LỜI NÓI ĐẦU

Ngày nay, chúng ta đang sống và làm việc trong thời đại phát triển của công nghệ thông tin. Nhu cầu sử dụng thông tin ngày càng được mọi người quan tâm hơn. Thông tin truy xuất không những phải chính xác mà yêu cầu cần phải tránh dư thừa dữ liệu và những dị thường trong cập nhật dữ liệu. Liên quan đến vấn đề này, lý thuyết cơ sở dữ liệu đóng góp một vai trò hết sức quan trọng.

Nhập môn cơ sở dữ liệu là một trong những giáo trình thiết yếu đối với sinh viên chuyên ngành Công nghệ thông tin. Thông qua giáo trình này, chúng tôi muốn trình bày một số kiến thức nền tảng mang tính chất giới thiệu lý thuyết về cơ sở dữ liệu. Nội dung của giáo trình được chia làm 6 chương. Phần bài tập cũng được đính kèm trong giáo trình này nhằm tạo điều kiện cho việc nâng cao hiểu biết của sinh viên trong từng chủ đề. Ngoài những bài tập ở mức trung bình nhằm giúp sinh viên có thể tự kiểm tra kiến thức, trong giáo trình còn nêu một số bài tập khác nhằm tạo điều kiện cho sinh viên bước đầu làm quen với các nghiên cứu chuyên sâu trong lĩnh vực này.

Trong quá trình biên soạn giáo trình này, tác giả đã nhận được nhiều ý kiến đóng góp quý báu về nội dung chuyên môn của PGS. TS. Lê Mạnh Thạnh, TS. Nguyễn Mậu Hân, TS. Trương Công Tuấn cùng nhiều đồng nghiệp khác trong khoa Công nghệ thông tin - trường Đại học Khoa học Huế. Tác giả xin chân thành cảm ơn sự giúp đỡ quý giá đó.

Mặc dù đã có nhiều cố gắng, nhưng giáo trình này không thể tránh khỏi những khuyết điểm. Tác giả mong muốn nhận các ý kiến đóng góp để hy vọng chất lượng giáo trình sẽ được tốt hơn trong các lần tái bản sau.

Huế, ngày 15 tháng 12 năm 2009

Hoàng Quang

Chương 1. KHÁI QUÁT VỀ CƠ SỞ DỮ LIỆU

1.1. Các khái niệm cơ bản

Cơ sở dữ liệu (database)

Một cơ sở dữ liệu (CSDL) là một tập hợp các dữ liệu có liên quan với nhau được lưu trữ trong máy tính theo một quy định nhất định nhằm phục vụ cho một mục đích quản lý nào đó.

Ví dụ: CSDL phục vụ cho việc quản lý các chuyến bay của một hãng hàng không cung cấp các thông tin về số hiệu chuyến bay, nơi xuất phát, nơi đến, số chỗ ngồi, ngày bay của tất cả các chuyến bay trong năm; CSDL phục vụ cho công tác quản lý đào tạo trong trường chứa các dữ liệu phản ánh thông tin về học sinh, giáo viên, môn học, phòng học,...

Hệ quản trị cơ sở dữ liệu (*Database Management System)*
Hệ quản trị CSDL (HQTCSDL) là phần mềm cho phép người dùng giao tiếp với cơ sở dữ liệu, và thông qua đó cung cấp một môi trường thuận lợi và hiệu quả để tìm kiếm và lưu trữ thông tin của cơ sở dữ liệu.
Ví dụ: Hệ quản trị cơ sở dữ liệu FOXPRO, hệ quản trị cơ sở dữ liệu SQL SERVER,...

Việc phân biệt một HQTCSDL với các phần mềm khác được dựa trên hai đặc trưng cơ bản sau:

- *Quản lý dữ liệu bền*. Đây là khả năng lưu trữ các cơ sở dữ liệu phục vụ cho mục tiêu khai thác lâu dài.

- *Truy cập các khối lượng lớn dữ liệu một cách hiệu quả*. HQTCSDL cung cấp các thao tác cũng như phương pháp tổ chức dữ liệu nhằm giúp cho người sử dụng truy cập vào các dữ liệu trong CSDL một cách nhanh chóng và thuận lợi.

Ngoài ra, một hệ quản trị cơ sở dữ liệu còn có các khả năng sau:

- ❖ Quản lý cơ sở dữ liệu lâu dài (dữ liệu không bị mất khi kết thúc).
- ❖ Quản lý một số lượng lớn dữ liệu.
- ❖ Cho phép truy cập vào mỗi khối lượng dữ liệu lớn với yêu cầu xử lý nhanh.
- ❖ Cung cấp ít nhất một mô hình dữ liệu, qua đó người dùng có thể quan sát được dữ liệu.
- ❖ Cung cấp một ngôn ngữ bậc cao, qua đó người dùng có thể định nghĩa dữ liệu và xử lý dữ liệu.

Hệ cơ sở dữ liệu (*Database System)*

Hệ cơ sở dữ liệu là một hệ thống phần mềm nhằm quản lý cơ sở dữ liệu của một hệ thống thông tin cụ thể nào đó.

Như vậy các thành phần bên trong một hệ cơ sở dữ liệu gồm có: chương trình, cơ sở dữ liệu, hệ quản trị cơ sở dữ liệu, người sử dụng.

1.2. Các mức trừu tượng hóa mô hình cơ sở dữ liệu

Mô hình dữ liệu là sự hình thức hóa toán học của dữ liệu bao gồm hai phần:

- Ký hiệu mô tả dữ liệu
- Tập hợp các phép toán

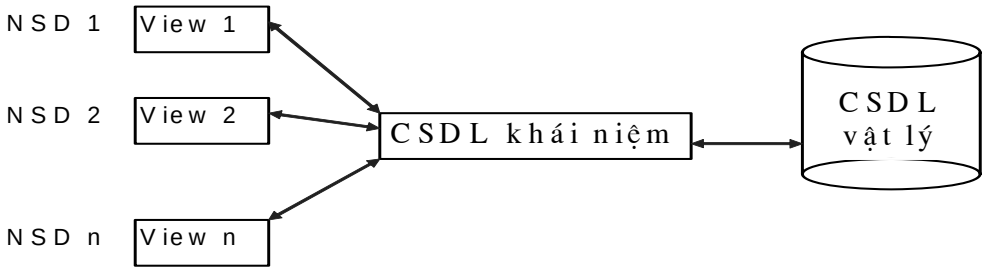
Mỗi mô hình CSDL có ba mức trừu tượng:

- Mức vật lý: Mức thấp nhất của sự trừu tượng mô tả dữ liệu được lưu trữ một cách thực sự như thế nào. Tại mức này CSDL được xem là một bộ các tập tin, các tập tin chỉ mục hoặc cấu trúc lưu trữ khác gọi chung là CSDL vật lý. CSDL vật lý tồn tại bền trong các thiết bị lưu trữ phụ và nhiều CSDL vật lý có thể được quản lý bởi một HQTCSDL.

- Mức khái niệm: Mức cao hơn tiếp theo của sự trừu tượng mô tả những dữ liệu nào được lưu trữ trong CSDL và các mối quan hệ nào tồn tại giữa các dữ liệu này. CSDL mức khái niệm là sự trừu tượng hóa thế giới thực khi nó gắn liền với người sử dụng CSDL. Một HQTCSDL cung cấp một ngôn ngữ định nghĩa mức khái niệm (*DDL-Data Definition Language*) thường gọi là mô hình cơ sở dữ liệu (database model).

- Mức khung nhìn (View): Mức cao nhất của sự trừu tượng mô tả chỉ một phần của toàn bộ CSDL. Mặc dầu sử dụng cấu trúc đơn giản hơn mức logic, một số phức tạp vẫn còn tồn tại do kích thước lớn của CSDL.Hệ thống có thể cung cấp nhiều khung nhìn đối với cùng một CSDL.

Ta có thể hình dung các mức trừu tượng của CSDL như trong hình vẽ sau đây:



Ví dụ. Xét một mảng dữ liệu hai chiều n x m phần tử .

- Tại mức quan niệm ta có thể hình dung mảng này như là một bảng có n dòng và m cột. Mảng này được được mô tả trong một vài ngôn ngữ lập trình C như sau:

int A[n][m] ;

Ở mức này ta không biết được dữ liệu được lưu trữ như thế nào trong thiết bị nhớ mà chỉ có thể biết rằng phần tử ở dòng i cột j là A[i][j].

- Mức vật lý của mảng chính là mảng được lưu trữ trong thiết bị nhớ.

- Mức khung nhìn: Từ mức trừu tượng ở trên ta có thể mô tả các khung nhìn của mảng. Chẳng hạn, cho biết tổng tất cả các giá trị trên dòng i.

$$f(i) = \sum_{j=1}^{m} A[i][j]$$

1.3. Lược đồ và thể hiện

Để xây dựng các CSDL chúng ta phải hoạch định cho các dữ liệu trong CSDL một qui cách lưu trữ. Qui cách này này được hình thức hoá như là khung cho mọi dữ liệu trong CSDL, gọi là *lược đồ CSDL*.

Ví dụ: Để tổ chức CSDL về các nhân viên trong một cơ quan chúng ta tạo trước lược đồ NHANVIEN(HOTEN,NAMSINH ,DIACHI, SODT) để lưu trữ các dữ liệu bao gồm: họ tên, năm sinh, địa chỉ, số điện thoại của các nhân viên.

Nội dung hiện thời của các dữ liệu trên lược đồ gọi là một thể hiện của lược đồ. Chẳng hạn, với lược đồ trên thì bảng sau đây là một thể hiện:

Nguyễn Van Anh	1957	Phú vang	3837756
Hoàng Thị Lan	1958	14. Lê Lợi, Huế	3825424
Huỳnh Dung	1958	315 Chi Lăng, Huế	3545245
Trần Quang	1956	12 Đống Đa, Huế	3822789
Phan Thanh Bình	1960	32 Chi Lăng, Huế	3345053

Như vậy một lược đồ có thể có nhiều thể hiện khác nhau.

1.4. Các ngôn ngữ CSDL

Trong các ngôn ngữ chương trình thì các chỉ thị mô tả và các chỉ thị thực hiện là hai bộ phận của một ngôn ngữ. Trong các HQTCSDL hai chức năng mô tả và tính toán nói chung được chia thành hai ngôn ngữ khác nhau.

Các ngôn ngữ định nghĩa dữ liệu (DDL)

Ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language) không phải là ngôn ngữ thủ tục mà chỉ là mô tả các loại đối tượng và quan hệ giữa các đối tượng.

Chẳng hạn, các chuyến bay của một hãng hàng không có thể được lưu trữ trong một CSDL được định nghĩa như sau:

```
CREATE  TABLE  CHUYEN-BAY
(SO:INT, NG: CHAR(6),CH:INT , TU : CHAR (3),DEN : CHAR (3) );
```

trong đó SO là số hiệu chuyến bay, được mô tả là một số nguyên; NG là ngày tháng thực hiện chuyến bay, được mô tả là một chuỗi 6 ký tự; CH là số chỗ ngồi chưa được đặt trên chuyến bay, được mô tả là một số nguyên; TU, DEN là ký hiệu nơi xuất phát và nơi đến, mỗi địa điểm được viết tắt bởi 3 ký tự.

Các ngôn ngữ xử lý dữ liệu (DML)

Các hệ QTCSDL đòi hỏi phải có ngôn ngữ để xử lý các phép toán trên CSDL gọi là DML (Data Manipulation Language) hoặc ngôn ngữ truy vấn (Query Language).

Chẳng hạn, xét các yêu cầu xử lý dưới đây trên ngôn ngữ SQL đối với cơ sở dữ liệu CHUYEN-BAY.

- Giảm 4 chỗ ngồi trên chuyến bay 123 ngày 31 tháng 8 có thể viết:

```
UPDATE   CHUYEN-BAY
SET  CH = CH - 4
WHERE  SO = 123 AND  NG = ‘ AUG 31 ‘ ;
```

- Thêm vào chuyến bay 147 có 72 chỗ ngồi từ Hà nội (HAN) đến Huế (HUI).

```
INSERT INTO  CHUYEN-BAY
VALUES (147, ‘AUG 21‘ , 72 , ‘HAN‘ , ‘HUI‘ );
```

- Tìm lại từ CSDL số chỗ ngồi còn trống trên chuyến bay 148 ngày 24 tháng 7.

```
SELECT  CH
FROM  CHUYEN-BAY
WHERE SO= 148 AND  NG = ‘JULY 24‘;
```

- Tìm lại tất cả các chuyến bay từ Hà nội đến Huế ngày 20 tháng 8.

```
SELECT  SO
FROM CHUYEN-BAY
WHERE  NG = ‘AUG 20‘
TU = ‘HAN‘ AND  DEN  = ‘HUI‘;
```

Các ngôn ngữ chủ (Host Language)

Ngoài những phép xử lý thông thường sẵn có trong các HQTCSDL các chương trình ứng dụng cần phải có thêm một số công việc phức tạp hơn.

Chẳng hạn, một chương trình được sử dụng bằng một hãng hàng không để đăng ký chỗ không chỉ cần truy tìm số chỗ trống từ CSDL mà còn phải làm các công việc phức tạp hơn như in vé, làm giấy hẹn, đối thoại với người sử dụng,... Như vậy, các chương trình để xử lý dữ liệu cần được viết chung trong một ngôn ngữ chủ - đó là ngôn ngữ thuận tiện cho việc lập trình, chẳng hạn C, C++. Ngôn ngữ chủ được sử dụng cho các quyết định, hiển thị câu hỏi, đọc câu trả lời,...

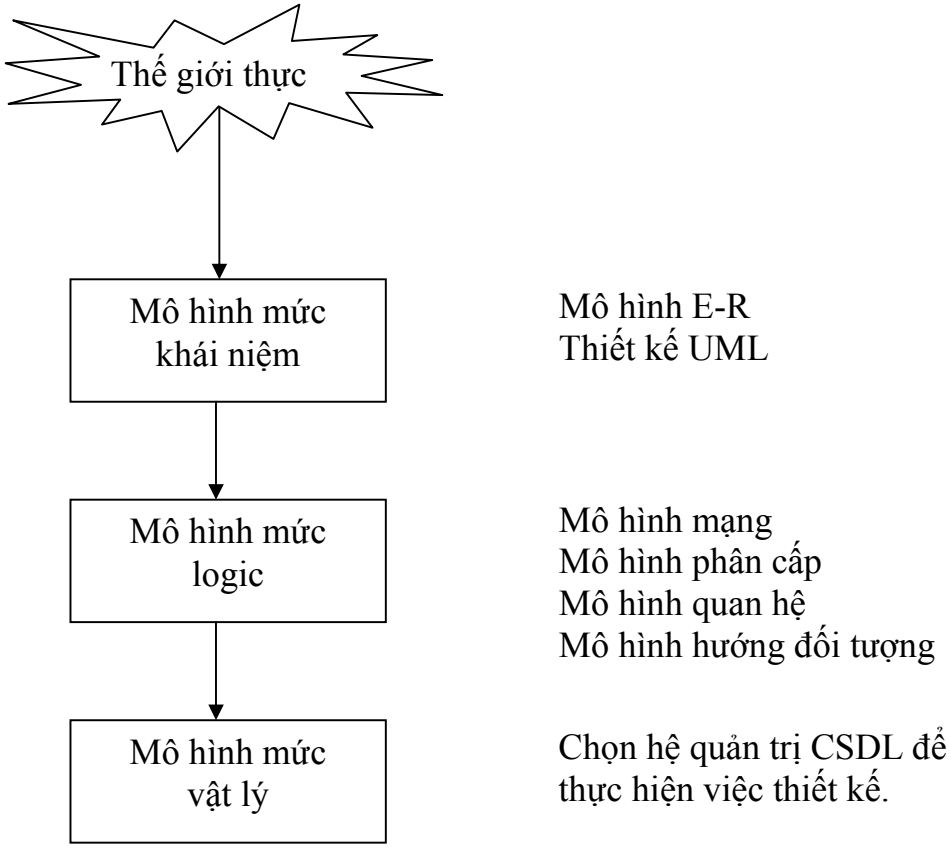
1.5. Các mô hình cơ sở dữ liệu

Mỗi hệ quản trị cơ sở dữ liệu cung cấp một mô hình cho cơ sở dữ liệu và thông qua mô hình đó người dùng có thể thấy được bản chất của dữ liệu và mối quan hệ giữa các dữ liệu. Có các loại mô hình cơ sở dữ liệu như sau:

- Mô hình E-R (Entity – Relationship)
- Mô hình mạng
- Mô hình phân cấp
- Mô hình quan hệ
- Mô hình hướng đối tượng

1.6. Quy trình thiết kế cơ sở dữ liệu

Quá trình thiết kế một cơ sở dữ liệu cho một hệ cơ sở dữ liệu có thể được thực hiện bởi sơ đồ sau:



Chương 2. MÔ HÌNH THỰC THỂ-MỐI QUAN HỆ

2.1. Giới thiệu

Mô hình E-R được đề xuất bởi P. Chen (1976). Đây là một mô hình khái niệm dựa vào việc nhận thức thế giới thực thông qua tập các đối tượng được gọi là các thực thể và các mối quan hệ giữa các đối tượng này.

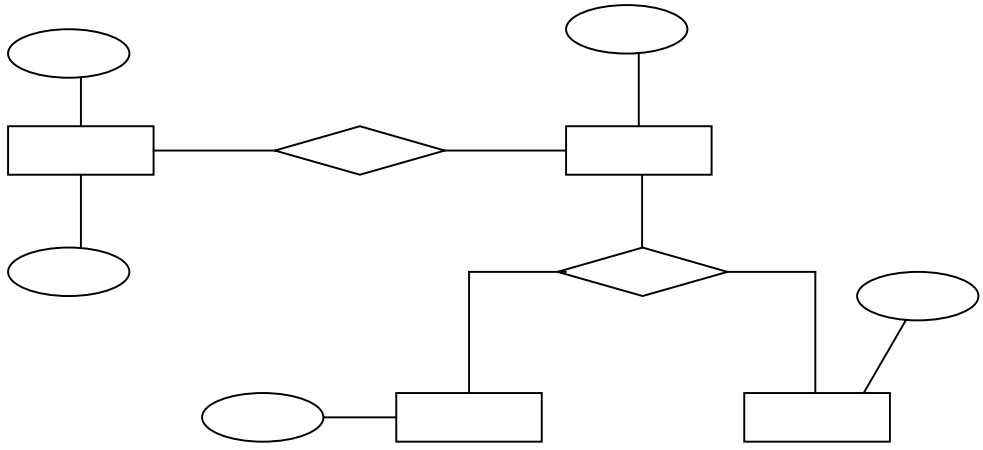
Thực thể (entity) là một vật thể tồn tại và phân biệt được với các vật thể khác. Chẳng hạn, mỗi cán bộ giảng dạy trong trường đại học là một thực thể, mỗi sinh viên là một thực thể, mỗi môn học là một thực thể,...

Một nhóm bao gồm các thực thể "tương tự" nhau tạo thành một *tập thực thể*.

Chẳng hạn, tập hợp các sinh viên trong khoa Công nghệ Thông tin là một tập thực thể, trong đó mỗi sinh viên là một thực thể; tập hợp các môn học cho sinh viên ngành Tin học là tập thực thể, trong đó mỗi môn học là một thực thể.

Để xác định một tập thực thể cần phải thiết lập một số hữu hạn các tính chất đặc trưng của tất cả các thực thể trong tập thực thể đó, gọi là các thuộc tính. Lựa chọn các tập thực thể là một bước quan trọng trong việc xây dựng sơ đồ về mối quan hệ thực thể phản ánh thông tin quản lý cho một thế giới thực nào đó.

Mô hình E-R thường được biểu diễn dưới dạng sơ đồ (sơ đồ E – R).



Trong đó:

- Các hình chữ nhật biểu diễn các tập thực thể

- Hình thoi biểu diễn mối quan hệ, chúng được liên kết với các tập thực thể bằng các cạnh vô hướng hoặc có hướng.

- Hình Oval biểu diễn thuộc tính, chúng được liên kết với các tập thực thể bằng các cạnh vô hướng.

Trong thực tế, có nhiều hệ thống thông tin được thiết kế xuất phát từ mô hình E – R.

Dựa vào mô hình E-R, các mô hình E-R mở rộng cũng đã được đề xuất nhằm biểu diễn sự phong phú và phức tạp của thế giới thực (thiết kế UML là một ví dụ).

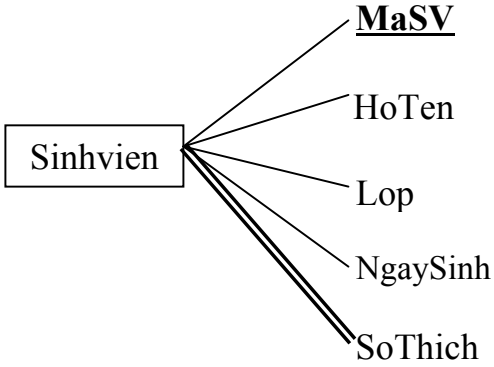
2.2. Các thành phần cơ bản

2.2.1. Tập thực thể

Mỗi tập thực thể có một tập các tính chất đặc trưng, mỗi tính chất đặc trưng này được đặt bởi một tên gọi là **thuộc tính** (attribute) của tập thực thể. Thông tin về mỗi thực thể trong tập thực thể được xác định bởi một bộ giá trị các thuộc tính. Ứng với mỗi thuộc tính có một tập các giá trị cho thuộc tính đó gọi là miền. Lựa chọn tập các thuộc tính cho các tập thực thể là một bước quan trọng trong thiết kế một sơ đồ CSDL quan niệm.

Một thuộc tính hay một tập tối thiểu các thuộc tính mà các giá trị của nó xác định *duy nhất* một thực thể trong một tập thực thể gọi là **khóa** (key) cho tập thực thể đó. Trong các ví dụ sau này ta quy ước khóa của một tập thực thể chỉ có một thuộc tính và thuộc tính đó được gọi là thuộc tính khóa.

Ví dụ: Để quản lý tập các sinh viên trong một trường đại học, người ta có thể sử dụng tập thực thể sinh viên bao gồm một số các thuộc tính sau:



Ta có một thể hiện của tập thực thể Sinhvien, chẳng hạn:
(K25-15, Lê Văn Nam, Tin_K30B, 15/12/83, {âm nhạc, bóng đá})

Lưu ý trong việc thiết kế các tập thực thể:

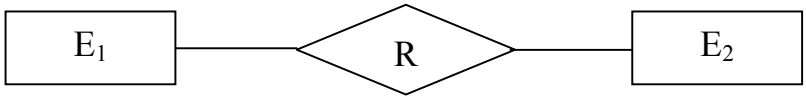
- Thứ nhất, phát hiện một tập thực thể bằng cách phát hiện tập các đối tượng mà ta cần quản lý (có 2 phần tử trở lên). Từ đó xác định các thông tin cần quản lý cho tập thực thể (các thông tin đó chính là các thuộc tính).
- Thứ hai, cần có thuộc tính khoá cho mỗi tập thực thể.

- Thứ ba, không sử dụng thuộc tính mà dữ liệu của nó được lấy từ thuộc tính của tập thực thể khác, mà thay vào điều đó là mối quan hệ giữa các tập thực thể.

2.2.2. Mối quan hệ giữa các tập thực thể

Một mối quan hệ trong mô hình E - R biểu thị quan hệ giữa các thực thể của các tập thực thể.

Mối quan hệ R giữa hai tập thực thể E_1 và E_2 được biểu diễn trong sơ đồ E - R như sau:

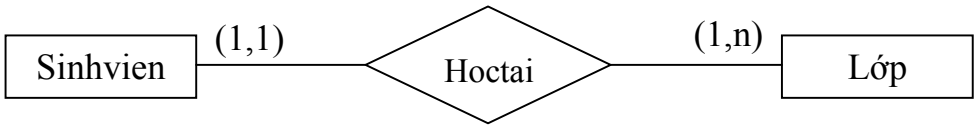


Ta có thể diễn tả khái niệm mối quan hệ giữa các tập thực thể một cách hình thức như sau**:** Mối quan hệ R trên các tập thực thể E_1, E_2,..., E_n là một tập con của tích Descartes E_1 x E_2 x...x E_n. Vì vậy, một thể hiện của R là một bộ n thành phần (e_1, e_2,..., e_n), gọi tắt là n-bộ, trong đó $e_i \in E_i$ (i = 1..n). Nếu n-bộ (e_1,e_2,...,e_n) là một thể hiện của R thì ta nói rằng e_1, e_2,..., e_n có mối quan hệ R với nhau.

Lưu ý: Một mối quan hệ phải tương ứng với một ngữ nghĩa xác định. Ví dụ xét hai tập thực thể: Sinhvien (tập các thực thể sinh viên) và Lớp (tập các thực thể lớp học), xét mối quan hệ Hoctai có ngữ nghĩa như sau:

(s, l) $\in$ Hoctai (với s $\in$ Sinhvien, l $\in$ Lớp) $\Leftrightarrow$ Sinh viên s học tại lớp l.

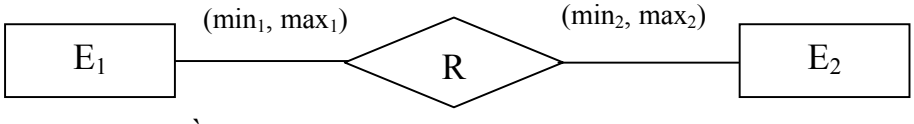
Sơ đồ biểu diễn mối quan hệ bằng E - R.



Lưu ý:

- **Ràng buộc về các bản số của một mối quan hệ:** trên mỗi cung nối giữa hình chữ nhật và hình thoi phải có cặp (min, max) được gọi là bản số của mối quan hệ. Nếu min/max lớn hơn 1, ta có thể viết tắt là n.

Để xác định một mối quan hệ là thuộc loại nào (1-1, 1-n, hay n-n), ta cần dựa vào bản số. Cụ thể, cho mối quan hệ R như sau:



Ràng buộc này chỉ ra rằng :

- Mỗi phần tử (thực thể) của E_1 có mối quan hệ R với ít nhất là min_1 phần tử của E_2, và nhiều nhất là max_1 phần tử của E_2.

- Tương tự, mỗi phần tử của E_2 có mối quan hệ R với ít nhất là min_2 phần tử của E_1, và nhiều nhất là max_2 phần tử của E_1.

Khi đó, mối quan hệ R giữa E_1 và E_2 là mối quan hệ: $\mathbf{max_2 - max_1}$

- **Các thuộc tính của một mối quan hệ:** một mối quan hệ cũng có thể có các thuộc tính của riêng nó (đặc biệt là các mối quan hệ n - n). Các thuộc tính của một mối quan hệ quy ước rằng chỉ là các thuộc tính đơn trị.

Trong trường hợp mối quan hệ R có thuộc tính, nếu R là mối quan hệ 1-1 thì ta có thể chuyển thuộc tính này thành thuộc tính của một trong hai tập thực thể tham gia, và nếu R là mối quan hệ 1-n thì chuyển thuộc tính này thành thuộc tính của tập thực thể tương ứng với phía nhiều.

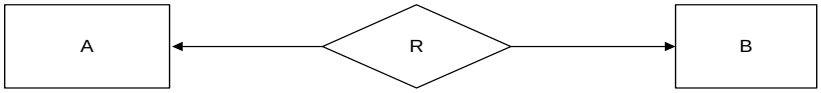
2.3. Phân loại mối quan hệ

2.3.1. Mối quan hệ nhị nguyên

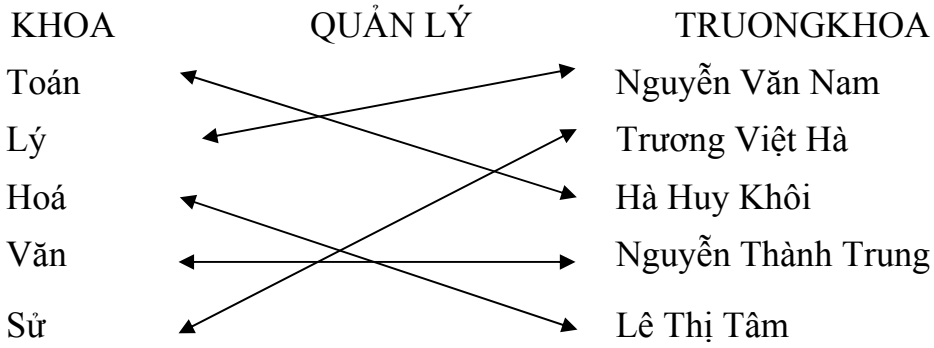
Đây là mối quan hệ giữa hai tập thực thể, bao gồm:

Quan hệ một - một: Mối quan hệ R giữa tập thực thể A và tập thực thể B được gọi là mối quan hệ một-một (hay 1-1) nếu mỗi thực thể của A có quan hệ R với duy nhất một thực thể của B và ngược lại mỗi thực thể của B có quan hệ R duy nhất với một thực thể của A.

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

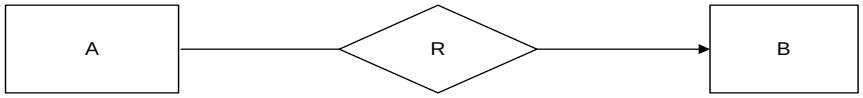


Ví dụ: Giả sử chúng ta đang xét hai tập thực thể sau: tập thực thể KHOA, bao gồm tất cả các khoa trong một trường đại học nào đó và tập thực thể TRUONGKHOA, bao gồm tất cả các trưởng khoa trong trường này. Mối quan hệ QUANLY giữa các tập thực thể KHOA và TRUONGKHOA theo nghĩa trưởng khoa X có quan hệ QUANLY với khoa Y nếu X là trưởng khoa của khoa Y. Rõ ràng rằng mối quan hệ này là một-một, vì rằng mỗi khoa có một trưởng khoa và mỗi trưởng khoa quản lý một khoa duy nhất. Ta có thể thấy mối quan hệ này một cách trực quan bởi hình sau:

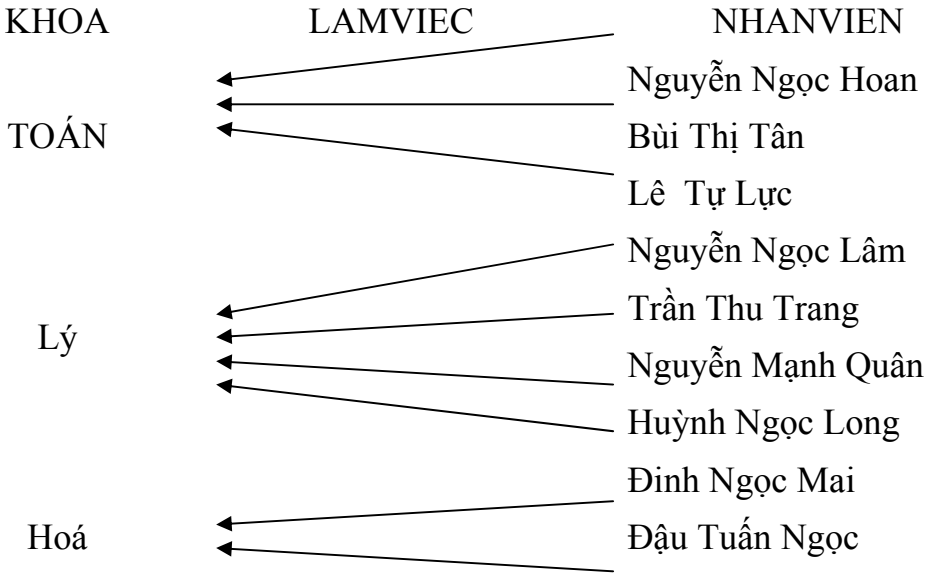


Quan hệ nhiều - một: Giả sử R là mối quan hệ giữa hai tập thực thể E_1 và E_2. Nếu một thực thể E_2 liên kết với 0 hoặc nhiều thực thể của E_1, và mỗi thực thể trong E_1 liên kết với nhiều nhất một thực thể của tập thực thể E_2 thì nói rằng R là mối quan hệ nhiều - một từ E_1 vào E_2.

Nếu R là mối quan hệ nhiều - một từ A vào B thì ta vẽ một cạnh định hướng từ hình thoi nhãn R vào hình chữ nhật nhãn B và một cạnh không định hướng từ hình thoi nhãn R vào hình chữ nhật nhãn A.

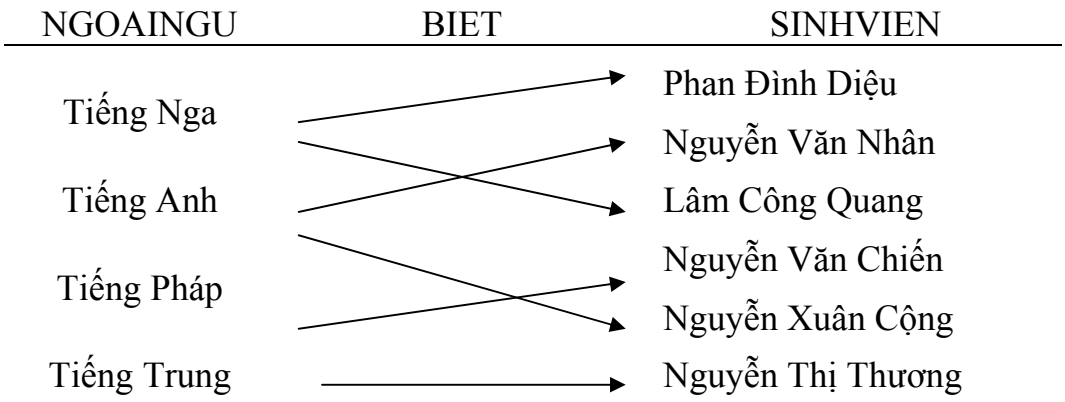


Ví dụ: Giả sử KHOA là tập thực thể bao gồm tất cả các khoa còn NHANVIEN là tập thể tất cả các cán bộ giảng dạy trong một trường đại học. Mối quan hệ LAMVIEC giữa KHOA và NHANVIEN theo nghĩa " nhân viên x làm việc trong khoa Y" là quan hệ nhiều - một từ tập thực thể NHANVIEN vào tập thực thể KHOA . Mối quan hệ này có thể được diễn tả bởi hình sau:



Quan hệ nhiều - nhiều: Cho hai tập thực thể E_1, E_2 và mối quan hệ R giữa chúng. Nếu một thực thể của E_1 có quan hệ R với 0 hoặc nhiều thực thể của E_2 và ngược lại, mỗi thực thể của E_2 có quan hệ R với 0 hoặc nhiều thực thể của E_1 thì ta nói rằng R là mối quan hệ nhiều-nhiều giữa E_1 và E_2 .

Ví dụ: Giả sử SINHVIÊN là tập thực thể các sinh viên cần khảo sát trong một trường đại học, còn NGOAINGU là tập thực thể các ngoại ngữ mà các sinh viên đã được học, thì mối quan hệ **BIET** với nghĩa "sinh viên x biết ngoại ngữ y" là một quan hệ nhiều - nhiều, vì rằng một sinh viên có thể biết nhiều ngoại ngữ và mỗi một ngoại ngữ có thể được biết bởi nhiều sinh viên.

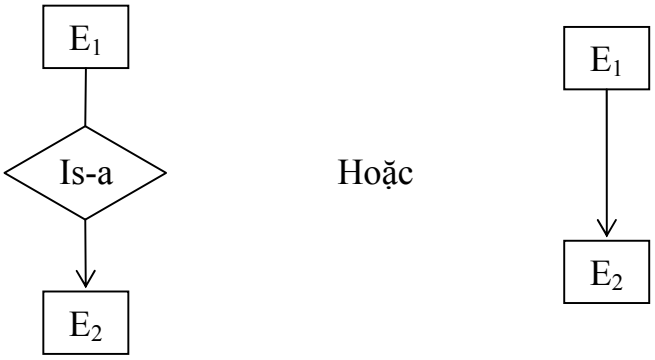


2.3.2. Mối quan hệ Is-a (mối quan hệ kế thừa)

Cho hai tập thực thể A và B chúng ta nói rằng A có mối quan hệ I-sa với B, ký hiệu là **A Isa B**, nếu mỗi thực thể của A là một thực thể của B.

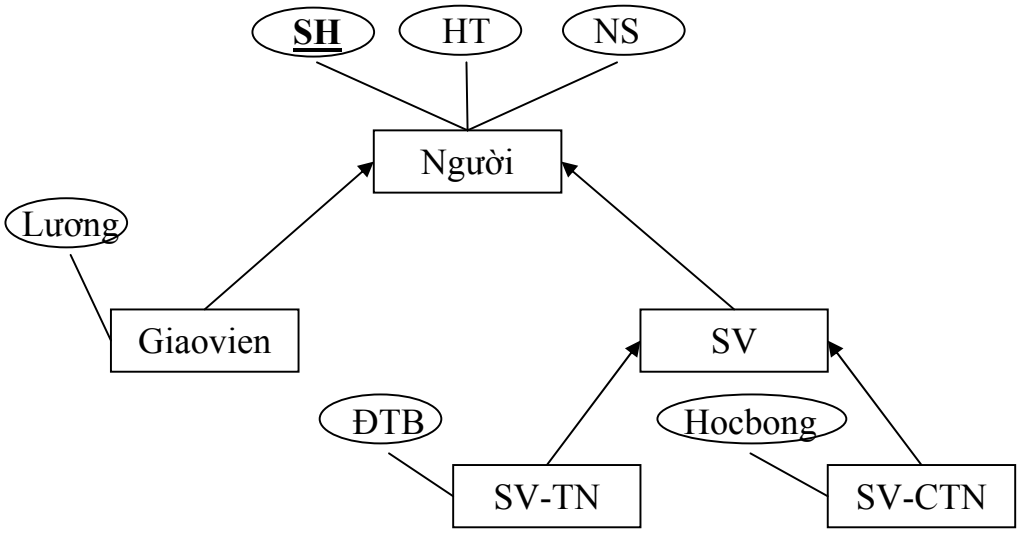
Như vậy A bao gồm các thuộc tính của B đồng thời thêm các thuộc tính khác. Chẳng hạn, B là tập thực thể các nhân viên của khoa Công nghệ Thông tin, A là tập thực thể các cầu thủ bóng đá trong khoa thì A Isa B, vì rằng một cầu thủ trong khoa cũng là một nhân viên của khoa. Ngoài những thuộc tính chung với tập thực thể A như họ tên, tuổi, học hàm, học vị, địa chỉ B còn thêm một số thuộc tính khác chẳng hạn như vị trí cầu thủ sẽ tham gia trong sân.

Mối quan hệ "Is-a" là trường hợp đặc biệt của mối quan hệ nhị nguyên 1-1. Ta có thể biểu diễn nó trong mô hình E-R như sau:



Nhận xét: Nếu E_1 Is-a E_2 thì mọi thực thể thuộc E_1 thì cũng thuộc E_2 và mọi thuộc tính nào có trong E_2 thì cũng có trong E_1.

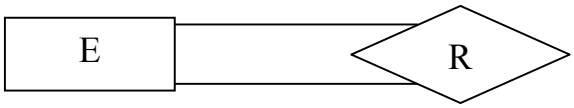
Ví dụ:



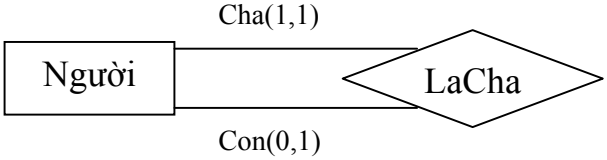
2.3.3. Mối quan hệ phản xạ (mối quan hệ đệ quy)

Mối quan hệ phản xạ là mối quan hệ giữa các thực thể của cùng một tập thực thể.

Biểu diễn:



Ví dụ:



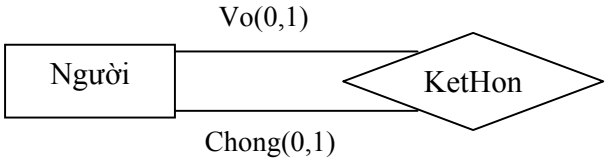
Ngữ nghĩa: $(n_1, n_2) \in$ LaCha $\Leftrightarrow$ n_1 là bố của n_2.

Lưu ý:

Đối với mối quan hệ phản xạ, chúng ta cần xác định rõ **tên vai trò** cho mỗi bản số có trong mối quan hệ này.

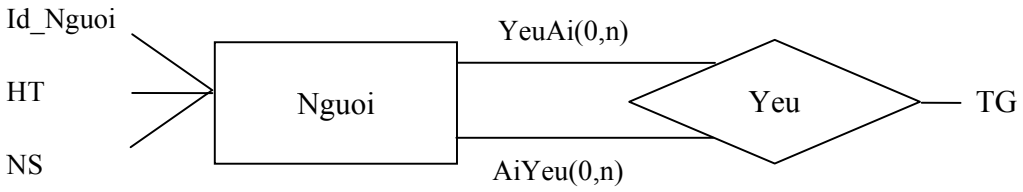
Các mối quan hệ phản xạ 1-1, 1-n, hoặc n-n cũng tương tự như mối quan hệ nhị nguyên 1-1, 1-n, n-n.

Ví dụ: (Mối quan hệ phản xạ 1-1)



Ngữ nghĩa: $(n_1, n_2) \in$ KetHon $\Leftrightarrow$ n_1 hiện là chồng của n_2

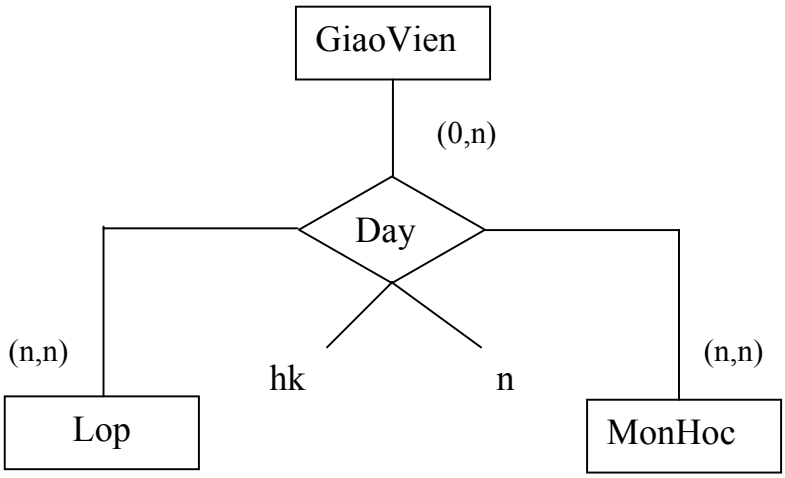
Ví dụ: (Mối quan hệ phản xạ n-n)



Ngữ nghĩa: $(n_1, n_2) \in$ Yeu $\Leftrightarrow$ n_1 từng yêu n_2 trong thời gian TG

2.3.4. Mối quan hệ đa nguyên

Mối quan hệ đa nguyên là mối quan hệ giữa 3 tập thực thể trở lên.



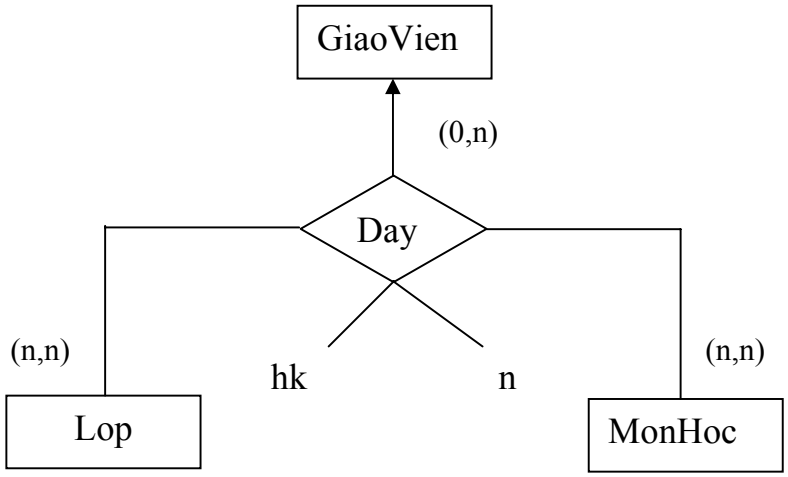
Ngữ nghĩa: (g, l, m) $\in$ Day $\Leftrightarrow$ giáo viên g dạy môn m cho lớp l vào học kỳ là hk của năm học n.

Lưu ý :

Ràng buộc hàm của mối quan hệ đa nguyên: Trong mối quan hệ đa nguyên, ngoài ràng buộc về bản số còn có "ràng buộc hàm". Ví dụ mối quan hệ Day nêu trên có ràng buộc hàm như sau:

$$\{Lop, MonHoc\} \rightarrow \{GiaoVien\}$$

(đọc là: Lop và MonHoc xác định GiaoVien). Điều này có nghĩa rằng: nếu biết trước lớp học l và môn học m thì xác định tối đa một giáo viên g của mối quan hệ. Khi đó, trên mô hình E-R sử dụng cung mũi tên như sau:



Chương 3. MÔ HÌNH QUAN HỆ

3.1. Quan hệ - lược đồ quan hệ

3.1.1. Quan hệ (Relation)

Quan hệ là một bảng mà không có dòng nào giống nhau, mỗi dòng của bảng được gọi là bộ (tuple) và mỗi cột của bảng được ký hiệu bằng một tên gọi là thuộc tính của quan hệ.

Ví dụ:

r =

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

Như vậy, ta có thể xem quan hệ như là một tập các bộ.

Ví dụ: $t = (a, d, b) \in r$

Vì vậy, người ta còn định nghĩa: quan hệ r trên tập thuộc tính $A_1, A_2, \ldots, A_n$:

$$r \subseteq Dom(A_1) \times Dom(A_2) \times \ldots \times Dom(A_n)$$

Trong đó, $Dom(A_i)$ là tập các giá trị có thể có của A_i (miền trị của A_i), với i = 1, 2, ..., n.

3.1.2. Lược đồ quan hệ (Relational Schema)

Lược đồ quan hệ là một cặp có thứ tự: $R = <U , SC>$

Trong đó:

- U là tập hữu hạn các thuộc tính của lược đồ quan hệ R.
- SC là tập các ràng buộc của lược đồ quan hệ R.

Ví dụ: Sinhvien = <U, SC>

U = {MaSV, Hoten, Ngaysinh}

SC: MaSV xác định duy nhất (khoá của Sinhvien).

Lưu ý: (Quan hệ r trên lược đồ quan hệ R)

Cho lược đồ quan hệ R = <U, SC>, khi đó quan hệ r được gọi là quan hệ trên R nếu r có tập thuộc tính U và thỏa mãn các ràng buộc trong SC

Ta xem: R = { r | r có tập thuộc tính U và thỏa các ràng buộc trong SC}

Vài thuật ngữ thông dụng:

Một tập các lược đồ quan hệ trong một hệ thống thông tin thì được gọi là một ***mô hình cơ sở dữ liệu quan hệ*** (có thể được gọi tắt là ***mô hình quan hệ***, hay: ***lược đồ cơ sở dữ liệu quan hệ***).

Tập hợp các quan hệ (hiện hành) của các lược đồ quan hệ trong một mô hình quan hệ thì được gọi là *cơ sở dữ liệu quan hệ*.

3.2. Khoá của quan hệ

Định nghĩa: Cho quan hệ r của lược đồ quan hệ R với tập thuộc tính $U = \{A_1, A_2, \ldots, A_n\}$, gọi tắt là lược đồ $R(A_1, A_2, \ldots, A_n)$.

Tập $X \subseteq U$ (X : tập thuộc tính) được gọi là ***khoá*** của quan hệ r nếu nó thỏa mãn cả 2 điều kiện:

i) Với mọi bộ $t \in r$ đều có giá trị khác nhau trên X, khi đó X được gọi là ***siêu khoá*** của r.

ii) $\nexists X' \subset X$ (tập con thực sự của X): X' là siêu khoá của r.

Ví dụ:

r =

A	B	C
a	b	c
b	b	a
c	a	a

Suy ra: r có hai khoá: {A}, {B, C}, viết tắt là A và BC.

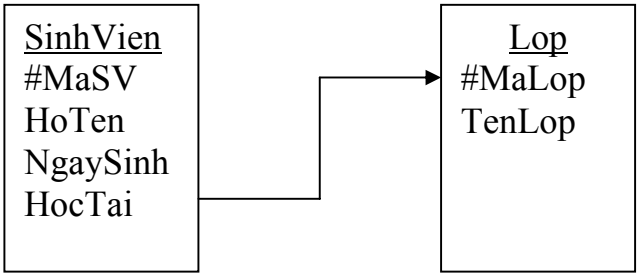
Lưu ý:

- X được gọi là *khoá của lược đồ quan hệ* R nếu X là khoá của mọi quan hệ r trên lược đồ quan hệ R.
- **Khoá chính (Primary key) của một lược đồ quan hệ**: Một lược đồ quan hệ có thể có nhiều khóa. Trong số đó, phải có đúng một khóa chính do người thiết kế cơ sở dữ liệu quy ước. Khóa chính của một lược đồ quan hệ thường được sử dụng cho việc tham chiếu dữ liệu. Trong sơ đồ biểu diễn một mô hình quan hệ, để biểu diễn ràng buộc khóa chính, ta sử dụng ký hiệu # ở ngay trước tên các thuộc tính của khóa chính
- **Khóa ngoại (Foreign key) của một lược đồ quan hệ:** Cho 2 lược đồ quan hệ $R_1 = <U_1, SC_1>$ và $R_2 = <U_2, SC_2>$. Gọi $PK \subseteq U_1$ là khoá chính của R_1. Xét $FK \subseteq U_2$. Khi đó, ta gọi FK *là khoá ngoại của lược đồ quan hệ* R_2 *tham chiếu đến* R_1 nếu cơ sở dữ liệu luôn thỏa mãn 2 điều kiện sau:

  i. Miền trị của FK là trùng với miền trị của PK.

  ii. Giá trị của FK hoặc là NULL hoặc phải bằng một giá trị hiện có nào đó của PK.

Ví dụ: Hình vẽ sau biểu diễn ràng buộc khóa ngoài: {HocTai} là khóa ngoài của lược đồ quan hệ SinhVien tham chiếu đến lược đồ quan hệ Lop.



3.3. Chuyển đổi mô hình E-R sang mô hình quan hệ

- Vào: Sơ đồ E-R
- Ra: Tập các lược đồ quan hệ (DB).

Quy ước về một số ký hiệu:

- ER: mô hình ER (giả thiết).
- DB: lược đồ cơ sở dữ liệu quan hệ cần tạo lập.
- U_R: tập tất cả các thuộc tính của lược đồ quan hệ R.
- Ω_E: tập tất cả các thuộc tính đơn vị của tập thực thể E.
- Ω_R: tập tất cả các thuộc tính của mối quan hệ R.
- PK_R: khoá chính của lược đồ quan hệ R (Primary Key).
- K_E: tập các thuộc tính khoá của tập thực thể E.
- min(E;R), max(E;R): các chỉ số tối thiểu & cực đại của bảng số trên cung nối tập thực thể F với mối quan hệ R.
- FK_R: tập tất cả các khoá ngoại của lược đồ quan hệ R.

Để chỉ FK là một khoá ngoại của lược đồ quan hệ R (tức là $FK \in FK_R$) tham chiếu đến khoá chính của quan hệ R' ta sử dụng ký hiệu: $FK \cong PK_{R'}$. Tên các thuộc tính có trong FK vẫn có thể khác với tên các thuộc tính có trong $PK_{R'}$ nhưng FK cần thỏa mãn đồng thời 2 điều kiện sau:

- Các thuộc tính trong FK phải có cùng miền trị với các thuộc tính tương ứng trong $PK_{R'}$.
- Giá trị của FK tại một bộ t chỉ có thể là NULL hoặc bằng giá trị $PK_{R'}$ tại một bộ t' nào đó $\in$ R'.

Các điều kiện trên của khoá ngoại FK đặc tả một ràng buộc toàn vẹn tham chiếu giữa hai quan hệ R & R'.

Lưu ý rằng, để chỉ ràng buộc toàn vẹn tham chiếu này đồng thời tên các thuộc tính có trong FK phải trùng tên với các thuộc tính tương ứng có trong $PK_{R'}$, ta sẽ ký hiệu: $FK \cong PK_{R'}$.

Khi đó, thuật toán chuyển đổi từ mô hình ER sang mô hình quan hệ lần lượt trải qua các bước sau:

Bước 1: Chuyển đổi các tập thực thể thành các lược đồ quan hệ.

Mỗi tập thực thể E được chuyển thành lược đồ quan hệ R(E) có cùng tên và cùng tập thuộc tính. Thuộc tính khoá của E chuyển thành khoá chính của R(E).

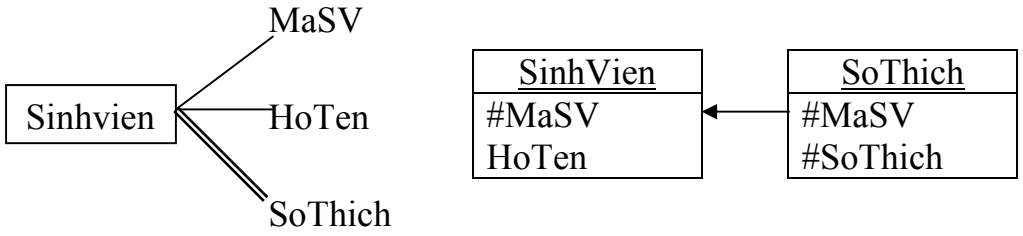
Lưu ý: Chuyển đổi thuộc tính đa trị

Nếu E có thuộc tính đa trị A thì trong lược đồ quan hệ, chúng ta phải tạo thêm lược đồ quan hệ mới để biểu diễn thuộc tính đa trị này.

$U_{R(A)} = \{PK_{R(E)}, A\}$

R(A) có khoá ngoại la $PK_{R(E)}$ tham chiếu đến $PK_{R(E)}$ cua R(E).

Lấy ví dụ thuộc tính sở thích của lược đồ quan hệ Sinhvien.



Chẳng hạn, ta có các quan hệ tương ứng với dữ liệu như sau:

SinhVien

MaSV	HoTen	
1	Lê Văn	A
2	Lê Văn	B
3	Lê Văn	C

SoThich

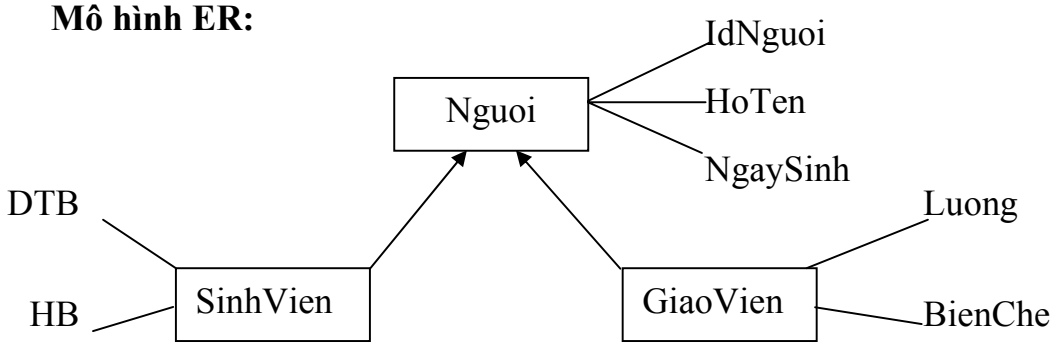
MaSV	SoThich
1	Phim
1	Bóng Đá
1	Nội Trợ
2	Phim
2	Bóng Đá

Chuyển đổi mối quan hệ Is – a (mối quan hệ kế thừa)

Để chuyển đổi mối quan hệ “Is - s” có hai cách thực hiện.

Ví dụ:

Mô hình ER:



Kết quả chuyển đổi:

Cách 1: (Không sử dụng lược đồ quan hệ biểu diễn lớp cha)

SinhVien
#MaSV
HoTen
NgaySinh
DTB
HB

GiaoVien
#MaGV
HoTen
NgaySinh
Luong
BienChe

Trong đó: MaSV, MaGV chính là IdNguoi

Cách 2: (Bổ sung khoá ngoại cho các lược đồ quan hệ biểu diễn lớp con)

SinhVien
#MaSV
DTB
HB

Nguoi
#IdNguoi
HoTen
NgaySinh

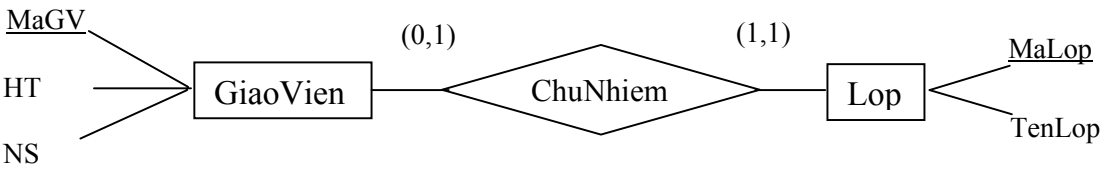
GiaoVien
#MaGV
Luong
BienChe

Bước 2: Chuyển đổi các mối quan hệ nhị nguyên 1 – 1.

Phương pháp: Bổ sung khoá ngoại cho một trong hai lược đồ quan hệ

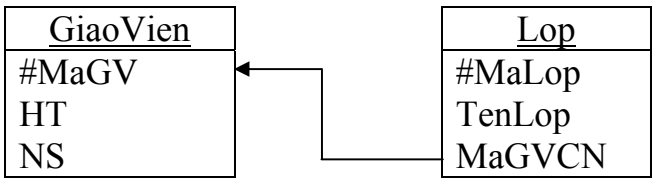
Ví dụ:

Mô hình ER



Ngữ nghĩa: (g, l)∈ ChuNhiem ⇔ Giáo viên g "hiện" là giáo viên chủ nhiệm của lớp l.

Kết quả chuyển đổi

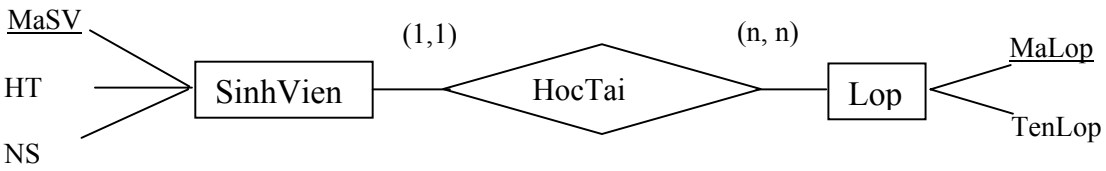


Bước 3: Chuyển đổi các mối quan hệ nhị nguyên 1 – n (một - nhiều)

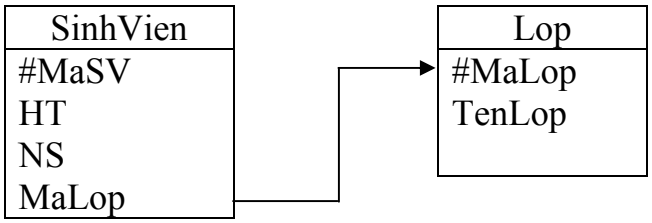
Phương pháp: Bổ sung khoá ngoại cho lược đồ quan hệ tương ứng với "phía nhiều".

Ví dụ:

Mô hình E-R



Kết quả chuyển đổi



Bước 4: Chuyển đổi các mối quan hệ nhị nguyên n-n (nhiều – nhiều)

Phương pháp: Xét mối quan hệ R giữa E_1 và E_2 là mối quan hệ n-n. Khi đó, ta cần tạo thêm một lược đồ quan hệ mới S để biểu diễn mối quan hệ R (đặt cùng tên với mối quan hệ). Trong đó, tập thuộc tính được xác định như sau: $U_S = PK_{R(E1)} \cup PK_{R(E2)} \cup \Omega$, với Ω là tập thuộc tính của mối quan hệ.

Các ràng buộc:

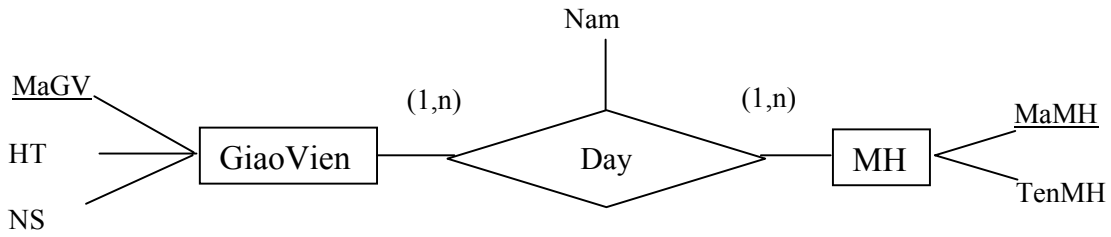
Khoá chính: $PK_S = PK_{R(E1)} \cup PK_{R(E2)}$

Khoá ngoại: S có hai khoá ngoại:

- $PK_{R(E1)}$ của S tham chiếu đến $R(E_1)$,
- $PK_{R(E2)}$ của S tham chiếu đến $R(E_2)$.

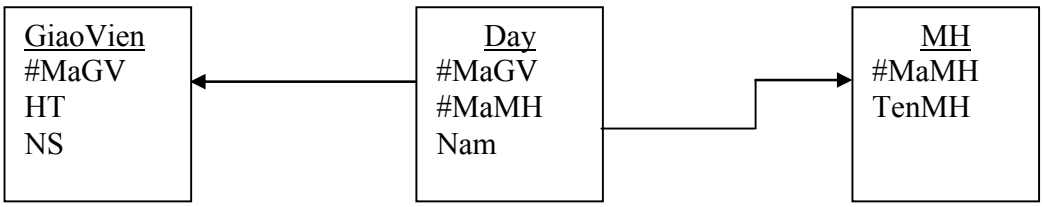
Ví dụ:

Mô hình E-R



Ngữ nghĩa: (g, m)∈ Day ⇔ Giáo viên g dạy môn học m bắt đầu từ năm Nam.

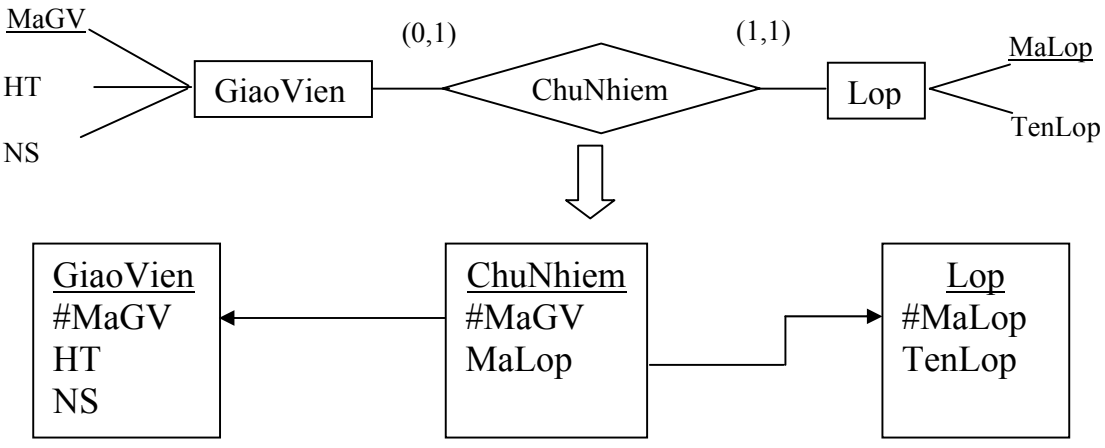
Kết quả chuyển đổi:



Lưu ý:

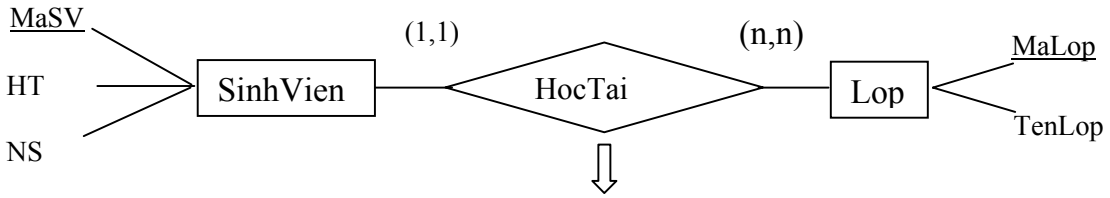
Đối với mối quan hệ 1-1, 1-n ta cũng có thể chuyển đổi tương tự như việc chuyển đổi đối với mối quan hệ n-n. Việc chuyển đổi chỉ khác về ràng buộc khoá chính mà thôi.

Mối quan hệ 1-1

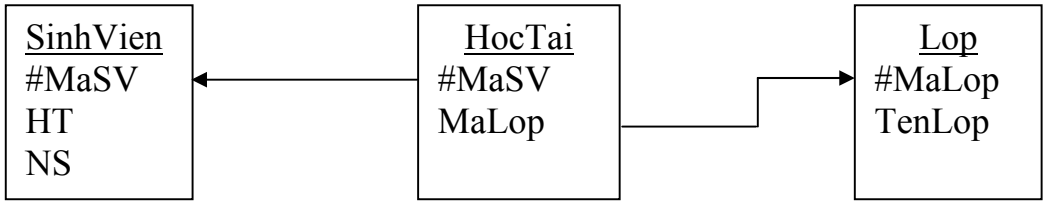


Trong trường hợp này, ta có thể chọn MaLop là khoá chính cũng được.

Mối quan hệ 1-n.



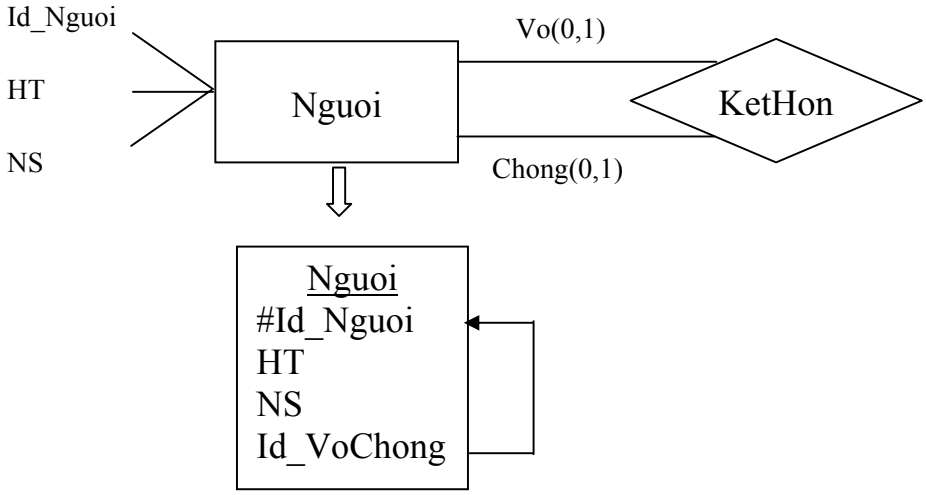
Kết quả chuyển đổi:



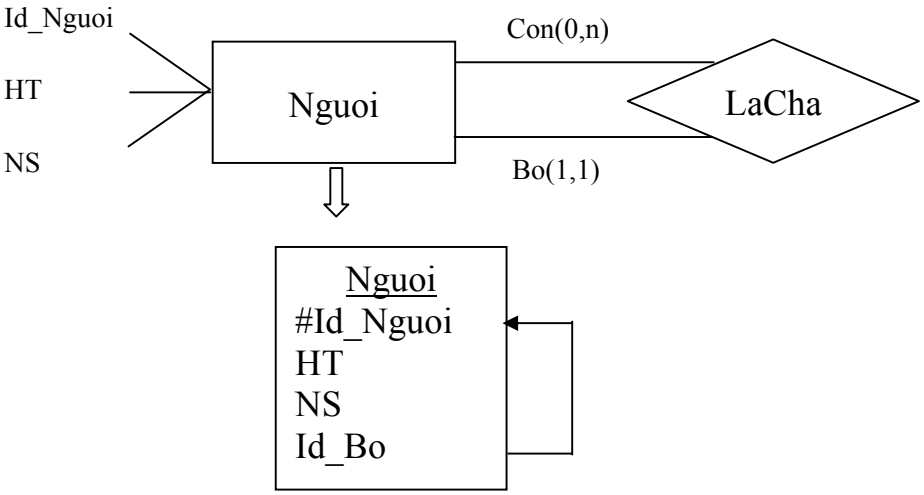
Bước 5: Chuyển đổi mối quan hệ phản xạ.

Được thực hiện tương tự như việc chuyển đổi mối quan hệ nhị nguyên 1-1, 1-n, n-n.

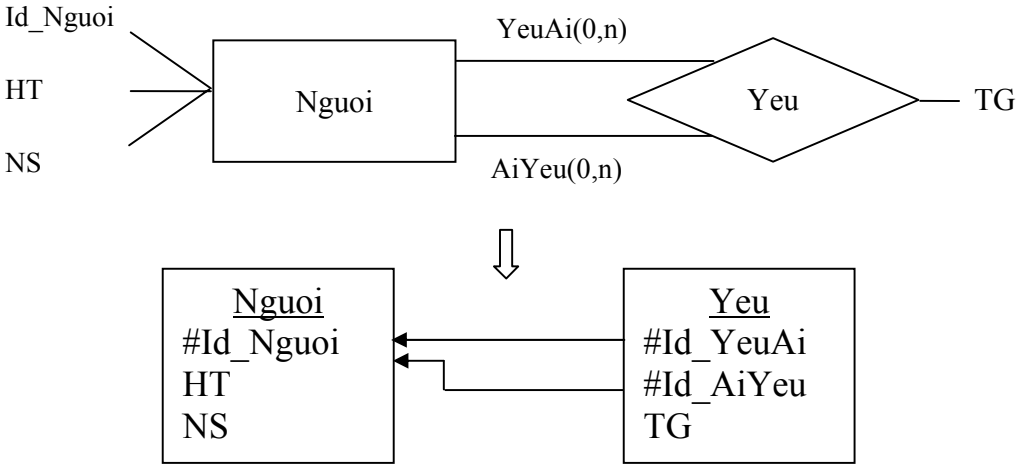
Ví dụ 1 (Mối quan hệ phản xạ 1-1)



Ví dụ 2 (Mối quan hệ phản xạ 1-n)



Ví dụ 3 (Mối quan hệ phản xạ n-n)

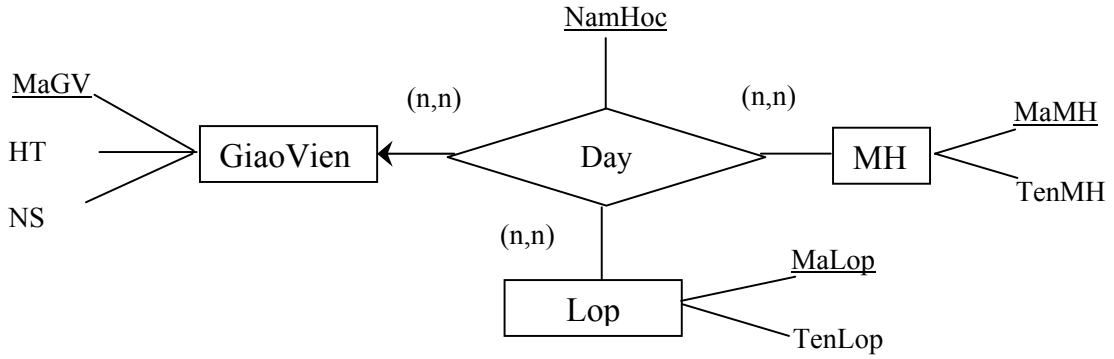


Bước 6: Chuyển đổi mối quan hệ đa nguyên.

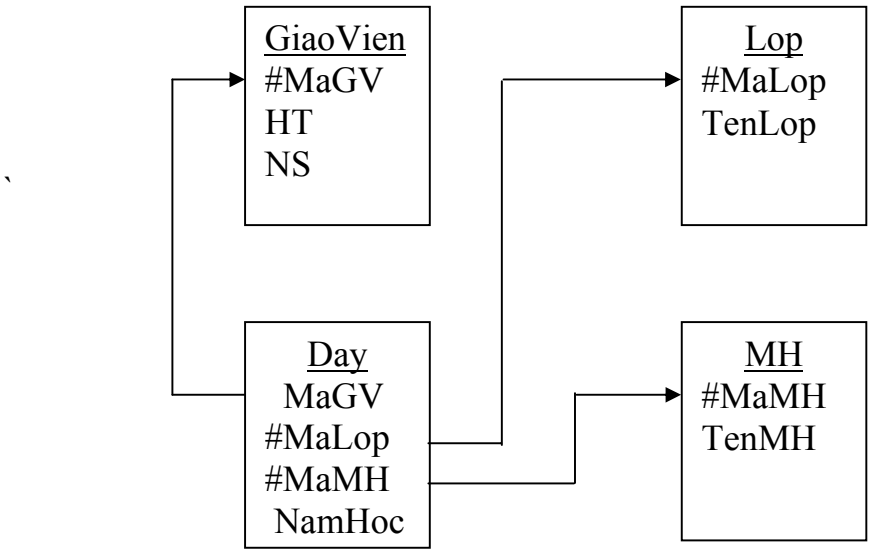
Phương pháp: Tương tự như phương pháp chuyển đổi mối quan hệ nhị nguyên n-n.

Ví dụ:

Mô hình E-R



Kết quả chuyển đổi:



Lưu ý:

Khoá chính của lược đồ quan hệ S là hợp các khoá chính của các lược đồ liên quan. Tuy nhiên, nếu có ràng buộc hàm thì phải loại bỏ khoá chính tương ứng với tập thực thể có mũi tên.

3.4. Đại số quan hệ

3.4.1. Phép hợp (Union)

Hợp của hai quan hệ r và s có cùng một lược đồ, ký hiệu $r \cup s$, được xác định như sau:

$r \cup s = \{t \mid t \in r \vee t \in s\}$

3.4.2. Phép giao (Intersection)

Giao của hai quan hệ r và s có cùng một lược đồ, ký hiệu $r \cap s$, được xác định như sau:

$r \cap s = \{t \mid t \in r \wedge t \in s\}$

3.4.3. Phép hiệu (Difference)

Hiệu của hai quan hệ r và s có cùng một lược đồ, ký hiệu r - s, được xác định như sau:

$r - s = \{t \mid t \in r \wedge t \notin s\}$

Lưu ý: Phép giao có thể biểu diễn thông qua phép hiệu: $r \cap s = r - (r - s)$

3.4.4. Tích Descartes (Cartersian Product)

Cho r và s là các quan hệ xác định lần lượt trên tập các thuộc tính $\{A_1, A_2, \ldots, A_n\}$ và $\{B_1, B_2, \ldots, B_m\}$. Tích Descartes của r và s, ký hiệu: $r \times s$, là tập các bộ có (m+n) thành phần sao cho n thành phần đầu là một bộ thuộc r và m thành phần sau là một bộ thuộc s. Hay:

$r \times s = \{(t_1,t_2) \mid t_1 \in r \wedge t_2 \in s\}$

Ví dụ: Xét

r =

A	B
a	b
a	c
b	c

s =

A	B
a	b
a	a

⇨ $r \cup s =$

A	B
a	b
a	c
b	c
a	a

$r \cap s =$

A	B
a	b

$r - s =$

A	B
a	c
b	c

Xét

q =

C	D	E
a	b	c
a	c	d

r =

A	B
a	b
a	c
b	c

$r \times q =$

A	B	C	D	E
a	b	a	b	c
a	b	a	c	d
a	c	a	b	c
a	c	a	c	d
b	c	a	b	c
b	c	a	c	d

Nhận xét: $|r \times s| = |r| \times |s|$

3.4.5. Phép chiếu (Projection)

Cho quan hệ r trên lược đồ quan hệ R = <U_R,SC>, và X ⊆ U_R, phép chiếu trên tập X của quan hệ r, ký hiệu: $\prod_X(r)$, được định nghĩa:

$$\prod_X(r) = \{t[X] \mid t \in r\}$$

Ở đây, t[X] để chỉ một bộ t chỉ chứa các thành phần có trong X.

Ví dụ: Xét ví dụ ở trên.

$\prod_A(r)$

A
a
b

$\prod_{CD}(q)$ =

C	D
a	b
a	c

Lưu ý:

$| \prod_X(r) | \leq | r |$

Để đơn giản ta có thể ký hiệu CD thay cho {C, D}

3.4.6. Phép chọn (Selection)

Cho quan hệ r và f là một biểu thức logic mà mỗi bộ trên r có thể thỏa mãn hoặc không. Khi đó, phép chọn trên r các bộ thỏa mãn f, ký hiệu $\delta_f(r)$, được xác định như sau:

$$\delta_f(r) = \{t \mid f(t) \text{ là đúng}\}$$

Ví dụ:

r =

A	B
a	b
a	c
b	c

$\delta_{A='a'}(r)$ =

A	B
a	b
a	c

3.4.7. Phép nối (Join)

Xét hai quan hệ r và s. Gọi f là một biểu thức logic mà mỗi bộ thuộc tích Descartes r × s có thể thỏa mãn hoặc không. Khi đó, phép nối r và s theo điều kiện f, được ký hiệu: $r \bowtie_f s$, được xác định như sau:

$$r \bowtie_f s = \delta_f(r \times s)$$

Ví dụ:

r =

A	B	C
a	1	1
b	2	1
c	0	2

s =

C	D
1	2
2	1

Với f = r.B ≥ s.D, ta có:

$r \underset{f}{\bowtie} s =$

A	B	r.C	s.C	D
a	1	1	2	1
b	2	1	1	2
b	2	1	2	1

Lưu ý: Nếu r và s có các thuộc tính trùng tên nhau thì phải đổi tên các thuộc tính này trong quan hệ thu được từ $r \underset{f}{\bowtie} s$.

Phép nối tự nhiên: Nếu r và s có các thuộc tính chung và f là một biểu thức logic để chỉ mỗi cặp thuộc tính chung này đều có giá trị bằng nhau, thì ta gọi phép nối này là phép nối tự nhiên và ký hiệu là: $r \bowtie s$.

Ví dụ: Xét ví dụ trên.

$r \bowtie s =$

A	B	C	D
a	1	1	2
b	2	1	2
c	0	2	1

Lưu ý: Quan hệ thu được từ phép nối tự nhiên có thể loại bỏ các thuộc tính lặp.

3.4.8. Phép chia (Division)

Gọi r là quan hệ trên tập thuộc tính $\{A_1, A_2, \ldots, A_n\}$

s là quan hệ trên tập thuộc tính $\{A_{p+1}, A_{p+2}, \ldots, A_n\}$, với $p > 0$.

Khi đó:

$r \div s = \{(a_1, a_2, \ldots, a_p) \mid \forall(a_{p+1}, a_{p+2}, \ldots, a_n) \in s; (a_1, a_2, \ldots, a_n) \in r\}$

Ví dụ:

$r =$

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

$s =$

C	D
c	d
e	f

$\Rightarrow r \div s =$

A	B
a	b
e	d

Nhận xét:

Phép chia có thể được định nghĩa thông qua phép toán khác. Với r, s lần lượt là quan hệ trên tập thuộc tính Z, X ($X \subseteq Z$).

$r \div s = T_1 - T_2$ (có tập thuộc tính: $Y = Z - X$)

$= T_1 - \prod_Y((s \times T_1) - r)$, với $T_1 = \prod_Y(r)$

Ví dụ: Xét ví dụ trên.

$T_1 = \prod_Y(r) =$

A	B
a	b
b	c
e	d

$T_1 \times s =$

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

$\theta = (T_1 \times s) - r =$

A	B	C	D
b	c	c	d
b	c	e	f

$\Rightarrow \prod_Y(\theta) =$

A	B
b	c

$\Rightarrow r \div s =$

A	B
a	b
e	d

Ứng dụng của đại số quan hệ

Dựa vào Đại số quan hệ được xét ở trên, người ta có thể xây dựng các biểu thức đại số quan hệ để trả lời các câu truy vấn.

Ví dụ:

Xét lược đồ quan hệ đã trình bày trong mục trước. Xét 1 CSDL hiện hành bao gồm các quan hệ sau: NGUOI, GIAOVIEN, SINHVIEN, LOP, MONHOC, DAY. Sử dụng các phép toán Đại số quan hệ để trả lời các câu hỏi sau:

- Liệt kê họ tên những sinh viên sinh vào ngày 1/1/1985.

  $\prod_{Hoten}(\delta_{Ngaysinh = \{1/1/1985\}}(Nguoi \bowtie SinhVien))$

  $= \prod_{Hoten}(Sinhvien \bowtie (\delta_{Ngaysinh = \{1/1/1985\}}(Nguoi)))$

- Cho biết họ tên của những giáo viên có dạy môn CSDL.

  $\prod_{Hoten}(\delta_{IdMon = 'CSDL'}(Nguoi \bowtie Giaovien \bowtie Day))$

- Tìm họ tên và lương những giáo viên dạy môn học trên 60 tiết cho lớp nào đó có sĩ số > 60.

$\prod_{Hoten,Luong}(\delta_{(Sotiet>60)\wedge(Siso>60)}(Nguoi \bowtie Giaovien \bowtie Day \bowtie Monhoc \bowtie Lop))$

$= \prod_{Hoten,Luong}(Nguoi \bowtie Giaovien \bowtie Day \bowtie \delta_{(Sotiet>60)}(Monhoc) \bowtie \delta_{(Siso>60)}(Lop))$

Chương 4. MÔ HÌNH HƯỚNG ĐỐI TƯỢNG

4.1. Giới thiệu chung

Mô hình hướng đối tượng (Object-Oriented Model) hình thành vào cuối những năm 1980, phát triển ở Mỹ với nhiều hệ quản trị cơ sở dữ liệu được xây dựng dựa trên nền tảng lý thuyết của mô hình này như: O_2, Object Store, Objective, Orion, Jasmin, ...

Cho đến nay, mô hình này vẫn chưa có chuẩn thống nhất.

4.2. Các thành phần cơ bản

4.2.1. Lớp, đối tượng và định danh đối tượng

Lớp được hiểu như là một tập các thực thể, hay các đối tượng có cùng các đặc tính và hành vi giống nhau. Các đặc tính này được mô tả như các thuộc tính bên trong một lớp đối tượng. Các hành vi chính là các phương thức (methods) được thực hiện trên mỗi đối tượng của lớp đó.

Mỗi đối tượng trong một lớp được xác định thông qua tên của đối tượng. Người ta sử dụng thuộc tính định danh OID (Object Identifier) để xác định tên duy nhất cho các đối tượng trong mỗi lớp.

Mô Hình Hướng Đối Tượng	Mô Hình E-R
Lớp	Tập thực thể
Đối tượng	Thực thể
Định danh đối tượng *(mặc định)*	Thuộc tính khoá *(do người sử dụng thiết kế)*

4.2.2. Thuộc tính và phương thức

Việc khai báo các thuộc tính thể hiện cấu trúc của lớp được khai báo. Mỗi thuộc tính có thể là thuộc tính đơn trị hoặc thuộc tính đa trị (sử dụng từ khoá "set" để khai bào). Ngoài ra, một thuộc tính có thể là một thuộc tính phức hợp: là thuộc tính được xác định từ tập các thuộc tính khác (sử dụng từ khoá "tuple" để khai báo).

Thuộc tính mối quan hệ là thuộc tính biểu diễn mối quan hệ giữa lớp này với lớp kia và giá trị của nó là OID của lớp kia.

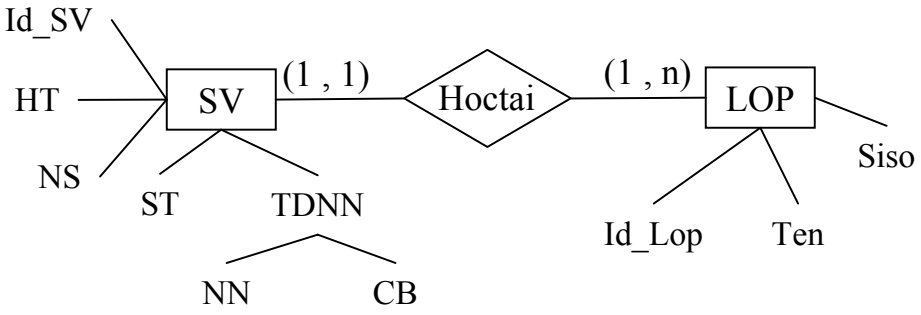
Việc khai báo các phương thức của mỗi đối tượng trong một lớp nhằm phản ánh các hành vi được thực hiện trên mỗi đối tượng thuộc lớp đó.

Mẫu đặc tả cho một lớp có thể được xác định như sau:

Class <tên lớp>
    properties
        {khai báo các thuộc tính}
    Operations
        {khai báo các phương thức}
end <tên lớp>.

Ví dụ:

- Mô hình ER mở rộng



⇒ Mô hình hướng đối tượng:

```
class  SV
  properties
      Id_SV :      string;
      HT :         string;
      NS :         Date;
      ST :         set(string);
      TDNN :       set( tuple( NN : string; CB : char(1)));
      Hoctai :     LOP;  {→ mang giá trị OID của class LOP}
  operations
      …….
end   SV;
{=======================================}
class  LOP
  properties
      Id_Lop :     string;
      Ten :        string;
      Siso :       integer;
      [Gom :       set(SV); {Inverse SV.Hoctai}]
  operations
      ……
end   LOP;
```

- Minh hoạ dữ liệu:

LOP:

OID	Id_Lop	Ten	Siso	Hoctai
Lop001	A	TinK25A	70	{SV1, SV2}
Lop002	B	TinK25B	80	{SV2, SV4}

SV:

OID	HT	NS	ST	TDNN (NN, CB)	Hoctai
SV1	X	1/1/85	{BB, BD, CN}	{(Anh, C), (Nga, B)}	Lop001
SV2	Y	11/3/84	{BĐ}	∅	Lop001
SV3	Z	1/1/85	∅	{(Anh, B)}	Lop002
SV4	T	11/3/84	{BB, CN}	{(Anh, C), (Nga, A)}	Lop002

Lưu ý:

Để khai báo thuộc tính ngược của một thuộc tính quan hệ, ta dùng từ khoá Inverse. Ví dụ: [Gom : set(SV); {Inverse SV.Hoctai}]

Điều này có nghĩa Gom là thuộc tính mối quan hệ ngược của thuộc tính Hoctai thuộc lớp SV.

Việc dùng thuộc tính ngược cũng có ưu điểm là thuận tiện trong việc biểu diễn các câu truy vấn. Tuy nhiên nó cũng có nhược điểm đó là gây dư thừa dữ liệu.

4.2.3. Phân cấp lớp và sự kế thừa

Các lớp trong mô hình hướng đối tượng có thể được tổ chức theo một phân cấp lớp, tương tự như mối quan hệ "Is-a" của mô hình E-R. Ta nói rằng lớp C_2 là lớp con của lớp C_1, có nghĩa là tập các đặc tính (các thuộc tính và phương thức) của lớp C_1 là tập con của tập các đặc tính của lớp C_2, đồng thời tập các đối tượng của lớp C_2 lại là tập con của tập các đối tượng của lớp C_1.

4.3. Chuyển đổi mô hình ER sang mô hình hướng đối tượng

Quá trình chuyển đổi được thực hiện qua các bước như sau:

Bước 1: Chuyển đổi các tập thực thể E

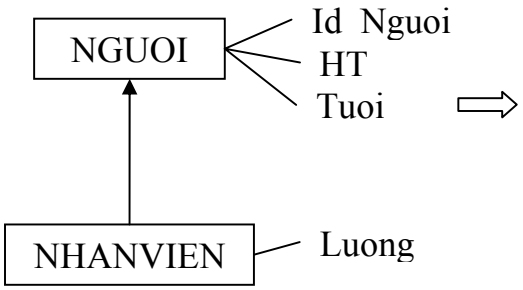
Mỗi tập thực thể của mô hình ER sẽ được chuyển đổi thành một lớp đối tượng có cùng tên và cùng tập thuộc tính. Ngoài ra, ta giả thiết rằng bất kỳ một mối quan hệ được xác định trong một mô hình ER chỉ có thể là giả thiết của một trong các bước được trình bày ở đây:

Lưu ý: (chuyển đổi mối quan hệ Is-a)

Nếu tập thực thể A có mối quan hệ Is-a với tập thực thể B thì lớp A sẽ kế thừa tất cả các thuộc tính trong lớp B bằng cách sử dụng từ khoá Inherits, đồng thời có thể bổ sung thêm các thuộc tính của lớp A.

Ví dụ:

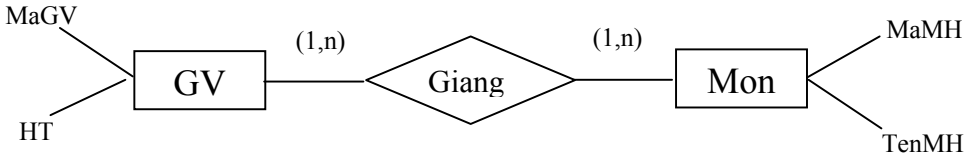
Mô hình ER **Mô hình hướng đối tượng**




```
class NGUOI
    Id_Nguoi : string;
    HT : string;
    Tuoi : integer;
end NGUOI;
class NHANVIEN
    inherits : NGUOI;
    Luong : integer;
end NHANVIEN;
```

Bước 2: Chuyển đổi mối quan hệ nhị nguyên 1-1, 1-n, n-n (không có thuộc tính)

Nếu 2 tập thực thể A và B có mối quan hệ R (R không có các thuộc tính đính kèm), thì mỗi lớp tương ứng A và B, ngoài các thuộc tính có trong tập thực thể A và B, sẽ được bổ sung thêm thuộc tính R (thuộc tính mối quan hệ).




```
Class GV
        MaGV:       string;
        HT:         string;
        Giang:      Set(Mon);
End   GV;
Class Mon
        MaMH:       string;
        TenMH:      string
        [DuocGiangBoi: Set(GV); {Inverse GV.Giang}]
End Mon;
```

Bước 3: Chuyển đổi mối quan hệ nhị nguyên n-n (có thuộc tính)

Nếu mối quan hệ R của hai tập thực thể A_1 và A_2 có kèm các thuộc tính, khi đó, ngoài 2 lớp A_1 và A_2 tương ứng, ta cần bổ sung thêm một lớp mới C đóng vai trò trung gian. Cụ thể:

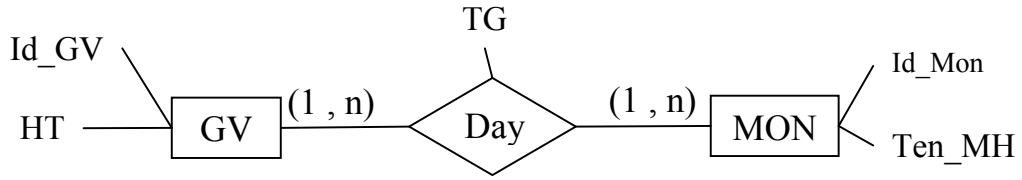
Lớp C bao gồm các thuộc tính sau:

- Các thuộc tính của mối quan hệ R.
- Hai thuộc tính có khai báo:

  <Tên thuộc tính > : <Lớp A_1>

  <Tên thuộc tính > : <Lớp A_2>

Ví dụ:



Mô hình hướng đối tượng:

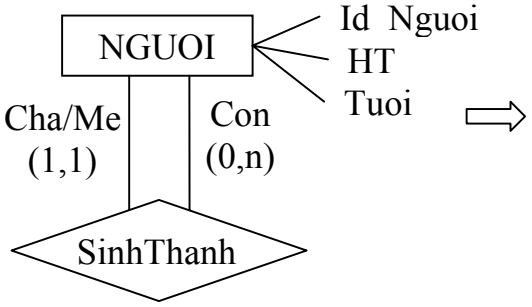
class GV Id_GV : string; HT : string; day1 : set(DAY); end GV;	class MON Id_Mon : string; Ten_MH : string; day2 : set(DAY); end MON;	class DAY day1 : GV; day2 : MON; TG : string; end DAY;

Bước 4: Chuyển đổi mối quan hệ phản xạ

Đối với mối quan hệ phản xạ, việc chuyển đổi được thực hiện tương tự như mối quan hệ nhị nguyên (bước 2 và bước 3).

Ví dụ:

Mô hình ER **Mô hình hướng đối tượng**




```
class NGUOI
    Id_Nguoi : string;
    HT : string;
    Tuoi : integer;
    Cha, Me : NGUOI;
    Con : set(NGUOI);
end NGUOI;
```

Bước 5: Chuyển đổi mối quan hệ đa nguyên

Đối với mối quan hệ đa nguyên, việc chuyển đổi được thực hiện tương tự như mối quan hệ nhị nguyên n-n có thuộc tính (bước 3).

Nếu k tập thực thể A_1, A_2, …, A_k có quan hệ với nhau thông qua mối quan hệ đa nguyên R bậc k, thì ngoài k lớp A_1, A_2, …, A_k ta sẽ bổ sung thêm một lớp mới C đóng vai trò trung gian. Cụ thể:

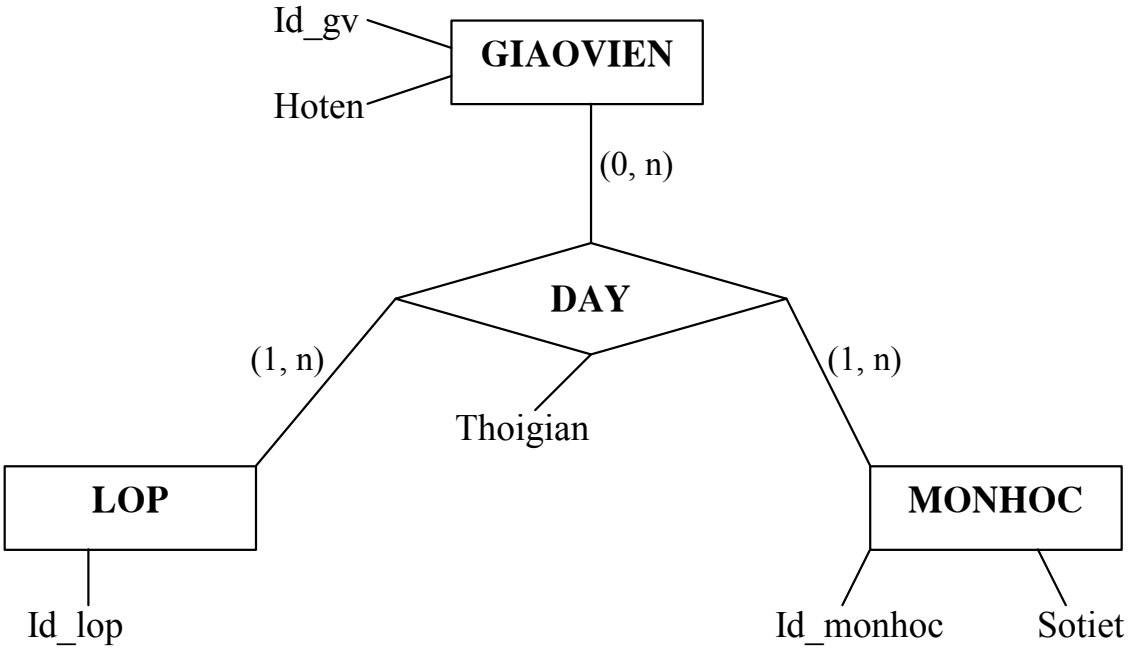
Lớp C bao gồm các thuộc tính sau:

- Các thuộc tính của mối quan hệ R.
- Các thuộc tính R_i có khai báo:

  <Tên thuộc tính > : <Lớp A_i>

Ví dụ:

Mô hình ER



Mô hình hướng đối tượng

Class **GIAOVIEN**
properties
Id_gv: String;
Hoten: String;
End GIAOVIEN.

Class **LOP**
properties
Id_lop: String;
End LOP.

Class **MONHOC**
properties
Id_monhoc: String;
Sotiet: Integer;
End MONHOC.

Class **LICHDAY**
properties
Thoigian: String;
Giang: GIAOVIEN;
Gomco: MONHOC;
Botri: LOP;
End LICHDAY.

Chương 5. LÝ THUYẾT THIẾT KẾ CƠ SỞ DỮ LIỆU QUAN HỆ

Như đã trình bày trong chương 3 mô hình CSDL quan hệ là một mô hình có tính độc lập dữ liệu cao, thuận lợi cho người sử dụng nên trong chương này chúng ta chúng ta đặc biệt quan tâm nghiên cứu một số phương pháp thiết kế một *mô hình CSDL tốt* từ một mô hình CSDL đã cho. Nội dung của chương chủ yếu trình bày các ràng buộc dữ liệu trong các quan hệ, các dạng chuẩn của các lược đồ CSDL quan hệ, và việc tách một lược đồ thành một số lược đồ con theo một dạng chuẩn nào đó.

5.1. Giới thiệu

Trong quá trình thiết kế một cơ sở dữ liệu quan hệ, một số vấn đề thường có thể xảy ra:

- ☞ Dư thừa dữ liệu
- ☞ Dị thường trong cập nhật dữ liệu

Ví dụ: Xét lược đồ quan hệ NC(TCC,DCC,TMH,GIA), trong đó TCC là tên người cung cấp, DCC là địa chỉ người cung cấp hàng, TMH là tên mặt hàng được cung cấp, GIA là giá tiền tương ứng của mặt hàng được cung cấp. Có một vài vấn đề nẩy sinh đối với lược đồ này:

Sự dư thừa: Địa chỉ của người cung cấp được lặp lại mỗi khi một mặt hàng được cung cấp. Chẳng hạn, trong quan hệ sau địa chỉ của ông Trần Ngọc An lặp lại 3 lần.

TCC	DCC	TMH	GIÁ
Trần Ngọc An	23. Lê Lợi	Khoai tây	4000
Lê Thị Nga	17. Hà nội	Cà chua	5000
Trần Ngọc An	23. Lê Lợi	Bắp cải	2000
Trần Ngọc An	23. Lê Lợi	Hành Tây	4000
Lê Văn Hoàn	18. Thuận An	Tỏi	100000
Đỗ Anh Tuấn	30. An Lăng	Cà rốt	8000

Bảng 5.1

Sự mâu thuẫn khi bổ sung: Nếu một người cung cấp hàng quen thuộc nào đó cung cấp mặt hàng mới nhưng địa chỉ anh ta đã thay đổi thì trong quan hệ sẽ xuất hiện sự mâu thuẫn (một người có hai địa chỉ). Chẳng hạn, ông Lê Văn Hoàn sau khi chuyển về cư trú tại 15. Võ Thị Sáu bán thêm cho siêu thị mặt hàng su hào với giá 7000 đồng/kg quan hệ sẽ trở thành như sau:

TCC	DCC	TMH	GIÁ
Trần Ngọc An	23. Lê Lợi	Khoai tây	4000
Lê Thị Nga	17. Hà nội	Cà chua	5000
Trần Ngọc An	23. Lê Lợi	Bắp cải	2000
Trần Ngọc An	23. Lê Lợi	Hành Tây	4000
Lê Văn Hoàn	18. Thuận An	Tỏi	1000
Đỗ Anh Tuấn	30. An Lăng	Cà rốt	8000
Lê Văn Hoàn	15. Võ Thị Sáu	Su hào	7000

Bảng 5.2

Sự bất thường khi loại bỏ*:* Khi cần xóa các mặt hàng được cung cấp bởi một người chúng ta lại xóa hết các thông tin về người đó. Như vậy có thể xãy ra trường hợp thông tin về một người cung cấp mặt hàng nào đó không thể tìm thấy trong CSDL. Chẳng hạn, nếu siêu thị không cần mặt hàng Cà chua thì cơ sở dữ liệu ở bảng trên cần phải loại dòng thứ 2 ra khỏi bảng khi đó còn lại là:

TCC	DCC	TMH	GIÁ
Trần Ngọc An	23. Lê Lợi	Khoai tây	4000
Trần Ngọc An	23. Lê Lợi	Bắp cải	2000
Trần Ngọc An	23. Lê Lợi	Hành Tây	4000
Lê Văn Hoàn	18. Thuận An	Tỏi	100000
Đỗ Anh Tuấn	30. An Lăng	Cà rốt	8000
Lê Văn Hoàn	15. Võ Thị Sáu	Su hào	7000

Bảng 5.3

Do đó các thông tin về nhà cung cấp cà chua cũng không còn trong CSDL và chúng ta không thể tìm khi cần thiết.

Sự bất thường khi bổ sung: Một nhà cung cấp chưa cung cấp hàng thì không thể đưa địa chỉ, tên nhà cung cấp vào quan hệ. Chúng ta có thể đặt giá trị null cho các thành phần TMH và GIA của một bộ cho người cung cấp mới nhưng khi chúng ta đưa vào một mặt hàng với người cung cấp này, liệu chúng ta có nhớ để xóa bộ chứa giá trị null hay không?

Trong ví dụ đã nêu, những nhược điểm trên được khắc phục nếu chúng ta thay quan hệ NC bằng hai quan hê:
NCC(TCC,DCC) và CC(TCC,TMH,GIA)
Khi đó quan hệ ở bảng 5.1 có thể được tách thành 2 bảng:

TCC	DCC
Trần Ngọc An	23. Lê Lợi
Lê Thị Nga	17. Hà nội
Lê Văn Hoàn	18. Thuận An
Đỗ Anh Tuấn	30. An Lăng

Bảng 5.4

TCC	TMH	GIA
Trần Ngọc An	Khoai tây	4000
Lê Thị Nga	Cà chua	5000
Trần Ngọc An	Bắp cải	2000
Trần Ngọc An	Hành Tây	4000
Lê Văn Hoàn	Tỏi	1000
Đỗ Anh Tuấn	Cà rốt	8000

Bảng 5.5

Rõ ràng rằng dữ liệu ở bảng 5.4 và 3.5 được tổ chức trên hai lược đồ mới cùng cho một lượng thông tin như quan hệ trên lược đồ ban đầu, đồng thời tránh được các nhược điểm đã nêu ở trên.

Có hai phương pháp để thiết kế một cơ sở dữ liệu quan hệ

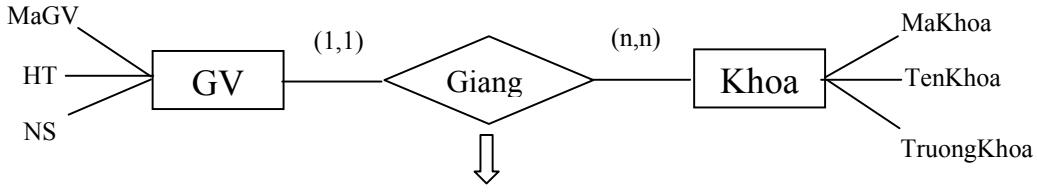
- ☞ Phương pháp 1: Thiết kế xuất phát từ mô hình E-R.
- ☞ Phương pháp 2: Thiết kế bằng phương pháp chuẩn hóa.

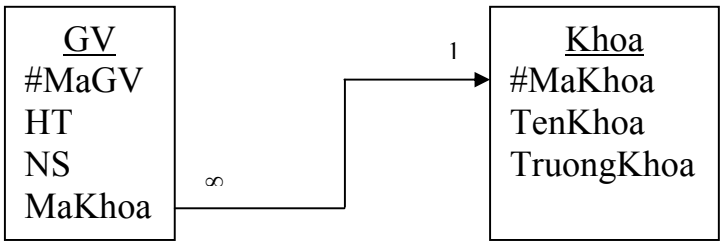
Ví dụ:

Giả sử ta cần xây dựng một hệ thống để quản lý thông tin của các sinh viên trong một trường đại học (MaSV, HT, NS) và thông tin về các khoa trong trường (MaKhoa, TenKhoa, TruongKhoa). Ngoài ra ta còn phải quản lý sinh viên thuộc khoa nào.

Thiết kế theo phương pháp 1.

Xây dựng mô hình E-R.





Thiết kế theo phương pháp 2.

Giả sử người ta chỉ sử dụng một lược đồ quan hệ để quản lý hệ thống này.

GiaoVien = <U, SC>

Trong đó: U = {MaGV, HT, NS, MaKhoa, TenKhoa, TruongKhoa}

Cho quan hệ r như sau:

MaGV	HT	NS	MaKhoa	TenKhoa	TruongKhoa
1	Le Van A	1/1/08	A1	CNTT	Mr.Han
2	Le Van B	1/2/08	A1	CNTT	Mr.Han
3	Le Van C	1/3/08	A2	Toan	Mr.Phung
4	Le Van D	1/4/08	A2	Toan	Mr.Phung

SC = {MaSV $\rightarrow$ U ; MaKhoa $\rightarrow$ TenKhoa.TruongKhoa}

Việc thiết kế cơ sở dữ liệu chỉ sử dụng một lược đồ quan hệ như trên có thể nảy sinh các vấn đề như sau: Dư thừa dữ liệu, dị thường trong cập nhật dữ liệu : sửa, bổ sung, xoá.

Dựa theo phương pháp thứ hai, ta sẽ phát hiện ra những điều này do lược đồ quan hệ SinhViên không đạt chuẩn 3NF (hoặc BCNF). Từ đó ta có thể thực hiện việc phân tách lược đồ quan hệ SinhVien thành hai lược đồ con SV và Khoa tương tự như kết quả cuối cùng của phương pháp 1.

5.2. Cơ sở lý thuyết của phụ thuộc hàm

5.2.1. Qui ước về các ký hiệu

- ☞ Các thuộc tính: A, B, C, …, A_1, A_2, …
- ☞ Tập các thuộc tính: X, Y, Z, …; ABC $\leftrightarrow$ {A, B, C}
- ☞ Hợp của các tập thuộc tính: XY $\leftrightarrow$ X$\cup$Y; XYZ $\leftrightarrow$ X$\cup$Y$\cup$Z; …
- ☞ Lược đồ quan hệ: R, S, …, RS = <U, SC> ;(SC : Set of Constraint)
- ☞ Quan hệ: r, s, …
- ☞ Bộ: t, t_1, t_2, …
- ☞ Với t là một bộ, X là tập thuộc tính thì ký hiệu t[X] để chỉ giá trị của bộ t trên tập thuộc tính X.

5.2.2. Phụ thuộc hàm (Functional Dependency)

Định nghĩa: (Quan hệ thỏa mãn phụ thuộc hàm)

Cho lược đồ quan hệ R = <U, SC>, cho X, Y $\subseteq$ U. Xét quan hệ r trên R. Quan hệ r được gọi là thỏa phụ thuộc hàm: X $\rightarrow$ Y (đọc là X xác định Y, hoặc Y phụ thuộc hàm vào X) nếu và chỉ nếu: $\forall$ t_1, t_2 $\in$ r sao cho:

$$t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$$

Nhận xét :

Do đó, r không thỏa $X \rightarrow Y \Leftrightarrow \exists t_1,t_2 \in r : t_1[X] = t_2[X] \Rightarrow t_1[Y] \neq t_2[Y]$

Ví dụ:

r =

A	B	C	D
a	b	c	a
a	c	a	a
b	c	b	a

- ☞ r không thỏa : $A \rightarrow B$
- ☞ r thỏa : $AB \rightarrow C$
- ☞ r không thỏa : $B \rightarrow C$
- ☞ r thỏa : $C \rightarrow A$
- ☞ r thỏa : $CD \rightarrow A$
- ☞ r không thỏa : $AD \rightarrow C$

Thuật toán: Kiểm tra quan hệ r có thỏa mãn phụ thuộc hàm $X \rightarrow Y$ không?

```
Function      Ktra(r, X, Y);
Begin
      Temp := true ;
      for i := 1 to n-1 do
        for j := i+1 to n do
            if  (ti[X]= tj[X] and ti[Y]<> tj[Y]) then
                  begin
                        temp:= false;
                        break;
                  end;
      Ktra:= temp;
End;
```

Định nghĩa: (Lược đồ quan hệ thỏa mãn phụ thuộc hàm)

Cho lược đồ quan hệ R = <U, SC>, cho X, Y $\subseteq$ U. R được gọi là thỏa phụ thuộc hàm $X \rightarrow Y \Leftrightarrow \forall r \in R$: r thỏa $X \rightarrow Y$.

$\Leftrightarrow \forall r \in R, \forall t_1, t_2 \in r : t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$

Lưu ý:

Thông thường ta quy ước rằng tập các ràng buộc SC chính là tập các phụ thuộc hàm và được ký hiệu là F. Vì vậy, một lược đồ quan hệ được ký hiệu là **R = <U, F>** thì R phải sẽ thỏa tất cả các phụ thuộc hàm trong F.

Định nghĩa: (Suy diễn logic)

Cho lược đồ quan hệ R = <U, F>, X, Y $\subseteq$ U.

Ta nói phụ thuộc hàm $X \rightarrow Y$ *được suy diễn logic từ* F, ký hiệu: $F \models X \rightarrow Y$, nếu với mỗi quan hệ r của R thỏa mãn các phụ thuộc hàm của F thì cũng thỏa $X \rightarrow Y$. Khi đó ta cũng nói rằng $X \rightarrow Y$ *là phụ thuộc hàm hệ quả* của F.
Ví dụ: $F = \{A \rightarrow B, B \rightarrow C\}$ thì $A \rightarrow C$ được suy diễn logic từ F (dễ thấy !)

Định nghĩa: (Bao đóng của tập phụ thuộc hàm)

Cho R = <U, F>. Bao đóng của tập phụ thuộc hàm F, ký hiệu là F^+, là tập tất cả các phụ thuộc hàm được suy diễn logic từ F, nghĩa là:

$$F^+ = \{X \rightarrow Y \mid X, Y \subseteq U \text{ và } F \models X \rightarrow Y\}$$

Ví dụ:

Cho R = <ABC, $\{A \rightarrow B, B \rightarrow C\}$>

Ta có $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, AB \rightarrow C, A \rightarrow A, ...\}$

Lưu ý: $F \subseteq F^+$

Định nghĩa: (Khoá của lược đồ quan hệ)

Cho R = <U, F>, cho $X \subseteq U$. Khi đó X được gọi là khoá của R nếu và chỉ nếu thỏa mãn đồng thời 2 điều kiện:

1. $X \rightarrow U \in F^+$ (hay: X là siêu khoá của lược đồ quan hệ R)
2. Không $\exists X' \subset X$: X' là siêu khoá của R

Ví dụ 1: Cho R = <U, F>

U = ABC; F = $\{A \rightarrow B, B \rightarrow C\}$

$\Rightarrow$ X = A là khoá của lược đồ quan hệ R

Ta chứng minh: $A \rightarrow ABC \in F^+ \Leftrightarrow$ R thỏa $A \rightarrow ABC \Leftrightarrow \forall r \in R$: R thỏa $A \rightarrow ABC$ (dễ dàng!).

Ví dụ 2: Cho R = <U, F>

U = ABC; F = $\{AB \rightarrow C, C \rightarrow A\}$

R có hai khoá là AB và AC (vì sao ?)

Lưu ý:

- ☞ Khoá của một lược đồ quan hệ là không duy nhất.
- ☞ Một lược đồ quan hệ luôn $\exists$ khoá.

5.2.3. Hệ tiên đề Amstrong

Cho R = <U, F>, Armstrong đã đưa ra ba qui tắc (tiên đề, luật) sau:

- ☞ *Luật phản xạ:*

Nếu $X, Y \subseteq U$ và $X \subseteq Y$ thì $Y \rightarrow X \in F^+$

- ☞ *Luật gia tăng (tăng trưởng):*

Nếu $X, Y, Z \subseteq U$ và $X \rightarrow Y \in F^+$ thì $XZ \rightarrow YZ \in F^+$

- ☞ *Luật bắc cầu:*

Nếu $X, Y, Z \subseteq U$ và $X \rightarrow Y \in F^+$, $Y \rightarrow Z \in F^+$ thì $X \rightarrow Z \in F^+$

Ta có các quy tắc suy diễn mở rộng:

Mệnh đề: Cho R = <U, F>, X, Y, Z, W $\subseteq$ U. Lúc đó:

1) Luật hợp: Nếu X$\rightarrow$Y và X$\rightarrow$Z thì X$\rightarrow$YZ

2) Luật tách: Nếu X$\rightarrow$Y và Z $\subseteq$ Y thì X$\rightarrow$Z

3) Luật tựa bắc cầu: Nếu X$\rightarrow$Y và WY$\rightarrow$Z thì WX$\rightarrow$Z

Định lý: Hệ tiên đề Armstrong là đúng đắn và đầy đủ.

Lưu ý:

- ☞ Tính đầy đủ của hệ tiên đề trên được hiểu là với bất kỳ phụ thuộc hàm hệ quả nào ta luôn có thể sử dụng một số hữu hạn quy tắc trong hệ tiên đề này để chứng minh.

Ví dụ:

Cho R = <U, F>, trong đó: U = ABC, F = {A$\rightarrow$B, A$\rightarrow$C}
Chứng minh: A $\rightarrow$ BC $\in F^+$
Ta có: A$\rightarrow$B (1)
A$\rightarrow$C (2)
Từ (1) $\Rightarrow$ A $\rightarrow$ AB (3) (Luật gia tăng)
Từ (2) $\Rightarrow$ AB $\rightarrow$ BC (4) (Luật gia tăng)
Từ (3) & (4) $\Rightarrow$ A $\rightarrow$ BC (Luật bắc cầu)
$\Rightarrow$ đpcm

5.2.4. Bao đóng của tập thuộc tính

Định nghĩa:

Cho R = <U, F>. Cho X $\subseteq$ U. Khi đó, bao đóng của X, ký hiệu là X^+, được định nghĩa như sau:

$$X^+ = \{A \mid X \rightarrow A \in F^+\}$$

Lưu ý:

- ☞ Để chỉ rõ bao đóng của X được xác định trên tập phụ thuộc hàm F, ta ký hiệu: X^+_F
- ☞ X là siêu khoá $\Leftrightarrow$ X $\rightarrow$ U $\in F^+ \Leftrightarrow X^+ = U$
- ☞ X $\subseteq X^+$

Định lý: (Bài toán thành viên: Điều kiện cần và đủ để X$\rightarrow$Y $\in F^+$)

Cho R = <U, F> và X, Y $\subseteq$ U. Khi đó:

$$X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X^+$$

Chứng minh: Giả sử Y = $A_1 \ldots A_n$, với $A_i \in U$ là các thuộc tính.

Đủ: Giả sử Y $\subseteq X^+$, suy ra $A_i \in X^+$, với mọi i =1,…,n.

Từ định nghĩa của X^+ ta có $X \to A_i$, suy ra $X \to Y$ (luật hợp).

Cần: Giả sử có $X \to Y \in F^+$, suy ra $X \to A_i \in F^+$ (luật tách).

Từ đó suy ra $Y \subseteq X^+$.

Ví dụ:

Cho R = <U, F>, trong đó: U = ABC, F = {A→B, A→C}

$AB \to C \in F^+$ do $(AB)^+ = ABC \supseteq C$

$B \to A \notin F^+$ (do $B^+ = B$ không chứa A)

5.2.5. Thuật toán tính bao đóng của tập thuộc tính

Vào: R = <U, F> và $X \subseteq U$

Ra: X^+_F

Phương pháp:

```
Function BaoDong(X);
Begin
    new:= X;
    Repeat
        old:= new;
        for mỗi X → Y ∈ F do
            if X ⊇ new thenm
                new:= new ∪ Y;
    Until old = new;
    Return new;
End;
```

Ví dụ:

Cho R = <U, F>, trong đó:

U = ABCDEG

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

Tính $(BD)^+$:

$X^{(0)} = BD$

$X^{(1)} = BD \cup \{EG\} = BDEG$

$X^{(2)} = BDEG \cup \{C\} = BDEGC$

$X^{(3)} = BDDEGC \cup \{A, AG\} = BDEGCA$

$X^{(4)} = BDEGCA = X^{(3)}$

$\Rightarrow (BD)^+ = ABCDEG$

Chứng minh BD là khoá của R:

Theo câu trên ta có BD là siêu khoá. Mặt khác ta có:

$B^+ = B \neq U \Rightarrow B \to U \notin F^+$

$D^+ = DEG \neq U \Rightarrow D \to U \notin F^+$

$\Rightarrow$ đpcm

Bài tập:

Viết thuật toán tựa Pascal nhằm kiểm tra tập X nào đó có phải là khoá của R hay không?

- ☞ Vào: R = <U, F> và $X \subseteq U$
- ☞ Ra: Yes/No

5.3. Phủ tối thiểu (phủ cực tiểu)

Định nghĩa: (Hai tập phụ thuộc hàm tương đương)

Cho 2 tập phụ thuộc hàm F và G. Khi đó: F được gọi là tương đương với G, ký hiệu: $F \Leftrightarrow G$, nếu và chỉ nếu $F^+ = G^+$.

Ví dụ:

Cho $F = \{A \rightarrow B, B \rightarrow C\}$, $G = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

$\Rightarrow F \Leftrightarrow G$

Ý nghĩa:

Cho R = <U, F> và G tương đương với F thì ta có thể dùng G thay cho F trong lược đồ quan hệ R. Tức là R = <U, F> = <U, G>

- ☞ Bổ đề 1: Nếu $F \subseteq G \Rightarrow F^+ \subseteq G^+$.
- ☞ Bổ đề 2: $(F^+)^+ = F^+$

Tính chất:

$$F \Leftrightarrow G \Leftrightarrow \begin{cases} F \subseteq G^+ (1) \\ G \subseteq F^+ (2) \end{cases}$$

Chứng minh, dùng 2 bổ đề ở trên:

($\Rightarrow$): $F \subseteq F^+ = G^+ \Rightarrow$ (1) đúng

$G \subseteq G^+ = F^+ \Rightarrow$ (2) đúng

($\Leftarrow$): $F \subseteq G^+ \Rightarrow (F)^+ \subseteq (G^+)^+ = G^+$ (bổ đề 1)

$G \subseteq F^+ \Rightarrow (G)^+ \subseteq (F^+)^+ = F^+$

Vậy $F^+ = G^+$, do đó F tương đương G.

Ý nghĩa:

Cho phép xây dựng thuật toán để kiểm tra xem hai tập phụ thuộc hàm có tương đương với nhau hay không.

Thuật toán kiểm tra F tương đương G

```
Procedure   KTTD(F, G) ;
Begin
    1. {Kiểm tra F ⊆ G+}
    for mỗi X→Y ∈ F do
        if X+G ⊉ Y then
            begin
```

write('F và G là không tương đương');
exit;
end;
2. {Kiểm tra $G \subseteq F^+$}
for mỗi $X \to Y \in G$ do
if $X^+_F \not\supseteq Y$ then
begin
write('F và G là không tương đương');
exit;
end;
3. writeln(' F tương đương với G');
End;

Bổ đề:

Cho R = <U, F>, $X \subseteq U$, $A_i \in U$, i = 1..n

Ta có: $X \to A_1A_2..A_n \in F^+ \Leftrightarrow \begin{cases} X \to A_1 \in F^+ \\ X \to A_2 \in F^+ \\ \vdots \\ X \to A_n \in F^+ \end{cases}$

Tính chất:

Cho R=<U, F>. Khi đó bao giờ cũng ∃ một phụ thuộc hàm G

- ☞ $G \Leftrightarrow F$
- ☞ Vế phải của mọi phụ thuộc hàm trong G chỉ có một thuộc tính.

Định nghĩa: (Phủ tối thiểu của một lược đồ quan hệ)

Cho R = <U, F>, F được gọi là phủ tối thiểu của R khi và chỉ khi:

a. Vế phải của mọi phụ thuộc hàm trong F là chỉ có 1 thuộc tính.

b. Mọi phụ thuộc hàm trong F không có thuộc tính dư thừa ở vế trái, nghĩa là: $\forall X \to A \in F, \forall B \in X,$

$$((F \setminus \{X \to A\}) \cup (X \setminus \{B\} \to A))^+ \neq F^+$$

c. Trong F không có phụ thuộc hàm nào là dư thừa, nghĩa là:

$$\forall X \to A \in F,\ X \to A \notin (F \setminus \{X \to A\})^+$$

Nhận xét:

1) Kiểm tra điều kiện a) khá dễ dàng.
2) Điều kiện b) $\Leftrightarrow \forall X \to A \in F, \forall B \in X,$

$$((F \setminus \{X \to A\}) \cup (X \setminus \{B\} \to A))^+ \neq F^+$$

Ta có thể xem điều kiện trên $\Leftrightarrow$ điều kiện sau:

$$\forall X \to A \in F, \forall B \in X \text{ thì } X \setminus \{B\} \to A \notin F^+$$

$\Leftrightarrow \forall X \to A \in F, \forall B \in X$ **thì** $(X \setminus \{B\})^+ \not\supseteq A$

3) Điều kiện c) $\Leftrightarrow \forall X \to A \in F,\ X \to A \notin (F \setminus \{X \to A\})^+$

$\Leftrightarrow \forall X \to A \in F, X^+_{F\setminus\{X\to A\}} \not\supseteq A$

Ví dụ:

R = <U, F>, U = ABC, F = {A→B, AB→C}

Hỏi F có phải là phủ tối thiểu của R không?

- ☞ Kiểm tra điều kiện a: đúng
- ☞ Kiểm tra điều kiện b:

Xét AB→C ∈ F, ta có:

X = AB, xem thử thuộc tính B có dư thừa không?

Ta có: $(X\setminus B)^+ = A^+ = ABC \supseteq C$ Suy ra thuộc tính B ở vế trái của phụ thuộc hàm AB→C là dư thừa. Vậy F không phải là phủ tối thiểu của R.

Ví dụ:

Cho R = <U, F>, U = ABC, F = {A→B, B→C, A→C}. Hỏi F có phải là phủ tối thiểu của R không?

- ☞ Kiểm tra điều kiện a: đúng
- ☞ Kiểm tra điều kiện b: đúng
- ☞ Kiểm tra điều kiện c:

Xét A→C ∈ F, ta có:

F \ {A→C} = {A→B, B→C}

$A^+_{\{A\to B,\ B\to C\}} = ABC \supseteq C$ Vậy trong F có phụ thuộc hàm A→C là dư thừa, do đó F không phải là phủ tối thiểu của R.

Bài tập:

Cho R = <U, F>. Viết giải thuật tựa Pascal để kiểm tra F có phải là một phủ tối thiểu của R hay không?

Định nghĩa: (Phủ tối thiểu của tập phụ thuộc hàm)

Cho R = <U, F>. Tập phụ thuộc hàm G được gọi là một phủ tối thiểu của F nếu thỏa hai điều kiện:

- ☞ $G \Leftrightarrow F$
- ☞ G là phủ tối thiểu của R' = <U, G>

Lưu ý:

Phủ tối thiểu của một phụ thuộc hàm là không duy nhất

Ví dụ:Cho R = <U, F>, U = ABC

F = {A→B, B→A, A→C, C→A, B→C, C→B}

Khi đó, F có các phủ tối thiểu khác nhau như sau: (?)

G_1 = {A→B, B→A, A→C, C→A}

G_2 = {A→B, B→A, B→C, C→B}

G_3 = {A→B, B→C, C→A}

Thuật toán (Tìm một phủ tối thiểu của F)

- ☞ Vào: R = <U, F>
- ☞ Ra: G (G là một phủ tối thiểu của T)
- ☞ Phương pháp:

Procedure TimPhuCucTieu(F);

1. Phân rã các phụ thuộc hàm trong F mà vế phải có 2 thuộc tính trở lên (làm cho F thỏa điều kiện a). Cụ thể:

$$X \to A_1A_2...A_n \xrightarrow{\text{Phân rã}} \begin{cases} X \to A_1 \\ X \to A_2 \\ \vdots \\ X \to A_n \end{cases}$$

2. (Làm cho F thỏa điều kiện b)

for (mỗi $X \to A \in F$) do

  for (mỗi $B \in X$) do

    if $((X \setminus \{B\})^+_F \supseteq A)$ then

      $X := X \setminus \{B\}$;

3. (Làm cho F thỏa điều kiện c)

for (mỗi $X \to A \in F$) do

  if $(X^+_{F\setminus\{X\to A\}} \supseteq A)$ then

    $F := F \setminus \{X \to A\}$;

4. Kết luận:

$G := F$;

End;

Ví dụ

Cho R = <U, F>, U = ABC, $F = \{A \to BC, AB \to C\}$

Tìm một phủ tối thiểu của F.

Bước 1:

$F = \{A \to B, A \to C, AB \to C\}$

Bước 2:

Xét $AB \to C \in F$, ta có:

$(AB\backslash B)^+ = A^+ = ABC \supseteq C$. Vậy thuộc tính B là dư thừa.

$(AB\backslash A)^+ = B^+ = B \not\supseteq C$. Vậy thuộc tính A không dư thừa.

$\Rightarrow F = \{A \to B, A \to C\}$

Bước 3:

Xét $A \to B$: $A^+_{F\setminus\{A\to B\}} = AC \not\supseteq B$

Xét $A \to C$: $A^+_{F\setminus\{A \to C\}} = AB \not\supseteq C$

Kết luận:

Vậy $G = \{A \to B, A \to C\}$ là một phủ tối thiểu của F cần tìm.

Lưu ý:

Trong thuật toán tìm một phủ tối thiểu của F, không thể thực hiện làm cho F thỏa điều kiện c) trước khi làm cho F thỏa điều kiện b) (nói cách khác, việc là cho F thỏa điều kiện c) luôn phải được thực hiện sau khi làm cho F thỏa điều kiện b)).

Phương pháp sai: (Bước 3 thực hiện trước bước 2)

Cho R = <U, F>, U = ABD, F = {B→D, B→A, D→A, AB→D}

☞ Nếu ta loại bỏ các phụ thuộc hàm dư thừa trước, thì lần lượt xét:

$B^+_{F\setminus\{B\to D\}} = BAD \supseteq D \Rightarrow$ loại bỏ $B \to D$.

$\Rightarrow F = \{B\to A, D\to A, AB\to D\}$

$B^+_{F\setminus\{B\to A\}} = B \not\supseteq A$

$D^+_{F\setminus\{D\to A\}} = D \not\supseteq A$

$(AB)^+_{F\setminus\{AB\to D\}} = AB \not\supseteq D$

☞ Tiếp đến, loại bỏ thuộc tính dư thừa ở vế trái, chỉ xét: $AB \to D$

Ta có: $B^+_F = BAD \supseteq D \Rightarrow$ loại bỏ A

$\Rightarrow F = \{B\to A, D\to A, B\to D\}$

☞ Thế nhưng, ta thấy $B\to A \in F$ là phụ thuộc hàm dư thừa.

Phương pháp đúng:

☞ Loại bỏ thuộc tính dư thừa ở vế trái, chỉ xét: AB→D

Ta có: $B^+_F = BAD \supseteq D \Rightarrow$ loại bỏ A

$\Rightarrow F = \{B\to A, D\to A, B\to D\}$

☞ Loại bỏ các phụ thuộc hàm dư thừa:

B→D:

Ta có: $B^+_{F\setminus\{B\to D\}} = BA \not\supseteq D$

B→A:

Ta có: $B^+_{F\setminus\{B\to A\}} = BDA \supseteq A \Rightarrow$ loại bỏ B→A

$\Rightarrow F = \{D\to A, B\to D\}$

D→A:

Ta có: $D^+_{F\setminus\{D\to A\}} = D \not\supseteq A$

☞ Kết luận: $F = \{D\to A, B\to D\}$

5.4. Các thuật toán xác định khoá của một lược đồ quan hệ

5.4.1. Các thuật toán xác định một khoá của một lược đồ quan hệ

X là khoá của lược đồ quan hệ R = <U, F> nếu:

1) $X \to U \in F^+$ hay $X^+_F = U$ (X là siêu khoá)

2) Không tồn tại $X' \subset X$, $X'^+ = U$

Thuật toán tìm một khoá của một lược đồ quan hệ.

☞ Vào: R =<U,F>

☞ Ra : K (một khoá của R)

☞ Phương pháp:

```
Function Key(R)
Begin
        K:= U;
        For mỗi A ∈ U do
                If (K\A)+F = U then
                        K := K \ {A}; {loại bỏ thuộc tính A}
        Return K;
End;
```

Ví dụ

R = <U,F>, U = ABCDEG; F = {B→C,C→B,A→GD}

K = ABCDEG

Ta có:

☞ $(K\backslash A)_F^+ = (BCDEG)^+ = BCDEG \neq U$

☞ $(K\backslash B)_F^+ = (ACDEG)^+ = ACDEGB = U$

⇒ K = ACDEG (loại B)

☞ $(K\backslash C)_F^+ = (ADEG)^+ = ADEG \neq U$

☞ $(K\backslash D)_F^+ = (ACEG)^+ = ACBEGD = U$

⇒ K = ACEG (loại D)

☞ $(K\backslash E)_F^+ = (ACG)^+ = ACGBD \neq U$

☞ $(K\backslash G)_F^+ = (ACE)^+ = ACEGDB = U$

Vậy **K = ACE** là khoá của lược đồ quan hệ.

Chú ý: Khi thay đổi thứ tự việc loại bỏ các phần tử của K, ta có thể thu được khóa khác của lược đồ quan hệ.

Chúng ta có thể cải tiến thuật toán trên thông qua định lý sau:

Định lý Hồ Thuần - Nguyễn Văn Bào: (Điều kiện cần để X là khoá).

Cho R = <U, F>, cho X ⊆ U. Khi đó, nếu X là khoá của R thì:

$$(U \setminus P) \subseteq X \subseteq (U \setminus P) \cup (T \cap P)$$

Trong đó: $T = \bigcup_{X \to Y \in F} X$

$$P = \bigcup_{X \to Y \in F} Y$$

Ý nghĩa:

Định lý này cho phép ta thu hẹp phạm vi tìm kiếm khoá

Ví dụ: Cho R = <U,F>, trong đó:

U = ABCDEG; F = {B→C,C→B,A→GD}

$$\Rightarrow \left.\begin{array}{l} T = ABC \\ P = BCDG \end{array}\right\} \Rightarrow T \cap P = BC$$

Gọi X là khoá của R: $U \setminus P = AE \subseteq X \subseteq (U \setminus P) \cup (T \cap P) = AEBC$

Nếu gọi K là tập tất cả các khoá của R $\Rightarrow$ K chỉ có thể chứa AE, AEB, AEC, AEBC.

Ta có: $(AE)^+ = AEGD \neq U \Rightarrow AE \notin K$

$(AEB)^+ = AEBGDC = U \Rightarrow AEB \in K$

$(AEC)^+ = AEBGDC = U \Rightarrow AEC \in K$

Có thể chỉ ra K = {AEB, AEC}

Ngoài ra, AEBC chỉ là siêu khoá của R

Hệ quả 1:

Nếu $(U \setminus P)^+ = U$ thì $U \setminus P$ là khoá duy nhất của R.

Ví dụ:

Xét R = <U, F>, với $U = A_1A_2A_3A_4A_5A_6$

$F = \{A_1 \rightarrow A_2, A_3 \rightarrow A_4, A_5 \rightarrow A_6\}$

$\Rightarrow P = A_2A_4A_6$ mà $(U \setminus P)^+ = (A_1A_3A_5)^+ = A_1A_3A_5A_2A_4A_6 = U$

$\Rightarrow$ R có 1 khoá duy nhất là $A_1A_3A_5$

Hệ quả 2:

Nếu $T \cap P = \varnothing$ thì $U \setminus P$ là khoá duy nhất của R.

Hệ quả 3:

$(U \setminus P) \cup (T \cap P)$ là siêu khoá của R.

Nhận xét:

Từ 3 hệ quả trên cho phép ta xác định giải thuật để tìm một khoá của lược đồ quan hệ R như sau :

Vào: R = <U, F>

Ra: Tìm 1 khoá của R

Phương pháp:

```
Function Key(R)
Begin
    K := (U \ P) ∪ (T ∩ P)
    For <mỗi A ∈ (T ∩ P)> do
        If <(K \ A)+ = U> then
            K := K \ A;  {loại bỏ thuộc tính A}
    Return K.
End;
```

Ví dụ:

R = <U,F>, U = ABCDEG; $F = \{B \to C, C \to B, A \to GD\}$

$$\left.\begin{array}{l} T = ABC \\ P = BCDG \end{array}\right\} \Rightarrow T \cap P = BC$$

$X = (U \setminus P) \cup (T \cap P) = ABCE$

$(X \setminus B)^+ = U \Rightarrow X = ACE$ (loại B)

$(X \setminus C)^+ = (AE)^+ \neq U$

$\Rightarrow ACE \in K$

Ngoài ra ta có giải thuật tìm một khoá K nhận S làm siêu khoá như sau :

Vào: + R = <U, F>
+ Siêu khóa S

Ra: Tìm khoá K của R nhận S làm siêu khoá

Phương pháp:

```
Function Key(R, S)
Begin
        K := S
        For  <mỗi A ∈ S \ (U \ P)> do
                If <(K \ A)+ = U> then
                        K := K \ A;   {loại bỏ thuộc tính A}
        Return K.
End;
```

5.4.2. Giải thuật xác định tất cả các khoá của một lược đồ quan hệ.

Phương pháp 1: Sử dụng cây tìm khóa

Ý tưởng:

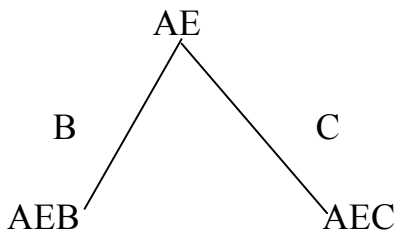
Dựng một cây có nút gốc là U\P, rồi thêm dần các thuộc tính còn thiếu trong T∩P cho đến khi được các siêu khoá

Ví dụ:

Cho R = <U, F> với U = ABCDEG

$F = \{B \to C, C \to B, A \to GD\}$ $\begin{cases} P = BCGD \\ T = ABC \end{cases} \Rightarrow \begin{cases} U \setminus P = AE \\ T \cap P = BC \end{cases}$

Dựng cây:



Tính $(AE)^+ = AEGD \neq U$

$(AEB)^+ = AEBCGD = U$

$(AEC)^+ = AECBGD = U$

Kết luận:

K = {AEB, AEC}

Mô tả phương pháp:

Bước 1: tính $(U\backslash P)^+$. Nếu $(U\backslash P)^+ = U$ thì kết luận K = {U\P}. Ngược lại, dựng cây có nút gốc là U\P và các nút con là: (U\P)A, với $A \in T \cap P$

Bước 2: lần lượt duyệt cây theo chiều rộng đối với các nút chưa được đánh dấu đến chừng nào mọi nút lá của cây đều được đánh dấu.

Lưu ý:

Tại mỗi nút X chưa được đánh dấu ta thực hiện thủ tục sau:

```
Procedure   DuyetNut (X);
Begin
    If X chứa nút đã đánh dấu khoá then
        Đánh dấu X là nút đã xét
    Else if  X+ = U then
        Đánh dấu X là nút khoá
    Else
        Tạo các nút con của X bằng cách bổ sung các thuộc tính còn
        thiếu trong T∩P.
End;
```

Phương pháp 2: Sử dụng định lý Lucchessi-Osborn

Định lý Lucchessi và Osborn: (điều kiện cần và đủ để bổ sung khoá)

Cho R = <U, F>. Gọi $\mathcal{K}$ là một tập khác rỗng các khoá của lược đồ quan hệ R. Khi đó, điều kiện cần và đủ để có thể bổ sung khoá mới vào $\mathcal{K}$ là:

$\exists K \in \mathcal{K}, \exists X \rightarrow Y \in F: T = X \cup (K \setminus Y)$ không chứa phần tử nào của K.

Lưu ý:

Phần chứng minh của định lý này chứng tỏ rằng nếu điều kiện trên được thỏa mãn thì khoá mới được bổ sung vào $\mathcal{K}$ là khoá k᾽ nhận T làm siêu khoá. Từ đây, ta có thể xác định được giải thuật tìm tất cả các khoá của R như sau:

Thuật toán tìm tất cả các khoá của một lược đồ quan hệ

Vào: R = <U, F>

Ra: K

Phương pháp:

```
Procedure TimTatCaKhoa(R);
Begin
    1. Tìm một khoá K ∈ 𝒦  {theo thuật toán tìm một khóa}
        𝒦 := {K};
```

2. For <mỗi K $\in \mathcal{K}$> do
For <mỗi $X \to Y \in F$> do
begin
$T := X \cup (K \setminus Y)$;
If <T không chứa phần tử nào của $\mathcal{K}$> then
begin
Tìm một khoá K' nhận T làm siêu khoá ;
$\mathcal{K} := \mathcal{K} \cup \{K'\}$
goto 2; {không cần xét lại (K, $X \to Y$) đã xét}
endIf;
end;
Return;
End;

Ví dụ:

Cho R = <U, F> với U = ABCDEG

$F = \{B \to C, C \to B, A \to GD\}$

Tìm một khóa của R theo thuật toán tìm 1 khóa: ABE

Đặt $\mathcal{K} = \{ABE\}$

☞ Xét K = ABE $\in \mathcal{K}$:

Xét $B \to C \in F \Rightarrow T = B \cup (K \setminus C) = ABE$ chứa phần tử của $\mathcal{K}$

Xét $C \to B \in F \Rightarrow T = C \cup (K \setminus B) = ACE$ không chứa phần tử nào của $\mathcal{K}$

Tìm khoá K' nhận T = ACE làm siêu khoá (theo thuật toán đã biết), ta tìm được khóa là K' = ACE $\Rightarrow \mathcal{K} = \{ABE, ACE\}$

Xét $A \to GD \in F \Rightarrow T = ABE$ chứa phần tử của $\mathcal{K}$

☞ Xét K = ACE $\in \mathcal{K}$:

Xét $B \to C \in F \Rightarrow T = ABE$ chứa phần tử của $\mathcal{K}$

Xét $C \to B \in F \Rightarrow T = ACE$ chứa phần tử của $\mathcal{K}$

Xét $A \to GD \in F \Rightarrow T = ACE$ chứa phần tử của $\mathcal{K}$

Vậy $\mathcal{K} = \{ABE, ACE\}$

5.5. Lý thuyết phân tách

Từ một CSDL lớn, có thể làm nảy sinh sự dư thừa dữ liệu và những dị thường trong cập nhập dữ liệu. Chính vì vậy cần phân tách lược đồ này thành các lược đồ con. Trong lí thuyết phân tách, yêu cầu việc phân tách này phải đảm bảo được tính chất "bảo toàn thông tin".

Ngoài ra, phân tích này sẽ tạo điều kiện thuận lợi cho người lập trình nếu nó còn bảo đảm được tính chất "bảo toàn phụ thuộc hàm".

Định nghĩa: (Phân tách / Phép tách)

Cho lược đồ quan hệ R = <U, F>, các lược đồ con R_1 = <U_1, F_1>, R_2 = <U_2, F_2>, …, R_n = <U_n, F_n>. ρ được gọi là một phân tách của R thành các lược đồ con R_1, R_2, …, R_n, (ký hiệu $\left[\begin{array}{l}\rho = (R_1, R_2, ..., R_n) \\ \rho = (U_1, U_2, ..., U_n)\end{array}\right)$) nếu $\bigcup_{i=1}^{n} U_i = U$.

Ví dụ:

Cho R = <U, F> với U = ABCD

Suy ra: ρ = (AB, BC, ACD) là 1 phân tách của R.

Nhận xét:

Cho r là một quan hệ trên R và $\rho = (U_1, U_2, \ldots, U_n)$ là một phân tách của R. Khi đó, phân tách ρ cho ta các quan hệ con tương ứng r_1, r_2, …, r_n được xác định như sau:

$$r_i = \prod_{U_i}(r) \qquad (i = \overline{1,n})$$

5.5.1. Phân tách bảo toàn thông tin

(Phép tách có kết nối không mất thông tin)

Định nghĩa: (Phân tách bảo toàn thông tin trên một lược đồ quan hệ)

Cho lược đồ quan hệ R = <U, F>. Khi đó, phép tách $\rho = (U_1, U_2, \ldots, U_n)$ được gọi là bảo toàn thông tin trên R nếu:

$$\prod_{U_1}(r) \bowtie \prod_{U_2}(r) \bowtie \ldots \bowtie \prod_{U_n}(r) = r, \quad \forall r \in R$$

Định lý 1: (Điều kiện đủ để một phân tách thành hai lược đồ con là bảo toàn thông tin)

Cho R = <U, F> và X, Y $\subseteq$ U.

Khi đó, nếu $X \rightarrow Y \in F^+$ sao cho $X \cap Y = \varnothing$ và $Z = (U \setminus XY) \neq \varnothing$ thì: ρ = (XY, XZ) là bảo toàn thông tin (BTTT).

Ví dụ: Cho NKBH = <U, F>, với F = {STT→U, MH→TH, MH→ĐG}

ρ = (HG, NK) với HG (MH, TH, ĐG); NK (STT, TK, NG, MH, SL)

Từ F suy ra MH → {TH, ĐG} $\in F^+$

Hay: $\rho = (U_1, U_2)$ là BTTT.

Định lý 2: (Điều kiện cần và đủ để một phân tách thành hai lược đồ con là bảo toàn thông tin)

Cho R = <U, F>. Khi đó, phân tách $\rho = (U_1, U_2)$ trên R bảo toàn thông tin nếu và chỉ nếu: $U_1 \cap U_2 \rightarrow U_1 \setminus U_2 \in F^+$ hoặc $U_1 \cap U_2 \rightarrow U_2 \setminus U_1 \in F^+$

Ví dụ: Xét ví dụ NKBH ở trên

$(U_1 \cap U_2)^+ = (MH)^+ = \{MH, TH, ĐG\}$

$(U_1 \setminus U_2) = \{TH, ĐG\}$
Vậy $(U_1 \setminus U_2) \subseteq (U_1 \cap U_2)^+$
Suy ra: $U_1 \cap U_2 \rightarrow U_1 \setminus U_2 \in F^+$
Hay: $\rho = (U_1, U_2)$ là bảo toàn thông tin.

Thuật toán kiểm tra tính chất bảo toàn thông tin của một phân tách:

Vào: R = <U, F> với U = $\{A_1, A_2, \ldots, A_n\}$ và $\rho = (U_1, U_2, \ldots, U_k)$

Ra: Yes/No

Phương pháp:

Bước 1 (Lập bảng):

Thành lập 1 bảng gồm k dòng và n cột. Các cột được ký hiệu bởi các thuộc tính A_i ($i = \overline{1,n}$), các dòng được kí hiệu bởi các tập thuộc tính U_j ($j=\overline{1,k}$), các phần tử của bảng sẽ được ghi là a_i hoặc b_{ij} theo nguyên tắc sau:

- phần tử dòng i cột j là a_j nếu $A_j \in U_i$
- phần tử cột dòng i cột j là b_{ij} nếu ngược lại

Bước 2 (Biến đổi bảng):

Repeat

For mỗi $X \rightarrow Y \in F$ do

Thực hiện việc bằng trên Y với những dòng có chung X (ưu tiên biến đổi b_{ij} thành a_j)

Until bảng không thay đổi đối với vòng for

Bước 3: (Kết luận)

☞ Nếu tồn tại một dòng (bộ) hoàn toàn các giá trị a_i ($i = \overline{1,n}$) $\Rightarrow \rho$ bảo toàn thông tin

☞ Ngược lại $\Rightarrow \rho$ không bảo toàn thông tin.

Ví dụ: Cho R = <U, F>, với U = ABCDE

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

Và ρ = (AD, AB, BE, CDE, AE)

Bước 1: (Lập bảng)

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{23}	b_{24}	b_{25}
BE	b_{31}	a_2	b_{33}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{53}	b_{54}	a_5

Bước 2: (Biến đổi bảng)

Xét A → C

	A	B	C	D	E
AD	**a_1**	b_{12}	**b_{13}**	a_4	b_{15}
AB	**a_1**	a_2	**b_{13}**	b_{24}	b_{25}
BE	b_{31}	a_2	b_{33}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	**a_1**	b_{52}	**b_{13}**	b_{54}	a_5

Xét B → C:

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	**a_2**	**b_{13}**	b_{24}	b_{25}
BE	b_{31}	**a_2**	**b_{13}**	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{13}	b_{54}	a_5

Xét C → D

	A	B	C	D	E
AD	a_1	b_{12}	**b_{13}**	**a_4**	b_{15}
AB	a_1	a_2	**b_{13}**	**a_4**	b_{25}
BE	b_{31}	a_2	**b_{13}**	**a_4**	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	**b_{13}**	**a_4**	a_5

Xét DE→ C:

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	a_4	b_{25}
BE	b_{31}	a_2	**a_3**	**a_4**	**a_5**
CDE	b_{41}	b_{42}	**a_3**	**a_4**	**a_5**
AE	a_1	b_{52}	**a_3**	**a_4**	**a_5**

DE → A

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	a_4	b_{25}
BE	**a_1**	**a_2**	**a_3**	**a_4**	**a_5**
CDE	**a_1**	b_{42}	a_3	**a_4**	**a_5**
AE	**a_1**	b_{52}	a_3	**a_4**	**a_5**

Từ bảng này ta thấy dòng 3 có bộ (a_1, a_2, a_3, a_4, a_5). Vậy phân tách ρ là bảo toàn thông tin.

Lưu ý: Trong quá trình thực hiện bước 2 (chưa xong) nếu đã tồn tại một dòng toàn là các giá trị a_i thì ta cũng đã có thể kết luận ngay phân tách ρ là bảo toàn thông tin.

5.5.2. Phân tách bảo toàn phụ thuộc hàm

Định nghĩa: (Phân tách bảo toàn phụ thuộc hàm)

Cho R = <U, F>, $\rho = (R_1,R_2,\ldots,R_k)$ là phân tách trên R với $R_i = <U_i, F_i>$ ($i = \overline{1,k}$). Khi đó, ρ được gọi là phân tách bảo toàn phụ thuộc hàm nếu $\bigcup_{i=1}^{k} F_i$ tương đương F (hay $(\bigcup_{i=1}^{k} F_i)^+ = F^+$).

Ví dụ: Cho R = <U, F>, trong đó: U = ABCD, F = {A → BCD, C → D},
và ρ = (R_1, R_2), với R_1 = <CD, {C → D}>, R_2 = <ABC, {A → BC}>.

Do G = {C → D, A → BC} tương đương F nên ρ là phép phân tách bảo toàn phụ thuộc hàm.

Nếu cho trước R = <U, F>, $\rho = (U_1, U_2, \ldots, U_k)$ thì: việc xác định các F_i ($i = \overline{1,k}$) được thực hiện dựa vào định nghĩa sau:

Định nghĩa: (Chiếu của tập phụ thuộc hàm trên tập thuộc tính)

Cho R = <U, F> và $U_i \subseteq U$. Khi đó, chiếu của F trên tập thuộc tính U_i, ký hiệu là $\Pi_{Ui}(F)$, được xác định như sau:

$$\Pi_{Ui}(F) = \{X \rightarrow A \in F^+ \mid X, A \subseteq U_i \text{ và } A \notin X\}$$

Ví dụ:

Cho R = <U,F>, với U = ABCD, F = {A → D, D → B, A → C}
Xét ρ = (ABC, AD) (U_1 = ABC, U_2=AD)
⇒ F_1 = (A→C, A→B) (chiếu của F lên tập U_1)
và F_2 = {A→D} (chiếu của F lên tập U_2)

Nhận xét: $\Pi_{Ui}(F)$ là bao gồm các phụ thuộc hàm tầm thường và không tầm thường (phụ thuộc hàm tầm thường là phụ thuộc hàm mà vế phải là con của vế trái). Ở đây chúng ta dùng (A ∉ X) là để loại bỏ các phụ thuộc hàm tầm thường. Theo đó, việc xác định các F_i ($i = \overline{1,k}$) là dựa vào việc xác định phủ tối thiểu của $\Pi_{Ui}(F)$.

Thuật toán: Tìm $\Pi_X(F)$

Bước 1:

Tính các X'^+, ∀X' ⊂ X và X' ≠ ∅

Bước 2:

Xác định các phụ thuộc hàm của $\Pi_X(F)$, dựa vào thuật toán sau:

Begin

F' = ∅

For mỗi $X^{'+} = A_1A_2...A_k$ do

For i = 1 to k do

If ($A_i \in X^{'+}$) and ($A_i \notin X^{'}$) then

$F' = F' \cup \{X' \rightarrow A_i\}$

$\Pi_X(F) = F'$;

End;

Ví dụ:

Cho R = <ABCD, {A→D, D→B, A→C}>, và X = ABC

$A_F^+ = ABCD$ $(AB)_F^+ = ABCD$

$B_F^+ = B$ $(AC)_F^+ = ABCD$

$C_F^+ = C$ $(BC)_F^+ = BC$.

$F_1^{'} = \{A \rightarrow B, A \rightarrow C, AB \rightarrow C, AC \rightarrow B\}$

Nhận xét: Các phụ thuộc hàm tầm thường sẽ bị loại bỏ khi xét một phủ tối thiểu với F_i.

Ví dụ: Xét ví dụ trên:

Ta có 1 phủ tối thiểu của $F_1^{'}$ là:

$F_1 = \{A \rightarrow C, A \rightarrow B\}$

Để kiểm tra một phân tách có bảo toàn phụ thuộc hàm hay không, theo phương pháp nêu trên ta phải thực hiện việc xác định các F_i ($i = \overline{1,k}$). Từ đó suy ra $G = \bigcup_{i=1}^{k} F_i$ và việc kiểm tra F tương đương G dựa vào tính chất tương đương sau:

$$F \text{ tương đương } G \Leftrightarrow \begin{cases} G \subseteq F^+ & (1) \\ F \subseteq G^+ & (2) \end{cases}$$

Do (1) là hiển nhiên nên ta chỉ cần kiểm tra (2) dựa vào thuật toán sau:

Thuật toán: Kiểm tra $F \subseteq G^+$

Vào: F, G

Ra: Yes/No

Phương pháp:

Procedure KiemTra;

Begin

For mỗi $X \rightarrow Y \in F$ do

If $X_G^+ \not\supseteq Y$ then

```
                Begin
                    Write('ρ không bảo toàn phụ thuộc hàm');
                    Exit;
                End;
        Write('ρ bảo toàn phụ thuộc hàm');
End;
```

Thuật toán: Tính X_G^+ mà không cần phải xác định G

Để có thể tính được X_G^+ mà không cần phải xác định G (không cần xác định các F_i) ta có thể dựa vào thuật toán sau:

Vào: R = <U, F>
$\rho = (R_1, R_2, \ldots, R_n)$
$X \subseteq U$

Ra: X_G^+

Phương pháp:

```
Procedure Tính X_G^+;
Begin
    Z := X;
    Repeat
        For  i:= 1 to n do
            Z := Z ∪ ((Z ∩ U_i)^+_F ∩ U_i)
    Until <Z không thay đổi>
    X_G^+ = Z;
End;
```

Ví dụ:

Cho R = <U, F>, U = ABCD, F = $\{A \to B, B \to C, C \to D, D \to A\}$, và ρ = (AB, BC, CD) có bảo toàn phụ thuộc hàm

- Xét $D \to A \in F$, tính D_G^+ như sau:
  1. Z := D
  2. • i := 1; U_1:= AB
     $\Rightarrow$ Z := Z ∪ $((Z \cap U_i)_F^+ \cap U_i) = D \cup ((D \cap AB)_F^+ \cap AB) = D$
     • i := 2; U_2 = BC
     $\Rightarrow$ Z:= D ∪ $((D \cap BC)_F^+ \cap BC) = D$
     • i := 3; U_3 = CD
     $\Rightarrow$ Z:= D ∪ $((D \cap CD)_F^+ \cap CD) = CD$: Z thay đổi $\Rightarrow$ quay lại
     • i := 1 $\Rightarrow$ Z = CD
     • i := 2 $\Rightarrow$ Z = BCD
     • i := 3 $\Rightarrow$ Z = BCD
     • i := 1 $\Rightarrow$ Z = ABCD
     • i := 2 $\Rightarrow$ Z = ABCD

• i := 3 $\Rightarrow$ Z = ABCD

$\Rightarrow$ D_G^+ = ABCD $\Rightarrow$ D $\rightarrow$ A $\in$ G^+

- Xét A $\rightarrow$ B $\in$ F, tính A^+_G = ABCD $\supseteq$ B
- Xét B $\rightarrow$ C $\in$ F, tính B^+_G = ABCD $\supseteq$ C
- Xét C $\rightarrow$ D $\in$ F, tính C^+_G = ABCD $\supseteq$ D

$\Rightarrow$ Kết luận: ρ bảo toàn phụ thuộc hàm

Lưu ý:

Một phân tách ρ bảo toàn thông tin nhưng có thể không bảo toàn phụ thuộc hàm.

Ví dụ:

R = <U, F> , với U =CSZ, F = {CS $\rightarrow$ Z, Z $\rightarrow$ C}

Xét ρ = (ZC, SZ), ρ bảo toàn thông tin (bài tập) nhưng ρ không bảo toàn phụ thuộc hàm.

Một phân tách bảo toàn phụ thuộc hàm nhưng cũng có thể không bảo toàn thông tin.

Ví dụ:

Cho R = <U, F>, với U = ABCD, F = {A $\rightarrow$ B, C $\rightarrow$ D}

Xét ρ = (AB, CD), ρ là bảo toàn phụ thuộc hàm vì:

F_1 = {A $\rightarrow$ B }

F_2 = {C $\rightarrow$ D }

Nhưng rõ ràng ρ là không bảo toàn thông tin.

Nhận xét: Một phân tách bảo toàn thông tin nhưng không bảo toàn phụ thuộc hàm thì vẫn có thể sử dụng được. Tuy nhiên điều này sẽ gây phiền phức đối với người lập trình, bởi lẽ việc cập nhật dữ liệu trên một quan hệ $r_i \in R_i$ ($i = \overline{1,k}$) sẽ có thể làm cho quan hệ gốc r không thỏa F. Do đó việc cập nhật thường được thực hiện theo quy trình sau:

- ☞ *Bước 1:* Cập nhật dữ liệu trên r_i
- ☞ *Bước 2: Tính r*
- ☞ *Bước 3:* Nếu r thỏa F thì việc cập nhật được chấp nhận, còn nếu không việc cập nhật sẽ bị huỷ bỏ.

5.6. Lý thuyết chuẩn hoá

Một lược đồ quan hệ quan hệ thiết kế không tốt sẽ gây ra những dị thường dữ liệu như dư thừa dữ liệu và do việc cập nhật dữ liệu. Để tránh dị thường dữ liệu, lược đồ quan hệ cần thiết phải biến đổi thành các dạng phù hợp. Quá trình đó được xem là quá trình chuẩn hóa lược đồ quan hệ.

Lý thuyết chuẩn hoá sẽ xác định các dạng chuẩn (Norm Form) của một lược đồ quan hệ. Đồng thời cho phép xây dựng các thuật toán để phân tách

một lược đồ thành các lược đồ con sao cho các lược đồ con đều thuộc một dạng chuẩn nào đó.

5.6.1. Dạng chuẩn 1 (1NF)

Định nghĩa: (Dạng chuẩn 1)

Một lược đồ R được gọi là thuộc **dạng chuẩn 1** (ký hiệu: $R \in 1NF$) nếu miền giá trị của các thuộc tính trong R chỉ chứa những giá trị nguyên tố (không thể phân chia được), hay các thuộc tính này đều là đơn và đơn trị.

Ngược với thuộc tính đơn là thuộc tính phức hợp, là thuộc tính được xây dựng từ nhiều thuộc tính khác.

Ngược với thuộc tính đơn trị là thuộc tính đa trị, là thuộc tính mà giá trị của nó là một tập hợp.

Quy ước: Tất cả các lược đồ quan hệ được xét đến sau này thuộc 1NF.

5.6.2. Dạng chuẩn 2 (2NF)

Định nghĩa: (Thuộc tính khoá)

Cho R= <U, F>. Khi đó, thuộc tính A được gọi là **thuộc tính khoá** nếu A thuộc một khoá nào đó của R. Ngược lại, A được gọi là **thuộc tính không khoá.**

Ví dụ:

1) R = <U, F>, với U = ABCD, $F = \{AB \rightarrow C, C \rightarrow D\} \Rightarrow$ tập các khóa K của R là K = {AB}. Suy ra: R có hai thuộc tính khoá: A và B, và R có hai thuộc tính không khoá: C và D.

2) R = <U, F>, với U = ABCD, $F = \{AB \rightarrow C, B \rightarrow D, BC \rightarrow A\}$
Suy ra: tập các khóa K của R là $K = \{AB, BC\} \Rightarrow$ R có ba thuộc tính khoá: A, B và C, còn D là thuộc tính không khoá.

Định nghĩa: (Phụ thuộc hàm đầy đủ)

Cho R = <U, F>. Khi đó, $X \rightarrow Y \in F$ được gọi là một **phụ thuộc hàm đầy đủ** (đọc là: Y phụ thuộc hàm đầy đủ vào X) nếu:

$$\nexists Z \subset X \text{ sao cho } Z \rightarrow Y \in F^+$$

Ví dụ:

NKBH = <U, F>

với U = {NG, SP, MH, ĐG, SL}, F = {{SP, MH}→U, MH→ĐG}

Suy ra: $\{SP, MH\} \rightarrow ĐG \in F^+$ không là phụ thuộc hàm đầy đủ (vì $\exists \{MH\} \subset \{MH, SP\}: MH \rightarrow ĐG \in F^+$)

Nhận xét:

$X \rightarrow U \in F^+$ là một phụ thuộc hàm đầy đủ khi X là khoá của R.

Định nghĩa: (2NF)

Cho R = <U, F>, R được gọi là thuộc **dạng chuẩn 2** (ký hiệu R ∈ 2NF) nếu với mọi thuộc tính không khoá của R là phụ thuộc hàm đầy đủ vào mọi khoá của R.
Như vậy:

- R ∈ 2NF nếu và chỉ nếu ∀A là thuộc tính không khoá, ∀ X ∈ thuộc tập các khoá của R thì $X \rightarrow A \in F^+$ là phụ thuộc hàm đầy đủ.
- R ∉ 2NF nếu và chỉ nếu ∃ A là thuộc tính không khoá và ∃X ∈ thuộc tập các khóa của R sao cho $X \rightarrow A \in F^+$ không là phụ thuộc hàm đầy đủ.

Các ví dụ:
Ví dụ 1: Xét ví dụ trên, lược đồ NKBH có tập các khoá là K = {SP, MH}, do đó hai thuộc tính khoá là SP và MH. Suy ra các thuộc tính không khoá: NG, ĐG, SL.

Theo trên ta có : $\{SP, MH\} \rightarrow ĐG \in F^+$ không là phụ thuộc hàm đầy đủ vì MH→ĐG. Suy ra: NKBH ∉ 2NF.

Ví dụ 2: Cho R = <SAIP, {SI → P, S → A}. Lúc đó R ∉ 2NF.
Thật vậy, R chỉ có khóa SI nên A là thuộc tính không khóa, mặt khác A không phụ thuộc hàm đầy đủ vào khóa SI vì S → A.

Ví dụ 3: Cho R = <SAIP, {SI → P}>. R ∈ 2NF.
Thật vậy, R chỉ có khóa SI nên A, P là các thuộc tính không khóa, mặt khác A và P phụ thuộc hàm đầy đủ vào khóa SI vì không có tập con nào của SI xác định hàm A và P.

Thuật toán: Kiểm tra R thuộc 2NF
Vào: R = <U, F>
Ra: Yes/No
Phương pháp:

```
Procedure   KiemTra2NF (R);
Begin
      For mỗi A không phải là thuộc tính khoá do
            For mỗi X ∈ K do          {K là tập các khóa của R}
                  If  <X → A không là phụ thuộc hàm đầy đủ> Then
                        Return No;
      Return Yes;
```

End;

Nhận xét:

1. Nếu mọi khoá của lược đồ quan hệ R chỉ có một thuộc tính thì R $\in$2NF. Hay ($\forall$ X $\in$ K: |X| = 1) $\Rightarrow$ R $\in$ 2NF.

2. Lược đồ quan hệ R$\in$2NF $\Leftrightarrow$ $\forall$ X$\rightarrow$A $\in$ F^+, với A $\notin$ X và X $\subset$ K (K là khoá của R) thì A là thuộc tính khoá. (bài tập)

5.6.3. Dạng chuẩn 3 (3NF)

Ví dụ:

Cho NKBH = <U, F>, với U = {STT, NGAY, MH, TH, ĐG, SL},

F = {STT$\rightarrow$U, MH$\rightarrow$TH, MH$\rightarrow$ĐG}

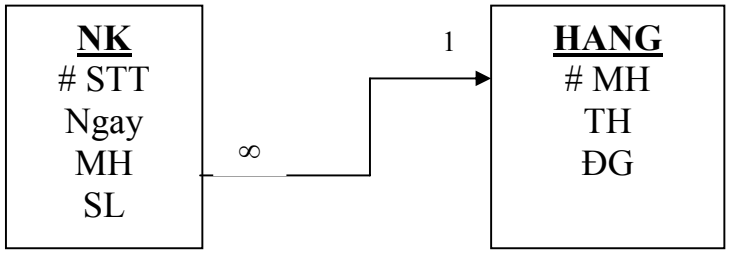
Suy ra: NKBH $\in$ 2NF (do lược đồ có một khoá duy nhất là STT chỉ có một thuộc tính) nhưng vẫn tồn tại dư thừa dữ liệu trong lược đồ này.

STT	Ngay	MH	TH	ĐG	SL
1	01/01/01	A1	Tiêu	50	10
2	02/01/01	A2	Cafe	20	20
3	03/01/01	A1	Tiêu	50	30
4	04/01/01	A2	Cafe	20	40

Suy ra: Phân tách NKBH thành 2 lược đồ con:

HANG = <U_1, F_1>, với U_1 = {MH, TH, ĐG} và F_1 = {MH $\rightarrow$ U_1}

NK = <U_2, F_2>, với U_2 = {STT, NGAY, MH, SL} và F_2 = {STT $\rightarrow$ U_2}



Ta sẽ chứng minh rằng:

- ☞ NKBH ∉ 3NF
- ☞ HANG, NK ∈ 3NF

Định nghĩa: (R ∈ 3NF)

Cho R = <U, F>. Khi đó R được gọi là thuộc **dạng chuẩn 3** (ký hiệu R ∈ 3NF) nếu ∀ X→A ∈ F^+ với A ∉ X thì:

- ☞ hoặc X là siêu khoá
- ☞ hoặc A là thuộc tính khoá

Từ định nghĩa suy ra:

R ∉ 3NF ⇔ ∃ X → A ∈ F^+ với A ∉ X sao cho thỏa hai điều kiện sau:

1. X không là siêu khoá
2. A là thuộc tính không khoá

Ví dụ 1:

Xét ví dụ trên, NKBH = <U, F>, với

U = {STT, NGAY, MH, TH, ĐG, SL},

F = {STT→U, MH→TH, MH→ĐG}

NKBH chỉ có một khóa STT và dễ dàng thấy NKBH ∉ 3NF bởi vì tồn tại phụ thuộc hàm MH → TH ∈ F^+ mà MH không là siêu khoá, TH cũng không phải là thuộc tính khoá.

Xem hai lược đồ con của NKBH là HANG và NK:

HANG = <U_1, F_1>, với U_1 = {MH, TH, ĐG} và F_1 = {MH → U_1}

NK = <U_2, F_2>, với U_2 = {STT, NGAY, MH, SL} và F_2 = {STT → U_2}

Rõ ràng HANG có duy nhất khóa là MH và NK có duy nhất khóa là STT, từ đó dễ thấy HANG và NK ∈ 3NF.

Ví dụ 2:

Cho R = <CSZ, {CS → Z, Z → C}>. Lược đồ này có hai khóa là CS và SZ. Vậy R không có thuộc tính không khóa nên R ∈ 3NF.

Ví dụ 3:

Cho R = <SIDM, {SI → D, SD → M}>. Lược đồ này chỉ có một khóa là SI, D, M là các thuộc tính không khóa. R ∉ 3NF vì ∃ SD → M.

Chú ý: Nếu R không chứa thuộc tính không khóa thì R ∈ 3NF.

Thuật toán: Kiểm tra 3NF

Vào: R = <U, F>

Ra: Yes/No

Phương pháp:

```
Procedure   KiemTra3NF(R);
Begin
      For mỗi X không là siêu khoá của R do
            If <X+ ⊇ A là thuộc tính không khoá và A ∉ X> then
                  Return No; {R không thuộc 3NF}
      Return Yes; {R thuộc 3NF}
End;
```

Nhận xét:

Nếu R ∈ 3NF thì R ∈ 2NF

Nếu R ∈ 2NF thì R ∈ 1NF

Ta có thể chứng minh rằng: R ∉ 3NF ⇔ ∃ X → Y ∈ F^+ và Y → A ∈ F^+ với:

- ☞ X là khoá
- ☞ Y không là siêu khoá
- ☞ A là thuộc tính không khoá và A ∉ Y

Hay:

R ∉ 3NF ⇔∃ X là khoá $\not\rightrightarrows$ Y, Y→A và A không là thuộc tính khoá, A ∉ Y

Ví dụ:

Xét ví dụ trên: NKBH = <U, F>, với

U = {STT, NGAY, MH, TH, ĐG, SL},

F = {STT→U, MH→TH, MH→ĐG}

NKBH ∉ 3NF, vì: ∃{STT}_khoá $\not\rightrightarrows$ MH → TH_không khoá.

Thuật toán 1: Phân tách thành các lược đồ con 3NF bảo toàn thông tin

Vào: R = <U, F>

Ra: $\rho = (R_1, R_2, \ldots, R_k)$ với $R_i \in$ 3NF ($i = \overline{1,k}$) và ρ là bảo toàn thông tin.

Phương pháp:

Bước 1: Kiểm tra R $\in$ 3NF

- ☞ Nếu R $\in$ 3NF: không phân tách và dừng
- ☞ Nếu R $\notin$ 3NF thì $\exists$ X_khoá $\not\rightrightarrows$ Y, mà Y $\rightarrow$ A_không khoá, A $\notin$ Y. Phân tách R thành 2 lược đồ con: ρ = (YA, U\A)

Bước 2: Kiểm tra các lược đồ con

Kiểm tra lần lượt các lượt các lược đồ con có thuộc 3NF không, nếu không thuộc thì lại phân tách tiếp (theo quy tắc chỉ ra trong bước 1) cho đến khi nào tất cả các lược đồ con đều thuộc dạng chuẩn 3NF. Bấy giờ chúng ta sẽ có một cây phân tách (cây nhị phân) mà các nút lá là các lược đồ con thuộc chuẩn 3NF. Đó là kết quả cần tìm.

```
Procedure    PhanTach(U, F);
Begin
        If (∃ X → A, X_không siêu khoá, A_không khoá, A ∉ X) then
                begin
                        PhanTach(XA, F1);
                        PhanTach(U\A, F2);
                end;
End;
```

Lưu ý:

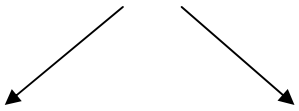
Nếu phân tách $\rho = (U_1, \ldots, U_n)$ tồn tại U_i là tập con của U_j ($i \neq j$) thì ta loại bỏ lược đồ tương ứng với tập thuộc tính U_i.

Ví dụ:

Xét ví dụ trên: $\exists$ {STT} $\rightarrow$ {MH, TH} $\rightarrow$ ĐG

Suy ra: ρ = ({MH, TH, ĐG}, {STT, NGAY, MH, TH, SL})

$\in$ 3NF $\quad$ $\notin$ 3NF, do $\exists$STT $\not\rightrightarrows$ MH$\rightarrow$TH



{MH,TH} $\quad$ {STT,NGAY,MH,SL}

$\in$ 3NF $\quad$ $\in$ 3NF

Vậy: ρ = ({MH,TH, ĐG},{STT,NGAY,MH,SL})

Lưu ý:

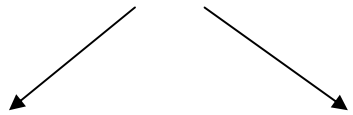
Thuật toán trên không duy nhất

Ví dụ: Xét ví dụ trên

$\exists$ {STT} $\not\rightarrow$ {MH} $\rightarrow$ TH.

Suy ra: ρ = ({MH, TH} , {STT, NGAY, MH, ĐG, SL})

$\in$ 3NF $\notin$ 3NF, do: $\exists$ STT $\not\rightarrow$ MH$\rightarrow$DG



{MH, ĐG} $\in$ 3NF

{STT,NGAY,MH,SL} $\in$ 3NF

Vậy: ρ = ({MH, TH}, {MH, ĐG}, {STT, NGAY, MH, SL})

Lưu ý:

Ta có thể gộp các lược đồ có dạng như sau để tạo thành một lược đồ mà vẫn bảo toàn thông tin và phụ thuộc hàm.

$$\left.\begin{array}{l} R_1 =< AA_1, \{A \rightarrow A_1\} > \\ R_2 =< AA_2, \{A \rightarrow A_2\} > \\ \ldots \\ R_k =< AA_k, \{A \rightarrow A_k\} > \end{array}\right\} \Leftrightarrow R =< AA_1A_2\ldots A_k, \{A \rightarrow A_1A_2\ldots A_k\} > \in 3NF$$

Thuật toán 2: Phân tách thành các lược đồ con 3NF bảo toàn thông tin và bảo toàn phụ thuộc hàm

Vào: R = <U, F>

Ra: ρ = (R_0, R_1, R_2, …, R_k) với $R_i \in$ 3NF (i = $\overline{0,k}$), ρ là bảo toàn thông tin và bảo toàn phụ thuộc hàm.

Phương pháp:

{Phân tách thành các lược đồ con 3NF bảo toàn thông tin và bảo toàn phụ thuộc hàm}

Bước 1: Xác định phủ tối thiểu của F:
F' = $\{X_i \rightarrow A_i \mid i = \overline{1,m}\}$

Bước 2: Tìm một khoá X bất kì của R.

Bước 3: Xác định các lược đồ con R_0 = <U_0, F_0> với U_0 = X

R_i = <U_i, F_i> với $U_i = X_iA_i$ (i = $\overline{1,m}$)

Lưu ý

☞ Nếu $\exists i \neq j$ mà $U_i \subseteq U_j$ (i, j = $\overline{0,m}$) thì loại bỏ R_i. Quá trình này sẽ tiếp tục cho đến khi không thể loại bỏ được một R_i nào nữa.

☞ Chúng ta có thể gộp các lược đồ như đã bàn đến.

Ví dụ

Cho R = <U, F>, với U = ABCD, F = {A→B, B→C, CD→A, AC→D}

Bước 1: Ta có một phủ tối thiểu của F là:

F = {A→B, B→C, CD→A, A→D}

Bước 2: Ta có A là một khoá của R.

Bước 3: $R_0 = <U_0, F_0>$ với $U_0 = A$, $F_0 = \varnothing$

$R_1 = <AB, \{A \rightarrow B\}>$

$R_2 = <BC, \{B \rightarrow C\}>$

$R_3 = <ACD, \{CD \rightarrow A, A \rightarrow D, A \rightarrow C\}>$

$R_4 = <AD, \{A \rightarrow D\}>$

Loại R_0 và R_4

Kết luận: $\rho = (AB, BC, ACD)$

Lưu ý:

Thuật toán không duy nhất, bởi vì:

☞ Do cách tìm một khoá ban đầu có thể khác nhau.

☞ Do cách xác định phủ tối thiểu là không duy nhất.

5.6.4. Dạng chuẩn BCNF (Boyce - Codd Normal Form)

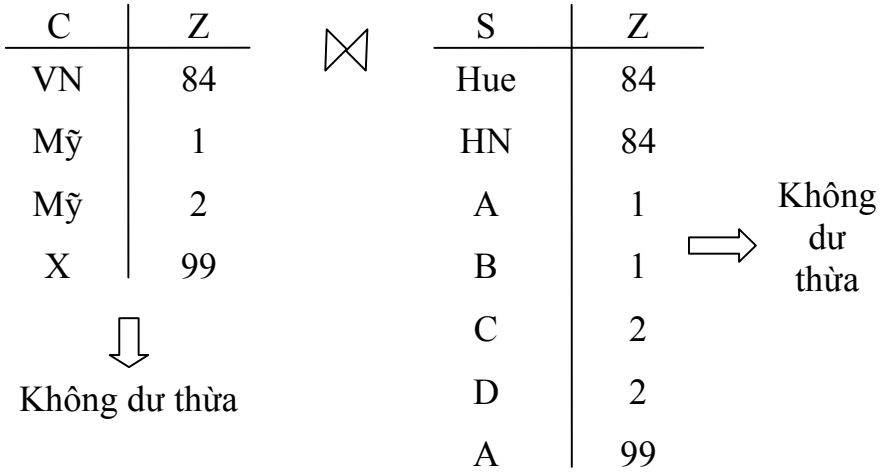
Ví dụ

Cho R = <U, F>, với U = CSZ và F = {CS→Z, Z→C}. Như đã thấy ở trên, R ∈ 3NF, tuy nhiên lược đồ này vẫn còn dư thừa dữ liệu, chẳng hạn xét quan hệ r ∈ R như sau:

r =

C	S	Z
VN	Hue	84
VN	HN	84
Mỹ	A	1
Mỹ	B	1
Mỹ	C	2
Mỹ	D	2
X	A	99

Tách thành lược đồ con:



Ta thấy R ∈ 3NF (vì R có 2 khoá: CS và SZ), nhưng R vẫn dư thừa. Ta sẽ chứng minh R ∉ BCNF.

Định nghĩa: (Dạng chuẩn BCNF)

Cho R = <U, F>. Khi đó R thuộc dạng chuẩn BCNF (ký hiệu R∈BCNF) nếu với mọi $X \rightarrow A \in F^+$ với $A \notin X$ thì X là siêu khoá của R.

Nhận xét:

- ☞ $R \in BCNF \Rightarrow R \in 3NF$
- ☞ $R \notin BCNF \Leftrightarrow \exists X \rightarrow A \in F^+$ với $A \notin X$ và X không là siêu khoá.

Ví dụ:

Cho R = <U, F>, với U = CSZ và F = {CS→Z, Z→C}, R ∉ BCNF vì $\exists Z \rightarrow C \in F^+$ mà vế trái (Z) không phải là siêu khoá.

Thuật toán: Kiểm tra một lược đồ quan hệ có thuộc BCNF

Vào: R = <U, F>

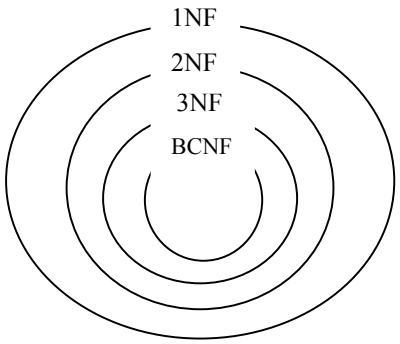
Ra: Yes/No

Phương pháp:

```
Procedure   KiemTraBCNF(U, F);
Begin
      For <mỗi X không là siêu khoá> do
            If X+F ≠ X then
                  Return No;
      Return Yes;
End;
```

Nhận xét:

- ☞ Tất cả các dạng chuẩn: 1NF, 2NF, 3NF đều có thể dư thừa dữ liệu.
- ☞ R ∉ BCNF có nghĩa là R vẫn tồn tại dư thừa dữ liệu.
- ☞ Khi R ∈ BCNF, tất cả các phụ thuộc dữ liệu đều là các ràng buộc về khoá.



Thuật toán 1 (Phân tách R thành các lược đồ thỏa BCNF và bảo toàn thông tin)

Phải tìm chiếu của các phụ thuộc hàm trên các lược đồ con.

Vào: R = <U, F>

Ra: $\rho = (R_0, R_1, R_2, \ldots, R_k)$ bảo toàn thông tin và $R_i \in$ BCNF

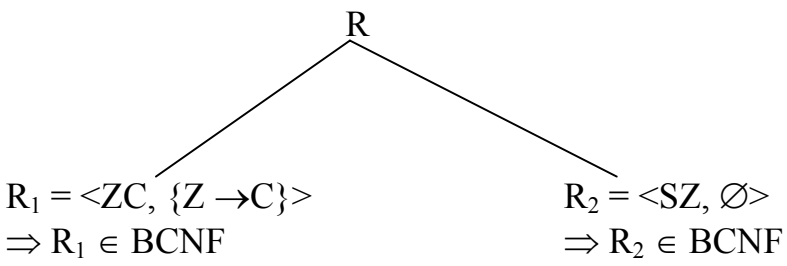
Phương pháp: xây dựng cây phân tách theo các thủ tục sau:

```
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;
```

Ví dụ:

Cho R = <U, F>, với U = CSZ và F = {CS→Z, Z→C}.

R ∉ BCNF vì Z→C ∈ F^+, Z không là siêu khoá (tập khoá: {CS, ZS}). Do vậy ta phân tách R thành 2 lược đồ con như sau:



Nhận xét:

Ưu điểm: việc phân tách các lược đồ thành các lược đồ con BCNF là dừng.

Nhược điểm: có độ phức tạp hàm mũ (đó là số thuộc tính của mỗi lược đồ con).

Bổ đề 1:

Mọi lược đồ có hai thuộc tính đều thuộc dạng chuẩn BCNF. Điều này có nghĩa: $|U| = 2 \Rightarrow R = <U, F> \in$ BCNF.

Chứng minh:

Giả sử R = <AB, F>, xet các trường hợp sau:

$$\left.\begin{array}{l} F = \phi \\ F = \{A \to B\} \\ F = \{B \to A\} \\ F = \{A \to B, B \to A\} \end{array}\right\} \Rightarrow R =< U, F > \in BCNF$$

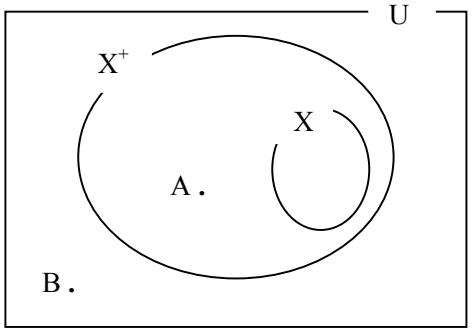
Bổ đề 2:

Nếu $R \notin$ BCNF thì $\exists$ A, B $\in$ U (A $\neq$ B): U\AB $\to$ A $\in F^+$. Tức là: nếu $\nexists$ A, B $\in$ U (A $\neq$ B): U\AB $\to$ A $\in F^+$ thì R $\in$ BCNF.

Chứng minh: bằng phương pháp phản chứng

Giả sử $R \notin$ BCNF $\Rightarrow \exists$ X $\to$ A $\in F^+$: A $\notin$ X và X không là siêu khoá ($X^+_F \neq U$).

Do $X^+ \neq U \Rightarrow \exists$ B $\in$ U: B $\notin X^+$ (A $\neq$ B) $\Rightarrow$ B $\notin$ X $\Rightarrow$ X $\subseteq$ U\AB. $\Rightarrow$ U\AB $\to$ X $\in F^+$. Ngoài ra: X $\to$ A $\in F^+ \Rightarrow$ U\AB $\to$ A $\in F^+$ (mâu thuẫn) $\Rightarrow$ điều phải chứng minh.



Dựa vào hai bổ đề ở trên ta có thuật toán sau đây:

Thuật toán 2: Phân tách R thành các lược đồ thỏa BCNF và bảo toàn thông tin (không cần tính chiếu của F trên các lược đồ con).

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:

```
Procedure    BCNF_PTH(U, F);
Begin
        ρ:= ∅;
        Z:= U;
        Repeat
                PhanTach(Z, Y, A');{nhằm xác định Y và A' để phân tách Z
        thành hai lược đồ con Y ∈ BCNF và Z \ A'};
                ρ := ρ ∪{Y};
                Z := Z \ A' ;
        Until ∄ A, B ∈ Z (A ≠ B): A∈(Z-AB)+;
End;
```

Trong đó:

```
Procedure    PhanTach(Z, Y, A');
Begin
        Y:= Z;
        While ∃ A, B ∈ Y: A ≠ B và A ∈ (Y-AB)+ do
                begin
                        Y:= Y - B;
                        A':= A;
                end;
End;
```

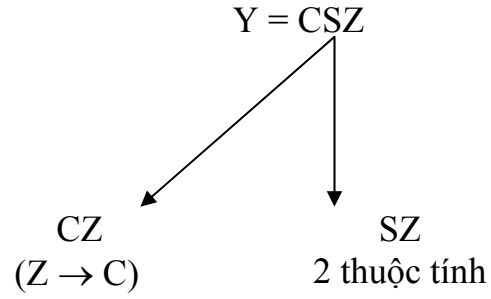
Ví dụ

Cho R = <U, F>, với U = CSZ và F = {CS→Z, Z→C}

Y= CSZ

∃ Z→C ∈ F^+

A	C
B	S
Y	CZ



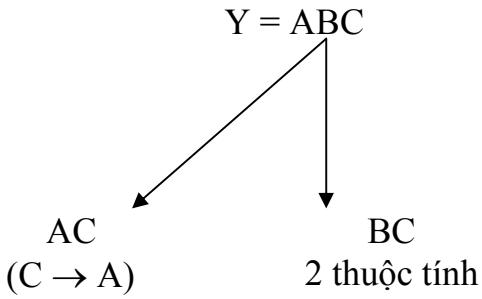
⇒ρ = (CZ, SZ) là phân tách cần tìm.

Nhận xét

Ưu điểm: thuật toán này có độ phức tạp tính toán là đa thức $O(n^2m)$ với n = |U|, m = |F|.

Nhược điểm: Đôi khi lại phân tách một lược đồ quan hệ đã ở dạng chuẩn BCNF.

Ví dụ 1: Cho R = <ABC , F = {C → A, C → B}>



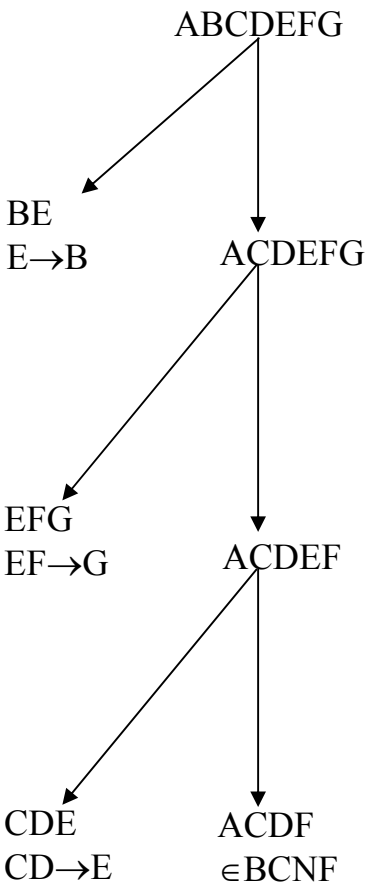
A	A
B	B
Y	AC

$C^+ = CAB$

ρ = (AC, BC) là phân tách cần tìm.

Ví dụ 2:

Cho R = <ABCDEFG, {AB → D, CD → E, E → B, EF → G}>



A	B	B	B	B	B
B	A	C	D	F	G
Y	BCDEFG	BDEFG	BEFG	BEG	BE

$BCDEFG = (CDEFG)^+$

A	G	G	G
B	A	C	D
Y	CDEFG	DEFG	EFG

A	E	E
B	A	F
Y	CDEF	CDE

⇒ ρ = (BD, EFG, CDE, ACDF).

Nhận xét:

- Cả hai thuật toán trên đều có thể cho phân tách ρ không duy nhất.
- Rõ ràng không tồn tại một thuật toán phân tách thành BCNF vừa bảo toàn thông tin, vừa bảo toàn phụ thuộc hàm.

Chương 6. GIỚI THIỆU VỀ SQL

6.1. Định nghĩa dữ liệu trong SQL và các kiểu dữ liệu.

SQL sử dụng tập hợp các bảng, dòng và cột để biểu diễn một mô hình CSDL quan hệ, trong đó các bảng, dòng và cột lần lượt tương ứng với các quan hệ, các bộ và các thuộc tính. Câu lệnh chính của SQL được sử dụng để khai báo một lược đồ cơ sở dữ liệu là CREATE, câu lệnh này cho phép tạo ra các lược đồ, các quan hệ và các domain (như các view, assertion, trigger).

6.1.1. Lược đồ và những khái niệm danh mục trong SQL

Trong version đầu tiên, SQL không có ý tưởng xây dựng cho một lược đồ cơ sở dữ liệu quan hệ. Ý tưởng về xây dựng một lược đồ cơ sở dữ liệu được bắt đầu từ SQL version 2.0, ở đó các bảng được tổ chức kết nhóm và có những quan hệ với nhau để cùng giải quyết một ứng dụng. Một lược đồ trong SQL được xác định thông qua tên lược đồ, quyền hạn truy xuất và các yếu tố bên trong như: các quan hệ, các ràng buộc, các khung nhìn, các domain và cách mô tả lược đồ.

Câu lệnh tạo lược đồ:

CREATE SCHEMA <SchemaName> AUTHORIZATION <UserName>

Ví dụ 6.1: Cần tổ chức một lược đồ CSDL để quản lý sự hoat động của một công ty có tên COMPANY. Biết rằng, công ty này có một tập thể nhân viên của công ty, các bộ phận quản lý con trực thuộc công ty và các đề án mà công ty đang thực hiện. Một số thông tin mô tả cơ bản tình hình hoạt động của công ty COMPANY nhằm phục vụ cho việc thiết kế là:

- Công ty cần quản lý số bộ phận (DEPARTMENT) trực thuộc. Mỗi bộ phận có tên bộ phận, một mã số bộ phận, một người quản lý bộ phận và nơi đặt bộ phận. Một bộ phận có thể đặt tại một vài địa phương khác nhau.

- Một bộ phận có thể quản lý một số đề án (PROJECT). Mỗi đề án có một tên, một số hiệu và nơi đặt đề án.

- Công ty cần quản lý một tập thể nhân viên (EMPLOYEE) trực thuộc công ty. Công ty quản lý nhân viên thông qua tên nhân viên, mã số nhân viên, địa chỉ, lương, giới tính và ngày sinh của nhân viên. Ngoài ra, công ty cần biết quan hệ giữa các thành viên trong công ty để có những chính sách thích hợp với từng thành viên.

Với cách mô tả như trên, chúng ta có thể đưa ra lược đồ CSDL COMPANY để quản lý với dữ liệu của công ty như sau:

EMPLOYEE

Fname	Mvinit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731Fondren,Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss,Houston,TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle,Sping,TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry,Bellaire,TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire,Houston,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice,Houston,TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas,Houston,TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone,Houston,TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

6.1.2. Lệnh tạo bảng trong SQL

Câu lệnh CREATE TABLE được sử dụng để tạo một quan hệ mới với các chỉ định kèm theo gồm: tên quan hệ, các thuộc tính và các ràng buộc kèm theo nếu có. Các thuộc tính cần được xác định trước và mỗi thuộc tính có một tên, kiểu dữ liệu cho thuộc tính và các ràng buộc vùng kèm theo nếu có. Các khóa chính, khóa phụ, khóa ngoại cũng như các ràng buộc cũng có thể được tạo ra trong lệnh CREATE. Lưu ý rằng, khi một quan hệ đã được tạo ra nhưng vẫn cần sửa đổi hoặc bổ sung một số yếu tố cần thiết khác chúng ta có thể sử dụng câu lệnh ALTER TABLE để thực hiện điều chỉnh. Dó đó chúng ta không quá quan tâm đến tính đầy đủ của một bảng được tạo.

Cú pháp chung:

CREATE TABLE [<Schema>].<TableName>
(FieldName1 Type1[(size)] [NOT NULL], ...
FieldNameN TypeN[(size)] [NOT NULL],
PRIMARY KEY (FieldName),
FOREIGN KEY (FieldName) REFERENCES TableName(Field));

Giải thích các thành phần:

- TableName: Tên quan hệ.

- FieldName: Tên các thuộc tính; cần khai báo kiểu, kích thước, qui định về nhận giá trị NULL/NOT NULL (mặc định cho nhận giá trị NULL).

- PRIMARY KEY: Chỉ định khóa của quan hệ. Nếu quan hệ có nhiều khóa thì các khóa được liệt kê cách nhau một dấu phẩy.

- FOREIGN KEY: Chỉ định khóa ngoại

Với lệnh Creat Table có thể tạo dựng một lược đồ quan hệ với các quan hệ, các khóa chỉ định, các ràng buộc theo yêu cầu. Sau đây là một ví dụ minh họa.

Tạo lược đồ COMPANY với tên người dùng Jsmith

CREATE SCHEMA COMPANY AUTHORIZATION Jsmith.

CREATE TABLE EMPLOYEE

(Fname VARCHAR(15) NOT NULL,
Minit CHAR,
Ssn VARCHAR(9) NOT NULL,
Bdate DATE,

```
    Address     VARCHAR(30)    NOT NULL,
    Sex         CHAR,
    Salary      DECIMAL(10,2),
    Super_ssn   CHAR(9),
    Dno         INT            NOT NULL,
    PRIMARY KEY (ssn),
    FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(ssn),
    FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber);
```

CREATE TABLE DEPARTMENT

```
    (Dname          VARCHAR(15)    NOT NULL,
    Dnumber         INT            NOT NULL,
    Mgr_ssn         CHAR(9)        NOT NULL,
    Mgr_start_date  DATE,
    PRIMARY KEY (Dnumber),
    UNIQUE(Dname),
    FOREIGN KEY (mgr_ssn) REFERENCES EMPLOYEE(ssn));
```

CREATE TABLE DEPT_LOCATIONS

```
    (Dnumber        INT            NOT NULL,
    Dlocation       VARCHAR(15)    NOT NULL,
    PRIMARY KEY (Dnumber, Dlocation),
    FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber));
```

CREATE TABLE PROJECT

```
    (Pname          VARCHAR(15)    NOT NULL,
    Pnumber         INT            NOT NULL,
    Plocation       VARCHAR(15),
    Dnum            INT            NOT NULL
    PRIMARY KEY (Pnumber),
    UNIQUE(Pname),
    FOREIGN KEY (Dnum) REFERENCES
DEPARTMENT(Dnumber));
```

CREATE TABLE WORKS_ON

(Essn CHAR(9) NOT NULL,
Pno INT NOT NULL,
Hours DECIMAL(3,1) NOT NULL,
PRIMARY KEY(Essn, Pno),
FOREIGN KEY (Essn) REFRENCES EMPLOYEE(ssn),
FOREIGN KEY (Pno) REFRENCES PROJECT(Pnumber));

CREATE TABLE DEPENDENT

(Essn CHAR(9) NOT NULL,
Dependent_name VARCHAR(15) NOT NULL,
Sex CHAR,
Bdate DATE,
Relationship VARCHAR(8),
PRIMARY KEY(Essn, Dependent_name),
FOREIGN KEY (Essn) REFRENCES EMPLOYEE(Ssn));

Ở đây có một vài khoá ngoại có thể bị lỗi cú pháp bởi vì có những quan hệ vòng hoặc có một vài bảng liên quan tới một bảng chưa được tạo. Ví dụ, thuộc về khoá Super_ssn trong bảng EMPLOYEE là quan hệ vòng bởi vì nó liên quan tới chính bảng đó. Khóa ngoại Dno trong bảng EMPLOYEE có liên quan tới bảng DEPARTMENT mà bảng này chưa được tạo ra. Để giải quyết vấn đề này, những ràng buộc này có thể để ra khỏi các câu lệnh CREATE TABLE trong ví dụ trên và sẽ được bổ sung sau bằng câu lệnh ALTER TABLE (mục 6.3.2).

6.1.3. Các kiểu dữ liệu trong SQL

Những kiểu dữ liệu cơ bản có giá trị cho thuộc tính bao gồm: Numeric, Character string, Bit string, Boolean, Date và Time.

- Dữ liệu kiểu số (Numeric): là các kiểu dữ liệu số bao gồm các số nguyên với các kích cở khác nhau (INTEGER hoặc INT và SMALLINT) và các số với dấu phẩy động (real) khi cần biểu diễn độ chính xác (FLOAT hoặc REAL và DOUBLE PRECISION). Đinh dạng các số có thể được thể hiện bởi việc sử dụng cách biểu diễn DECIMAL(i,j) - hoặc DEC(i,j) hoặc NUMERIC (i,j) – Trong đó i là tổng độ dài chính xác của dãy số kể cả dấu chấm thập phân và j là số của chữ số sau dấu chấm thập phân. Mặc định j=0 và i là độ dài của kiểu dữ liệu được định nghĩa.

- Dữ liệu chuỗi ký tự (Character string): Là một chuỗi kí tự với độ dài là cố định hoặc có thể thay đổi.

Khai báo:

+ CHAR (n) và VARCHAR (n): để khai báo các chuỗi kí tự với độ dài chính xác, trong đó n là số kí tự.

+ VARCHAR((n), CHAR VARYING(n) hoặc CHARACTER VARYING(n): Chuỗi kí tự với độ dài có thể thay đổi được, trong đó n là số kí tự lớn nhất mà chuỗi có thể nhận được.

- Dữ liệu chuỗi Bit: Sử dụng khai báo BIT(n) khi cần khai báo chuỗi bit có độ dài cố định n; hoặc BIT VARYING(n), khi cần khai báo chuỗi BIT có độ dài có thể thay đổi- n là số lớn nhất của Bits.

- Dữ liệu Boolean: là những giá trị truyền thống TRUE hoặc FALSE. Trong SQL, vì sự hiện diện của những giá trị NULL nên kiểu dữ liệu logic ba giá thường được sử dụng, như vậy ba đánh giá đó như một kiểu dữ liệu Boolean như UNKNOWN. Chúng ta thảo luận sự cần thiết cho giá trị UNKNOWN và logic ba trị trong mục 6.5.6.

- Dữ liệu kiểu Date và Time: Kiểu dữ liệu DATE có độ dài 10 và nó gồm các thành phần YEAR, MONTH và DAY trong dạng YYYY-MM-DD. Dữ liệu kiểu TIME có ít nhất 8 vị trí với các thành phần HOUR, MINUTE và SECOND trong dạng HH:MM:SS. Chỉ những dữ liệu ngày và giờ hợp lệ mới cho phép thực hiện đầy đủ các phép tính toán trong SQL. Ngoài ra, sự so sánh "<" có thể sử dụng với tất cả dữ liệu ngày hoặc giờ.

6.2. Chỉ định những ràng buộc trong SQL

6.2.1. Ràng buộc vùng và giá trị mặc định của thuộc tính

- Giá trị NOT/NOT NULL.

Bởi vì SQL cho phép các thuộc tính nhận các giá trị NULL, một ràng buộc NOT NULL là được xác định nếu trong thực tế thuộc tính liên quan không thể nhận giá trị NULL. Gia trị NOT NULL luôn được xác định cho những thuộc tính khoá chính trong mỗi quan hệ, các thuộc tính còn lại có thể nhận giá trị này hay không tùy thuộc vào yêu cầu của lược đồ.

- Giá trị mặc định trên thuộc tính.

Giá trị mặc định cho mỗi thuộc tính có thể được xác định bằng cách sử dụng mệnh đề DEFAULT <value>. Khi có một bộ mới được thêm vào quan hệ và nếu có một thuộc tính của bộ không nhận được giá trị thì giá trị mặc định (nếu có) sẽ được thay vào.

- Ràng buộc vùng với mệnh đề CHECK

Cú pháp: [CONSTRAINT <constrainName>]
CHECK (condition)

Một kiểu ràng buộc khác được tác động trên phạm vi giá trị thuộc tính hoặc giá trị vùng là mệnh đề CHECK. Ví dụ, mã số bộ phận yêu cầu là những số nguyên nằm giữa 1 và 20, khi đó chúng ta có thể chuyển đổi khai báo thuộc tính của Dnumber trong bảng DEPARTMENT như sau:

Dnumber INT NOT NULL CHECK (Dnumber > 0 AND Dnumber < 21);

Mệnh đề CHECK có thể cũng được sử dụng cùng với câu lệnh CREATE DOMAIN. Ví dụ, chúng ta viết câu lệnh sau đây:

CREATE DOMAIN D_NUM AS INTEGER

CHECK (D_NUM > 0 AND D_NUM <21)

Các thuộc tính có thể sử dụng D_NUM là Dnumber của Department, Dnum của PROJECT, Dno của EMPLOYEE,...

Ví dụ 6.2:

Ví dụ minh hoạ cách khai báo giá trị mặc định của các thuộc tính và một số ràng buộc.

```
CREATE TABLE EMPLOYEE
        (...,
          Dno          INT          NOT NULL          DEFAULT 1,
        CONSTRAINT EMPPK
         PRIMARY KEY(Ssn)
        CONSTRAINT EMPSUPERFK
         FOREIGN KEY (Super_Ssn) REFERENCES EMPLOYEE(Ssn)
                        ON DELETE SET NULL        ON UPDATE CASCADE,
        CONSTRAINT EMPDEPTFK
          FOREIGN KEY (Dno) REFERENCES DEPARTMENT(Dnumber)
                        ON DELETE SET NULL        ON UPDATE CASCADE;
CREATE TABLE DEPARTMENT
        (...,
          Mgr_ssn  CHAR(9)    NOT NULL DEFAULT '888665555',
        ...,
        CONSTRAINT DEPTPK
          PRIMARY KEY(Dnumber)
```

CONSTRAINT DEPTSK

UNIQUE(Dname)

CONSTRAINT DEPTMGRFK

FOREIGN KEY (Mgr_ssn) REFERENCES EMPLOYEE(Ssn)

ON DELETE SET DEFAULT ON UPDATE CASCADE;

CREATE TABLE DEPARTMENT

(...,

PRIMARY KEY (Dnumber, Dlocation),

FOREIGN KEY (Dnumber) REFERENCES DEPARTMENT(Dnumber)

ON DELETE CASCADE ON UPDATE CASCADE;

6.2.2. Khoá chỉ định và các ràng buộc toàn vẹn

Các khóa và những ràng buộc toàn vẹn có liên quan đóng vai trò rất quan trọng, các đối tượng này được chỉ định trong câu lệnh CREATE TABLE thông qua những mệnh đề đặc biệt. Một hoặc nhiều thuộc tính được chỉ định trong mệnh đề PRIMARY KEY để tạo khoá chính của quan hệ. Nếu khoá chính có thuộc tính đơn thì mệnh đề có thể đi theo sau ngay thuộc tính.

Cú pháp tạo khóa chính:

```
[CONSTRAINT ConstrainName]
PRIMARY KEY [(ColumnList)]
```

Ví dụ, khoá chính của DEPARTMENT có thể được chỉ định sau đây (thay thế phương pháp được chỉ định trong Ví dụ 6.1):

Dnumber INT PRIMARY KEY;

Mệnh đề UNIQUE khai báo các khoá phụ như minh hoạ trong bảng DEPARTMENT và PROJECT được khai báo trong ví du 6.6.

Cú pháp tạo khóa phụ:

```
[CONSTRAINT ConstrainName]
 UNIQUE [(ColumnList)]
```

Như chúng ta đã bàn trong các mục trước, ràng buộc toàn vẹn có thể bị vi phạm khi các bộ được chèn hoặc xoá hoặc khi các khoá ngoài hoặc khoá chính có giá trị thuộc tính bị thay đổi. Thao tác mặc định mà SQL thực hiện là khi có lỗi toàn vẹn nó sẽ loại bỏ quá trình cập nhật và điều này sẽ gây ra lỗi. Tuy nhiên, người thiết kế lược đồ có thể chỉ định một tác động khác để thực hiện nếu ràng buộc toàn vẹn có liên quan bị vi phạm gắn liền với mệnh đề tác động toàn vẹn có liên quan đến mọi ràng buộc khoá ngoài. Bao gồm các lựa

chọn SET NULL, CASCADE và SET DEFAULT. Một lựa chọn sẽ mô tả với một trong hai trường hợp ON DELETE hoặc ON UPDATE. Ở đây, người thiết kế cơ sở dữ liệu chọn SET NULL ON DELETE và CASCADE UPDATE cho khoá ngoài Super_ssn của EMPLOYEE. Biện pháp này nếu có thành viên bị xoá, giá trị của Super_ssn trở thành bộ tự động về NULL cho tất cả các bộ thành viên nghĩa là liên quan đến việc xóa bộ thành viên. Một hướng khác, nếu giá trị Ssn trong quản lý thành viên được cập nhật, giá trị mới được xếp lớp với Super_ssn cho tất cả các bộ thành viên có liên quan với bộ thành viên đã cập nhật.

Cú pháp chung để tạo một khóa ngoại:

[CONSTRAINT ConstaintName]

FOREIGN KEY [(ColumnList)]

REFERENCES ReferenceTableName(ReferenceColumnName)

[ON DELETE CASCADE | NO ACTION | SET NULL | SET DEFAULT]

[ON UPDATE CASCADE | NO ACTION | SET NULL | SET DEFAULT]

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).

Mặc nhiên, lựa chọn thực hiện bởi DBMS đại diện cho SET NULL hoặc SET DEFAULT là như nhau cho cả hai ON DELETE và ON UPDATE: giá trị của các thuộc tính có liên quan thiếu chính xác sẽ được chuyển đổi về NULL bằng SET NULL và giá trị mặc nhiên được chỉ định là SET DEFAULT. Lựa chọn CASCADE ON DELETE sẽ xoá các bộ liên quan, ngược lại lựa chọn CASCADE ON UPDATE sẽ chuyển đổi dữ liệu của khoá ngoài tới cập nhật giá trị khoá chính cho tất cả các bộ có liên quan. Trách nhiệm của người thiết kế cơ sở dữ liệu là hướng tới sự chọn lựa thích hợp và

chỉ định trong lược đồ cơ sở dữ liệu. Bằng quy tắc thông thường, chọn lựa CASCADE là thích hợp cho những quan hệ "có quan hệ" như WORKS_ON; đại diện cho nhiều quan hệ có thuộc tính đa dạng như DEPT_LOCATIONS; đại diện cho nhiều quan hệ có các bộ thực thể yếu như DEPARTMENT.

6.2.3. Đặt tên cho những ràng buộc

Tên của tất cả các ràng buộc trong phạm vi là lược đồ riêng biệt bắt buộc tồn tại duy nhất. Tên một ràng buộc được dùng để nhận biết một ràng buộc riêng biệt trong trường hợp ràng buộc phải được trả về sau và thay thế với ràng buộc khác như chúng ta thảo luận trong mục 6.3. Tên cho các lược đồ là tuỳ ý.

6.2.4. Chỉ định những ràng buộc trên những bộ sử dụng CHECK

Ngoài ra, khoá và ràng buộc toàn vẹn có liên quan được chỉ định với từ khoá đặc biệt, bảng các ràng buộc khác có thể chỉ định rõ thông qua mệnh đề bổ sung CHECK kết thúc của câu lệnh CREATE TABLE. Điều đó có thể gọi các ràng buộc bộ dữ liệu cơ sở bởi vì nó áp dụng cho mỗi bộ riêng lẽ và kiểm tra mỗi khi một bộ được thêm vào hoặc giảm bớt. Ví dụ, giả sử rằng bảng DEPARTMENT có thêm thuộc tính Dept_create_date, được lưu trữ ngày khi thành viên đã được tạo ra. Lúc đó chúng ta có thể thêm sau mệnh CHECK kết thúc của câu lệnh CREATE TABLE cho bảng DEPARTMENT để có chắc chắn của việc bắt đầu quản lý ngày ban đầu và các ngày được tạo ra về sau.

CHECK (Dept_create_date <= Mgr_start_date);

Mệnh đề CHECK có thể cũng đã được sử dụng để chỉ rõ những ràng buộc chung hơn của việc sử dụng câu lệnh CREATE ASSERTION của SQL. Chúng ta thảo luận điều này trong mục 6.7 bởi vì nó qui định khả năng đầy đủ của truy vấn và đã được đã thảo luận trong mục 6.4 và 6.5..

6.3. Thay đổi lược đồ báo cáo trong SQL

Trong phần này, chúng ta có cách nhìn tổng quan về các câu lệnh sẵn có để phát triển lược đồ trong SQL, các câu lệnh này thường được sử dụng để sửa đổi một lược đồ như việc bổ sung hoặc xoá các bảng, các thuộc tính, các ràng buộc và các thành phần khác của lược đồ.

6.3.1. Lệnh DROP

Lệnh DROP có thể được sử dụng xoá tên các yếu thành phần của lược đồ như các bảng, các vùng hoặc các ràng buộc. Hơn nữa, ta có thể xoá một lược đồ. Ví dụ, nếu một lược đồ không cần sử dụng nữa thì có thể lệnh DROP SCHEMA để xóa nó. Ở đây có hai chọn lựa cách thực hiện CASCADE và RESTRICT. Ví dụ xoá lược đồ cơ sở dữ liệu COMPANY và tất cả các thành phần bên trong như các bảng, các vùng và thành phần khác ta sử dụng từ khóa CASCADE như sau:

DROP SCHEMA COMPANY CASCADE;

Nếu sử dụng RESTRICT thay cho CASCADE thì lược đồ sẽ bị xóa nếu các thành phần bên trong nó là rỗng; nếu ngược lại lệnh DROP sẽ không được thực hiện.

Nếu quan hệ cơ sở trong một lược đồ là cần thiết nữa thì quan hệ và các định nghĩa liên quan của nó có thể được xoá bởi lệnh DROP TABLE. Ví dụ, nếu chúng ta không cần theo dõi mối quan hệ của các nhân viên trong cơ sở dữ liệu COMPANY, chúng ta có thể bỏ đi quan hệ DEPENDENT bằng cách sử dụng lệnh sau đây:

DROP TABLE DEPENDENT CASCADE;

Nếu chọn RESTRICT thay vì chọn CASCADE, một bảng bị xoá khi và chỉ khi nó không có liên trong bất kỳ ràng buộc nào (ví dụ, khoá ngoài được định nghĩa thông qua một quan hệ khác) hoặc những khung nhìn (xem mục 6.8). Với lựa chọn CASCADE, tất cả những ràng buộc và những khung nhìn có mối quan hệ với các bảng bị xóa sẽ được tự động xoá.

Chú ý rằng lệnh DROP TABLE không chỉ xoá tất cả những mẫu tin trong bảng nếu được phép mà còn xóa cả cấu trúc bảng khỏi lược đồ. Nếu có yêu cầu xoá các mẫu tin nhưng cho phép giữ lại cấu trúc bảng để sử dụng về sau thì lệnh DELETE (xem mục 6.6.2) sẽ được sử dụng thay vì chọn lệnh DROP TABLE.

Ngoài ra, lệnh DROP còn thường được dùng để xoá các thành phần khác của lược đồ, như xóa ràng buộc hoặc vùng.

6.3.2. Lệnh ALTER

Các bảng cơ sở hoặc các thành phần khác của lược đồ sau khi đã định nghĩa có thể được sửa đổi bằng lệnh ALTER. Với một quan hệ cơ sỏ cho trước, lệnh Alter có thể thực hiện việc: bổ sung hoặc xóa một cột; thay đổi cách định nghĩa cột; bổ sung hoặc xóa bỏ các ràng buộc trong quan hệ.

Cú pháp chung của câu lệnh ALTER TABLE như sau:

ALTER TABLE *TableName*

ADD *ColumnDefine*

ALTER COLUMN <*ColumnName*> <*DataType*> [NULL | NOT NULL] |

DROP COLUMN <*ColumnName* >

ADD CONSTRAINT <*ConstraintName*> < *ConstraintDefine* >

DROP CONSTRAINT <*ConstraintName*>

Ví dụ, cần bổ sung thuộc tính để theo dõi các công việc của các nhân viên trong quan hệ cơ sở EMPLOYEE của lược đồ COMPANY, chúng ta có thể sử dụng lệnh:

ALTER TABLE COMPANY.EMPLOYEE

ADD COLUMN job VARCHAR(12);

Khi đó chúng ta cần nhập vào giá trị cho thuộc tính mới Job cho từng bộ riêng lẻ của quan hệ EMPLOYEE. Điều này có thể thực hiện qua mệnh đề Default hoặc sử dụng lệnh UPDATE (xem mục 6.6). Nếu mệnh đề Default không được chỉ định, thuộc tính mới sẽ nhận giá trị NULL trong tất cả các bộ của quan hệ khi lệnh được thực thi, và do đó ràng buộc NOT NULL sẽ không cho phép trong trường hợp này.

Để xoá cột, chúng ta bắt buộc phải chọn CASCADE hoặc RESTRICT. Nếu CASCADE được chọn, tất cả các ràng buộc và các khung nhìn có liên quan đến cột bị xóa sẽ được tự động xóa khỏi lược đồ cùng với cột bị xóa. Nếu RESTRICT được chọn, lệnh chỉ thực hiện được nếu không có các ràng buộc hoặc các khung nhìn (hoặc các thành phần khác) có liên quan đến cột. Ví dụ, câu lệnh sau sẽ xóa thuộc tính Adress khỏi bảng cơ sở EMPLOYEE:

ALTER TABLE COMPANY.EMPLOYEE DROP COLUMN Address CASCADE;

Câu lệnh ALTER cũng có thể chỉnh sưa một cột đã được định nghĩa như bỏ đi một mệnh đề DEFAULT đang tồn tại hoặc bổ sung một mệnh đề DEFAULT mới. Sau đây là ví dụ minh hoạ mệnh đề này:

ALTER TABLE COMPANY.DEPARTMENT ALTER COLUMN Mgr_ssn

DROP DEFAULT;

ALTER TABLE COMPANY.DEPARTMENT ALTER COLUMN Mgr_ssn

SET DEFAULT ‘333445555’;

Có thể chuyển đổi chỉ định các ràng buộc trên bảng bằng cách thêm vào hoặc xoá một ràng buộc. Khi nó được chỉ định xoá một ràng buộc bắt buộc phải có tên của nó. Ví dụ, xoá ràng buộc có tên EMPSUPERFK trong ví dụ 6.2 từ quan hệ EMPLOYEE, chúng ta viết:

ALTER TABLE COMPANY.EMPLOYEE

DROP CONSTRAINT EMPSUPERFK CASCADE;

Chúng ta có thể thêm vào một ràng buộc mới cho quan hệ, bằng cách sử dụng từ khoá ADD trong câu lệnh ALTER TABLE và theo sau là ràng buộc mới; ràng buộc mới có thể có tên hoặc không có.

6.4. Truy vấn cơ bản trong SQL

SQL có một câu lệnh căn bản cho việc tìm kiếm thông tin từ cơ sở dữ liệu: câu lệnh SELECT. Câu lệnh SELECT không có mối quan hệ tới tính toán SELECT của đại số quan hệ được thảo luận trong chương trước. Ở đây có nhiều lựa chọn và thêm vào câu lệnh SELECT trong SQL, vì vậy chúng ta sẽ dần dần giới thiệu nét đặc biệt của nó.

Trước khi bắt đầu, chúng ta xem xét điểm khác biệt quan trọng giữa SQL và mô hình quan hệ: SQL cho phép một bảng (quan hệ) có hai hoặc nhiều bộ là giống nhau trong tất cả các giá trị thuộc tính của nó. Vì lý do đó, toàn bộ bảng của SQL không phải thiết lập của các bộ bởi vì thiết lập đó không cho phép hai bộ phận giống nhau, hơn nữa một bảng trong SQL không phải là một tập hợp của những bộ, bởi vì một tập hợp không cho phép có những phần tử giống hệt nhau. Thông qua một số ví dụ chúng ta sẽ rõ hơn các vấn đề này.

6.4.1. Cấu trúc truy vấn cơ bản SELECT-FROM-WHERE

Truy vấn trong SQL có thể rất phức tạp. Chúng ta sẽ bắt đầu với ví dụ tiêu biểu về các truy vấn và sau đó phát triển cái phức tạp hơn theo cách xử lý từng bước một. Dạng căn bản của câu lệnh SELECT, một vài đòi hỏi sự sắp xếp hoặc một khối Select-From-Where là sự thiết lập của ba mệnh đề SELECT, FROM và WHERE và như sự thiết lập sau đây:

SELECT <attribute list>

FROM <table list>

WHERE <condition>

Trong đó:

- <attribute list>: liệt kê các tên thuộc tính liên quan đến những giá trị cần tìm trong câu truy vấn.

- <table list>: liệt kê các tên quan hệ cần xử lý của câu truy vấn.

- <condition>: Là biểu thức điều kiện (Logic) trích lọc các bộ cần tìm trong câu truy vấn.

Trong SQL, các toán tử so sánh cơ bản dùng để so sánh các giá trị thuộc tính với một giá trị khác. Các toán tử so sánh gồm =, <, <=, >, >= và <>. Điều này phù hợp với các toán tử đại số quan hệ =, <, ≤ , >, ≥ và ≠ theo thứ tự và với các toán tử của ngôn ngữ lập trình C/C++: =, <, <=, >, >= và !=. Sự khác nhau chính là toán tử không giống nhau.

Chúng ta minh hoạ câu lệnh SELECT cơ bản trong SQL với một vài ví dụ truy vấn. Các câu truy vấn có nhãn ở đây với những giá trị truy vấn giống nhau đã được đề cập và dễ dàng tham khảo trong chương trước.

Truy vấn 0: Tìm ngày sinh và địa chỉ của các thành viên có tên là ‘John B. Smith’.

```
Q0:  SELECT   Bdate, Adress
     FROM     EMPLOYEE
     WHERE    Fname=’John’ AND Minit=’B’ AND Lname=’Smith’;
```

Câu truy vấn này chỉ sử dụng quan hệ EMPLOYEE trong mệnh đề FROM. Truy vấn thực hiện lựa chọn các bộ trong quan hệ EMPLOYEE thoả mãn điều kiện của mệnh đề WHERE, kết quả được liệt kê theo thuộc tính Bdate và Adress trong mệnh đề SELECT. Q0 tương tự với biểu diễn đại số quan hệ sau đây:

$$\pi_{Bdate,Adress}(\sigma_{Fname=’John’\ AND\ Minit=’B’\ AND\ Lname=’Smith’}(EMPLOYEE))$$

Mệnh đề SELECT của SQL chỉ rõ những thuộc tính được tham chiếu và mệnh đề WHERE chỉ rõ việc lựa chọn điều kiện. Sự khác nhau trong câu truy vấn SQL là chúng ta có được những bộ giống nhau trong kết quả bởi vì sự ràng buộc của một quan hệ là một tập hợp không bắt buộc.

Kết quả của các truy vấn SQL khi ứng dụng hiển thị trạng thái cơ sở dữ liệu COMPANY: (a) Q0. (b) Q6. (c) Q2. (d) Q8. (e) Q9. (f) Q10. (g) Q1C.

(a)

Bdate	Adress
1965-01-09	731 Fondren, Houston, TX

(b)

Fname	Lname	Adress
Jonh	Smith	731 Fondren, Houston, TX
Frankin	Wong	638 Voss, Houston, TX
Ramesh	Narayan	975 Fire Oak, Humble, TX
Joyce	English	5631 Rice, Houston, TX

(c)

Pnumer	Dnum	Lname	Adress	Bdate
10	4	Wallace	291 Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291 Berry, Bellaire, TX	1941-06-20

(d)

E.Fname	E.Lname	S.Fname	S.Lname
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennier	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	Englísh	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

(e)

E.Fname
123456789
333445555
999887777
987654321
666884444
453453453
987987987
888665555

(f)

Ssn	Dname
123456789	Research
333445555	Research
999887777	Research
987654321	Research
666884444	Research
453453453	Research
987987987	Research
888665555	Research
123456789	Administration
333445555	Administration
999887777	Administration
987654321	Administration
666884444	Administration
453453453	Administration
987987987	Administration
888665555	Administration
123456789	Headquarters
333445555	Headquarters
999887777	Headquarters
987654321	Headquarters
666884444	Headquarters
453453453	Headquarters
987987987	Headquarters
888665555	Headquarters

(g)

Fname	Minit	Lname	Ssn	Bdate	Adress	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-09-01	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Ramesh	K	Narrayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

Truy vấn Q0 cũng tương tự với phép tính biểu thức quan hệ bộ dữ liệu sau, trừ những bản sao, nếu có, lần nữa không được loại trừ trong câu hỏi SQL.

Q0: (t.Bdate, t.Adress| EMPLOYEE(t) AND t.Fname=’John’ AND t.Minit=’B’

AND t.Lname=’Smith’)

Do đó, chúng ta có thể thấy một bộ biến dữ liệu ẩn trong câu truy vấn SQL trên phạm vi mỗi bộ trong bảng EMPLOYEE và đánh giá điều kiện trong mệnh đề WHERE. Chỉ những bộ thoả mãn điều kiện, điều đó có nghĩa là đánh giá diều kiện của các bộ la TRUE sau khi thay thế các giá trị thuộc tính đúng với nó được chọn.

Query 1: Tìm tên và địa chỉ của của tất các thành viên thuộc bộ phận ‘Reseach’.

Q1: SELECT Fname, Lname, Adress
FROM EMPLOYEE, DEPARTMENT
WHERE Dname=’Research’ AND Dnumber=Dno;

Truy vấn Q1 tương tự với trình tự SELECT-PROJECT-JOIN của phép toán đại số quan hệ. Thí dụ như câu truy vấn thường gọi **select-project-join.** Trong mệnh đề SQL của Q1, điều kiện Dname=’Research’ là điều kiện được chọn và phù hợp với phép toán SELECT trong đại số quan hệ. Điều kiện Dnumber=Dno là điều kiện kết hợp và phù hợp với điều kiện JOIN trong đại số quan hệ. Nói chung, nhiều giá trị của các điều kiện Select và Join được chỉ rõ trong một câu lệnh SQL: Tiếp theo là ví dụ về câu truy vấn **select-project-join** với hai điều kiện nối liền với nhau:

Query 2: Liệt kê số lượng đề án, số lượng bộ phận quản lý, tên người quản lý bộ phận, địa chỉ và ngày sinh tại địa phương ’Stafford’.

Q2: SELECT Pnumber, Dnum, Lname, Adress, Bdate
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn AND
Plocation=’Stafford’;

Điều kiện phù hợp Dnum=Dnumber có liên quan giữa đề án và bộ phận quản lý của nó nhưng ngược lại điều kiện tiếp theo Mgr_ssn=Ssn có liên quan giữa bộ phận quản lý và người quản lý thành viên.

6.4.2. Các tên, bí danh và các biến bộ thuộc tính ảo

Trong SQL một tên có thể dùng cho hai (hoặc nhiều) thuộc tính trong các quan hệ khác nhau. Nếu đây là trường hợp được xem là truy vấn đến hai hoặc nhiều thuộc tính cùng tên, chúng ta cần phải nói rõ tên thuộc tính với tên quan hệ để ngăn ngừa sự nhập nhằng. Điều này được thực hiện bằng việc thêm vào đầu là tên quan hệ rồi tới tên thuộc tính và phân cách bởi dấu chấm. Để minh hoạ điều này, giả sử rằng thuộc tính Dno và Lname của quan hệ EMPLOYEE được xem là Dnumber và Name, và thuộc tính Dname của DEPARTMENT đã được xem là Name; để ngăn ngừa sự nhập nhằng, câu truy vấn Q1 sẽ được nói lại bằng cách đưa vào trong Q1A. Chúng ta cần phải thêm vào đầu những thuộc tính Name và Dnumber trong Q1A để chỉ rõ cái nào để chúng ta cần xem xét bởi vì các tên thuộc tính được sử dụng cho cả hai quan hệ.

Q1A: SELECT Fname, EMPLOYEE.Name, Adress
FROM EMPLOYEE , DEPARTMENT
WHERE DEPARTMENT.Name=’Research’ AND

DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;

Tình trạng có nhiều nghĩa cũng xảy ra trong trường hợp câu truy vấn có hai quan hệ giống nhau trong ví dụ sau đây:

Query 8: Liệt kê họ và tên từng nhân viên và họ và tên của người quản lý trực tiếp anh ấy.

```
Q8:  SELECT   E.Fname, E.Lname, S.Fname, S.Lname
     FROM     EMPLOYEE AS E, EMPLOYEE AS S
     WHERE    E.super_ssn=S.Ssn;
```

Trong trường hợp này, chúng ta chấp nhận khai báo một trong hai tên quan hệ E và S, gọi là các bí danh hoặc các biến bộ cho quan hệ EMPLOYEE. Một bí danh có thể đi theo sau từ khoá AS, được trình bày trong Q8, hoặc nó có thể đi theo sau ngay tên quan hệ - Ví dụ, viết EMPLOYEE E, EMPLOYEE S trong mệnh đề FROM của Q8. Nó cũng có thể thực hiện được việc đổi tên các thuộc tính quan hệ trong phạm vi câu truy vấn trong SQL bằng cách cho nó bí danh. Ví dụ, nếu chúng ta viết:

EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex, Sal, Sssn, Dno)

trong mệnh đề FROM, Fn trở thành một bí danh cho Fname, Mi cho Minit, Ln cho Lname,...

Trong Q8, chúng ta có thể thấy E và S là hai bản sao khác nhau của quan hệ EMPLOYEE, đầu tiên E mô tả các thành viên trong vai trò chịu sự quản lý; thứ hai S mô tả các thành viên trong vai trò quản lý. Bây giờ chúng ta có thể có hai bản sao phù hợp. Một vấn đề dĩ nhiên, trong thực tế ở đây chỉ có một quan hệ EMPLOYEE phù hợp với điều kiện nối quan hệ với chính nó bởi sự phù hợp các bộ thảo mãn điều kiện nối E.Super_ssn=S.Ssn. Chú ý rằng trong ví dụ này một mức truy vấn đệ quy bằng cách chúng ta thảo luận trong mục 6.4.2. Trong phiên bản đầu tiên của SQL, đại số quan hệ là không thể thực hiện được với toàn bộ truy vấn đệ quy được chỉ định với một ẩn số của các vị trí trong một câu lệnh SQL. Một xây dựng bằng cách chỉ rõ các truy vấn đệ quy được kết hợp chặt chẽ vào trong SQL-99, được thảo luận trong chương 22.

Các kết quả của câu truy vấn Q8 được trình bày trong hình (d). Bất cứ lúc nào một hoặc nhiều bí danh được gán cho quan hệ, chúng ta có thể sử dụng các tên này để có kết quả khác nhau có liên quan trong quan hệ. Điều này cho phép sự liên quan đến cùng quan hệ trong phạm vi câu truy vấn. Chú ý rằng, nếu cần chúng ta có thể dùng kỹ thuật tên bí danh trong nhiều câu truy vấn SQL với việc chỉ rõ bộ các biến cho mỗi bảng trong mệnh đề WHERE. Trong thực tế, hành động này ngược lại với lý thuyết được giới thiệu từ những

kết quả trong các câu truy vấn mà được hiểu dễ dàng. Ví dụ, chúng ta có thể chỉ rõ câu truy vấn Q1A bằng câu Q1B:

Q1B: SELECT E.Fname, E.Lname, E.Adress
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.Name='Research' AND D.Dnumber=E.Dnumer;

Nếu chúng ta chỉ rõ bộ các biến cho mỗi bảng trong mệnh đề WHERE, một câu truy vấn select-project-join trong SQL chặt chẽ tương tự đúng với biểu thức phép tính có quan hệ bộ dữ liệu tương ứng. Ví dụ, so sánh Q1B với biểu thức phép tính có quan hệ bộ dữ liệu tương ứng sau trên các bảng gốc:

Q1: {e.Fname, e.Lname, e. Adress| EMPLOYEE(e) AND (∃d)
(DEPARTMENT(d) AND d.Dname='Research' AND d.Dnumber=e.Dno)}

6.4.3. Không sử dụng mệnh đề WHERE và cách sử dụng dấu *

- Không sử dụng mệnh đề WHERE: Khi không chỉ định mệnh đề WHERE, tức không có điều kiện để lựa chọn bộ dữ liệu thì tất cả các bộ dữ liệu của quan hệ chỉ được xác định qua mệnh đề FROM và SELECT để lựa chọn cho kết quả truy vấn. Nếu có nhiều hơn một quan hệ được xác định trong mệnh đề FROM và mệnh đề WHERE ở đây thì CROSS PRODUCT – có thể kết hợp tất cả các bộ - của lựa chọn này. Ví dụ, Query 9 chọn tất cả EMPLOYEE Ssns và truy vấn 10 chọn tất cả tổ hợp của một EMPLOYEE Ssn và một DEARTMENT Dname.

Query 9 và 10: Chọn tất cả EMPLOYEE Ssns (Q9) và tất cả tổ hợp của EMPLOYEE Ssn và DEPARMENT Dname Q(10) trong cơ sở dữ liệu.

Q9: SELECT Ssn
FROM EMPLOYEE;

Q10: SELECT Ssn
FROM EMPLOYEE, DEPARTMENT;

Cực kỳ quan trọng khi chỉ rõ một lựa chọn và điều kiện thành viên trong mệnh đề WHERE nếu như nhiều điều kiện bỏ qua không đúng và cho kết quả quan hệ rất lớn. Chú ý trong Q10 là giống nhau CROSS PRODUCT thoả mãn phép toán tiếp theo bằng phép toán PROJECT trong đại số quan hệ. Nếu chúng ta chỉ định tất cả các thuộc tính của EMPLOYEE và DEPARMENT trong Q10, chúng ta có được CROSS PRODUCT.

- Cách sử dụng dấu *

Khi cần liệt kê tất cả các giá trị của các thuộc tính của các bộ dữ liệu được lựa chọn, chúng ta không cần liệt kê các tên thuộc tính một cách rõ ràng; chúng ta sử dụng dấu * sau mệnh đề SELECT đại diện cho tất cả các thuộc

tính. Ví dụ, câu truy vấn Q1C tìm tất cả các nhân viên trong quan hệ EMPLOYEE làm việc tại bộ phận số 5 trong quan hệ DEPARTMENT. Truy vấn Q1D tìm tất cả các giá trị thuộc tính của nhân viên EMPLOYEE và các thuộc tính của DEPARTMENT trong đó anh ta hoặc cô ta là các thành viên của bộ phận "Research". Và Q10A chỉ định CROSS PRODUCT của các quan hệ EMPLOYEE và DEPARTMENT.

Q1C: SELECT *
FROM EMPLOYEE
WHERE Dno=5;

Q1D: SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dno=Dnumber;

Q10A: SELECT *
FROM EMPLOYEE, DEPARTMENT

6.4.4. Các bảng như các tập hợp trong SQL

Như chúng ta đề cập ở phần trước, với SQL một bảng không được xem như một tập hợp hay đúng hơn nó được xem như một Mulltiset, các bộ trùng có thể được xuất hiện khá nhiều trong một bảng và trong kết quả của một truy vấn. SQL không tự động loại ra các bộ trung trong các kết quả của các câu truy vấn bởi vì lý do sau đây:

. Sự loại trừ các bộ trùng lặp là phép toán quá đắt. Một phương pháp để thực hiện là: trước hết nó phân loại bộ dữ liệu và sau đó loại trừ bộ trùng lặp.

. Người sử dụng có thể cần thấy các bộ trùng trong kết quả truy vấn.

. Khi một hàm tổng hợp (xem mục 6.5.7) được áp dụng cho các bộ dữ liệu, hầu hết chúng ta không muốn loại bỏ các bộ trùng lặp.

Một bảng SQL có một khoá cho ta một tập hợp, với giá trị khoá giúp phân biệt giá trị giữa các bộ dữ liệu trong quan hệ. Nếu chúng ta muốn loại trừ các bộ trùng từ kết quả của câu truy vấn SQL, chúng ta dùng từ khoá DISTINCT trong mệnh đề SELECT, có nghĩa là những bộ dữ liệu phân biệt còn lại trong kết quả. Nói chung, câu truy vấn với SELECT DISTINCT loại trừ nhân đôi, nhưng ngược lại không có câu truy vấn với SELECT ALL. Chỉ định SELECT không kèm từ khóa ALL và cũng không có từ khóa DISTINCT – trong ví dụ trước - là tương đương mệnh đề SELECT ALL. Ví dụ, Q11 tìm tiền lương mỗi thành viên, nếu có một vài thành viên khác nhau có tiền lương giống nhau thì giá trị tiền lương sẽ xuất hiện nhiều lần trong kết quả của câu truy vấn. Nếu chúng ta quan tâm chỉ các giá trị tiền lương riêng, chúng ta cần

có mỗi giá trị chỉ xuất hiện một lần, không chú ý tới việc có bao nhiêu thành viên nhận được tiền lương đó, chúng ta sử dụng từ khoá DISTINCT trong Q11A để giải quyết điều này.

(a)

Salary
30000
40000
25000
43000
38000
25000
25000
55000

(b)

Salary
30000
40000
25000
43000
38000
55000

(c)

Fname	Lname

(d)

Fname	Lname
James	Borg

Query 11: Tìm tiền lương của mỗi thành viên (Q11) và tất cả các giá trị tiền lương riêng.

```
Q11:    SELECT   ALL Salary
        FROM     EMPLOYEE;
Q11A:   SELECT   DISTINCT Salary
        FROM     EMPLOYEE;
```

SQL có ngay tức khắc một vài sáp nhập của phép toán tập hợp trong đại số quan hệ. Ở đây các phép tính tập hợp liên kết (UNION), tập hợp khác nhau (EXCEPT) và tập hợp giao nhau (INTERSECT). Kết quả của các quan hệ từ phép tính tập hợp này nhiều tập hợp của các bộ dữ liệu, điều đó các bộ nhân đôi sẽ bị loại trừ từ kết quả. Bởi vì chỉ thêm các phép tính của tập hợp này từ các quan hệ liên kết-tương thích, chắc chắn rằng chúng ta phải tạo hai quan hệ trên đó chúng ta dùng phép toán có các thuộc tính giống nhau và các thuộc tính xuất hiện theo thư tự trong cả hai quan hệ. Ví dụ tiếp theo minh hoạ cách sử dụng UNION.

Query 4: Tạo một danh sách về số đề án của các thành viên có last name là ‘Smith’ trừ các nhân viên hoặc người quản lý của bộ phận điều khiển đề án.

```
Q4:  (SELECT   DISTINCT Pnumber
     FROM      PROJECT, DEPARTMENT, EMPLOYEE
     WHERE     Dnum=Dnumber AND Mgr_ssn=Ssn
               AND  Lname='Smith')
     UNION
     (SELECT   DISTINCT Pnumber
     FROM      PROJECT, WORK_ON, EMPLOYEE
```

```
WHERE    Pnumber=Pno AND Essn=Ssn
         AND  Lname='Smith');
```

Đầu tiên câu truy vấn SELECT tìm các thành viên có tên 'Smith' là bộ phận quản lý điều khiển đề án, thứ hai tìm các thành viên có tên 'Smith'. Chú ý rằng nếu các thành viên khác nhau có tên cuối cùng là 'Smith' thì các tên của thành viên, các tên đề án kéo theo bất kỳ của chúng sẽ tìm được. Áp dụng thao tác UNION với hai SELECT truy vấn sẽ cho kết quả mong muốn.

SQL cũng đúng với các phép tính nhiều tập hợp, đi theo sau bởi từ khoá ALL (UNION ALL, EXCEPT ALL, INTERSECT ALL). Những kết quả của nó là nhiều tập hợp (nhân đôi không loại trừ). Về cơ bản, có phải nó là một bản sao hoặc không khi được xem xét như một bộ dữ liệu khác nhau khi áp dụng vào thao tác này.

6.4.5. Sự thích hợp mẫu chuỗi con và toán tử số học

Trong mục này chúng ta thảo luận một số điểm đặc trưng của SQL. Điểm đặc trưng đầu tiên là so sánh trên các thành phần của chuỗi ký tự, dùng toán tử LIKE để so sánh. Có thể sử dụng chuỗi phù hợp với mẫu. Các chuỗi bộ phận được chỉ định sử dụng hai ký tự dành riêng: Kí tự "%" thay thế một số bất kỳ zero hoặc nhiều ký tự; Ký tự gạch dưới "_" thay thế cho một ký tự. Ví dụ, tính toán của câu truy vấn sau đây:

Query 12: Tìm tất cả các thành viên có địa chỉ là Houston, Texas.

```
Q12: SELECT    Fname, Lname
     FROM      EMPLOYEE
     WHERE     Address LIKE '%Houston, TX%';
```

Tìm tất các thành viên có năm sinh trong thời gian 1950, chúng ta có thể sử dụng Query 12A. Ở đây, '5' là ký tự thứ ba của chuỗi (theo định dạng ngày), nhưng chúng ta sử dụng giá trị '_ _ 5 _ _ _ _ _ _ _ _' với mỗi dấu gạch dưới giữ chỗ cho ký tự bất kỳ.

Kết quả của các phép toán multiset SQL: (a) Hai bảng R(A) và S(A), (b) R(a) hợp toàn bộ S(A), (c) R(A) loại trừ tàon bộ S(A), (d) R(A) phân cắt toàn bộ S(A)

(a) R

A
a1
a2
a2
a3

S

A
a1
a2
a4
a5

(b) T

A
a1
a1
a2
a2
a2
a3
a4

(c) T

A
a1
a2

(d) T

A
a2
a3

Query 12A: Tìm tất cả các thành viên có năm sinh trong thời gian 1950.

Q12: SELECT Fname, Lname
FROM EMPLOYEE
WHERE Bdate LIKE '_ _ 5 _ _ _ _ _ _ _';

Nếu dấu gạch dưới hoặc dấu % cần phải là các ký tự đúng trong chuỗi thì ký tự sẽ đi trước bởi một ký tự *escape* (thoát khỏi), điều này được chỉ định sau chuỗi so sánh bằng từ khoá ESCAPE. Ví dụ, 'AB_CD\%EF' ESCAPE '\' tương ứng với chuỗi 'AB_CD%EF' bởi vì \ được chỉ định bằng ký tự *escape*. Ký tự bất kỳ không sử dụng trong chuỗi có thể được chọn bằng ký tự *escape*. Ngoài ra, chúng ta cần phải chỉ định đúng quy tắc của dấu móc lững hoặc ghi trong dấu (' ') bao gồm trong cả chuỗi ký tự bởi vì nó được sử dụng cho bắt đầu và kết thúc các chuỗi. Nếu thiếu dấu móc lững (') hoặc có hai dấu móc lững liên tiếp ('') thì nó sẽ không thể hiện được việc kết thúc chuỗi.

Một vấn đề cần quan tâm khác là việc dùng các phép tính số học trong câu truy vấn. Các toán tử số học chuẩn là phép cộng (+), phép trừ (-), phép nhân (*) và phép chia (/) có thể ứng dụng vào các giá trị số hoặc các thuộc tính với các phạm vi số. Ví dụ, giả sử rằng chúng ta muốn xem kết quả của tất cả các thành viên có công việc trong đề án 'ProductX' là trên 10%; chúng ta có thể đưa ra Query 13 để xem họ được trả tiền lương thích hợp. Ví dụ này cũng trình bày làm thế nào chúng ta có thể đổi tên một thuộc tính trong kết quả câu truy vấn bằng cách sử dụng AS trong mệnh đề SELECT.

Query 13: Trình bày kết quả tiền lương của mỗi thành viên có công việc trong đề án 'ProductX'

Q13: SELECT Fname, Lname, 6.1*Salary AS Increased_sal

```
FROM    EMPLOYEE, WORKS_ON, PROJECT
WHERE   Ssn=Essn AND Pno=Pnumber AND
        Pname='ProductX';
```

Cho các kiểu dữ liệu chuỗi, toán tử nối ‖ có thể sử dụng trong câu truy vấn để nối hai giá trị chuỗi ký tự. Cho ngày, giờ, kế hoạch làm việc và các kiểu dữ liệu khoảng thời gian, các toán tử bao gồm gia tăng (+) hoặc giảm (-) một ngày, giờ hoặc kế hoạch làm việc trong một khoảng thời gian. Trong phép cộng, giá trị khoảng thời gian là có kết quả khác nhau giữa giá trị hai ngày, giờ hoặc kế hoạch làm việc. Tương tự cho toán tử so sánh có thể sử dụng sự thuận tiện bằng BETWEEN được minh hoạ trong Query 14.

Query 14: Tìm tất cả các thành viên thuộc bộ phận 5 có tiền lương giữa $30,000 và $40,000.

Q14:
```
SELECT  *
FROM    EMPLOYEE
WHERE   (Salary BETWEEN 30000 AND 40000) AND Dno = 5;
```

Điều kiện (Salary BETWEEN 30000 AND 40000) trong Query 14 là tương đương với điều kiện ((Salary >= 30000) AND (Salary <= 40000)).

6.4.6. Thứ tự của những kết quả truy vấn

SQL cho phép người sử dụng sắp thứ tự các bộ dữ liệu trong kết quả của câu truy vấn bởi những giá trị của một hoặc nhiều thuộc tính bằng cách dùng mệnh đề ORDER BY. Điều này được minh họa trong Query 15.

Query 15: Tìm liệt kê các thành viên và các đề án mà họ đang làm việc, sắp thư tự bộ phận, phạm vi mỗi bộ phận, sắp xếp a,b,c họ, tên.

Q15:
```
SELECT  Dname, Lname, Fname, Pname
FROM    DEPARTMENT, EMPLOYEE, WORKS_ON, PROJECT
WHERE   Dnumber=Dno AND Ssn=Essn AND Pno=Pnumber
ORDER BY    Dname, Lname
```

Sắp xếp mặc nhiên là sắp xếp các giá trị tăng dần. Chúng ta có thể chỉ định từ khoá DESC nếu chúng ta muốn có kết quả sắp xếp các giá trị giảm dần. Từ khoá ASC có thể được sử dụng để chỉ định dứt khoát sắp xếp tăng dần. Ví dụ, nếu chúng ta muốn sắp xếp giảm dần trên Dname và sắp xếp giảm dần trên Lname, Fname, mệnh đề ORDER của Q15 có thể được viết là:

```
ORDER BY  Dname DESC, Lname ASC, Fname ASC
```

6.5. Những truy vấn SQL phức tạp

Trong mục trước, chúng ta đã mô tả một vài kiểu căn bản của câu truy vấn trong SQL. Bởi vì một cách tổng quát và có ý nghĩa sức mạnh của ngôn ngữ, ở đây có nhiều các điểm đặc trưng được thêm vào để cho phép những người sử dụng chỉ định các câu truy vấn phắc tạp hơn. Chúng ta có nhiều thảo luận về các điểm đặc trưng đó trong mục này.

6.5.1. Những so sánh kéo theo NULL và logic ba trị

SQL có các quy tắc khác nhau để xử lý các giá trị NULL. NULL được sử dụng tương ứng với giá trị không thể tìm thấy nhưng nó thường có một của ba sự thể hiện khác nhau – không nhận biết (có tồn tại nhưng không nhận biết), giá trị không thể dùng được (có tồn tại nhưng cố ý giấu) hoặc thuộc tính không thể dùng được (không xác định được bộ này). Tính toán của các ví dụ sau đây minh hoạ cho mỗi ý nghĩa của NULL.

. Giá trị không nhận biết.

. Giá trị không dùng được hoặc cố ý dấu.

. Thuộc tính không thể dùng được.

Không thể xác định được các ý nghĩa như dự định, Ví dụ, một NULL cho số điện thoại nhà riêng của một người có thể có ba ý nghĩa. Do đó, SQL không phân biệt giữa các ý nghĩa khác nhau của NULL.

Nói chung, mỗi NULL được xem là khác nhau từ mỗi NULL khác nhau trong cơ sở dữ liệu. Khi một NULL là phức tạp trong phép toán điều kiện có kết quả xem như UNKNOWN (nếu có thể TRUE hoặc có thể FALSE). Do đó, SQL dùng logic ba trị với các giá trị TRUE, FALSE và UNKNOW để thay thế cho chuẩn logic hai trị với các giá trị TRUE hoặc FALSE. Vì thế sự cần thiết để chỉ rõ tính chất thể hiện các kết quả của logic ba trị khi hợp lý sử dụng các liên kết AND, OR, và NOT.

Các giá trị của các dòng và các cột tương ứng với các kết quả thể hiện logic của logic hai trị (các điều kiện so sánh) điều này sẽ xuất hiện tiêu biểu trong mệnh đề WHERE của câu truy vấn SQL. Mỗi kết quả sẽ có giá trị của TRUE, FALSE hoặc UNKNOW. Kết quả của việc sử dụng hai kết nối AND sẽ được trình bày trong bảng (a) và bảng (b) sẽ trình bày kết quả của việc sử dụng kết nối OR. Ví dụ, kết quả của (FALSE AND UNKNOW) là FALSE, ngược lại kết quả của (FALSE AND UNKNOW) là UNKNOWN. Bảng 8.1(c) trình bày kết quả phép toán NOT. Chú ý rằng BOOLEAN chuẩn chỉ cho phép có các giá trị TRUE hoặc FALSE, ở đây không có giá trị UNKNOWN.

Trong câu truy vấn select-project-join, toàn bộ quy tắc là sự kết hợp của các bộ dữ liệu được đánh giá là hợp lý của việc chọn TRUE trong câu truy

vấn. Bộ dữ liệu được đánh giá là hợp lý là không chọn lựa FALSE hoặc UNKNOWN.

SQL cho phép các câu truy vấn kiểm tra có hay không giá trị thuộc tính là NULL. Để sử dụng dấu = hoặc <> so sánh giá trị thuộc tính NUL, SQL sử dụng IS hoặc IS NOT.

Query 18: Tìm tên của các thành viên đó là những người không phải giám sát.

```
Q18: SELECT   Fname, Lname
     FROM     EMPLOYEE
     WHERE    Super_ssn IS NULL;
```

Những liên kết hợp lý trong logic ba trị:

(a)

AND	TRUE	FALSE	UNKNOWN
TRUE	TRUE	FALSE	UNKNOWN
FALSE	FALSE	FALSE	FALSE
UNKNOWN	UNKNOWN	FALSE	UNKNOWN

(b)

OR	TRUE	FALSE	UNKNOWN
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	UNKNOWN
UNKNOWN	TRUE	UNKNOWN	UNKNOWN

(c)

NOT	
TRUE	FALSE
FALSE	TRUE
UNKNOWN	UNKNOWN

6.5.2. Những so sánh truy vấn lồng nhau, nhiều bộ và nhiều tập hợp

Một vài câu truy vấn phụ thuộc vào các giá trị hiện tại trong cơ sở dữ liệu có bẫy các truy vấn và sử dụng điều kiện so sánh. Các câu truy vấn như vậy có thể đưa vào công thức thuận tiện bằng cách sử dụng kỹ thuật các câu truy vấn với đầy đủ select-from-where bên trong mệnh đề WHERE của câu truy vấn khác. Câu truy vấn khác là câu truy vấn phía bên ngoài. Query 4 đưa vào công thức trong Q4 không có bẫy truy vấn nhưng nó có thể phát biểu lại

bẫy các truy vấn trong Q4A. Q4A giới thiệu toán tử điều kiện IN dùng để so sánh giá trị *v* với một tập hợp (hoặc nhiều tập hợp) của các giá trị *V* và đánh giá TRUE nếu *v* là một yếu tố trong *V*:

```
Q4A: SELECT   DISTINCT Pnumber
     FROM     PROJECT
     WHERE    Pnumber IN
              (SELECT   Pnumber
              FROM      PROJECT, DEPARTMENT, EMPLOYEE
              WHERE     Dnum=Dnumber AND
                        Mgr_ssn=Ssn AND Lname='Smith')
              OR
              Pnumber IN
              (SELECT   Pno
              FROM      WORK_ON, EMPLOYEE
              WHERE     Essn=Ssn AND Lname='Smith');
```

Lựa chọn thứ nhất của bẫy câu truy vấn là các số đề án của các đề án nghĩa là có 'Smith' kéo theo giống người quản lý, lựa chọn thứ hai là các số đề án của các đề án có 'Smith' kéo theo giống nhân viên. Trong câu truy vấn ở phía ngoài, chúng ta sử dụng logic nối OR để tìm bộ PROJECT nếu giá trị PNUMBER của bộ dữ liệu này có kết quả của câu truy vấn khác.

Nếu bẫy câu truy vấn trả về một thuộc tính đơn và một bộ dữ liệu đơn thì kết quả câu truy vấn sẽ là giá trị đơn. Trong trường hợp nó có thể chấp nhận được bằng cách sử dụng dấu = thay cho IN trong toán tử so sánh. Nói chung, bẫy câu truy vấn sẽ trả về một bảng (quan hệ) với một tập hợp hoặc nhiều tập hợp của các bộ dữ liệu.

SQL cho phép sử dụng của các bộ của các giá trị trong các điều kiện bằng cách đặt phạm vi của nó trong dấu ngoặc đơn. Thảo luận điều này, xem câu truy vấn sau đây:

```
SELECT   DISTINCT Essn
FROM     WORK_ON
WHERE    (Pno, Hours) IN  ( SELECT     Pno, Hours
                            FROM       WORK_ON
                            WHERE      Ssn='123456789' );
```

Câu truy vấn này sẽ lựa chọn Ssn của tất cả các thành viên có công việc giống nhau kết hợp trên một đề án có thành viên là 'John Smith' (của Ssn='123456789') được tiếp tục làm việc. Trong ví dụ này, toán tử so sánh IN là bộ con của các giá trị trong dấu ngoặc đơn (Pno, Hours) cho mỗi bộ trong WORK_ON với tập hợp của sự kết hợp tương thích các bộ sinh ra bởi bẫy câu truy vấn.

Thêm vào toán tử IN, một số toán tử so sánh có thể sử dụng để so sánh một giá trị *v* (tiêu biểu là tên thuộc tính) tới một tập hợp hoặc nhiều tập hợp *V* (tiêu biểu là bẫy câu truy vấn). Toán tử =ANY (hoặc =SOME) trả về TRUE nếu giá trị *v* bằng với một vài giá trị trong tập hợp *V* và do đó tương đương với IN. Từ khoá ANY và SOME có ý nghĩa giống nhau. Toán tử có thể kết hợp với ANY (hoặc SOME) bao gồm: >, >=, <, <= và <>. Từ khoá ALL cũng có thể kết hợp với mỗi toán tử này. Ví dụ, điều kiện so sánh (*v* > ALL *V*) trả về TRUE nếu giá trị *v* lớn hơn tất cả các giá trị trong tập hợp (hoặc nhiều tập hợp) *V*. Ví dụ câu truy vấn sau đây sẽ trả về tên của nhân viên nào có tiền lương lớn hơn tiền lương của tất cả các nhân viên bộ phận số 5:

```
SELECT    Lname, Fname
FROM      EMPLOYEE
WHERE     Salary > ALL    ( SELECT    Salary
                            FROM      EMPLOYEE
                            WHERE     Dno=5 );
```

Nói chung, chúng ta có thể có nhiều mức độ của bẫy các câu truy vấn. Chúng ta có thể có một lần lặp lại phần phía trước với sự nhập nhằng giữa các tên thuộc tính nếu các thuộc tính của tồn tại-một tên giống nhau ở quan hệ trong mệnh đề FROM của bẫy câu truy vấn. Minh hoạ khả năng nhập nhằng của các tên thuộc tính trong bẫy các câu truy vấn.

Query 16: Tìm tên của mỗi thành viên là những người có phụ thuộc tên giống nhau và giới tính giống nhau.

```
Q16:  SELECT    E.Fname, E.Lname
      FROM      EMPLOYEE AS E
      WHERE     E.Ssn IN  ( SELECT    Essn
                            FROM      DEPARTMENT
                            WHERE     E.Fname=Dependent_name
                                      AND E.Sex=Sex );
```

Trong bẫy Query 16, chúng ta phải chỉ rõ E.Sex bởi vì nó dựa vào thuộc tính Sex của EMPLOYEE từ câu truy vấn ở phía ngoài và DEPARTMENT cũng có thuộc tính Sex.

Nói chung cách thích hợp để tạo bộ các biến (bí danh) cho tất cả các bảng có liên quan trong câu truy vấn SQL là tránh khả năng bị lỗi và sự nhập nhằng.

6.5.3. Những truy vấn lồng nhau tương quan

Khi nào một điều kiện trong mệnh đề WHERE của bẫy câu truy vấn có liên quan đến một vài thuộc tính của một quan hệ rõ ràng trong câu truy vấn ở phía ngoài, hai câu truy vấn này nói là tương quan với nhau.

Nói chung, một câu truy vấn có thể viết với bẫy select-from-where và sử dụng toán tử so sánh = hoặc IN thì luôn luôn có thể biểu diễn một câu truy vấn đơn. Ví dụ, Q16 có thể viết như trong Q16A:

```
Q16A:   SELECT   E.Fname, E.Lname
        FROM     EMPLOYEE AS E, DEPARTMENT AS D
        WHERE    E.Ssn=D.Essn AND E.Sex=D.Sex
                 AND E.Fname=D.Department_name;
```

Đầu tiên SQL thực hiện đầy đủ trên SYSTEM R cùng có toán tử so sánh CONTAINS với việc so sánh hai tập hợp hoặc nhiều tập hợp. Query 13 minh hoạ sử dụng của toán tử CONTAINS.

Query 3: Tìm tên của mỗi thành viên là những người có công việc trong tất cả điều khiển đề án bởi bộ phận số 5.

```
Q3:  SELECT   E.Fname, E.Lname
     FROM     EMPLOYEE
     WHERE    ( (   SELECT   Pno
                    FROM     WORKS_ON
                    WHERE    Ssn=Essn)
                CONTAINS
                    SELECT   Pnumber
                    FROM     PROJECT
                    WHERE    Dnum=5 ) );
```

Trong Q3, bẫy thứ hai của câu truy vấn tìm các số đề án của tất cả các điều khiển đề án bởi bộ phận số 5. Chúng ta có thể sử dụng kỹ thuật khác bằng

hàm EXISTS để chỉ định kiểu dữ liệu này của các câu truy vấn, như mô tả trong mục 8.5.4 (xem mục 6.3.4 và mục 6.6.7 chương 6).

6.5.4. Hàm EXISTS và UNIQUE trong SQL

Hàm EXISTS trong SQL sử dụng kiểm tra có hay không kết quả của một bẫy câu truy vấn có tương quan với nhau là rỗng (không chứa các bộ) hoặc không. Kết quả của EXISTS là Boolean có giá trị TRUE hoặc FALSE. Chúng ta minh hoạ cách sử dụng của EXISTS và NOT EXISTS với một vài ví dụ. Đầu tiên, chúng ta đưa vào công thức Query 16 bằng một trong hai dạng để sử dụng EXISTS. Điều này được trình bày như Q16B:

```
Q16B: SELECT   E.Fname, E.Lname
      FROM     EMPLOYEE AS E
      WHERE    EXISTS ( SELECT   *
                        FROM     DEPARTMENT
                        WHERE    E.Ssn=Essn AND E.Sex=Sex
                                 AND
                                 E.Fname=Department_name);
```

EXISTS và NOT EXISTS thường được sử dụng trong sự liên kết có bẫy câu truy vấn tương quan với nhau. Trong Q16B, bẫy câu truy vấn có liên quan đến các thuộc tính Ssn, Fname và Sex của quan hệ EMPLOYEE từ câu truy vấn phía bên ngoài. Chúng ta có thể hiểu về Q16B như sau: mỗi bộ EMPLOYEE, đánh giá bẫy câu truy vấn bằng cách tìm tất cả các bộ DEPENMENT với Esn, Sex và Depenment_name giống như bộ EMPLOYEE; nếu ít nhất một bộ EXISTS trong kết quả của bẫy câu truy vấn thì chọn bộ EMPLOYEE. Nói chung, EXISTS(Q) trả về TRUE nếu có ít nhất một bộ có kết quả của bẫy câu truy vấn Q và trong trường hợp ngược lại nó trả về FALSE. Mặt khác, NOT EXISTS(Q) trả về TRUE nếu không có các bộ trong kết quả của bẫy câu truy vấn Q và trường hợp hợp ngược lại nó trả về FALSE. Tiếp theo, chúng ta minh hoạ cách sử dụng NOT EXISTS.

Query 6: Tìm các tên của các thành viên là những người không có phụ thuộc.

```
Q6:  SELECT   Fname, Lname
     FROM     EMPLOYEE
     WHERE    EXISTS ( SELECT   *
                       FROM     DEPARTMENT
                       WHERE    Ssn=Essn );
```

Trong Q6, dùng bẫy câu truy vấn có tương quan với nhau để tìm tất cả các bộ DEPENMENT có liên quan riêng với bộ EMPLOYEE. Nếu không tồn tại thì bộ EMPLOYEE sẽ được chọn.

Query 7: Liệt kê các tên của các nhà quản lý là những người có ít nhất một phụ thuộc.

```
Q7: SELECT   Fname, Lname
    FROM     EMPLOYEE
    WHERE    EXISTS  ( SELECT   *
                       FROM     DEPARTMENT
                       WHERE    Ssn=Essn );
             AND
             EXISTS  ( SELECT   *
                       FROM     DEPARTMENT
                       WHERE    Ssn=Mgr_ssn );
```

Có thể có tình trạng sử dụng EXISTS hoặc NOT EXISTS trong các hệ thống SQL, ở đây có hai lựa chọn. Đầu tiên sử dụng đúng-hiểu nguyên lý của sự biến đổi (S1 CONTAINS S2) tương đương với (S1 EXCEPT S2) là rỗng. Lựa chọn này được trình bày như Q3A.

```
Q3A: SELECT   Lname, Fname
     FROM     EMPLOYEE
     WHERE    NOT EXISTS  ( ( SELECT   Pnumber
                            FROM     PROJECT
                            WHERE    Dnum=5);
                            EXCEPT   ( SELECT   Pno
                                       FROM     WORKS_ON
                                       WHERE    Ssn=Essn) );
```

Lựa chọn thứ hai được trình bày như Q13B:

Q3B:

```
SELECT   Lname, Fname
FROM     EMPLOYEE
WHERE    NOT EXISTS  ( ( SELECT   *
```

```
                FROM        WORKS_ON B
                WHERE       B.Pno IN ( ( SELECT
                                Pnumber
                                        FROM
PROJECT
                                        WHERE
                                        Dnum=5);
                AND
                NOT EXISTS  ( SELECT    *
                                FROM    WORKS_ON
C
                                WHERE   C.Essn=Ssn
                                AND     C.Pno=B.Pno)
                            );
```

Có hàm SQL khác, UNIQUE(Q) trả về TRUE nếu không có các bộ trùng nhau trong kết quả của câu truy vấn Q; ngược lại nó trả về FALSE. Điều này có thể sử dụng kiểm tra được hay không kết quả của bẫy câu truy vấn là một tập hợp hoặc nhiều tập hợp.

6.5.5. Những tập hợp hiện và đổi tên các thuộc tính trong SQL

Chúng ta thấy nhiều cấu truy vấn với bẫy câu truy vấn trong mệnh đề WHERE. Nó cũng có thể thực hiện được bằng cách sử dụng một tập hợp *những giá trị rõ ràng* trong mệnh đề WHERE hơn một câu truy vấn được lồng vào. Như một tập hợp được đưa vào trong dấu ngoặc đơn trong SQL.

Query 17:

```
Q17:  SELECT    DISTINCT Essn
      FROM      WORKS_ON
      WHERE     Pno IN (1,2,3);
```

Trong SQL, nó có thể thực hiện được đổi tên thuộc tính bất kỳ xuất hiện trong kết quả của câu truy vấn bằng cách thêm vào sau điều kiện AS bởi đề nghị tên mới. Ví dụ, Q8A cho thấy làm câu truy vấn Q8 có thể có chuyển đổi không đáng kể để tìm họ của mỗi thành viên và anh ta hoặc cô ấy là giám sát, đổi tên trong kết quả tên thuộc tính như Employee_name và Supervisor_name. Các tên mới sẽ xuất hiện bằng tiêu đề cột trong kết quả câu truy vấn.

Q8A: SELECT E.Lname AS Employee_name, S.Lname AS Supervisor_name

```
FROM    EMPLOYEE AS E, EMPLOYEE AS S
WHERE   E.Super_ssn=S.Ssn;
```

6.5.6. Kết nối các quan hệ trong SQL và các kết nối ngoài

Tư tưởng cơ bản của việc kết nối các quan hệ là thực hiện việc sử dụng câu lệnh SQL để kết nối các bảng trong mệnh đề FROM của câu truy vấn. Cách xây dựng này là dễ lĩnh hội hơn cách lựa chọn trong một tập hợp các bảng qua mệnh đề WHERE. Chẳng hạn như trong Q1, cần truy xuất đến tên và địa chỉ của mỗi nhân viên làm việc cho bộ phận "research". Việc này có thể thực hiện đơn giản bằng việc kết giữa 2 bảng Employee và Department, từ đó có thể chọn các bộ và các thuộc tính mong muốn. Câu truy vấn Q1 có thể được viết lại như sau:

```
Q1A: SELECT   Fname, Lname, Address
     FROM     (EMPLOYEE JOIN, DEPARTMENT ON Dno=Dnumber)
     WHERE    Dname='Research';
```

Mệnh đề FROM trong query Q1A chứa một ràng buộc đơn giữa 2 table. Các thuộc tính như thuộc vào một bảng, trước hết là tất cả các thuộc tính của table Employee, tiếp theo là tất cả các thuộc tính của table Department. Có thể thực hiện nhiều kiểu kết nối khác nhau như: kết nối tự nhiên (natural join) , kết nối ngoài (outer join). Kết nối tự nhiên giữa hai quan hệ R và S là không có điều kiện được chỉ định. Điều kiện ngầm hiểu ở đây chính là mỗi cặp thuộc tính có tên giống nhau giữa hai quan hệ R và S. Mỗi cặp thuộc tính trong kết nối cho một và chỉ một quan hệ kết quả.

Nếu tên của các thuộc tính trong kết nối là không giống nhau trong các quan hệ cơ sở ta có thể thực hiện việc đổi tên các thuộc tính cần thiết thông qua từ khóa AS trong mệnh đề FROM. Câu truy vấn Q1B sau đây sẽ minh họa cách thực hiện kết nối này.

```
Q1B: SELECT   Fname, Lname, Address
     FROM     (EMPLOYEE NATURAL JOIN
              (DEPARTMENT AS DEFT (Dname, Dno, Mssn,Msdate)
     WHERE    Dname='Research';
```

Với ví dụ trên, quan hệ DEPARTMENT được đổi tên thành DEPT cùng với việc đổi tên các thuộc tính của nó thành: Dname, Dno (vì ta cần kết nối thuộc tính Dname của DEPARTMENT với Dnumber của EMPLOYEE), Mssn và Msdate. Khi đó, kết nối tự nhiên sẽ thực hiện kết nối giữa 2 thuộc tính giống nhau là: EMPLOYEE.Dno và DEPT.Dno.

Trong SQL, kết nối tự nhiên là kết nối mặc định, ở đó một bộ sẽ được đưa vào quan hệ kết quả nếu nó tồn tại trong một quan hệ khác qua phép kết

nối và các bộ không thỏa sẽ không được đưa vào bảng kết quả kết nối. Tuy nhiên, đôi khi ta cũng cần giữ lại những thông tin này bằng cách cho phép những dòng không thoả mãn điều kiện nối có mặt trong kết quả của phép nối. Để làm điều này, ta có thể sử dụng phép nối ngoài (OUTER JOIN).

Phép nối ngoài cũng được chỉ định ngay trong mệnh đề FROM theo cú pháp:

FROM <TableName1> LEFT|RIGHT|FULL [OUTER] JOIN <TableName2>
ON <Condition>

- Phép nối ngoài trái (LEFT OUTER JOIN)

Phép nối ngoài trái hiển thị trong kết quả truy vấn những bộ thỏa điều kiện kết nối và bổ sung những bộ không thỏa điều kiện kết nối của bảng bên trái (các thuộc tính tương ứng còn lại của các bộ không thỏa của bảng bên phải mang giá trị Null).

- Phép nối ngoài phải (RIGHT OUTER JOIN)

Phép nối ngoài phải hiển thị trong kết quả truy vấn những bộ thỏa điều kiện kết nối và bổ sung những bộ không thỏa điều kiện kết nối của bảng bên phải (các thuộc tính còn lại của các bộ không thỏa của bảng bên trái mang giá trị Null).

- Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Phép nối đầy đủ, hiển thị trong kết quả truy vấn cả những bộ dữ liệu không thỏa điều kiện kết nối của cả hai bảng tham gia kết nối.

Câu truy vấn sau đây cho phép in ra các nhân viên làm công tác quản lý (Supper_ssn= Null) trong bảng EMPLOYEE:

Q8B: SELECT E.Lname AS Employee_name,
S.Lname AS Supervisor_name
FROM (EMPLOYEE AS E LEFT OUTER JOIN, EMPLOYEE AS S
ON E.Super_ssn=S.Ssn);

Một kết nối các bảng có thể tiếp tục kết nối đến một bảng khác. Câu truy vấn Q2 có thể được viết lại thành Q2A theo cách này.

Q2A: SELECT Pnumber, Dnum, Lname, Adress, Bdate
FROM ((PROJECT JOIN DEPARTMENT ON Dnum=Dnumber)
JOIN EMPLOYEE ON Mgr_ssn=Ssn)
WHERE Plocation='Stafford';

Phép kết nối ngoài còn một số cách thể hiện cú pháp khác thông qua các toán tử sánh như: +=, =+, +=+ là tương ứng với các phép nối trái, phải và đầy đủ của phép kết nối ngoài đã xét. Truy vấn Q8C sau đây minh họa một dạng cú pháp theo kiểu này.

Q8C: SELECT E.Lname, S.Lname

FROM EMPLOYEE E, EMPLOYEE S

WHERE E.Super_ssn=S.Ssn;

6.5.7. Một số hàm gộp trong SQL

Vấn đề kết nhóm và tổng hợp thông tin là khá cần thiết trong các ứng dụng cơ sở dữ liệu. SQL rất chú trọng đến các chức năng này. SQL có xây dựng một số hàm gộp cơ bản là: COUNT, SUM, MAX, MIN và AVG. Chúng ta minh hoạ cách sử dụng các hàm này với ví dụ câu truy vấn:

Query 19: Tìm tổng tiền lương của tất cả các thành viên, tiền lương cao nhất, tiền lương thấp nhất và tiền lương trung bình.

Q19: SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)

FROM EMPLOYEE;

Nếu chúng ta muốn có được những giá trị hàm về các thành viên của bộ phận cụ thể chọn bộ phận 'Reseach', chúng ta viết Query 20:

Query 20: Tìm tổng tiền lương của tất cả các thành viên thuộc bộ phận 'Reseach', cùng với tiền lương cao nhất, tiền lương thấp nhất và tiền lương trung bình trong bộ phận này.

Q20: SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)

FROM (EMPLOYEE JOIN, DEPARTMENT ON Dno=Dnumber)

WHERE Dname='Research';

Query 21 và 22: Tìm tổng số thành viên trong công ty (Q21) và số thành viên trong bộ phận 'Research' (Q22).

Q21: SELECT COUNT (*)

FROM EMPLOYEE;

Q22: SELECT COUNT (*)

FROM EMPLOYEE, DEPARTMENT;

WHERE Dno=Dnumber AND Dname='Research';

Đến đây dấu (*) là các dòng (các bộ), như thế COUNT (*) trả về số dòng trong kết quả của câu truy vấn.

Query 23: Đếm số giá trị tiền lương phân biệt trong cơ sở dữ liệu.

Q23: SELECT COUNT (DISTINCT Salary)

FROM EMPLOYEE;

Nếu chúng ta viết COUNT(Salary) thay vì viết COUNT (DISTINCT Salary) trong Q23 thì các giá trị trùng nhau sẽ không loại trừ.

Ví dụ, tìm tên của tất cả các thành viên là những người có hai hoặc nhiều bộ phận Query 5), chúng ta có thể viết sau đây:

```
Q5: SELECT  Lname, Fname
    FROM    EMPLOYEE
    WHERE   ( SELECT  COUNT (*)
              FROM    DEPARTMENT
              WHERE   Ssn=Essn ) >= 2;
```

6.5.8. Các mệnh đề GROUP BY và HAVING

Trong SQL, 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 .

Dưới đây là một số ví dụ minh hoạ mệnh đề GROUP BY và cách sử dụng một số hàm gộp để thống kê dữ liệu.

Query 24: Cho mỗi bộ phận, tìm số bộ phận, số thành viên trong bộ phận và trung bình tiền lương của họ.

```
Q24: SELECT     Dno, COUNT (*), AVG (Salary)
     FROM       EMPLOYEE;
     GROUP BY   Dno;
```

Trong Q24, các bộ EMPLOYEE là sự phân chia vào các nhóm - mỗi nhóm có giá trị trùng nhau cho thuộc tính nhóm Dno. Hàm COUNT và AVG áp dụng cho các bộ của mỗi nhóm.

Ví dụ: Nếu bảng EMPLOYEE có vài bộ NULL thuộc nhóm thuộc tính Dno thì sẽ cóviệc tách nhóm ở các bộ này trong kết quả của Q24.

Query 25: Trong mỗi đề án, tìm số đề án, tên đề án và số thành viên là những người có công việc trong dự án.

```
Q25: SELECT     Pnumber, Pname, COUNT (*)
```

```
FROM        PROJECT; WORKS_ON
WHERE       Pnumber=Pno
GROUP BY    Pnumber, Pname;
```

Ví dụ: Giả sử chúng ta muốn tạo Query 25 sao cho các đề án có nhiều hơn hai thành viên xuất hiện trong kết quả. SQL cung cấp mệnh đề HAVING, đây là mục đích để làm xuất hiện sự kết hợp với mệnh đề GROUP BY. HAVING đưa ra điều kiện trên nhóm của việc kết hợp các bộ với mỗi giá trị của các thuộc tính nhóm. Chỉ có các nhóm thoả mãn điều kiện tìm trong kết quả của câu truy vấn, điều này được minh hoạ bởi Query 26.

Query 26: Trong mỗi đề án có nhiều hơn hai thành viên làm việc, tìm số đề án, tên đề án và số thành viên là những người có làm việc trong đề án.

```
Q26:  SELECT      Pnumber, Pname, COUNT (*)
      FROM        PROJECT; WORKS_ON
      WHERE       Pnumber=Pno
      GROUP BY    Pnumber, Pname;
      HAVING      COUNT (*) > 2
```

Query 27: Trong mỗi đề án, tìm số đề án, tên đề án và số thành viên từ bộ phận 5 là những người có công việc trong đề án.

```
Q27:  SELECT      Pnumber, Pname, COUNT (*)
      FROM        PROJECT; WORKS_ON, EMPLOYEE
      WHERE       Pnumber=Pno AND Ssn=Essn AND Dno=5
      GROUP BY    Pnumber, Pname;
```

Kết quả của GROUP BY và HAVING: (a) Q24, (b) Q26.

(a)

Fname	Minit	Lname	Ssn	. . .	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	. . .	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennier	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Dno	Count (*)	AVG (Salary)
5	4	33250
4	3	31000
1	1	55000

(b)

Pname	Pnumber	. . .	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	. . .	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0

Các nhóm này không được lựa chọn bởi điều kiện HAVING

Sau khi áp dụng mệnh đề WHERE nhưng trước khi áp dụng HAVING.

Pname	Pnumber	. . .	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10	. . .	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Giả sử chúng ta viết câu truy vấn không đúng sau đây:

```
SELECT      Dname, COUNT (*)
FROM        DEPARTMENT, EMPLOYEE
WHERE       Dnumber=Dno AND Salary>40000
GROUP BY    Dname;
HAVING      COUNT (*) > 5;
```

Điều này không đúng bởi vì nó sẽ chọn chỉ các bộ phận có hơn 5 thành viên là mỗi người được lĩnh hơn $40,000. Một cách khác viếat câu truy vấn

này đúng cách thức bằng cách sử dụng bẫy câu truy vấn, như trình bày trong Query 28.

Query 28: Trong mỗi bộ phận có hơn 5 thành viên, tìm số bộ phận và số thành viên trong bộ phân đó là những người làm hơn $40,000.

```
Q28: SELECT     Dname, COUNT (*)
     FROM       DEPARTMENT, EMPLOYEE
     WHERE      Dnumber=Dno AND Salary>40000 AND
                Dno IN (  SELECT     Dno
                          FROM       EMPLOYEE
                          GROUP BY   Dno
                          HAVING     COUNT (*) > 5)
     GROUP BY   Dnumber;
```

6.5.9. Thảo luận và kết luận của truy vấn SQL

Xây dựng câu truy vấn trong SQL gồm có sáu mệnh đề nhưng bắt buộc phải có hai mệnh đề SELECT và FROM. Các mệnh đề đó có chỉ định theo thứ tự sau đây nhưng với các mệnh đề nằm trong dấu ngoặc [...] là không bắt buộc.

SELECT <Thuộc tính và danh sách chọn>

FROM <Danh sách bảng>

[WHERE <Điều kiện>]

[GROUP BY < Thuộc tính nhóm>]

[HAVING <Điều kiện nhóm>]

[ORDER BY <Danh sách thuộc tính>];

6.6. Các lệnh INSERT, DELETE và UPDATE

Trong SQL, có ba lệnh có thể sử dụng để sửa đổi cơ sở dữ liệu: INSERT, DELETE và UPDATE. Chúng ta lần lượt tìm hiểu 3 lệnh này.

6.6.1. Lệnh chèn (INSERT)

Dạng đơn giản nhất của nó, chèn(INSERT) được sử dụng để thêm một bộ dữ liệu đơn vào một quan hệ. Chúng ta phải chỉ rõ tên quan hệ và một danh sách những giá trị cho bộ dữ liệu. Những giá trị cần phải được liệt kê trong cùng thứ tự, trong đó những thuộc tính tương ứng được chỉ rõ bên trong lệnh tạo bảng. Cho ví dụ: thêm một bộ dữ liệu mới vào quan hệ EMPLOYEE, chúng ta có thể sử dụng U1:

```
U1: INSERT TABLE     EMPLOYEE
      VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30',
```

’98 Oak Forest, Katy, TX’, ‘M’, 37000, ‘653298653’,4);

Một dạng thứ hai của lệnh INSERT, cho phép người dùng chỉ rõ tên thuộc tính tương ứng trong lệnh, nếu một quan hệ có nhiều thuộc tính nhưng chỉ một ít thuộc tính đó được gán những giá trị trong bộ dữ liệu mới. 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). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

Những thuộc tính được phép nhận giá trị NULL hoặc giá trị DEFAULT có thể được để lại ở ngoài. Cho ví dụ, để vào một bộ dữ liệu mới cho EMPLOYEE mà chúng ta biết duy nhất Fname, Lname, Dno và thuộc tính Ssn, chúng ta có thể sử dụng U1A:

U1A: INSERT INTO EMPLOYEE(Fname, Lname, Dno,Ssn)
VALUES (‘Richard’, ‘Marini’, 4,‘653298653’);

Những thuộc tính không được xác định trong U1A là tập hợp DEFAULT hoặc NULL, và những gía trị được liệt kê theo thứ tự những thuộc tính được liệt kê trong lệnh Insert. Có thể chèn vào trong một quan hệ nhiều bộ dữ liệu cách nhau bởi dấu phẩy trong lệnh chèn. Những giá trị thuộc tính chèn vào cho mỗi bộ dữ liệu được bao trong dấu ngoặc.

Một DBMS mà hoàn toàn thực hiện SQL-99 cần phải hổ trợ và bắt buộc tất cả các ràng buộc toàn vẹn mà có thể chỉ rõ trong DDL. Tuy nhiên, DBMS nào đó không hợp nhất tất cả các sự ràng buộc để bảo trì hiệu quả DBMS và sự phức tạp của tất cả các sự ràng buộc. Nếu một hệ thống không hỗ trợ sự ràng buộc, sự toàn vẹn nào đó thì người dùng hoặc người lập trình phải tự đưa vào ràng buộc. Cho ví dụ, nếu chúng ta phát hành lệnh trong U2 trên cơ sở dữ liệu một DBMS không hỗ trợ toàn vẹn sẽ không lồng vào mặc dù bộ dữ liệu của một bộ phận tồn tại trong cơ sở dữ liệu với Dnumber = 2. Đó là trách nhiệm của người dùng để kiểm tra bất kỳ sự ràng buộc nào vi phạm khi có sự kiểm tra được thực hiện bởi DBMS. Tuy nhiên, DBMS phải thực hiện kiểm tra bắt buộc tất cả sự ràng buộc toàn vẹn mà SQL hỗ trợ. Một DBMS bắt buộc NOT NULL sẽ không chấp nhận một lệnh chèn vì các thuộc tính được khai báo NOT NULL không được nhận giá trị. Ví dụ câu truy vấn U2A sau là không được chấp nhân vì không có giá trị Ssn nào được cung cấp.

U2: INSERT INTO EMPLOYEE(Fname, Lname, Dno,Ssn)
VALUES (‘Robert’, ‘Hatcher’, ‘980760540’,2);
(U2 được loại bỏ nếu sự kiểm tra toàn vẹn do DBMS cung cấp)

U2A: INSERT INTO EMPLOYEE(Fname, Lname, Dno)
VALUES (‘Robert’, ‘Hatcher’, 5);
(U2A được loại bỏ nếu sự kiểm tra NOT NULL do DBMS cung cấp)

Một biến đổi khác của lệnh chèn là chèn một số bộ vào một quan hệ cùng với việc tạo ra một quan hệ mới trong kết quả truy vấn. Ví dụ như tạo một table có tên, mã số của nhân viên và tổng lương của mỗi bộ phận chúng ta có thể thực hiện như các truy vấn U3A và U3B như sau:

```
U3A:  CREATE TABLE    DEPTS_INFO
      ( Dept_name     VARCHAR(15),
      No_of_emps      INTEGER,
      Total_sal       INTEGER);

U3B:  INSERT INTO  DEPTS_INFO ( Dept_ name, No_of_emps, Total_sal)
      SELECT  Dname, COUNT(*), SUM(Salary)
      FROM   (DEPARTMENT JOIN EMPLOYEE ON Dnumber=Dno)
      GROUP BY    Dname;
```

Một bảng DEPTS_INFO được tạo ra bởi U3A và được tải thông tin tóm lược từ cơ sở dữ liệu bởi câu hỏi bên trong U3B. Chúng ta có thể hỏi DEPTS_INFO như mọi quan hệ khác; khi chúng ta không có nhu cầu, chúng ta loại bỏ nó bởi việc sử dụng bảng DEPTS ra lệnh. Chú ý rằng bảng DEPTS_INFO có thể không hợp thời; nếu chúng ta cập nhật Ban hoặc Người làm thuê là những quan hệ sau phát biểu U3B thì thông tin trong DEPTS_INFO trở thành lỗi thời. Chúng ta phải tạo ra theo ý định (Mục 6.8) để có một bảng hợp thời.

6.6.2. Lệnh Xóa (DELETE)

Cú pháp chung để thực hiện lệnh xóa:

```
DELETE FROM <TableName>
      [WHERE <Condition>]
```

Lệnh xóa những bộ dữ liệu từ một quan hệ. Về cú pháp câu lệnh xóa được sử dụng tương tự như câu lệnh truy vấn, để lựa chọn những bộ dữ liệu cần xóa. Tuy nhiên, lệnh xóa có thể lan truyền tới những bộ dữ liệu trong những quan hệ khác có liên quan được xác định từ những sự ràng buộc toàn vẹn (ở mục 6.2.2). Mệnh đề cần chỉ rõ tất cả các bộ dữ liệu trong quan hệ cần xóa. Ví dụ: U4A thực hiện xóa các nhân viên có Lname = ‘Brown’ từ bảng EMPLOYEE; U4B để xóa các nhân viên có Ssn= ‘123456789’; U4C thực hiện xóa các nhân viên có Dname=‘Research’ , để thực hiện yêu cầu này ta cần tham chiếu đến bảng DEPARTMENT để xác định giá trị Dnumber tương ứng và từ đó xác định giá trị Dno trong bảng EMPLOYEE, giá trị Dno chính là điều kiện cần xóa;

```
U4A:  DELETE FROM   EMPLOYEE
      WHERE         Lname = ‘Brown’;
U4B:  DELETE FROM   EMPLOYEE
      WHERE         Ssn = ‘123456789’;
U4C:  DELETE FROM   EMPLOYEE
```

```
        WHERE       Dno IN (SELECT  Dnumber
                            FROM    DEPARTMENT
                            WHERE   Dname = 'Research');
U4D:    DELETE FROM EMPLOYEE;
```

6.6.3. Lệnh cập nhật (UPDATE)

Lệnh cập nhật được sử dụng để sửa đổi những giá trị thuộc tính của một hoặc nhiều bộ dữ liệu. Như trong lệnh xóa, một mệnh đề cập nhật ra lệnh lựa chọn những bộ dữ liệu sẽ được sửa đổi từ một quan hệ đơn. Tuy nhiên, việc cập nhật một giá trị khóa sơ cấp có thể sinh ra những giá trị khóa ngoài của những bộ dữ liệu trong những quan hệ khác, nếu có liên quan thì hoạt động được chỉ rõ trong những ràng buộc toàn vẹn của DDL (mục 6.2.2). Một mệnh đề tập hợp bổ sung trong lệnh cập nhật chỉ rõ những thuộc tính sẽ được sửa đổi và những giá trị mới của nó. Cho ví dụ, thay đổi vị trí số ban của dự án ghi số 10 tới 'Bellaire' và 5 tương ứng, chúng ta sử dụng U5:

```
U5:     UPDATE      PROJECT
        SET         Plocation = 'Bellaire', Dnum = 5
        WHERE       Pnumber = 10;
```

Vài bộ dữ liệu có thể được sửa đổi với một lệnh cập nhật đơn. Ví dụ, cho tất cả những người làm thuê trong ban 'Nghiên cứu' được nâng lương 11% (ở U6). Trong yêu cầu này, khi sửa đổi tiền lương thì những giá trị phụ thuộc vào tiền lương nguyên bảng được đánh giá trong mỗi bộ dữ liệu, như vậy có 2 tham chiếu đến thuộc tính tiền lương. Trong mệnh đề tập hợp, sự tham chiếu tới thuộc tính tiền lương ở bên phải là giá trị tiền lương cũ trước cải biến, và bên trái tham chiếu tới giá trị tiền lương mới sau khi cải biến:

```
U6:     UPDATE      EMPLOYEE
        SET         Salary = Salary * 6.1
        WHERE       Dno IN (SELECT      Dnumber
                            FROM        DEPARMENT
                            WHERE       Dname = 'Research');
```

Có thể chỉ rõ NULL hoặc DEFAULT các giá trị thuộc tính mới. Chú ý rằng mỗi lệnh cập nhật chỉ tham chiếu tới quan hệ đơn. Muốn sửa đổi nhiều quan hệ, chúng ta phải dùng vài lệnh cập nhật.

6.7. Chỉ định các ràng buộc ASSERTION và TRIGGERS

TRIGGER: Một trigger là ràng buộc trong đó có chứa một tập các câu lệnh SQL và tập các câu lệnh này sẽ được thực thi khi trigger được gọi. Điểm khác biệt giữa thủ tục lưu trữ và trigger là: các thủ tục lưu trữ được thực thi khi người sử dụng có lời gọi đến chúng còn các trigger lại được tự động thực hiện khi xảy ra những giao tác làm thay đổi dữ liệu trong các bảng.

Mỗi một trigger được tạo ra và gắn liền với một bảng nào đó trong cơ sở dữ liệu. Khi dữ liệu trong bảng bị thay đổi (tức là khi bảng chịu tác động của các câu lệnh INSERT, UPDATE hay DELETE) thì trigger sẽ được tự đông kích hoạt.

Câu lệnh CREATE TRIGGER được sử dụng để đinh nghĩa trigger và có cú pháp như sau:

CREATE TRIGGER *tên_trigger*
ON *tên_bảng*
FOR {[INSERT][,][UPDATE][,][DELETE]}
AS
[IF UPDATE(*tên_cột*)
[AND UPDATE(*tên_cột*)|OR UPDATE(*tên_cột*)]
...]
các_câu_lệnh_của_trigger

Ví dụ:

```
CREATE TRIGGER EmployeeDelete
ON Employee
FOR DELETE
AS
DELETE FROM Works_On
WHERE works.Essn=Deleted.ssn
```

Với Trigger vừa tạo thì sau khi ta thực hiện câu lênh:

```
DELETE FROM Employee
Where ssn='123456789'
```

Các mẩu tin trong bảng Works_on có Essn='123456789' sẽ bị xóa.

Trong câu lệnh CREATE TRIGGER ở ví dụ trên, sau mệnh đề ON là tên của bảng mà trigger cần tạo sẽ tác động đến. Mệnh đề tiếp theo chỉ định câu lệnh sẽ kích hoạt trigger (FOR DELETE). Ngoài DELETE, ta còn có thể chỉ định UPDATE hoặc INSERT cho mệnh đề này, hoặc có thể kết hợp chúng lại với nhau. Phần thân của trigger nằm sau từ khoá AS bao gồm các câu lệnh mà trigger sẽ thực thi khi được kích hoạt.

Chuẩn SQL định nghĩa hai bảng logic INSERTED và DELETED để sử dụng trong các trigger. Cấu trúc của hai bảng này tương tự như cấu trúc của bảng mà trigger tác động. Dữ liệu trong hai bảng này tuỳ thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger; cụ thể trong các trường hợp sau:

- Khi câu lệnh DELETE được thực thi trên bảng, các dòng dữ liệu bị xoá sẽ được sao chép vào trong bảng DELETED. Bảng INSERTED trong trường hợp này không có dữ liệu.

- Dữ liệu trong bảng INSERTED sẽ là dòng dữ liệu được bổ sung vào bảng gây nên sự kích hoạt đối với trigger bằng câu lệnh INSERT. Bảng DELETED trong trường hợp này không có dữ liệu.

- Khi câu lệnh UPDATE được thực thi trên bảng, các dòng dữ liệu cũ chịu sự tác động của câu lệnh sẽ được sao chép vào bảng DELETED, còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật.

ASSERTION: Trong SQL, người dùng có thể chỉ định những sự ràng buộc chung (không rơi vào các ràng buộc đã xét) đó là ràng buộc ASSERTION. Để tạo ràng buộc loại này ta dùng lệnh CREATE ASSERTION. Cho ví dụ, chỉ rõ sự ràng buộc mà tiền lương của một nhân viên không được lớn hơn tiền lương của giám đốc của anh ta, chúng ta có thể thực hiện sau:

```
CREATE ASSERTION SALARY_CONSTRAINT
CHECK (NOT EXISTS ( SELECT *
                    FROM    EMPLOYEE E, EMPLOYEE M, DEPARTMENT D
                    WHERE   E.Salary > M.Salary
                            AND E.Dno = D.Dnumber
                            AND D.Mgr_ssn = M.Ssn ));
```

6.8. Khung nhìn trong SQL

Phần này chúng ta tập trung giới thiệu về khung nhìn trong SQL, cách xác định một khung nhìn cũng như cách cập nhật và thực hiện một khung nhìn trong một hệ quản trị CSDL.

6.8.1. Khái niệm về khung nhìn trong SQL

Một khung nhìn trong SQL là một table đơn lẻ nhận được từ các table khác nhau hoặc từ các khung nhìn khác nhau trong một DBMS thông qua một câu truy vấn. Một khung nhìn không được xem như là một cấu trúc lưu trữ dữ liệu tồn tại trong CSDL, và vì vậy khung nhìn có thể xem như một table ảo được kết xuất từ các bộ thuộc các table trong một CSDL.

Chúng ta có thể xem khung nhìn như một cách thuận lợi để quan sát các bảng khi các bảng có sự thay đổi thường xuyên. Chẳng hạn như, với ví dụ về lược đồ cơ sở dữ liệu COMPANY đã được xét ở các mục trước, chúng ta có thể thường xuyên lấy ra các thông tin như tên nhân viên và tên đề án mà nhân viên đang thực hiện vào bất kỳ thời điểm nào và được lưu trữ dưới dạng một view.

6.8.2. Tạo View

Để tạo view ta sử dụng câu lệnh CREATE VIEW, các thành phần kèm theo là tên view, danh sách các thuộc tính (fiedList) và câu lệnh truy vấn để xác định nội dung cho view.

Cú pháp chung để tạo view:

```
CREATE VIEW viewName[(fiedList)]
```

AS SELECT ...

Ví dụ: -V1: *CREATE VIEW WORK_ON1*
AS SELECT Fname, Lname, Pname, Hours
FROM EMPLOYEE, PROJECT, WORKS_ON
WHERE Ssn = Essn AND Pno=Pnumber;

-V2: *CREATE VIEW DEPT_INFO(Dept_name, No_of_emps, Total_sal)*
AS SELECT Dname, COUNT(*), SUM (salary)
FROM EMPLOYEE, DEPARTMENT
WHERE Dnumber= Dno
GROUP BY Dname;

Với V1 chúng ta không chỉ định các thuộc tính cho view, trong trường hợp này WORKS_ON1 tên các thuộc tính của view nhận được từ các thuộc tínhcủa các quan hệ *EMPLOYEE, PROJECT và WORKS_ON*. Với view V2, có xác định cụ thể tên các thuộc tính mới cho view DEPT_INFO, các thuộc tính này có tương ứng 1-1 với các thuộc tính từ kết quả của mệnh đề truy vấn SELECT.

Ngoài ra chúng ta có thể thực hiện truy vấn trên một view, trường hợp này chúng ta thực hiện như truy vấn trên table. Ví dụ sau cho phép truy xuất đến các nhân viên thuộc đề án 'ProjectX' dựa trên view WORKS_ON1

QV1: SELECT Fname, Lname
FROM WORKS_ON1
WHERE Pname= 'ProjectX'

Như vậy, chúng ta cũng có thể thực hiện các truy vấn trên nhiều table thông qua view với điều kiện có sự kết nối giữa các table. Điều thuận lợi chính là thao tác trên view là đơn giản hơn, ngoài ra view cũng thường được sử dụng trong kỹ thuật bảo mật và xác thực người dùng.

Một số lưu ý khi tạo view:

- Tên khung nhìn và tên các thuộc tính của khung nhìn được đặt tương tự như bảng, nghĩa là cũng phải tuân theo qui tắc định danh.

- Không thể qui định ràng buộc và tạo chỉ mục cho khung nhìn

- Phải đặt tên cho các cột trong view trong các trường hợp sau:

+ Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức là không phải là một tên cột trong bảng cơ sở) và cột đó không được đặt tiêu đề.

+ Tồn tại hai cột trong kết quả của câu lệnh SELECT có cùng tiêu đề cột.

Khi không cần thiết sử dụng một view chúng ta có thể xóa view bằng lệnh Drop view:

Ví dụ: Xóa view WORKS_ON1
DROP VIEW WORKS_ON1

6.8.3. Thực thi và cập nhật View

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.

Chúng ta có thể thực hiện thao tác bổ sung, cập nhật và xoá, một khung nhìn với điều kiện câu lệnh SELECT trong định nghĩa khung nhìn không được sử dụng từ khoá DISTINCT, TOP, GROUP BY và UNION; trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp. Ngoài ra 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: Với view WORKS_ON1, ta cần cập nhật lại thuộc tính Pname cho các bộ của nhân viên ‘John Smith’ , ta có thể thực hiện truy vấn sau:

```
UV1: UPDATE  WORKS_ON1
     SET  Pname = ‘ProductY’
     WHERE  Lname = ‘Smith’  AND  Fname = ‘John’
                 AND  Pname = ‘ProductX’;
```

Khi thực hiện truy vấn UV1 này có thể làm thay đổi dữ liệu trên các quan hệ có liên quan đến view. Có hai thay đổi có thể xảy ra là:

```
(a):  UPDATE  WORKS_ON
      SET  Pno = (SELECT  Pnumber
                  FROM  PROJECT
                  WHERE  Pname = ‘ProductY’)
      WHERE  Essn  IN  (SELECT  Ssn
                        FROM  EMPLOYEE
                        WHERE  Lname= ‘Smith’  AND
Fname=’John’)
                AND
                 Pno = (SELECT  Pnumber
                        FROM  PROJECT
                        WHERE  Pname = ‘ProductX’);
(b):  UPDATE  PROJECT  SET  Pname = ‘ProductY’
      WHERE  Pname = ‘ProductX’;
```

Vì khi tên sản phẩm bị thay đổi dẫn đến mã sản phẩm Pnumber cũng bị thay đổi do đó theo (a) bảng WORKS_ON được cập nhật lại giá trị Pno bằng Pnumber của sản phẩm ‘ProductY’ tại những bộ thỏa điều kiện: Pname của bảng PROJECT là ‘ProductX’ và do ông John Smith quản lý. Theo (b) tên sản phẩm (Pname) của bảng PROJECT được thay bằng ‘ProductY’ tại những Pname= ‘ProductX’.

Dưới đây là một số lưu khi cập nhật trên view:

- Một view được định nghĩa với một bảng đơn sẽ có thể cập nhật nếu các thuộc tính của view có chứa khóa chính của bảng cơ sở.

- Các view được định nghĩa trên nhiều bảng có sử dụng kết nối thì thường không nên cập nhật.

- Các view được định nghĩa có dùng các lệnh Group hoặc các hàm kết hợp thì không nên cập nhật.

6.9. Một số tính năng khác của SQL

Dưới đây là một số tính năng quan trọng khác nhưng không được thảo luận chi tiết trong chương này. Các vấn đề này có thể tìm hiểu trong những chương tiếp theo.

- SQL được sự hỗ trợ nhiều từ một số ngôn ngữ lập trình khác, nghĩa là từ các ngôn ngữ lập trình này ta có thể thực hiện xử lý dữ liệu bằng câu lệnh SQL. Các kỹ thuật thường dùng như: Kỹ thuật nhúng SQL, SQL/CLI (call language interface), ODBC, SQL/PSM. Trong chương 2 chúng ta sẽ tìm hiểu các kỹ thuật này.

- SQL có các câu lệnh để điều khiển các giao tác, các lệnh này thường được sử dụng trong xử lý CSDL với các điều khiển tranh chấp và phục hồi dữ liệu.

- SQL cho phép tạo các trigger, kỹ thuật này thường dùng trong việc kích hoạt và cập nhật tự động CSDL.

- SQL tương thích với mô hình hướng đối tượng với nhiều tính năng hữu ích. SQL có thể sử dụng để cải tiến một mô hình quan hệ thành mô hình hướng đối tượng-quan hệ.

- SQL và CSDL quan hệ có thể tương tác lẫn nhau với công nghệ như XML và OLAP.

BÀI TẬP NHẬP MÔN CƠ SỞ DỮ LIỆU
SƠ ĐỒ THỰC THỂ MỐI QUAN HỆ - MÔ HÌNH QUAN HỆ

1. Một cửa hàng cần phải tổ chức CSDL nhằm hỗ trợ công tác quản lý bán hàng. Ngoài việc phải quản lý các thông tin lên quan đến các mặt hàng như mã hàng, tên hàng, số lượng, giá hàng,… CSDL này còn phải hỗ trợ cho việc quản lý nhập và xuất (bán) hàng. Cho biết một số thông tin liên quan đến hoạt động của cửa hàng này như sau:

Hàng hoá trong của hàng được lấy từ các nguồn hàng do nhà cung cấp cung cấp. Mỗi đợt nhập hàng như vậy phải được ghi vào các phiếu nhập có mẫu như sau:

PHIẾU NHẬP HÀNG

Số phiếu:..... Ngày nhập hàng ...../...../.....

Người cung cấp:.....

Địa chỉ:.....

DANH MỤC HÀNG NHẬP

Mã hàng	Tên hàng	Số lượng	Giá nhập	Thành tiền
....	.....	.....	.....	.....
	Tổng cộng:	.....	.....	.....

Khi khách hàng mua hàng, cửa hàng sẽ tiến hành lập hoá đơn bán hàng cho khách. Việc thành toán tiền của khách sẽ dựa vào các hoá đơn này. Các hoá đơn của hàng sử dụng có mẫu như sau:

HOÁ ĐƠN BÁN HÀNG

Số hoá đơn:..... Ngày bán ...../...../.....

Tên người mua hàng:.....

Địa chỉ:.....

DANH MỤC HÀNG ĐƯỢC BÁN

Mã hàng	Tên hàng	Số lượng	Giá bán	Thành tiền
....	.....	.....	.....	.....
	Tổng cộng:	.....	.....	.....

a. Vẽ sơ đồ thực thể - mối quan hệ cho cơ sở dữ liệu trên.

b. Chuyển sơ đồ quan hệ - thực thể trên sang mô hình dữ liệu quan hệ.

c. Chuyển sơ đồ thực thể - mối quan hệ trên sang mô hình dữ liệu hướng đối tượng.

2. Để hỗ trợ công tác quản lý đào tạo trong một trường đại học, người ta tổ chức một cơ sở dữ liệu trong đó quản lý các thông tin về sinh viên, các lớp học, các khoa trong trường, các khoa - bộ môn trong trường, các học phần và giáo viên giảng dạy trong trường nhằm cho biết các thông tin sau:

- Sinh viên học ở lớp nào.
- Lớp học thuộc sự quản lý của khoa-bộ môn nào trong trường.
- Giáo viên làm việc ở khoa-bộ môn nào trong trường.
- Một lớp vào một học kỳ của một năm học được học những môn học nào và do giáo viên nào giảng dạy.
- Sinh viên học một môn học với kết quả điểm thi ra sao (cả điểm thi lần 1 lẫn điểm thi lần 2).

a. Hãy vẽ sơ đồ thực thể-mối quan hệ cho cơ sở dữ liệu trên.

b. Biểu diễn sơ đồ thực thể-mối quan hệ trên trong mô hình dữ liệu quan hệ, mô hình dữ liệu hướng đối tượng.

c. Nếu bạn là người được lựa chọn để triển khai hệ thống trên, hãy cho biết những khuyến cáo của bạn đối với hệ thống cần được xây dựng.

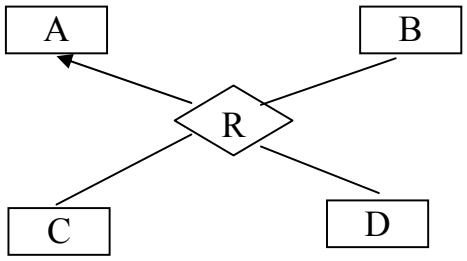
3. Cơ sở dữ liệu VẬT TƯ được sử dụng để quản lý công tác phân phối vật tư cho các phân xưởng trong nhà máy. Vật tư được lưu trữ trong các kho của nhà

máy. Khi một phân xưởng nào đó cần vật tư thì gởi đơn yêu cầu cung cấp vật tư. Nhà kho sẽ căn cứ vào đơn yêu cầu để chuyển vật tư cho phân xưởng. Khi chuyển vật tư cho phân xưởng phải lưu trữ thông tin về việc chuyển giao vật tư trong các phiếu xuất kho để tiện cho công tác thống kê sau này.

a. Hãy biểu diễn sơ đồ thực thể-mối quan hệ tương ứng với cơ sở dữ liệu trên.

b. Chuyển sơ đồ thực thể - mối quan hệ ở trên sang mô hình dữ liệu quan hệ, mô hình dữ liệu hướng đối tượng.

3. Xét sơ đồ thực thể mối quan hệ dưới đây:

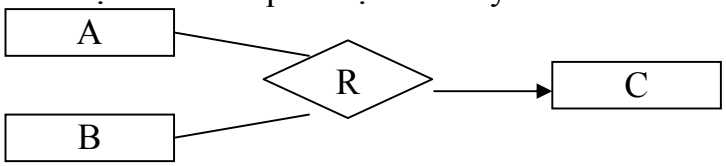


Giả sử khoá của A là A, khoá của B là B, khoá của C là C và khoá của D là D. Nếu chuyển sơ đồ trên sang mô hình dữ liệu quan hệ thì tập tất cả các khoá của quan hệ biểu diễn cho mối quan hệ R là gì.

Trả lời:

BCD

4. Cho sơ đồ thực thể mối quan hệ dưới đây:



Giả sử A có 10 thực thể, B có 100 thực thể và C có 20 thực thể. Nếu chuyển sơ đồ trên sang mô hình dữ liệu quan hệ thì quan hệ biểu diễn cho mối quan hệ R có thể có tối đa bao nhiêu bộ dữ liệu.

Trả lời:

Có tối đa $10 \times 100 = 1000$ bộ dữ liệu

4. Dựa trên cơ sở dữ liệu quan hệ có được ở câu 1, hãy viết các biểu thức đại số quan hệ biểu diễn cho các yêu cầu truy vấn dữ liệu sau đây:

a. Cho biết tên và giá của những mặt hàng có số lượng lớn hơn 100.

b. Cho biết mặt hàng ‘Đường’ có số lượng là bao nhiêu.

c. Cho biết khách hàng ‘Nguyễn Văn A’ đã mua những mặt hàng nào?
d. Những người nào đã cung cấp ‘Bơ’ cho cửa hàng?
e. Vào ngày 2/3/1998 đã bán được những mặt hàng nào?
f. Những khách hàng nào đã mua hàng của cửa hàng vào ngày 4/6/1998?
g. Ông ‘Lê Văn B’ đã cung cấp cho của hàng những mặt hàng nào?
h. Hóa đơn số 102 mua những mặt hàng nào, với số lượng và giá tiền là bao nhiêu?

5. Dựa trên cơ sở dữ liệu quan hệ có được ở câu 2, hãy viết các biểu thức đại số quan hệ biểu diễn cho các yêu cầu truy vấn dữ liệu sau đây:

a. Cho biết danh sách các sinh viên học lớp ‘Tin K2A’.
b. Điểm thi môn ‘Cơ sở dữ liệu’ của các sinh viên lớp ‘Tin K1B’
c. Những giáo viên nào được phân công giảng dạy lớp ‘Tin K22A’ học kỳ 1 năm học 2000-2001?
d. Trong học kỳ 1 năm học 2001-2002, giáo viên ‘Mai Thanh C’ được phân công dạy những lớp nào, môn học gì?
e. Cho biết sinh viên ‘Lâm Chí Thành’ phải thi lại những môn học nào (điểm lần 1 nhỏ hơn 5)

6. Biểu diễn các yêu cầu truy vấn dữ liệu ở câu 4 và 5 bằng câu lệnh SELECT.

7. Cho các quan hệ SINHVIEN, DIEMTHI và MONHOC lần lượt như sau:

MASV	HOTEN	GIOITINH	DIACHI	MALOP
CQK21001	Lê Hoài Nam	1	Huế	TIK21C
CQK22001	Nguyễn Văn Thanh	1	Đà Nẵng	TI22A
TCK8007	Hoàng Thị Thảo	0	Quảng Trị	TIK8
CQK23005	Lê Thị Hoa	0	Huế	TIK23

MASV	MAMH	DIEMLAN1	DIEMLAN2
CQK21001	TI01	4	7
CQK21001	TI02	8	
TCK8007	TI03	4	3
TCK8007	TI05	8	
CQK23005	TI01	2	3

MAMH	TENMH	SODVHT
TI01	Pascal	4
TI02	Ngôn ngữ C	4
TI03	Đồ hoạ	4
TI05	CSDL	5

Tính các biểu thức đại số quan hệ dưới đây và cho biết chức năng của mỗi biểu thức

a. $\pi_{HOTEN,GIOITINH}(\sigma_{DIACHI='Huế'}(SINHVIEN))$

b. $\pi_{HOTEN,TENMH,DIEMLAN1}(SINHVIEN \bowtie DIEMTHI \bowtie MONHOC)$

c. $\pi_{TENMH,DIEMLAN1,DIEMLAN2}(\sigma_{HOTEN='Lê\ Hoài\ Nam'}(SINHVIEN) \bowtie DIEMTHI \bowtie MONHOC))$

8. Cho hai quan hệ r và s sau đây:

A	B	C	D
2	3	1	1
1	2	2	3
1	1	2	1
1	2	1	1
2	3	2	3

Quan hệ r

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

Quan hệ s

Tính các biểu thức đại số quan hệ sau đây:

a. $\pi_{AB}(r) - \pi_{AB}(s)$

b. $\pi_{AD}(r) \bowtie s$

c. $\pi_{AB}(r) \div \pi_{AB}(\sigma_{A \le B}(s))$

d. $r \div \pi_{AB}(\sigma_{A \le B}(s))$

8. Cho r và s là hai quan hệ như sau:

A	B	C	D	E
a	c	c	e	a
b	c	a	a	g
b	a	c	h	a
a	c	f	e	g
b	b	c	e	a
b	a	f	h	g

D	E	F	G
b	b	c	f
g	h	a	c
h	g	g	f

Tính các biểu thức đại số quan hệ sau:

a. $\pi_{ACB}(r) - \pi_{EFG}(\sigma_{D=E}(s))$

b. $\pi_{CDE}(r) \triangleright\triangleleft \pi_{DEF}(s)$

c. $\pi_{ABCE}(r) \div \pi_{GF}(\sigma_{F\neq'c'}(s))$

d. $\pi_{ABC}(r) \div \pi_{G}(s)$

9. Cho hai quan hệ r và s có lược đồ quan hệ tương ứng là R(A,B) và S(A,B). Chứng minh rằng: $r \cap s = r \bowtie s$

10. Cho r_1 và r_2 là hai quan hệ trên lược đồ quan hệ là R với khóa là K. Quan hệ nào trong số các quan hệ dưới đây cũng có khóa là K.

a. $r_1 \cup r_2$

b. $r_1 \cap r_2$

c. $r_1 - r_2$

d. $r_1 \bowtie r_2$

11. Cho các quan hệ r, s và u. Chứng minh rằng:

a. $(r \bowtie s) \bowtie u = r \bowtie (s \bowtie u)$

b. $r \bowtie r = r$

12. Cho hai quan hệ r(R) và s(S). A là một thuộc tính của R. Chứng minh rằng:

$$\sigma_{A=a}(r \bowtie s) = \sigma_{A=a}(r) \bowtie s$$

13. Cho hai quan hệ r(R) và s(S) với $R \cap S = \varnothing$. Chứng minh rằng:

$$(r \bowtie s) \div s = r$$

BÀI TẬP CƠ SỞ DỮ LIỆU
LÝ THUYẾT THIẾT KẾ CSDL QUAN HỆ

1. Quan hệ:

r = (A B C D E F)

A	B	C	D	E	F
1	2	0	3	1	1
2	1	0	2	1	1
0	2	1	3	1	0
2	1	1	0	1	1

thoả mãn những phụ thuộc hàm nào trong số các phụ thuộc hàm sau:

AB → C, D → E, AB→E, BC→D, A→F

Trả lời:

D→E, AB→E, BC→D, A→F

2. Quan hệ:

r = (A B C D E F)

A	B	C	D	E	F
a	a	b	c	a	c
b	a	c	c	c	d
c	a	c	c	d	d
a	a	b	c	b	c
c	a	c	c	a	d
a	a	c	c	a	c
a	c	b	c	a	a

thoả mãn những phụ thuộc hàm nào trong số các phụ thuộc hàm sau:

AB→D, AB→E, ∅→B, ∅→D, CD→F

Trả lời:

AB→D, ∅→D

3. Cho quan hệ r sau đây:

r = (A B C D E F)

A	B	C	D	E	F
1	1	0	0	1	2
1	0	2	0	1	1
1	1	0	1	1	0
0	0	1	0	1	2

Tập phụ thuộc hàm nào dưới đây đúng trên quan hệ r?

a. F = {AB → C, C →E, CD → F}

b. F = {AB → CE, C→D, D →E }

c. F = {CD →EF, A →B, F →E }

d. F = {BC→D, D→E}

Trả lời: a

4. Cho quan hệ r:

r = (	A	B	C	D	E	F	)
	a	b	a	h	c	a	
	c	a	b	b	a	c	
	a	b	a	h	c	c	
	b	b	b	b	c	a	

Quan hệ r thoả mãn tập phụ thuộc hàm nào trong số các tập phụ thuộc hàm sau:

a. F = {AB→C, C→D, DE → F}

b. F = {AB→CD, DE →AF}

c. F = {A→C, B→E}

Trả lời: c

5. Chứng minh rằng quan hệ r thoả X→Y nếu và chỉ nếu X là khoá của $\pi_{XY}(r)$.

Chứng minh:

Xét quan hệ r thoả X→Y. Giả sử X không phải là khoá của $\pi_{XY}(r)$

Đặt $s = \pi_{XY}(r)$

=> $\exists\ t_1, t_2 \in s$ sao cho $t_1 \neq t_2$ và $t_1[X] = t_2[X]$ (1)

=> $t_1[Y] \neq t_2[Y]$ (2)

Do $s = \pi_{XY}(r)$ nên $\exists\ q_1, q_2 \in r$ sao cho: $q_1[XY] = t_1$ và $q_2[XY] = t_2$

=> $q_1[X] = t_1[X]$, $q_1[Y] = t_1[Y]$, $q_2[X] = t_2[X]$ và $q_2[Y] = t_2[Y]$ (3)

Từ (1), (2) và (3) => $q_1[X] = q_2[X]$ và $q_1[Y] \neq q_2[Y]$

=> Quan hệ r không thoả X→Y (mâu thuẫn)

Vậy X là khoá của $\pi_{XY}(r)$

6. Cho quan hệ r trên lược đồ quan hệ R. X và Y là tập con các thuộc tính của R. Chứng minh rằng nếu $\pi_X(R)$ có số bộ dữ liệu bằng số bộ dữ liệu của r thì phụ thuộc hàm X→Y luôn đúng trên quan hệ r.

Hướng dẫn: Chứng minh bằng phản chứng

7. Các khẳng định dưới đây đúng hay sai? Nếu đúng, hãy chứng minh; nếu sai, hãy chỉ ra một phản ví dụ.

a. Nếu $X \rightarrow Y$ và $W \rightarrow Z$ thì $XW \rightarrow YZ$

b. Nếu $X \rightarrow Y$ và $Y \rightarrow Z$ thì $X \rightarrow YZ$

c. Nếu $XY \rightarrow Z$ và $Z \rightarrow X$ thì $Z \rightarrow Y$

d. Nếu $X \rightarrow Y$, $W \rightarrow Z$ và $W \subseteq Y$ thì $X \rightarrow Z$

a. Đúng

$X \rightarrow Y \Rightarrow XW \rightarrow YW$ (tăng trưởng)

$W \rightarrow Z \Rightarrow YW \rightarrow YZ$ (tăng trưởng)

$\Rightarrow XW \rightarrow YZ$ (bắc cầu).

b. Đúng

c. Sai. Xét quan hệ sau

X	Y	Z
11...1	11...1	11...1
11...1	00...0	11...1

d. Đúng

8. Cho lược đồ quan hệ R(A,B,C,D,E,F,G,H,I) và tập phụ thuộc hàm F trên R.

$F = \{AB \rightarrow D, AH \rightarrow I, D \rightarrow H, HI \rightarrow G, BD \rightarrow I\}$

Chứng minh rằng nếu quan hệ r trên lược đồ quan hệ R thoả mãn F thì r cũng thoả phụ thuộc hàm $AB \rightarrow GI$ bằng các cách:

a. Sử dụng hệ tiên đề Armstrong

b. Tính bao đóng của tập thuộc tính

a. Chứng minh bằng Armstrong

$AB \rightarrow D$ và $D \rightarrow H \Rightarrow AB \rightarrow H$ (bắc cầu)

$\Rightarrow AB \rightarrow AH$ (tăng trưởng)

$AH \rightarrow I \Rightarrow AB \rightarrow I$ (bắc cầu)

$\Rightarrow AB \rightarrow HI$ (hợp)

$HI \rightarrow G \Rightarrow AB \rightarrow G$ (bắc cầu)

$\Rightarrow AB \rightarrow GI$ (hợp)

b. Bao đóng của AB theo F là: ABDHIG

Vậy $GI \subseteq AB^+ \Rightarrow F \models AB \rightarrow GI$

9. Cho lược đồ quan hệ R(A,B,C,D,E,F) và tập thuộc hàm trên R.

F = {A→B, D→F, E→D, D→C}

Tính bao đóng của các tập thuộc tính AE, BD

AE^+ = ABDCDEF, BD^+ = BDCF

10. Cho quan hệ r trên lược đồ quan hệ R. A và B là các thuộc tính của R, X và Y là tập con các thuộc tính của R. Bằng lập luận logic, chứng minh rằng nếu quan hệ r thoả mãn phụ thuộc hàm A→X, Y→B và Y⊆X thì r cũng thoả A→B.

Xét quan hệ r thoả A→X và Y→B.

Giả sử r không thoả A→B

⇒ ∃ t_1, t_2 ∈ r sao cho $t_1[A] = t_2[A]$ và $t_1[B] \neq t_2[B]$ (1)

Do r thoả A→X nên $t_1[X] = t_2[X]$

⇒ $t_1[Y] = t_2[Y]$ (do Y⊆ X) (2)

(1) và (2) ⇒ r không thoả Y→B (mâu thuẫn)

Vậy r thoả A→B

11. Cho lược đồ quan hệ R và tập phụ thuộc hàm F trên R. Nếu F = ∅ thì F^+ là tập phụ thuộc hàm như thế nào?

F^+ = {X→Y | Y⊆X, X, Y⊆R}

12. Kiểm tra tính chất bảo toàn thông tin của phép tách ρ=(ABC, ADE, DFG) của lược đồ quan hệ R(A, B, C, D, E, F, G) tương ứng với các tập phụ thuộc hàm dưới đây :

a. F = {AB→C, C→DE, AF→G}

b. F = {A→CB, A→DE, BD→FG}

c. F = {AC→DE, BE→FG}

d. F = {BD→AE, B→CFG}

R= ABCDEFG, ρ=(ABC, ADE, DFG)

Hướng dẫn:

Câu b:

F = {A→CB, A→DE, BD→FG}

Xây dựng bảng như sau :

	A	B	C	D	E	F	G
ABC	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}	b_{17}
ADE	a_1	b_{22}	b_{23}	a_4	a_5	b_{26}	b_{27}
DFG	b_{31}	b_{32}	b_{33}	a_4	b_{35}	a_6	a_7

Xét A→CB:

	A	B	C	D	E	F	G
ABC	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}	b_{17}
ADE	a_1	**a_2**	**a_3**	a_4	a_5	b_{26}	b_{27}
DFG	b_{31}	b_{32}	b_{33}	a_4	b_{35}	a_6	a_7

Xét A→DE:

	A	B	C	D	E	F	G
ABC	a_1	a_2	a_3	**a_4**	**a_5**	b_{16}	b_{17}
ADE	a_1	**a_2**	**a_3**	a_4	a_5	b_{26}	b_{27}
DFG	b_{31}	b_{32}	b_{33}	a_4	b_{35}	a_6	a_7

Xét BD→FG:

	A	B	C	D	E	F	G
ABC	a_1	a_2	a_3	**a_4**	**a_5**	b_{16}	b_{17}
ADE	a_1	**a_2**	**a_3**	a_4	a_5	**b_{16}**	**b_{17}**
DFG	b_{31}	b_{32}	b_{33}	a_4	b_{35}	a_6	a_7

Xét lại một lượt các phụ thuộc hàm, ta nhận thấy không có sự thay đổi nào trên bảng.
Vậy: Phép tách không bảo toàn phụ thuộc hàm do trong bảng cuối cùng không xuất hiện dòng có giá trị là a.

13. Kiểm tra tính chất bảo toàn phụ thuộc hàm của phép tách ρ = (ABD, ACE, BEF, DEG) của lược đồ quan hệ R(A,B,C,D,E,F,G) tương ứng với các tập phụ thuộc hàm dưới đây :

a. F = {BD→A, A→CF, D→F, BE→G}
b. F = {A→EF, BD→CD, F→G}
c. F = {A→B, A→D, D→E, E→C, BD→FG}

14. Cho lược đồ quan hệ R(A,B,C,D,E,F) và tập phụ thuộc hàm F trên R:

F={AB→C, A→DE, D→F, A→B, F→A}

a. Tập phụ thuộc hàm F có cực tiểu hay không ? Nếu không hãy xác định một phủ cực tiểu của F.
b. Phép tách ρ = {ABC, ADE, DF, AF} có bảo toàn thông tin hay không ?
c. Phép tách trên có bảo toàn phụ thuộc hàm hay không ?

a. Tìm phủ cực tiểu của F.

Bước 1: Thay tất cả các phụ thuộc hàm có từ hai thuộc tính ở vế phải trở lên bởi những phụ thuộc hàm chỉ có một thuộc tính ở vế phải.

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

Bước 2: Loại bỏ khỏi vế trái của các phụ thuộc hàm trong F những thuộc tính thừa

Xét $AB \to C$: $A^+_F = ABDCEF \Rightarrow C \in A^+_F \Rightarrow$ thuộc tính B thừa.

$\Rightarrow$ $F = \{A \to C, A \to D, A \to E, D \to F, A \to B, F \to A\}$

Bước 3: Loại bỏ khỏi F những phụ thuộc hàm thừa

Xét $A \to C$: $A^+_{F\setminus\{A\to C\}} = ABDEF \Rightarrow C \notin A^+_{F\setminus\{A\to C\}}$

Xét $A \to D$: $A^+_{F\setminus\{A\to D\}} = ABC \Rightarrow D \notin A^+_{F\setminus\{A\to D\}}$

Xét $A \to E$: $A^+_{F\setminus\{A\to E\}} = ACDFB \Rightarrow E \notin A^+_{F\setminus\{A\to E\}}$

Xét $D \to F$: $D^+_{F\setminus\{D\to F\}} = D \Rightarrow F \notin D^+_{F\setminus\{D\to F\}}$

Xét $A \to B$: $A^+_{F\setminus\{A\to B\}} = ACDEF \Rightarrow B \notin A^+_{F\setminus\{A\to B\}}$

Xét $F \to A$: $F^+_{F\setminus\{F\to A\}} = F \Rightarrow A \notin F^+_{F\setminus\{F\to A\}}$

Vậy không có phụ thuộc hàm nào thừa.

$\Rightarrow$ Phủ cực tiểu của F là: $\{A \to C, A \to D, A \to E, D \to F, A \to B, F \to A\}$

b. Phép tách $\rho = \{ABC, ADE, DF, AF\}$ bảo toàn thông tin

c. Phép tách $\rho = \{ABC, ADE, DF, AF\}$ bảo toàn phụ thuộc hàm.

15. Cho lược đồ quan hệ R(A,B,C,D,E,F,G) và tập phụ thuộc hàm F trên R:

$F = \{AB \to C, AC \to D, D \to A, DC \to EF, B \to G\}$

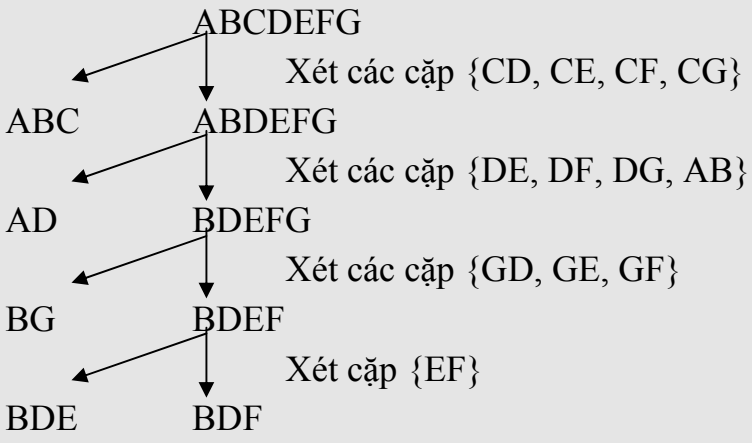
a. Hãy tìm một phủ cực tiểu của F.
b. Xác định tất cả các khoá của lược đồ quan hệ R.
c. Trên R có tồn tại dư thừa và dị thường về dữ liệu hay không? Nếu có, hãy tìm một phép tách thành BCNF của R.
d. Phép tách tìm được ở trên có bảo toàn phụ thuộc hàm hay không?

Hướng dẫn:

a. Phủ cực tiểu của F: {AB→C, AC→D, D→A, DC→E, DC→F, B→G}

b. Tập các khoá của R: {AB, BD}

c. Tìm một phép tách thành BCNF của R:



Phép tách tìm được ρ = (ABC, AD, BG, BDE, BDF)

d. Phép tách trên không bảo toàn phụ thuộc hàm

16. Cho lược đồ quan hệ R = ABCDEFGH. Xác định tất cả các khoá của R tương ứng với các tập phụ thuộc hàm dưới đây:

a. F_a = {A→B, A→C, D→ B, D→C, CE→A, CE→D, G→H}
b. F_b = {CD→E, AD→F, A→G, C→H, B→A, B→C, GH→B}
c. F_c = {AB→E, CD→F, EF→G, C→H, H→A, H→B}
d. F_d = {AB→C, AB→D, AB→E, C→A, D→B, E→F, H→B, FG→H}

a. F_a = {A→B, A→C, D→ B, D→C, CE→A, CE→D, G→H}
Sử dụng thuật toán tìm tất cả các khoá ta có:
Tập khoá ban đầu: *K* = {EFGC}
Xét khoá: EFGC

A→B Siêu khóa: AEFGC
A-->C Siêu khóa: AEFG ==> Tìm được khoá mới AEFG
K = {EFGC, AEFG}
D→B Siêu khóa: DEFGC
D→C Siêu khóa: DEFG ==> Tìm được khoá mới DEFG
K = {EFGC, AEFG, DEFG}
CE→A Siêu khóa: CEFG
CE→D Siêu khóa: CEFG
G→H Siêu khóa: GEFC

Xét khóa: AEFG

A→B	Siêu khóa: AEFG
A→C	Siêu khóa: AEFG
D→B	Siêu khóa: DAEFG
D→C	Siêu khóa: DAEFG
CE→A	Siêu khóa: CEFG
CE→D	Siêu khóa: CEAFG
G→H	Siêu khóa: GAEF

Xét khóa: DEFG

A→B	Siêu khóa: ADEFG
A→C	Siêu khóa: ADEFG
D→B	Siêu khóa: DEFG
D→C	Siêu khóa: DEFG
CE→A	Siêu khóa: CEDFG
CE→D	Siêu khóa: CEFG
G→H	Siêu khóa: GDEF

Tập tất cả các khoá là: K = {EFGC, AEFG, DEFG}

b. F_b = {CD→E, AD→F, A→G, C→H, B→A, B→C, GH→B}
K = {DGH, ADH, CDG, CAD, BD}

c. F_c = {AB→E, CD→F, EF→G, C→H, H→A, H→B}
K = {CD}

d. F_d = {AB→C, AB→D, AB→E, C→A, D→B, E→F, H→B, FG→H}
K = {GCF, AGF, EGC, EAG, BGC, ABG, DGC, HGC, DAG, HAG}

17. Cho lược đồ quan hệ R(A, B, C, D, E) và tập phụ thuộc hàm trên R:

F = {A→BC, CD→E, ♣→D }

Trong đó ♣ là một thuộc tính nào đó trong số các thuộc tính của R và ♣→D là phụ thuộc hàm không tầm thường. Có thể đưa ra được kết luận gì đối với khoá của R cho dù ♣ là thuộc tính nào.

Mọi khoá của R đều phải chứa thuộc tính A.

18. Cho lược đồ quan hệ R, X→Y là phụ thuộc hàm trên R và K là một khoá của R. Chứng minh rằng X∪(K\Y) là siêu khoá của R.

Ta có:

X→Y ⇒ X∪(K\Y) → Y∪(K\Y) (tăng trưởng)

$K \subseteq Y \cup (K \setminus Y) \Rightarrow Y \cup (K \setminus Y) \rightarrow K$

Do K là khoá nên $K \rightarrow R$

$\Rightarrow X \cup (K \setminus Y) \rightarrow R$ (bắc cầu)

$\Rightarrow X \cup (K \setminus Y)$ là siêu khoá

19. Dựa vào lược đồ quan hệ R và các tập phụ thuộc hàm ở câu 16:

a. Hãy tìm một phép tách thành BCNF của R ứng với mỗi tập phụ thuộc hàm F_a, F_b, F_c.

b. Hãy tìm một phép tách thành 3NF của R ứng với mỗi tập phụ thuộc hàm F_a, F_b, F_c.

c. Các phép tách tìm được ở câu b có bảo toàn thông tin hay không?

20. Cho lược đồ quan hệ R(A,B,C,D) và tập phụ thuộc hàm F={B→C} trên R. Có thể bổ sung vào F phụ thuộc hàm nào trong số các phụ thuộc hàm dưới đây để R là 3NF nhưng không phải là BCNF.

(a) D→AB

(b) AC →D

(c) CD→B

(d) AD→B

Để R là 3NF nhưng không là BCNF thì với mọi phụ thuộc hàm X→A∈F, A phải là thuộc tính khoá. Vậy có thể bổ sung vào F phụ thuộc hàm CD→B để R là 3NF nhung không là BCNF

21. Cho lược đồ quan hệ R(A, B, C, D) và tập phụ thuộc hàm F ={AB→C} trên R. Phải bổ sung vào F phụ thuộc hàm nào trong số các phụ thuộc hàm dưới đây để R là 3NF nhưng không phải là BCNF

a. AB→D

b. C→B

c. C→D

Có thể bổ sung vào F phụ thuộc hàm C→B để R là 3NF nhưng không là BCNF

22. Cho lược đồ quan hệ R(A,B,C,D,E). Xác định dạng chuẩn của R tương ứng với các tập phụ thuộc hàm sau:

a. F = {AB→C, AB→D, C→A, D→B}

b. F = {AB→E, CD→A, E→B}

c. F = {CD→A, BCD→E}

. a. 3NF b. 1NF c. 1NF

23. Cho lược đồ quan hệ R(A, B, C, D). và tập phụ thuộc hàm F = {AB→C, C→D, D→A} trên R.
a. Tìm tập tất cả các phụ thuộc hàm không tầm thường và các phụ thuộc hàm mà vế phải không có thuộc tính xuất hiện trong vế trái được suy dẫn từ F.
b. Xác định tất cả các khoá của R.

a. Trả lời:

{AB→C, C→D, D→A, C→A, C→DA, AB→D, AB→CD, AC→D, BC→A, BC→D, BC→AD, BD→A, BD→C, BD→AC, CD→A, ABC→D, ABD→C, BCD→A}

b. Tập tất cả các khoá: AB, CB, BD

24. Cho lược đồ quan hệ R(A, B, C, D). và tập phụ thuộc hàm F = {A→B, B→C, B→D} trên R.
a. Tìm tập tất cả các phụ thuộc hàm không tầm thường và các phụ thuộc hàm mà vế phải không có thuộc tính xuất hiện trong vế trái được suy dẫn từ F.
b. Xác định tất cả các khoá của R.

25. Cho lược đồ quan hệ R(A, B, C, D). và tập phụ thuộc hàm F = {Ab→C, BC→D, CD→A, AD→B} trên R.
a. Tìm tập tất cả các phụ thuộc hàm không tầm thường và các phụ thuộc hàm mà vế phải không có thuộc tính xuất hiện trong vế trái được suy dẫn từ F.
b. Xác định tất cả các khoá của R.

26. Cho lược đồ quan hệ R(A, B, C, D). và tập phụ thuộc hàm F = {A→B, B→C, C→D, D→A} trên R.
a. Tìm tập tất cả các phụ thuộc hàm không tầm thường và các phụ thuộc hàm mà vế phải không có thuộc tính xuất hiện trong vế trái được suy dẫn từ F.
b. Xác định tất cả các khoá của R.

27. Hãy chỉ ra một phản ví dụ để chứng tỏ rằng các phát biểu dưới đây là sai.

a. Nếu quan hệ r thoả A→B thì r thoả B→A

b. Nếu quan hệ r thoả AB→C và A→C thì r thoả B→C

c. Nếu quan hệ r thoả AB→C thì r thoả A→C hoặc B→C

a. Quan hệ thoả A→B nhưng không thoả B→A

A	B
1	0
0	0

b. Quan hệ r thoả AB→C và A→C nhưng không thoả B→C

A	B	C
1	1	0
0	1	1

28. Cho X và Y là hai tập thuộc tính. Chứng minh rằng nếu $X \subseteq Y$ thì $X^+ \subseteq Y^+$ (với bao đóng của X và Y được tính theo cùng một tập phụ thuộc hàm)

Chứng minh:

$\forall A \in X^+$, ta có $F \models X \rightarrow A$

Do $X \subseteq Y$ nên $Y \rightarrow X$

Theo tiên đề bắc cầu: $F \models Y \rightarrow A \quad \Rightarrow A \in Y^+$

Vậy $X^+ \subseteq Y^+$

29. Cho F là tập phụ thuộc hàm. Chứng tỏ rằng phụ thuộc hàm $X \rightarrow A \in F$ là dư thừa nếu và chỉ nếu A thuộc X^+ khi bao đóng này được tính theo tập phụ thuộc hàm $F \setminus \{X \rightarrow A\}$

Chứng minh

* Chứng minh nếu $X \rightarrow A \in F$ dư thừa thì $A \in X^+_{F \setminus \{X \rightarrow A\}}$

Nếu $X \rightarrow A \in F$ là dư thừa thì $F^+ = (F \setminus \{X \rightarrow A\})^+$

$\Rightarrow$ Với mọi phụ thuộc hàm $W \rightarrow V \in F^+$, ta có $F \setminus \{X \rightarrow A\} \models W \rightarrow V$

$X \rightarrow A \in F$ nên $F \setminus \{X \rightarrow A\} \models X \rightarrow A$

$\Rightarrow A \in X^+_{F \setminus \{X \rightarrow A\}}$

* Chứng minh nếu $A \in X^+_{F \setminus \{X \rightarrow A\}}$ thì $X \rightarrow A \in F$ là dư thừa

Do $F \setminus \{X \rightarrow A\} \subseteq F$ nên $(F \setminus \{X \rightarrow A\})^+ \subseteq F^+$

Ta chứng minh $F^+ \subseteq (F \setminus \{X \rightarrow A\})^+$

Thật vậy, $\forall W \rightarrow V \in F$ ta có $W \rightarrow V \in F \setminus \{X \rightarrow A\}$ hoặc $W \rightarrow V$ chính là $X \rightarrow A$

$\Rightarrow V \subseteq W^+_{F \setminus \{X \rightarrow A\}}$

$\Rightarrow F^+ \subseteq (F \setminus \{X \rightarrow A\})^+$

Vậy $F^+ = (F \setminus \{X \rightarrow A\})^+ \quad \Rightarrow X \rightarrow A \in F$ là thừa.

30. Chứng tỏ rằng thuộc tính $B \in X$ trong phụ thuộc hàm $X \to A \in F$ là dư thừa nếu và chỉ nếu $A \in (X \setminus \{B\})^+_F$.

31. Cho hai lược đồ quan hệ R_1 và R_2, $R_1 \cap R_2 = X$. Chứng tỏ rằng với mọi quan hệ $r(R_1R_2)$ thoả $X \to R_2$ ta luôn có:

$$r = \pi_{R1}(r) \bowtie \pi_{R2}(r)$$

Dễ dàng chứng minh được $r \subseteq \pi_{R1}(r) \bowtie \pi_{R2}(r)$

Ta chứng minh $\pi_{R1}(r) \bowtie \pi_{R2}(r) \subseteq r$.

Xét bộ $\mu \in \pi_{R1}(r) \bowtie \pi_{R2}(r)$

$\Rightarrow \exists \mu_1 \in \pi_{R1}(r)$ và $\exists \mu_2 \in \pi_{R2}(r)$ sao cho:

$$\Rightarrow \begin{cases} \mu_1[X] = \mu_2[X] = \mu[X] \\ \mu_1[R_1 \setminus R_2] = \mu[R_1 \setminus R_2] \\ \mu_2[R_2 \setminus R_1] = \mu[R_2 \setminus R_1] \end{cases}$$

$\Rightarrow \exists \nu_1, \nu_2 \in r$ sao cho $\mu_1 = \nu_1[R_1]$ và $\mu_2 = \nu_2[R_2]$

$$\Rightarrow \begin{cases} \mu_1[X] = \nu_1[X] \\ \mu_1[R_1 \setminus R_2] = \nu_1[R_1 \setminus R_2] \\ \mu_2[X] = \nu_2[X] \\ \mu_2[R_2 \setminus R_1] = \nu_2[R_2 \setminus R_1] \end{cases}$$

$$\Rightarrow \begin{cases} \nu_1[X] = \nu_2[X] = \mu[X] \\ \nu_1[R_1 \setminus R_2] = \mu[R_1 \setminus R_2] \\ \nu_2[R_2 \setminus R_1] = \mu[R_2 \setminus R_1] \end{cases}$$

Do r thoả $X \to R_1 \setminus R_2$ nên $\nu_1[R_1 \setminus R_2] = \nu_2[R_1 \setminus R_2]$

$\Rightarrow \nu_2 = \mu \quad \Rightarrow \mu \in r$

$\Rightarrow \pi_{R1}(r) \bowtie \pi_{R2}(r) \subseteq r$

Vậy $r = \pi_{R1}(r) \bowtie \pi_{R2}(r)$

32. Cho hai tập phụ thuộc hàm F và G. Chứng minh rằng nếu $\forall X \to Y \in F$, bao đóng của X theo G chứa Y thì $F^+ \subseteq G^+$.

Chứng minh:

Xét $W \to V \in F^+$, ta có: $F \models W \to V$

Do $Y \subseteq X^+_G \quad \forall X \to Y \in F$ nên $F \subseteq G^+$

$\Rightarrow G^+ \models W \to V$

$\Rightarrow G \models W \rightarrow V$

Vậy: $W \rightarrow V \in G^+$

$\Rightarrow F^+ \subseteq G^+$

33. Cho quan hệ sau đây:

TENLOP	MONHOC	PHONG	BUOI
Tin K1	Pascal	1	Sáng
Tin K2	Foxpro	2	Chiều
Tin K2	Pascal	1	Sáng

Quan hệ trên có thoả mãn phụ thuộc đa trị:

F = {TENLOP $\rightarrow\rightarrow$PHONG, TENLOP$\rightarrow\rightarrow$BUOI}

hay không? Nếu không, phải bổ sung vào quan hệ trên những bộ dữ liệu nào để quan hệ thoả mãn F.

Quan hệ không thoả mãn tập phụ thuộc đa trị:

F = {TENLOP $\rightarrow\rightarrow$PHONG, TENLOP$\rightarrow\rightarrow$BUOI}

Để quan hệ trên thoả F, phải bổ sung vào F các bộ dữ liệu sau:

TENLOP	**MONHOC**	**PHONG**	**BUOI**
Tin K2	Pascal	2	Sáng
Tin K2	Foxpro	1	Chiều
Tin K2	Pascal	1	Chiều
Tin K2	Foxpro	2	Sáng

34. Cho quan hệ r sau đây:

A	B	C	D	E
1	2	3	4	5
1	0	3	6	5
8	2	3	4	7

Phải bổ sung vào r những bộ dữ liệu nào để quan hệ r thoả mãn phụ thuộc đa trị A$\rightarrow\rightarrow$BC và CD$\rightarrow\rightarrow$BE

35. Quan hệ r sau đây:

A	B	C	D	E
1	2	2	0	1
1	1	2	1	0
0	2	2	1	1
1	1	2	0	0
1	2	2	1	1

Thoả mãn phụ thuộc đa trị nào trong số các phụ thuộc đa trị sau đây:

$A \to\to B$, $AC \to\to D$, $AC \to\to BE$

Quan hệ r thoả mãn phụ thuộc đa trị $AC \to\to BE$

36. Cho quan hệ r(A,B,C,D). Quan hệ r phải như thế nào để phụ thuộc đa trị $\varnothing \to\to CD$ luôn đúng trên r.

. $r = \pi_{AB}(r) \times \pi_{CD}(r)$

37. Cho quan hệ r trên lược đồ quan hệ R. X, Y, Z, W là tập con các thuộc tính của R. Chứng minh rằng các phát biểu sau đây là đúng:

a. Nếu r thoả $X \to\to Y$ thì r thoả $XZ \to\to Y$
b. Nếu r thoả $X \to\to Y$ và $X \to\to Z$ thì r thoả $X \to\to YZ$
c. Nếu r thoả $\{X \to\to Y, X \to\to Z\}$ thì r thoả $\{X \to\to Y \cap Z, X \to\to Y-Z\}$
d. Nếu r thoả $\{X \to\to Y, Y \to\to Z\}$ thì r thoả $\{X \to\to YZ, X \to\to Z-Y\}$

a. Xét quan hệ r thoả $X \to\to Y$

Giả sử $\exists\, t_1, t_2 \in r$ sao cho $t_1[XZ] = t_2[XZ]$

Ta có: $t_1[X] = t_2[X]$ và $t_1[Z] = t_2[Z]$

Do r thoả $X \to\to Y$ nên tồn tại bộ t_3 sao cho:

$t_3[X] = t_1[X]$, $t_3[Y] = t_1[Y]$ và $t_3[R-XY] = t_2[R-XY]$

=> $t_3[R-XYZ] = t_2[R-XYZ]$

Mặt khác:

$t_3[(R-XY) \cap Z] = t_1[(R-XY) \cap Z]$ (do $(R-XY) \cap Z \subseteq R-XY$ và $(R-XY) \cap Z \subseteq Z$)

Vậy: $t_3[X] = t_1[X]$, $t_3[Y] = t_1[Y]$ và $t_3[(R-XY) \cap Z] = t_1[(R-XY) \cap Z]$

=> $t_1[XZ] = t_1[XZ]$

Vậy, $\exists t_3 \in r$ sao cho:

$t_3[XZ] = t_1[XZ]$, $t_3[Y] = t_1[Y]$ và $t_3[R-XYZ] = t_2[R-XYZ]$

=> r thoả $XZ \to\to Y$

b. Xét quan hệ r thoả $X \to\to Y$ và $X \to\to Z$.

Giả sử $\exists\, t_1, t_2 \in r$ sao cho: $t_1[X] = t_2[X]$

Do r thoả $X \to\to Y$ nên $\exists t_3 \in r$ sao cho

$t_3[X] = t_1[X]$

$t_3[Y] = t_1[Y]$

$t_3[R-XY] = t_2[R-XY]$ (1)

Do r thoả $X \to\to Z$ nên $\exists t_4 \in r$ sao cho

$t_4[X] = t_1[X]$

$t_4[Z] = t_1[Z]$

$t_4[R-XZ] = t_3[R-XZ]$ (2)

(1) $\Rightarrow t_3[R-XYZ] = t_2[R-XYZ]$

(2) $\Rightarrow t_4[R-XYZ] = t_3[R-XYZ]$

$\Rightarrow t_4[R-XYZ] = t_2[R-XYZ]$ (*)

Ta có:

$t_4[Y \cap (R-XZ)] = t_3[Y \cap (R-XZ)]$ (Do $Y \cap (R-XZ) \subseteq R-XZ$ và $t_4[R-XZ]=t_3[R-XZ]$)

$t_3[Y \cap (R-XZ)] = t_1[Y \cap (R-XZ)]$ (Do $Y \cap (R-XY) \subseteq Y$ và $t_3[Y] = t_1[Y]$)

$\Rightarrow t_4[Y \cap (R-XZ)] = t_1[Y \cap (R-XZ)]$

Vậy: $t_4[X] = t_1[X]$, $t_4[Z] = t_1[Z]$ và $t_4[Y \cap (R-XZ)] = t_1[Y \cap (R-XZ)]$

$\Rightarrow t_4[YZ] = t_1[XZ]$

$\Rightarrow \exists t_4 \in r$ thoả $t_4[X] = t_1[X]$, $t_4[YZ] = t_1[YZ]$ và $t_4[R-XYZ] = t_2[R-XYZ]$

$\Rightarrow$ r thoả $X \rightarrow\rightarrow YZ$

c. Sử dụng qui tắc hợp và bù để chứng minh

d. Chứng minh $\{X \rightarrow\rightarrow Y, Y \rightarrow\rightarrow Z\} \models X \rightarrow\rightarrow YZ$

Xét quan hệ r thoả $\{X \rightarrow\rightarrow Y, Y \rightarrow\rightarrow Z\}$. Giả sử $\exists t_1, t_2 \in r$ sao cho:

$t_1[X] = t_2[X]$

Do r thoả $X \rightarrow\rightarrow Y$ nên $\exists t_3 \in r$ sao cho:

$t_3[X] = t_1[X]$ $t_3[Y] = t_1[Y]$ và $t_3[R-XY] = t_2[R-XY]$

Do r thoả $Y \rightarrow\rightarrow Z$ nên $\exists t_4 \in r$ sao cho:

$t_4[Y] = t_1[Y]$ $t_4[Z] = t_1[Z]$ và $t_4[R-YZ] = t_3[R-YZ]$

Ta nhận thấy:

$t_4[YZ] = t_1[YZ]$ (1)

$t_4[R-XYZ] = t_3[R-XYZ] = t_2[R-XYZ]$ (2)

Ta có:

$$\begin{cases} t_3[X] = t_1[X] \\ t_4[R-YZ] = t_3[R-YZ] \end{cases}$$

$\Rightarrow t_4[X \cap (R-YZ)] = t_1[X \cap (R-YZ)]$ (3)

Từ (1) và (3), suy ra:

$t_4[(X \cap (R-YZ)) \cup YZ] = t_1[(X \cap (R-YZ)) \cup YZ]$

Do $X \subseteq (X \cap (R-YZ)) \cup YZ$ => $t_4[X] = t_1[X]$ (4)

Từ (4), (1) và (2) => $X \rightarrow\rightarrow YZ$

Sử dụng kết quả trên và câu (c) để chứng minh r thoả $X \rightarrow\rightarrow Z-Y$

38. Cho quan hệ r trên lược đồ quan hệ R. X, Y, Z, W là tập con các thuộc tính của R. Chứng minh rằng nếu r thoả $X \rightarrow\rightarrow Y$ và $Z \subseteq W$ thì r thoả $XW \rightarrow\rightarrow YZ$.

Hướng dẫn

Giả sử $\exists t_1, t_2 \in r$ thoả $t_1[XW] = t_2[XW]$. Chứng minh rằng tồn tại bộ $t \in r$ thoả:

$t[XW] = t_1[XW]$, $t[YZ] = t_1[YZ]$ và $t[R-XYW] = t_2[R-XYW]$

39. Cho lược đồ quan hệ R(A, B, C, D, E). Chứng minh rằng nếu quan hệ r trên R thoả tập phụ thuộc đa trị:

$F = \{A \rightarrow\rightarrow BC, DE \rightarrow\rightarrow C\}$

thì r cũng thoả $A \rightarrow\rightarrow BDE$, $A \rightarrow\rightarrow CDE$

Hướng dẫn

Ta có: $A \rightarrow\rightarrow BC \Rightarrow A \rightarrow\rightarrow DE$ (tính bù)

$DE \rightarrow\rightarrow C \Rightarrow A \rightarrow\rightarrow C$ (bắc cầu)

$\Rightarrow A \rightarrow\rightarrow B$ (luật tách)

$\Rightarrow A \rightarrow\rightarrow BDE$ và $A \rightarrow\rightarrow CDE$ (luật hợp)

40. Cho quan hệ r(R). X và Y là tập con các thuộc tính của R, Z=R-XY. Chứng minh rằng phép tách $\rho = (XY, XZ)$ là phép tách bảo toàn thông tin nếu và chỉ nếu r thoả $X \rightarrow\rightarrow Y$.

Hướng dẫn

($\Rightarrow$) Giả sử phép tách $\rho = (XY, XZ)$ là phép tách bảo toàn thông tin

tức là $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$

Giả sử $\exists t_1, t_2 \in r$ sao cho $t_1[X] = t_2[X]$

=> $\exists t_1' \in \pi_{XY}(r)$ sao cho $t_1' = t_1[XY]$

$\exists t_2' \in \pi_{XZ}(r)$ sao cho $t_2' = t_2[XZ]$

Do $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$ nên trong r phải tồn tại bộ t_3 sao cho:

$t_3[XY] = t_1[XY]$ và $t_3[XZ] = t_2[XZ]$

Vậy trong r phải tồn tại bộ t_3 thoả điều kiện

$t_3[X] = t_1[X]$, $t_3[Y] = t_1[Y]$ và $t_3[R-XY] = t_2[R-XY]$

$\Rightarrow$ r thoả $X \rightarrow\rightarrow Y$

($\Leftarrow$) Xét quan hệ r thoả phụ thuộc đa trị $X \rightarrow\rightarrow Y$, ta chứng minh $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r)$

Hiển nhiên $r \subseteq \pi_{XY}(r) \bowtie \pi_{XZ}(r)$

Xét $t \in \pi_{XY}(r) \bowtie \pi_{XZ}(r)$

$\Rightarrow \exists t_1 \in \pi_{XY}(r)$ và $\exists t_2 \in \pi_{XZ}(r)$ sao cho:

$t_1[X] = t_2[X] = t[X]$, $t_1[Y] = t[Y]$ và $t_2[Z] = t[Z]$

Ta lại có:

$\exists t_1' \in r$ sao cho $t_1'[XY] = t_1$

$\exists t_2' \in r$ sao cho $t_2'[XZ] = t_2$

Do r thoả $X \rightarrow\rightarrow Y$ nên $\exists t' \in r$ sao cho:

$t'[X] = t_1'[X] = t[X]$, $t'[Y] = t_1'[Y] = t[Y]$ và $t[Z] = t_2'[Z] = t[Z]$

$\Rightarrow t' = t$

$\Rightarrow t \in r$

Vậy $r = \pi_{XY}(r) \bowtie \pi_{XZ}(r) \Rightarrow$ Phép tách ρ là phép tách bảo toàn thông tin

41. Cho lược đồ quan hệ R(A, B, C, D, E, F) và tập phụ thuộc đa trị trên R:

$F = \{A \rightarrow\rightarrow BC, AD \rightarrow\rightarrow EF\}$

Tinh cơ sở phụ thuộc của A, AD.

42. Cho quan hệ R(A,B,C,D,E) và tập phụ thuộc hàm trên R:

$F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow AE\}$

a. Tìm tất cả các khoá của R.
b. Phụ thuộc hàm nào trong tập phụ thuộc hàm F vi phạm BCNF.
c. Giả sử ta tách R thành hai lược đồ $R_1(A,D,E)$ và $R_2(B,C,D)$. Hãy xác định tập phụ thuộc hàm cực tiểu trên R_1 và R_2.
d. Tìm tất cả các khoá của R_1 và R_2
e. Xác định dạng chuẩn của R_1 và R_2

R = ABCDE, $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow AE\}$

a. Tập các khoá của R: {BD, CB, AB}

b. Phụ thuộc hàm $C \rightarrow D$ và $D \rightarrow AE$ vi phạm BCNF

c. $\pi_{ADE}(F) = \{D \rightarrow AE\}$

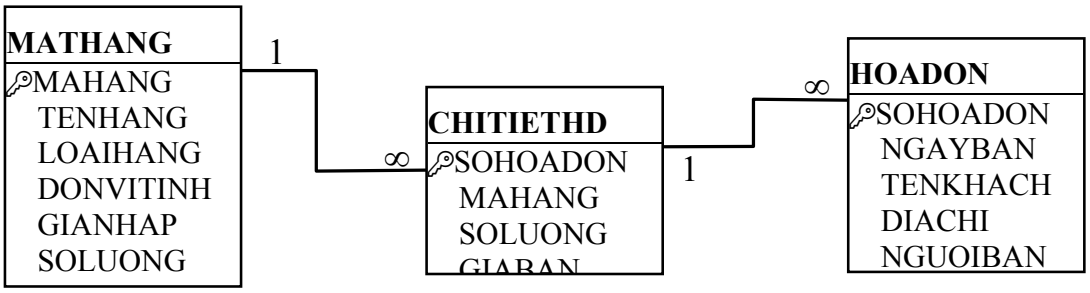
$\pi_{BCD}(F) = \{BD \rightarrow C, C \rightarrow D\}$

d. $R_1(ADE)$ ở dạng chuẩn Boyce-Codd, $R_2(BCD)$ ở dạng chuẩn 3.

43. Cho lược đồ quan hệ R(A,B,C) và tập phụ thuộc hàm $F = \{AB \rightarrow C, A \rightarrow B\}$ trên R. Hãy cho một ví dụ để chứng tỏ rằng không có phụ thuộc hàm nào trong F có thể suy dẫn logic ra phụ thuộc hàm còn lại.

BÀI TẬP CƠ SỞ DỮ LIỆU
SQL

Để tổ chức quản lý bán hàng cho một cửa hàng nhỏ, người ta sử dụng các bảng dữ liệu như sơ đồ sau đây:



1. Sử dụng câu lệnh CREATE TABLE để tạo các bảng dữ liệu trên.

```
CREATE TABLE    MATHANG
(
        MAHANG              char  (10)  PRIMARY KEY,
        TENHANG         char  (50) ,
        LOAIHANG        char  (20),
        DONVITINH       char  (10),
        GIANHAP             int,
        SOLUONG         int
)

CREATE TABLE    HOADON
(
        SOHOADON        int     PRIMARY KEY,
        NGAYBAN         datetime,
        TENKHACH        char (20) ,
        DIACHI          char (50),
        NGUOIBAN        char (20)
)

CREATE TABLE CHITIETHOADON
(
       SOHOADON         int             NOT NULL ,
       MAHANG           char (10)       NOT NULL ,
       SOLUONG          int             NOT NULL ,
       GIABAN           int             NOT NULL
```

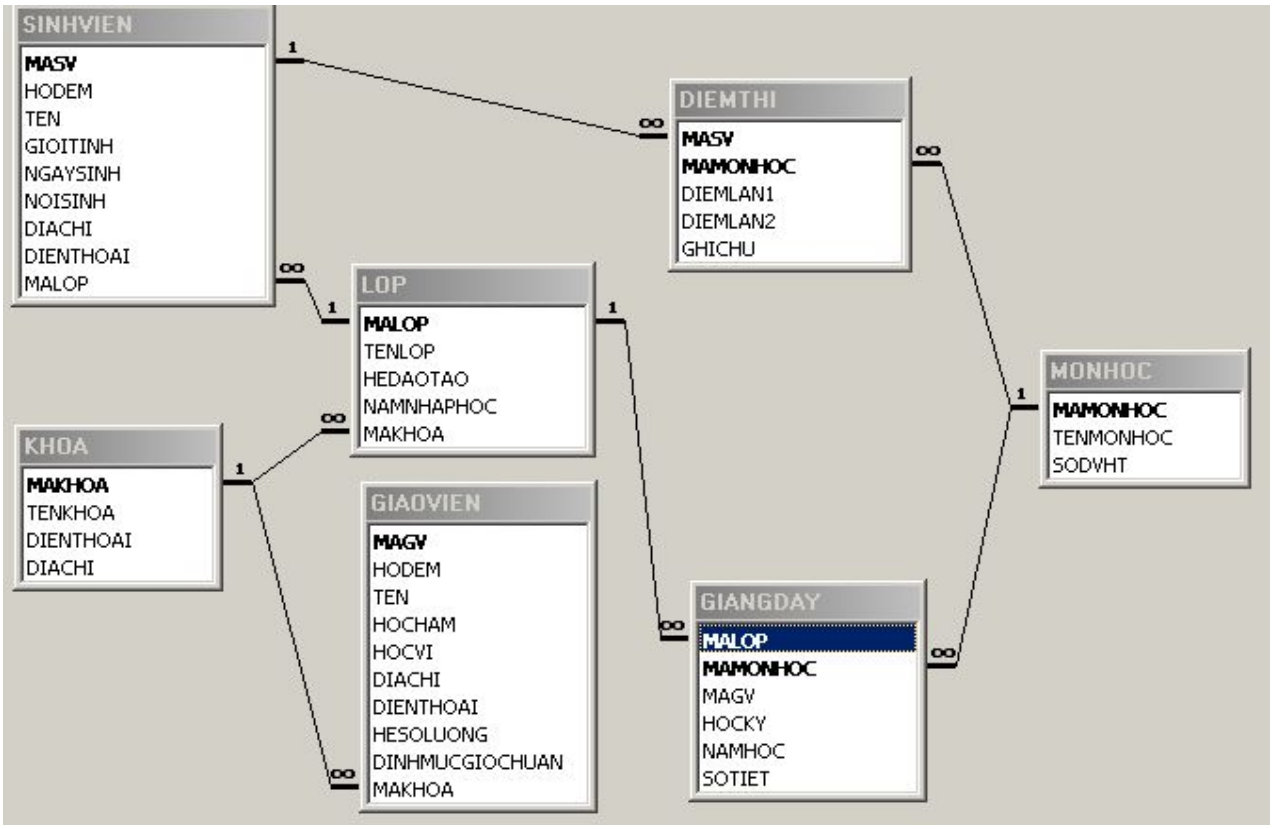
```
        PRIMARY KEY (SOHOADON,MAHANG)
)
```

2. Sử dụng câu lệnh ALTER TABLE để tạo mối quan hệ giữa hai bảng MATHANG và CHITIETHD thông qua trường MAHANG.

```
        ALTER TABLE    CHITIETHOADON
        ADD
                FOREIGN KEY    ( SOHOADON )
                REFERENCES    HOADON  ( SOHOADON )
```

3. Sử dụng câu lệnh ALTER TABLE để tạo mối quan hệ giữa hai bảng HOADON và CHITIETHD thông qua trường SOHOADON.

Để quản lý công tác đào tạo trong một trường đại học, người ta sử dụng CSDL như sau:



Sử dụng câu lệnh SELECT để thực hiện các yêu cầu sau đây:

4. Cho biết mã sinh viên, họ tên và ngày sinh của các sinh viên

```
        SELECT MASV,HODEM,TEN,NGAYSINH FROM SINHVIEN
```

5. Cho biết tên lớp và năm nhập học của các lớp tại chức (hệ đào tạo là tại chức)

```
SELECT TENLOP,NAMNHAPHOC
FROM LOP WHERE HEDAOTAO='Tại chức'
```

6. Cho biết thông tin về các giáo viên có hệ số lương lớn hơn 2.11

7. Những môn học nào có số đơn vị học trình nhỏ hơn 4?

8. Định mức giờ chuẩn của giáo viên *Lê Văn A* là bao nhiêu?

9. Điện thoại của khoa *CNTT* là gì?

10. Cho biết danh sách các giáo viên làm việc tại khoa *CNTT*.

```
SELECT MAGV,HODEM,TEN
FROM GIAOVIEN, KHOA
WHERE KHOA.TENKHOA = 'Cong nghe thong tin' AND
        GIAOVIEN.MAKHOA=KHOA.MAKHOA
```

11. Cho biết danh sách các lớp thuộc khoa *CNTT*.

12. Lớp *Tin A* trong học kỳ 1 năm học 2001-2002 học những môn học nào và do giáo viên nào giảng dạy.

13. Cho biết họ tên và địa chỉ liên hệ của các sinh viên lớp *Tin A*.

14. Giáo viên *Nguyễn Văn A* trong năm học 2001-2002 dạy những lớp nào với số tiết là bao nhiêu.

15. Cho biết bảng phân công giảng dạy của các giáo viên thuộc khoa *CNTT*.

16. Cho biết điểm thi các môn học của sinh viên *Le Van A*.

17. Cho biết điểm thi môn *Cơ sở dữ liệu* của các sinh viên lớp *Tin A*.

18. Định mức giờ chuẩn lớn nhất của giáo viên khoa *CNTT* là bao nhiêu.

```
SELECT MAX(DINHMUCGIOCHUAN)
FROM GIAOVIEN,KHOA
WHERE KHOA.TENKHOA='Cong nghe thong tin' AND
        GIAOVIEN.MAKHOA=KHOA.MAKHOA
```

19. Cho biết sĩ số của các lớp.

```
SELECT LOP.MALOP,TENLOP,COUNT(MASV)
FROM LOP,SINHVIEN
WHERE LOP.MALOP=SINHVIEN.MALOP
GROUP BY LOP.MALOP,TENLOP
```

20. Những sinh viên nào của lớp *Tin A* phải thi lại môn *Pascal*.

21. Cho biết tổng số tiết dạy của các giáo viên trong năm học 2001-2002

```
SELECT GIAOVIEN.MAGV,HODEM,TEN,SUM(SOTIET)
FROM GIAOVIEN,GIANGDAY
WHERE NAMHOC='2001-2002' AND
        GIAOVIEN.MAGV=GIANGDAY.MAGV
GROUP BY GIAOVIEN.MAGV,HODEM,TEN
```

22. Cho biết họ tên các sinh viên và tổng số đơn vị học trình những môn học mà sinh viên phải thi lại.

23. Giáo viên *Nguyễn Văn B* dạy lớp *Tin A* những môn học nào, vào học kỳ và năm học nào.

24. Cho biết điểm trung bình năm học 2001-2002 của các sinh viên lớp *Tin A*.

25. Những giáo viên nào không thuộc khoa *CNTT* có giảng dạy lớp *Tin A*.

26. Những sinh viên nào của lớp *Tin A* có điểm lần 1 môn *Pascal* cao nhất.

```
SELECT HODEM,TEN,DIEMLAN1
FROM LOP,SINHVIEN,DIEMTHI,MONHOC
WHERE TENLOP='Tin A' AND
```

```
                TENMONHOC='Pascal' AND
                LOP.MALOP=SINHVIEN.MALOP AND
                SINHVIEN.MASV=DIEMTHI.MASV AND
                DIEMTHI.MAMONHOC=MONHOC.MAMONHOC AND
                DIEMLAN1 = ( SELECT MAX(DIEMLAN1)
                             FROM LOP,SINHVIEN,DIEMTHI,MONHOC
                             WHERE TENLOP='Tin A' AND
                             TENMONHOC='Pascal' AND
                             LOP.MALOP=SINHVIEN.MALOP AND
                             SINHVIEN.MASV=DIEMTHI.MASV AND
DIEMTHI.MAMONHOC=MONHOC.MAMONHOC)
```

37. Giáo viên nào có tổng số tiết dạy nhiều nhất.

```
SELECT GIAOVIEN.MAGV,HODEM,TEN,SUM(SOTIET)
FROM GIAOVIEN,GIANGDAY
WHERE GIAOVIEN.MAGV=GIANGDAY.MAGV
GROUP BY GIAOVIEN.MAGV,HODEM,TEN
HAVING SUM(SOTIET)=(SELECT TOP 1 SUM(SOTIET)
                    FROM GIANGDAY
                    GROUP BY MAGV
                    ORDER BY 1 DESC)
```

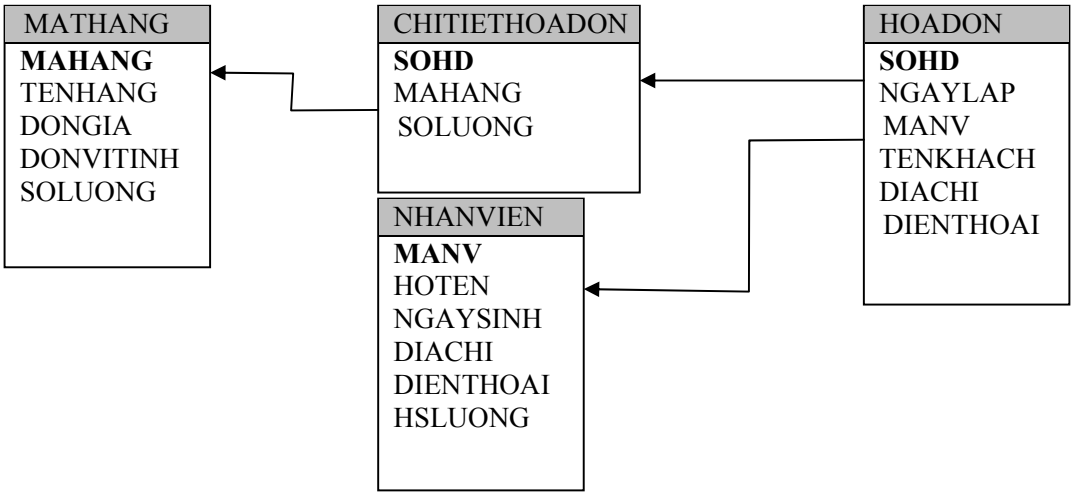
38. Trong năm học 2001-2002, những giáo viên nào dạy vượt chuẩn (có tổng số giờ dạy lớn hơn số giờ chuẩn).

29. Cho biết danh sách sinh viên các lớp và sĩ số của mỗi lớp.

30. Cho biết danh sách các giáo viên của các khoa và tổng số giáo viên của mỗi khoa.

31. Với mỗi giáo viên, hãy cho biết danh sách các môn học được phân công giảng dạy, số lượng môn học được phân công và tổng số tiết dạy.

Các bảng dưới đây dược sử dụng để tổ chức lưu trữ thông tin quản lý công việc bán hàng trong một cửa hàng



Sử dụng câu lệnh SELECT, hãy thực hiện các yêu cầu sau:

32. Xem thông tin trong từng bảng liệu.

33. Hiển thị thông tin về các mặt hàng hiện có trong của hàng và sắp xếp kết quả theo đơn vị tính của mỗi mặt hàng.

34. Cho biết thông tin về các nhân viên hiện đang làm việc trong của hàng.

35. Hãy cho biết những nhân viên nào có cùng số điện thoại và địa chỉ với nhau.

36. Cho biết nhân viên nào đã lập hoá đơn bán hàng cho khách hàng có tên là '*Nguyen Thi C*.

37. Những khách hàng nào đã từng mua *Bơ* của cửa hàng.

38. Cho biết *John Smith* đã mua những mặt hàng nào và tổng số tiền mà ông ta phải trả là bao nhiêu.

39. Hiển thị thông tin về các khách hàng đã mua hàng của cửa hàng trong tháng 6 năm 2001.

40. Thống kê xem trong năm 2000, mỗi mặt hàng bán được với tổng số lượng bao nhiêu mỗi tháng.

41. Cho biết khách hàng nào mua hàng nhiều nhất và với số lượng là bao nhiêu.

42. Mặt hàng *Sữa* trong năm 2001 được khách hàng nào mua nhiều nhất và với số lượng là bao nhiêu.

43. Trong năm 2000, nhân viên nào đã bán được hàng nhiều nhất.

44. Cho biết mỗi hoá đơn bao gồm những mặt hàng nào, số lượng bao nhiêu và tổng số tiền của mỗi hoá đơn.

45. Cho biết trong năm 2000 các mặt hàng đã đượoc bán cho những khách hàng nào và tổng số lượng hàng đã bán được là bao nhiêu.

TÀI LIỆU THAM KHẢO

[1]. Ullman, J.D., *Nguyên lý các hệ cơ sở dữ liệu và hệ tri thức - tập* 1, 2, 3, Trần Đức Quang biên dịch, Nhà xuất bản Thống kê, 1999.

[2]. Hồ Thuần, Hồ Cẩm Hà, *Các hệ cơ sở dữ liệu – Lí thuyết và thực hành - tập 1, 2*, NXB Giáo dục, 2004

[3]. Nguyễn Kim Anh, *Nguyên lý các hệ cơ sở dữ liệu*, NXB Đại học Quốc gia Hà Nội, 2004

[4]. Đỗ Trung Tuấn, *Cơ sở dữ liệu*, NXB Giáo dục, 1998.

[5]. Nguyễn Tuệ, *Giáo trình Cơ sở dữ liệu*, Đại học Quốc gia Hà Nội, 1999.

[6]. Nguyễn Bá Tường, *Cơ sở dữ liệu - Lý thuyết và thực hành*, NXB Khoa học kỹ thuật, 2001.

[7]. Lê Tiến Vương, *Nhập môn cơ sở dữ liệu quan hệ*, NXB Khoa học kỹ thuật, 1996.

[8]. Anstey, D., E. Bertino, E. Marcros, *Advanced Database Technology and Design*, Artech House, 2000.

[9]. Communication of the ACM, *Special Issue on Next-Generation Database Systems*, Vol. 34, No. 10, 1991.

[10]. Date, C. J., *An Introduction to Database Systems*, Wesley, 1994.

[11]. Elmasri, R., S. B. Navathe, *Fundamentals of Database Systems*, 5th Edition, Addison Wesley, 2006.

MỤC LỤC