảng_1 [INNER] JOIN tên_bảng_2 ON
điều_kiện_nối
```

Ví dụ 4.2.26. Để hiển thị họ tên và ngày sinh của các sinh viên lớp *Tin 2010*, thay vì sử dụng câu lệnh:

```
SELECT hodem,ten,ngaysinh
FROM sinhvien,lop
WHERE tenlop='Tin 2010' AND
sinhvien.malop=lop.malop
```

ta có thể sử dụng câu lệnh như sau:

```
SELECT hodem,ten,ngaysinh
FROM sinhvien INNER JOIN lop
     ON sinhvien.malop=lop.malop
WHERE tenlop='Tin 2010'
```

4.2.2.5. Phép nối ngoài

SQL cung cấp các phép nối ngoài sau đây:

- Phép nối ngoài trái (LEFT OUTER JOIN)
- Phép nối ngoài phải (RIGHT OUTER JOIN)
- Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Ví dụ 4.2.27. Giả sử ta có hai bảng dữ liệu như sau:

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Phép nối ngoài trái giữa hai bảng NHANVIEN và DONVI được biểu diễn bởi câu lệnh:

```
SELECT *
FROM nhanvien LEFT OUTER JOIN donvi
     ON nhanvien.madv=donvi.madv
```

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Câu lệnh:

```
SELECT *
FROM nhanvien RIGHT OUTER JOIN donvi
ON nhanvien.madv=donvi.madv
```

thực hiện phép nối ngoài phải giữa hai bảng NHANVIEN và DONVI, có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

Nếu phép nối ngoài trái (tương ứng phải) hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của bảng bên trái (tương ứng phải) trong phép nối thì phép nối ngoài đầy đủ hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của cả hai bảng tham gia vào phép nối.

Ví dụ 4.2.28. Với hai bảng NHANVIEN và DONVI như ở trên, câu lệnh

```
SELECT *
FROM nhanvien FULL OUTER JOIN donvi
ON nhanvien.madv=donvi.madv
```

cho kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL
NULL	NULL	4	Kinh doanh
NULL	NULL	3	Ke toan

Ví dụ 4.2.29. Câu lệnh dưới đây hiển thị họ tên và ngày sinh của các sinh viên thuộc *Khoa Tin học*

```
SELECT hodem,ten,ngaysinh
FROM (sinhvien INNER JOIN lop
ON sinhvien.malop=lop.malop)
INNER JOIN khoa ON lop.makhoa=khoa.makhoa
WHERE tenkhoa=N'Khoa Tin học'
```

4.2.3. Thống kê dữ liệu với GROUP BY và HAVING

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 gộp 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 gộp dưới đây:

Hàm gộp	Chức năng
SUM([ALL | DISTINCT] *biểu_thức*)	Tính tổng các giá trị.
AVG([ALL | DISTINCT] *biểu_thức*)	Tính trung bình của các giá trị
COUNT([ALL | DISTINCT] *biểu_thức*)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn.
MAX(*biểu_thức*)	Tính giá trị lớn nhất
MIN(*biểu_thức*)	Tính giá trị nhỏ nhất

Trong đó:

- Hàm SUM và 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.

Mặc định, các hàm gộp thực hiện tính toán thống kê trên toàn bộ dữ liệu. Trong trường hợp cần loại bỏ bớt các giá trị trùng nhau (chỉ giữ lại một giá trị), ta chỉ định thêm từ khoá DISTINCT ở trước biểu thức là đối số của hàm.

4.2.3.1. 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 gộp 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 gộp.

Ví dụ 4.2.30. Để thống kê trung bình điểm lần 1 của tất cả các môn học, ta sử dụng câu lệnh như sau:

```
SELECT AVG(diemlan1)
FROM diemthi
```

còn câu lệnh dưới đây cho biết tuổi lớn nhất, tuổi nhỏ nhất và độ tuổi trung bình của tất cả các sinh viên sinh tại *Huế*:

```
SELECT MAX(YEAR(GETDATE())-YEAR(ngaysinh)),
   MIN(YEAR(GETDATE())-YEAR(ngaysinh)),
   AVG(YEAR(GETDATE())-YEAR(ngaysinh))
FROM sinhvien WHERE noisinh='Huế'
```

4.2.3.2. 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 vào trong các nhóm. Các hàm gộp được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ 4.2.31. Câu lệnh dưới đây cho biết sĩ số (số lượng sinh viên) của mỗi lớp

```
SELECT lop.malop,tenlop,COUNT(masv) AS siso
FROM lop,sinhvien
WHERE lop.malop=sinhvien.malop
GROUP BY lop.malop
```

và có kết quả là

MALOP	TENLOP	SISO
C24101	Toán K24	5
C24102	Tin K24	8
C24103	Lý K24	7
C24301	Sinh K24	5
C25101	Toán K25	5
C25102	Tin K25	6
C25103	Lý K25	6
C25301	Sinh K25	8
C26101	Toán K26	5
C26102	Tin K26	5

còn câu lệnh:

```
SELECT sinhvien.masv,hodem,ten,
sum(diemlan1*sodvtc)/sum(sodvtc)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND
       diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv
```

cho biết trung bình điểm thi lần 1 các môn học của các sinh viên.

4.2.3.4. Từ khóa HAVING

Khác với điều kiện sau WHERE nhằm lựa chọn các ***bản ghi*** thỏa điều kiện truy vấn. Mệnh đề HAVING được dùng kèm với GROUP BY để chọn lựa các ***nhóm*** thoả điều kiện mới được truy xuất. Nó được sử dụng các hàm kết hợp trong mệnh đề SELECT để kiểm tra điều kiện trên các nhóm và 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 4.2.32. Để biết trung bình điểm thi lần 1 của các sinh viên có điểm trung bình lớn hơn hoặc bằng 5, ta sử dụng câu lệnh như sau:

```
SELECT sinhvien.masv,hodem,ten,
       SUM(diemlan1*sodvtc)/sum(sodvtc)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND
       diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv
HAVING sum(diemlan1*sodvtc)/sum(sodvtc)>=5
```

4.2.4. Thống kê dữ liệu với COMPUTE

Ví dụ 4.2.33. Câu lệnh

```
SELECT khoa.makhoa,tenkhoa,COUNT(malop) AS solop
FROM khoa,lop
WHERE khoa.makhoa=lop.makhoa
GROUP BY khoa.makhoa,tenkhoa
```

Cho ta biết được số lượng lớp của mỗi khoa với kết quả như sau:

MAKHOA	TENKHOA	SOLOP
DHTO1	Khoa Toán cơ - Tin học	3
DHTO2	Khoa Công nghệ thông tin	3
DHTO3	Khoa Vật lý	2
DHTO5	Khoa Sinh học	2

Nhưng cụ thể mỗi khoa bao gồm những lớp nào thì chúng ta không thể biết được trong kết quả truy vấn trên.

Mệnh đề COMPUTE …BY có cú pháp như sau:

```
COMPUTE hàm_gộp(tên_cột) [,…, hàm_gộp (tên_cột)]
BY danh_sách_cột
```

Trong đó:

• Các hàm gộp có thể sử dụng bao gồm SUM, AVG, MIN, MAX và COUNT.

• *danh_sách_cột*: là danh sách cột sử dụng để phân nhóm dữ liệu

Ví dụ 4.2.34. Câu lệnh dưới đây cho biết danh sách các lớp của mỗi khoa và tổng số các lớp của mỗi khoa:

```
SELECT khoa.makhoa,tenkhoa,malop,tenlop FROM khoa,lop
WHERE khoa.makhoa=lop.makhoa
ORDER BY khoa.makhoa
COMPUTE COUNT(malop) BY khoa.makhoa
```

kết quả của câu lệnh như sau:

```
MAKHOA  TENKHOA      MALOP TENLOP
DHT01   Khoa Toán    C24101 Toán S24
DHT01   Khoa Toán    C25101 Toán S25
DHT01   Khoa Toán    C26101 Toán S26
          CNT                    3
MAKHOA  TENKHOA      MALOP TENLOP
DHT02 Khoa Tin  C26102 Tin S26
DHT02 Khoa Tin  C25102 Tin S25
DHT02 Khoa Tin  C24102 Tin S24
          CNT                    3
MAKHOA  TENKHOA      MALOP TENLOP
DHT03 Khoa Vật lý    C24103 Lý S24
DHT03 Khoa Vật lý    C25103 Lý S25
          CNT                    2
MAKHOA  TENKHOA      MALOP TENLOP
DHT05 Khoa Sinh học    C25301 Sinh S25
DHT05 Khoa Sinh học    C24301 Sinh S24
```

CNT 2

Khi sử dụng mệnh đề COMPUTE ... BY cần tuân theo các qui tắc dưới đây:

• Từ khóa DISTINCT không cho phép sử dụng với các hàm gộp dòng

• Hàm COUNT(*) không được sử dụng trong COMPUTE.

• Sau COMPUTE có thể sử dụng nhiều hàm gộp, khi đó các hàm phải phân cách nhau bởi dấu phẩy.

• Các cột sử dụng trong các hàm gộp xuất hiện trong mệnh đề COMPUTE phải có mặt trong danh sách chọn.

• Không sử dụng SELECT INTO trong một câu lệnh SELECT có sử dụng COMPUTE.

• Mệnh đề COMPUTE và ORDER BY không được phép sử dụng trong truy vấn con.

4.2.5. Bổ sung, cập nhật và xoá dữ liệu

Trong SQL có các lệnh sau để thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu.

• INSERT

• UPDATE

• DELETE

4.2.5.1. Bổ sung dữ liệu

Câu lệnh INSERT để bổ sung một dòng dữ liệu mới vào bảng, cú pháp:

```
INSERT  INTO tên_bảng[(danh_sách_cột)]
VALUES(danh_sách_trị)
```

Ví dụ 4.2.35. Câu lệnh dưới đây bổ sung thêm một dòng dữ liệu vào bảng KHOA

```
INSERT INTO khoa
VALUES('DHT10','Khoa Tin học','054821135')
```

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 chấp nhận giá trị NULL).

Ví dụ 4.2.36. Câu lệnh dưới đây bổ sung một bản ghi mới cho bảng SINHVIEN.

```
  INSERT INTO
sinhvien(masv,hodem,ten,gioitinh,malop)
VALUES('0241020008','NguyễnCông','Vinh',1,'C24102')
```

câu lệnh trên còn có thể được viết như sau:

```
INSERT INTO sinhvien
VALUES('0241020008','Nguyễn Công','Vinh',
NULL,1,NULL,'C24102')
```

Một cách sử dụng khác của câu lệnh INSERT được sử dụng để bổ sung nhiều dòng dữ liệu vào một bảng, các dòng dữ liệu này được lấy từ một bảng khác thông qua câu lệnh SELECT

Cú pháp:

```
INSERT  INTO tên_bảng[(danh_sách_cột)]
câu_lệnh_SELECT
```

Ví dụ 4.2.37. Giả sử ta có bảng LUUSINHVIEN bao gồm các trường HODEM, TEN, NGAYSINH. Câu lệnh dưới đây bổ sung vào bảng LUUSINHVIEN các dòng dữ liệu có được từ câu truy vấn SELECT:

```
INSERT INTO luusinhvien
SELECT  hodem,ten,ngaysinh
FROM sinhvien WHERE noisinh like '%Huế%'
```

4.2.5.2 Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh có cú pháp như sau:

```
UPDATE tên_bảng
SET tên_cột = biểu_thức
[, ..., tên_cột_k = biểu_thức_k]
```

```
[FROM  danh_sách_bảng]
[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 các 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 (nếu không chỉ định, phạm vi tác động của câu lệnh được hiểu là toàn bộ các dòng trong bảng)

Ví dụ 4.2.38. Câu lệnh dưới đây cập nhật lại số đơn vị tín chỉ của các môn học có số đơn vị tín chỉ nhỏ hơn 2

```
UPDATE monhoc
SET sodvtc = 3
WHERE sodvtc = 2
```

Cấu trúc CASE có thể được sử dụng trong biểu thức khi cần phải đưa ra các quyết định khác nhau về giá trị của biểu thức

Ví dụ 4.2.39. Giả sử ta có bảng NHATKYPHONG sau đây

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	
202	B	5	
101	A	2	
102	C	3	

Sau khi thực hiện câu lệnh:

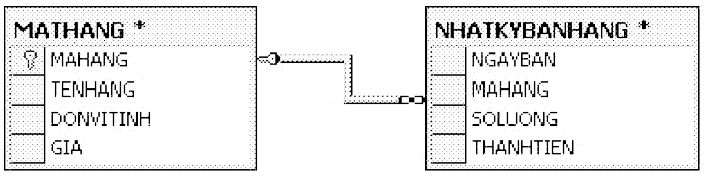
```
UPDATE nhatkyphong
SET tienphong=songay*CASE WHEN loaiphong='A' THEN
100 WHEN loaiphong='B' THEN 70 ELSE 50 END
```

Dữ liệu trong bảng sẽ là:

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	500
202	B	5	350
101	A	2	200
102	C	3	150

Mệnh đề FROM trong câu lệnh UPDATE được sử dụng khi cần chỉ định các điều kiện liên quan đến các bảng khác với bảng cần cập nhật dữ liệu. Trong trường hợp này, trong mệnh đề WHERE thường có điều kiện nối giữa các bảng.

Ví dụ 4.2.40. Giả sử ta có hai bảng MATHANG và NHATKYBANHANG như sau:



Câu lệnh dưới đây sẽ cập nhật giá trị trường THANHTIEN của bảng NHATKYBANHANG theo công thức THANHTIEN = SOLUONG × GIA

```
UPDATE nhatkybanhang
SET  thanhtien = soluong*gia
FROM mathang
WHERE nhatkybanhang.mahang = mathang.mahang
```

Ví dụ 4.2.41. Câu lệnh ở trên có thể được viết như sau:

```
UPDATE nhatkybanhang
SET  thanhtien = soluong*gia
FROM mathang
WHERE mathang.mahang =(SELECT mathang.mahang
FROM mathang
WHERE mathang.mahang=nhatkybanhang.mahang)
```

4.2.5.3. Xoá dữ liệu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE. Cú pháp của câu lệnh này như sau:

```
DELETE FROM tên_bảng
FROM danh_sách_bảng]
WHERE điều_kiện]
```

Trong câu lệnh này, 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 dữ liệu trong bảng đều bị xoá.

Ví dụ 4.2.42: Câu lệnh dưới đây xoá khỏi bảng SINHVIEN những sinh viên sinh tại Huế

```
DELETE FROM sinhvien
WHERE noisinh LIKE '%Huế%'
```

Nếu điều kiện trong câu lệnh DELETE liên quan đến các bảng không phải là bảng cần xóa dữ liệu, ta phải sử dụng thêm mệnh đề FROM và sau đó là danh sách tên các bảng đó. Trong trường hợp này, trong mệnh đề WHERE ta chỉ định thêm điều kiện nối giữa các bảng.

Ví dụ 4.2.43. Câu lệnh dưới đây xoá ra khỏi bảng SINHVIEN những sinh viên lớp *Tin S24*

```
DELETE FROM sinhvien
FROM lop
WHERE lop.malop=sinhvien.malop AND tenlop='Tin S24'
```

Một câu lệnh SELECT có thể được lồng vào trong mệnh đề WHERE trong câu lệnh DELETE để làm điều kiện cho câu lệnh tương tự như câu lệnh UPDATE.

Ví dụ 4.2.44. Câu lệnh dưới đây xoá khỏi bảng LOP những lớp không có sinh viên nào học

```
DELETE FROM lop
WHERE malop NOT IN (SELECT DISTINCT malop
FROM sinhvien)
```

4.2.5.4. Xoá toàn bộ dữ liệu trong bảng

Câu lệnh DELETE không chỉ định điều kiện đối với các dòng dữ liệu cần xoá trong mệnh đề WHERE sẽ xoá toàn bộ dữ liệu trong bảng. Thay vì sử dụng câu lệnh DELETE trong trường hợp này, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp như sau:

```
TRUNCATE TABLE tên_bảng
```

Ví dụ 4.2.45. Câu lệnh sau xoá toàn bộ dữ liệu trong bảng *DIEMTHI*:

```
DELETE FROM DIEMTHI
```

có tác dụng tương tự với câu lệnh

```
TRUNCATE TABLE DIEMTHI
```

4.3. Ngôn ngữ định nghĩa dữ liệu

Trong phần này, chúng ta sẽ tìm hiểu nhóm các câu lệnh được sử dụng để định nghĩa và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... - ngôn ngữ định nghĩa dữ liệu (DLL).

Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

- CREATE: định nghĩa và tạo mới đối tượng CSDL.
- ALTER: thay đổi định nghĩa của đối tượng CSDL.
- DROP: Xoá đối tượng CSDL đã có.

4.3.1. Tạo bảng dữ liệu

Câu lệnh CREATE TABLE được sử dụng để định nghĩa một bảng dữ liệu mới trong cơ sở dữ liệu.

Câu lệnh CREATE TABLE có cú pháp như sau:

```
CREATE TABLE tên_bảng
(
  tên_cột   thuộc_tính_cột  các_ràng_buộc
```

```
  [,...
  ,tên_cột_n thuộc_tính_cột_n
các_ràng_buộc_cột_n]
  [,các_ràng_buộc_trên_bảng]
 )
```

Ví dụ 4.3.1. Câu lệnh dưới đây định nghĩa bảng NHANVIEN với các trường MANV (mã nhân viên), HOTEN (họ và tên), NGAYSINH (ngày sinh của nhân viên), DIENTHOAI (điện thoại) và HSLUONG (hệ số lương)

```
CREATE TABLE nhanvien
(
manv  NVARCHAR(10)  NOT NULL,
hoten NVARCHAR(50)  NOT NULL,
ngaysinh DATETIME  NULL,
dienthoai NVARCHAR(10) NULL,
hsluong DECIMAL(3,2)  DEFAULT (1.92)
)
```

Trong câu lệnh trên, trường MANV và HOTEN của bảng NHANVIEN không được NULL (tức là bắt buộc phải có dữ liệu), trường NGAYSINH và DIENTHOAI sẽ nhận giá trị NULL nếu ta không nhập dữ liệu cho chúng còn trường HSLUONG sẽ nhận giá trị mặc định là 1.92 nếu không được nhập dữ liệu.

4.3.1.1 Ràng buộc CHECK

Ràng buộc CHECK được sử dụng nhằm chỉ định điều kiện hợp lệ đối với dữ liệu. Mỗi khi có sự thay đổi dữ liệu trên bảng (INSERT, UPDATE), những ràng buộc này sẽ được sử dụng nhằm kiểm tra xem dữ liệu mới có hợp lệ hay không.

Ràng buộc CHECK được khai báo theo cú pháp như sau:

```
[CONSTRAINT tên_ràng_buộc]
CHECK (điều_kiện)
```

Trong đó, điều_kiện là một biểu thức logic tác động lên cột nhằm qui định giá trị hoặc khuôn dạng dữ liệu được cho phép. Trên mỗi một bảng cũng như trên

mỗi một cột có thể có nhiều ràng buộc CHECK.

Ví dụ 4.3.2. Câu lệnh dưới đây tạo bảng DIEMTOTNGHIEP trong đó qui định giá trị của cột DIEMVAN và DIEMTOAN phải lớn hơn hoặc bằng 0 và nhỏ hơn hoặc bằng 10.

```
CREATE TABLE diemtotnghiep
(
hoten   NVARCHAR(30) NOT NULL,
ngaysinh  DATETIME,
diemvan DECIMAL(4,2)
  CONSTRAINT chk_diemvan
          CHECK(diemvan>=0 AND diemvan<=10),
diemtoan DECIMAL(4,2)
  CONSTRAINT chk_diemtoan
          CHECK(diemtoan>=0 AND diemtoan<=10),
)
```

Thay vì chỉ định ràng buộc trên mỗi cột, ta có thể chỉ định các ràng buộc ở mức bảng bằng cách khai báo các ràng buộc sau khi đã khai báo xong các cột trong bảng.

Ví dụ 4.3.3. Câu lệnh

```
CREATE TABLE lop
(
malop    NVARCHAR(10)  NOT NULL ,
tenlop    NVARCHAR(30)  NOT NULL ,
khoa    SMALLINT    NULL ,
hedaotao    NVARCHAR(25)  NULL
    CONSTRAINT chk_lop_hedaotao
    CHECK (hedaotao IN ('chính quy','tại chức')),
namnhaphoc  INT    NULL
```

```
    CONSTRAINT chk_lop_namnhaphoc
    CHECK (namnhaphoc<=YEAR(GETDATE())),
makhoa    NVARCHAR(5)
)
```

có thể được viết lại như sau:

```
CREATE TABLE lop
(
malop    NVARCHAR(10)  NOT NULL ,
tenlop    NVARCHAR(30)  NOT NULL ,
khoa    SMALLINT    NULL ,
hedaotao    NVARCHAR(25)  NULL,
namnhaphoc  INT    NULL ,
makhoa    NVARCHAR(5),
CONSTRAINT chk_lop
CHECK (namnhaphoc<=YEAR(GETDATE()) AND
    hedaotao IN ('chính quy','tại chức'))
)
```

4.3.1.2. Ràng buộc PRIMARY KEY

Ràng buộc PRIMARY KEY được sử dụng để định nghĩa khoá chính của bảng. Ràng buộc PRIMARY KEY là cơ sở cho việc đảm bảo tính toàn vẹn thực thể cũng như toàn vẹn tham chiếu.

Để khai báo một ràng buộc PRIMARY KEY, ta sử dụng cú pháp như sau:

```
[CONSTRAINT tên_ràng_buộc]
PRIMARY KEY [(danh_sách_cột)]
```

Nếu việc khai báo khoá chính được tiến hành ở mức bảng (sử dụng khi số lượng các cột tham gia vào khoá là từ hai trở lên) thì bắt buộc phải chỉ định danh sách cột ngay sau từ khóa PRIMARY KEY và tên các cột được phân cách nhau bởi dấu phẩy.

Ví dụ 4.3.4. Câu lệnh dưới đây định nghĩa bảng SINHVIEN với khoá chính là MASV

```
CREATE TABLE sinhvien
(
masv  NVARCHAR(10)
   CONSTRAINT pk_sinhvien_masv PRIMARY KEY,
hodem NVARCHAR(25)   NOT NULL ,
ten   NVARCHAR(10)   NOT NULL ,
ngaysinh  DATETIME,
gioitinh  BIT,
noisinh  NVARCHAR(255),
malop  NVARCHAR(10)
)
```

Ví dụ 4.3.5. Câu lệnh dưới đây tạo bảng DIEMTHI với khoá chính là tập bao gồm hai cột MAMONHOC và MASV

```
CREATE TABLE diemthi
(
mamonhoc NVARCHAR(10) NOT NULL ,
masv  NVARCHAR(10) NOT NULL ,
diemlan1  NUMERIC(4, 2),
diemlan2  NUMERIC(4, 2),
CONSTRAINT pk_diemthi PRIMARY KEY(mamonhoc,masv)
)
```

4.3.1.3. Ràng buộc UNIQUE

Ràng buộc UNIQUE được sử dụng trong câu lệnh CREATE TABLE để định nghĩa khoá phụ cho bảng và được khai báo theo cú pháp sau đây:

```
[CONSTRAINT tên_ràng_buộc]
```

```
UNIQUE [(danh_sách_cột)]
```

Ví dụ 4.3.6. Giả sử ta cần định nghĩa bảng LOP với khoá chính là cột MALOP nhưng đồng thời lại không cho phép các lớp khác nhau được trùng tên lớp với nhau, ta sử dụng câu lệnh như sau:

```
CREATE TABLE lop
(
malop    NVARCHAR(10)  NOT NULL,
tenlop    NVARCHAR(30)  NOT NULL,
khoa    SMALLINT    NULL,
hedaotao    NVARCHAR(25)  NULL,
namnhaphoc  INT    NULL,
makhoa    NVARCHAR(5),
CONSTRAINT pk_lop PRIMARY KEY (malop),
CONSTRAINT unique_lop_tenlop UNIQUE(tenlop)
)
```

4.3.1.4. Ràng buộc FOREIGN KEY

Ràng buộc FOREIGN KEY được định nghĩa theo cú pháp dưới đây:

```
CONSTRAINT tên_ràng_buộc]
FOREIGN KEY [(danh_sách_cột)]
REFERENCES
tên_bảng_tham_chiếu(danh_sách_cột_tham_chiếu)
ON DELETE CASCADE | NO ACTION | SET NULL | SET
DEFAULT]
ON UPDATE CASCADE | NO ACTION | SET NULL | SET
DEFAULT]
```

Việc định nghĩa một ràng buộc FOREIGN KEY bao gồm các yếu tố sau:

- Tên cột hoặc danh sách cột của bảng được định nghĩa tham gia vào khoá ngoài.

- Tên của bảng được tham chiếu bởi khoá ngoài và danh sách các cột được tham chiếu đến trong bảng tham chiếu.
- Cách thức xử lý đối với các bản ghi trong bảng được định nghĩa trong trường hợp các bản ghi được tham chiếu trong bảng tham chiếu bị xoá (ON DELETE) hay cập nhật (ON UPDATE). SQL chuẩn đưa ra 4 cách xử lý:

  CASCADE: Tự động xoá (cập nhật) nếu bản ghi được tham chiếu bị xoá (cập nhật).

  NO ACTION: (Mặc định) Nếu bản ghi trong bảng tham chiếu đang được tham chiếu bởi một bản ghi bất kỳ trong bảng được định nghĩa thì bàn ghi đó không được phép xoá hoặc cập nhật (đối với cột được tham chiếu).

  SET NULL: Cập nhật lại khoá ngoài của bản ghi thành giá trị NULL (nếu cột cho phép nhận giá trị NULL).

  SET DEFAULT: Cập nhật lại khoá ngoài của bản ghi nhận giá trị mặc định (nếu cột có qui định giá trị mặc định).

Ví dụ 4.3.7. Câu lệnh dưới đây định nghĩa bảng DIEMTHI với hai khoá ngoài trên cột MASV và cột MAMONHOC (giả sử hai bảng SINHVIEN và MONHOC đã được định nghĩa)

```
CREATE TABLE diemthi
(
mamonhoc NVARCHAR(10) NOT NULL ,
masv  NVARCHAR(10) NOT NULL ,
diemlan1  NUMERIC(4, 2),
diemlan2  NUMERIC(4, 2),
CONSTRAINT pk_diemthi PRIMARY KEY(mamonhoc,masv),
CONSTRAINT fk_diemthi_mamonhoc
  FOREIGN KEY(mamonhoc)
  REFERENCES monhoc(mamonhoc)
```

```
    ON DELETE CASCADE
    ON UPDATE CASCADE,
CONSTRAINT fk_diemthi_masv
  FOREIGN KEY(masv)
  REFERENCES sinhvien(masv)
  ON DELETE CASCADE
   ON UPDATE CASCADE
)
```

4.3.2. Sửa đổi định nghĩa bảng

Một bảng sau khi đã được định nghĩa bằng câu lệnh CREATE TABLE có thể được sửa đổi thông qua câu lệnh ALTER TABLE. Câu lệnh này cho phép chúng ta thực hiện được các thao tác sau:

- Bổ sung một cột vào bảng.
- Xoá một cột khỏi bảng.
- Thay đổi định nghĩa của một cột trong bảng.
- Xoá bỏ hoặc bổ sung các ràng buộc cho bảng

Cú pháp của câu lệnh ALTER TABLE như sau:

```
ALTER TABLE tên_bảng
  ADD định_nghĩa_cột |
  ALTER COLUMN tên_cột  kiểu_dữ_liệu [NULL | NOT
NULL] |
  DROP COLUMN tên_cột |
  ADD CONSTRAINT tên_ràng_buộc
định_nghĩa_ràng_buộc |
  DROP CONSTRAINT tên_ràng_buộc
```

Ví dụ 4.3.8. Các ví dụ dưới đây minh hoạ cách sử dụng câu lệnh ALTER TABLE trong các trường hợp.

Giả sử ta có hai bảng DONVI và NHANVIEN với định nghĩa như sau:

```
CREATE TABLE donvi
(
madv   INT    NOT NULL PRIMARY KEY,
tendv  NVARCHAR(30)  NOT NULL
)
CREATE TABLE nhanvien
(
manv  NVARCHAR(10)  NOT NULL,
hoten NVARCHAR(30)  NOT NULL,
ngaysinh DATETIME,
diachi CHAR(30)   NOT NULL
)
```

Bổ sung vào bảng NHANVIEN cột DIENTHOAI với ràng buộc CHECK nhằm qui định điện thoại của nhân viên là một chuỗi 6 chữ số:

```
ALTER TABLE nhanvien
ADD
  dienthoai NVARCHAR(6)
  CONSTRAINT chk_nhanvien_dienthoai
  CHECK  (dienthoai  LIKE  '[0-9][0-9][0-9][0-9][0-
9][0-9]')
```

Bổ sung thêm cột MADV vào bảng NHANVIEN:

```
ALTER TABLE nhanvien
ADD
 madv INT NULL
```

Định nghĩa lại kiểu dữ liệu của cột DIACHI trong bảng NHANVIEN và cho phép cột này chấp nhận giá trị NULL:

```
ALTER TABLE nhanvien
```

```
ALTER COLUMN diachi NVARCHAR(100) NULL
```

Xoá cột ngày sinh khỏi bảng NHANVIEN:

```
ALTER TABLE nhanvien
DROP COLUMN ngaysinh
```

Định nghĩa khoá chính (ràng buộc PRIMARY KEY) cho bảng NHANVIEN là cột MANV:

```
ALTER TABLE nhanvien
ADD
CONSTRAINT pk_nhanvien PRIMARY KEY(manv)
```

Định nghĩa khoá ngoài cho bảng NHANVIEN trên cột MADV tham chiếu đến cột MADV của bảng DONVI:

```
ALTER TABLE nhanvien
ADD
  CONSTRAINT fk_nhanvien_madv
 FOREIGN KEY(madv) REFERENCES donvi(madv)
 ON DELETE CASCADE
 ON UPDATE CASCADE
```

Xoá bỏ ràng buộc kiểm tra số điện thoại của nhân viên

```
ALTER TABLE nhanvien
DROP CONSTRAINT CHK_NHANVIEN_DIENTHOAI
```

4.3.3. Xoá bảng

Câu lệnh có cú pháp như sau:

```
DROP TABLE tên_bảng
```

Câu lệnh này cũng đồng thời xoá tất cả những ràng buộc, chỉ mục, trigger liên quan đến bảng đó.

Ví dụ 4.3.9. Giả sử cột MADV trong bảng DONVI đang được tham chiếu bởi khoá ngoài *fk_nhanvien_madv* trong bảng NHANVIEN. Để xoá bảng DONVI ra khỏi cơ sở dữ liệu, ta thực hiện hai câu lệnh sau:

Xoá bỏ ràng buộc *fk_nhanvien_madv* khỏi bảng NHANVIEN:

```
ALTER TABLE nhanvien
DROP CONSTRAINT fk_nhanvien_madv
```

Xoá bảng DONVI:

```
DROP TABLE donvi
```

4.3.4. Khung nhìn

Một khung nhìn (view) có thể được xem như là một bảng "ảo" trong cơ sở dữ liệu có nội dung được định nghĩa thông qua một truy vấn (câu lệnh SELECT). Như vậy, một khung nhìn trông giống như một bảng với một tên khung nhìn và là một tập bao gồm các dòng và các cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. Thực chất dữ liệu quan sát được trong khung nhìn được lấy từ các bảng thông qua câu lệnh truy vấn dữ liệu.

Việc sử dụng khung nhìn trong cơ sở dữ liệu đem lại các lợi ích sau đây:

- **Bảo mật dữ liệu:** Người sử dụng được cấp phát quyền trên các khung nhìn với những phần dữ liệu mà người sử dụng được phép. Điều này hạn chế được phần nào việc người sử dụng truy cập trực tiếp dữ liệu.
- **Đơn giản hoá các thao tác truy vấn dữ liệu:** Một khung nhìn đóng vai trò như là một đối tượng tập hợp dữ liệu từ nhiều bảng khác nhau vào trong một "bảng". Nhờ vào đó, người sử dụng có thể thực hiện các yêu cầu truy vấn dữ liệu một cách đơn giản từ khung nhìn thay vì phải đưa ra những câu truy vấn phức tạp.
- **Tập trung và đơn giản hóa dữ liệu:** Thông qua khung nhìn ta có thể cung cấp cho người sử dụng những cấu trúc đơn giản, dễ hiểu hơn về dữ liệu trong cơ sở dữ liệu đồng thời giúp cho người sử dụng tập trung hơn trên những phần dữ liệu cần thiết.
- **Độc lập dữ liệu:** Một khung nhìn có thể cho phép người sử dụng có được cái nhìn về dữ liệu độc lập với cấu trúc của các bảng trong cơ sở dữ liệu cho dù các bảng cơ sở có bị thay đổi phần nào về cấu trúc.

Câu lệnh CREATE VIEW như sau:

```
CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)]
AS
câu_lệnh_SELECT
```

Ví dụ 4.3.10. Câu lệnh dưới đây tạo khung nhìn có tên DSSV.

```
CREATE VIEW dssv AS
 SELECT masv,hodem, ten, datediff(YY,ngaysinh,
 getdate()) as tuoi, tenlop
 FROM sinhvien,lop
 WHERE sinhvien.malop=lop.malop
```

Bảng DSSV có nội dung như sau:

MASV	HODEM	TEN	TUOI	TENLOP
0241010001	Ngô Thi Nhật	Anh	22	Toán K24
0241010002	Nguyễn Thị Ngọc	Anh	21	Toán K24
0241010003	Ngô Việt	Bắc	22	Toán K24
0241010004	Nguyễn Đình	Bình	22	Toán K24
0241010005	Hồ Đăng	Chiến	22	Toán K24
0241020001	Nguyễn Tuấn	Anh	25	Tin K24
0241020002	Trần Thị Kim	Anh	22	Tin K24
0241020003	Võ Đức	Ân	22	Tin K24
0241020004	Nguyễn Công	Bình	25	Tin K24
0241020005	Nguyễn Thanh	Bình	22	Tin K24
...	...	...	...	...

Khi sử dụng lệnh CREAT VIEW cần lưu ý một số điểm sau:

- Không thể tạo chỉ mục và qui định ràng buộc cho khung nhìn.

- Không dùng COMPUTE… BY

- Phải đặt tên cột với các lênh SELECT có các cột là biểu thức.

Ví dụ 4.3.11. Câu lệnh dưới đây là câu lệnh sai do cột thứ 4 không xác định được tên cột

```
CREATE VIEW tuoisinhvien
AS
 SELECT
masv,hodem,ten,DATEDIFF(YY,ngaysinh,GETDATE())
 FROM sinhvien
```

4.3.4.2. Cập nhật, bổ sung và xoá dữ liệu thông qua khung nhìn

Đối với một số khung nhìn, ta có thể tiến hành thực hiện các thao tác cập nhập, bổ sung và xoá dữ liệu. Thực chất, những thao tác này sẽ được chuyển thành những thao tác tương tự trên các bảng cơ sở và có tác động đến những bảng cơ sở.

Để có thể thực hiện thao tác bổ sung, cập nhật và xoá, một khung nhìn trước tiên phải thoả mãn các điều kiện sau đây:

• Trong câu lệnh SELECT định nghĩa khung nhìn không được sử dụng từ khoá DISTINCT, TOP, GROUP BY và UNION.

• Các thành phần xuất hiện trong danh sách chọn của câu lệnh SELECT phải là các cột trong các bảng cơ sở. Trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp.

Ngoài những điều kiện trên, các thao tác thay đổi đến dữ liệu thông qua khung nhìn còn phải đảm bảo thoả mãn các ràng buộc trên các bảng cơ sở, tức là vẫn đảm bảo tính toàn vẹn dữ liệu.

Ví dụ 4.3.12. Xét định nghĩa hai bảng DONVI và NHANVIEN như sau:

```
CREATE TABLE donvi
(
madv    INT    PRIMARY KEY,
tendv   NVARCHAR(30) NOT NULL,
dienthoai   NVARCHAR(10) NULL,
)
CREATE TABLE  nhanvien
(
```

```
manv   NVARCHAR(10)  PRIMARY KEY,
hoten  NVARCHAR(30) NOT NULL,
ngaysinh  DATETIME  NULL,
diachi  NVARCHAR(50) NULL,
madv   INT    FOREIGN KEY
       REFERENCES donvi(madv)
     ON DELETE CASCADE
     ON UPDATE CASCADE
)
```

Giả sử trong hai bảng này đã có dữ liệu như sau:

Bảng DONVI

MADV	TENDV	DIENTHOAI
1	P. Kinh doanh	822321
2	P. Tiep thi	822012

Bảng NHANVIEN

MANV	HOTEN	NGAYSINH	DIACHI	MADV
NV01	Tran Van A	1975-02-03 00:00:00	77 Tran Phu	1
NV02	Mai Thi B	1977-05-04 00:00:00	34 Nguyen Hue	2
NV03	Nguyen Van C	NULL	NULL	2

Câu lệnh dưới đây định nghĩa khung nhìn NV1 cung cấp các thông tin về mã nhân viên, họ tên và mã đơn vị nhân viên làm việc:

```
CREATE VIEW NV1
AS
SELECT manv,hoten,madv FROM nhanvien
```

Nếu ta thực hiện câu lệnh

```
INSERT INTO NV1 VALUES('NV04','Le Thi D',1)
```

Một bản ghi mới sẽ được bổ sung vào bảng NHANVIEN và dữ liệu trong bảng này sẽ là:

MANV	HOTEN	NGAYSINH	DIACHI	MADV
NV01	Tran Van A	1975-02-03 00:00:00	77 Tran Phu	1
NV02	Mai Thi B	1977-05-04 00:00:00	34 Nguyen Hue	2
NV03	Nguyen Van C	NULL	NULL	2
NV04	Le Thi D	NULL	NULL	1

Thông qua khung nhìn này, ta cũng có thể thực hiện thao tác cập nhật và xoá dữ liệu. Chẳng hạn, nếu ta thực hiện câu lệnh:

```
DELETE FROM nv1 WHERE manv='NV04'
```

Thì bản ghi tương ứng với nhân viên có mã NV04 sẽ bị xoá khỏi bảng NHANVIEN.

Nếu trong danh sách chọn của câu lệnh SELECT có sự xuất hiện của biểu thức tính toán đơn giản, thao tác bổ sung dữ liệu thông qua khung nhìn không thể thực hiện được. Tuy nhiên, trong trường hợp này thao tác cập nhật và xoá dữ liệu vẫn có thể có khả năng thực hiện được (hiển nhiên không thể cập nhật dữ liệu đối với một cột có được từ một biểu thức tính toán).

Ví dụ 4.3.13. Xét khung nhìn NV2 được định nghĩa như sau:

```
CREATE VIEW NV2
AS
SELECT manv,hoten,YEAR(ngaysinh) AS
namsinh,madv FROM nhanvien
```

Đối với khung nhìn NV2, ta không thể thực hiện thao tác bổ sung dữ liệu nhưng có thể cập nhật hoặc xoá dữ liệu trên bảng thông qua khung nhìn này. Câu lệnh dưới đây là không thể thực hiện được trên khung nhìn NV2

```
INSERT INTO NV2(manv,hoten,madv)
VALUES('NV05','Le Van E',1)
```

Nhưng câu lệnh:

```
UPDATE NV2 SET hoten='Le Thi X' WHERE manv='NV04'
```

hoặc câu lệnh

```
DELETE FROM NV2 WHERE manv='NV04'
```

lại có thể thực hiện được và có tác động đối với dữ liệu trong bảng

NHANVIEN.

Trong trường hợp khung nhìn được tạo ra từ một phép nối (trong hoặc ngoài) trên nhiều bảng, ta có thể thực hiện được thao tác bổ sung hoặc cập nhật dữ liệu nếu thao tác này chỉ có tác động đến đúng một bảng cơ sở (câu lệnh DELETE không thể thực hiện được trong trường hợp này).

Ví dụ 4.3.14. Với khung nhìn được định nghĩa như sau:

```
CREATE VIEW NV3
AS
 SELECT manv,hoten,ngaysinh,
     diachi,nhanvien.madv AS noilamviec,
     donvi.madv,tendv,dienthoai
 FROM nhanvien FULL OUTER JOIN donvi
       ON nhanvien.madv=donvi.madv
```

Câu lệnh:

```
INSERT INTO NV3(manv,hoten,noilamviec)
VALUES('NV05','Le Van E',1)
```

sẽ bổ sung thêm vào bảng NHANVIEN một bản ghi mới. Hoặc câu lệnh:

```
INSERT INTO NV3(madv,tendv) VALUES(3,'P. Ke toan')
```

bổ sung thêm vào bảng DONVI một bản ghi do cả hai câu lệnh này chỉ có tác động đến đúng một bảng cơ sở.

Câu lệnh dưới đây không thể thực hiện được do có tác động một lúc đến hai bảng cơ sở.

```
INSERT INTO NV3(manv,hoten,noilamviec,madv,tendv)
VALUES('NV05','Le Van E',1,3,'P. Ke toan')
```

4.4. Khả năng bảo mật cơ sở dữ liệu trong SQL

Các khả năng bảo mật cơ sở dữ liệu trong SQL gồm những chức năng như:

• Cấp phát quyền truy cập cơ sở dữ liệu cho người dùng và các nhóm người dùng, phát hiện và ngăn chặn những thao tác trái phép của người sử dụng trên cơ

sở dữ liệu.

• Cấp phát quyền sử dụng các câu lệnh, các đối tượng cơ sở dữ liệu đối với người dùng.

• Thu hồi (huỷ bỏ) quyền của người dùng.

Bảo mật dữ liệu trong SQL được thực hiện dựa trên ba khái niệm chính sau đây:

• **Người dùng cơ sở dữ liệu (Database user):** Là đối tượng sử dụng cơ sở dữ liệu, thực thi các thao tác trên cơ sở dữ liệu như tạo bảng, truy xuất dữ liệu,... Mỗi một người dùng trong cơ sở dữ liệu được xác định thông qua tên người dùng (User ID). Một tập nhiều người dùng có thể được tổ chức trong một nhóm và được gọi là nhóm người dùng (User Group). Chính sách bảo mật cơ sở dữ liệu có thể được áp dụng cho mỗi người dùng hoặc cho các nhóm người dùng.

• **Các đối tượng cơ sở dữ liệu (Database objects):** Tập hợp các đối tượng, các cấu trúc lưu trữ được sử dụng trong cơ sở dữ liệu như bảng, khung nhìn, thủ tục, hàm được gọi là các đối tượng cơ sở dữ liệu. Đây là những đối tượng cần được bảo vệ trong chính sách bảo mật của cơ sở dữ liệu.

• **Đặc quyền (Privileges):** Là tập những thao tác được cấp phát cho người dùng trên các đối tượng cơ sở dữ liệu. Chẳng hạn một người dùng có thể truy xuất dữ liệu trên một bảng bằng câu lệnh SELECT nhưng có thể không thể thực hiện các câu lệnh INSERT, UPDATE hay DELETE trên bảng đó.

SQL cung cấp hai câu lệnh cho phép chúng ta thiết lập các chính sách bảo mật trong cơ sở dữ liệu:

• Lệnh GRANT: Sử dụng để cấp phát quyền cho người sử dụng trên các đối tượng cơ sở dữ liệu hoặc quyền sử dụng các câu lệnh SQL trong cơ sở dữ liệu.

• Lệnh REVOKE: Được sử dụng để thu hồi quyền đối với người sử dụng.

4.4.1. Cấp phát quyền

Câu lệnh GRANT được sử dụng để cấp phát quyền cho người dùng hay nhóm người dùng trên các đối tượng cơ sở dữ liệu. Câu lệnh này thường được sử dụng trong các trường hợp sau:

• Người sở hữu đối tượng cơ sở dữ liệu muốn cho phép người dùng khác

quyền sử dụng những đối tượng mà anh ta đang sở hữu.

• Người sở hữu cơ sở dữ liệu cấp phát quyền thực thi các câu lệnh (như CREATE TABLE, CREATE VIEW,...) cho những người dùng khác.

4.4.1.1. Cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu

Chỉ có người sở hữu cơ sở dữ liệu hoặc người sở hữu đối tượng cơ sở dữ liệu mới có thể cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu. Câu lệnh GRANT trong trường hợp này có cú pháp như sau:

```
GRANT ALL [PRIVILEGES]| các_quyền_cấp_phát
    [(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
    |ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
    |ON tên_thủ_tục
    |ON tên_hàm
TO danh_sách_người_dùng | nhóm_người_dùng
[WITH GRANT OPTION ]
```

Ví dụ 4.4.1. Cấp phát cho người dùng có tên *thuchanh* quyền thực thi các câu lệnh SELECT, INSERT và UPDATE trên bảng LOP

```
GRANT SELECT,INSERT,UPDATE
ON lop
TO thuchanh
```

Cho phép người dùng *thuchanh* quyền xem họ tên và ngày sinh của các sinh viên (cột HODEM,TEN và NGAYSINH của bảng SINHVIEN)

```
GRANT SELECT
(hodem,ten,ngaysinh) ON sinhvien
TO thuchanh
```

hoặc:

```
GRANT SELECT
ON sinhvien(hodem,ten,ngaysinh)
TO thuchanh
```

Với quyền được cấp phát như trên, người dùng *thuchanh* có thể thực hiện câu lệnh sau trên bảng SINHVIEN

```
SELECT hoden,ten,ngaysinh
```

Nhưng câu lệnh dưới đây lại không thể thực hiện được

```
SELECT * FROM sinhvien
```

Trong trường hợp cần cấp phát tất cả các quyền có thể thực hiện được trên đối tượng cơ sở dữ liệu cho người dùng, thay vì liệt kê các câu lệnh, ta chỉ cần sử dụng từ khoá ALL PRIVILEGES (từ khóa PRIVILEGES có thể không cần chỉ định). Câu lệnh dưới đây cấp phát cho người dùng *thuchanh* các quyền SELECT, INSERT, UPDATE, DELETE VÀ REFERENCES trên bảng DIEMTHI

```
GRANT ALL
 ON DIEMTHI
 TO thuchanh
```

Khi ta cấp phát quyền nào đó cho một người dùng trên một đối tượng cơ sở dữ liệu, người dùng đó có thể thực thi câu lệnh được cho phép trên đối tượng đã cấp phát. Tuy nhiên, người dùng đó không có quyền cấp phát những quyền mà mình được phép cho những người sử dụng khác.Trong một số trường hợp, khi ta cấp phát quyền cho một người dùng nào đó, ta có thể cho phép người đó chuyển tiếp quyền cho người dùng khác bằng cách chỉ định tuỳ chọn WITH GRANT OPTION trong câu lệnh GRANT.

Ví dụ 4.4.2. Cho phép người dùng *thuchanh* quyền xem dữ liệu trên bảng SINHVIEN đồng thời có thể chuyển tiếp quyền này cho người dùng khác

```
GRANT SELECT
ON sinhvien
TO thuchanh
WITH GRANT OPTION
```

4.4.1.2. Cấp phát quyền thực thi các câu lệnh

Ngoài chức năng cấp phát quyền cho người sử dụng trên các đối tượng cơ sở dữ liệu, câu lệnh GRANT còn có thể sử dụng để cấp phát cho người sử dụng một số quyền trên hệ quản trị cơ sở dữ liệu hoặc cơ sở dữ liệu. Những quyền có

thể cấp phát trong trường hợp này bao gồm:

- Tạo cơ sở dữ liệu: CREATE DATEBASE.
- Tạo bảng: CREATE RULE
- Tạo khung nhìn: CREATE VIEW
- Tạo thủ tục lưu trữ: CREATE PROCEDURE
- Tạo hàm: CREATE FUNCTION
- Sao lưu cơ sở dữ liệu: BACKUP DATABASE

Câu lệnh GRANT sử dụng trong trường hợp này có cú pháp như sau:

```
GRANT  ALL | danh_sách_câu_lênh
TO danh_sách_người_dùng
```

Ví dụ 4.4.3. Để cấp phát quyền tạo bảng và khung nhìn cho người dùng có tên là *thuchanh*, ta sử dụng câu lệnh như sau:

```
GRANT CREATE TABLE,CREATE VIEW
TO thuchanh
```

Khác với trường hợp sử dụng câu lệnh GRANT để cấp phát quyền trên đối tượng cơ sở dữ liệu, câu lệnh GRANT trong trường hợp này không thể sử dụng tuỳ chọn WITH GRANT OPTION, tức là người dùng không thể chuyển tiếp được các quyền thực thi các câu lệnh đã được cấp phát.

4.4.2. Thu hồi quyền

Câu lệnh REVOKE được sử dụng để thu hồi quyền đã được cấp phát cho người dùng. Tương ứng với câu lệnh GRANT, câu lệnh REVOKE được sử dụng trong hai trường hợp:

- Thu hồi quyền đã cấp phát cho người dùng trên các đối tượng cơ sở dữ liệu.
- Thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu đã cấp phát cho người dùng.

4.4.2.1. Thu hồi quyền trên đối tượng cơ sở dữ liệu

Cú pháp câu lệnh REVOKE sử dụng để thu hồi quyền đã cấp phát trên đối tượng cơ sở dữ liệu có cú pháp như sau:

```
REVOKE [GRANT OPTION FOR]
       ALL [PRIVILEGES]| các_quyền_cần_thu_hồi
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
|ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
|ON tên_thủ_tục
|ON tên_hàm
FROM danh_sách_người_dùng
[CASCADE]
```

Câu lệnh REVOKE có thể sử dụng để thu hồi một số quyền đã cấp phát cho người dùng hoặc là thu hồi tất cả các quyền (ALL PRIVILEGES).

Ví dụ 4.4.4. Thu hồi quyền thực thi lệnh INSERT trên bảng LOP đối với người dùng *thuchanh*.

```
REVOKE INSERT
ON lop
FROM thuchanh
```

Giả sử người dùng *thuchanh* đã được cấp phát quyền xem dữ liệu trên các cột HODEM, TEN và NGAYSINH của bảng SINHVIEN, câu lệnh dưới đây sẽ thu hồi quyền đã cấp phát trên cột NGAYSINH (chỉ cho phép xem dữ liệu trên cột HODEM và TEN)

```
REVOKE SELECT
ON sinhvien(ngaysinh)
FROM thuchanh
```

Khi ta sử dụng câu lệnh REVOKE để thu hồi quyền trên một đối tượng cơ sở dữ liệu từ một người dùng náo đó, chỉ những quyền mà ta đã cấp phát trước đó mới được thu hồi, những quyền mà người dùng này được cho phép bởi những người dùng khác vẫn còn có hiệu lực. Nói cách khác, nếu hai người dùng khác nhau cấp phát cùng các quyền trên cùng một đối tượng cơ sở dữ liệu cho một

người dùng khác, sau đó người thứ nhất thu hồi lại quyền đã cấp phát thì những quyền mà người dùng thứ hai cấp phát vẫn có hiệu lực.

Ví dụ 4.4.5. Giả sử trong cơ sở dữ liệu ta có 3 người dùng là *A*, *B* và *C*. A và B đều có quyền sử dụng và cấp phát quyền trên bảng R. A thực hiện lệnh sau để cấp phát quyền xem dữ liệu trên bảng R cho C:

```
GRANT SELECT
 ON R TO C
```

và B cấp phát quyền xem và bổ sung dữ liệu trên bảng R cho C bằng câu lệnh:

```
GRANT SELECT, INSERT
ON R TO C
```

Như vậy, C có quyền xem và bổ sung dữ liệu trên bảng R. Bây giờ, nếu B thực hiện lệnh:

```
REVOKE SELECT, INSERT
ON R FROM C
```

Người dùng C sẽ không còn quyền bổ sung dữ liệu trên bảng R nhưng vẫn có thể xem được dữ liệu của bảng này (quyền này do A cấp cho C và vẫn còn hiệu lực).

Nếu ta đã cấp phát quyền cho người dùng nào đó bằng câu lệnh GRANT với tuỳ chọn WITH GRANT OPTION thì khi thu hồi quyền bằng câu lệnh REVOKE phải chỉ định tuỳ chọn CASCADE. Trong trường hợp này, các quyền được chuyển tiếp cho những người dùng khác cũng đồng thời được thu hồi.

Ví dụ 4.4.6. Ta cấp phát cho người dùng A trên bảng R với câu lệnh GRANT như sau:

```
GRANT SELECT
ON R TO A
WITH GRANT OPTION
```

Sau đó người dùng A lại cấp phát cho người dùng B quyền xem dữ liệu trên R với câu lệnh:

```
GRANT SELECT
ON R TO B
```

Nếu muốn thu hồi quyền đã cấp phát cho người dùng A, ta sử dụng câu lệnh REVOKE như sau:

```
REVOKE SELECT
ON NHANVIEN
FROM A CASCADE
```

Câu lệnh trên sẽ đồng thời thu hồi quyền mà A đã cấp cho B và như vậy cả A và B đều không thể xem được dữ liệu trên bảng R.

Trong trường hợp cần thu hồi các quyền đã được chuyển tiếp và khả năng chuyển tiếp các quyền đối với những người đã được cấp phát quyền với tuỳ chọn WITH GRANT OPTION, trong câu lệnh REVOKE ta chỉ định mệnh đề GRANT OPTION FOR.

Ví dụ 4.4.7. Trong ví dụ trên, nếu ta thay câu lệnh:

```
REVOKE SELECT
ON NHANVIEN FROM A CASCADE
```

bởi câu lệnh:

```
REVOKE GRANT OPTION FOR SELECT
ON NHANVIEN
FROM A CASCADE
```

Thì B sẽ không còn quyền xem dữ liệu trên bảng R đồng thời A không thể chuyển tiếp quyền mà ta đã cấp phát cho những người dùng khác (tuy nhiên A vẫn còn quyền xem dữ liệu trên bảng R).

4.4.2.2. Thu hồi quyền thực thi các câu lệnh

Việc thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu (CREATE DATABASE, CREATE TABLE, CREATE VIEW,...) được thực hiện đơn giản với câu lệnh REVOKE có cú pháp:

```
REVOKE ALL | các_câu_lệnh_cần_thu_hồi
FROM danh_sách_người_dùng
```

Ví dụ 4.4.8. Để không cho phép người dùng *thuchanh* thực hiện lệnh CREATE TABLE trên cơ sở dữ liệu, ta sử dụng câu lệnh:

```
REVOKE CREATE TABLE
FROM thuchanh
```

4.4.3. Xây dựng mô hình mã hóa mức ứng dụng nhờ khung nhìn để bảo mật dữ liệu

Một trong những phương pháp bảo mật thông dụng nhất là hình thức mã hóa.

Mã hóa ở mức độ tập tin: Đây là giải pháp đơn giản nhất để bảo vệ dữ liệu trong cơ sở dữ liệu, chống lại sự truy cập trái phép vào các tập tin cơ sở dữ liệu. Tuy nhiên, mã hóa dữ liệu ở mức độ này không cung cấp mức độ bảo mật truy cập đến CSDL ở mức độ bảng (table), cột (column) và dòng (row). Một điểm yếu nữa của giải pháp này là bất cứ ai với quyền truy xuất CSDL đều có thể truy cập vào tất cả dữ liệu trong CSDL. Điều này phát sinh một nguy cơ nghiêm trọng, cho phép các đối tượng với quyền quản trị (admin) truy cập tất cả các dữ liệu nhạy cảm. Thêm vào đó, giải pháp này bị hạn chế vì không cho phép phân quyền khác nhau cho người sử dụng CSDL.

Mã hóa ở mức ứng dụng: Giải pháp này xử lý mã hóa dữ liệu trước khi truyền dữ liệu vào CSDL. Những vấn đề về quản lý khóa và quyền truy cập được hỗ trợ bởi ứng dụng. Truy vấn dữ liệu đến CSDL sẽ trả kết quả dữ liệu ở dạng mã hóa và dữ liệu này sẽ được giải mã bởi ứng dụng. Giải pháp này giải quyết được vấn đề phân tách quyền an ninh và hỗ trợ các chính sách an ninh dựa trên vai trò (Role Based Access Control – RBAC). Tuy nhiên, xử lý mã hóa trên tầng ứng dụng đòi hỏi sự thay đổi toàn diện kiến trúc của ứng dụng, thậm chí đòi hỏi ứng dụng phải được viết lại. Đây là một vấn đề đáng kể cho các công ty có nhiều ứng dụng chạy trên nhiều nền CSDL khác nhau.

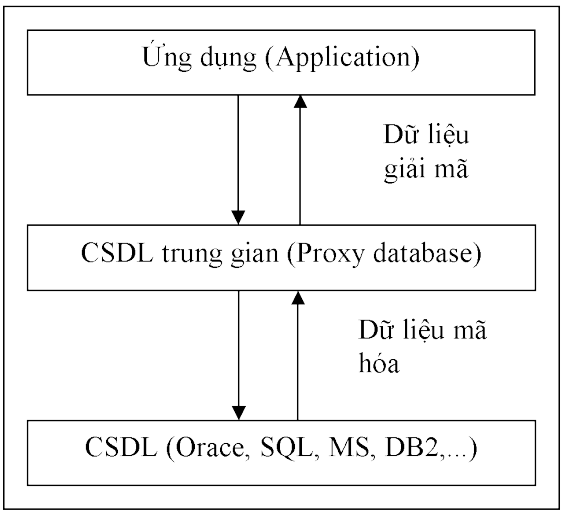
Bên cạnh đó một giải pháp bảo mật CSDL tối ưu cần hỗ trợ các yếu tố chính sau:

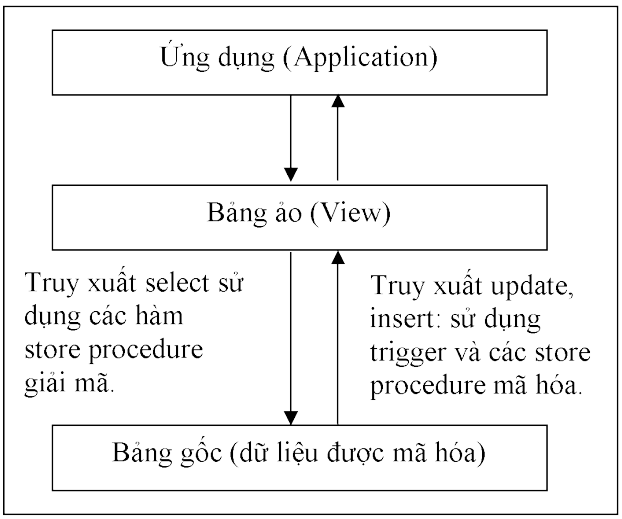
- Hỗ trợ mã hóa tại các mức dữ liệu cấp bảng, cột, hàng.

- Hỗ trợ chính sách an ninh phân quyền truy cập đến mức dữ liệu cột, hỗ trợ RBAC.

- Cơ chế mã hóa không ảnh hưởng đến các ứng dụng hiện tại.

Để đảm bảo các yếu tố trên một số mô hình được xây dựng như sau:





Một số mô hình xây dựng tầng cơ sở dữ liệu trung gian nhờ khung nhìn

Tóm tắt chương 4

* 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 định nghĩa dữ liệu*, *Ngôn ngữ thao tác dữ liệu*, *Ngôn ngữ truy vấn dữ liệu*, *Ngôn ngữ điều khiển dữ liệu*.

* Câu truy vấn trong SQL là phép ánh xạ được miêu tả như một khối SELECT - FROM – WHERE.

Cú pháp chung của câu lệnh SELECT có dạng:

```
SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn
[INTO tên_bảng_mới]
FROM  danh_sách_bảng/khung_nhìn
[WHERE     điều_kiện]
[GROUP BY  danh_sách_cột]
[HAVING điều_kiện]
[ORDER BY  cột_sắp_xếp]
[COMPUTE  danh_sách_hàm_gộp [BY danh_sách_cột]]
```

* Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

- CREATE: định nghĩa và tạo mới đối tượng CSDL.
- ALTER: thay đổi định nghĩa của đối tượng CSDL.
- DROP: Xoá đối tượng CSDL đã có.

* Các khả năng bảo mật cơ sở dữ liệu trong SQL gồm những chức năng như:

- Cấp phát quyền truy cập cơ sở dữ liệu cho người dùng và các nhóm người dùng, phát hiện và ngăn chặn những thao tác trái phép của người sử dụng trên cơ sở dữ liệu.
- Cấp phát quyền sử dụng các câu lệnh, các đối tượng cơ sở dữ liệu đối với người dùng.
- Thu hồi (huỷ bỏ) quyền của người dùng.

Câu hỏi ôn tập và bài tập chương 4

1. Hãy xem xét tính đầy đủ của SQL, tức SQL có thể thực hiện được tất cả các biểu thức của các phép toán đại số quan hệ (chỉ cần xem xét việc thực hiện các phép toán thỏa tính đầy đủ của đại số quan hệ.

2. Cho cơ sở dữ liệu về cung cấp hàng hoá, gồm các quan hệ sau:

CC(MSNCC, TEN, DCCC) và MH(MSNCC,MSMH,SL)

Trong đó:

MSNCC là mã số người cung cấp; TEN là tên người cung cấp

DCCC là địa chỉ người cung cấp; MSMH là mã số mặt hàng

và SL là số lượng đã cung cấp

Hãy viết các biểu thức đại số quan hệ và các lệnh trong SQL để thực hiện các yêu cầu sau:

Mặt hàng có mã số MSMH = 'A1' xuất phát từ địa chỉ nào?

Mặt hàng có mã số MSMH = 'A2' đã được những người nào cung cấp?

Danh sách tên của những người đã cung ứng hàng hoá

Danh sách địa chỉ của những người đã cung ứng hàng hoá

Danh sách những người đã

+ Cung cấp ít nhất một mặt hàng

+ Không cung cấp mặt hàng nào

+ Cung cấp mặt hàng có mã số 15

+ Cung cấp ít nhất một mặt hàng nhưng không có mặt hàng có mã số 10

+ Cung cấp tất cả các mặt hàng

+ Các mặt hàng có mã số là 11 và 12 xuất phát từ địa chỉ nào.

3. Cho cơ sở dữliệu

S(S#, SNAME, CITY); P(P#, PNAME, COLOR); SP(S#, P#, SLUONG)

Trình bày các yêu cầu sau bằng đại số quan hệ và SQL

Mặt hàng có mã số P# = 'P7' xuất phát từ thành phố nào?

Mặt hàng có mã số P# = 'P8' đã được những hãng nào cung cấp?

Tổng số lượng theo từng loại hàng hoá đã cung ứng

Các mặt hàng đã được cung ứng xuất phat từ thành phố nào?

Danh sách tên của những hãng đã cung ứng hàng hoá

Danh sách địa chỉ của những hãng đã cung ứng hàng hoá

4. Tạo CSDL quản lý các sinh viên thi vào trường đại học của bạn, các thuộc tính gồm : mã số (số báo danh), tên, năm sinh, quê, điểm thi, ngành thi.

a) In danh sách các thí sinh đỗ vào trường (điểm thi >= điểm chuẩn) và thống kê số lượng thí sinh trúng tuyển theo từng ngành thi.

b) Danh sách những sinh viên ở Huế.

c) Giả sử tại Bộ Giáo dục có kết quả thi của 5 trường đại học gửi về theo từng danh sách nói trên. Hãy thực hiện các yêu cầu sau:

+ Làm bảng tổng kết gồm có: Số thứ tự, tên trường, số học sinh đậu vào trường, điểm chuẩn.

+ Điểm số trung bình của thí sinh thi vào từng trường

+ Tính phần trăm theo từng loại cho thí sinh trúng tuyển theo từng trường.

5. Cho CSDL gồm các quan hệ sau:

DAIHOC(TENTRUONG, HIEUTRUONG,DIACHI)

KHOA(TENTRUONG, MSKHOA, TENKHOA LOP)

SINHVIEN(TENTRUONG, MSKHOA, MSSV, TENSV,DIACHI)

Bằng đại số quan hệ và ngôn ngữ SQL biểu diễn các câu hỏi sau:

a. Trường nào có môn Tin học.

b. Tổng số sinh viên học Văn trong tất cả các trường đại học.

c. Sinh viên nào học tại quê nhà

d. Khoa nào từng có số sinh viên cao nhất

6. Cho CSDL gồm 2 quan hệ.

CC(MASONCC, TENCC, DCCC); MH(MASONCC,MSMH)

a. Tìm mã số người cung cấp đã cung cấp

+ Ít nhất 1 mặt hàng (đã có tham gia bán)

+ Không cung cấp mặt hàng nào.

+ Cung cấp mặt hàng có MSMH là 15

+ Cung cấp ít nhất 1 mặt hàng nhưng không là mặt hàng 15

b. Mặt hàng có mã số 11, 12, 13 được cung cấp bởi nhà cung cấp ở địa chỉ nào.

7. Thực hiện các câu hỏi truy vấn dữ liệu bằng đại số quan hệ trong phần bài tập của chương 3, bằng câu lệnh SQL.

8. Thực hiện các câu lện SQL ở các bài tập ở trên bằng một hệ quản trị cơ sở dữ liệu cụ thể như FOXPRO, ACCESS hay SQL Server.

Chương 5. RÀNG BUỘC TOÀN VẸN

Mục đích

Trình bày ràng buộc toàn vẹn và toàn vẹn dữ liệu.

Yêu cầu

Hiểu được ràng buộc toàn vẹn, ý nghĩa của ràng buộc toàn vẹn.

Nắm các yếu tố của ràng buộc toàn vẹn và phân loại ràng buộc toàn vẹn.

Trong chương này chúng ta trình bày một vấn đề quan trọng trong phân tích - thiết kế và khai thác cơ sở dữ liệu: Ràng buộc toàn vẹn và kiểm tra ràng buộc toàn vẹn. Ràng buộc toàn vẹn cần được quan tâm đúng mức nhằm tăng tính ngữ nghĩa của cơ sở dữ liệu, thể hiện rõ mối liên hệ giữa các tập thực thể trong mô hình dữ liệu. Đồng thời khi phân tích rõ các ràng buộc toàn vẹn ta đã chỉ ra các yêu cầu quản lý dữ liệu của bài toán thực tế đặt ra, cũng như nhằm giảm thiểu các hậu quả nghiêm trọng khi lưu trữ và xử lý dữ liệu.

5.1. Định nghĩa ràng buộc toàn vẹn

Trong một CSDL luôn luôn có nhiều mối liên hệ, nhiều sự ràng buộc qua lại giữa các thuộc tính, giữa các bộ với nhau. Các mối liên hệ ràng buộc 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 bất kỳ thời điểm nào. Những điều kiện bất biến đó được gọi là "ràng buộc toàn vẹn".

Trong thực tế, ràng buộc toàn vẹn thường là các qui tắc quản lí được áp đặt lên trên các đối tượng của thế giới thực. Việc xác định rõ ràng và đầy đủ các ràng buộc toàn vẹn trên cơ sở dữ liệu không những nâng cao tính ngữ nghĩa của cơ sở dữ liệu mà còn là những ràng buộc, yêu cầu để phân tích thiết kế hệ thống thông tin. Nhằm bảo đảm tính kết dính của các thành phần cấu tạo nên CSDL, tính nhất quán của dữ liệu và làm cho CSDL luôn biểu diễn đúng ngữ nghĩa thực tế.

Ví dụ: Cho CSDL quản lý sinh viên, gồm các lược đồ quan hệ sau.

1. Sinh viên (MASV, HOSV, TENSV, NGSINH , MAKH, HBONG)
2. Kết quả (MASV, MAMH, LANTHI, DIEM).
3. Khoa (MAKH, TENKH, DIACHI).

Trong CSDL này, chúng ta có một số ràng buộc toàn vẹn sau:

R_1: Mỗi sinh viên có một mã số (MASV) riêng biệt không trùng với bất kì với một sinh viên nào khác.

R_2: Mỗi sinh viên phải đăng kí vào một khoa của một trường nhất định.

R_3: Mỗi sinh viên chỉ được thi tối hai lần cho một môn học...

Công việc kiểm tra ràng buộc toàn vẹn có thể tiến hành vào một trong các thời điểm sau:

+ Kiểm tra ngay sau khi thực hiện một thao tác cập nhập CSDL (thêm, sửa, huỷ,...). Thao tác cập nhập được xem là hợp lệ nếu như nó không vi phạm bất cứ một ràng buộc toàn vẹn nào (nghĩa là không làm cho CSDL rơi vào tình trạng mất tính kết dính).

+ Kiểm tra định kì hay đột xuất. Đối với những trường hợp vi phạm ràng buộc toàn vẹn hệ thống sẽ có xử lí ngầm hoặc yêu cầu người sử dụng xử lí những sai sót một cách tường minh.

5.2. Các yếu tố của ràng buộc toàn vẹn

Một ràng buộc toàn vẹn có 3 yếu tố:

+ Nội dung.

+ Bối cảnh.

+ Tầm ảnh hưởng.

5.2.1. Nội dung

Nội dung của ràng buộc toàn vẹn có thể được biểu diễn bằng: ngôn ngữ tự nhiên, thuật giải (mã giả, ngôn ngữ tựa lập trình), đại số quan hệ, phụ thuộc hàm...

Ví dụ 1

Với ràng buộc biểu diễn bằng ngôn ngữ tự nhiên:

"Mức lương của một người nhân viên không được vượt quá lương của trưởng phòng", có biểu diễn bằng ngôn ngữ hình thức như sau:

$\forall t \in$ NHANVIEN (

$\exists u \in$ PHONGBAN ($\exists v \in$ NHANVIEN (

u.TRPHG = v.MANV $\wedge$

u.MAPHG = t.PHG $\wedge$

t.LUONG ≤ v.LUONG)))

Ví dụ 2. Trong lược đồ CSDL $T_{sinhviên}$.

R_1: ∀ SV1 ∈ $T_{sinh\ viên}$, ∀ SV2 ∈ $T_{sinh\ viên}$.

SV1 <> SV2 ⇒ SV1.MASV <> SV2.MASV.

R_2: $T_{sinh\ viên}$ [MAKH] ⊆ T_{Khoa}[MAKH].

R_3: ∀ SV ∈ $T_{kết\ quả}$.

Card ({k ∈ $T_{kết\ quả}$ | K.MASV = SV.MASV}) ≤ 2. (Card(A) = lực lượng của tập A).

Việc biểu diễn nội dung của ràng buộc liên quan đến vấn đề đặc tả, xin xem thêm các sách về đặc tả.

5.2.2. Bối cảnh

Bối cảnh của ràng buộc toàn vẹn là những quan hệ mà ràng buộc toàn vẹn có hiệu lực, nói cách khác đó là những quan hệ cần sử dụng để kiểm tra ràng buộc toàn vẹn. Bối cảnh của ràng buộc toàn vẹn có thể là một hoặc nhiều quan hệ.

Ví dụ Bối cảnh của ràng buộc toàn vẹn trong R_3 chỉ là một quan hệ $T_{kết\ quả}$ (mỗi sinh viên (MASV) thì tối đa hai lần (LANTHI) thì hai thuộc tính MASV, LANTHI ∈ $T_{kết\ quả}$). Nhưng ở R_2, bối cảnh của ràng buộc toàn vẹn phải là hai quan hệ (muốn biết sinh viên có MASV thuộc khoa nào thì $T_{sinh\ viên}$ [MAKH]= T_{khoa}[MAKH] từ đó tìm ra tên khoa.

5.2.3. Tầm ảnh hưởng

Trong quá trình thiết kế CSDL cần chỉ ra các ảnh hưởng của các thao tác trên dữ liệu có ảnh hưởng như thế nào đến các ràng buộc toàn vẹn, các ràng buộc nào bị vi phạm. Bảng tầm ảnh hưởng là một trong số các cách để thể hiện việc ấy.

Ví dụ. Ràng buộc R_i với các thao tác thêm, sửa, xoá và bối cảnh là các quan hệ T_{Qi1}, T_{Qi2} ,..., T_{Qik} ta thành lập bảng tầm ảnh hưởng sau

R_I	Thêm	Sửa	Xoá
T_{Qi1}	+	+	-
T_{Qi2}	-	-	+
.....			
T_{Qik}	+	-	-

Trong đó: dấu "+" được hiểu là cần kiểm tra ràng buộc toàn vẹn R_i.
dấu "-" được hiểu là không cần kiểm tra ràng buộc toàn vẹn R_i.

Từ bảng tầm ảnh hưởng tổng quát ta lập bảng tầm ảnh hưởng cho các quan hệ R_1, R_2, R_3 trong ví dụ 1.

R_I	Thêm	Sửa	Xoá
$T_{sinh\ viên}$	+	-(*)	-

(*) Qui ước là không được phép sửa đổi giá trị thuộc tính MASV.

R_I	Thêm	Sửa	Xoá
$T_{sinh\ viên}$	+	+	-
T_{Khoa}	-	-(*)	+

(*) Qui ước là không được sửa đổi giá trị thuộc tính MAKH.

R_I	Thêm	Sửa	Xoá
$T_{kết\ quả}$	+	-(*)	-

(*) Qui ước là không được sửa đổi giá trị thuộc tính MASV, MAMH, LANTHI.

Ta có "bảng tầm ảnh hưởng tổng hợp" cho CSDL T như sau.

	$T_{sinh\ viên}$			T_{Khoa}			$T_{kết\ quả}$		
	Thêm	Sửa	Xoá	Thêm	Sửa	Xoá	Thêm	Sửa	Xoá
R_1	+	-	-						
R_2	+	+	-	-	-	+			
R_3							+	-	-

5.3. Phân loại ràng buộc toàn vẹn

Trước hết phải khẳng định rằng không thể xác định được hết tất cả các RBTV trên một CSDL, do đó việc phân loại dưới đây chỉ nhằm đưa ra một cơ sở để nhận diện các RBTV có thể trên CSDL.

Ràng buộc toàn vẹn có thể chia làm hai loại chính:

+ Ràng buộc toàn vẹn có bối cảnh là một quan hệ.

+ Ràng buộc toàn vẹn có bối cảnh gồm nhiều quan hệ.

5.3.1. Ràng buộc toàn vẹn có bối cảnh là một quan hệ

5.3.1.1. Ràng buộc toàn vẹn về miền giá trị

Những ràng buộc về miền giá trị của các thuộc tính xuất phát từ cơ cở thực tế như về kiểu, độ rộng, miền giá trị của thuộc tính. Những ràng buộc này đảm bảo rằng giá trị được lưu giữ trong 1 cột phải nằm trong một miền trị hợp lệ được xác định trước. Trong SQL ràng buộc miền trị được hiện thực bằng ràng buộc CHECK.

Ví dụ 1: Trong lược đồ quan hệ $T_{\text{kết quả}}$, ta có: MGT[DIEM]=[0..10].

Ngoài ra, Phòng đào tạo có thể quy định thêm một số ràng buộc khác là điểm thi có độ chính xác đến $0,5^d$. Khi đó ta có $\forall$ kq $\in T_{\text{kết quả}}$. ((kq .DIEM*4) mod 2)=0.

Ví dụ 2: Trên quan hệ NHANVIEN(MANV, TENNV, LUONG, TAMUNG, CONLAI). Ràng buộc về miền giá trị là: TAMUNG ≤ LUONG.

Ví dụ 3: Trong lược đồ quan hệ CONGNHAN(MACN, MUCLG, NGXEPLG). Khi một công nhân thay đổi mức lương, mức lương đó có thể tăng hoặc giảm nhưng rõ ràng thuộc tính NGXEPLG bắt buộc phải tăng theo thời gian. Nếu gọi S là phép sửa đổi lương(NGXEPLG và MUCLG) của các công nhân .

$T_{\text{công nhân}}$ là một quan hệ trên lược đồ quan hệ CONGNHAN.

S: $T_{\text{côngnhân}} \rightarrow T_{\text{côngnhân}}$

Ta có $\forall$ cn $\in T_{\text{côngnhân}}$. cn.NGXEPLG ≤ S(cn).NGXEPLG

5.3.1.2. Ràng buộc liên thuộc tính

Ràng buộc liên thuộc tính là ràng buộc giữa các thuộc tính trong một lược đồ quan hệ.

Ví dụ 1: Cho lược đồ quan hệ HOADON, ta có một ràng buộc toàn vẹn liên thuộc tính như sau:

"Hàng hoá chỉ được xuất kho sau khi đã được lập hoá đơn"

$\forall$ hd $\in T_{\text{HOADON}}$

hd.NGAYHD ≤ hd.NGAYXUAT (*) (thứ tự thời gian).

Nếu giá trị của một thuộc tính A (thuộc tính dẫn xuất) được tính toán từ các thuộc tính khác trong cùng một lược đồ quan hệ với A thì ta cũng có một ràng

buộc liên thuộc tính nhưng trong quá trình lưu trữ chúng ta có thể loại bỏ thuộc tính A ra khỏi lược đồ quan hệ.

Ví dụ 2: Trong lược đồ quan hệ CTIETHD nếu bổ sung thêm thuộc tính THANHTIEN thì ta sẽ có một ràng buộc toàn vẹn liên thuộc tính.

$\forall$ cthd $\in T_{CTIETHD}$

cthd.THANHTIEN = cthd.SLBAN*cthd.GIABAN.

nhưng nếu thiết kế lược đồ quan hệ như trên sẽ làm tăng chi phí lưu trữ và chi phí kiểm tra ràng buộc toàn vẹn một cách vô ích vì thế nên loại bỏ thuộc tính trên.

5.3.1.3. Ràng buộc toàn vẹn liên bộ (ràng buộc thực thể)

Sự tồn tại của một hay nhiều bộ phụ thuộc vào sự tồn tại của một hay nhiều bộ khác trong cùng quan hệ. Các trường hợp đặc biệt là ràng buộc khóa chính (PRIMARY KEY) và ràng buộc duy nhất (UNIQUE). Đây là loại ràng buộc toàn vẹn rất phổ biến, nó có mặt trong mọi lược đồ quan hệ của CSDL và thường được các hệ quản trị CSDL tự động kiểm tra.

Ví dụ: Với r là một quan hệ của Khach ta có ràng buộc toàn vẹn sau

R_1: $\forall t_1, t_2 \in r, t_1.MAKH \neq t_2.MAKH$

R_1	Thêm	Sửa	Xóa
R	+	+	-

5.3.2. Ràng buộc toàn vẹn có bối cảnh gồm nhiều mối quan hệ

5.3.2.1. Ràng buộc toàn vẹn về phụ thuộc tồn tại (ràng buộc tham chiếu)

Giá trị xuất hiện tại các thuộc tính trong một quan hệ nào đó phải tham chiếu đến giá trị khóa chính của một quan hệ khác cho trước. Một trường hợp đặc biệt là ràng buộc khóa ngoại. RBTV tham chiếu còn gọi là phụ thuộc tồn tại thường có bối cảnh là hai quan hệ, nhưng có trường hợp suy biến thành một quan hệ.

Ví dụ 1:

Giả sử có cơ sở dữ liệu gồm các quan hệ sau:

Sinh viên: (1)

Mã số sinh viên	Tên sinh viên	Mã khoa
101	Lê Anh	M01

102	Nguyễn Hoài	P01
103	Vũ Tuấn	G01

Khoa (2)

Mã khoa	Tên khoa
M01	Information
P01	History
G01	Mathematics
F01	Fine Arts

Ở Mã số sinh viên, Tên sinh viên và mã khoa của sinh viên là khoá ngoại chỉ đến bảng Khoa chứa mã khoa và tên khoa. Bảng (2) là bảng được truy xuất qua khoá ngoại được xem như là bảng "cha", bảng (1) tham chiếu đến bảng khác qua khoá ngoại được gọi là bảng "phụ thuộc".

Nếu chúng ta muốn thêm một sinh viên vào bảng (1).

104	Nguyễn Tuấn Long	C01

Chúng ta có một khoá ngoại C01 không tham chiếu đến một khoá nào trong bảng (2). Chúng ta đã vi phạm ràng buộc toàn vẹn qua phép chèn trong bảng phụ thuộc.

Nếu chúng ta xoá một trường trong bảng "cha" chẳng hạn:

M01	Information

Một lần nữa ta có khoá ngoại M01 trong bảng (1) nhưng không còn tham chiếu đến một khoá nào trong bảng "cha" như vậy chúng ta đã vi phạm ràng buộc tồn tại.

Ví dụ 2: Trong quan hệ $T_{kết\ quả}$. Sự tồn tại của bộ KQ = (MASV1, MON1, LANTHI, DIEM1) $\in T_{kết\ quả}$ là hoàn toàn phụ thuộc vào sự tồn tại của bộ sv $\in T_{sinh\ viên}$ sao cho: sv.MASV = MASV1, Ví dụ: sv = (MASV1, "Tran Hue", "Chi", False, 01/04/79, "B1 KTX Đống Đa", "Tin học", "DHSP").

Nếu không tồn tại một bộ sv $\in T_{sinh\ viên}$ Sao cho: sv.MASV = MASV1 thì bộ kq = (MASV1, MON1, LANTHI, DIEM1) không được phép tồn tại trong $T_{kết\ quả}$.

Ví dụ 3: Trong quan hệ $T_{\text{sinh viên}}$ của một tình trạng của cơ sở dữ liệu T, sự tồn tại của bộ sv = ("6760", "MASV1", "Tran Hue", "Chi", False, 01/04/79, "B1 KTX Đống Đa", "Tin học", "DHSP") $\in T_{\text{sinh viên}}$ là hoàn toàn phụ thuộc vào sự tồn tại của bộ không thuộc T_{khoa} : k = ("Tin Học", " khoa Tin Học").

Nếu không tồn tại một bộ k$\in T_{\text{khoa}}$ sao cho k.MAKH = "Tin học" thì bộ sv như trên không được phép tồn tại trong $T_{\text{sinh viên}}$.

Đối với bảng "cha" phép chèn không bị hạn chế, tuy nhiên khi một hàng bị xoá ra khỏi bảng "cha", quy tắc cập nhật đồng thời (CASCADE) buộc hệ QTCSDL phải xoá tất cả các hàng trong bảng phụ thuộc có tham chiếu đến hàng bị xoá trong bảng cha, còn quy tắc SET NULL đặt khoá ngoại tương ứng trong bảng phụ thuộc thành NULL và một quy tắc khác (RESTRICT) cấm xoá các hàng trong bảng cha.

Đối với bảng phụ thuộc, phép xoá không bị hạn chế, nhưng khi một hàng đã được chèn vào bảng phụ thuộc thì tất cả các giá trị trong khoá ngoại của hàng đó phải là giá trị NULL hoặc phải có trong bảng cha tương ứng và khi cập nhật bảng phụ thuộc giá trị cập nhật phải là giá trị NULL hoặc phải có trong một hàng của bảng "cha" tương ứng.

5.3.2.2. Ràng buộc toàn vẹn về liên bộ liên quan hệ

Ràng buộc này có tác dụng đối với từng nhóm các bộ của nhiều quan hệ khác nhau.

Ví dụ:

HOADON(SOHD, MAKH, NGAYHD)

CTHD(SOHD, MAHH, DGIA, SLG)

Với RBTV "mỗi hóa đơn phải có ít nhất một chi tiết hóa đơn"

5.3.2.3. Ràng buộc toàn vẹn liên thuộc tính - liên quan hệ

Là mối liên hệ giữa các thuộc tính trong nhiều lược đồ quan hệ.

Ví dụ: Xét hai quan hệ sau : $T_{\text{đặt hàng}}$, $T_{\text{hoá đơn}}$

$\forall$ dh $\in T_{\text{đặt hàng}}$

$\forall$ hd $\in T_{\text{hoá đơn}}$: hd. SOHD = dh.SODH,

dh.NGAYDH $\leq$ hd.NGAYHD

Cuối $\forall$ dh

5.3.2.4. Ràng buộc toàn vẹn về thuộc tính tổng hợp

Thuộc tính A của lược đồ quan hệ Q được tính toán giá trị từ các thuộc tính của các lược đồ quan hệ khác.

Ví dụ: Tiền nợ của khách hàng bằng ∑ giá trị của các hoá đơn bán cho A - ∑ tiền thu của A.

$\forall$ kh $\in T_{Khách}$.

kh.congno = ∑hd.trigia_hd - ∑pt.SOTIEN.

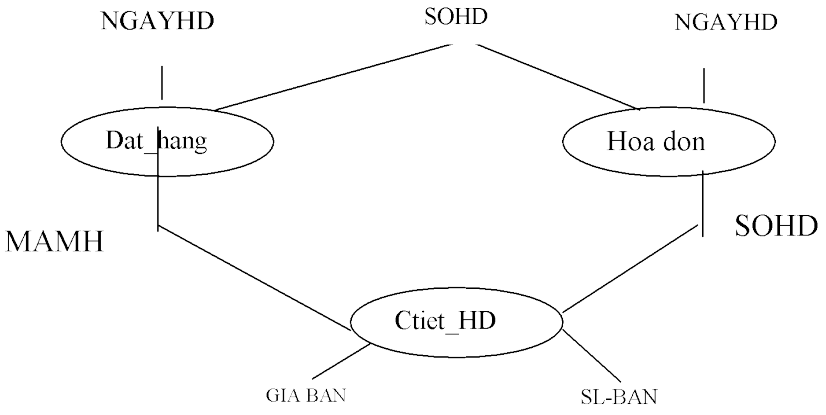
hd $\in H_{kh}$; pt $\in P_{kh}$

Trong đó $H_{kh} = (T_{hoá\ đơn} \bowtie T_{đặt\ hàng})$:(MAKH = kh.MAKH) và $P_{kh} = T_{phiếu\ thu}$: (MAKH = kh.MAKH)

5.3.2.5. Ràng buộc toàn vẹn có chu trình trong đồ thị biểu diễn của lược đồ cơ sở dữ liệu

Một lược đồ cơ sở dữ liệu có thể biểu diễn bằng một đồ thị vô hướng. Trong đồ thị này có hai loại nút: nút thuộc tính và nút lược đồ quan hệ. Một cung vô hướng trong đồ thị nối một nút thuộc tính A với một nút lược đồ quan hệ Q có nghĩa: A $\in Q^+$.

Ví dụ: Xét một phần của đồ thị biểu diễn lược đồ cơ sở dữ liệu T gồm các lược đồ quan hệ Đat_hang, Hoa_don và Ctiet_hd.



Trong trường hợp trên, ta thấy đồ thị biểu diễn có chứa một chu trình giữa 3 nút Đat_hang, Hoa_don và Ctiet_hd. Lúc này xảy ra một ràng buộc toàn vẹn giữa các thuộc tính MAMH, SOHD trong cả ba quan hệ và hiện tượng chờ đợi để xác

định thuộc tính, chờ đợi để phụ thuộc tồn tại. Vì vậy mà trong một số hệ quản trị CSDL như FOXPRO chẳng hạn không cho phép đặt quan hệ lòng vòng A với B, B với C và C với A.

5.4. Cài đặt ràng buộc toàn vẹn với SQL

Trong các hệ QTCSDL ngày nay đều có các kỹ thuật để thể hiện các RBTV, trong SQL Server các RBTV được cài đặt bởi các ràng buộc Primary key, Foreign key, Check contraint và các thủ tục Assertion, Trigger và Transaction.

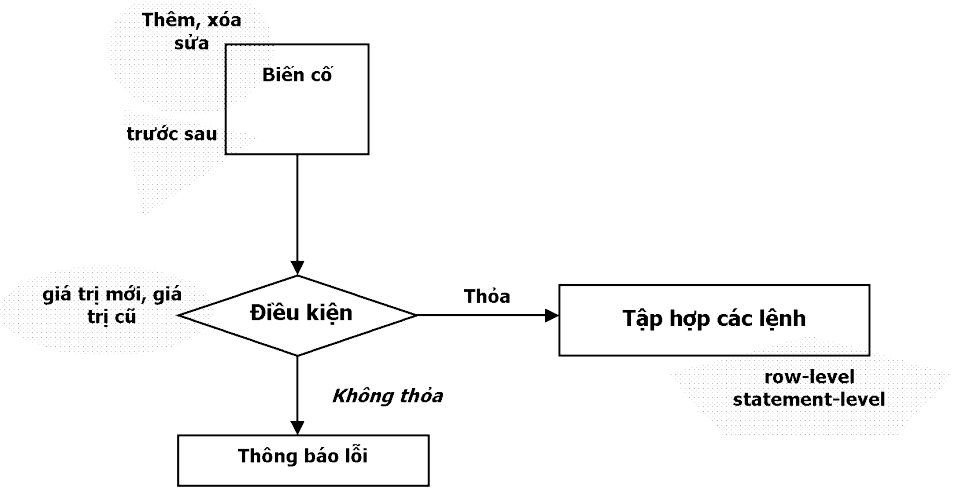
Ở đây trình bày sơ lược về Assertion, Trigger và Transaction, chi tiết về chúng có thể xem thêm trong các tài liệu chuyên khảo về SQL Server.

- Assertion là một biểu thức SQL luôn mang giá trị TRUE tại mọi thời điểm và người sử dụng cần cho biết cái gì phải đúng. Các câu lệnh liên quan trong SQL là:

`CREATE ASSERTION <Tên_assertion> CHECK (<Điều_kiện>)`

Và `DROP ASSERTION <Tên_assertion>`

- Trigger là tập hợp các lệnh được thực hiện tự động khi xuất hiện một biến cố nào đó (tựa như một ngắt (interup) trong hệ điều hành). Một trigger hoạt động theo sơ đồ sau:



Các câu lệnh liên quan trigger trong SQL là:

```
CREATE TRIGGER <Tên_trigger>
AFTER|BEFORE INSERT|UPDATE|DELETE ON   <Tên_bảng>
REFERENCING
     NEW   ROW|TABLE   AS   <Tên_1>
     OLD    ROW|TABLE   AS   <Tên_2>
FOR EACH ROW | FOR EACH STATEMENT
WHEN (<Điều kiện>)
     <Tập_lệnh_SQL>
và DROP TRIGGER <Tên_trigger>
```

- Transaction là tập các lệnh thực hiện một xử lý nào đó trong một ứng dụng CSDL, sao cho:

- Hoặc là tất cả các lệnh đều được thực hiện thành công
- Hoặc là không có lệnh nào được thực hiện

Ví dụ 1 Quá trình xử lý chuyển tiền trong ngân hàng

*Giao tác **Chuyển_tiền***

Giảm tiền trong tài khoản người gửi

Tăng tiền trong tài khoản người nhận

Nếu tất cả đều thành công thì hoàn tất giao tác

Ngược lại quay lui giao tác

Cuối giao tác.

Một giao tác phải đảm bảo:

Tính nguyên tố (atomicity)

Tính nhất quán của CSDL (consistency).

Các RBTV phải đảm bảo không bị vi phạm trong khi thực hiện giao tác, trước và sau khi thực hiện giao tác.

Ví dụ 2 Với RBTV mỗi hóa đơn phải có ít nhất một chi tiết hóa đơn.

*Giao tác **Thêm_hóa_đơn***

Thêm HOADON

Thêm chi tiết thứ 1 vào CTHD

Thêm chi tiết thứ 2 vào CTHD

...

Nếu có một thao tác thêm thất bại thì

Quay lui giao tác

Ngược lại

Hoàn tất giao tác

Cuối nếu

Cuối giao tác.

Các DBMS thương mại cung cấp cách thức lưu trữ các hàm hay thủ tục nhằm thực hiện các giao tác. Các thủ tục này được lưu trữ trong lược đồ CSDL và được sử dụng trong các câu lệnh. Trong SQL cung cấp kỹ thuật ***Stored Procedure*** có cú pháp tổng quát như sau:

```
CREATE PROCEDURE <Tên_thủ_tục> <DS_tham_số>
AS
            Khai báo biến cục bộ
            Thân chương trình
GO
EXEC <Tên_thủ_tục> <DS_ tham_số>
```

Ví dụ 3 Với RBTV "mỗi trận đấu là cuộc thi đấu của đúng 2 đội", ta có giao tác thể hiện việc thêm trận đấu như sau:

Giao tác Thêm_trận_đấu(t, s)

Thêm t vào THIDAU

Thêm s vào THIDAU

Nếu có một thao tác thất bại thì

Quay lui giao tác

Ngược lại

Hoàn tất giao tác

Cuối nếu

Cuối giao tác.

Và thủ tục tương ứng trong SQL sẽ là:

```
CREATE PROCEDURE Thêm_trận_đấu
t THIDAU , s THIDAU
```

```
    AS
begin tran
        Thêm t vào THIDAU
        If @@error<>0 rollback tran

        Thêm s vào THIDAU
        If @@error<>0        rollback tran
commit tran
    GO
    EXEC Thêm_trận_đấu  x, y.
```

Tóm tắt chương 5

* Trong một CSDL luôn luôn có nhiều mối liên hệ, nhiều sự ràng buộc qua lại giữa các thuộc tính, giữa các bộ với nhau. Các mối liên hệ ràng buộc 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 bất kỳ thời điểm nào. Những điều kiện bất biến đó được gọi là "ràng buộc toàn vẹn".

* Việc xác định rõ ràng và đầy đủ các ràng buộc toàn vẹn trên cơ sở dữ liệu không những nâng cao tính ngữ nghĩa của cơ sở dữ liệu mà còn là những ràng buộc, yêu cầu để phân tích thiết kế hệ thống thông tin. Nhằm bảo đảm tính kết dính của các thành phần cấu tạo nên CSDL, tính nhất quán của dữ liệu và làm cho CSDL luôn biểu diễn đúng ngữ nghĩa thực tế.

* Một ràng buộc toàn vẹn có 3 yếu tố: Nội dung, bối cảnh, tầm ảnh hưởng.

* Trước hết phải khẳng định rằng không thể xác định được hết tất cả các RBTV trên một CSDL, do đó việc phân loại trên đây chỉ nhằm đưa ra một cơ sở để nhận diện các RBTV có thể trên CSDL.

* Trong các hệ QTCSDL ngày nay đều có các kỹ thuật để thể hiện các RBTV, trong SQL Server các RBTV được cài đặt bởi các ràng buộc Primary key, Foreign key, Check contraint và các thủ tục Assertion, Trigger và Transaction.

Câu hỏi ôn tập và bài tập chương 5

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 BII6

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.

2. 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. Và các lược đồ quan hệ được chuyển từ mô hình thực thể quan hệ của bài toán là:

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)

Hãy phát biểu các ràng buộc toàn vẹn có trong lược đồ cơ sở dữ liệu trên.

3. Cho lược đồ cơ sở dữ liệu ở bài tập 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 vẹn có trong lược đồ cơ sở dữ liệu trên.

CHƯƠNG 6. PHỤ THUỘC HÀM VÀ KHÓA

Mục đích

- Trình bày một số vấn đề nảy sinh trong thiết kế cơ sở dữ liệu quan hệ.

- Các khái niệm phụ thuộc hàm, các bài toán về khóa.

Yêu cầu

- Nắm được và vận dụng một số kỹ năng trong các bài toán về phụ thuộc hàm, khóa trong lý thuyết thiết kế cơ sở dữ liệu quan hệ.

- Hiểu và khái quát được các ứng dụng của các bài toán về phụ thuộc hàm, phủ phụ thuộc hàm và khóa trong các lĩnh vực của cơ sở dữ liệu.

- Hiểu và vận dụng được ý nghĩa của các bài toán nói trên trong thực tiễn.

Trong một số tài liệu khác, thường trình bày các vấn đề liên quan đến phụ thuộc hàm, khoá, phân tách và chuẩn hoá... thành lý thuyết thiết kế cơ sở dữ liệu quan hệ. Nhưng nếu làm như vậy thì chương này sẽ quá nặng nề. Ở đây chúng tôi chia các vấn đề về lý thuyết thiết kế cơ sở dữ liệu quan hệ nói trên thành 3 chương, chương 6. Phụ thuộc hàm và Khoá, chương 7. Lý thuyết phân tách và chương 8. Chuẩn hoá, để vấn đề được rõ ràng hơn. Các kiến thức tổng quan về một số loại ràng buộc mở rộng trên cơ sở dữ liệu quan hệ như phụ thuộc đa trị, phụ thuộc kết nối được đưa vào chương 9. Tách riêng vấn đề phân tách và chuẩn hóa, theo chúng tôi sẽ giúp ta có nhìn tốt trong khi nghiên cứu cơ sở dữ liệu phân tán.

Chương này sẽ trình bày phụ thuộc hàm như một công cụ toán học trợ giúp cho việc biểu đạt ngữ nghĩa dữ liệu và đảm bảo tính nhất quán của dữ liệu trong cơ sở dữ liệu. Các tính chất của phụ thuộc hàm và các hệ tiên đề cho phụ thuộc hàm được mô tả đầy đủ, trong đó hệ tiên đề Armstrong được sử dụng nhiều hơn cả. Một trong những khái niệm quan trọng của phụ thuộc hàm là bao đóng của tập thuộc tính và các tính chất cơ bản của phép toán lấy bao đóng cũng được trình bày. Tiếp theo giới thiệu các loại phủ phụ thuộc hàm quan trọng nhất đóng vai trò thu gọn các tập phụ thuộc hàm, tạo thuận tiện cho tối ưu hóa các thao tác ngữ nghĩa. Cuối phần này là các vấn đề về khóa của lược đồ, bài toán tìm mọi khóa cũng được đặt ra.

6.1. Các vấn đề thường gặp trong thiết kế cơ sở dữ liệu quan hệ

Để hình dung được những vấn đề thường xảy ra trong thiết kế cơ sở dữ liệu, ta xét một ví dụ như sau:

Ví dụ: Xét lược đồ quan hệ SINHVIEN(HOTEN, NGAYSINH, LOP, MONHOC, DIEM) được sử dụng để biểu diễn thông tin về các sinh viên và điểm thi các mô học của sinh viên. Ta xét quan hệ là thể hiện của lược đồ này như sau:

HOTEN	NGAYSINH	LOP	MONHOC	DIEM
Nam	23/05/1989	Tin SP20	Pascal	6
Châu	11/06/1988	Tin SP20	Pascal	9
Hoa	15/07/1989	Tin SP20	Đại số	10
Nam	23/05/1989	Tin SP20	Đại số	9
Hoa	15/07/1989	Tin SP20	Pascal	5

Trên quan hệ này, ta có thể nhận thấy một số vấn đề tồn tại như sau:

Thông tin về họ tên, ngày sinh và tên lớp của mỗi sinh viên sẽ phải lưu trữ lặp đi lặp lại nhiều lần nếu sinh viên đó có nhiều điểm ứng với nhiều môn học. (Chẳng hạn như ở quan hệ trên, thông tin về sinh viên có tên là *Nam* phải lưu trữ lặp lại hai lần). Điều này gây nên sự *dư thừa về dữ liệu* trong cơ sở dữ liệu. Sự dư thừa này lại dẫn đến các dị thường về dữ liệu như sau:

+ *Dị thường khi cập nhật dữ liệu*: Giả sử ta cần thay đổi thông tin liên quan đến sinh viên có tên là *Nam*. Điều này đòi hỏi phải cập nhật lại tất cả các bộ dữ liệu có liên quan đến sinh viên này. Tuy nhiên, do thông tin về sinh viên này phải lưu trữ lặp đi lặp lại nhiều lần nên có thể còn "sót" một số bản ghi không được cập nhật và dẫn đến sự thiếu nhất quán đối với dữ liệu.

+ *Dị thường khi bổ sung dữ liệu*: Với việc tổ chức lưu trữ thông tin như trên, ta lại không thể bổ sung thêm một bộ dữ liệu lưu trữ thông tin về một sinh viên nào đó nếu như sinh viên đó chưa có điểm thi ít nhất một môn học.

+ *Dị thường khi xoá dữ liệu*: Mỗi khi ta cần xoá toàn bộ điểm thi của một sinh viên nào đó sẽ dẫn đến xoá cả những thông tin về sinh viên đó ra khỏi cơ sở dữ liệu trong khi thực sự đây không phải là điều mà ta mong muốn.

Những vấn đề tồn tại ở trên sẽ không còn nếu ta sử dụng hai lược đồ SINHVIEN(HOTEN, NGAYSINH, TENLOP) và DIEMTHI(HOTEN,

MONHOC, DIEM) thay cho lược đồ đề cập ở trên. Khi đó, thay vì một quan hệ như ở trên, ta lại có hai quan hệ:

HOTEN	NGAYSINH	TENLOP
Nam	23/05/1989	Tin SP20
Châu	11/06/1988	Tin SP20
Hoa	15/07/1989	Tin SP20

HOTEN	MONHOC	DIEM
Nam	Pascal	6
Châu	Pascal	9
Hoa	Đại số	10
Nam	Đại số	9
Hoa	Pascal	5

Tuy nhiên, trong trường hợp này lại nảy sinh những vấn đề khác như mỗi khi muốn biết các thông tin và điểm thi các môn học của sinh viên, ta phải thực hiện phép nối tự nhiên đối với hai quan hệ trên. Tức là phải thực hiện một phép toán phức tạp và tốn kém về chi phí. Một câu hỏi khác được đặt ra là liệu khi thực hiện phép nối như vậy ta có thể có được thông tin chính xác như ban đầu hay không? Nói cách khác, liệu có tốt không khi phải tách các lược đồ quan hệ thành những lược đồ "nhỏ" hơn?

Trong một CSDL quan hệ, các quan hệ được sử dụng để mô hình hoá thế giới thực. Mỗi một quan hệ được sử dụng để tổ chức các thông tin về một tập các thực thể và/hoặc mối quan hệ giữa các tập thực thể. Tức là, mỗi một bộ trong quan hệ diễn tả thông tin về một thực thể và/hoặc mối quan hệ giữa các thực thể thông qua những giá trị cụ thể của các thuộc tính. Tuy nhiên, không phải bất kỳ một tập các bộ dữ liệu nào cũng là một thể hiện đúng của quan hệ.

Ta xét một số ví dụ sau:

Ví dụ: Xét lược đồ quan hệ HOCSINH(HOTEN, TUOI, DIACHI) và tập các bộ dữ liệu như sau:

HOTEN	TUOI	DIACHI
Nam	19	Huế
Hoa	15	Đà Nẵng
Anh	150	Hà Nội

Ta có thể nhận thấy bộ dữ liệu (Anh,150,Hà Nội) không thể là một thể hiện đúng bởi sẽ là không hợp lý nếu có một học sinh có tuổi là 150.

Ví dụ: Xét lược đồ quan hệ XEMAY(SOXE, LOAIXE, CHUXE). Khi đó tập các bộ dữ liệu dưới đây cũng không thể là một thể hiện đúng

SOXE	LOAIXE	CHUXE
75 F1 3103	Dream II	Châu
75 F3 2543	Wave	Tùng
75 F1 3103	Viva 110	Châu

Do thực sự không thể xảy ra trường hợp xe có biển số 75F1 3103 lại thuộc vào hai loại xe khác nhau.

Như vậy, ta có thể nhận thấy trên các quan hệ phải có những ràng buộc đối với giá trị của các bộ dữ liệu nhằm đảm bảo được tính hợp lệ của dữ liệu. Những ràng buộc này được phân làm hai loại:

- Những ràng buộc qui định miền giá trị cho các thành phần của các bộ dữ liệu. (Chẳng hạn bắt buộc tuổi của học sinh phải nằm trong khoảng từ 6 đến 20).

- Những ràng buộc liên quan đến tính bằng nhau giữa các giá trị thuộc tính của các bộ dữ liệu (Chẳng hạn ràng buộc qui định nếu hai bộ dữ liệu nào đó trong quan hệ XEMAY ở trên có cùng số xe thì cũng phải có cùng loại xe và chủ sở hữu). Ràng buộc dữ liệu dạng này được gọi là các *phụ thuộc hàm* và được bàn luận đến trong chương này.

6.2. Phụ thuộc hàm

Định nghĩa 6.1.

Cho tập thuộc tính U. Một *phụ thuộc hàm* (*PTH*) trên U là công thức dạng

$f: X \to Y$; $X, Y \subseteq U$.

Nếu $f: X \to Y$ là một phụ thuộc hàm trên U thì ta nói tập thuộc tính Y *phụ thuộc* vào tập thuộc tính X, hoặc tập thuộc tính X *xác định hàm* tập thuộc tính Y. X là *vế trái* và Y là *vế phải* của PTH. Ta cũng dùng hai toán tử *Left*(f) và *Right*(f) để lấy vế trái và vế phải của PTH f.

Cho quan hệ $R(U)$ và một *PTH* $f: X \to Y$ trên U. Ta nói quan hệ R *thoả PTH* f và viết $R(f)$, nếu hai bộ tuỳ ý trong R giống nhau trên X thì chúng cũng giống nhau trên Y,

$$R(X \to Y) \Leftrightarrow (\forall u, v \in R): (u.X = v.X) \Rightarrow (u.Y = v.Y)$$

Ta dùng ký hiệu $X \nrightarrow Y$ với ý nghĩa tập thuộc tính Y không phụ thuộc hàm vào tập thuộc tính X.

Cho tập *PTH* F trên tập thuộc tính U. Ta nói quan hệ $R(U)$ *thoả tập PTH* F, và viết $R(F)$, nếu R thoả mọi *PTH* trong F,

$R(F) \Leftrightarrow (\forall f \in F)$: $R(f)$

Ví dụ

Cho quan hệ $R(A,B,C,D)$ như sau:

A	B	C	D
A	1	X	2
A	1	Y	2
B	2	X	1
B	2	Y	1

và các PTH: f_1: $A \rightarrow A$; f_2: A→B; f_3: AC→C; f_4: A→D; f_5: D→A; f_6: A→C.

Khi đó các PTH f_1 - f_5 đúng trong R, mặt khác R không thỏa PTH f_6.

Cho trước tập PTH F trên tập thuộc tính U, ký hiệu REL(U), REL2(U) lần lượt là tập toàn thể các quan hệ trên tập thuộc tính U và tập các quan hệ trên tập thuộc tính U có không quá hai bộ, SAT(F) là tập toàn thể các quan hệ trên U thoả tập PTH F, SAT2(U) là tập toàn thể các quan hệ có không quá hai bộ trên U thoả tập PTH F, cụ thể là:

SAT(F) = { R | R∈REL(U), R(F)}

SAT2(F) = { R | R∈REL2(U), R(F)}

Cho tập $\Re$ các quan hệ trên U, ký hiệu FD($\Re$) là tập các PTH trên U thoả trong mọi quan hệ của $\Re$.

6.2.1. Hệ tiên đề Armstrong

6.2.1.1. Bao đóng của tập PTH và hệ tiên đề Armstrong

Cho tập PTH F trên tập thuộc tính U. *Bao đóng* của F, ký hiệu F^+ là tập nhỏ nhất các PTH trên U chứa F và thoả các tính chất *A1-A3* của *hệ tiên đề Armstrong* A^o sau đây:

Với $\forall X, Y, Z \subseteq U$:

A1. *Tính phản xạ*: Nếu $Y \subseteq X$ thì $X \rightarrow Y \in F^+$

A2. *Tính gia tăng*: Nếu $X \rightarrow Y \in F^+$ thì $XZ \rightarrow YZ \in F^+$

A3. *Tính bắc cầu*: Nếu $X \rightarrow Y \in F^+$ và $Y \rightarrow Z \in F^+$ thì $X \rightarrow Z \in F^+$

Chú ý:

Các PTH có vế trái chứa vế phải như mô tả trong (F1) được gọi là PTH tầm thường. Các PTH tầm thường thoả trong mọi quan hệ. Ngoài ra, các quan hệ trên tập thuộc tính U có không quá một bộ thỏa mọi PTH trên U.

6.2.1.2. Suy dẫn theo tiên đề

Ta nói PTH f được *suy dẫn theo tiên đề* từ tập PTH F và ký hiệu là $F \models f$, nếu $f \in F^+$.

$F \models f \Leftrightarrow f \in F^+$

Nói cách khác PTH f được suy dẫn theo tiên đề từ tập PTH F nếu xuất phát từ F, áp dụng các luật A1, A2 và A3 của hệ tiên đề Armstrong sau hữu hạn lần ta sẽ thu được PTH f.

6.2.1.3. Suy dẫn theo quan hệ (suy dẫn logic)

Cho tập PTH F trên tập thuộc tính U và f là một PTH trên U. Ta nói PTH f *được suy dẫn theo quan hệ* (hoặc *suy dẫn logic*) từ tập PTH F và viết $F \vdash f$, nếu mọi quan hệ $R(U)$ thoả F thì R cũng thoả f.

$F \vdash f \Leftrightarrow SAT(F) \subseteq SAT(f)$

Cho tập thuộc tính U và tập PTH F trên U, ta định nghĩa F^* là tập toàn bộ các PTH f trên U được suy dẫn theo quan hệ từ tập PTH F

$F^* = \{ f: X \rightarrow Y \mid X,Y \subseteq U,\ F \vdash f \}$

6.2.1.4. Suy dẫn theo quan hệ có không quá hai bộ

Cho tập PTH F trên tập thuộc tính U và f là một PTH trên U. Ta nói PTH f *được suy dẫn theo quan hệ có không quá hai bộ* từ tập PTH F và viết $F \vdash_2 f$, nếu mọi quan hệ R trong $REL2(U)$ thoả F thì R cũng thoả f.

$F \vdash_2 f \Leftrightarrow SAT2(F) \subseteq SAT2(f)$

Cho tập thuộc tính U và tập PTH F trên U, ta định nghĩa F' là tập toàn bộ các PTH f trên U được suy dẫn theo quan hệ có không quá hai bộ từ tập PTH F

$F' = \{ f: X \rightarrow Y \mid X,Y \subseteq U,\ F \vdash_2 f \}$

6.2.1.5. Một số tính chất của PTH

Cho tập thuộc tính U và các tập phụ thuộc hàm F, G trên U, tập các quan hệ $\Re$ trên U, các quan hệ R và S trên U. Gọi FD(U) là tập các PTH trên tập thuộc tính U. Khi đó:

1. Nếu $F \subseteq G$ thì $SAT(F) \supseteq SAT(G)$
2. $SAT(FG) = SAT(F) \cap SAT(G)$
3. $FD(R \cup S) \subseteq FD(R) \cap FD(S)$
4. $R \subseteq S \Rightarrow FD(R) \supseteq FD(S)$
5. $F \subseteq FD(SAT(F))$
6. $\Re \subseteq SAT(FD(\Re))$
7. $SAT(FD(SAT(F))) = SAT(F)$
8. $FD(SAT(FD(\Re))) = FD(\Re)$

Thí dụ

Chứng tỏ $FD(R \cup S) \subset FD(R) \cap FD(S)$.

$U=AB$; R chứa một bộ duy nhất $u=(a,x)$; S chứa một bộ duy nhất $v=(a,y)$, $x \neq y$. R và S thỏa mọi PTH trên U. Quan hệ $P=R \cup S$ chứa 2 bộ u và v. P không thỏa PTH $A \rightarrow B$.

6.2.1.6. Một số tính chất mở rộng của PTH

Sử dụng ba tiên đề Armstrong ta dễ dàng chứng minh các tính chất A4 - A11 sau đây:

Với mọi tập con X, Y, Z, V của U và với mọi thuộc tính A trong U:

A4. Tính tựa bắc cầu: Nếu X→Y, YZ→V thì XZ→V

A5. Tính phản xạ chặt: X→ X

A6. Mở rộng vế trái và thu hẹp vế phải: Nếu X→Y thì XZ→Y\V

A7. Cộng tính đầy đủ: Nếu X→Y và Z→V thì XZ→YV

A8. Mở rộng vế trái: Nếu X→Y thì XZ→Y

A9. Cộng tính ở vế phải (hợp): Nếu X→Y và X→Z thì X→YZ

A10. Bộ phận ở vế phải (tách): Nếu X→YZ thì X→Y

A11. Tính tích luỹ: Nếu X→YZ, Z→AV thì X→YZA

6.2.1.7. Lược đồ quan hệ

Lược đồ quan hệ (*LĐQH*) là một cặp $p = (U,F)$, trong đó U là tập hữu hạn các thuộc tính, F là tập các *ràng buộc* trên các miền trị của các thuộc tính trong U.

Trong chương này chúng ta chỉ xét một loại ràng buộc PTH và một số biến thể của PTH.

Theo quy ước trên chúng ta hiểu l*ược đồ quan hệ* (*LĐQH*) là một cặp $p = (U,F)$, trong đó U là tập hữu hạn các thuộc tính, F là tập các PTH trên U.

Quy ước

Trong trường hợp không chỉ rõ tập F, ta xem LĐQH chỉ là một tập hữu hạn các thuộc tính U.

Các hệ tiên đề khác cho PTH

Các hệ tiên đề cho PTH sau đây tương đương với hệ tiên đề Armstrong A^o

B^o = {A5, A10, A11}

S^o = {A1, A4}

D^o = {A3, A5, A6, A7}

M^o = {A4, A5, A8}

Mệnh đề 6.1. *Hệ tiên đề Amstrong là đúng.*

Chứng minh

- Tiên đề phản xạ A1 là hiển nhiên.

- Tiên đề tăng trưởng: Với mọi quan hệ r trên R(U) và với hai bộ t_1, t_2 bất kỳ thuộc r, giả sử $t_1[XZ]=t_2[XZ]$, ta phải chứng minh $t_1[YZ]=t_2[YZ]$. Thật vậy từ $t_1[XZ]=t_2[XZ]$, ta có $t_1[X]=t_2[X]$ và $t_1[Z]=t_2[Z]$ (1) .

Từ giả thiết X → Y và (1) ta có $t_1[Y]=t_2[Y]$ (2).

Từ (2) và (1) ta có $t_1[YZ]=t_2[YZ]$. Hay XZ → YZ.

- Tiên đề bắc cầu: Với mọi quan hệ r trên R(U) và với hai bộ t_1, t_2 bất kỳ thuộc r. Do X→ Y, nên với t_1 và t_2 thoả: $t_1[X]=t_2[X]$ ta có $t_1[Y]=t_2[Y]$. Hơn nữa do Y → Z nên từ $t_1[Y]=t_2[Y]$ ta có $t_1[Z]=t_2[Z]$. Tóm lại từ $t_1[X]=t_2[X]$ ta có

$t_1[Z]=t_2[Z]$. hay X → Z.

Bài tập

Hãy chứng minh tính đúng của các tiên đề A4 – A11 và sự tương đương giữa các hệ tiên đề A^0 và các hệ tiên đề vừa nêu trên.

I.2.2. Bao đóng của tập thuộc tính

Định nghĩa 6.2.

Cho tập PTH F trên tập thuộc tính U và một tập con các thuộc tính X trong U. *Bao đóng của tập thuộc tính X*, ký hiệu X^+ là tập thuộc tính

$X^+ = \{A \in U \mid X \to A \in F^+\}$

6.2.2.1. Thuật toán tìm bao đóng của một tập thuộc tính

Cho tập PTH F trên tập thuộc tính U và một tập con các thuộc tính X trong U. Để xác định bao đóng X^+ của tập thuộc tính X ta xây dựng dãy bao nhau $X_0 \subseteq X_1 \subseteq \ldots X_i$ như sau.

Xuất phát: Đặt $X_0=X$,

Với i ≥ 0 ta đặt

$$X_{i+1} = X_i \bigcup_{\substack{L \to R \in F \\ L \subseteq X_i}} R$$

Nếu $X_{i+1} = X_i$ dừng thuật toán và cho kết quả $X^+ = X_i$

```
Algorithm Closure
Input: - Tập PTH F trên U
       - Tập con thuộc tính X của U
Output:   - Y = X+ = {A∈U | X→A∈F+}
Format: Closure(X,F)
Method
  Y:=X;
  repeat
     Z:=Y;
     for each FD L→R in F do
        if L ⊆ Y then
```

```
        Y:=Y∪R;
            endif;
          endfor;
        until Y=Z;
        return Y;
      end Closure.
```

Ví dụ 1

Cho U={ABCDEG}

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

Cho X ={BD} tính X^+

Bước 1: X_o =BD

Bước 2: X_1 = BD ∪ EG = BDEG (sử dụng FD thứ 2)

X_2 = X_1 ∪ C = BDEGC (sử dụng FD thứ 6)

X_3 = X_2 ∪ A = ABDEGC (sử dụng FD thứ 4)

X_4 = X_3 dừng lại

Kết luận: X^+ = {ABDEGC}

Ví dụ 2

Cho U={A BCDEGH}, và F ={B → A; DA → CE; D → H; GH → C;

và AC → D }. Cho X = {BD} Tính X^+ =?

Ta có:

X_o = X = BD

X_1 = BD ∪ A = BDA

X_2 = BDA ∪ CE = BDACE

X_3 = X_2 ∪ H = BDACEH

X_4 = X_3 (dừng lại)

Kết luận: X+ = { BDACEH}

Mệnh đề 6.2.

Thuật toán tính bao đóng của tập các thuộc tính là đúng và hữu hạn dừng.

Chứng minh

Tính hữu hạn: Do U có số thuộc tính là hữu hạn và do F có số phụ thuộc hàm là hữu hạn, nên các bước kết nạp A để xây dựng X_i là hữu hạn dừng. Tức sẽ xảy ra trường hợp $X_i=X_{i+1}$ (nhiều nhất $X_i = U$). Có thể chỉ ra độ phức tạp thuật toán trên là O(np.min{n,p}), với n = Card(U) (lực lượng - số phần tử của U) và p=Card(F).

Tính đúng: Trước hết, ta thấy mỗi thuộc tính được kết nạp vào X_i là thuộc vào X^+. Thật vậy, qui nạp theo i ta có:

Bước cơ sở i=0, với $A \in X$, rõ ràng $X \rightarrow A$ (tính phản xạ) nên $A \in X^+$ (tức $X \subseteq X^+$).

Qui nạp: Với i> 0, giả sử X_i chỉ chứa các thuộc tính của X^+. Giả sử A được kết nạp vào X_{i+1} do A thuộc vào W, $V \rightarrow W$ thuộc F, $V \subseteq X_i$. Do $X_i \subseteq X^+$, nên $X \rightarrow V$. Nhờ tính bắc cầu, ta có $X \rightarrow W$ và vì vậy $X \rightarrow A$. tức $A \in X^+$. Hay $X_{i+1} \subseteq X^+$. Theo giả thiết qui nạp, ta có $X_{i+1} \subseteq X^+$ với mọi i.

Ngược lại, giả sử $A \in X^+$ nhưng A không thuộc tập X_i nào cả. Lưu ý rằng thuật toán dừng lại khi $X_i = X_{i+1}$. Xét X_i thoả $X_i = X_{i+1}$ ở bước 2, và quan hệ r gồm hai bộ như sau

Các thuộc tính X_i	Các thuộc tính thuộc $U \backslash X_i$
11111111111111	11111111111111111111
11111111111111	00000000000000000000

Ta thấy r có hai bộ giống nhau ở các thuộc tính của X^i nhưng khác nhau ở tất cả các thuộc tính khác. Ta khẳng định rằng r thoả F. Thật vậy, gọi $Y \rightarrow Z$ là một FD trong F nhưng không thoả bởi r. Thế thì $Y \subseteq X_i$ và Z không thể là một tập con của X_i. Do đó X_{i+1} không thể bằng X_i như đã giả định (vì Z còn có thể kết nạp thêm vào X_i theo cách làm của thuật toán). Điều này là vô lý với giả thiết của X_i nên ta có: mọi phụ thuộc hàm của F thoả r.

Theo giả thiết $A \in X^+$ nên $X \rightarrow A \in F^+$, theo mệnh đề 8.1. hệ tiên đề Amstrong là đúng, nên $X \rightarrow A \in F^*$. Hơn nữa r thoả mãn F, nên r thoả mãn $X \rightarrow A$. Từ đây ta có $A \in X_i$, vì nếu ngược lại thì hai bộ của r giống nhau trên X ($X \subseteq X_i$) nhưng lại không giống nhau trên A và như thế thì vi phạm $X \rightarrow A$. Tóm lại $X^+ \subseteq X_i$ và ta có $X_i = X^+$. Nói cách khác thuật toán tính đúng ở bước kết luận ta thu được $X^+ = X_i$.

Bài tập

Hãy cài đặt thuật toán tìm bao đóng nói trên.

Quy ước giản lược

Ta thường viết $X \to Y$ thay vì viết $X \to Y \in F^+$ hoặc $F \models X \to Y$.

6.2.3. Bài toán thành viên

Cho tập thuộc tính U, một tập các PTH F trên U và một PTH $f: X \to Y$ trên U. Hỏi rằng $f \in F^+$ (f có phải là thành viên của F^+) *hay không* ?

Định lý 6.1.

$$X \to Y \in F^+ \text{ khi và chỉ khi } Y \subseteq X^+$$

Chứng minh

[=>]. Giả sử $X \to Y \in F^+$, giả sử $Y = A_{i1}, ..., A_{ik}$, theo luật tách ta có $X \to A_{ij} \in F^+$ với $\forall j=1,...,k$. Theo định nghĩa X^+ thì $A_{ij} \in X^+$ với $\forall j=1,...,k$. Do đó $Y \subseteq X^+$.

[<=]. Giả sử $Y \subseteq X^+$, lúc đó $A_{ij} \in X^+$ với $\forall j=1,...,k$. Theo định nghĩa X^+ thì ta có $X \to A_{ij} \in F^+$, theo luật hợp ta có $X \to Y \in F^+$

Thuật toán cho bài toán thành viên

```
Algorithm IsMember
Format: IsMember(f,F)
Input: - Tập PTH F trên U
       - PTH f trên U
Output:   - True f∈F+;
       - False trong trường hợp phủ định.
Method
  IsMember := Right(f) ⊆ Close(Left(f),F);
end IsMember.
```

Nhận xét:

+ Bài toán thành viên vừa phát biểu có ý nghĩa thực tiễn như sau: Trên một lược đồ quan hệ R(U), ta có tập các ràng buộc toàn vẹn dạng phụ thuộc hàm F. Bài toán xác định tất cả các ràng buộc có thể suy luận từ F một cách logic, hay suy luận từ F nhờ các tiên đề của hệ Amstrong - tức bài toán tính F^+ có độ phức tạp là $O(2^m)$, vì với $X \to Y \in F$, m =Card(Y), ta có 2^m-1 tập con Y_i của Y. Hơn

nữa theo tiên đề tách ta có $X \rightarrow Y_i \in F^+$. Như vậy F^+ có ít nhất là 2^m phụ thuộc hàm. Vì vậy rất khó có thể suy hết tất cả các FD suy dẫn từ F (có ai biết hết được điều gì sẽ xảy ra!).

Nhưng với bài toán thành viên, khi có một ràng buộc dạng FD nảy sinh trong thực tế ta có thể kiểm tra ràng buộc này có thể suy luận từ F hay không. Nếu nó được suy dẫn từ F thì không cần bổ sung vào F, nếu ngược lại nó không thể suy dẫn từ F nhưng nó đáp ứng đúng thực tế thì ta phải bổ sung nó vào F thu được F' chẳng hạn, và lúc này ta có một lược đồ quan hệ mới trên U là R'(U,F') đáp ứng đúng thông tin của thực tiễn hơn!

Định lý 6.2. (Tính đầy đủ và xác đáng của hệ tiên đề Armstrong)

$$F^+ = F^*$$

Nói cách khác, suy dẫn theo quan hệ và suy dẫn theo logic là một, tức là

$$F \models f \Leftrightarrow F \vdash f$$

Chứng minh:

- Tính đúng: đã chứng minh ở Mệnh đề 6.1.

- Tính đủ: Cho p=(U,F); $X,Y \subseteq U$. Giả sử $X \rightarrow Y \in F^*$ nhưng $X \rightarrow Y$ không thể suy dẫn từ F nhờ hệ tiên đề Amstrong, tức $X \rightarrow Y \notin F^+$.

Xây dựng quan hệ r (R) như sau:

Các thuộc tính thuộc X^+	Các thuộc tính không thuộc X^+
11111111111111111	11111111111111111 1 1 1 1
11111111111111111	000000000000000000000000

Trước hết ta thấy mọi phụ thuộc hàm trên F là đúng trên r. Thật vậy, giả sử có $V \rightarrow W \in F$ không đúng trên r. Lúc này $V \subseteq X^+$ (nếu không thì hai bộ của r không giống nhau trên V), và W phải có ít nhất một thuộc tính không thuộc về X^+ (vì nếu không thì $V \rightarrow W$ sẽ thoả r, do lúc đó hai bộ của r cũng giống nhau trên W). Gọi A là một thuộc tính của W nhưng $A \notin X^+$. Do $V \subseteq X^+$, theo định lý 8.1. ta có $X \rightarrow V \in F^+$. Hơn nữa do $V \rightarrow W \in F$, nên theo tính bắc cầu ta có $X \rightarrow W \in F^+$. Lại do tính phản xạ nên $W \rightarrow A \in F^+$, vậy theo tính bắc cầu ta có $X \rightarrow A \in F^+$, nhưng theo lập luận trên thì $A \notin X^+$, điều này là vô lý với định lý 6.1. Tóm lại ta có mọi FD của F đều đúng trên r.

Tiếp theo ta chứng minh $X \rightarrow Y$ không đúng trên r. Thật vậy, nếu $X \rightarrow Y$ đúng trên r, lúc này $X \subseteq X^+$ và $Y \subseteq X^+$ (vì nếu không hai bộ của r không thể giống

nhau trên Y và như thế X→Y không đúng trên r ngay). Nhưng theo định lý 6.1 thì X→Y∈F^+, điều này vô lý với giả thiết X→Y không suy diễn được từ F nhờ các tiên đề trong hệ Armstrong. Do đó X→ Y không đúng trên r. Tức X→Y ∉F^*.

Đến đây có thể kết luận rằng khi X→Y ∉F^+ thì X→ Y ∉F^*. Tóm lại ta có F^*=F^+. Điều này khẳng định hệ tiên đề Amstrong là đầy đủ.

Định lý 6.3.

$$F^+ = F^* = F'$$

Nói cách khác, ba loại suy dẫn sau là tương đương

Suy dẫn logic (F ⊨ f) ,

Suy dẫn theo quan hệ (F ⊢ f) , và

Suy dẫn theo quan hệ có không quá hai bộ (F $\vdash_2$ f).

Một số tính chất của bao đóng

Cho LĐQH p=(U,F). Khi đó với mọi tập con các thuộc tính X và Y của U ta có:

- Tính phản xạ: X⊆ X^+
- Tính đồng biến: X⊆ Y ⇒X^+⊆Y^+
- Tính lũy đẳng: $(X^+)^+$=X^+

Ngoài ra, sử dụng ba tính chất nói trên ta có thể chứng minh các tính chất sau đây:

$(XY)^+ \supseteq X^+Y^+$

$(X^+Y)^+ = (XY^+)^+ = (XY)^+$

$X \to Y$ khi và chỉ khi $Y^+ \subseteq X^+$

$X \to X^+$ và $X^+ \to X$

$X^+ = Y^+$ khi và chỉ khi $X \to Y$ và $Y \to X$

Bài tập: Hãy chứng minh các tính chất trên.

6.3. Phủ phụ thuộc hàm

Cho hai tập PTH F và G trên cùng một tập thuộc tính U. Ta nói F suy dẫn ra được G, ký hiệu F ⊨ G, nếu ∀g∈G: F ⊨ g. Ta nói F tương đương với G, ký hiệu F ≡ G, nếu F ⊨G và G ⊨F.

Nếu $F \equiv G$ ta nói G là một *phủ* của F.

Ký hiệu F $\not\equiv$ G có nghĩa F và G không tương đương.

Cho tập PTH F trên tập thuộc tính U và X là tập con của U, ta dùng ký hiệu X_F^+ trong trường hợp cần chỉ rõ bao đóng của tập thuộc tính X lấy theo tập PTH F.

6.3.1. Phủ thu gọn tự nhiên

Cho hai tập PTH F và G trên cùng một tập thuộc tính U. G là *phủ thu gọn tự nhiên* của F nếu:

1) G là một phủ của F, và

2) G có dạng thu gọn tự nhiên theo nghĩa sau:

Hai vế trái và phải của mọi PTH trong G rời nhau (không giao nhau.)

Các vế trái của mọi PTH trong G khác nhau đôi một.

Thuật toán tìm phủ thu gọn tự nhiên

```
Algorithm Natural_Reduced
Format: Natural_Reduced(F)
Input:     - Tập PTH F
Output:    - Một phủ thu gọn tự nhiên G của F
  G ≡ F
  ∀ L→R ∈ G: L∩R=∅
  ∀L_i→R_i, ∀L_j→R_j∈G: i≠j ⇒ L_i≠L_j
Method
  G:=∅;
  for each FD L→R in F do
Z:=R-L;
if Z≠∅ then
        if there is an FD L→Y in G then
replace L→Y in G by L→YZ
  else add L→Z to G;
endif;
endif;
```

```
  endfor;
  return G;
end Natural_Reduced.
```

Nếu dùng kỹ thuật chỉ dẫn để tổ chức truy nhập trực tiếp tới các thuộc tính và PTH thì thuật toán thu gọn tự nhiên đòi hỏi độ phức tạp tính toán là O(mn) trong đó m là số lượng PTH trong tập F, n là số lượng thuộc tính trong U. Để ý rằng O(mn) chính là độ phức tạp tuyến tính theo chiều dài dữ liệu vào.

Mệnh đề 6.3.

Nếu F và G là hai tập PTH trên cùng một tập thuộc tính U thì F≡G khi và chỉ khi $\forall X \subseteq U: X_F^+ = X_G^+$.

Bài tập: Hãy chứng minh mệnh đề trên.

6.3.2. Phủ không dư

Cho hai tập PTH *F* và *G* trên tập thuộc tính *U*. *G* được gọi là *phủ không dư* của *F* nếu

1) G là một phủ của F, và

2) G có dạng không dư: $\forall g \in G: G \setminus \{g\} \neq G$

Thuật toán tìm phủ không dư của tập PTH F

```
Algorithm Nonredundant
Format: Nonredundant(F)
Input: - Tập PTH F
Output:- Một phủ không dư G của F
G ≡ F
∀g ∈ G: G\{g} ≠ G
Method
  G:=F;
  for each FD g in F do
if IsMember(g,G\{g})then
        G:=G\{g};
endif;
  endfor;
```

```
  return G;
end Nonredundant.
```

6.3.3. Phủ thu gọn

Cho hai tập PTH F và G trên tập thuộc tính U. G được gọi là *phủ thu gọn trái* của F nếu:

1) G là một phủ của F, và

2) $(\forall L \to R \in G, \forall A \in L)$: $G \setminus \{L \to R\} \cup \{(L \setminus \{A\}) \to R\} \not\equiv G$

Thuật toán tìm phủ thu gọn trái của tập PTH

Để ý rằng $\forall A \in L$: $L \setminus \{A\} \subseteq L$, nên ta có

$\forall g$: $L \to R \in G, \forall A \in L$: $G \setminus \{g\} \cup \{L \setminus \{A\} \to R\} \models L \to R$,

do đó ta chỉ cần kiểm tra $G \models (L \setminus \{A\}) \to R$.

```
Algorithm Left_Reduced
Format: Left_Reduced(F)
Input:     - Tập PTH F
Output:    - Một phủ thu gọn trái G của F
G ≡ F
∀g:L→R ∈ G,∀A ∈ L: G\{g}∪{L\{A}→R}≠G
Method
  G:=F;
  for each FD g:L → R in F do
     X:=L;
     for each attribute A in X do
        if R ⊆ Closure(L\{A},G)then
           delete A from L in G;
        endif;
     endfor;
  endfor;
  return G;
end Left_Reduced.
```

Cho hai tập PTH F và G trên tập thuộc tính U. G được gọi là *phủ thu gọn phải* của F nếu

1) G là một phủ của F, và

2) $(\forall L \to R \in G,\ \forall A \in R)$: $G \setminus \{L \to R\} \cup \{L \to (R \setminus \{A\})\} \not\equiv G$

Thuật toán tìm phủ thu gọn phải của tập PTH

Để ý rằng, $\forall A \in R$: $R \setminus \{A\} \subseteq R$, nên $\forall g$: $L \to R \in G, \forall A \in R$: $G \models L \to R \setminus \{A\}$

do đó ta chỉ cần kiểm tra $G \setminus \{L \to R\} \cup \{L \to R \setminus \{A\}\} \models L \to R$.

```
Algorithm Right_Reduced
Format: Right_Reduced(F)
Input:     - Tập PTH F
Output:    - Một phủ thu gọn phải G của F
           G ≡ F
           ∀g:L→R ∈ G,∀A ∈ R:G\{g}∪{L→R\{A}}≢ G
Method
  G:=F;
  for each FD g:L → R in F do
     H:=G\{L→R};
        X:=R;
     for each attribute A in X do
        if A in Closure(L,H ∪{L→R\{A}})then
           delete A from R in G;
        endif;
     endfor;
  endfor;
  return G;
end Right_Reduced.
```

Cho hai tập PTH F và G trên tập thuộc tính U. G được gọi là *phủ thu gọn* của F nếu G đồng thời là phủ thu gọn trái và thu gọn phải của F.

Thuật toán tìm phủ thu gọn của tập PTH

```
Algorithm Reduced
```

```
Format: Reduced(F)
Input:     - Tập PTH F
Output:    - Một phủ thu gọn của F
Method
  return Right_Reduced(Left_Reduced(F));
end Reduced.
```

6.3.4. Phủ tối tiểu

Định nghĩa 6.3.

Cho hai tập PTH F và G trên tập thuộc tính U. G được gọi là *phủ tối tiểu* của F nếu:

1) Vế phải của mọi PTH trong G chỉ chứa một thuộc tính.

2) G là một phủ thu gọn trái của F.

3) G là một phủ không dư của F.

Thuật toán tìm phủ tối tiểu của tập PTH

```
Algorithm MinCover
Format: MinCover(F)
Input: - Tập PTH F
Output:    - Một phủ tối tiểu G của F
Method
     // Tách mỗi PTH L→R trong F thành các PTH có
     // vế phải chỉ chứa một thuộc tính  L→A, A ∈ R
  G:=∅;
  for each FD L→R in F do
     for each attribute A in R do
        if L→A not_in G then
           add L→A to G;
        endif;
     endfor;
  endfor;
```

```
  G:=Nonredundant(Left_Reduced(G));
  return G;
end MinCover.
```

Định lý 6.4.

Mỗi tập phụ thuộc hàm F đều tương đương với tập phụ thuộc hàm F' tối thiểu.

Nhận xét

Trong thực tiễn thiết kế dữ liệu, bao giờ ta cũng mong muốn tìm ra tập phụ thuộc hàm tối thiểu F', vấn đề là có tồn tại hay không? và cách tìm F'? định lý trên chỉ tính tồn tại của F'. Các thuật toán nói ở các phần trên chỉ ra cách tìm F'.

Ví dụ 1

Cho tập F={A→B; A→C; B→A; C→A; B→C}

G_1={A→B; C→A; B→C} và G_2={A→B; A→C; B→A; C→A} là hai tập phụ thuộc hàm tối thiểu khác nhau của F.

Ví dụ 2

Cho F={AB→C D→E CG→D
C→A D→G CE→A
BC→D BE→C CE→G
ACD→B CG→B}

Ta thấy CE→A có thể suy ra từ C→A; CG→B có thể suy ra từ CG→D, C→A và ACD→B, còn ACD→B có thể thay bởi CD→B vì C→A.

Nên G_1={AB→C; C→A; BC→D; CD→B; D→E; D→G BE→C; CG→D; CE→G} là một phủ tối thiểu của F.

Nếu loại bỏ CE→A, CG→D và ACD→B sẽ có một phủ tối thiểu khác của F là G_2 = {AB→C; C→A; BC→D; D→E; D→G; BE→C; CG→B; CE→G}.

Lưu ý rằng số các phụ thuộc hàm trong G_1 và G_2 là khác nhau và còn có phủ tối thiểu của F nào khác không?

Nhận xét

+ Cho tập phụ thuộc hàm F, có nhiều phủ tối thiểu G của F. Trong các G như thế - tập G có lực lượng nhỏ nhất được gọi là ***phủ phụ thuộc hàm tối ưu*** của F. Có thể xác định phủ phụ thuộc hàm tối ưu của F bằng cách tìm tất cả các phủ tối thiểu của G, sau đó chọn ra G có lực lượng bé nhất (chọn được vì số các G như

thế là hữu hạn) để có phủ phụ thuộc hàm tối ưu của F. Tuy vậy cách làm trên dễ thấy sẽ có độ phức tạp là $O(2^n.2^p)$, điều này là không khả thi!. Bài toán tìm phủ phụ thuộc hàm tối ưu của F cho đến nay vẫn còn để mở.

+ Trên đây chúng ta nêu ra rất nhiều loại phủ của F, quan trọng nhất là phủ tối thiểu. Nhưng vấn đề là các khái niệm phủ có ý nghĩa gì trong thực tiễn thiết kế CSDL quan hệ!

6.3.5. Một số phụ thuộc dữ liệu mở rộng từ phụ thuộc hàm

Phụ thuộc đầy đủ

Tập thuộc tính $Y \subseteq U$ được gọi là *phụ thuộc đầy đủ* vào tập thuộc tính $X \subseteq U$, và được ký hiệu là $X \twoheadrightarrow Y$ nếu

1) $X \rightarrow Y$, *và*

2) $(\forall A \in X): X \setminus \{A\} \not\twoheadrightarrow Y$

Phụ thuộc bắc cầu

Tập thuộc tính $Y \subseteq U$ được gọi là *phụ thuộc bắc cầu* vào tập thuộc tính $X \subseteq U$, và được ký hiệu là $X \rightsquigarrow Y$ nếu

$(\exists Z \subseteq U): Y \setminus Z \neq \emptyset$, $X \rightarrow Z$, $Z \not\twoheadrightarrow X$, $Z \rightarrow Y$.

Nếu $X \rightarrow Y$ và Y không phụ thuộc bắc cầu vào X thì ta nói Y *phụ thuộc trực tiếp* vào X và ký hiệu là $X \hookrightarrow Y$.

Phụ thuộc mạnh, yếu và đối ngẫu

Cho tập thuộc tính U và hai tập con các thuộc tính $X, Y \subseteq U$.

Quan hệ $R(U)$ thỏa *phụ thuộc mạnh* $X\ (s) \rightarrow Y$ nếu với hai bộ tùy ý u và v trong R giống nhau tại một thuộc tính A nào đó trong X thì hai bộ đó giống nhau trên Y.

$\forall u,v \in R: (\exists A \in X: u.A = v.A \Rightarrow u.Y = v.Y)$

Quan hệ $R(U)$ thỏa *phụ thuộc yếu* $X\ (w) \rightarrow Y$ nếu với hai bộ tùy ý u và v trong R giống nhau trên X thì hai bộ đó giống nhau tại một thuộc tính B nào đó của Y.

$\forall u,v \in R: (u.X = v.X) \Rightarrow (\exists B \in Y: u.B = v.B)$

Quan hệ $R(U)$ thỏa *phụ thuộc đối ngẫu* $X\ (d)\rightarrow Y$ nếu với hai bộ tùy ý u và v trong R giống nhau tại một thuộc tính A nào đó của X thì hai bộ đó giống nhau tại một thuộc tính B nào đó của Y.

$\forall u,v \in R: (\exists A \in X: u.A = v.A) \Rightarrow (\exists B \in Y: u.B = v.B)$

Tính chất

$R(X\ (s)\rightarrow Y) \Rightarrow R(X \rightarrow Y)$

$R(X\ (s)\rightarrow Y) \Rightarrow R(X\ (d)\rightarrow Y)$

$R(X \rightarrow Y) \Rightarrow R(X\ (w)\rightarrow Y)$

$R(X\ (d)\rightarrow Y) \Rightarrow R(X\ (w)\rightarrow Y)$

6.4. Khóa của lược đồ quan hệ

Trước hết chúng ta hãy nhớ lại định nghĩa và ý nghĩa của siêu khóa và khóa ở chương 3. Ở đây chúng ta xem xét khái niệm khóa của lược đồ quan hệ dưới cái nhìn của ràng buộc dữ liệu phụ thuộc hàm và các bài toán có liên quan đến khóa với công cụ phụ thuộc hàm.

Định nghĩa 6.4. (Khóa của lược đồ quan hệ)

Cho LĐQH $p = (U, F)$. Tập thuộc tính $K \subseteq U$ được gọi là *khoá* của LĐQH p nếu

(i). $K^+ = U$

(ii). $\forall A \in K: (K \setminus \{A\})^+ \neq U$

Hai điều kiện trên tương đương với

(i'). $K \rightarrow U$

(ii'). $\forall A \in K: (K \setminus \{A\}) \nrightarrow U$

Nếu K thoả điều kiện (i) (hoặc (i')) thì K được gọi là một *siêu khoá*.

Thuộc tính $A \in U$ được gọi là *thuộc tính khoá* (*nguyên thuỷ* hoặc *cơ sở*) nếu A có trong một khoá nào đấy. A được gọi là thuộc tính *không khoá* (*phi nguyên thuỷ* hoặc *thứ cấp*) nếu A không có trong bất kỳ khoá nào.

Cho LĐQH $p = (U, F)$. Ta ký hiệu U_K là tập các *thuộc tính khóa* của p và U_0 là *tập các thuộc tính không khóa* của p. Dễ thấy $U_K | U_0$ là một phân hoạch của U.

Lưu ý

- Trong một số tài liệu thuật ngữ *khoá* được dùng theo nghĩa *siêu khoá* và thuật ngữ *khoá tối tiểu* được dùng theo nghĩa *khoá.*

Bài tập

+ Hãy so sánh định nghĩa siêu khóa và khóa ở định nghĩa trên và định nghĩa siêu khóa và khóa ở chương 3, để thấy được ý nghĩa của các định nghĩa này.

+ Hãy chỉ ra rằng mọi lược đồ đề có ít nhất 1 siêu khóa và do đó mọi lược đồ đều có khóa.

Thuật toán tìm một khóa của LĐQH

Ý tưởng: Xuất phát từ một siêu khóa M tùy ý của LĐQH, duyệt lần lượt các thuộc tính A của M, nếu bất biến $(M\backslash\{A\})^+ = U$ được bảo toàn thì loại A khỏi M. Có thể thay kiểm *tra* $(M\backslash\{A\})^+ = U$ bằng kiểm tra $A \in (M\backslash\{A\})^+$. Sau quá trình duyệt ta thu được khóa K:=M.

```
Algorithm Key
Format:    Key(U,F,M)
Input:  - Tập thuộc tính U
        - Tập PTH F
Output:    - Khóa K ⊆ U thỏa
           K+ = U
           ∀A∈K: (K\{A})+ ≠ U
Method
  K:=M;
  for each attribute A in K do
     if A in Closure(K\{A},F) then
        K:=K\{A}
     endif;
  endfor;
  return K;
end Key.
```

Vấn đề đặt ra là:

Bắt đầu từ siêu khóa M như thế nào? (Đơn giản nhất là hãy chọn M=U)

Các thuộc tính A nào trong M thì nên duyệt và không cần duyệt, nhằm tăng tốc độ thực hiện thuật toán.

Các kết quả trong phần dưới đây nhằm giải quyết điều này.

6.4.1. Một số tính chất của khóa

Các tính chất đơn giản

Ta có một số tính chất của khoá như sau

a) K là siêu khoá khi và chỉ khi $K^+ = U$.

b) Cho $K \subseteq U$ và $A \in K$. Nếu $K\backslash\{A\} \rightarrow \{A\} \in F^+$ thì K không thể là khoá. Vì dễ thấy nó chỉ có thể là siêu khoá.

- $K \subseteq U$ là một khoá khi và chỉ khi U phụ thuộc đầy đủ vào K.
- Hai khoá khác nhau của một LĐQH không bao nhau.
- Mọi LĐQH đều có ít nhất một khoá.

Cho p=(U,F) là một lược đồ quan hệ, $U = \{A_1A_2...A_n\}$,

$F=\{L_i \rightarrow R_i \mid L_i, R_i \subseteq U; i=1,...p\}$. Có thể giả sử $L_i \cap R_i = \emptyset$. Vì với $L_i \rightarrow R_i$ mà $L_i \cap R_i \neq \emptyset$, ta thay bởi phụ thuộc hàm $L_i \rightarrow R_i\backslash L_i$. Ta có $L_i \rightarrow R_i$ và $L_i \rightarrow R_i\backslash L_i$ là tương đương, do $L_i \rightarrow R_i \rightarrow R_i\backslash L_i$ suy ra $L_i \rightarrow R_i$ và ngược lại từ $L_i \rightarrow R_i\backslash L_i$ kết hợp với $L_i \rightarrow L_i$, dựa vào luật hợp ta có $L_i \rightarrow R_i$. Ký hiệu $L = \bigcup_{i=1}^{p} L_i, R = \bigcup_{i=1}^{p} R_i$

Gọi $\underline{K}$ = {Tập tất cả các khoá của p=(U,F)}

$$H = \bigcup_{K \in \underline{K}} K$$

$$\underline{H} = U\backslash H$$

Thuộc tính $A \in H$ được gọi là thuộc tính khoá, còn $A \in \underline{H}$ được gọi là thuộc tính không khoá.

Ta có một số kết quả sau

Định lý 6.5. (Đặc trưng của các thuộc tính khóa)

Cho K là một khóa của LĐQH p = (U,F). Khi đó với mọi tập con X của K ta có:

$$X^+ \cap K = X.$$

Chứng minh

Vì $X \subseteq X^+$ và $X \subseteq K$ nên $X \subseteq X^+ \cap K$. Ta cần chứng minh $X^+ \cap K \subseteq X$. Giả sử $A \in X^+ \cap K$ và $A \notin X$. Ta xét tập $M = K\backslash\{A\}$. Dễ thấy $X \subseteq M$. Ta có, theo tính chất đồng biến của bao đóng, $A \in X^+ \subseteq M^+$. Từ đây suy ra $K \subseteq M^+$, do đó, theo tính chất lũy đẳng của bao đóng và tính chất khóa của K ta có, $U = K^+ \subseteq M^{++} = M^+$, tức là M là bộ phận thực sự của khóa K lại đồng thời là siêu khóa, trái với định nghĩa khóa. Vậy $A \in X$ ■

Bổ đề 6.1.

Với $A \notin L$, nếu $X \rightarrow Y \in F^+$ thì $X\backslash\{A\} \rightarrow Y\backslash\{A\} \in F^+$.

Định lý 6.6.

Cho p=(U,F) với các ký hiệu như đã nói ở trên. Với $X \subseteq U$ là khoá của R thì:

$$(U\backslash R) \subseteq X \subseteq (U\backslash R) \cup (L \cap R)$$

Chứng minh

Trước hết ta chứng minh nếu X là khoá thì $U\backslash R \subseteq X$. Thật vậy, do X là khoá nên $X^+ = U$. Mặt khác theo thuật toán tìm bao đóng, ta có $X^+ \subseteq X \cup R \subseteq U$. Từ đây suy ra $X \cup R = U$. Vậy $U\backslash R \subseteq X$.

Tiếp tục, ta chứng minh nếu X là khoá thì $X \subseteq (U\backslash R) \cup (L \cap R)$ (*).

Ta có $X \subseteq U = (U\backslash R) \cup (L \cap R) \cup (R\backslash L)$. Nếu chứng minh được $X \cap (R\backslash L) = \emptyset$ ta sẽ có ngay (*).

Giả sử $X \cap (R\backslash L) \neq \emptyset$. Khi đó tồn tại thuộc tính A: $A \in X$, $A \in R$ và $A \notin L$. Do X là khoá nên $X \rightarrow L$. Lại do $A \notin L$ và $X\backslash\{A\} \rightarrow L\backslash\{A\}$ nên $X\backslash\{A\} \rightarrow L$. Dễ thấy $L \rightarrow R$ và $R \rightarrow A$. Từ đây suy ra $X\backslash\{A\} \rightarrow A$, điều này mâu thuẫn với giả thiết X là khoá (vì lúc này $X\backslash\{A\}$ lại là một khoá của p=(U,F).

Hệ quả 6.1.

$K = (U\backslash R) \cup (L \cap R)$ là một siêu khoá của p=(U,F).

Hệ quả 6.2.

Cho p = (U,F) là một lược đồ quan hệ. Nếu $R\backslash L \neq \emptyset$ thì tồn tại một khoá $K \neq U$.

Hệ quả 6.3.

Cho p= (U,F) là một lược đồ quan hệ. Nếu $R \cap L = \emptyset$ thì $U\backslash R$ là khoá duy

nhất của p=(U,F).

Mệnh đề 6.4.

$$\bigcap_{K \in \underline{K}} K = U \setminus R$$

Tức là giao của tất cả các khoá K trong $\underline{K}$ đúng bằng tập các thuộc tính không nằm ở trên vế phải của các phụ thuộc hàm trong F.

Chứng minh

Trước hết dễ thấy với A là một thuộc tính khoá, khi đó A không thể chỉ nằm ở vế phải của các phụ thuộc hàm trong F, do đó $A \in U \setminus R$. Tức $\bigcap_{K \in \underline{K}} K \subseteq U \setminus R$.

Ngược lại, ta chứng minh $\bigcap_{K \in \underline{K}} K \supseteq U \setminus R$.

Thật vậy với một thuộc tính A bất kỳ, $A \in U \setminus R$, giả sử $A \notin \bigcap_{K \in \underline{K}} K$.

Tức tồn tại một khoá K sao cho $A \notin K$. Do K là khoá nên $K^+ = U$, nhưng vô lý ở chỗ $A \notin K$ và A cũng không thuộc vế phải của các FD trong F, như thế K^+ không thể nào bằng U được (vì thiếu A). Vì vậy ta phải có $\bigcap_{K \in \underline{K}} K \supseteq U \setminus R$.

Tóm lại ta có: $\bigcap_{K \in \underline{K}} K = U \setminus R$.

Thuật toán xác định giao các khóa trong LĐQH

```
Algorithm KeyIntersec
Format: KeyIntersec(U,F)
Input:     - Tập thuộc tính U
        - Tập PTH F
Output:    - Giao các khóa
                    M = U \ ⋃_{L→R∈F} (R \ L)
Method
  M:=U;
  for each FD L→R in F do
     M:=M\(R\L);
  endfor;
```

```
  return M;
end KeyIntersec.
```

Tích *nm* chính là chiều dài của biểu diễn LĐQH $p = (U,F)$ tức là chiều dài của dữ liệu vào trong thuật toán KeyIntersec.

Định lý 6.7.

Cho p=(U,F) là một lược đồ quan hệ, với các ký hiệu như đã nói trên. Ta có p=(U,F) có khoá duy nhất khi và chỉ khi $(U\backslash R)^+ = U$. Lúc này K=U\R là khoá duy nhất của p.

Chứng minh

[<=]. Nếu X = U\R là siêu khoá suy ra $\exists K \subseteq X = U\backslash R$ (1) là khoá.

Ta có $U = K^+ \subseteq K \cup R \Rightarrow U\backslash R \subseteq K$ (2). Từ (1) và (2) ta có K=U\R.

Mặt khác nếu K' là một khoá tối thiểu, ta thấy $U\backslash R \subseteq K'$ nên K=K'. Hay K là duy nhất.

[=>]. Theo mệnh đề 6.4. ta có $\bigcap_{K \in \underline{K}} K = U \setminus R$

Nếu p=(U,F) có khoá duy nhất K thì K=U\R hay $(U\backslash R)^+ = U$.

Thí dụ

Cho LĐQH $p = (U, F)$, $U = ABCDE$, $F = \{AB \rightarrow C, AD \rightarrow B, B \rightarrow D\}$.

Ta có, giao của các khóa là $U_I = ABCDE \backslash BCD = AE$. $U_I^+ = (AE)^+ = AE \neq U$ nên p có hơn một khóa. Vì U_I là giao các khóa nên ta có thể bổ sung cho U_I một số thuộc tính để thu được các khóa. Dễ xác định được p có hai khóa là $K_1 = ABE$ và $K_2 = ADE$. Ta còn có, tập các thuộc tính khóa là $U_K = ABDE$, tập các thuộc tính không khóa là $U_0 = C$.

Định lý 6.8.

Điều kiện đủ để p=(U,F) có 1 khoá duy nhất là $Card(L \cap R) \leq 1$.

Chứng minh

Xét hai trường hợp

Trường hợp 1: $Card(L \cap R) = 0$ tức $L \cap R = \varnothing$. Từ định lý 6.7. suy ra ngay p=(U,F) có khoá duy nhất là U\R.

Trường hợp 2: $Card(L \cap R) = 1$. Trong trường hợp này ta sẽ chứng minh $X = (U\backslash R) \cup (L \cap R)$ không là khoá bằng phản chứng như sau:

Nếu X là khoá , do U\R $\subseteq$ X $\subseteq$ (U\R)$\cup$(L$\cap$R) và để ý rằng L$\cap$R chỉ gồm một phần tử, nên X là khoá duy nhất. Như vậy giao tất cả các khoá của p = (U,F) đúng bằng X. Mặt khác theo mệnh đề 6.4 ta có giao tất cả các khoá của p = (U,F) lại là U\R $\neq$ X. Điều này vô lý. Vậy X không thể là khoá mà chỉ là một siêu khoá. Loại bỏ thuộc tính duy nhất A của L$\cap$R ta được khoá duy nhất của p= (U,F) là U\R. Định lý chứng minh xong.

Nhận xét

Ta có một số trường hợp đặc biệt của định lý trên như sau

+ $(U\backslash R)^+ \neq U$ suy ra được lược đồ p = (U,F) có nhiều hơn một khoá.

+ Nếu $(U\backslash R)^+ \neq U$ và (L$\cap$R) ={A_{i1},A_{i2}}, khi đó p = (U,F) có đúng hai khoá $K_1 = (U\backslash R)A_{i1}$ và $K_2 = (U\backslash R)A_{i2}$.

Thật vậy, lúc này dễ thấy lược đồ có nhiều hơn 1 khoá và (U\R)(L$\cap$R) là siêu khoá nhưng không là khoá. Như vậy chỉ có hai khả năng $K_1=(U\backslash R)A_{i1}$ và $K_2 = (U\backslash R)A_{i2}$ là khoá. Nhưng nếu một trong hai tập này không là khoá thì lược đồ của ta lại chỉ có một khoá duy nhất. Do đó P = (U,F) có hai khoá là K_1 và K_2.

Ví dụ

Cho U={ABCDEG}; F={B$\rightarrow$C; C$\rightarrow$B; A$\rightarrow$GD}. Ta có L={ABC} và R={BCDG}. Suy ra L$\cap$R={B,C}. Mặt khác U\R={AE}; $(AE)^+$ = {AEGD}$\neq$U. Theo nhận xét trên lược đồ có đúng hai khoá K_1={AEB} và K_2={AEC}.

6.4.2. Định lý Luccheisi - Osborn và bài toán tìm mọi khoá của lược đồ quan hệ

Ở trên chúng ta đã khẳng định một lược đồ luôn tồn tại ít nhất 1 khóa và chỉ ra cách tìm để tìm mọi khóa của lược đồ, tương tự như các bài toán tìm kiếm khác, một cách tự nhiên ta phải giải quyết các câu hỏi sau:

- Có còn (khóa) nữa không?

- Nếu còn thì nên tìm thêm 1 (khóa) bắt đầu từ đâu? (một siêu khóa như thế nào) để khỏi trùng lắp với cái (khóa) đã có?

Định lý sau trả lời 2 câu hỏi đó.

Định lý 6.9. (Luccheisi-Osborn)

Cho lược đồ quan hệ p = (U,F), Gọi H là tập không rỗng các khoá của p=(U,F) và P_U là tập các tập con của U. Điều kiện cần và đủ để họ $P_U\backslash H$ có chứa

khoá của p=(U,F) là tồn tại phần tử K∈H và tồn tại phụ thuộc hàm L_{i0}→R_{i0} ∈F sao cho tập T=L_{io} ∪(K\R_{io}) không chứa một phần tử nào của H.

Chứng minh

[<=]. Giả sử ∃K∈H, ∃L_{io} →R_{io} ∈F sao cho T=L_{io} ∪(K\$R_{io}$) không chứa một phần tử nào của H. Dễ thấy L_{io} ∪(K\$R_{io}$) là một siêu khoá nên chứa một khoá của p=(U,F). Do T không chứa một phần tử nào của H nên T phải chứa một khoá thuộc họ P_U\H. Nói cách khác P_U\H có chứa khoá của p=(U,F).

[=>]. Giả sử P_U\H có chứa khoá K' của p=(U,F). Gọi K" là tập tối đại thoả hai tính chất sau:

i) K'⊆K"

ii) K" không chứa phần tử nào của H.

Dễ thấy K" là siêu khoá và tập K" như thế là tồn tại (ít nhất là bằng K').

Do ii) nên trong quá trình tính bao đóng của K" phải tồn tại phụ thuộc hàm L_{io}→R_{io} sao cho L_{io} ⊆K" và R_{io}⊄ K".

Xét tập K"∪R_{io}. Vì K" là tập tối đại thoả ii) nên tồn tại K∈H sao cho K⊆K"∪R_{io}. Từ K⊆K"∪R_{io} và R_{io}⊄K" nên

K\$R_{io}$ ⊆K". Hơn nữa L_{io}⊆K", nên ta có T=L_{io} ∪(K\$R_{io}$) ⊆K". Chứng tỏ sự tồn tại của K∈H và sự tồn tại L_{io} →R_{io} ∈F sao cho T=L_{io} ∪(K\$R_{io}$) không chứa khoá nào thuộc H. □

Dựa vào định lý trên và thuật toán xác định một khoá đã trình bày ở trên, ta có thể đưa ra một thuật toán xác định tập tất cả các khoá của lược đồ quan hệ p=(U,F) như sau.

Thuật toán tìm mọi khoá

```
Vào: p=(U,F);
Ra: Tập HR tất cả các khoá của p=(U,F);
Phương pháp
Bước 1: HR:=KEY(U,F,(U\R)∪(L∩R));
Bước 2: For mỗi K trong HR do
     Begin
        For mỗi Li→Ri ∈F do
        Begin
```

```
            T:= Lj∪(K\Rj);
            Test:= True;
            For mỗi P  trong HR do
               If  T chứa P then Test:=False;
            If Test then HR:=HR∪(KEY(U,F,T);
            End;
         End;
RETURN HR;
```

Ví dụ

Cho U={ABCDEG}; F={A→BC; B→D; AD→E; CD→A}

Xét lược đồ quan hệ p=(U,F).

a) Tìm một khoá của p=(U,F) và xét xem khoá đó có duy nhất không.

Ta có U\R={G} và L∩R={ABCD}. Như vậy thuộc tính G thuộc về mọi khoá. Dựa vào phụ thuộc hàm thứ nhất ta có ngay K=ADEG là một siêu khoá của p = (U,F). Áp dụng thuật toán KEY ta có K_1={AG} là một khoá. Ta có $(U\backslash R)^+=(G)^+=\{G\}\neq U$, nên p=(U,F) có nhiều hơn một khoá (định lý 6.7)

b) Tìm thêm một khoá

Ta thấy $(CDG)^+$=U, nên K'={CDG} là một siêu khoá. Áp dụng thuật toán KEY ta có K_2=K'={CDG} là khoá.

c) Ngoài hai khoá trên còn có khoá nào nữa không

H={K_1,K_2}.

Xét K_1 : Với mọi phụ thuộc hàm trong F ta thấy $L_i\cup(K_1\backslash R_i)$ có chứa K_1 hoặc K_2 .

Xét K_2: Với B→D thì B∪(CDG\D)={BCG} không chứa K_1 và K_2. Nên p=(U,F) có chứa thêm khoá nữa đó là K_3 = {BCG} chẳng hạn.

Bài tập chương 6

1. Cho lược đồ quan hệ Q(ABCD), r là quan hệ 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 ?

$AB \to D$; $C \to B$; $B \to C$; $BC \to A$; $BD \to A$.

2. Cho lược đồ quan hệ R và tập phụ thuộc hàm

F = {AB→C, B→D, CD→E, CE→GH, G→A}

Chứng minh rằng nếu quan hệ r xác định trên R thỏa F thì cũng thỏa AB→E và AB→G.

3. Cho lược đồ quan hệ R và tập phụ thuộc hàm F xác định trên R

F = {A→C, BC→D, D→E, E→A}.

Tính: $(AB)^+$, $(BD)^+$.

4. Cho lược đồ quan hệ R và tập phụ thuộc hàm F xác định trên R

F = {B→C, AC→D, D→G, AG→E}. Cho biết:

a) $AB \to G \in F^+$?

b) $BD \to AD \in F^+$?

5. Xác định các khoá của từng Q_i và một phủ tối thiểu của từng F_i

a) $Q_1(ABCDEGH)$

$F_1 = \{A \to H; AB \to C; BC \to D; G \to B\}$

b) $Q_2(ABCSXYZ)$

$F_2 = \{S \to A; AX \to B; S \to B; BY \to C; CZ \to X\}$

c) $Q_3(ABCD)$

$F_3 = \{A \to B; BC \to D; D \to A\}$

d) $Q_{15}(MNLPRS)$

$F_{15} = \{M \to N; MR \to N; PN \to LR; L \to S; S \to R\}$

e) $Q_{16}(ABCDE)$

$F_{16} = \{DE \to A; C \to DE; AD \to B; BE \to C\}$

6. Chứng minh rằng quan hệ r thoả mãn phụ thuộc hàm X → Y nếu và chỉ nếu X là khoá của lược đồ quan hệ r(XY).

7. Cho LĐQH p. Biết p có một khóa K. Hãy xây dựng thuật toán tìm một khóa thứ hai M của p. Nếu p không có khóa thứ hai thuật toán cho kết quả là một tập rỗng.

8. Xây dựng một LĐQH có 5 thuộc tính A, B, C, D, E, mỗi thuộc tính là một khóa.

9. LĐQH có 5 thuộc tính có thể có tối đa bao nhiêu khóa. Cho thí dụ.

10. Xây dựng một LĐQH có 5 thuộc tính A, B, C, D, E và chỉ có một khóa duy nhất.

11. Cho K là một khóa của LĐQH p = (U,F). Chứng minh rằng với mọi tập con X của K ta có: $X^{+} \cap K = X$.

12. Khẳng định tính đúng của các mệnh đề sau:

a) Hai khoá khác nhau của một LĐQH không bao nhau.

b) Mọi LĐQH đều có ít nhất một khoá.

c) Số khoá của một LĐQH không thể lớn hơn số thuộc tính.

d) U không thể là khoá của LĐQH (U,F).

e) Mọi LĐQH không thể có hai khoá đơn tức là khoá chỉ gồm một thuộc tính.

CHƯƠNG 7. PHÂN TÁCH

Mục đích

- Trình bày lý thuyết phân tách lược đồ quan hệ.

- Một số bài toán có liên quan và một số ví dụ thực tiễn cho việc phân tách cũng được trình bày.

Yêu cầu

- Sinh viên nắm được các kỹ thuật phân tách. Hiểu được ý nghĩa của các bài toán này trong thực tiễn và thiết kế một số CSDL mẫu trong bài tập.

Trong quá trình thiết kế cơ sở dữ liệu quan hệ, giả sử ta có một lược đồ quan hệ R xây dựng trên tập thuộc tính U với tập ràng buộc phụ thuộc hàm F. Một số yêu cầu là làm sao hạn chế dư thừa dữ liệu trên R và giảm tính dị thường dữ liệu trên R, hoặc đôi khi ta phải phân chia các công việc quản lý dữ liệu trên R cho nhiều người cùng làm rồi ghép nối kết quả lại. Một cách giải quyết là phân chia R thành các lược đồ con bằng cách phân tách dọc hay các mảnh bằng phân tách ngang. Ở đây ta tập trung vào lý thuyết phân tách dọc. Khi phân tách R thành các lược đồ con như thế, một điều hết sức tự nhiên là các lược đồ con nên chia nhỏ như thế nào? bao gồm tập con các thuộc tính của U như thế nào và làm sao để khi kết ghép các kết quả làm việc trên các lược đồ con ta vẫn có các thông tin như ban đầu nếu ta làm việc trên toàn bộ R. Phần phân tách ngang, nếu có điều kiện sẽ được nêu trong cơ sở dữ liệu phân tán.

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:

Cho lược đồ Ketquathi(MSSV, TenSV, Tuoi, Qquan, Dchi, Mônthi, điểmthi) với giả sử một sinh viên có thể thi nhiều môn thi. Chúng ta có thể nhận thấy một số vấn đề nảy sinh sau:

1) Dư thừa (redundancy): TênSV, Tuoi, Qquan, Dchi của các sinh viên được lặp lại mỗi lần cho mỗi môn thi.

2) Mâu 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), chúng ta có thể xoá tất cả các môn thi của một sinh viên, vô tình sẽ làm mất dấu vết để tìm ra họ tên của sinh viên này.

Một trong các cứu cánh để giải quyết vấn đề này là phân tách. Giả sử ta xét 2 phân tách Ketquathi như sau:

Phân tách 1: KQ1(MSSV, TenSV, Tuoi, Qquan) và KQ2(Dchi, Mônthi, điểmthi).

Phân tách 2: Sinhviên(MSSV, TenSV, Tuoi, Qquan, Dchi) và KQ(MSSV, Mônthi, điểmthi). Ta có suy nghĩ gì về hai phân tách trên.

7.1. Phân tách có kết nối không tổn thất (không mất thông tin)

Định nghĩa 7.1. Cho $U=\{A_1,...,A_n\}$, R(U) là một lược đồ quan hệ trên U - ta đồng nhất R với tập thuộc tính của nó $R=\{A_1,...,A_n\}$. Một phân tách của R là việc thay thế lược đồ R bằng một họ các lược đồ con.

$$\rho = (R_1,...,R_m)$$

Sao cho: $\bigcup_{i=1}^{m} Ri = R$

Thí dụ Với R(A,B,C)

thì $\rho 1=(AB,BC)$ là một phân tách,

Trong khi $\rho 2=(A,AC)$ không phải là một phân tách.

Cho $\rho = (R_1,...,R_m)$ là một phân tách của lược đồ quan hệ $(A_1,...,A_n)$ và r là một thể hiện của lược đồ quan hệ <R,F>.

Gọi $m_p(r)$ là ánh xạ được xác định bởi:

$$m_p(r) = r_1 \bowtie r_2 \bowtie ... \bowtie r_m$$

Trong đó $r_i = \prod_p(r)$ với i = 1, 2, ...,m, còn $\bowtie$ là ký hiệu của phép kết nối tự nhiên.

Mệnh đề 7.1.

a) $r \subseteq m_p(r)$;

b) Nếu $s = m_p(r)$ thì $\pi_{Ri}(s) = r_i$;

c) $m_p(m_p(r)) = m_p(r)$.

Chứng minh

a) Gọi t là một bộ thuộc r. Vậy với mỗi i, $t_i = t[R_i] \in r_i$. Theo định nghĩa của phép kết nối tự nhiên, $t \in m_p(r)$ vì $t[R_i]=t_i$ cho mọi i trên các thuộc tính của R_i .

b) Theo a) ta có $r \subseteq s$, suy ra $\prod_{Ri}(r) \subseteq \prod_{Ri}(s)$. Điều đó có nghĩa là $r_i \subseteq \prod_{Ri}(s)$. Cần chỉ ra rằng $\prod_{Ri}(s) \subseteq r_i$.

Giả sử với một i mà $t_i \in \prod_{Ri}(s)$. Khi đó có $t \in s$ sao cho $t[R_i]=t_j$. Cũng vì $t \in s$ sao cho $t[R_i]=u_j$ cho nên có $u_j \in r_i$ sao cho $t[R_i]=u_j$. Trong trường hợp này $t[Ri] \in r_i$. Nhưng vì $t[R_i]=t_i$; do vậy $t_i = r_j$ và do đó $\prod_{Ri}(s) \subseteq r_i$. Từ đó có $r_i = \prod_{Ri}(s)$.

c) Nếu $s=m_p(r)$ thì theo b) ta có $\prod_{Ri}(s) = r_i$. Do vậy:

$$m_p(s) = \overset{k}{\underset{i=1}{\rhd\lhd}} r_i = m_p(r).$$

Cần chú ý trong trường hợp tổng quát, với mỗi i, r_i là một quan hệ trên R_i và $s = \overset{k}{\underset{i=1}{\rhd\lhd}} r_i$ thì khi đó $\prod_{Ri}(s)$ không nhất thiết phải bằng r_i .Mối quan hệ r_i có thể thiếu hoặc thừa một số bộ nào đó (đều gọi là mất thông tin).

Chẳng hạn nếu $R_1=AB$, $R_2 =BC$ và các quan hệ tương ứng là $r_1=\{a_1b_1\}$, $r_2 =\{b_1c_1, b_2c_2\}$, khi đó $s =\{ a_1b_1c_1\}$. Và xảy ra $\prod_{BC}(s)=\{b_1c_1\} \neq r_2$ (Trường hợp này bị mất bộ $\{b_2c_2\} \in r_2$).

Định nghĩa 7.2. (Phân tách có kết nối không mất thông tin)

Phân tách $\rho = (R_1, R_2,...,R_m)$ được gọi là phân tách có kết nối không tổn thất (gọi tắt là không mất thông tin) của lược đồ quan hệ <R,F> nếu như với mọi r là thể hiện của R mà thỏa F thì:

$$r = r_1 \rhd\lhd r_2 \rhd\lhd \ldots \rhd\lhd r_m \text{ (tức } r = m_p(r)).$$

Thuật toán 7.1. (kiểm tra một phân tách không mất thông tin)

Vào: Lược đồ quan hệ <R,F>, phân tách $p = (R_1,R_2,...,R_n)$

Ra: Cho câu trả lời đúng hay sai, tuỳ thuộc phân tách có mất thông tin hay không ?

Phương pháp

1. Lập bảng ban đầu là ma trận m hàng (ứng với m lược đồ con R_i) và n cột ứng với tập thuộc tính $\{A_1,A_2,...,A_n\}$. Phần tử (i,j) của ma trận được xác định theo công thức sau:

$$(i,j) = \begin{cases} a_i, & if\ A_j \in R_i \\ b_{ij}, & if\ A_j \notin R_i \end{cases}$$

trong đó a_j, $b_{ij} \in Dom(A_i)$.

2. Biến đổi bảng:

Mục đích của việc biến đổi bảng là để thu được bảng cuối cùng, xem như một quan hệ, thoả tập F. Muốn vậy, lần lượt xem xét các phụ thuộc hàm $(X \rightarrow Y) \in F$.

Với mỗi phụ thuộc hàm như vậy nếu phát hiện trên bảng có hai hàng giống nhau trên X và khác nhau trên Y thì phải làm cho hai hàng đó cũng giống nhau trên Y. Quá trình làm bằng đó được thực hiện như sau:

Không làm mất tổng quát, giả sử hai hàng i_1 và i_2 của bảng giống nhau trên X và khác nhau trên $A_j \in Y$.

Ta có các trường hợp sau (xử lý tương tự cho các trường hợp đối xứng):

1. $(i_1,j) = a_j$ và $(i_2,j) = b_{i2j}$ Khi đó phải cho $b_{i2j} = a_j$
2. $(i_1,j) = b_{i1j}$ và $(i_2,j) = b_{i2j}$

Khi đó, tuỳ ý có thể $b_{i1j} = b_{i2j}$ (hoặc $b_{i2j} = b_{i1j}$)

(Tức làm bằng trên Y nếu hai hàng giống nhau trên X, có ưu tiên cho a_i

Làm như vậy cho tới khi không biến đổi bảng được nữa. Khi đó, ta thu được bảng cuối cùng.

Kết luận

+ Nếu trong bảng cuối cùng có chứa hàng gồm toàn ký hiệu a (tức hàng $(a_1,..., a_n)$), ta kết luận $p = (R_1,..., R_m)$ là phân tách không mất thông tin (tức cho câu trả lời đúng).

+ Trường hợp ngược lại, p là phân tách mất thông tin.

Thí dụ

a) Dùng kỹ thuật bảng để kiểm tra tính kết nối không tổn thất của phép tách trong LĐQH

$p = (U,F)$, $U = ABCD$, $F = \{ A \to B, AC \to D \}$, $\rho = (AB, ACD)$. T==> T*

	1. A	2. B	3. C	4. D
1. AB	a_1	a_2	b_{13}	b_{14}
2. ACD	a_1	b_{22} / a_2	a_3	a_4

Vì T^* chứa dòng thứ hai gồm toàn ký hiệu phân biệt nên phép tách đã cho là *không tổn thất*.

b) Dùng kỹ thuật bảng để kiểm tra tính kết nối không tổn thất của phép tách trong LĐQH $p = (U,F)$, $U = ABCDE$, $F = \{ A \to C, B \to C, C \to D, DE \to C, CE \to A \}$, $\rho = (AD, AB, BE, CDE)$. T==> T*

	1. A	2. B	3. C	4. D	5. E
1. AD	a_1	b_{12}	b_{13} / a_3	a_4	b_{15}
2. AB	a_1	a_2	b_{23} / b_{13} / a_3	b_{24} / a_4	b_{25}
3. BE	b_{31}	a_2	b_{33} / b_{13} / a_3	b_{34} / a_4	a_5
4. CDE	b_{41} / b_{31}	b_{42}	a_3	a_4	a_5

Vì T^* không chứa dòng nào gồm toàn ký hiệu phân biệt nên phép tách đã cho là *tổn thất*.

Nhận xét

Thực ra thuật toán có thể dừng khi xuất hiện một hàng toàn a_i.

Bài tập

Dùng thuật toán trên kiểm tra với lược đồ quan hệ:

<{ A, B, C, D, E, F }, {A -> C, B -> C, C -> D, DE -> C, CE -> A)} phân tách ρ=(AD, AB, BE, CDE, AE) mất thông tin hay không?

Định lý 7.1. Thuật toán 7.1. trả lời đúng liệu phép tách có kết nối mất thông tin hay không.

Chứng minh: Giả sử bảng cuối cùng được tạo ra bởi thuật toán 7.1. không có dòng nào tất cả là a_i. Chúng ta có thể xem bảng này như là một lược đồ quan hệ

cho lược đồ R (các dòng là các bộ của quan hệ). Theo sự biến đổi của thuật toán quan hệ R thỏa mãn các phụ thuộc hàm trong F. Rõ ràng rằng r không chứa bộ $a_1a_2...a_n$ nhưng đối với mỗi R_i có một bộ α_I trong r sao cho $\alpha_I = \Pi_{Ri}(r)$. Hay nói cách khác phép tách ρ không có kết nối không mất thông tin.

Ngược lại, giả sử bảng cuối cùng có một dòng tất cả đều là $a(a_1,a_2.,..,.a_n)$. Ứng với mỗi bảng T được tạo ra bởi thuật toán trên chúng ta xét biểu thức

$$\{ a_1a_2...a_n / (\exists b_{11})..(\exists b_{kn}) (R(s_1)\wedge.. \wedge(s_k))\} \qquad (1)$$

Ở đây s_i là dòng thứ i của T. Khi T là bảng cuối cùng công thức (1) xác định ánh xạ $m_\rho(r)$ chứa bộ $a_1a_2...a_n$ nếu và chỉ nếu đối với mỗi i,r chứa một bộ có a_j trong thành phần thứ j nếu A_j là một thuộc tính của R_i và giá trị tùy ý được biểu diễn bởi b_j trong mỗi thuộc tính còn lại của A_j.

Do chúng ta giả thiết rằng quan hệ R đối với quan hệ R thỏa mãn các phụ thuộc hàn F, chúng ta có thể thấy rằng mỗi một phép biến đổi bảng được thực hiện bởi thuật toán làm thay đổi bảng theo cách mà nó không làm ảnh hưởng đến tập các bộ được tạo ra bởi (1). Chứng minh chi tiết của khẳng định này là phức tạp nhưng về trực quan có thể nhận thấy rằng: chúng ta chỉ có thể đặt tên ký hiệu nếu trong (1) đã áp dụng đối với quan hệ r thõa mãn F, những ký hiệu như vậy không có thể được gán cùng một giá trị.

Vì bảng cuối cùng chứa một dòng có tất cả là a_i biểu thức tính toán miền cho bảng cuối cùng có dạng

$$\{ a_1a_2...a_n / (\exists b_{11})..(\exists b_{kn}) (R(a_1a_2...a_n)...)\} \qquad (2)$$

Rõ ràng giá trị của (2) đã áp dụng đối với quan hệ r của R là một tập con của r. Tuy nhiên nếu r thỏa mãn F thì giá trị của (2) tính chính xác r, và vì vậy $r = m_\rho(r)$.

Định lý 7.2. (Delobel)

Cho <R, F> là một lược đồ quan hệ, khi đó $p = (R_1, R_2)$ là phân tách có kết nối không tổn thất (đối với F) nếu và chỉ nếu:

$$(R_1 \cap R_2) \rightarrow (R_1 - R_2) \text{ hay } (R_1 \cap R_2) \rightarrow (R_2 - R_1)$$

Chứng minh:

Xét bảng sau

	$R_1 \cap R_2$	$R_1 \backslash R_2$	$R_2 \backslash R_1$
Hàng cho R_1	aa...aaa	aaa...aa	aa...aa

Hàng cho R_2 aa...aaa bbb...bb aa...aa

Đó là bảng ban đầu trong thuật toán kiểm tra tách không mất thông tin nhưng ta đã lược bỏ các chỉ số. Ta thấy một ký hiệu b ở một cột A được đổi thành a thì A thuộc vào $(R_1 \cap R_2)^+$. Đồng thời cũng chỉ ra được rằng nếu $R_1 \cap R_2 \rightarrow Y \in F^+$ thì các ký hiệu b của các cột trong Y sẽ được đổi thành a. Vì vậy hàng cho R_1 đổi thành a hết nếu và chỉ nếu $R_1 \cap R_2 \rightarrow R_2\text{-}R_1$ và tương tự hàng cho R_2 đổi thành a hết nếu và chỉ nếu $R_1 \cap R_2 \rightarrow R_1\text{-}R_2$.

Dưới đây là một cách phát biểu khác của định lý Delobel (với R_1=XY và R_2=XZ, $R_1 \cap R_2$=X và $R_1\text{-}R_2$=Y), ở đây chúng tôi đề nghị một cách chứng minh khác.

Định lý 7.3: Nếu R=(XYZ) và $X \rightarrow Y \in F^+$ thì phân tách ρ=(XY, XZ) là một phân tách không mất thông tin.

Chứng minh:

Đặt $M=XY$, $Z=U\text{-}Y$, ta có $MZ=U$ và $M \cap Z = X$. Ta cần chứng minh $\forall R \in SAT(X \rightarrow Y)$: $R[M]*R[Z]=R$. Theo mệnh đề 7.1. ta có $R[M]*R[Z] \supseteq R$, ta còn phải chứng minh $R[M]*R[Z] \subseteq R$.

Giả sử $u \in R[M]$, $v \in R[Z]$ và u và v thoả điều kiện kết nối tự nhiên, tức là $u.X = v.X$, ta chứng minh $u*v \in R$.

Vì $u \in R[M]$ nên $\exists\, u' \in R$: $u = u'.M$. Vì $v \in R[Z]$ nên $\exists\, v' \in R$: $v = v'.Z$. Vì $X = M \cap Z$ nên $X \subseteq M$ và $X \subseteq Z$, từ đó suy ra $u.X = u'.M.X = u'.X = v.X = v'.Z.X = v'.X$.

Vì R thỏa PTH $X \rightarrow Y$ nên từ $u'.X = v'.X$ ta suy ra $u'.Y = v'.Y$ và do đó $u'.M = u'.XY = v'.XY = v'.M$. Từ đây suy ra $u*v = v' \in R$, *đpcm.*

+ Dạng phát biểu trên của định lý sẽ là cơ sở cho việc kiểm tra pháp tách BCNF nêu ra trong một thuật toán ở phần tiếp theo là có kết nối không tổn thất.

Trong thực tiễn thiết kế cơ sở dữ liệu, đôi khi các phân tách là có kết nối không mất thông tin, các lược đồ con đảm bảo đúng các ràng buộc trên nó. Tuy vậy khi kết nối các lược đồ con lại với nhau thường dẫn đến các vi phạm toàn vẹn dữ liệu vì có một số ràng buộc bị mất đi.

Nhận xét:

+ Định lý còn được phát biểu ở một dạng khác là:

Cho <R, F> là một lược đồ quan hệ, giả sử $X \rightarrow Y \in F^+$ và $X \cap Y = \varnothing$ khi đó p=(XY, (R\Y)) là phân tách có kết nối không tổn thất (đối với F).

Bây giờ chúng ta hãy xét ví dụ sau để thấy một tính chất đáng mong muốn nữa của phép tách.

Ví dụ: Xét lược đồ sau: Họcsinh(MSHS, Tên, Tuổi, ĐiểmHT, Điểm RL, Xếp loại). Biết rằng Xếp loại được tính từ ĐiểmHT và Điểm RL. Nếu phân tách Họcsinh thành 2 lược đồ con Họctập(MSHS, Tên, Tuổi, ĐiểmHT) và Renluyen(MSHS, Tên, Điểm RL, Xếp loại). Rõ ràng phân tách này là kết nối không mất thông tin, nhưng ràng buộc để tính Xếp loại cho học sinh là không thể tính được!

Ta chuyển sang một tính chất mong muốn khác của phân tách. Đó là những phân tách bảo toàn phụ thuộc dữ liệu.

7.2. Phân tách bảo toàn phụ thuộc dữ liệu

Định nghĩa 7.3. Cho lược đồ quan hệ <R,F> và $Z \subseteq R$. Hình chiếu của F lên tập thuộc tính Z, ký hiệu $\pi_Z(F)$ được định nghĩa là:

$$\pi_Z(F) = \{X \rightarrow Y | (X \rightarrow Y) \in F^+ \text{ và } XY \subseteq Z\}$$

Định nghĩa 7.4.

Phân tách $\rho = (R_1, R_2, ..., R_m)$ của <R,F> được gọi là bảo toàn phụ thuộc F nếu:

$$(\bigcup_{i=1}^{m} \prod_{R_i}(F)) = F^+ .$$

Chú ý:

Hai tính chất của phép tách có kết nối không tổn thất và bảo toàn tập phụ thuộc hàm F là độc lập với nhau. Có thể cho ví dụ về phân tách có kết nối không tổn thất nhưng không bảo toàn tập F và ngược lại.

Ví dụ

a) Với quan hệ:

<{C, S, Z}, {CS -> Z, Z -> C}>,

Phân tách p = (SZ, CZ) có kết nối không tổn thất, nhưng không bảo toàn tập F.

b) Với lược đồ quan hệ

<{A, B, C, D}, {A -> B, C -> D}>

Phân tách p = (AB, CD) là một phân tách bảo toàn phụ thuộc hàm F nhưng có kết nối tổn thất.

7.2.1. Thuật toán kiểm tra phép tách bảo toàn phụ thuộc hàm

Thuật toán 7.2. (Kiểm tra phép tách bảo toàn phụ thuộc hàm)

Vào: Lược đồ quan hệ R, tập phụ thuộc hàm F trên R và phép tách $\rho=(R_1,..., R_k)$

Ra: Kết luận phép tách ρ có bảo toàn phụ thuộc hàm hay không.

Phương pháp:

Gọi G là hợp của các $\pi_{Ri}(F)$ (i =1,.., k) . Ta phải kiểm tra xem mỗi phụ thuộc hàm $X\rightarrow Y\in F$ có thuộc G^+ hay không bằng cách kiểm tra xem X_G^+ có chứa Y hay không. Để làm điều này ta xét lần lượt từng phụ thuộc hàm $X\rightarrow Y$ trong F. Với mỗi phụ thuộc hàm này, ta lặp đi lặp lại quá trình tính tập Z thông qua phép gán:

$$Z := Z \cup ((Z\cap R_i)_F^+ \cap R_i) \quad (\text{với } i = 1, 2,..., k)$$

Cho đến khi không còn sự thay đổi nào trên Z hoặc tính được một giá trị Z chứa Y. Nếu $Y\subseteq Z$ thì kết luận được $X\rightarrow Y \in G^+$. Và nếu tất cả các phụ thuộc hàm $X\rightarrow Y \in F$ đều thuộc G^+ thì ta kết luận được phép tách là bảo toàn phụ thuộc hàm; ngược lại thì ta kết luận phép tách là không bảo toàn phụ thuộc hàm.

Thuật toán được mô tả như sau (thuật toán trả về giá trị True nếu phép tách bảo toàn phụ thuộc hàm, ngược lại thuật toán trả về giá trị False):

```
For mỗi phụ thuộc hàm X→Y∈F do {Kiểm tra từng phụ
thuộc hàm trong F}
    Begin
        Z:=X;
        Repeat
            For i:=1 to k do
                Begin
                    Z:=Z ∪ ((Z∩Ri)F+ ∩ Ri);
```

```
                    If Y ⊆ Z Then Exit For
                End;
        Until  (không còn sự thay đổi trên Z) or (Y
⊆ Z);
        If  not (Y⊆Z) Then
             Return  False; { phép tách không bảo
toàn PTH}
    End;
Return  True; {Kết luận phép tách bảo toàn phụ thuộc
hàm }
```

Định lý 7.3.

Thuật toán 7.2. xác định đúng phép tách có bảo toàn phụ thuộc hàm hay không.

Bài tập chương 7

1. Cho <R, F> là một lược đồ quan hệ và p = (R_1, R_2,..., R_k) là một phân tách của R có kết nối không tổn thất đối với F. Với mỗi i xác định gọi $F_i = \pi_{Ri}(F)$ và giả sử σ = (S_1, S_2,...,S_m) là một phân tách của R_i có kết nối không tổn thất đối với F_i. Hãy chứng minh rằng phân tách của R: ρ' = (R_1, ...,R_i,S_1,...,S_m,R_{i+1},...,R_k) có kết nối không tổn thất đối với F.

2. Giả sử R, F và ρ giống như trong bài tập 1.

Gọi τ = (R_1,..., R_k, R_{k+1},..., R_n) là một phân tách của R thành một tập các lược đồ quan hệ chứa tất cả các lược đồ quan hệ của phân tách p. Hãy chứng minh rằng τ là một phân tách có kết nối không tổn thất đối với F.

CHƯƠNG 8. CHUẨN HOÁ

Mục đích

- Trình bày lý thuyết chuẩn hóa lược đồ quan hệ.

- Một số bài toán có liên quan và một số ví dụ thực tiễn cho việc chuẩn hóa cũng được trình bày.

Yêu cầu

- Sinh viên nắm được các kỹ thuật chuẩn hóa. Hiểu được ý nghĩa của các bài toán này trong thực tiễn và thiết kế một số CSDL mẫu trong bài tập.

Trong phần trước, ta thấy việc phân tách các lược đồ quan hệ nhằm

i) Tránh dư thừa dữ liệu.

ii) Tránh các dị thường trong cập nhật dữ liệu.

Đồng thời nhằm phân chia công việc, chia xẻ dữ liệu trong các hệ phân tán.

Vấn đề đặt ra là: Nên tách làm sao để khi kết nối ta vẫn có được các thông tin ban đầu (tách kết nối không mất thông tin), bảo toàn phụ thuộc dữ liệu và nên tách như thế nào để các lược đồ con được tách ra là "tốt" tức đảm bảo các mong muốn trên - tức phải đưa ra các tiêu chuẩn mà các lược đồ quan hệ cần thoả mãn để đáp ứng các yêu cầu đặt ra.

Giới hạn trong việc xét các lược đồ quan hệ với phụ thuộc hàm, ta có thể định nghĩa các dạng chuẩn (NF - Normal Form) 1 NF, 2 NF 3 NF và BCNF.

Với ý nghĩa trên, ta sẽ thấy các dạng chuẩn BCNF "tốt" hơn 3NF; 3NF "tốt" hơn 2NF; 2NF "tốt" hơn 1NF. Xét tập hợp các lược đồ quan hệ ở các dạng chuẩn nói trên, ta có quan hệ sau: Tập các lược đồ quan hệ BCNF$\subseteq$3NF$\subseteq$ 2NF$\subseteq$1NF.

8.1 Một số định nghĩa và khái niệm liên quan

Dưới đây để được thuận tiện hơn trong trình bày ta thay đổi một số ký hiệu mà không thay đổi bản chất và nhắc lại một số định nghĩa.

Ta đồng nhất tên lược đồ quan hệ với tập thuộc tính R của nó. Giả sử tập các phụ thuộc hàm trên R là F. Khi đó ta ký hiệu lược đồ quan hệ đã cho là <R,F>.

Định nghĩa 8.1. (thuộc tính khoá và không khoá)

Cho <R,F> là lược đồ quan hệ. Thuộc tính $A_i \in R$ được gọi là thuộc tính khoá nếu A_i có tham gia (có mặt trong) ít nhất một khoá của <R,F> . Trường hợp ngược lại, A_i được gọi là một thuộc tính không khoá.

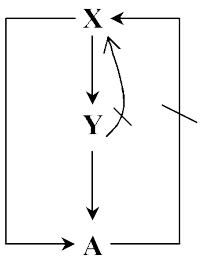
Thí dụ

Lược đồ $<\{S,A,I,P\}, \{S \rightarrow A, SI \rightarrow P\}>$ có một khoá duy nhất là SI. Do đó S,I là thuộc tính khoá và A, P là các thuộc tính không khoá.

Định nghĩa 8.2. (Phụ thuộc bắc cầu)

Cho lược đồ quan hệ <R,F>, X là một tập con các thuộc tính $X \subseteq R$, A là thuộc tính thuộc R. A được gọi là phụ thuộc bắc cầu vào X, nếu tồn tại một tập con Y của R sao cho $X \rightarrow Y$, $Y \rightarrow A$ nhưng $Y \not\rightarrow X$ với $A \notin XY$.

Tính bắc cầu thể hiện qua sơ đồ sau:



8.2. Các dạng chuẩn

Một trong các nguyên do quan trọng nhất dẫn đến các lược đồ quan hệ bị dư thừa, dị thường, cô lập… dữ liệu là mối quan hệ ràng buộc giữa thuộc tính khóa, thuộc tính không khóa với khóa chính, điều này dẫn đến sự phụ thuộc xác định giữa các thuộc tính và là nguyên nhân xảy dư thừa dữ liệu. Do đó trong thiết kế cơ sở dữ liệu quan hệ dựa trên ràng buộc phụ thuộc hàm người ta ra các dạng chuẩn sau:

Định nghĩa 8.3. (dạng chuẩn 1 - 1NF)

Lược đồ quan hệ <R,F> được gọi là ở dạng chuẩn 1NF, nếu các miền trị của các thuộc tính đề là nguyên tố, tức là chỉ chứa các giá trị không thể chia nhỏ được.

Ví dụ

MSGV	Môn dạy
GV01	Pascal, Excell
GV02	GT1, GT2

Lược đồ chưa chuẩn hóa

MSGV	Môn dạy
GV01	Pascal
GV01	Excell
GV02	GT1
GV02	GT2

Lược đồ ở 1NF

Định nghĩa 8.4. (dạng chuẩn 2 - 2NF)

Lược đồ quan hệ <R,F> được gọi là ở dạng chuẩn 2NF nếu:

i) <R,F. ở dạng chuẩn 1NF.

ii) Mọi thuộc tính không khoá đều phụ thuộc hàm đầy đủ vào mỗi khoá.

Mệnh đề 8.1.

Lược đồ quan hệ <R,F> là ở dạng chuẩn 2NF nếu và chỉ nếu:

i) <R,F> ở dạng chuẩn 1NF

ii) Với mọi thuộc tính không khoá A, với mọi khoá K của <R,F> không tồn tại $K' \subset K$, sao cho $K' \rightarrow A \in F^+$.

Ví dụ 1

a) Lược đồ $<\{A,B,C\}, \{A \rightarrow B, B \rightarrow C\}>$ rõ ràng là 2NF

b) Lược đồ $\{S,A,I,P\}, \{S \rightarrow A, SI \rightarrow P\}>$ không là 2NF

Ví dụ 2

Lược đồ quan hệ R(monthi, mssv, ten,diachi,diemthi), với

F={mssv→ten,diachi; monthi,mssv→diemthi}.

là không ở dạng chuẩn 2 vì K={mssv,monthi} là khoá, {tên} là thuộc tính không khoá và phụ thuộc hàm không đầy đủ vào K.

Định nghĩa 8.5. (dạng chuẩn 3 - 3NF)

Lược đồ <R,F> được gọi là ở dạng chuẩn 3NF nếu $X \rightarrow A \in F^+$ và $A \in X$ thì X phải là siêu khoá, hoặc A phải là thuộc tính khoá.

Ví dụ 3

1) Lược đồ quan hệ <{S,A,I,P}, {S → A, SI → P}> rõ ràng không là 3NF.

2) Lược đồ quan hệ <{C,S,Z,}, {CS → Z,Z → C}> có hai khoá là {SC} và {SZ}. Do đó tất cả các thuộc tính đều là thuộc tính khoá và do đó lược đồ quan hệ này ở 3NF.

3) Lược đồ quan hệ <{A,B,C}, {A → B, B → C}> rõ ràng ở 2NF nhưng không là 3NF.

Mệnh đề 8.2.

Lược đồ quan hệ <R,F> ở dạng 3NF nếu và chỉ nếu: <R,F> ở dạng 2NF và mọi thuộc tính không khoá không phụ thuộc bắc cầu vào khoá chính của <R,F>.

Chứng minh

[=>]. Giả sử <R,F> ở dạng 3NF, nếu <R,F> không ở 2NF, khi đó tồn tại A thuộc tính không khoá, A phụ thuộc không đầy đủ vào khoá K của R. Tức tồn tại $X \subset K$ sao cho $X \rightarrow A$, do đó X không là siêu khoá, A không là thuộc tính khoá nhưng $X \rightarrow A \in F^+$, vô lý. Nên <R,F> phải ở dạng chuẩn 2NF.

Hơn nữa giả sử trên R, tồn tại A là thuộc tính không khoá, tồn tại $X \subseteq R$, $A \notin X$ sao cho A phụ thuộc bắc cầu vào khoá chính K của R. Như vậy trên R đã tồn tại $X \rightarrow A$ mà $A \notin X$ nhưng X không là siêu khóa, mà A là thuộc tính không khoá. Vô lý vậy ta có điều kiện cần.

[<=]. Giả sử <R,F> là 2NF và mọi thuộc tính không khoá không phụ thuộc bắc cầu vào khoá chính của <R,F>. Nếu tồn tại $X \rightarrow A \in F^+$, $A \notin X$, A là thuộc tính không khoá mà X lại không là siêu khoá. Khi đó với K là một khoá của R, ta có A phụ thuộc bắc cầu vào K qua X. Điều này vô lý với giả thiết trên. Ta có điều kiện đủ.

Nhận xét

Qua mệnh đề trên ta đã chứng minh được <R,F> ở 3NF thì ở 2NF.

Định nghĩa 8.6. (dạng chuẩn Boye-Codd - BCNF)

Lược đồ quan hệ <R,F> được gọi là ở dạng chuẩn BCNF nếu với mọi $X \rightarrow A \in F^+$, $A \notin X$ thì X phải là siêu khoá.

Nhận xét

Từ định nghĩa của 3NF và BCNF, dễ thấy mọi lược đồ quan hệ ở BCNF cũng là 3NF.

Lưu ý: Điều ngược lại thì không! Chẳng hạn lược đồ quan hệ <{C,S,Z,}, {CS→Z,Z→C}> có hai khoá là {SC} và {SZ}. Do đó tất cả các thuộc tính đều là thuộc tính khoá và lược đồ quan hệ này ở 3NF. Nhưng lược đồ này không là BCNF vì ta có Z→C ∈F nhưng {Z} không siêu khoá.

Mệnh đề 8.3.

Cho <R,F>, với F={L_i→R_i, i=1,...,m} $R_i \not\subset L_i$, tức trong F không chứa các phụ thuộc hàm tầm thường. Ta có

<R,F> là BCNF nếu $(L_i)^+$ =R với mọi i=1,...,m.

Chứng minh:

[=>]. Giả sử <R,F> là BCNF , khi đó rõ ràng $(L_i)^+$=R.

[<=]. Giả sử <R,F> thoả $(L_i)^+$ =R với mọi i=1,...,m, nhưng nó không ở BCNF, tức tồn tại X→A∈F^+ , A∉X nhưng X không là siêu khoá, tức X^+≠R. Điều vô lý xảy ra ở chỗ là: X→A∈F^+ thì A∈X^+, do A∉X, nên trong quá trình tính bao đóng của X, ta phải tồn tại i: $L_i \subseteq X$ còn A∈R_i , như vậy X→L_i ∈F^+, nhưng L_i là siêu khoá nên X phải là siêu khoá. Vô lý với giả thiết X không là siêu khoá. Tóm lại <R,F> phải là BCNF.

Nhận xét

Như vậy bài toán kiểm tra xem một lược đồ quan hệ là BCNF hay không có độ phức tạp đa thức.

Ví dụ

Xác định và giải thích dạng chuẩn cao nhất của LĐQH sau:

p = (U, F); U = ABCD, F = {A → C, D → B, C → ABD}

LĐQH p có 2 khoá là A và C vì $A^+ = C^+ = ABCD = U$. Tập thuộc tính khoá là $U_k = AC$, tập thuộc tính không khoá là $U_0 = BD$. Ta có, $D \rightarrow B$, B là thuộc tính không khóa và D không phải là một siêu khóa do đó p không ở 3NF và đương nhiên không ở BCNF. Vì hai khoá A và C đều chỉ có một thuộc tính nên các thuộc

tính không khoá khác là *B* và *D* đều phụ thuộc đầy đủ vào hai khoá này. Vậy p ở 2NF.

Ví dụ

1. Cho <U,F>. Gọi H là tập các thuộc tính khoá. Giả sử H=U (tức mọi thuộc tính trong U là thuộc tính khoá). Hãy xác định dạng chuẩn của lược đồ.

2. Cho <U,F> là 3NF. Gọi R là tập các thuộc tính chỉ nằm ở vế phải các phụ thuộc hàm trong F. Giả sử <U,F> thoả $(U\backslash R)^+ = U$. Chứng tỏ lược đồ quan hệ này ở dạng BCNF.

Giải

Từ giả thiết $(U\backslash R)^+ = U$, nên lược đồ có khoá duy nhất là K=U\R (1). Trước hết thấy rằng, do <U,F> là 3NF nên khi $X \rightarrow A \in F$ thì ta phải có A là thuộc tính khoá, hay $A \in K$.

Giả sử <U,F> không ở BCNF => $\exists (X \rightarrow A) \in F^+$ sao cho X không là siêu khoá. Dễ thấy $Y=X(K\backslash\{A\})$ là một siêu khoá (vì $X \rightarrow A$; $K\backslash\{A\} \rightarrow K\backslash\{A\}$), nên Y phải chứa một khoá K' mà K' không chứa A. Tức $K' \neq K$. Vô lý với (1), tức lược đồ đã cho có khoá duy nhất.

3. Cho U={ABCD}; F={A→B; B→A; B→CD}

Xác định dạng chuẩn của <U,F>.

8.3. Chuẩn hóa 3NF

Khi thiết kế CSDL quan hệ, nếu cho môt lược đồ quan hệ <R,F>, ta mong muốn xác định một phân tách $\rho=(R_1,...,R_m)$ sao cho ρ thoả 3 tính chất sau:

i) Có kết nối không tổn thất

ii) Bảo toàn tập phụ thuộc hàm F

iii) Mọi lược đồ con $R_1,...,R_m$ đều ở 3NF.

Một phân tách như thế sẽ tương đối đảm bảo các yêu cầu ta đưa ra trong phần phân tách lược đồ quan hệ.

8.3.1. Thuật toán: (Tổng hợp về dạng chuẩn 3NF và bảo toàn tập F)

Vào: Lược đồ quan hệ <R,F> với giả thiết F là một phủ tối thiểu.

Ra: Một phân tách của R bảo toàn F sao cho mỗi lược đồ con đều ở 3NF đối với hình chiếu của F lên lược đồ con đó.

Phương pháp:

b1) Nếu có một thuộc tính nào đó của R không có mặt trong mọi FD thuộc F thì về nguyên tắc những thuộc tính nào đó làm thành một lược đồ quan hệ và có thể loại nó ra khỏi R.

b2) Nếu một trong các phụ thuộc hàm chứa tất cả các thuộc tính thuộc R, ta cho ra chính R.

b3) Trường hợp ngược lại, các phân tách được cho ra sẽ bao gồm các lược đồ XA ứng với mỗi phụ thuộc $X \rightarrow A$ trong F. Tuy nhiên nếu $X \rightarrow A_1$, $X \rightarrow A_2$,.., $X \rightarrow A_p$, đều thuộc F, ta có thể dùng lược đồ $XA_1...A_p$ thay cho các lược đồ XA_i với i=1...p.

Mệnh đề 8.4.

Thuật toán 8.3.1. là đúng, tức các lược đồ con được tách ra là 3NF và phép tách là bảo toàn phụ thuộc F.

Chứng minh:

Vì các phụ thuộc hình chiếu là chứa các phụ thuộc hàm trong F, nên phép tách là bảo toàn F.

Tiếp theo ta chứng minh rằng các lược đồ XA ứng với mỗi phụ thuộc $X \rightarrow A$ mà chúng ta đã xây dựng là ở 3NF.

Giả sử $Y \rightarrow B$ vi phạm 3NF trong XA. Tức là $B \notin Y$, Y không là siêu khoá của XA, B là thuộc tính không khoá. Ta thấy $YB \subseteq XA$ và $Y \rightarrow B$ được suy ra từ F.

Xét hai trường hợp:

+ B=A. Do $B \notin Y$, mà $Y \subseteq X$, do Y không là siêu khoá của XA, nên Y phải là tập con thực sự của X (vì nếu Y=X, do $Y \rightarrow B$ và $X \rightarrow A$ => Y là siêu khoá của XA ngay). Lúc này ta có $Y \rightarrow A$ (do A=B), tức A phụ thuộc không đầy đủ vào X. Điều này vô lý với giả thiết F là tối thiểu.

+ $B \neq A$. Do X là siêu khoá của XA, nên tồn tại $Z \subseteq X$ là khoá của XA. Do B thuộc X (vì $B \in \{XA\}$, $B \neq A$) , hơn nữa $B \notin Z$ vì B là thuộc tính không khoá. Do đó Z là tập con thực sự của X, do Z là khoá nên $Z \rightarrow A$, điều này cho thấy A phụ thuộc không đầy đủ vào X. Vô lý giả thiết F là tối thiểu.

Tóm lại theo nguyên lý phản chứng, ta có các lược đồ con được tách ra trong thuật toán là 3NF.

Định lý 8.1.

Gọi p là phân tách về dạng chuẩn 3NF và bảo toàn tập F được xây dựng bởi thuật toán 8.3.1 và gọi X là một khoá của R. Khi đó $\tau = \{X\} \cup p$ là một phân tách của R trong đó mọi lược đồ con đều là 3NF; phân tách đó bảo toàn tập F và có kết nối không tổn thất.

Chứng minh:

Dễ chứng minh rằng mọi phụ thuộc bắc cầu hay phụ thuộc một phần trong X đều kéo theo kết quả là một tập con thực sự của X xác định hàm X, và do đó xác định hàm R và như vậy mâu thuẫn với việc X là khoá của R.

Vậy X cũng như tất cả các lược đồ con trong p đều 3NF.

Rõ ràng τ bảo toàn tập F vì p có tính chất đó.

Để chứng minh τ có kết nối không tổn thất, ta áp dụng thuật toán kiểm tra tách không mất thông tin và chứng tỏ rằng với hàng tương ứng với X – hàng cuối cùng gồm toàn ký hiệu a. Xét thứ tự $A_1,.., A_k$ theo đó các thuộc tính của R\X được thêm vào X^+ trong quá trình tính bao đóng X^+. Ta chứng minh bằng quy nạp theo i rằng cột tương ứng với A_i trong hàng tương ứng với X sẽ nhận được giá trị a_i khi thực hiện thuật toán kiểm tra trên.

Bước cơ sở, i=0 là rõ ràng

Giả sử kết quả đúng với i-1. Khi đó A_i sẽ được thêm vào X^+ vì có một phụ thuộc hàm đã cho $Y \rightarrow A_i$, trong đó $Y \subset X \cup \{A_1,.., A_{i-1}\}$

Khi đó YA_i có mặt trong p và các hàng tương ứng với YA_i và X giống nhau trên Y (chúng đều là a) sau khi các cột tương ứng với X đối với $A_1,.., A_{i-1}$ đã được cho bằng a. Như vậy, các hàng này được làm giống nhau trên A_i trong khi thực hiện thuật toán kiểm tra tách không mất thông tin. Vì hàng ứng với YA_i có a_i ở cột Y, hàng tương ứng với cột A_i cũng vậy. Tức cột A_i ở hàng ứng với X cũng được điền a_i. Tóm lại theo giả thiết qui nạp ta có hàng ứng với X sẽ được điền toàn a_i. Tức phép phân tách trên là có kết nối không tổn thất.

Ví dụ 1

Cho lược đồ quan hệ R(CTHRSG) với tập phụ thuộc hàm tối thiểu $C \rightarrow T$, $HR \rightarrow C$, $HT \rightarrow R$, $CS \rightarrow G$, $HS \rightarrow R$. Áp dụng thuật toán III.3.1 ta có tập lược đồ ở dạng 3NF: $R_1(CT)$, $R_2(HRC)$, $R_3(HTR)$, $R_4(CSG)$, $R_5(HSR)$.

Ví dụ 2

Cho lược đồ quan hệ R xây dựng trên tập thuộc tính U={A_1B_1, C_1, B_2, C_2, D, E, I_1, I_2, I_3 , J} và tập các phụ thuộc hàm

$F = \{A \rightarrow B_1\ B_2\ C_1C_2\ DEI_1I_2I_3J$

$B_1\ B_2\ C_1 \rightarrow AC_2DEI_1I_2I_3J$

$B_1\ B_2\ C_2 \rightarrow AC_1DEI_1I_2I_3J$

$E \rightarrow I_1I_2I_3$

$C_1D \rightarrow J$

$C_2D \rightarrow J$

$I_1I_2 \rightarrow I_3$

$I_2I_3 \rightarrow I_1$

$I_3I_1 \rightarrow I_2\ \}$

Phân tách R thành các lược đồ con ở 3NF, thoả mãn kết nối không mất thông tin và bảo toàn phụ thuộc hàm F.

Giải:

Bước 1: (Tìm phủ phụ thuộc hàm tối thiểu của F)

Bước 1.1. Tách vế phải

$A \rightarrow B_1$	$B_1\ B_2\ C_1 \rightarrow A$	$I_1I_3 \rightarrow I_2$
$A \rightarrow B_2$	$B_1\ B_2\ C_2 \rightarrow C_1$	———
…	…	$I_2I_3 \rightarrow I_1$
$A \rightarrow J$	$B_1\ B_2\ C_2 \rightarrow J$	$I_1I_2 \rightarrow I_3$
$B_1\ B_2\ C_1 \rightarrow A$	———	gồm 9 nhóm
$B_1\ B_2\ C_1 \rightarrow C_2$	$E \rightarrow I_1$	
$B_1\ B_2\ C_1 \rightarrow D$	$E \rightarrow I_2$	
…	$E \rightarrow I_3$	
$B_1\ B_2\ C_1 \rightarrow J$	———	
	$C_1D \rightarrow J$	
	———	
	$C_2D \rightarrow J$	

Bước 1.2. Thu gọn trái

Không cần làm bước này, vì khi loại đi bất cứ một trong các thuộc tính vế trái thì không xác định được vế phải.

Bước 1.3: (Loại bỏ phụ thuộc hàm dư thừa)

- Trong nhóm 1, có thể loại bỏ $A \rightarrow I_1$

$A \rightarrow I_2$

$A \rightarrow I_3$

$A \rightarrow J$

Nhóm 2 còn $\{B_1B_2C_1 \rightarrow A\}$

Nhóm 3 còn $\{B_2B_1C_2 \rightarrow A\}$

Các nhóm còn lại để nguyên.

Bước 2: (gộp vế trái và vế phải các phụ thuộc hàm và gộp các phụ thuộc hàm có vế trái chung)

$R_1 = \{A\ B_1\ B_2\ C_1\ C_2\ DE\}$

$R_2 = \{B_1\ B_2\ C_1\ A\}$

$R_3 = \{B_1\ B_2\ C_2\ A\}$

$R_4 = \{E\ I_1\ I_2\ \}$

$R_5 = \{GDJ\}$

$R_6 = \{C_2DJ\}$

$R_7 = \{\ I_1\ I_2\ I_3\ \}$

$R_8 = R_7$

$R_9 = R_7$

Bước 3: Tìm khóa $K = \{A\} \subset R_1$

Vì vậy ta tách được các lược đồ con như sau:

$R_1 = \{AB_1B_2C_1C_2DE\}$

R_2, R_3, R_4, R_5, R_6, R_7 như trên

Bước 4: Với các FD trong từng lược đồ là:

R_1 có $F_1=\{A \rightarrow B_1\ B_2\ C_1\ C_2\ DE\}$

R_2 có $F_2=\{\ B_1\ B_2\ C_1 \rightarrow A\}$

R_3 có $F_3=\{\ B_1\ B_2\ C_2 \rightarrow A\}$

R_4 có $F_4=\{E \rightarrow I_1I_2\}$

R_5 có $F_5=\{C_1D \rightarrow I\}$

R_6 có $F_6=\{C_2D \rightarrow I\}$

R_7 khóa là $\{I_1I_2\}$; $\{I_2I_3\}$; $\{I_1I_3\}$

Lưu ý:

Có thể gộp R_1, R_2, R_3 làm một

$S = (AB_1B_2C_1C_2DE)$

$F = \{ AB_1B_2C_1C_2 \rightarrow DE$

$K = AB_1B_2C_1C_2 \}$

8.4. Phân tách BCNF

Trước hết ta có một số mệnh đề cơ sở sau:

Mệnh đề 8.5. Giả sử R là một lược đồ quan hệ với phụ thuộc hàm F.

$\rho = (R_1, R_2,..., R_n)$ là phép tách của R có nối không mất thông tin đối với F. Đặt $\sigma = (S_1,S_2)$ là phép tách không mất thông tin của R_1 với phụ thuộc hàm $\pi_{R1}(F)$ thì phép tách R thành $(S_1,S_2,R_2,..., R_n)$ cũng có nối không mất thông tin đối với F.

Chứng minh: Lấy hiện hành quan hệ r của R và chiếu lên R_1, R_2,..., R_n để nhận được các quan hệ r_1, r_2,..., r_n. Tiếp theo chiếu r_1 lên S_1, S_2 để nhận được s_1 và s_2. Do giả thiết cả 2 phép tách không mất thông tin nên từ s_1 và s_2 ta có thể phục hồi lại r_1 và từ r_1, r_2,..., r_n phục hồi được r.

Từ mệnh đề ta đi đến thuật toán tách một lược đồ quan hệ thành BCNF.

Nếu tìm thấy một phụ thuộc hàm $X \rightarrow A$ trong F không thỏa mãn điều kiện BCNF thì tách R thành hai lược đồ quan hệ R-A và XA. Chúng ta có R\A và XA là các tập con thực sự của R, XA không thể trùng với R.

Vì $(R- A)\cap XA = X$ và $X \rightarrow XA$ nên theo định lý Delobel chúng ta có phép tách R thành $R_1 = XA$ và $R_2 = R\backslash A$ là không mất thông tin đối với F.

Tìm chiếu của F lên R\A và XA và áp dụng phép tách như trên cho các lược đồ con này để nhận được BCNF.

Thuật toán (***Phân tách R thành các lược đồ con thoả BCNF và có kết nối không mất thông tin***).

Vào: R = <U, F>

Ra: $\rho = (R_0, R_1, R_2, \ldots, R_k)$ bảo toàn thông tin và $R_i \in$ BCNF

Phương pháp: Xây dựng cây phân tách theo các thủ tục sau:

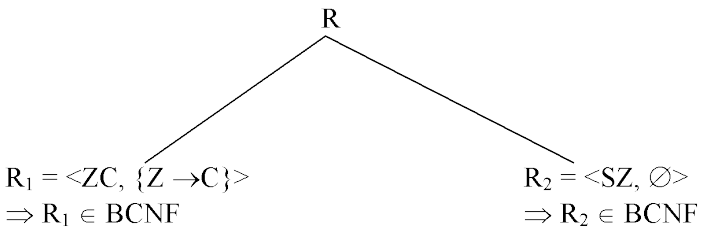
```
Procedure   BCNF_PTH (U, F);
Begin
        If ∃X→A ∈ F+ với A ∉ X với X  không là siêu khoá then
        Begin
                BCNF_PTH(XA, ∏XA(F));
                BCNF_PTH(U\A, ∏U\A(F));
        End;
End;
```

Nhận xét: Trong thuật toán trên ta phải tìm chiếu của các phụ thuộc hàm trên các lược đồ con.

Ví dụ:

Cho R = <U, F>, với U = CSZ và F = {CS→Z, Z→C}.

$R \notin$ BCNF vì $Z \to C \in F^+$, Z không là siêu khoá (tập khoá: {CS, ZS}). Do vậy ta phân tách R thành 2 lược đồ con như sau:



Mệnh đề 8.6.

i. Mọi lược đồ quan hệ gồm hai thuộc tính thì ở dạng BCNF.

ii. Lược đồ <R,F> với tập F chỉ gồm một phụ thuộc hàm thì ở dạng BCNF.

iii. Nếu <R,F> không ở BCNF thì tồn tại hai thuộc tính A,B sao cho

$$(R \backslash (A \cup B)) \to A$$

Chứng minh:

i) Giả sử lược đồ chỉ gồm hai thuộc tính A,B, khi đó chỉ có $A \rightarrow B$ hoặc $B \rightarrow A$ hoặc $AB \rightarrow A$ hoặc $AB \rightarrow B$ thì hoặc A là khoá, hoặc B là khoá, nên i) là hiển nhiên.

ii) Nếu $F=\{X \rightarrow Y\}$ suy ra X là một siêu khoá nên ii) là hiển nhiên.

iii) Nếu R không ở BCNF khi đó tồn tại $X \rightarrow A$ và $A \notin X$ mà X không là siêu khóa. Từ đây ta có $R \setminus (A \cup X) \neq \emptyset$, nên tồn tại $B \in R \setminus (A \cup X)$. Khi đó $X \subseteq R \setminus (A \cup B) \Rightarrow R \setminus (A \cup B) \rightarrow X \rightarrow A$. Theo tính chất bắc cầu ta có: $R \setminus (A \cup B) \rightarrow A$.

Mệnh đề 8.7.

Cho <R,F> , với $R_1 \subseteq R$ và $R_2 \subseteq R_1$.

Chiếu F lên R_1 ta được F_1, tiếp tục chiếu F_1 lên R_2 để được F_2.

Khi đó ta có $F_2 = \Pi_{R2}(F)$.

Chứng minh:

$F_1 = \Pi_{R1}(F) = \{X \text{ -> } Y | (X \text{ -> } Y) \in F^+ \text{ và } XY \subseteq R_1\}$

$F_2 = \Pi_{R2}(F_1) = \{X \rightarrow Y \mid (X \rightarrow Y) \in F^+_1 \text{ và } XY \subseteq R_2\}$

$F_{R2} = \Pi_{R2}(F) = \{X \rightarrow Y \mid (X \rightarrow Y) \in F^+ \text{ và } XY \subseteq R_2\}$

Do $F_1^+ \subseteq F^+$ nên ta thấy ngay $F_2 \subseteq F_{R2}$, ngược lại do $R_2 \subseteq R_1$, nên với $XY \subseteq R_2$, $X \rightarrow Y \in F^+$ thì $X \rightarrow Y \in F^+_1$, nên $F_{R2} \subseteq F_2$. Nên $F_2 = F_{R2}$.

8.4.1. Thuật toán (phân tách R thành các lược đồ con ở BCNF)

Tư tưởng của phương pháp này là phụ thuộc hàm $X \rightarrow A$ tách lược đồ quan hệ R thành 2 lược đồ quan hệ: Một quan hệ XA với quan hệ thứ hai là $R \setminus A$. Như vậy nối $R \setminus A$ với XA là không mất thông tin. Tiếp tục áp dụng một cách đệ qui bằng thay $R \setminus A$ cho R đến khi chúng ta gặp lược đồ thỏa mãn điều kiện của mệnh đề 8.6.

Thuật toán

Vào: Lược đồ quan hệ R và tập phụ thuộc hàm F trên R.

Ra: Phép tách $\rho=(R_1,.., R_k)$ của R ứng với F sao cho mỗi R_i (i=1,..,k) là BCNF và ρ là phép tách có nối không mất thông tin.

Phương pháp:

(* *Chương trình chính* *)

Z:= R; ρ = ∅;

Repeat

Tách Z thành Z\{A} và XA, trong đó XA là BCNF và X→A; (* *sử dụng thủ tục tách được trình bày ở dưới* *)

ρ:=ρ ∪ {XA};

Z:=X\{A};

Until không thể tách được Z;

ρ:=ρ ∪ {Z};

(* *Thủ tục tách Z thành XA và Z\{A}* *)

If không tồn tại A,B ∈Z sao cho A∈$(Z\{AB\})_F^+$ **Then**

Return Z là BCNF và không thể tách;

Tìm cặp A,B∈Z sao cho A∈$(Z\{AB\})_F^+$;

Y:=Z\{B};

While còn tồn tại A,B ∈ Y sao cho A∈$(Y\{AB\})_F^+$ **do**

Y:= Y\{B};

Return Tách Z thành Z\{A} và Y ; (* *Với Y đóng vai trò là XA trong chương trình chính* *)

Nhận xét: Với phương pháp này ta không cần tính chiếu của các phụ thuộc hàm trên các lược đồ con.

Ví dụ 1:

Cho lược đồ R(CTHRSG) có

F={ C→T - Mọi giáo trình (course) có một giáo viên (teacher)

HR→C – Tại 1 phòng học (Room), vào một giờ (Hour) nhất định có một chuyên đề.

HT→R – Mọi thầy giáo ở một giờ nhất định ở một phòng nhất định.

CS → G – Mọi sinh viên (Student) học cùng một chuyên đề thì xem cùng một nhóm (Group)

HS→R – Mọi sinh viên chỉ ở 1 phòng vào 1 giờ nhất định.

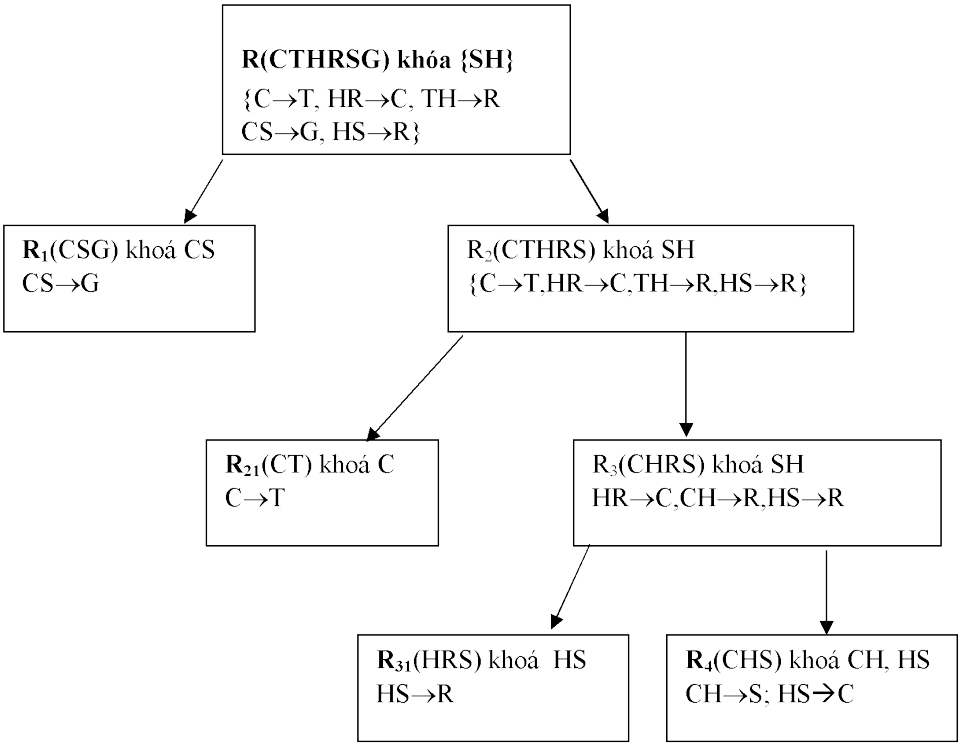
Ta thấy HS là khoá của R.

Tách R thành các lược đồ BCNF.

Xét CS→G phụ thuộc này vi phạm điều kiện BCNF vì CS không chứa khoá.

Áp dụng thuật toán ta tách R=R_1(CSG) ∪R_2(CTHRS), bước tiếp theo tính F^+ và chiếu xuống R_1 và R_2 sau đó kiểm tra xem lược đồ đã ở BCNF chưa. Cứ tiếp tục làm như vậy.

Sơ đồ biểu diễn



Kết luận: R tách thành **R_1, R_{21}, R_{31}, R_4** với các ràng buộc tương ứng là BCNF và tách có kết nối không mất mát thông tin.

Ví dụ 2:

Dưới đây chúng ta cho ví dụ minh họa việc phân tách một bảng (quan hệ) thành các bảng ở dạng chuẩn 3 NF.

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 hóa đơn, đã 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	TP. HCM	30101
1170003	17/03/97	Trần Hà	14 Lê Lợi	TP. HCM	10100
1170003	17/03/97	Trần Hà	14 Lê Lợi	TP. HCM	20904

Trong quan hệ phiếu xuất kho ta nhận thấy là tập {Số phiếu, Mã vật tư} là khóa tối thiểu. 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 lý 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

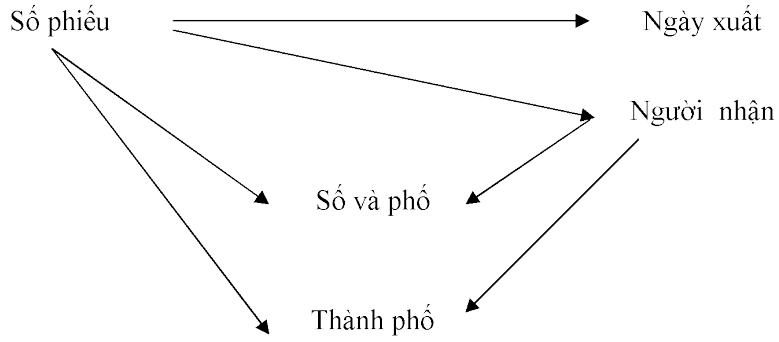
1020004	12/01/97	Trần Hà	14 Lê Lợi	TP. HCM
1170003	17/03/97	Trần Hà	14 Lê Lợi	TP. HCM

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 lý sẽ dẫn đến trùng lặp hai địa chỉ của người nhận. Do vậy ta tách nó thành 2 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.

8.4.2. Các vấn đề nảy sinh khi phân rã BCNF tuỳ tiện và một số nhắc nhở

Trong ví dụ trên ta đã phân rã ra các lược đồ con sau

a) Phòng học của mỗi sinh viên tại mỗi giờ (HRS)

b) Giảng viên của mỗi lớp (CT)

c) Chuyên đề của từng sinh viên theo từng lớp (CSG)

d) Sinh viên tại từng thời điểm học chuyên đề nào (CHS)

Ta thấy rằng trong quá trình phân tách trên, ta chọn phụ thuộc hàm X→A một cách ngẫu nhiên, nên các lược đồ con tách ra có thể khác nhau. Như vậy kết quả khi kết nối có thể khác nhau! Cụ thể ở đây ta không thể biết phòng học của một nhóm (group) nếu không nối CHS và HRS lại. Hơn nữa có thể thay HRS bằng CHR và ta có một kết quả phân rã khác. Phân rã để thu hẹp công việc bao nhiêu thì tốn công kết nối bấy nhiêu ("*được lợi bao nhiêu lần về lực thì thiệt hại bấy nhiêu lần về đường đi*")

Hơn nữa một số phụ thuộc trong F như TH→R và HR→C không được bảo toàn qua phép phân rã này. Tức hình chiếu của F lên HRS, CT, CSG và CHS là bao đóng của tập các phụ thuộc sau CS→G; HS→R; C→T; và HS→C

Để ý HS→C thuộc hình chiếu của F trên HSC, nhưng không phải là của F cho trước. Bốn phụ thuộc hàm trên không suy ra được TH→R và HR→C, Xét

quan hệ dưới đây không thoả TH→R và HR→C, nhưng hình chiếu của nó trên HRS, CT, CSG và CHS thoả tất cả các phụ thuộc được chiếu.

C	T	H	R	S	G
c_1	t	h	r_1	s_1	g_1
c_2	t	h	r_2	s_2	g_2
c_2	t	h	r_1	s_3	g_3

Điều cần nhớ là không phải mọi phân rã có nối không mất thông tin đều có ích. Một sai lầm là đi phân tách các lược đồ đã có dạng BCNF. Chẳng hạn ta có lược đồ trên tập thuộc tính U={INDS} với I là khoá (mã số) ; N (name); D(Department); S (Salary) . Với tập phụ thuộc hàm là F={I→ N; I→S; I→D}.

Có thể phân rã lược đồ thành IN, ID và IS. Phân rã trên là không mất thông tin và bảo toàn phụ thuộc I→DSN. Nếu phân rã như thế khi muốn "*biết tên và lương của các nhân viên ở một Department nào đấy*" ta phải kết nối cả 3 lược đồ IN, ID, IS và mã số nhân viên phải được lặp đi lặp lại trong cả 3 lược đồ con tách ra ở đây.

Trong khi bản thân lược đồ này là BCNF thành ra mã số I là không dư thừa và việc trả lời câu hỏi nêu trên ở trên chính nó lại rất đơn giản!.

Nên nhớ rằng phân rã chỉ là ***cứu cánh*** để giải quyết dư thừa và bất thường chứ không là mục đích. Nên tránh phân tách các lược đồ đã BCNF và nên tổ hợp các lược đồ con lại khi phân tách 3NF, nếu không xảy ra vi phạm gì về 3NF.

8.4.3. Một số bài toán liên quan đến khóa và các dạng chuẩn (xem [9]).

8.4.3.1. Các lớp P và NP

Ký hiệu P là lớp các bài toán có thể giải bằng một thuật toán *đơn định với thời gian đa thức*, NP là lớp các bài toán có thể giải bằng một thuật toán *không đơn định với thời gian đa thức*. Bài toán mở hiện nay là $P = NP$? Bài toán T được gọi là *NP-khó* nếu $T \in P \Rightarrow P=NP$. Bài toán T được gọi là *NP-đầy đủ* nếu T là *NP-khó* và $T \in NP$. Cho đến nay chưa tìm được thuật toán đơn định với thời gian đa thức cho bất kỳ bài toán *NP-khó* hoặc *NP-đầy đủ* nào. Hơn nữa, nếu có một thuật toán như vậy thì đó chính là một chứng minh cho đẳng thức $P=NP$.

8.4.3.2.Các bài toán sau đây thuộc lớp NP-đầy đủ

P1. Bài toán phi BCNF. Cho LĐQH a=(U,F). Có tồn tại hay không một họ LĐQH ở dạng chuẩn BCNF không mất thông tin và bảo toàn tập PTH F đối với a?

P2. Bài toán thỏa BCNF. Cho LĐQH a=(U,F) và một họ LĐQH s không mất thông tin đối với a. Họ s có ở dạng BCNF hay không? P2 là bài toán NP-đầy đủ ngay cả trong trường hợp họ s đã ở dạng 3NF hoặc s là kết quả của thuật toán chuẩn hóa 3NF.

P3. Bài toán khóa bổ sung. Biết trước một số khóa $\{K_1,K_2,...,K_p\}$, $p \geq 0$ của LĐQH a = (U,F). Cho biết a còn khóa nào nữa không?

P4. Bài toán liệt kê khóa. Liệt kê toàn bộ các khóa của LĐQH a = (U,F)?

P5. Bài toán khóa lực lượng m. Cho LĐQH a = (U,F) và một số nguyên m>1. Cho biết trong a có tồn tại một khóa với lực lượng nhỏ hơn m không?

P6. Bài toán thuộc tính khóa. Cho LĐQH a = (U,F) và một thuộc tính $A \in U$. Cho biết A có mặt trong một khóa nào không?

Các bài toán sau đây thuộc lớp NP-khó:

P7. Bài toán BCNF. Cho LĐQH a=(U,F). Có thể chuẩn hóa a về dạng BCNF không mất thông tin và bảo toàn tập PTH F hay không?

Chú ý rằng bài toán P7 và bài toán P1 là khác nhau bởi vì ta có thể tách một LĐQH thành hai lược đồ con, trong đó có một lược đồ ở dạng BCNF.

Bài tập chương 8.

1. Cho U={ABCD}; F={AB→C; B→D; BC→A}

a) Tìm tất cả các khoá của <U,F>

b) Xác định dạng chuẩn của lược đồ

c) Kiểm tra phân tách p=(ABC, DC, DBA) có kết nối không tổn thất không

d) Tìm phủ tối thiểu

e) Tìm phân tách 3NF thoả có kết nối không tổn thất và bảo toàn phụ thuộc F.

2. Cho U={ABCDEG}; F={B→C; C→B; A→GD}

a) Tìm tất cả các khoá của <U,F>

b) Xác định dạng chuẩn của lược đồ

c) Kiểm tra phân tách p=(AB, BDC, DEB,AEG) có kết nối không tổn thất không

d) Tìm phủ tối thiểu

e) Tìm phân tách 3NF thoả có kết nối không tổn thất và bảo toàn phụ thuộc F.

3. Ở một cửa hàng tổng hợp mỗi ngày có bảng tổng kết việc mua bán hàng như sau:

Ngày tháng	**Mã hàng**	**Tên hàng**	**Đơn giá**	**Số lượng**
190301	*M1*	*Radio*	*1000*	*1*
	M2	*Tivi*	*2000*	*2*
	M5	*Máy giặt*	*3000*	*3*
210301	*M5*	*Máy giặt*	*3100*	*2*
	M2	*Tivi*	*2000*	*1*

Tổng giá tiền: **22200**

Đã thanh toán: ***14200***

a) Hãy xây dựng một lược đồ quan hệ TONGHOP để lưu trữ dữ liệu cho việc mua bán hàng ở cửa hàng trên.

b) Xác định các ràng buộc dạng phụ thuộc hàm trên lược đồ vừa xây dựng.

c) Xác định dạng chuẩn của lược đồ, có nhận xét gì về lược đồ trên.

d) Nên tách lược đồ này thành các lược đồ như thế nào. Sau đó chỉ ra sự liên kết giữa chúng sẽ có được thông tin mà mỗi ngày ta vẫn ghi chép ở bảng tổng kết ngày.

4. 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 $\rightarrow$ C; A $\rightarrow$ 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 3.

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. Cho LĐQH $p = (U, F)$ với tập thuộc tính $U = ABCDEHG$ và tập phụ thuộc hàm $F = \{ DE \rightarrow G,\ H \rightarrow C,\ E \rightarrow A,\ CG \rightarrow H,\ DG \rightarrow EA,\ D \rightarrow B \}$.

a. Tìm tập M là giao của toàn bộ các khoá của p. Cho biết p có đúng 1 khoá hay không?

b. Tìm 1 khoá của p.

c. Tập BCE có phải là khoá của p không? Vì sao?

d. Hãy thêm hoặc bớt 1 phụ thuộc hàm cho F để LĐQH có đúng 1 khoá.

6. Cho F = {AB→C, A→D, BD→C}

a) Tìm phủ cực tiểu của F.

b) Tìm một phân rã của lược đồ ABCD thành 2 lược đồ có dạng 3NF.

c) Tìm các phụ thuộc hình chiếu của mỗi lược đồ tìm được ở câu b.

d) Kết quả câu b có phải là một phân rã có nối không mất hay không? Nếu không ta phải sửa đổi lược đồ như thế nào để có được phân rã bảo toàn phụ thuộc và nối không mất.

7. Cho lược đồ ABCD với F = {A→B, B→C, A→D, D→C}, ρ là một phân rã (AB, AC, BD)

a) Tìm các phụ thuộc hình chiếu cho mỗi lược đồ trong phân rã.

b) ρ có phải là một phân rã có nối không mất thông tin ứng với các phụ thuộc đã cho hay không?

c) ρ có phải là một phân rã bảo toàn các phụ thuộc đã cho hay không?

8. Xét lược đồ quan hệ có các thuộc tính sau: S (Store), D (Department), I (Item), M (Manager) và các phụ thuộc hàm SI→D, SD→M.

a) Tìm tất cả các khóa của SDIM.

b) CMR: SDIM đạt 2NF nhưng không đạt 3NF.

9. Cho lược đồ quan hệ có các thuộc tính sau: Makh, Tenkh, Madondathang, Tenmh, Dvt, Soluong(*số lượng hàng được đặt*), Diachi(*địa chỉ khách hàng*), Ngaydathang và tập phụ thuộc hàm như sau:

Makh → Tenkh, Diachi

Makh, Madondathang → Tenmh, Dvt, Soluong, Ngaydathang

Tenmh → Dvt

Hãy đưa lược đồ trên về dạng BCNF và kiểm tra tính kết nối không mất.

10. Cho CSDL của một công ty hoạt động đầu tư với các thuộc tính sau:

B (Broker, người môi giới); **O** (Office of Broker); **I** (Investor, nhà đầu tư); **S** (Stock, cổ phần); **Q** (Quantity, số lượng cổ phần của nhà đầu tư); **D** (Divident, lãi của mỗi cổ phần).

Với tập ràng buộc F = {S → D, I→B, IS→Q, B→O}

a. Hãy tìm một khóa cho lược đồ quan hệ R = BOSQID

b. Hãy tìm một phân rã thành dạng chuẩn Boyce-Codd có nối không mất.

c. Hãy tìm một phân rã thành dạng chuẩn 3 bảo toàn phụ thuộc và có nối không mất.

d. Nếu ta phân rã lược đồ R thành 2 lược đồ: ISQD và IBO thì phân rã này có đặc tính nối không mất hay không?

e. Nếu ta có một phân rã của lược đồ R như sau: SD, IB, ISQ, ISO. Phân rã này có bảo toàn phụ thuộc hay không?

11. Cho tập các phụ thuộc hàm sau, hãy tìm tập nhỏ nhất các bảng 3NF. Tìm khóa cho các quan hệ này. Tập các quan hệ thu được có đạt BCNF hay không?

a. F = {J→KLMNP, JKL→MNP, K→MQ, KL→MNP, KM→NP, N→KP}

b. F = {A→BCDEF, AB→CDEF, ABC→DEF, ABCD→EF, ABCDE→F, B→DG, BC→DEF, BD→EF, E→BF}

12. Cho CSDL của một công ty vận tải đường thủy có các thuộc tính sau:

S (Ship); **T** (Type of ship); **V** (Voyage identifier: mã số tầu); **C** (Cargo:k.lượng hàng được vận chuyển); **P** (Port: bến cảng); **D** (Day: ngày vận chuyển).

Giả sử chuyến tàu (voyage) lấy một loại hàng và phân phối hàng này cho các bến cảng. Một tàu trong một ngày chỉ được ghé qua 1 cảng. Vì thế chúng ta có thể giả định các phụ thuộc hàm sau: S→T, V→SC, SD→PV.

a. Hãy tìm một phân rã có nối không mất thành dạng BCNF. Kiểm tra tính bảo toàn phụ thuộc và nối không mất của phân rã.

b. Hãy tìm một phân rã thành dạng 3NF có nối không mất và bảo toàn phụ thuộc. Kiểm tra tính bảo toàn phụ thuộc và nối không mất của phân rã.

13. Cho bảng quan hệ r:

r

A	B	C	D
x	u	x	y
y	x	z	x
z	y	y	y
y	z	w	z

Trong các phụ thuộc hàm sau đây, phụ thuộc hàm nào không thỏa mãn r:

A → B, A → C, B → A, C → D, D → C, D → A ?

14.

Bảng 1

Vận động viên (V)	**Môn thể thao (M)**	**Huấn luyện viên (H)**	**Phòng tập (P)**
An	Bóng rổ	Anh	P1
Chi	Bóng rổ	Anh	P1
An	Nhảy cao	Lan	P3
Ba	Bóng rổ	Nga	P1
Nam	Bóng rổ	Anh	P2

Bảng 2

A	**B**
1.VM	a. Khóa
2. HP	b.Khóa bao hàm
3. H	c. Nguyên tố
4. M	

a. Dựa vào các dữ liệu cho trong quan hệ trên (Bảng 1), hãy chỉ ra phụ thuộc hàm nào thỏa mãn, phụ thuộc hàm nào không thỏa mãn. Tại sao? (Giải thích bằng cách chỉ ra các bộ dữ liệu).

a. VM → HP b. HP → VM c. H → M d. M → H

b. Dựa trên các phụ thuộc hàm ở câu 1, trong bảng 2, cho biết mỗi giá trị ở cột A sẽ ứng với những giá trị nào ở cột B.

(VD: H là khóa bao hàm và H là nguyên tố thì kết quả là 3bc).

c. Quan hệ trên (bảng 1) đạt đến dạng chuẩn nào? Tại sao?

d. Phân tách thành các bảng đạt chuẩn BCNF có nối không mất. Sau đó kiểm tra tính bảo toàn phụ thuộc.

e. Phân tách thành bảng đạt chuẩn 3NF bảo toàn phụ thuộc. Sau đó kiểm tra có mất mát thông tin của phân rã đó không?

15. Cho lược đồ R = (ABCDE) và F = {A → BC, B → C, A → B, AB → C, BD → A}.

a. Tìm khóa của R

b. Tìm phủ cực tiểu của F

c. Phân rã (BC, ABDE) có tính chất nối không mất và bảo toàn phụ thuộc không?

d. Tìm phân rã của R đạt 3NF.

16. Cho lược đồ R = (ABCDE) và tập phụ thuộc F = {A → BC, ACD → E, B → D, C → D, AB → E, E → BC}

a. Tìm phủ cực tiểu của F

b. Tìm khóa của R

c. R có đạt 2NF, 3NF không?

17. Cho R = (ABCD) và F = {A → B, C → D}. Với phân rã ρ = (AB, CD) có khẳng định "Phân rã trên bảo toàn phụ thuộc nhưng mất nối". Hãy chứng minh khẳng định trên.

18. Cho R = (ABCD), tập F = {A → C, D → C, BD → A} và phân rã ρ = (AB, ACD, BCD). Phân rã trên có tính chất nối không mất không? R có đạt chuẩn 3NF không?

19. Cho R = (ABCD), tập F = {A → B, B → C, D → B} và phân rã ρ = (ACD, BD). Phân rã trên có bảo toàn phụ thuộc không? R có đạt 2NF, 3NF không?

20. Cho R = (ABCD) và phân rã ρ = (AB, BC, CD). Với mỗi tập phụ thuộc hàm dưới đây, chứng minh rằng phân rã trên có nối không mất.

a. F1 = {A → B, B → C}

b. F2 = {B → C, C → D}

21. Tìm các khóa của mỗi R sau, kiểm tra R đạt BCNF không?

a. R = (CSZ) và F = {CS → Z, Z → C}

b. R = (ABCD) và F = {AB → C, BC → A, B → D}

22. Kiểm tra các phân rã của mỗi R sau có bảo toàn phụ thuộc và có nối không mất không?

a. R = (Student, Time, Room, Course, Grade), F = {Student, Time → Room; Student, Course → Grade} và ρ ({Student, Time, Room}, {Student, Course, Grade})

b. R = (Class, Time, Room), F = {Class → Room; Room, Time → Class} và ρ = ({Class, Room}, {Room, Time})

23. Cho lược đồ R = ABCDEF và tập F = {B→E, E→A, A→D, D→E, C→F, F→C}

a. Liệt kê các khóa dự tuyển của R; danh sách các thuộc tính nguyên tố và các thuộc tính không nguyên tố.

b. Tìm các phụ thuộc hàm vi phạm 3NF (nếu có). R có đạt 3NF không?

c. Tìm phụ thuộc hàm vi phạm BCNF nhưng không vi phạm 3NF. R có đạt BCNF không?

d. Phân rã R thành các lược đồ đạt BCNF (chú ý số lược đồ là ít nhất). Với mỗi quan hệ mới chỉ ra danh sách thuộc tính, các phụ thuộc hàm không tầm thường và danh sách các khóa dự tuyển.

e. Phân rã trên có bảo toàn phụ thuộc không? Nếu không, liệt kê các phụ thuộc hàm mà không được bảo đảm?

24. Cho lược đồ R = ABCDEFGH và tập F = {C→E, D→F, E→D, F→C, G→H, CD→B, EF→G}

a. Tìm phủ cực tiểu của tập F.

b. Tìm khóa dự tuyển của R, liệt kê danh sách thuộc tính nguyên tố và thuộc tính không nguyên tố.

c. Tìm các phụ thuộc hàm (nếu có) trong phủ cực tiểu mà vi phạm 3NF. Lược đồ R có đạt 3NF không?

d. Tìm các phụ thuộc hàm (nếu có) trong phủ cực tiểu mà vi phạm BCNF nhưng không vi phạm 3NF. Lược đồ R có đạt BCNF không?

e. Tìm phân rã của R đạt 3NF nhưng không mất thông tin và bảo toàn phụ thuộc (chú ý số lược đồ là ít nhất). Với mỗi lược đồ mới liệt kê danh sách thuộc tính, các phụ thuộc hàm không tầm thường và danh sách các khóa dự tuyển.

25. Cho R = (CDJPSQV) và tập F = {C → CSJDPQV, SD → P, JP → C, J → S}

a. Tìm phủ cực tiểu của F

b. Tìm phân rã của R đạt 3NF mà bảo toàn phụ thuộc. Phân rã trên có tính chất nối không mất thông tin không?

c. Tìm phân rã của R đạt BCNF có nối không mất thông tin. Phân rã có bảo toàn phụ thuộc không?

26. Cho lược đồ Khóa_học (Môn, Giáo viên, Giờ) và các khẳng định

- Giáo viên A hướng dẫn môn học B vào giờ C
- Một môn học chỉ được hướng dẫn bởi một giáo viên
- Vào một giờ cụ thể một giáo viên chỉ hướng dẫn một môn học

a. Tìm các phụ thuộc hàm theo khẳng định trên

b. Lược đồ trên có đạt BCNF không? Vì sao?

c. Chuyển lược đồ trên về dạng BCNF.

d. Phân rã trên có kết nối mất thông tin không? Vì sao?

e. Phân rã trên có bảo toàn phụ thuộc hàm không? Vì sao?

27. Cho lược đồ R gồm các thuộc tính: Tên chủ hộ, Số thành viên, Số hiệu căn hộ, Địa chỉ, Số phòng, Tên người sở hữu và tập phụ thuộc hàm như sau:

Tên chủ hộ $\rightarrow$ *Địa chỉ*

Tên chủ hộ $\rightarrow$ *Số hiệu căn hộ*

Tên chủ hộ $\rightarrow$ *Số thành viên*

Địa chỉ, Số căn hộ $\rightarrow$ *Tên người sở hữu*

Địa chỉ, Số căn hộ $\rightarrow$ *Tên chủ hộ*

Địa chỉ, Số căn hộ $\rightarrow$ *Số phòng*

a. Tìm khóa của quan hệ R.

b. R có đạt 3NF không? Nếu không, hãy phân rã R thành 3NF bảo toàn phụ thuộc và có nối không mất thông tin.

CHƯƠNG 9. PHỤ THUỘC ĐA TRỊ VÀ PHỤ THUỘC KẾT NỐI

Trên một lược đồ dữ liệu để lược đồ ấy có tính ngữ nghĩa cao, cần phải xác định rõ các ràng buộc toàn vẹn trên nó. Có nhiều phương pháp để biểu diễn ràng buộc toàn vẹn như lôgic tân từ, mạng ngữ nghĩa, đồ thị...Một trong những phương pháp ấy, phương pháp dùng khái niệm phụ thuộc hàm mang hình thức toán học và hiệu quả, nhiều kết quả về thiết kế dữ liệu quan hệ dựa trên nền tảng của lý thuyết phụ thuộc hàm đã được đưa ra. Tuy vậy, trong rất nhiều tình huống thực tế, khái niệm phụ thuộc hàm vẫn không thể biểu diễn hết các ràng buộc toàn vẹn trên cơ sở dữ liệu. Một trong những phương pháp khắc phục đó là đưa ra khái niệm phụ thuộc mới như phụ thuộc đa trị, phụ thuộc kết nối, phụ thuộc suy rộng, phụ thuộc miền-khoá (DK/NF)... Trong chương này chúng ta sẽ bàn đến các khái niệm cơ bản của phụ thuộc đa trị cũng như dạng chuẩn 4NF - dạng chuẩn của cơ sở dữ liệu với phụ thuộc hàm đa trị. Đồng thời, sơ lược một số vấn đề về phụ thuộc kết nối.

Ta xét ví dụ sau:

Ví dụ:

Cho quan hệ HANG HOA.

TENHANG	MAU	NOI SX
Quạt	Xanh	Thai Lan
Quạt	Đen	TQ
Quạt	Xanh	TQ
Quạt	Đen	Thai Lan

Do một mặt hàng có thể có nhiều màu sắc khác nhau và được sản xuất ở những nơi khác nhau nên dữ liệu trong bảng là dư thừa.

Bên cạnh đó quan hệ HANG HOA không thể phân rã ra các quan hệ nhỏ hơn để tránh dư thừa dữ liệu. Thực chất là do phụ thuộc hàm không thể diễn tả hết các ràng buộc trên dữ liệu nên ta không thể tìm hiểu về sự độc lập giữa các thuộc tính. Đó là một trong những lý do của việc đề ra phụ thuộc đa trị.

9.1. Phụ thuộc hàm đa trị (MultiValued Dependency - MVD)

Cho U là tập thuộc tính X,Y$\subset$ U. gọi R là quan hệ có tập thuộc tính U.

Ta nói *X xác định đa trị Y* hay *có một phụ thuộc đa trị của tập Y vào tập X* ký hiệu là X→→Y, nếu với mỗ giá trị của X có một tập rỗng hoặc một tập có giá trị tương ứng trên Y mà không liên quan gì đến các giá trị của các thuộc tính còn lại của U\(XY). Có thể hiểu đó là: tính độc lập của Y liên quan với giá trị X.

9.1.1. Một số định nghĩa

Định nghĩa 9.1. (Phụ thuộc đa trị)

Cho lược đồ quan hệ R, X, Y là hai tập con của R, Z=R-XY. Quan hệ r(R) thoả phụ thuộc đa trị X→→Y nếu với bất kỳ hai bộ $t_1 \in r$ và $t_2 \in r$ với $t_1[X]=t_2[X]$ tồn tại một bộ $t_3 \in r$ sao cho $t_3[X]= t_1[X]$; $t_3[Y]= t_1[Y]$; $t_3[Z]=t_1[Z]$.

Do tính chất đối xứng của t_1 và t_2 dễ dàng thấy rằng trong r còn tồn tại một bộ t_4 sao cho $t_4[X]= t_1[X]$; $t_4[Y]= t_2[Y]$; $t_4[Z]= t_1[Z]$.

Các trường hợp đặc biệt:

- Nếu Y= ∅ thì X→→∅ đúng với mọi quan hệ.

- Nếu X=∅ thì ∅→Y thoả trên một quan hệ khi và chỉ khi tập giá trị trên tập thuộc tính Y là độc lập với các phụ thuộc tuyến tính còn lại trong quan hệ.

- Gọi r[YZ] là một quan hệ, Y∩Z=∅ thì ∅→→Y thoả trên r khi và chỉ khi r=r[Y]*r[Z].

Ví dụ: Cho quan hệ

R(C	T	H	P	S	G)
101	J	M	2	K	B.....t_1
101	J	W	3	Z	C.....t_2
101	J	M	2	Z	C.....t_3
101	J	W	3	K	B.....t_4
101	J	F	2	Z	C.....

Quan hệ R có phụ thuộc đa trị C→→HP

Theo định nghĩa ta có $t_1,t_2 \in r$ với $t_1[C]=t_2[C]=101$ khi đó tồn tại bộ $t_3 \in r$ sao cho $t_3[C]=t_1[C]=101$, $t_3[HP]=t_1[HP]=(M,2)$, $t_3[TSG]=t_2[TSG]=(J,Z,C)$.

Định nghĩa 9.2. (phụ thuộc đa trị cơ sở).

Phụ thuộc đa trị X→→Y của quan hệ R được gọi là phụ thuộc đa trị cơ sở nếu thoả các tính chất sau:

i. $Y \neq \emptyset$ và $X \cap Y = \emptyset$.

ii. Tồn tại duy nhất $X' \subseteq X$ và $Y' \subset Y = \emptyset$ mà $X' \rightarrow\rightarrow Y'$ là một phụ thuộc đa trị của quan hệ R.

Ví dụ:

Cho quan hệ R (A B C D E F)

a	b	c_1	d_1	e_1	f_1
a	b	c_2	d_2	e_2	f_2
a	b	c_1	d_1	e_2	f_2
a	b	c_2	d_2	e_1	f_1

Trong quan hệ trên có $AB \rightarrow\rightarrow CD$, $A \rightarrow\rightarrow CD$ nhưng $A \rightarrow\rightarrow C$ và $A \not\rightarrow\rightarrow D$. Trong đó $AB \rightarrow\rightarrow CD$ không là phụ thuộc đa trị cơ sở.

Bổ đề 9.1. Với mọi thuộc tính X,Y và U sao cho $X,Y \subseteq U$, phụ thuộc đa trị $X \rightarrow\rightarrow Y$ thoả trên quan hệ r(U) khi và chỉ khi $X \rightarrow\rightarrow Y\text{-}X$ thoả trên r(U).

Định nghĩa 9.3. (Phụ thuộc đa trị dị thường)

Cho lược đồ quan hệ R, một phụ thuộc đa trị $X \rightarrow\rightarrow Y$ được gọi là dị thường nếu $Y \neq \emptyset, Y \not\subseteq X$ và $XY \neq U$.

9.1.2. Hệ tiên đề cho phụ thuộc đa trị

A1. (luật bù cho phụ thuộc đa trị): Nếu $X \rightarrow\rightarrow Y$ thì $X \rightarrow\rightarrow U\text{-}X\text{-}Y$.

A2. (luật tăng trưởng cho phụ thuộc đa trị):

Nếu $X \rightarrow\rightarrow Y$ và $V \subseteq W$ thì $WX \rightarrow\rightarrow VY$

A3. (luật bắc cầu cho phụ thuộc đa trị): $X \rightarrow\rightarrow Y$ và $Y \rightarrow\rightarrow Z$ thì $X \rightarrow\rightarrow Z\text{-}Y$

Các luật phối hợp giữa phụ thuộc hàm và phụ thuộc đa trị:

A4. Nếu $X \rightarrow Y$ thì $X \rightarrow\rightarrow Y$.

A5. Nếu $X \rightarrow\rightarrow Y$, $Z \subseteq Y$ và W là hai tập con phân biệt với $Y (W \cap Y = \emptyset)$ và $W \rightarrow\rightarrow Z$ thì $X \rightarrow\rightarrow Z$.

Định lý 9.1. Hệ tiên đề từ $A_1, .., A_4$ là đúng và đầy đủ cho các phụ thuộc hàm và phụ thuộc đa trị.

Chứng minh:

Các luật A1, A2, A4 dễ dàng suy ra trực tiếp từ định nghĩa.

Chứng minh luật A3:

Cho quan hệ r trên tập thuộc tính U. $X\rightarrow\rightarrow Y$ và $Y\rightarrow\rightarrow Z$ thoả trên r nhưng $X\rightarrow\rightarrow Z-Y$ là không thoả. Tồn tại các bộ t_1, $t_2\in r$ sao cho với $t_1[X]=t_2[X]$ nhưng bộ t_3 với $t_3[X]=t_1[X]$, $t_3[Z-Y]= t_1[Z-Y]$ và $t_3[U-X-(Z-Y)]=t_2[U-X-(Z-Y)]$ lại không thuộc r.

Vì $X\rightarrow\rightarrow Y$ thoả trên r nên tồn tại bộ $t_4\in r$ mà:

$t_4[X]=t_1[X]$; $t_4[Y]= t_2[Y]$

$t_4[U-X-Y]= t_1[U-X-Y]$

Ta thấy rằng từ $t_4[Y]= t_2[Y]$ và vì $Y\rightarrow\rightarrow Z$ suy ra tồn tại $t_5\in r$ mà:

$T_5[Y]= t_2[Y]$

$T_5[Z]= t_4[Z]$

$T_5[U-Y-Z]= t_2[U-Y-Z]$

Khi đó $t_5[X]=t_1[X]$. Vì $t_5[Z]= t_4[Z]$ và $t_4[X]= t_1[X]$

nên $t_5[Z\cap X]= t_4[Z\cap X]=t_1[Z\cap X]$.

Vì $t_5[U-Y-Z]= t_4[U-Y-Z]$ nên $t_5[X-Z]= t_2[X-Z]$ và $t_2[X]= t_1[X]$.

Từ đó suy ra: $t_5[X-Z]= t_1[X-Z]$

Do đó: từ $t_5[X\cap Z]= t_1[X\cap Z]$ và $t_5[X-Z]= t_1[X-Z]$ và $t_1[X]= t_5[X]$

Như vậy $t_5[Z-Y]= t_1[Z-Y]$ (vì $t_5[Z-Y]= t_4[Z-Y]$).

và $t_4[Z-Y]= t_1[Z-Y]$ (vì $t_4[U-X-Y]= t_1[U-X-Y]$).

Cuối cùng cần khẳng định $t_5[W]= t_1[V]$ với $V=U-X-(Z-Y)$.

Suy ra $t_2[V-Z]= t_5[V-Z]$ và nhờ phép biến đổi tập hợp có được:

$$V\cap Z=(Y\cap Z)-X.$$

Mặt khác: $t_5[Z]= t_4[Z]$ và $t_4[Y]= t_2[Y]$ do đó $t_5[W\cap Z]= t_2[W\cap Z]$.

Nên: $t_5[V]= t_2[V]$.

Theo định nghĩa của bộ t_3 thì $t_3=t_5$

Mà $t_5\in r$ nên $t_3\in r$. Trái với giả thiết.

Vì vậy $X\rightarrow\rightarrow Z-Y$.

Chứng minh luật A5:

Giả sử rằng trong quan hệ r có $X\rightarrow\rightarrow Y$ và $W\rightarrow Z$, $Z\subseteq Y$, $W\cap Y=\varnothing$ mà $X\rightarrow Z$ là không thoả trên r thì tồn tại hai bộ $t_1,t_2\in r$ sao cho $t_1[X]=t_2[X]$ nhưng $t_1[Z]$

$\neq t_2[Z]$. Vì $X \rightarrow\rightarrow Y$, với $t_1[X]=t_2[X]$ nên $\exists t_3 \in r$ sao cho $t_3[X]= t_1[X]=t_2[X]$, $t_3[Y]=t_2[Y]$ và $t_3[U-X-Y]=t_1[U-X-Y]$.

Vì $W \cap Y=\emptyset$ nên $t_3[W]=t_1[W]$. Từ $Z \subseteq Y$ nên $t_3[Z]=t_2[Z]$.

Do $t_1[Z] \neq t_2[Z]$ nên $t_3[Z] \neq t_1[Z]$. Trái với giả thiết $W \rightarrow Z$.

Vậy $X \rightarrow Z$ là đúng.

9.1.3. Các luật suy dẫn và bổ sung cho phụ thuộc đa trị

1. (luật hợp cho phụ thuộc đa trị):

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

2. (luật tựa bắc cầu cho phụ thuộc đa trị)

Nếu $X \rightarrow\rightarrow Y$ và $WX \rightarrow\rightarrow Z$ thì $WX \rightarrow\rightarrow Z-WY$.

3. (luật tựa bắc cầu tổng hợp)

Nếu $X \rightarrow\rightarrow Y$ và $XY \rightarrow\rightarrow Z$ thì $X \rightarrow\rightarrow Z-Y$.

4. (luật tách cho phụ thuộc đa trị)

Nếu $X \rightarrow\rightarrow Y$ và $X \rightarrow\rightarrow Z$ thì $X \rightarrow\rightarrow Y \cap Z$, $X \rightarrow\rightarrow Y-Z$, $X \rightarrow\rightarrow Z-Y$.

Chú ý:

- Đối với luật tách của phụ thuộc hàm thì $X \rightarrow Y$ suy ra $X \rightarrow A$, $\forall A \in Y$.

- Đối với luật tách của phụ thuộc đa trị thì $X \rightarrow\rightarrow A$ được suy ra từ $X \rightarrow\rightarrow Y$ với $A \in Y$ chỉ có thể đúng nếu Z mà $X \rightarrow Z$ và $Z \cap Y=A$ hoặc $Y-Z=A$.

9.1.4. Một số tính chất

i. Nếu $X \rightarrow Y$ thoả trên quan hệ r thì $X \rightarrow\rightarrow Y$ cũng thoả trên r.

ii. Cho $X \rightarrow\rightarrow Y$ là phụ thuộc đa trị dị thường trên R và K là khoá bất kỳ trong R, thì $X \rightarrow Y$ thoả trên R hoặc $K \cap Y \neq \emptyset$.

iii. Cho $X \rightarrow\rightarrow Y \backslash Z$ là phụ thuộc đa trị dị thường trên lược đồ quan hệ R. Cho K là khoá của R khi đó $Y \backslash K \neq \emptyset$.

Nếu $Y_1=Y \backslash K$ and $Y_2=Y \cap K$ thì phụ thuộc hàm $XY_2 \rightarrow Y_1$ thoả trên R.

iiii. Nếu $X \rightarrow\rightarrow Y \backslash Z$ là phụ thuộc đa trị trên lược đồ quan hệ R và K là khoá của R khi đó $Y \backslash K \neq \emptyset$.

Nếu $Y_1=Y \backslash K$ and $Y_2=Y \cap K$ thì phụ thuộc hàm $XY_2 \rightarrow Y_1$ thoả trên R nhưng phụ thuộc hàm $XY_2 \rightarrow Z$ không thoả trên R.

Lưu ý

Một FD X→Y là một MVD X→→Y, nhưng ngược lại thì chưa hẳn.

Bài tập

Hãy chứng minh các tính chất trên (có thể tham khảo trong [8], [2]).

Bổ đề 9.2.

Một phụ thuộc đa trị $X \rightarrow\rightarrow Y$ đúng trên R nếu và chỉ nếu $X \rightarrow\rightarrow Y \backslash X$ đúng trên R.

Định lý 9.2. Cho r là quan hệ của lược đồ $R(U), X, Y \subseteq R(U)$ và $Z=U-XY$.

Quan hệ r thoả phụ thuộc đa trị $X \rightarrow\rightarrow Y$ khi và chỉ khi r được tách không mất mát thông tin thành hai quan hệ trên các lược đồ tương ứng $R_1=R[XY]$ và $R_2=R[XZ]$.

Chứng minh:

Cho $X \rightarrow\rightarrow Y$ là phụ thuộc đa trị thoả trên r. Đặt $r_1=\prod_{R1}(r)$ và $r_2=\prod_{R2}(r)$.

Gọi t là một bộ thuộc $r_1 * r_2$. theo định nghĩa phép kết nối, tồn tại $t_1 \in r_1$ và $t_2 \in r_2$ sao cho $t[X]=t_1[X]=t_2[X]$, $t[Y]=t_1[Y]$, $t[Z]=t_2[Z]$.

Vì r_1 và r_2 là hình chiếu của r trên lược đồ tương ứng, do đó cần thiết phải tồn tại t_3 và t_4 sao cho $t_1[XY]=t_3[XY]$, $t_2[XZ]=t_4[XZ]$. Từ phụ thuộc đa trị $X \rightarrow\rightarrow Y$ suy ra rằng t phải thuộc r và r phải chứa bộ t' với $t'[X]=t_3[X]$, $t'[Y]=t_3[Y]$, $t'[Z]=t_4[Z]$.

Ngược lại, nếu r là tách không mất mát thông tin thành r_1 và r_2 trên $R_1[XY]$ và $R_2[XY]$ cần chỉ ra rằng $X \rightarrow\rightarrow Y$ thoả trên r.

Giả sử tồn tại $t_1, t_2 \in r$ sao cho $t_1[X]=t_2[X]$, $r_1=\prod_{R1}(r)$ và $r_2=\prod_{R2}(r)$. Quan hệ $r_1$chứa bộ $t_3=t_1[XY]$, quan hệ r_2 chứa bộ $t_4=t_2[XZ]$ vì $r=r_1 * r_2$ nên r phải chứa bộ t sao cho $t[XY]=t_1[XY]$ và $t[XZ]=t_2[XZ]$. Do vậy bộ t chính là kết quả kết nối giữa t_3 và t_4. Vậy có bộ ba t, $t_3, t_4 \in r$ thoả điều kiện của MVD. Do đó $X \rightarrow\rightarrow Y$.

9.2. Bao đóng của phụ thuộc hàm và phụ thuộc đa trị.

Cho một tập các phụ thuộc hàm và phụ thuộc đa trị D. Cần tìm tập D^+ của tất cả các phụ thuộc hàm và phụ thuộc đa trị được suy dẫn logic từ D.Có thể tính D^+ xuất phát từ D nhờ hệ tiên đề Armstrong cho đến khi không còn dẫn xuất thêm một phụ thuộc nào nữa. Thông thường người ta cần biết liệu một phụ thuộc hàm $X \rightarrow Y$ hoặc một phụ thuộc đa trị $X \rightarrow\rightarrow Y$ có suy ra được từ D hay không.

Để kiểm tra một phụ thuộc đa trị $X\rightarrow\rightarrow Y\in D^+$ hay không chỉ cần xác định cơ sở phụ thuộc của X và xem liệu Y-X có phải là hợp của một số tập trên đó hay không. Để tính cơ sở phụ thuộc của X đối với F chỉ cần tính cơ sở phụ thuộc đối với các tập của các phụ thuộc đa trị M là đủ.

9.2.1. Khái niệm cơ sở phụ thuộc

Gọi X là tập con của U, F là tập các phụ thuộc hàm và M là tập các phụ thuộc đa trị. Có rất nhiều tập $Y\subseteq U$ sao cho $(X\rightarrow\rightarrow Y)\in D^+=(F\cup M)^+$. Khi đó kí hiệu $X\rightarrow\rightarrow Y_1|Y_2...|Y_n$ và giả thiết rằng Y_i, i=1..n.

Gọi M(X) là họ tất cả những tập Y mà XY thì tồn tại một họ con duy nhất có các tính chất sau:

- Tất cả các tập trong họ con này là không rỗng.
- Mỗi cặp tập trong họ con là phân biệt.
- Mỗi tập thuộc M(X) là hợp của một số tập từ họ con.

Họ con này bao gồm tất cả các tập không rỗng, tối thiểu trong M(X), được gọi là cơ sở phụ thuộc của X.

9.2.2. Tính toán cơ sở phụ thuộc

Để tính cơ sở phụ thuộc của X đối với D chỉ cần tính cơ sở phụ thuộc đối với các tập các phụ thuộc đa trị M là đủ. Khi đó M phải bao gồm:

- Tất cả các phụ thuộc đa trị thuộc D.
- Với mỗi phụ thuộc hàm$(X\rightarrow Y)\in D$ thì thay bằng tập các phụ thuộc đa trị $X\rightarrow\rightarrow A_1,..,X\rightarrow\rightarrow A_n$, trong đó $Y=A_1A_2...A_n$, tức là $A_i\in Y$.

Bằng cách loại bỏ tất cả các phụ thuộc hàm tầm thường từ cơ sở phụ thuộc sẽ tương ứng với cơ sở của tập M, có thể chỉ ra rằng nếu X không chứa A thì $X\rightarrow A$ khi và chỉ khi:

i. A là một tập độc lập với cơ sở phụ thuộc của X đối với tập M.

ii. Tồn tại một tập thuộc tính Y không chứa A sao cho $(Y\rightarrow Z)\in D$ và $A\in Z$.

Thuật toán 9.1. (tính cơ sở phụ thuộc)

Vào: Tập các phụ thuộc đa trị M trên tập thuộc tính U và $X\in U$.

Ra: Cơ sở phụ thuộc của X đối với M.

Phương pháp:

+ Gọi T là tập hợp của các tập $Z \subseteq U$ sao cho với $(W \rightarrow Y) \in M$ có $W \subseteq X$ và $Z=Y-X$ hoặc $Z=U-X-Y$.

T sẽ thiết lập cho đến khi là một tập của các tập rời nhau, khi đó tìm một cặp $Z_1, Z_2 \in T$ nếu là không rời nhau thì thay chúng bằng Z_1-Z_2, Z_2-Z_1, $Z_1 \cap Z_2$.

Gọi S là tập đích.

+ Tiếp tục cho đến khi không thay đổi được nữa, tìm một phụ thuộc đa trị $(V \rightarrow\rightarrow W) \in M$ và một tập $Y \subseteq S$ sao cho $Y \cap W \neq V$. Thay Y bằng $Y \cap W$ và Y-W trong F. Tập cuối cùng của F sẽ là cơ sở phụ thuộc của X.

9.3. Kết nối không mất thông tin

Thuật toán 9.2. (Kiểm tra phép kết nối có mất mát thông tin hay không)

Vào: Lược đồ quan hệ R, tập phụ thuộc hàm F và tập các phụ thuộc đa trị D.

Ra: Trả lời kết nối là mất mát thông tin hay không.

Phương pháp:

a. Lập ma trận k hàng(k lược đồ quan hệ), n cột (n thuộc tính $A_1,..,A_n$). Phần tử (i,j) của ma trận được xác định: $(i,j)=a_j$ nếu $A_j \in R_j$

$(i,j)=b_{ij}$ nếu $A_j \notin R_i$, trong đó a_i, $b_{ij} \in Dom(A_j)$.

b. Biến đổi bảng:

+ Nếu là phụ thuộc hàm thì:

- Lần lượt xét các phụ thuộc hàm $X \rightarrow Y \in F$.

- Với mỗi phụ thuộc hàm $X \rightarrow Y$, nếu phát hiện có các hàng mà giá trị của các hàng đó giống nhau trên thuộc tính X và khác nhau trên thuộc tính Y và khác nhau trên thuộc tính Y thì làm cho các hàng đó cũng giống nhau trên Y bằng cách:(không mất tổng quát giả sử có hai hàng t_1 và t_2 mà $t_1[X]=t_2[X]$)

. Nếu $t_1[Y]=a_j$ và $t_2[Y]=b_{ij}$ thì $t_2[Y]=a_j$.

. Nếu $t_1[Y]=b_{ij}$ và $t_2[Y]=b_{ij}$ thì $t_1[Y]=t_2[Y]$ hoặc ngược lại.

- Quá trình biến đổi được lặp đi lặp lại cho đến khi không thể biến đổi được nữa.(xét đến phụ thuộc hàm cuối cùng thuộc F).

+ Nếu có phụ thuộc đa trị $X \rightarrow\rightarrow Y$ và có hai hàng t_1, t_2 mà $t_1[X]=t_2[X]$ thì thêm một hàng U với $U[X]=t_1[X]=t_2[X]$, $U[Y]=t_1[Y]$ và $U[U-X-Y]=t_2[U-X-Y]$ nếu U chưa có trong T.

Tiếp tục cho đến khi không có một giá trị a hoặc b được thiết lập nữa. Nếu xuất hiện một dòng hoàn toàn là các giá trị a thì phép kết nối là không mất mát thông tin, ngược lại là mất mát thông tin.

Định lý 9.3.

Gọi R là một lược đồ quan hệ, $\rho=(R_1,R_2)$ là một phép tách của R,D là tập các phụ thuộc hàm và phụ thuộc đa trị trên tập thuộc tính của R,ρ là phép tách không mất mát thông tin khi và chỉ khi $R_1 \cap R_2 = R_1 - R_2$.

Chứng minh:

ρ là phép tách không mất mát thông tin khi và chỉ khi với một quan hệ r thoả D và hai bộ bất kỳ t_1,t_2, $\exists t_3 \in r: t_3[R_1]=t_1[R_1]$ và $t_3[R_2]=t_2[R_2]$.

Nhưng t_3 chỉ tồn tại khi và chỉ khi $t_1[R_1 \cap R_2]=t_2[R_1 \cap R_2]$.

Như vậy, r luôn luôn tồn tại trong R chính xác với điều kiện $R_1 \cap R_2 \rightarrow\rightarrow R_2 - R_1$ hoặc $R_1 \cap R_2 \rightarrow\rightarrow R_1 - R_2$.

9.4. Dạng chuẩn 4 (4NF)

Định nghĩa 9.4.

Gọi R là một lược đồ quan hệ và D là tập các phụ thuộc (FD và MVD) đúng trên R. Ta nói rằng R ở dạng chuẩn 4 (4NF) nếu với mỗi $X \rightarrow\rightarrow Y$, trong đó $Y \neq \emptyset$ hoặc $Y \not\subset X$ và XY không chứa tất cả các thuộc tính của R thì X chứa một khoá của R.

Ta có một số tính chất của 4NF như sau:

Mệnh đề 9.1.

i) Một lược đồ quan hệ ở dạng chuẩn 4 khi và chỉ khi những phụ thuộc đa trị cơ sở là phụ thuộc trong đó xác định một thuộc tính. Tức là nếu có một phụ thuộc đa trị thì nó có dạng $X \rightarrow\rightarrow A$.

ii) R(U) ở dạng chuẩn 4NF nếu mọi phụ thuộc đa trị $X \rightarrow\rightarrow Y \in D^+$ là phụ thuộc cơ sở thì $X^+=U$.

iii) Với mọi phụ thuộc đa trị thuộc tập bao đóng của D: $X \rightarrow\rightarrow Y \in D^+$, Y khác rỗng, Y không nằm trong X, $XY \neq U$ thì $X^+=U$.

iv) R ở 4NF thì R ở BCNF và như vậy là 3NF.

Thuật toán 9.2. (Tìm một phép tách của R thành $\rho=(R_1,R_2,..,R_k)$ sao cho ρ là không mất mát thông tin đối với D và mỗi R_i đều ở 4NF)

Vào: Phân tách ρ trên lược đồ quan hệ R ở dạng chuẩn 3, tập phụ thuộc hàm và phụ thuộc đa trị D.

Ra: Phân tách $(R_1,..,R_k)$ là không mất mát thông tin đối với D và R_i ở dạng chuẩn 4.

Phương pháp:

+ Nếu có một lược đồ quan hệ trong ρ chưa ở dạng chuẩn 4 đối với D được chiếm ở trên S thì cần tìm phụ thuộc đa trị $X \rightarrow\rightarrow S$, trang đó X không chứa khoá của S, $Y \neq \varnothing$ hoặc $Y \subseteq X$ với $XY \neq S$. Gĩa sử $(X \cap Y = \varnothing)$. áp dụng tiên đề 1(của phụ thuộc hàm), tiên đề 4(của phụ thuộc đa trị) và luật tách cho phụ thuộc đa trị cho $X \rightarrow\rightarrow Y$ suy ra $X \rightarrow\rightarrow Y-X$.

Sau đó thay thế S bằng $S_1=XY$, $S_2=S-Y$

Vì $S_1 \cap S_2 \rightarrow\rightarrow S_1-S_2$ nên kết nối của S_1 và S_2 là mất mát thông tin đối với D trên tập S.

+ Tiếp tục tính toán đối với các lược đồ còn lại.

Ví dụ: Xét lược đồ quan hệ R(THRSG) khóa SH.

D=(C→T, HR→C, HT→R, CS→G, HS→R, C→→HR).

Ta có: C→HR, C→T suy ra C→→SG.

HR→C, C→T suy ra HR→T(luật bắc cầu), HR→CT(luật hợp) HR→CT, HR→→SG(luật bù).

Với lược đồ biến đổi R,ρ=R=CTHRSG là phép tách ra một quan hệ. Rõ ràng R không phải ở 4NF.

Chọn C→→ HR là không thoả mãn điều kiện của 4NF vì không chứa khoá.

Tách R thành hai lược đồ R_1=CHR, R_2=CTSG.

Lược đồ R_1 có khoá HT. C→→HR không vi phạm đối với 4NF. Ngoài ra, không một phụ thuộc hàm hoặc phụ thuộc đa trị nào được chiếu trên R_1 vi phạm đối với 4NF, do vậy R_1 không tách được nữa.

Lược đồ R_2=CTSG có khoá là CS.

Chọn C→→T(từ phụ thuộc hàm C→T)

R_2 được tách thành R_{21}= CT, R_{22}=CGS, hai lược đồ này ở dạng 4NF.

Vậy R được tách thành ba lược đồ quan hệ ở 4NF là (R_1,R_{21},R_{22}) hay (CHR,CT,CSG).

9.5. Phụ thuộc kết nối và dạng chuẩn 5 (5NF)

Tính chất nối các quan hệ không tổn thất thông tin là một trong các tính chất tạo điều kiện thuận lợi cho việc thiết kế cơ sở dữ liệu. Tuy nhiên khi nghiên cứu tập các quan hệ, nếu chỉ dùng các công cụ phụ thuộc hàm, phụ thuộc đa trị chúng ta chưa thể xét hết các đặc thù cần thiết của các quan hệ. Để tiếp tục đi sâu nghiên cứu các lớp các quan hệ chúng ta sẽ trình bày thêm khái niệm phụ thuộc kết nối. Một vấn đề được đặt ra ở đây là kết nối không bị mất thông tin và không làm cồng kềnh quan hệ ban đầu cũng như làm sai lệch nội dung bài toán quản lý. Sau đây chúng ta xét một quan hệ r có ngữ nghĩa như sau: Ta có bảng theo dõi chủ xe, mác xe cùng màu xe, mỗi chủ xe có thể có nhiều cùng mác và các xe khác nhau.

Ví dụ:

Quan hệ chủ, xe cùng các màu và mácкhác nhau

CHU	MAU	MAC
Châu	Đen	Honđa
Châu	Đen	Toyota
Châu	Đỏ	Toyota
Lan	Đen	Toyota

Quan hệ trên đây là 4NF vì nó không chứa ràng buộc FD và MD ta có thể tách quan hệ trên thành 3 quan hệ sau: r1 là quan hệ CHU sử dụngvà MAU xe, r2 là quan hệ chủ sử dụng và MAC xe, r3 là quan hệ MAU và MAC xe.

r1		r2		r3	
TÊN	MAU	TÊN	MAC	MAU	MAC
Châu	Đen	Châu	Honđa	Đen	Honđa
Châu	Đỏ	Châu	Toyota	Đen	Toyota
Lan	Đen	Lan	Toyota	Đỏ	Toyota

Ta nhận thấy điều lý thú là nối hai bất kỳ trong ba quan hệ trên không cho ta quan hệ ban đầu. Như vậy phép tách đã làm "tổn thất" thông tin.

Để nghiên cứu và giải quyết những vấn đề của các lớp quan hệ tượng tự như trên, một số tác giả đã đưa ra các đóng góp với ý tưởng sau:

"Nếu chỉ dừng lại ở phụ thuộc đa trị, chúng ta chưa đủ công cụ để giải quyết được tất cả các trường hợp dư thừa và tách không tổn thất thông tin trong một lớp khá lớn các quan hệ."

Sau đây chúng ta sẽ xét khái niệm phụ thuộc kết nối.

Định nghĩa 9.5. Phụ thuộc kết nối (joint dependency)

Cho $R=\{A_1,A_2,A_3,.....A_n\}$ là một lược đồ quan hệ r là quan hệ trên R. $X_1,X_2,.....,X_m$ là các tập con của R. Ta nói rằng có phụ thuộc kết nối giữa các X_1,X_2 ,…,Xm và kí hiệu $*\{X_1,X_2,\ldots,X_m\}$ nếu r là nối của các chiếu của r lên các tập $X_1,X_2,\ldots,X_m$ tương ứng.

Hay ta có $r = r.X_1 \bowtie r.X_2 \bowtie ... \bowtie r.X+m$.

Định nghĩa 9.6. (dạng chuẩn 5)

Cho lược đồ quan hệ $R=\{ A_1, A_2,\ldots,A_n\}$. r là một quan hệ trên R, ta nói r ở dạng chuẩn 5, kí hiệu là 5NF khi và chỉ khi tất cả các phụ thuộc kết nối thực hiện bởi các khoá.

Nhận xét:

Trong các lớp quan hệ ở dạng chuẩn 5 (5NF) còn nhiều vấn đề về lý thuyết cũng như thực tiễn chúng ta cần phải thực sự quan tâm và đầu tư thời gian hơn nữa, mà giáo trình chưa thể đề cập hết.

9.6. Mối liên hệ giữa các dạng chuẩn

Dựa vào định nghĩa của dạng chuẩn 4NF và phụ thuộc 5NF ta có các định lý sau.

Định lý 9.4.

Cho lược đồ quan hệ R(U) nếu thuộc dạng chuẩn 4NF thì thuộc dạng chuẩn BCNF. Nếu R(U) thuộc 5NF thì thuộc 4NF.

Cùng với mối liên hệ của các dạng chuẩn 1NF, 2NF, 3NF và BCNF ta có hình ảnh diễn tả mối liên hệ giữa các dạng chuẩn như sau: Tập các lược đồ quan hệ ở dạng chuẩn $5NF \subseteq 4NF \subseteq BCNF \subseteq 3NF \subseteq 2NF \subseteq 1NF$.

Câu hỏi và bài tập chương 9

1. Cho lược đồ quan hệ R bao gồm các thuộc tính sau:

C#, I, D, B, K, L, M, G và tập phụ thuộc hàm.

$C\# \rightarrow IDBKF, D \rightarrow B, K \rightarrow R$

Tìm dạng chuẩn cho R.

2. Cho lược đồ quan hệ với các thuộc tính A, B, C, D, D, F và tập phụ thuộc hàm :

$\{ AB \rightarrow C, C \rightarrow B, ABD \rightarrow E, F \rightarrow A\}$

Tìm dạng chuẩn BCNF cho R

3. Kiểm tra tính không mất mát thông tin của phép tách R thành :

$p (R) = (BC, AC, ABDE, ABDF)$

với tập phụ thuộc hàm như giả thiết ở bài 2.

4. Cho lược đồ quan hệ R trên tập thuộc tính $U = \{ABCDE\}$

Chứng minh rằng nếu quan hệ r xác định trên U thỏa các phụ thuộc đa trị.

$F = \{A \rightarrow\rightarrow BC, DE \rightarrow\rightarrow c\}$

thì R cũng thỏa $AD \rightarrow\rightarrow BE$

5. Chứng minh tính đúng của các qui tắc sau đây:

a. $X \rightarrow\rightarrow X$ cho mọi X

b. Nếu $X \rightarrow\rightarrow Y$ thì $XZ \rightarrow\rightarrow Y$ với $Z \subseteq U$

6. Cho $X, Y, Z, W \subseteq U$

Chứng minh rằng nếu quan hệ r thỏa trên R (U) thỏa các phụ thuộc đa trị

$X \rightarrow\rightarrow Y, Z \subseteq W$ thì r cũng thỏa $XW \rightarrow\rightarrow YZ$

7. Cho lược đồ R (ABCDEI) và

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

Tìm dạng chuẩn 4 qua phép tách.

8. Cho lược đồ CSDL R(BOISQD) và tập phụ thuộc đa trị và phụ thuộc hàm : $F = \{ S \rightarrow\rightarrow D, I \rightarrow B, IS \rightarrow Q, B \rightarrow Q\}$. Tìm dạng chuẩn 4 qua phép tách.

9. Kiểm tra tính không mất mát thông tin của phép tách kết nối R thành:

$\rho = (R_1, R_2, R_3, R_4, R_5)$

Với : $R_1 = AC, R_2 = CD, R_3 = BE, R_4 = BC, R_5 = AE$

và : R = ABCE và tập phụ thuộc hàm.

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

10. Cho lược đồ R = (BOISQD) và

$$F = \{S \rightarrow D, I \rightarrow B, IS \rightarrow Q, B \rightarrow O\}$$

- Chứng tỏ rằng phép tách :

R = (SD, IB, ISQ, BO)

là tách không mất mát thông tin.

- Chứng tỏ rằng kết quả các lược đồ con của phép tách trên là ở dạng 3 NF?

11. Tìm phủ không dư thừa của tập phụ thuộc hàm

$G = \{ A \rightarrow C, AB \rightarrow C, C \rightarrow DI, CD \rightarrow I, EC \rightarrow AB, EI \rightarrow C\}$

12. Cho lược đồ R (ABCDEI) và tập phụ thuộc dữ liệu

$\{ a \rightarrow\rightarrow BCD, C \rightarrow D, B \rightarrow\rightarrow AC\}$

Tìm dạng chuẩn 4 của lược đồ.

13. Chứng minh rằng nếu quan hệ r (R) thỏa các phụ thuộc đa trị

$X \rightarrow\rightarrow Y_1, X \rightarrow\rightarrow Y_2, \ldots X \rightarrow\rightarrow Y_K$ trong đó

$R = \{XY_1 \ldots Y_k\}$ thì r có thể tách thành các lược đồ $XY_1, XY_2, \ldots, XY_k$ không mất mát thông tin.

Tài liệu tham khảo

Tiếng Việt

[1]. *Bài giảng môn Cơ sở dữ liệu* – Trường Cao đẳng Công nghiệp 4, Thành phố Hồ Chí Minh – 2005.

[2]. Hồ Thuần, Hồ Cẩm Hà – *Các hệ cơ sở dữ liệu – lý thuyết và thực hành*, NXB Giáo dục, 2004.

[3]. Nguyễn An Tế, Đồng Thị Bích Thủy – *Nhập môn cơ sở dữ liệu* – NXB Giáo dục – 1996.

[4]. Nguyễn Xuân Huy, Lê Hoài Bắc – *Bài tập cơ sở dữ liệu* – NXB Thống kê, 2006.

[5]. Phạm Gia Tiến, Phạm Thế Phi, *Giáo trình Hệ Quản Trị cơ sở dữ liệu*, ĐH Cần Thơ, 2005.

[6]. Phạm Thế Quế, *Giáo trình Cơ sở dữ liệu*, Học viện Công nghệ Bưu chính Viễn thông, Hà Nội, 2006.

[7]. Trần Nguyên Phong, *Giáo trình SQL*, Đại học Khoa học Huế, 2004.

[8]. Ullman. J.D. "*Nguyên lý các hệ cơ sở dữ liệu và tri thức*", NXB KHKT, 1997.

[9]. Nguyễn Xuân Huy, *Các phụ thuộc logic trong cơ sở dữ liệu*, Viện Khoa học và Công nghệ Việt Nam, Hà Nội, 2009.

Tiếng Anh

[9]. C. J. Date, Hugh Darwen, *A Guide to the SQL Standard*, Addison-Wesley Publishing, 1992.

[10]. Diana Lorentz, *SQL Reference*, Oracle Corporation, 2001.

[11]. James R, Groff, Paul N.Weinberg, *SQL: The Complete Reference*, McGraw-Hill/Osborne, 2002.

[12]. King J.I. "*QUIST: A system for semantic query optimization in relation al databases*", Pro 7th Int Confr. On Very large Databases, Cannes, France, IEEE, 1981.

[13]. Marcilina S. Garcia, Jamie Reding, Edward Whalen, Steve Adrien DeLuca, *SQL Server 2000 Administrator's Companion*, Microsoft Press, 2000.

[14]. Smith J.M, Chang P.Y. "*Optimizing the performance of a relational algebra database interface*", Comm ACM, vol 18, 1975.

[15]. Wong E, Youssefi K, "*Decomposition – A strategy for query processing*", ACM IODS, vol 1, 1976.

[16]. Ramez Elmasri, Shamkant B. Navathe, Fundamentals of database systems, Fourth edition, Pearson Education, Inc. 2004.