CHƯƠNG I - CÁC KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU

I.1. CÁC KHÁI NIỆM CƠ BẢN

Có thể nói rằng bất kể lĩnh vực nào của Tin học đều ít nhiều liên quan tới việc tổ chức và khai thác cơ sở dữ liệu. Đặc biệt cơ sở dữ liệu có vai trò rất quan trọng trong hệ thống thông tin.

I.1.1 Dữ liệu (Data)

Dữ liệu là một phần tử hoặc một tập hợp các phần tử mà ta gọi là tín hiệu. Nó được biểu hiện dưới các dạng như hình ảnh, âm thanh, màu sắc, mùi vị... Từ những tín hiệu đó chúng ta có sự hiểu biết về một sự vật, hiện tượng hay quá trình nào đó trong thế giới khách quan thông qua quá trình nhận thức. Trong các dạng dữ liệu thì ngôn ngữ (chữ viết, chữ số, tiếng nói) là dạng dữ liệu phổ biến nhất được dùng trong lĩnh vực tin học (dùng để mô tả, định lượng các đặc tính của đối tượng).

Phạm vi của dữ liệu rất rộng lớn. Trong cuốn bài giảng này chúng ta chỉ đề cập đến dữ liệu trong lĩnh vực của Tin học. Các dữ liệu trong lĩnh vực tin học phải lượng hóa (cân đong đo đếm hay mô tả được).

I.1.2 Cơ sở dữ liệu (Database)

Cơ sở dữ liệu là một tập hợp các dữ liệu về các đối tượng cần được quản lý, được lưu trữ đồng thời trên các vật mang tin của máy tính điện tử và được quản lý theo một cơ chế thống nhất gọi là hệ quản lý (hoặc quản trị) cơ sở dữ liệu nhằm thực hiện ba chức năng sau đây:

(1). Tạo lập dữ liệu

(2). Cập nhật dữ liệu:

- Nạp dữ liệu vào cơ sở dữ liệu.
- Xóa dữ liệu khỏi cơ sở dữ liệu.
- Sửa dữ liệu đã có trong cơ sở dữ liệu.

(3). Tìm kiếm và kết xuất dữ liệu theo yêu cầu.

Nói một cách khác cơ sở dữ liệu là tập hợp các dữ liệu gọn nhất nhưng đầy đủ nhất về các đối tượng cần quản lý đủ đáp ứng tất cả các yêu cầu khai thác đặt ra.

I.1.3 Hệ quản trị cơ sở dữ liệu (Database Management System-DBMS)

Hệ quản trị cơ sở dữ liệu (HQTCSDL) là một hệ thống phần mềm (các chương trình) giúp cho người sử dụng khai thác các CSDL theo các chức năng:

(1). Tạo lập dữ liệu

(2). Cập nhật dữ liệu:

- Nạp dữ liệu vào cơ sở dữ liệu.

- Xóa dữ liệu khỏi cơ sở dữ liệu.
- Sửa dữ liệu có trong cơ sở dữ liệu.

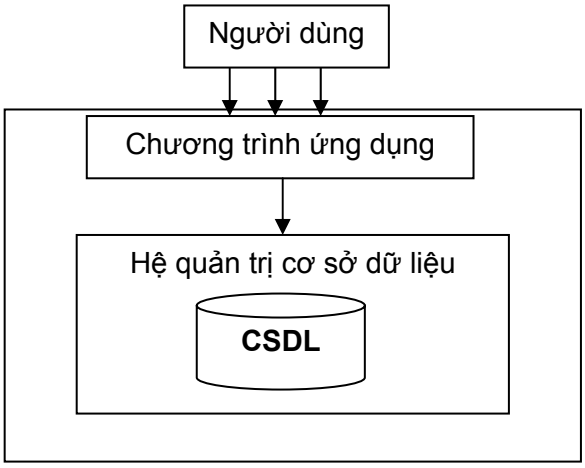
(3). Tìm kiếm và kết xuất dữ liệu theo yêu cầu.

(4). Bảo mật cơ sở dữ liệu.

Cho tới nay có rất nhiều hệ quản trị cơ sở dữ liệu với chất lượng, tính năng và giá cả khác nhau như: họ FOX, DB2, DBASE, ORACLE, SYBASE, PARADOX, INFORMIX, SQL SERVER, MYSQL, POSTGRESQL,... Còn các CSDL là đối tượng quản lý của các HQTCSDL. Chúng được tạo lập và lưu trữ trong các vật mang tin ngoài.

Các HQTCSDL thường cung cấp các công cụ cho phép người dùng thực hiện các thao tác trên. Tuy nhiên do yêu cầu đa dạng và chặt chẽ của người dùng mà để khai thác thông tin hiệu quả thì HQTCSDL chưa đủ mà phải cần tới những phần mềm chuyên dụng (specialized program) giúp cho việc tổ chức, lưu trữ và khai thác cơ sở dữ liệu được hiệu quả hơn. Thường người ta sử dụng các công cụ của HQTCSDL (có thể kết hợp với các ngôn ngữ lập trình) để viết các phần mềm này. Ví dụ hệ thống phần mềm kế toán, quản lý nhân sự tại các cơ quan xí nghiệp, hệ quản lý đào tạo trong các trường Đại học... Mỗi phần mềm như vậy thường chỉ phục vụ cho một lĩnh vực của một đơn vị cụ thể riêng biệt nào đó.

Một Hệ quản trị cơ sở dữ liệu là một hệ thống gồm một hoặc nhiều CSDL và các chương trình ứng dụng dùng để khai thác và xử lý dữ liệu trong CSDL đó.



Dạng so sánh thô thiển sau đây giữa kho vật chất và cơ sở dữ liệu, quản lý kho và hệ quản trị cơ sở dữ liệu, giúp chúng ta nhanh chóng hiểu được một vài thuật ngữ chuyên môn đầu tiên của CSDL.

Mỗi CSDL ứng với một kho. Giống như kho có người quản trị (thủ kho) thì CSDL cũng có người quản trị CSDL. Người quản trị có nhiệm vụ quản lý và theo dõi toàn bộ các thủ tục sau đây:

(1). Nạp dữ liệu vào CSDL ↔ Nạp hàng vào kho.

(2). Xóa dữ liệukhỏi CSDL ↔ Loại bỏ hàng bị hỏng hoặc thanh lý hàng không cần dùng nữa.

(3). Sửa dữ liệu trong CSDL ↔ Sửa lại hàng trong kho.

(4). Tạo lập CSDL ↔ Xây dựng thêm kho.

(5). Tìm kiếm và xuất dữ liệu ↔ Tìm và xuất hàng.

(6). Bảo trì dữ liệu trong CSDL không bị sai hỏng do các truy nhập không hợp phép hoặc các truy nhập không đúng qui cách dẫn đến sự sai lệch, mất mát dữ liệu ↔ Bảo vệ hàng cho khỏi mất mát, hư hỏng.

CSDL có một lớp người sử dụng cũng được phép thực hiện các thao tác (1)-(6) như người quản trị nhưng giới hạn ở một góc độ khai thác dữ liệu. Người sử dụng, tùy theo vai trò và trách nhiệm của mình, được người quản trị cho phép sử dụng một phần nào đó của CSDL và với phần đó họ được phép thực hiện một số thao tác nhất định. Nếu người sử dụng là một nữ nhân viên bán hàng tại một quầy nào đó thì cô ta có thể thông qua máy tính điện tử (MTĐT) theo dõi những mặt hàng bán được tại quầy của mình bao gồm các mục: mã hàng, tên hàng, đơn vị tính, đơn giá, thành tiền, số lượng còn lại, tổng số tiền đã bán... được phép nạp dữ liệu, sửa, xóa dữ liệu, phản ánh đúng biến động của các mặt hàng có trong quầy của mình. Còn các dữ liệu khác, mặc dù được lưu trữ trên cùng một MTĐT, thậm chí được lưu trữ trong cùng một CSDL, sẽ là "khuất" đối với cô ta. Dữ liệu khuất có thể là thông tin về các quầy khác, tổng thu chi của cửa hàng, danh sách và thông tin chi tiết về các quầy khác, tổng thu chi của cửa hàng, danh sách và thông tin chi tiết về các nhân viên…

Tóm lại, mặc dù dữ liệu được lưu trữ chung trong một CSDL nhưng người sử dụng chỉ được nhìn vào CSDL chỉ qua một cái khung (view) và họ cảm thấy CSDL chỉ là một cái khung đó thôi, như là dành riêng cho họ vậy.

Các thao tác (1)-(3) được gọi là thao tác cập nhật. Thao tác (4) được gọi là mô tả dữ liệu. Thao tác (5) được gọi là thao tác tìm kiếm.

Toàn bộ các máy móc, thiết bị, qui định, hướng dẫn phục vụ cho việc thực hiện các thao tác (1)-(6) đối với các kho hàng tạo thành hệ quản lý kho, trong hệ thống này có thể có các máy nâng đỡ, bốc xếp hàng, các loại phiếu và chứng từ xuất-nhập kho, các cách thức tìm hàng trong kho… Tương tự, ta có hệ quản trị CSDL – là chương trình máy tính giúp cho ta thực hiện các thao tác (1)-(6).

I.1.4 Sự cần thiết của cơ sở dữ liệu

Trước đây việc khai thác dữ liệu bằng các công cụ thủ công bằng giấy tờ mất nhiều thời gian và hiệu quả không cao. Với sự phát triển của công nghệ thông tin (cả phần cứng và phần mềm) đã cho ra đời một ngành công nghệ mới: công nghệ thông tin. Thông tin thường được tinh lọc từ dữ liệu. Ban đầu việc tổ chức, khai thác dữ liệu chưa được nghiên cứu đúng mức dẫn tới việc khai thác chúng vẫn còn hạn chế do thiếu tính nhất quán và bảo mật cũng như khả năng lưu trữ chưa nhiều. Thậm chí nếu không nghiên cứu thấu đáo sẽ gây nên tính mâu thuẫn trong dữ liệu và dẫn đến những hậu quả không lường. Có thể cùng một đối tượng dữ liệu do nhiều đơn vị cùng khai thác nên có khi được tổ chức lưu trữ tại nhiêu nơi, gây nên sự trùng lắp, tốn kém và nếu không cập nhật kịp thời sẽ có sự không nhất quán, làm cho hiệu quả khai thác thấp. Dần dần cơ sở dữ liệu được nghiên

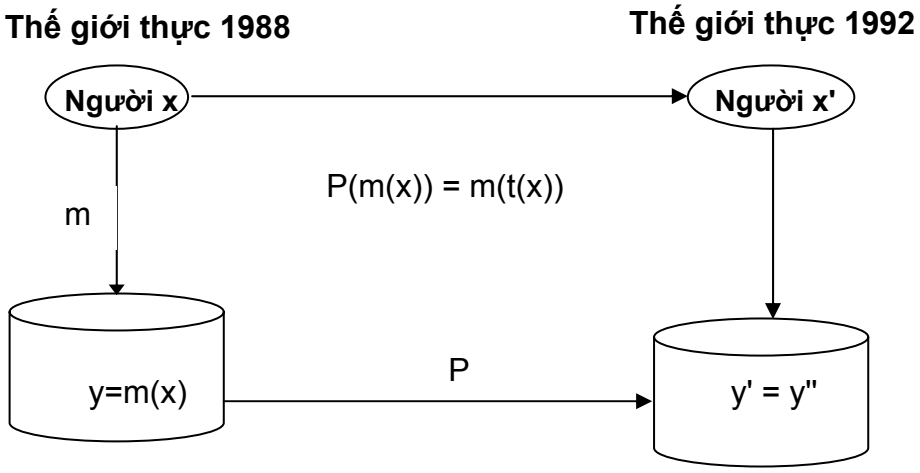
cứu nghiêm túc đáp ứng nhu cầu khai thác ngày càng cao của những nhóm người sử dụng khác nhau. Có được một cơ sở dữ liệu hợp lý, đáp ứng tất cả những đòi hỏi của tất cả những nhóm người trong mỗi hệ thống thông tin là điều đặt ra cho những người làm tin học nói chung và người những người nghiên cứu cơ sở dữ liệu nói riêng.

Nói tóm lại CSDL là bộ phận không thể thiếu được trong các hệ lưu trữ và tìm kiếm thông tin, các hệ thống quản lý kinh tế các ngành, các cấp, các hệ thống quản lý kho tàng, tư liệu, các hệ thống phục vụ công cộng như ngân hàng, bán vé máy bay và các phương tiện giao thông, các hệ thống thiết kế tự động. v. v.

I.2. CÁC MÔ HÌNH CSDL

I.2.1 Mô hình hóa trong tin học

Mô hình hóa là sự lập tương quan giữa các đặc trưng, thuộc tính của đối tượng với các phần tử của một tập nào đó sao cho thông tin về động thái của các phần tử trong tập này thể hiện sự vận động của đối tượng được mô hình hóa.



Thông tin của đối tượng x (năm 1988)

Họ & tên: Vũ Trung
Mã: 02
Năm sinh: 1960
Lương: 390.000
Số con: 1
Chức vụ: Nhân viên

Thông tin của đối tượng x (năm 1992)

Họ & tên: Vũ Trung
Mã: 02
Năm sinh: 1960
Lương: 505.000
Số con: 2
Chức vụ: trưởng phòng

Hãy xét một phần tử x - nhân viên của một cơ quan nào đó. Đó là một đối tượng cụ thể của thế giới thực. Chúng ta mô tả đố tượng x thông qua một cơ chế *m* như sau: *m* - liệt kê các thuộc tính của cán bộ x, cụ thể là họ và tên, mã (hồ sơ), năm sinh số con và chức vụ. Cần bao nhiêu thuộc tính và thuộc tính nào là do mục tiêu của việc mô hình hóa định đoạt. Nếu cần quản lý học sinh của nhà trường thì rõ ràng là không cần đến thuộc tính lương, số con và chức vụ. Chúng có thể được thay bằng các thuôc tính khác như: họ và tên phụ huynh, địa chỉ gia đình v. v.

Giả sử thời điểm mô tả x là năm 1988 và thông tin về x được lưu trong một CSDL. Đến năm 1992, do thời gian thay đổi, đối tượng x cũng thay đổi theo: lương tăng thêm, số con tăng thêm và nhận chức vụ mới v. v. Một hàm t đã biến đổi x sang hình trạng mới x'

$$x' = t(x)$$

Tại năm 1988, ánh xạ m, như đã biết, lập thông tin y cho x

$$y = m(x)$$

Nếu hệ CSDL lưu trữ thông tin về x có cơ chế P để cập nhật (sửa thông tin cho phù hợp với thực tế) thì P sẽ biến đổi y thành y'. Ta có:

$$y' = P(y) = P(m(x))$$

Nếu tại năm 1992 ta lại dùng cơ chế m để lấy thông tin cho x (ứng với hình trạng x') thì sẽ được:

$$y' = m(x') = m(t(x))$$

Cặp ánh xạ <m, p> được gọi là cơ chế mô hình hóa. Dễ thấy rằng cơ chế <m, p> làm việc đúng đắn nếu như nó theo dõi đúng đối tượng, tức là:

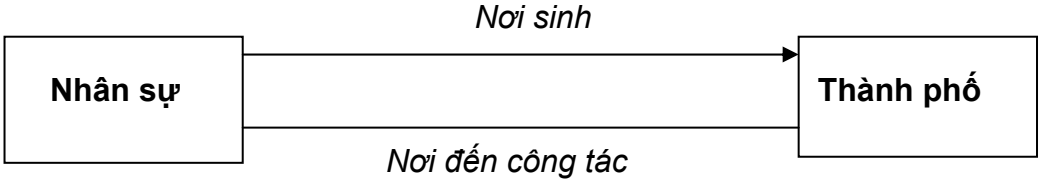
$$y' = y'' \text{ hay}$$

$$P(m(x)) = m(t(x)) \quad (*)$$

Đẳng thức (*) được gọi là điều kiện giao hoán của mô hình hóa <m, p>.

Chúng ta lần lượt xét ba loại mô hình – ba cách tiếp cận khi thiết kế các hệ quản trị CSDL.

I.2.2 Mô hình mạng



Mô hình mạng được xây dựng trên các *tập dữ liệu* và các *quan hệ*.

- **Tập dữ liệu** được tạo từ những dữ liệu cùng một kiểu gọi là bản ghi. Mỗi bản ghi được tạo bởi các trường. Theo hình trên ta có hai tập dữ liệu là *Nhân sự* và *Thành phố* với các bản ghi tương ứng như sau:
    - Mỗi phần tử của tập *nhân sự* được mô tả qua 6 trường (6 thuộc tính):

HỌ VÀ TÊN:
MÃ:
NĂM SINH:
LƯƠNG:
SỐ CON:
CHỨC VỤ:

- Mỗi phần tử (thành phố cụ thể) của tập **thành phố** được miêu tả qua 4 thuộc tính:

MÃ THÀNH PHỐ: TÊN GỌI: DIỆN TÍCH: KHOẢNG CÁCH TỚI THỦ ĐÔ:

- **Quan hệ** xác lập một tương quan - ánh xạ giữa hai tập dữ liệu. Theo hình trên ta có:

Quan hệ **NƠI SINH**: Nhân sự $\in$ Thành phố: Mỗi nhân sự cụ thể của tập *Nhân sự* có một nơi sinh cụ thể. Giữa hai tập trên còn có thể có quan hệ thứ hai, chẳng hạn **NƠI ĐẾN CÔNG TÁC** cho biết cán bộ x đến làm việc tại những thành phố nào.

Các quan hệ được phân loại theo kiểu *ánh xạ đơn trị*, nó được kí hiệu là 1:1 (mỗi người có một nơi sinh), còn NƠI ĐẾN CÔNG TÁC là *ánh xạ không đơn trị*, kí hiệu là 1:N - một người có thể tổ chức chuyến công tác ở nhiều thành phố.

Để tạo lập một CSDL theo mô hình mạng chúng ta cần các thao tác sau:

1. *Tạo lập một tập bao gồm việc khai báo tên tập và mô tả các thuộc tính của tập*.
   Ví dụ:

```
CREATE SET
    SET NAME:Nhân sự
        ATTRIBUTE 1: HỌ VÀ TÊN
            TYPE: string[30]
        ATTRIBUTE 2: MÃ
            TYPE: integer (2)
         ATTRIBUTE 3: NĂM SINH
            TYPE: integer (2)
        ATTRIBUTE 3:NĂM SINH
            TYPE: integer (2)
        . . . .
CREATE SET
    SET NAME:Thành phố
        ATTRIBUTE 1: MÃ THÀNH PHỐ
            TYPE: string[3]
        ATTRIBUTE 2: TÊN GỌI
            TYPE: string[30]
        ATTRIBUTE 3: DIỆN TÍCH
            TYPE: integer (12)
        ATTRIBUTE 3:KHOẢNG CÁCH TỚI THỦ ĐÔ
            TYPE: integer (5)
            . . . .
```

2. *Thiết lập một quan hệ giữa hai tập*: bao gồm việc khai báo tên quan hệ, tên tập nguồn và tên tập đích và kiểu quan hệ.

   Ví dụ:

   SET RELATION: NƠI SINH

   FROM: Nhân sự

   TO: Thành phố

   TYPE: 1:1

   Các thao tác đối xứng của CREATE SET và SET RELATION sẽ là DELETE SET và DELETE RELATION

   Các thao tác phụ trợ khác có thể là:

   - Sửa lại tên (tập, quan hệ, thuộc tính)
   - Sửa lại kiểu (thuộc tính, quan hệ)
   - Thêm, bớt thuộc tính vào một tập đã có v. v.

Ta xét một ví dụ minh họa việc truy nhập tới CSDL mạng.

Ví dụ: Cho biết tổng số lương của những cán bộ có nơi sinh ở Hải Phòng.

```
GET. Nhân sự      /* mở tập nhân sự */
GET. Thành phố /* và mở tập thành phố */
S:= 0                /* đặt tổng lương bằng 0 */
FOR each  x  in Nhân sự DO
        IF (NƠI SINH(x).TÊN GỌI= 'HAI PHONG') THEN
                S:= S +x.LƯƠNG
        ENDIF
ENDFOR
CLOSE ALL /* đóng tất cả các tập đã xét 0 */
WRITE ('Tổng lương: ',S);
```

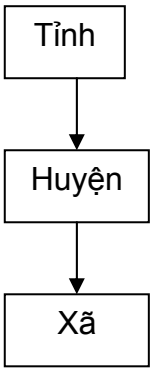
Các thao tác trong hệ cơ sở dữ liệu có thể thực hiện ở chế độ hội thoại trực tiếp người – máy (Ví dụ các thao tác tạo lập) hoặc ở chế độ chương trình (ví dụ truy nhập nói trên được viết dưới dạng ngôn ngữ tựa Pascal). Sử dụng chế độ nào hoặc ngôn ngữ gì không quan trọng bằng sự hiểu biết bản chất của mô hình và các thao tác để có thể tự thiết kế và cài đặt hệ cơ sở dữ liệu hoặc sử dụng một hệ có sẵn theo yêu cầu đòi hỏi.

I.2.3 Mô hình phân cấp

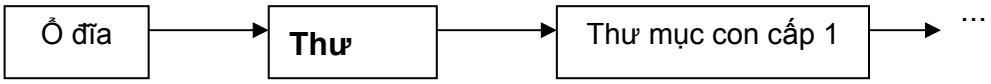
Đây là trường hợp riêng của mô hình mạng, trong đó khái niệm tập dữ liệu được giữ nguyên còn khái niệm quan hệ được giới hạn ở kiểu phân cấp. Giữa hai tập dữ liệu bất kỳ có tối đa một quan hệ và quan hệ này tuân thủ trật tự trên dưới. Loại mô hình phân cấp phù hợp với những tổ chức có hình thức phân cấp.

Ví dụ:

Tổ chức quản lý hành chánh:



Tổ chức thư mục:



Cấu trúc phân cấp là cấu trúc cây, nên để truy cập tới một đối tượng trong mô hình phân cấp là đi từ gốc (đỉnh đầu tiên) đến phần tử cần xét.

I.2.4 Mô hình quan hệ

Mô hình quan hệ lần đầu tiên được Codd – một nhân viên của hãng IBM đề xuất năm 1970. Một CSDL quan hệ được tạo lập từ các quan hệ có hình ảnh trực quan là các bảng (table). Mỗi một bảng bao gồm các cột được gọi là các thuộc tính và các dòng được gọi là các bộ.

Ví dụ: CSDL bán hàng lưu trữ thông tin về các phát sinh bán hàng tại các của hàng của một công ty được tạo thành từ các quan hệ sau đây:

Quan hệ **MẶT HÀNG**

MÃ HÀNG	TÊN HÀNG	ĐƠN VỊ TÍNH

Quan hệ **KHÁCH HÀNG**

MÃ KHÁCH	HỌ TÊN KHÁCH	ĐỊA CHỈ KHÁCH

Quan hệ **CỬA HÀNG**

MÃ CỬA HÀNG	TÊN CỬA HÀNG	ĐỊA CHỈ CỬA HÀNG

Quan hệ **GỐC HÓA ĐƠN** (Hóa đơn xác định bởi số thứ tự của nó được bán ngày nào, cho khách nào, từ cửa hàng nào, tỷ lệ và tiền thuế giá trị gia tăng là bao nhiêu và thuộc quyển hóa đơn nào).

STT_HĐ	NGÀY BÁN	MÃKHÁCH	MÃ CỬA HÀNG	TỶLỆVAT	THUẾ GTGT	QUYỂN_HĐ

Quan hệ **CHI TIẾT HÓA ĐƠN** (Hóa đơn bán những mặt hàng nào, số lượng, đơn giá và số tiền tương ứng là bao nhiêu)

STT_HĐ	MÃ HÀNG	SỐ LƯỢNG	ĐƠN GIÁ	THÀNH TIỀN

Các đặc điểm của mô hình quan hệ:

- Tự nhiên, gần với quan niệm thông thường.
- Có cơ sở toán học chặt chẽ cho phép áp dụng rộng rãi các công cụ đại số và logic.
- Ngôn ngữ thao tác trong sáng và có khả năng tổ hợp cao.
- Dễ đảm bảo tính an toàn dữ liệu: có thể đặt mật khẩu truy nhập ở nhiều mức: mức quan hệ, mức thuộc tính, mức bộ, mức thuộc tính – bộ.
- Dễ cập nhật tới các đơn vị dữ liệu.
- Dễ đảm bảo tính độc lập dữ liệu.

I.3. NGÔN NGỮ DỮ LIỆU

I.3.1 Khái niệm về ngôn ngữ

Ngôn ngữ là phương tiện giao tiếp giữa người – người, người – máy hoặc máy – máy. Có hai loại ngôn ngữ: ngôn ngữ tự nhiên và ngôn ngữ hình thức.

I.3.2 Ngôn ngữ tự nhiên

Ngôn ngữ tự nhiên là ngôn ngữ của những bộ tộc người, được phát triển và hoàn thiện từ khi xuất hiện con người nguyên thủy đầu tiên. Ngôn ngữ tự nhiên được thể hiện qua tiếng nói, chữ viết hoặc động tác (ngôn ngữ của những người câm điếc) và tuân theo những quy tắc nhất định được gọi là ngữ pháp.

I.3.3 Ngôn ngữ hình thức

Ngôn ngữ hình thức là tập hợp các ký hiệu và quy định do con người đặt ra. Thông thường mỗi lĩnh vực có những ngôn ngữ hình thức riêng (trong toán học, trong vật lý, trong hóa học, sinh học,vv..). Trong tin học ngôn ngữ hình thức là công cụ giao tiếp giữa người với máy và giữa máy và máy.

I.3.3.1 Ngôn ngữ dữ liệu:

Các hệ quản trị cơ sở dữ liệu bao giờ cũng cung cấp một ngôn ngữ hình thức làm công cụ giao tiếp giữa người sử dụng và hệ thống. Đó là ngôn ngữ dữ liệu. Các ngôn ngữ dữ liệu thường sử dụng từ khóa trong ngôn ngữ tự nhiên với cú pháp chặt chẽ, được người sử dụng đưa vào hệ thống bằng các phương tiện như bàn phím, màn hình và dĩ nhiên là cả bộ xử lý với phần cứng và phần mềm liên quan. Ngôn ngữ dữ liệu bao gồm ba lớp ngôn ngữ con ứng với ba chức năng chủ yếu của hệ cơ sở dữ liệu: là ngôn ngữ con mô tả dữ liệu, ngôn ngữ con cập nhật dữ liệu, và ngôn ngữ con tìm kiếm dữ liệu.

I.3.3.1.1 Ngôn ngữ con mô tả dữ liệu (Data Definition Language – DDL)

Ngôn ngữ con mô tả dữ liệu giúp người sử dụng khai báo tên đối tượng, cấu trúc của đối tượng và các quan hệ của đối tượng với các đối tượng khác.

Ví dụ:

```
CREATE TABLE KHÁCH HÀNG          /* Tạo bảng khách hàng */
    ATTRIBUTE 1 MÃ KHÁCH         /* Thuộc tính thứ nhất: mã khách hàng */
        TYPE INT(2)              /* Kiểu số nguyên 2 chữ số */
    ATTRIBUTE 2 HỌ TÊN           /* Thuộc tính thứ hai: họ tên khách hàng */
        TYPE STR(30)             /* Kiểu chuỗi ký tự: 30 ký tự */
    ATTRIBUTE 3 ĐỊA CHỈ          /* Thuộc tính thứ ba: địa chỉ của khách
hàng */
        TYPE STR(30)             /* Kiểu chuỗi ký tự: 30 ký tự */
    ATTRIBUTE 4 KHỐI LƯỢNG       /*T. tính thứ tư: khối lượng của khách
hàng */
        TYPE REAL(6,2)           /* Kiểu số thực: 6 chữ số trong đó có 2 số
lẻ */
    ATTRIBUTE 5 TỈNH             /*T. tính thứ năm: tỉnh/TP của khách hàng
*/
        TYPE STR(20)             /* Kiểu chuỗi ký tự: 20 ký tự */
END CREATE
```

I.3.3.1.2 Ngôn ngữ con cập nhật dữ liệu (Data Update Language – DUL)

Ngôn ngữ con cập nhật dữ liệu giúp người dùng thực hiện các thao tác sau:

- Đổi tên, đổi kiểu các đối tượng đã khai báo.
- Thêm hoặc bớt một số thành phần vào cấu trúc đã có
- Thêm, sửa, xóa dữ liệu trong cơ sở dữ liệu.

Ví dụ:

- RENAME TABLE KHÁCH HÀNG/KHÁCH /* Đổi tên bảng KHÁCH HÀNG thành KHACH */
- RENAME ATTRIBUTE MÃ KHÁCH/MAKH /* Đổi thuộc tính MÃ KHÁCH thành MAKH */
- ADD ATTRIBUTE TEL_NUM TO TABLE KHACH TYPE STR(10) /* Thêm thuộc tính số điện thoại của khách vào bảng KHACH */

- DELETE ATTRIBUTE KHỐI LƯỢNG FROM KHACH /* Xóa thuộc tính khối lượng khỏi cấu trúc bảng KHACH */
- ADD TUBLES TO TABLE KHACH /* Thêm các bộ sau đây vào bảng KHACH */

  <07, "Trần Phung Phí","123 - Thiên Đàng","103183">

  <95, "Nguyễn Tiến Tùng","99 – Trần Thái Hoàng","0918252819">

I.3.3.1.3 Ngôn ngữ con truy nhập dữ liệu (hay còn được gọi là ngôn ngữ hỏi (Query Langguage – QL)

Ngôn ngữ hỏi là phương tiện giúp người sử dụng tìm kiếm thông tin trong cơ sở dữ liệu. Thông thường trong ngôn ngữ hỏi có ba mục khai báo chủ yếu sau:

- Mục đường dẫn: cho biết vị trí của các đối tượng cần xử lý. Đối tượng thường được định danh bằng tên, nếu biết tên thì biết nơi lưu trữ đối tượng.
- Mục điều kiện: chứa điều kiện để lọc dữ liệu trong cơ sở dữ liệu.
- Mục khuôn ra: quy định khuôn dạng dữ liệu cần đưa ra.

Ví dụ:

Câu hỏi "Cho biết họ tên và địa chỉ khách hàng ở Cần Thơ" có thể có các mục sau:

- Mục đường dẫn: Cơ sở dữ liệu Bán hàng, bảng KHACH.
- Mục điều kiện: Tỉnh = "Cần Thơ"
- Mục khuôn ra: HỌ TÊN, ĐỊA CHỈ.

I.3.3.2 Phân loại ngôn ngữ

I.3.3.2.1 Phân loại theo hình thức thể hiện:

Ngôn ngữ là hình thức giao tiếp cho nên có hai dạng chính là hội thoại hay chương trình.

I.3.3.2.2 Chế độ hội thoại

Chế độ hội thoại hiểu nôm na là hỏi – trả lời trong thời gian thực. Một hình thái thể hiện của chế độ hội thoại được triển khai trong những năm gần đây là trao đổi theo thực đơn (menu). Hệ thống đưa ra một thực đơn gồm những "món" có sẵn. Người sử dụng, tùy theo mục đích của mình sẽ chọn một "món " – một chức năng nào đó. Tiếp theo, nếu món đó là đơn giản, hệ thống sẽ thực hiện và thông báo kết quả. Nếu không, tức là món vừa chọn có nhiều khả năng, hệ thống lại đưa ra một thực đơn thứ cấp để người sử dụng tiếp tục lựa chọn một khả năng trong số đó.

I.3.3.2.3 Chế độ chương trình:

Trong chế độ chương trình người sử dụng viết một dãy lệnh dưới dạng một chương trình rồi giao cho hệ thống thực hiện.

I.3.3.3 Phân loại theo theo kiểu (cấu trúc):

I.3.3.3.1 Kiểu thủ tục (procedure):

Các câu hỏi kiểu thủ tục thường chỉ rõ thủ tục tìm kiếm đối tượng mong muốn. Trong kiểu thủ tục thường bao gồm hai kiểu thành phần là kiểu đại số và kiểu logic.

Kiểu đại số: sử dụng các biểu thức đại số.

Kiểu logic: sử dụng các phép toán logic mệnh đề và logic vị từ.

I.3.3.3.2 Kiểu phi thủ tục (non-procedure):

Kiểu phi thủ tục thường được gọi là kiểu mô tả. Câu hỏi thuộc lớp ngôn ngữ này chỉ mô tả những gì cần có chứ không trình bày cách thức đạt tới. Cơ chế tìm kiếm do hệ thống đảm nhiệm dựa trên cách thức tổ chức cơ sở dữ liệu. Điển hình của lớp ngôn ngữ phi thủ tục là Prolog.

Một hướng nghiên cứu quan trọng của tin học là cài đặt ngôn ngữ tự nhiên cho hệ thống. Dĩ nhiên, giao tiếp bằng ngôn ngữ tự nhiên chỉ thực sự thuận tiện khi dùng tiếng nói. Chính vì vậy mà hướng nghiên cứu này liên quan tới lĩnh vực nhận dạng, phân tích và tổng hợp tiếng nói.

I.3.4 Chuyên ngành cơ sở dữ liệu

Chuyên ngành cơ sở dữ liệu là một lĩnh vực của tin học nghiên cứu các cơ chế, nguyên lý, phương pháp tổ chức các nhóm dữ liệu trên các vật mang tin ngoài (các loại đĩa trống từ, đĩa từ, đĩa quang học v. v) nhằm phục vụ cho việc khai thác dữ liệu trong các hệ thống thông tin (Ví dụ: các hệ thống lưu trữ và tra cứu thông tin các hệ quản lí xí nghiệp hoặc ngành v.v).

Trong số ba mô hình, ba cách tiếp cận cho việc tổ chức và khai thác các cơ sở dữ liệu là mô hình phân cấp, mô hình mạng và mô hình quan hệ, thì mô hình quan hệ được quan tâm hơn cả vào khoảng vài chục năm trở lại đây. Các hệ cơ sở dữ liệu quan hệ được thiết kế và cài đặt trên tất cả các hệ máy: hệ micro, hệ mini, hệ mainframe và các hệ siêu máy tính. Trong các chương trình xây dựng máy tính thế hệ mới, mô hình quan hệ được quan tâm đáng kể.

Sở dĩ mô hình quan hệ được phát triển rộng rãi như vậy là vì nó được xây dựng trên một cơ sở toán học chặt chẽ - lý thuyết về các quan hệ và có hình ảnh trực quan gần với các quan niệm thông thường của người dùng. Các ngôn ngữ thao tác trên các CSDL quan hệ khá dễ học và có hiệu suất phục vụ cao.

Mục đích của cuốn bài giảng này là cung cấp một cách nhìn tổng quát về các hệ quản trị CSDL và một vài khía cạnh lý thuyết và thực tiễn quan trọng nhất của mô hình quan hệ.

Cũng cần lưu ý rằng mọi mô hình CSDL đều có những đòi hỏi giống nhau, như đảm bảo tính toàn vẹn dữ liệu (không phát sinh các dữ liệu mâu thuẫn, không làm mất dữ liệu), đảm bảo tính độc lập của chương trình khai thác đối với tổ chức vật lý cụ thể của dữ liệu, đảm bảo sự tối ưu trong lưu trữ và khai thác v. v. Điều đáng quan tâm là dùng mô hình quan hệ chúng ta dễ dàng biểu đạt các vấn đề một cách chặt chẽ.

CHƯƠNG II - MÔ HÌNH QUAN HỆ VÀ ĐẠI SỐ QUAN HỆ

II.1. MÔ HÌNH QUAN HỆ

II.1.1 Định nghĩa quan hệ

II.1.1.1 Tích Đề-các (Decastersian)

Cho các tập hợp D_1, D_2, ... D_n đều khác rỗng. Tích Đề-các của n tập hợp D_1, D_2, ... D_n ký hiệu là **x D_i**= D_1 x D_2 x .... x D_n là tập hợp các phần tử (a_1, a_2,..., a_n) trong đó a_i thuộc D_i với mọi i=1,2,...,n.

II.1.1.2 Quan hệ

Một quan hệ R xây dựng trên các tập hợp D_1,D_2,..., D_n là tập con của tích Đề các **x Di**.

Ví dụ:

Nếu: *Tên_KháchHàng* = {Diễm, Nghe, Bảo, Cường }

Tên_ĐườngPhố = {3/2, 30/4, CMT8}

Tên_ThànhPhố = {Cần Thơ, Hà Nội, Sài Gòn}

Thì: R = { (Diễm, 3/2, Cần Thơ),

(Nghe, 30/4, Hà Nội),

(Bảo, 30/4, Hà Nội),

(Cường, CMT8, Sài Gòn) }

Là một quan hệ trên tích *Tên_KháchHàng* x *Tên_ĐườngPhố* x*Tên_ThànhPhố*

II.1.2 Mô hình CSDL quan hệ

Mô hình CSDL quan hệ dựa trên khái niệm quan hệ trong đó quan hệ được xem là xây dựng từ tập các thuộc tính.

II.1.2.1 Thuộc tính (attribute):

Mỗi dữ liệu, mỗi đối tượng được khảo sát đều có những đặc tính riêng biệt. Các đặc tính riêng biệt này được gọi là thuộc tính.

- Thuộc tính thường được ký hiệu bằng các chữ cái đầu trong bảng chữ cái: A, B, C (hay A_1, A_2,…)
- Tập hợp các thuộc tính thường được ký hiệu bằng chữ cái cuối trong bảng chữ cái: X, Y, Z.

  Ví dụ: X = {A, B, C} hay để cho đơn giản người ta viết X =ABC
- Mỗi thuộc tính phải thuộc một kiểu dữ liệu. Tập hợp các phần tử mà thuộc tính A có thể lấy giá trị gọi là *miền trị của thuộc tính* A, kí hiệu: **dom(A)** *(dom : Domain)*

II.1.2.2 Quan hệ (Relation) và bộ (tuple) trong CSDL quan hệ

Định nghĩa: Cho tập hữu hạn U={A_1, A_2, ..., A_n} (n ≥ 1) là tập hợp các thuộc tính, với mỗi $A_i \in U$ có dom(Ai) = D_i tương ứng.

- Một quan hệ R trên tập thuộc tính U, kí hiệu: R(U)

  $R(U) = R(A_1, A_2, ..., A_n) = \{t: U \rightarrow \cup d_i / \forall A_i \in U, t(A_i) \in d_i\}$

- Một ánh xạ $t : U \rightarrow \cup d_i$ được gọi là một bộ.

  Ví dụ: Quan hệ SINHVIEN (MASV,HOTEN, NSINH,QUE) với

  dom (MASV) : 999
  dom (HOTEN): A(30)
  dom (NSINH) : 9999
  dom (QUE) : A(15)

  t = <001, Le Van Trung, 1969, Hai Phong> là một bộ của quan hệ SINHVIEN

- Với tập $X \subseteq U$ và bộ $t \in R(U)$

  thì t.X = t[X] = t|X = là một thu hẹp của t trên tập X.

  Ví dụ: X = {MASV, HOTEN} → t [X] = t.X = <001, Le Van Trung>

II.1.2.3 Bậc (dimention), lực lượng (card):

- Số thuộc tính n của quan hệ R được gọi là bậc (hay ngôi) của R.
- Lực lượng của quan hệ R là số bộ mà R đang chứa.

II.1.2.4 Lược đồ quan hệ (Schema)- tân từ (predicate):

- Lược đồ quan hệ là tập gồm tên và kiểu của các thuộc tính.

  Kí hiệu: $R(A_1 : D_1, A_2 : D_2,..., A_n : D_n)$ hay $R(A_1, A_2,..., A_n)$

- Tân từ là đoạn văn bản dùng để mô tả ngữ nghĩa của lược đồ quan hệ.

  Ví dụ:

  Tân từ «*Hóa đơn xác định bởi số thứ tự của nó được bán ngày nào, cho khách nào, từ cửa hàng nào, tỷ lệ và tiền thuế giá trị gia tăng là bao nhiêu và thuộc quyển hóa đơn nào*» mô tả ngữ nghĩa cho quan hệ **GỐC_HÓA_ĐƠN**(STT_HĐ, NGÀY BÁN, MÃKHÁCH, MÃ CỬA HÀNG, TỶLỆVAT, THUẾ GTGT, QUYỂN_HĐ)

II.1.2.5 CSDL quan hệ:

CSDL quan hệ là một tập các quan hệ biến thiên theo thời gian.

Trong CSDL		**Trong lập trình**
Lược đồ quan hệ	↔	Kiểu biến
Quan hệ	↔	Biến
Một thể hiện của quan hệ	↔	Một giá trị của biến

II.1.3 Một số thao tác cơ bản trên CSDL

- Tạo quan hệ:
  RCREATE (<Tên quan hệ>)
  ATTR <Tên thuộc tính> TYPE <Kiểu>
  [ATTR <Tên thuộc tính> TYPE <Kiểu> FROM <Tên quan hệ>]
  END
- Mở CSDL:
  DBOPEN <Tên CSDL>
- Mở/ Đóng quan hệ:
  ROPEN <Tên quan hệ> | RCLOSE <Tên quan hệ>
- Hiển thị:
  LIST
- Thêm thuộc tính:
  RADD <Tên thuộc tính> TYPE <Kiểu>
- Thêm bộ / Xóa bộ:
  TADD <Bộ> | TDEL <Bộ>
- Đổi tên thuộc tính:
  AREN <Tên thuộc tính cũ> / <Tên thuộc tính mới>

II.2. ĐẠI SỐ QUAN HỆ

II.2.1 Định nghĩa đại số quan hệ

Đại số quan hệ là một bộ α = <M, P> trong đó:

M là tập các quan hệ cho trước

P là tập các phép toán cơ bản sau đây:

1. Phép chọn (Selection) ()
2. Phép chiếu (Projection) [ ]
3. Phép hợp (Union) $\cup$ hay +
4. Phép giao (Intersection) $\cap$ hay .
5. Phép trừ (Subtraction) - hay \
6. Phép chia (Division) $\div$ hay /
7. Phép nối kết tự nhiên (Natural Join) *
8. Phép tích Đề-các (Decastersian) x
9. Phép θ kết nối (delta - join)

II.2.2 Các phép toán cơ bản của đại số quan hệ:

CSDL dùng trong các ví dụ:

SINHVIEN (MASV, HOTEN, NSINH, QUEQUAN, HOCLUC)

DETAI (MADT, TENDT, KINHPHI, CNHIEM)

SV_DT (MASV, MADT, NOI_AD, KETQUA)

II.2.2.1 Phép chọn (Selection): kí hiệu ()

Mục đích của phép chọn là xây dựng một tập con gồm các bộ của quan hệ đã cho, thỏa *biểu thức logic* cho trước. Biểu thức logic phát biểu trên U gồm có:

- Các thuộc tính hoặc các hằng
- Các phép toán so sánh số học: $<, \leq, >, \geq, =, \neq$
- Các phép logic: $\wedge, \vee, \neg$
- Các dấu ngoặc (và).

Định nghĩa:

Cho t là một bộ trong quan hệ R, và E là một *biểu thức logic* phát biểu trên tập các thuộc tính của quan hệ R. Ta nói bộ t thỏa mãn biểu thức E, kí hiệu t(E), nếu sau khi thay mọi thuộc tính A trong E bằng giá trị t.A ta được một công thức logic mệnh đề đúng.

Ví dụ:

t = <MSSV:01, HOTEN: Lê Văn Trung, NSINH: 1969, QUEQUAN : Hải Phòng, HOCLUC : Khá>

Thì t là một bộ thỏa biểu thức *E = NSINH <1970 $\wedge$ QUEQUAN = 'Hải Phòng'*

Định nghĩa:

Cho quan hệ R(U) và biểu thức chọn E trên U. Phép chọn quan hệ R theo điều kiện E cho ta quan hệ P với tập thuộc tính U và các bộ của R thỏa E.

$P = R(E) = \{ t \in R \mid t(E) \}$

Ví dụ 1:

Cho biết thông tin của những sinh viên tuổi < 22 và học lực khá.

Điều kiện E= (year() - NSINH < 20) $\wedge$ HOCLUC ='Khá'

⇨ P = **SINHVIEN(**(year() – NSINH) < 20 $\wedge$ HOCLUC ='Kha')

Ví dụ 2:

Cho biết thông tin của các đề tài do 'An' chủ nhiệm và có kinh phí từ 4 triệu trở lên.

Điều kiện E= (CNHIEM='An' $\wedge$ KINHPHI >=4)

⇨ P = **DETAI(**CNHIEM='An' $\wedge$ KINHPHI >=4)

II.2.2.2 Phép chiếu (Projection): kí hiệu [ ]

Định nghĩa:

Cho quan hệ R(U) và tập thuộc tính $X \subseteq U$. Phép chiếu quan hệ R trên X cho ta *quan hệ P có tập thuộc tính là X* và các bộ là hạn chế trên X của các bộ trong R.

$P = R[X] = \{ t.X \mid t \in R \}$

Chú ý:

Vì quan hệ được định nghĩa như một tập các bộ, cho nên trong quan hệ không thể có hai bộ giống nhau hoàn toàn.

Ta có thể xác định P bằng hai bước sau:

- Bước 1: Xóa đi các thuộc tính không có trong X của R.
- Bước 2: Lược bớt các bộ giống nhau, chỉ giữ lại một bộ trong số các bộ giống nhau.

Ví du 1:

Cho biết danh sách HOTEN và QUEQUAN của các sinh viên.

⇨ SINHVIEN [HOTEN,QUEQUAN]

Ví du 2:

Cho biết tên các đề tài do 'An' chủ nhiệm.

⇨ DETAI (CNHIEM='An')[TENDT]

II.2.2.3 Tích Đề-Các: kí hiệu x

Định nghĩa:

Cho hai quan hệ R(U) và S(V), trong đó $U \cap V = \varnothing$. Tích Đề-Các của R và S là một quan hệ P, với tập thuộc tính là $U \cup V$ được định nghĩa như sau:

$P = R \times S = \{ <u,v> \mid u \in R(U) \wedge v \in S(V) \}$

Ví du:

Cho 2 quan hệ R(U) và S(V) như sau:

A	B
α	1
β	2

R

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

S

⟹

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

R x S

II.2.2.4 Khái niệm thông thương giữa các quan hệ:

Trong một số trường hợp, để trả lời được các câu hỏi, ta không thể làm việc trên từng quan hệ riêng rẻ. Chẳng hạn, ta có CSDL như sau:

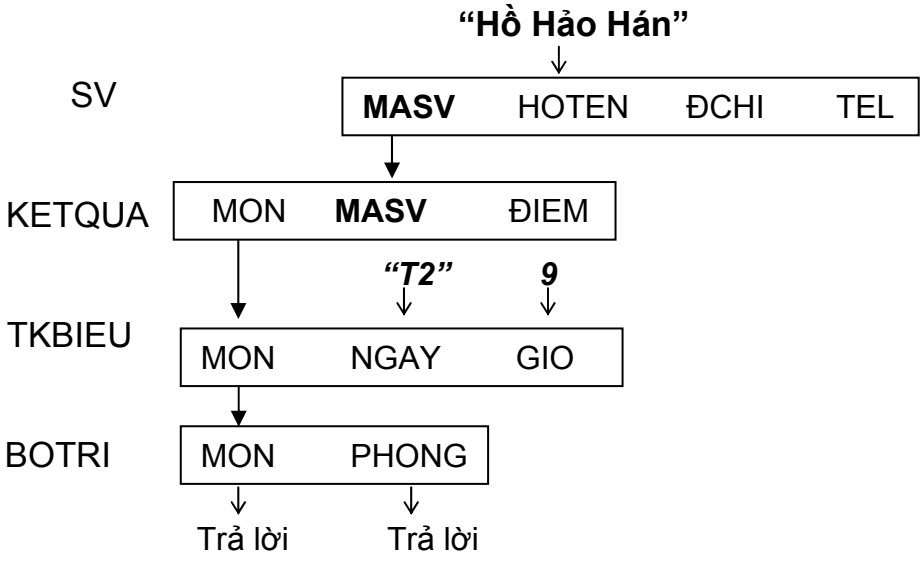
SV(MASV, HOTEN, DCHI, TEL)

KETQUA(MON, MASV, DIEM)

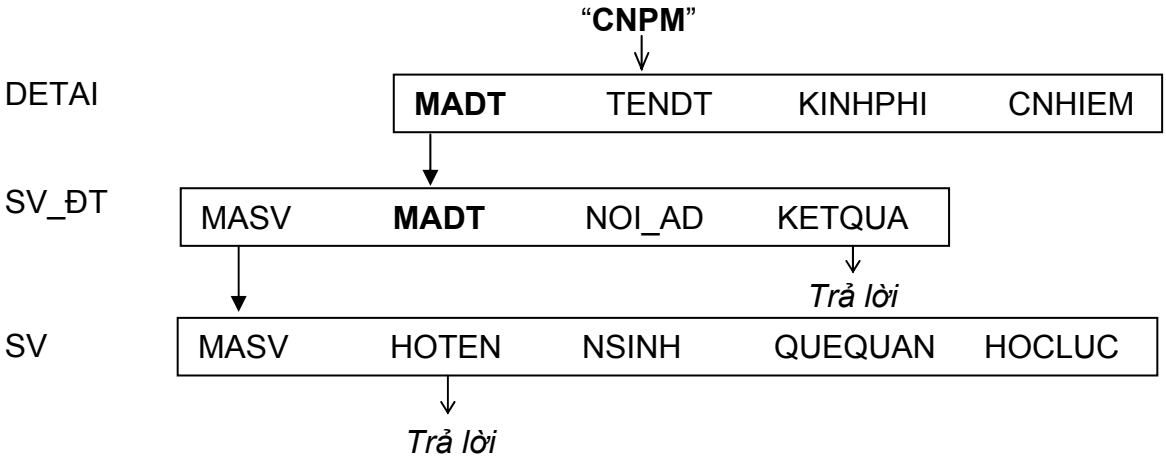
TKBIEU(MON, NGAY, GIO)

BOTRI(MON, PHONG)

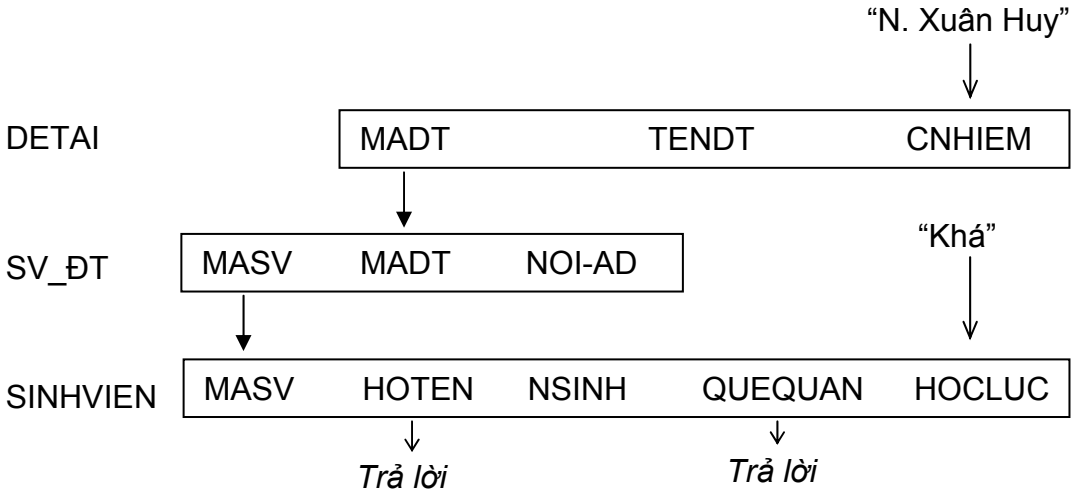
Để trả lời được câu hỏi: "Cho biết SV Hồ Hảo Hán học MÔN gì, *ở PHÒNG nào vào 9h mỗi sáng thứ hai",* thì ta phải nối các quan hệ lại như hình minh họa sau:



Hay với câu hỏi: *Cho biết TÊN SV và KẾT QUẢ đạt được của đề tài "CNPM*".



Hay "Cho biết TÊN và QUÊ QUÁN của SV làm đề tài của thầy N. Xuân Huy và học khá".



II.2.2.5 Phép θ - kết nối: kí hiệu $\triangleright\triangleleft_\theta$

Định nghĩa:

Cho hai quan hệ R(U) và S(V). Gọi θ là một phép toán logic liên quan tới các thuộc tính của U và V (thường thuộc trong các phép so sánh $<, \leq, >, \geq, =, \neq$). Phép θ - kết nối giữa quan hệ R với quan hệ S, là một quan hệ P với tập thuộc tính $U \cup V$, được định nghĩa như sau:

$$P = R \triangleright\triangleleft_\theta S = \{ <u,v> \mid u \in R \wedge v \in S \wedge u[U]\ \theta\ v[V] \}$$

Chú ý:

Phép kết nối này chỉ thực hiện được khi θ thực hiện được giữa các thuộc tính của U và V. Nếu không dựa trên θ và $U \cap V = \varnothing$, thì $R \triangleright\triangleleft S$ trở thành tích Đề-các của R và S. Nếu θ là phép so sánh bằng "=", thì gọi là kết nối bằng. Nếu kết nối bằng được thực hiện tại tất cả các thuộc tính cùng tên của hai quan hệ R và S thì đây là phép kết nối tự nhiên.

Ví du:

A	B
α	12
β	20

R

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

S

$\Longrightarrow$

A	B	C	D	E
α	12	β	20	b
β	20	β	20	b

$R \triangleright\triangleleft_{B <= D} S$

II.2.2.6 Phép kết nối tự nhiên (Natural Join): kí hiệu *

Định nghĩa:

Cho hai quan hệ R(U) và S(V), đặt $X = U \cap V$. Kết nối tự nhiên giữa R và S là quan hệ P với tập thuộc tính $U \cup V$, được định nghĩa như sau:

$$P = R * S = \{ <u,v\backslash X> \mid u \in R \wedge v \in S \wedge u.X = v.X \}$$

Ví dụ 1:

Cho biết tên sinh viên đã thực hiện đề tài 1 và nơi áp dụng tương ứng.

P = (SV_DT * SINHVIEN) (MADT =1) [HOTEN, NOI_AD]

Ví dụ 2:

Cho biết tên các đề tài được áp dụng ở Hải Phòng.

P = (SV_DT * DETAI) (NOIAD = 'Hải Phòng') [TENDT]

II.2.2.7 Phép chia (Division): kí hiệu /

Đinh nghĩa:

Cho hai quan hệ R(U) và S(V) với $U \supset V$, và $S \neq \emptyset$. Đặt X = U - V. Thương của phép chia quan hệ R cho quan hệ S là quan hệ P với tập thuộc tính X, được định nghĩa như sau:

$$P = R / S = R \div S = \{ u.X \mid u \in R \ \wedge \ (\forall v \in S), (<u.X,v> \in R)\}$$

Ví du 1:

Cho hai quan hệ R và S như sau:

A	B
α	1
α	2
α	3
β	1
γ	1
δ	1
δ	3
δ	4
∈	6
∈	1
β	2

R

B
1
2

S

⟹

A
α
β

R ÷ S

Ví dụ 2:

Tìm các mã đề tài có nơi áp dụng ít ra là như đề tài 2

SV_DT [MADT, NOI_AD] ÷ SV_DT (MADT = 2) [NOI_AD]

II.2.2.8 Các phép toán trên tập hợp:

Hai quan hệ R và S được gọi là tương thích nếu chúng có cùng tập thuộc tính.

Với các quan hệ tương thích ta có thể định nghĩa các phép toán tập hợp như hợp, giao, trừ.

II.2.2.8.1 Phép hợp (Union): kí hiệu ∪

Đinh nghĩa:

Hợp của hai quan hệ tương thích R(U) và S(V) là quan hệ P với tập thuộc tính U và các bộ thuộc một trong hai quan hệ cho trước

$$P = R \cup S = R + S = \{t \mid t \in R \vee t \in S\}$$

Ví du:

Liệt kê danh sách các tỉnh gồm: nơi áp dụng đề tài và quê quán của sinh viên

⇨ SINHVIEN [QUEQUAN] ∪ DETAI [NOI_AD]

II.2.2.8.2 Phép giao (Intersection): kí hiệu ∩

Đinh nghĩa:

Giao của hai quan hệ tương thích R(U) và S(V) là quan hệ P với tập thuộc tính U và các bộ đồng thời có mặt trong hai quan hệ cho trước

$P = P \cap S = P.\ S = \{ t \mid t \in R \wedge t \in S\}$

Ví du:

Cho biết tên của chủ nhiệm vừa có hướng dẫn đề tài áp dụng tại Cần Thơ vừa hướng dẫn đề tài áp dụng tại Bạc Liêu.

⇨ T_1 = (DETAI * SV_DT) (NOI_AD = 'Can tho') [CNHIEM]

⇨ T_2 = (DETAI * SV_DT) (NOI_AD = 'Bac lieu) [CNHIEM]

⇨ Kết quả: t= $T_1 \cap T_2$

II.2.2.8.3 Phép trừ (Subtraction): kí hiệu \

Đinh nghĩa:

Hiệu của hai quan hệ tương thích R(U) và S(V) là quan hệ P với tập thuộc tính U và các bộ đồng thời có trong R nhưng không có trong S.

$P = P \setminus S = P - S = \{ t \mid t \in R \wedge t \notin S\}$

Ví dụ 1:

Tìm MASV của các sinh viên có học lực giỏi nhưng kết quả đề tài nhỏ hơn 7 điểm.

⇨ SINHVIEN (HOCLUC = 'Gioi') [MASV] \ SV_DT (KETQUA>7)[MASV]

Ví dụ 2:

Tìm MASV của các sinh viên không có thực hiện đề tài ở TP HCM.

⇨ SINHVIEN[MASV] \ SV_DT (NOIAD='TP HCM')[MASV]

II.2.2.9 Đổi tên thuộc tính:

Trong một số trường hợp, ta cần thay đổi tên thuộc tính của quan hệ, chẳng hạn: Ta có quan hệ R(A, C) và S(B), trong đó dom(B) = dom(C). Theo định nghĩa của phép chia thì biểu thức:

$P = R(A,C) \div S(B)$ là vô nghĩa vì $B \notin \{A, C\}$

Vì vậy: muốn thực hiện được phép toán này ta phải đổi tên thuộc tính như sau: P = R(A, C/B) ÷ S(B) hoặc P = R(A,C) ÷ S(B/C)

II.2.3 Tính chất của các phép toán ĐSQH:

II.2.3.1 Tính giao hoán:

Đinh nghĩa:

Cho một đại số hệ α = <M, F>,với M là tập nền, F là tập các phép toán trên M. Một phép toán hai ngôi f(a,b) có tính chất giao hoán nếu:

$\forall a,b \in M$ đều có: $f(a,b) = f(b,a)$

Trong ĐSQH, các phép toán hợp, giao, kết nối có tính chất giao hóan, tức là:

$R * S = S * R$ (nếu thứ tự các thuộc tính là không quan trọng)

$R + S = S + R$ (R và S tương thích)

$R . S = S . R$

II.2.3.2 Tính kết hợp:

Đinh nghĩa:

Phép toán hai ngôi f trong đại số hệ $\alpha = <M, F>$ có tính chất kết hợp nếu:

$\forall a,b,c \in M$ đều có: $f(f(a,b), c) = f(a, f(b,c))$

Trong ĐSQH, các phép toán hợp, giao, kết nối có tính chất kết hợp, tức là:

$(R * S) * P = R * (P * S)$

$(R + S) + P = R + (P + S)$

$(R . S) . P = R . (P . S)$

II.2.3.3 Tính lũy đẳng:

Đinh nghĩa:

Phép toán một ngôi f trong đại số hệ $\alpha = <M, F>$ có tính chất lũy đẳng bậc m nếu m là số tự nhiên nhỏ nhất để

$\forall a \in M \quad f^m(a) = f(f(...f(a))...) = f(a)$

Ví dụ:

Trong ĐSQH, các phép chọn, chiếu là các phép tóan có tính lũy đẳng bậc 2. Tức là: với E là biểu thức điều kiện phát biểu trên tập thuộc tính U.

$R(E)(E) = R(E)$

Tương tự: $X \subseteq U$, ta có: $R[X][X] = R[X]$

Đinh nghĩa:

Phép toán hai ngôi f trong đại số hệ $\alpha = <M, F>$ có tính chất lũy đẳng bậc m nếu m là số tự nhiên nhỏ nhất để

$\forall a \in M \quad f(...f(f(a,a), a), ...,a) = a$

Trong ĐSQH, các phép toán +, kết nối, . (giao) là các phép toán 2 ngôi có tính lũy đẳng bậc nhất. Tức là:

$R + R = R$

$R . R = R$

$R * R = R$

II.2.3.4 Thác các phép chọn:

Cho quan hệ R(U) và E_1, E_2 là hai biểu thức chọn phát biểu trên U, khi đó:

a) Nếu $E_1 => E_2$ thì $R(E_1)(E_2) = R(E_1)$

b) $R(E_1 \wedge E_2) = R(E_2 \wedge E_1)$

Từ tính chất b suy ra rằng:

nếu $E_1 \Rightarrow E_2$ thì $R(E_1)(E_2) = R(E_2)(E_1) = R(E_1)$

II.2.3.5 Phép chọn theo hội, tuyển:

Cho quan hệ R(U) và các biểu thức điều kiện E_1, E_2 trên U, khi đó:

a) $R(E_1 \wedge E_2) = R(E_1) . R(E_2)$
b) $R(E_1 \vee E_2) = R(E_1) + R(E_2)$
c) Nếu $E_1 \Rightarrow E_2$ thì $R(E_1) \subseteq R(E_2)$ do đó: $R(E_1 \vee E_2) = R(E2)$
d) $R(T) = R$ với T là công thức hằng đúng
e) $R(F) = \varnothing$ với F là công thức hằng sai

II.2.3.6 Thác các phép chiếu:

Cho quan hệ R(U) và các tập thuộc tính X và Y thỏa điều kiện $X \subseteq Y \subseteq U$.

Khi đó: $R[Y][X] = R[X]$

R[X][Y] chỉ được xét khi X=Y và dĩ nhiên là cho kết quả R[X]

II.2.3.7 Thác các phép chọn - kết nối:

Cho hai quan hệ R(U) và hai biểu thức chọn E_R trên U, E_S trên V. Khi đó:

$(R*S)(E_R \wedge E_S) = R(E_R) * S(E_S)$

=> Việc thực hiện biểu thức theo **vế phải** là nhanh hơn, ít tốn không gian lưu trữ.

II.2.3.8 Thác phép chiếu - chọn:

Cho quan hệ R(U). Gọi E là biểu thức điều kiện trên U, E_X trên tập X ($X \subseteq U$). Khi đó ta có:

$R(E \wedge E_X)[X] = R(E)(E_X)[X] = R(E)[X](E_X)$

II.2.3.9 Biểu diễn phép giao qua phép trừ:

$R . S = R - (R - S)$

II.2.3.10 Biểu diễn phép chia qua các phép toán khác:

Cho hai quan hệ R(U) và S(V) với $V \subseteq U$. đặt X = U - V. Khi đó:

$R \div S = R[X] - (R[X] * S - R)[X]$

II.2.4 Biểu thức quan hệ và tối ưu hóa biểu thức

1) Biểu thức quan hệ (BTQH) được xây dựng từ các toán hạng là các quan hệ trong một CSDL cho trước và các phép toán quan hệ.

Thông thường, trong một BTQH người ta coi các phép toán một ngôi có độ ưu tiên cao hơn các phép toán hai ngôi. Ngoài ra, mỗi HQTCSDL cụ thể lại có những qui định riêng về thứ tự ưu tiên cho các phép toán khác. Chẳng hạn qui định phép giao có độ ưu tiên cao hơn phép hợp hai quan hệ.

Ta đã biết, mỗi câu hỏi truy cập CSDL thường được phát biểu dưới dạng một BTQH T. HQTCSDL, trong trường hợp có cài đặt các phép toán quan hệ thường được trang bị các chức năng sau đây (theo trình tự xử lý)

(1) Kiểm tra cú pháp của biểu thức T. Nếu T không có lỗi thì làm tiếp bước 2.

(2) Tổ chức tối ưu hóa cây thực hiện T. Tức là xác định trật tự thực hiện các phép toán trong T nhằm đạt 2 yêu cầu:

a. Cho cùng một kết quả như khi thực hiện T

b. Tốn ít vùng nhớ và thời gian thực hiện hơn T

(3) Thực hiện biểu thức đã tối ưu hóa.

2) Ta biết rằng các quan hệ càng nhỏ thì chi phí thời gian thực hiện và vùng nhớ càng thấp. Quan hệ có thể nhỏ theo hai kích thước: hẹp ngang (ít thuộc tính) và ngắn (ít bộ). Trong số các phép toán quan hệ thì phép chiếu và phép chọn có khả năng giảm kích thước ngang và dọc của các quan hệ. Phép chia cũng thu hẹp kích thước của quan hệ nhưng do nó ít được sử dụng cho nên ta không xét đến từng quá trình tối ưu hóa. Phép kết nối thường làm tăng kích thước các quan hệ.

Từ nhận xét trên, ta có thể tối ưu hóa biểu thức quan hệ như sau:

Tìm cách chuyển đổi vị trí các phép toán để thực hiện các phép toán để thực hiện các phép chiếu và chọn sớm nhất đến mức có thể.

Ví dụ:

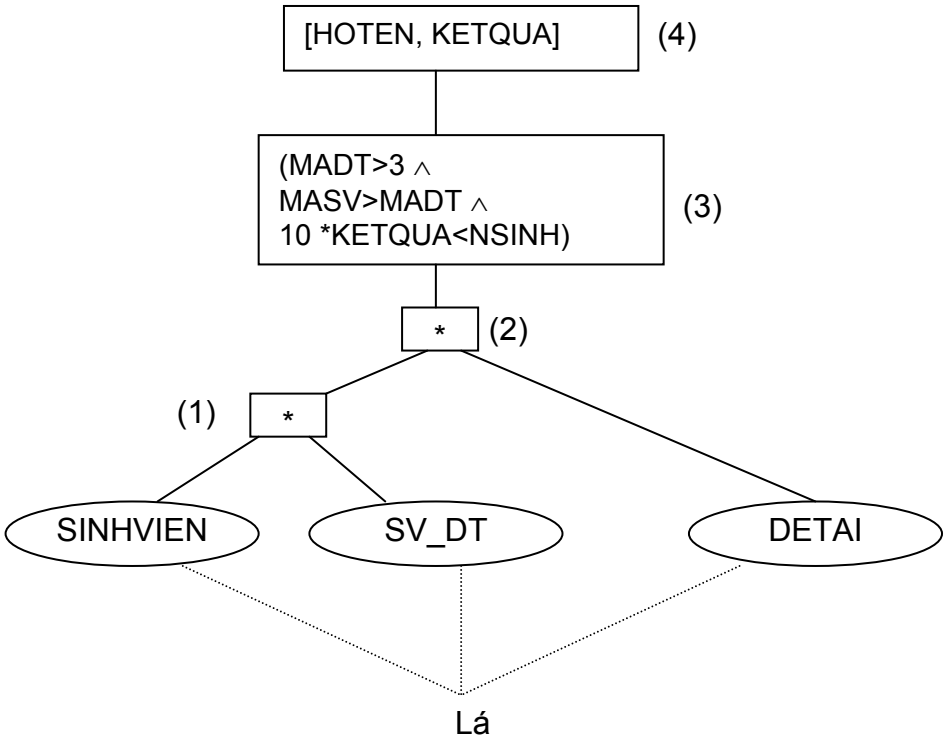
Cho biết tên đề tài và kết quả thực tập của những sinh viên tham gia đề tài có mã số trên 3 (MADT>3) đạt kết quả thấp hơn 1/10 giá trị của năm sinh và có mã (MASV) lớn hơn hoặc bằng mã đề tài

Giả sử BTQH cho câu hỏi trên là:

SINHVIEN*SV_DT*DETAI(MADT>3 $\wedge$

MASV>MADT $\wedge$ 10*KETQUA<NSINH)[HOTEN,KETQUA]

Quá trình thực hiện cây biểu thức này được thực hiện dưới dạng cây, gọi là cây thực hiện như sau:



Cây biểu thức được thực hiện từ lá đến gốc. Trước hết là phép (1) kết nối hai quan hệ SINHVIEN và SV_DT để được một quan hệ trung gian T, sau đó đến phép (2) kết nối T với quan hệ DETAI cho kết quả ghi vào T. Rồi đến phép chọn (3) và sau cùng là phép chiếu (4). Ta có thể viết trật tự thực hiện đó như sau:

(1) T = SINHVIEN * SV_DT

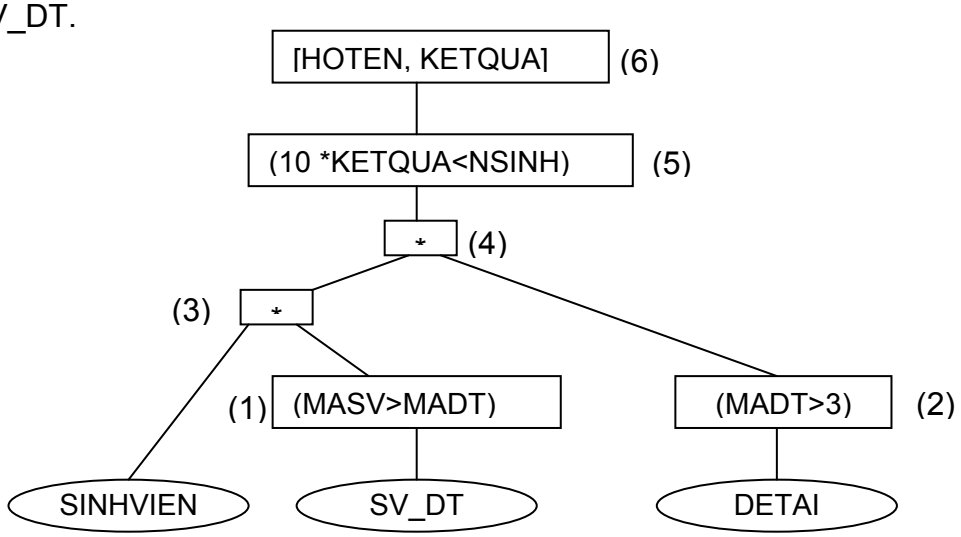
(2) T = T *DETAI

(3) T= T(MADT>3 $\wedge$ MS SV>MADT $\wedge$ 10 *KETQUA<NSINH)

(4) T=T[HOTEN, KETQUA]

Để ý rằng MADT là một thuộc tính của quan hệ DETAI cho nên ta có thể đưa phép chọn theo điều kiện MADT>3 và quan hệ DETAI. Tức là ta đẩy một phần phép chọn về phía lá. Tương tự. phép chọn MASV>MADT có thể đẩy xuống để thực hiện sớm trong quan hệ SV_DT.

Ta có:



Trình tự thực hiện được ghi rõ trong các nhãn ở mỗi nút:

(1) T_1 =SV_DT(MASV>MADT)

(2) T_2 = DETAI(MADT>3)

(3) T_1 = SINHVIEN * T_1

(4) T_1 = T_1 * T_2

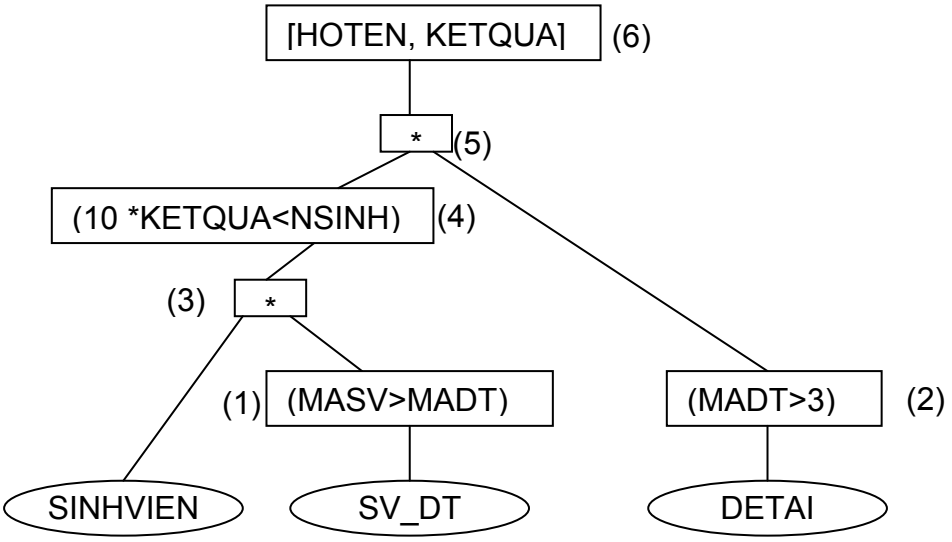
(5) T_1 = T_1(10 *KETQUA<NSINH)

(6) T_1 = T_1[HOTEN, KETQUA]

Biểu thức quan hệ của cây trên sẽ là:

(SINHVIEN*SV_DT(MASV>MADT)*DETAI(MADT>3))(10*KETQUA>NSINH)[HOTEN,KETQUA]

Sau phép nối kết (3) quan hệ trung gian sẽ chứa các thuộc tính KETQUA và NSINH. Do đó ta đẩy phép chọn (5) xuống dưới phép kết nối (4). Ta có:



Biểu thức tương ứng sẽ là:

((SINHVIEN*SV_DT(MASV>MADT))(10*KETQUA>NSINH)
* DETAI(MADT>3)) [HOTEN,KETQUA]

3) Phép chọn chiếu

Người ta nhận thấy rằng hai phép toán chọn và chiếu thường đi kèm với nhau. Đặc biệt trong quá trình tối ưu hóa biểu thức quan hệ, do đó chu yếu đẩy phép chọn và chiếu về phía lá, cho nên kết quả tất yếu là các phép toán này sẽ dồn sát lại nhau.

Mặt khác như sẽ nói bên dưới, nếu có thể dược, người ta sẽ tìm cách thu gọn bề ngang các quan hệ bằng phép chiếu.

Ta định nghĩa them một phép toán nữa cho ĐSQH gọi là phép chọn-chiếu như sau:

Định nghĩa:

Cho quan hệ R với tập thuộc tính U, E là biểu thức chọn trên U, $X \subseteq U$, phép chọn chiếu theo biểu thức E và trên tập thuộc tính X, Ký hiệu là R(E, [X]) cho ta quan hệ P với tập thuộc tính X và các bộ được xác định như sau:

$$P=\{t[X] \mid t \in R \text{ và } t \text{ thỏa } E\}$$

Trở lại ví dụ trên, ta thấy:

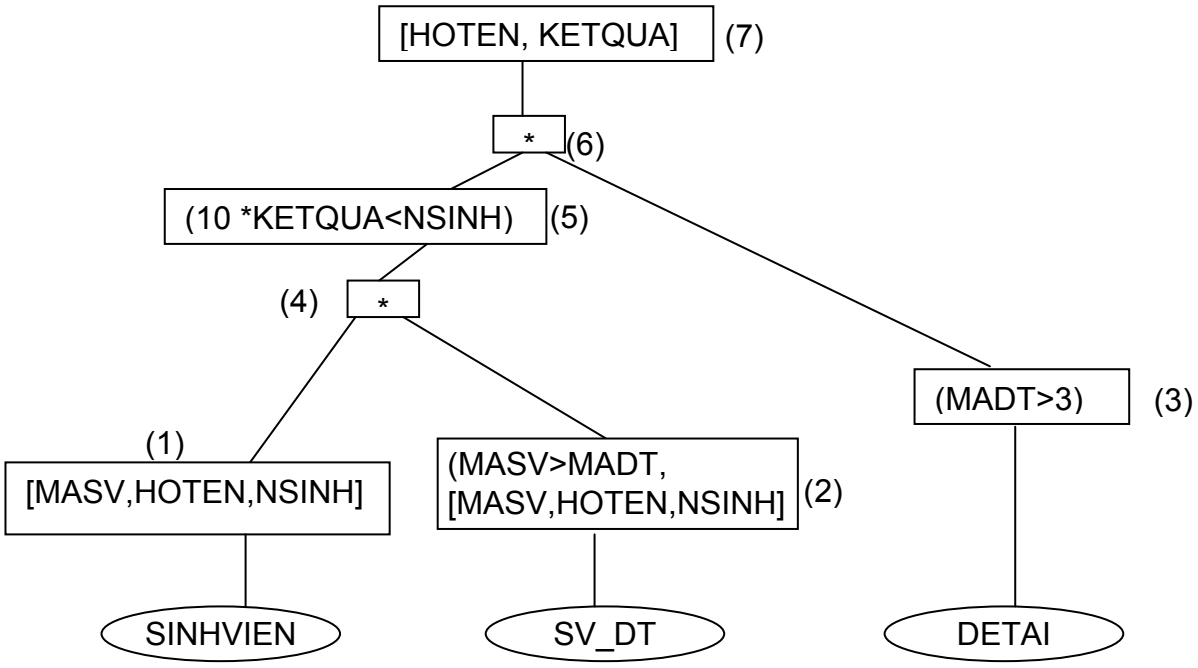
- Quan hệ DETAI trong biểu thức đã cho chỉ cần giữ lại thuộc tính MADT để phục vụ cho kết nối là đủ, do đó tat hay phép toán (2) bằng phép chon-chiếu DETAI(MADT>3,[MADT])
- Quan hệ SV_DT chỉ cần giữ lại các thuộc tính MASV, MADT và KETQUA, do đó ta có thể thay phép toán (1) bằng phép chọn chiếu

  SV_DT(MASV>MADT,[MASV,MADT,KETQUA])
- Quan hệ SINHVIEN cần giữ lại các thuộc tính MASV, HOTEN, do đó ta có thể thực hiện phép chiếu trên các thuộc tính đó để thu hẹp bề ngang của quan hệ

SINHVIEN[MASV,HOTEN]

- Sau khi thực hiện các phép kết nối (3), ta có thể tiến hành phép chọn (4) đồng thời với phép chiếu trên các thuộc tính MADT, HOTEN và KETQUA

Như vậy, ví dụ đã cho có thể tiếp tục được tối ưu hóa như sau:



CHƯƠNG III - NGÔN NGỮ SQL

ĐSQH và phép tính quan hệ đều hết sức cơ bản để biểu diễn các câu hỏi, nhưng vẫn còn mang nặng tính chất hình thức. Do đó, các hệ quản trị cơ sở dữ liệu cần một ngôn ngữ hỏi thân thiện với người sử dụng hơn. SQL đáp ứng yêu cầu đó. Ngôn ngữ SQL được phát triển bởi IBM vào thập niên 70 và nhanh chóng trở thành chuẩn rộng rãi khắp thế giới cho việc xử lý và điều khiển trong CSDL quan hệ.

Trong chương này ta xét SQL cụ thể trong FoxPro 2.6 vì dễ cài đặt trên máy nhỏ và đơn giản sẽ giúp dễ dàng nắm được các khái niệm chính.

III.1. CÁC KHÁI NIỆM CƠ BẢN

- ***Truy vấn***: tham chiếu dữ liệu trong CSDL
- ***Xử lý dữ liệu***: thêm, sửa, xóa dữ liệu trong CSDL
- ***Định nghĩa dữ liệu***: thêm bảng, view, và chỉ mục và CSDL
- ***Điều khiển dữ liệu***: hạn chế quyền xử lý dữ liệu cho các dữ liệu riêng trong CSDL.

Với SQL, dữ liệu được lưu trữ dưới dạng bảng. CSDL quan hệ là một tập các bảng hay các quan hệ. Trong mỗi quan hệ, dữ liệu được bố trí theo hai chiều: cột (trường) theo chiều dọc và dòng (mẫu tin) theo chiều ngang. Một dòng là một mục dữ liệu trong quan hệ, mỗi dòng được đặc trưng bằng một số các thuộc tính gọi là cột.

Trong chương này, chúng ta sẽ dùng CSDL bao gồm các quan hệ sau:

1. **PHONG**(MA_PHONG, TEN_PHONG, VI_TRI, TRUONG_PHONG)
2. **NHANVIEN**(MA_NV, TEN_NV, CONG_VIEC, LUONG, PHU_CAP, MA_PHONG, PHAI)
3. **HO_SO_NV**(MA_NV, THU_TU, NGAY_BĐ, NGAY_KT, LUONG, VIEC_CU, VIEC_MOI, THANG_CAP, PHONG_CU, PHONG_MOI)
4. **CONG_VIEC**(MA_CV, TEN_CV)

III.2. ĐỊNH NGHĨA BẢNG

III.2.1. Tạo bảng

Bước đầu tiên để tạo một CSDL là tạo bảng. Lệnh CREATE TABLE dùng tạo bảng với cú pháp như sau:

CREATE TABLE <tên bảng>
(<tên cột 1> <loại dữ liệu> [NOT NULL],
<tên cột 2> <loại dữ liệu> [NOT NULL],
...
<tên cột n> <loại dữ liệu> [NOT NULL])

Từ khóa **NOT NULL** có nghĩa là bắt buộc phải có giá trị trong cột này. Đây là một trong các ràng buộc toàn vẹn quan trọng cho phép người thiết kế: yêu cầu phải

nhập liệu hay không vào các cột chỉ định. Ít nhất một cột phải được định nghĩa. Dấu phẩy dùng để ngăn cách giữa các định nghĩa cột.

Ví du:

```
CREATE TABLE NHAN_VIEN(
        MA_NV       char(3)             not null,
        TEN_NV      char (8),
        CONG_VIEC   char(4)             not null,
        LUONG       decimal (8,2) ,
        PHU_CAP     decimal(8,2),
        MA_PHONG    smallint            not null,
        PHAI        char(1))
```

III.2.2. Thêm dòng vào bảng

Bảng nhân viên đã được tạo nhưng chưa có dữ liệu, chúng ta có thể thêm dữ liệu vào bảng bằng lệnh INSERT, có cú pháp như sau:

INSERT INTO <tên bảng>
(<tên cột 1>, <tên cột 2>,...)
VALUES (<giá trị 1>,<giá trị 2,...>)

Ví du:

Thêm một dòng dữ liệu mới vào bảng NHAN_VIEN

INSERT INTO NHAN_VIEN (MA_NV,TEN_NV,CONG_VIEC, MA_PHONG)

VALUES (“123”,”Nguyễn Văn An”,”TKY”,3)

III.3. LỆNH TRUY VẤN SELECT

Truy xuất dữ liệu CSDL là một trong các thao tác cơ bản SQL. Lệnh truy xuất xuất dữ liệu được gọi là lệnh truy vấn SELECT. Lệnh truy vấn tổng quát:

```
SELECT   [ALL/ DISTINCT] * | <các cột muốn kết xuất>
FROM  <các bảng tham gia vào câu hỏi>
        [ INTO <nơi kết xuất> | TO FILE <tên file> [ADDITIVE] |
       [TO SCREEN ]    [NOWAIT]
[ WHERE <đk kết nối> [AND <đk kết nối> ...]
        [AND <đk chọn> [AND | OR <đk chọn>...] ] ]
[GROUP BY <các cột để phân nhóm>]
[HAVING <các đk chọn trên nhóm>]
[ODER BY <cột để sắp xếp>[ASC|DESC] [,<cột để sắp xếp>...]]
[UNION <câu select khác>]
```

Để thi hành một truy vấn đơn giản chúng ta có thể dùng lệnh SELECT gồm hai thành phần được gọi là mệnh đề như sau:

SELECT **tên cột, danh sách các tên cột, hay ***

FROM **tên bảng, hay danh sách các tên bảng**

Mệnh đề SELECT luôn luôn phải có và sau đó phải kèm theo mệnh đề FROM

III.3.1. Hiển thị toàn bộ bảng

Một cách sử dụng SQL đơn giản là hiển thị nội dung của một bảng như lệnh sau:

SELECT * ;

FROM *tên bảng*

Dấu hoa thị (*) thay thế tất cả các cột trong bảng.

Ví dụ:

Nếu chúng ta muốn hiển thị tất cả nội dung của bảng NHAN_VIEN, cú pháp lệnh đúng như sau:

SELECT * ;

FROM *NHAN_VIEN*

III.3.2. Lưu kết quả câu hỏi

Ta sử dụng từ khóa ***INTO <nơi kết xuất>*** để lưu kết quả câu hỏi.Nếu không có INTO, kết quả có thể in ra màn hình, máy in, tập tin ... Nơi kết xuất có thể là:

- TABLE <tên của table>
- DBF <tên của dbf>
- ARRAY <tên mảng>
- CURSOR <tên cursor>

Ví dụ:

SELECT * INTO DBF tam ;
FROM *NHAN_VIEN*

III.3.3. Sắp xếp kết quả

SQL cho phép sắp xếp thứ tự kết quả của lệnh SELECT bằng cách bổ sung tiêu chuẩn thứ tự sắp xếp (ORDER) như ví dụ theo sau

Ví dụ:

Giả sử chúng ta muốn xem tất cả thông tin của bảng NHAN_VIEN theo thứ tự mã phòng tăng dần. Lệnh SQL có thể như sau:

SELECT * ;

FROM *NHAN_VIEN ;*

ORDER BY *MA_PHONG*

Chú ý là mệnh đề ORDER BY được thêm vào câu lệnh SELECT. SQL còn cho phép kết quả được sắp thứ tự theo nhiều cột bằng cách thêm các tiêu chuẩn bổ sung cho mệnh đề ORDER BY.

Ví dụ:

Hiển thị tất cả thông tin trong bảng NHAN_VIEN sắp xếp theo mã phòng và sau đó theo tên. Chú ý là trong lệnh truy vấn này, SQL trước tiên được sắp xếp thứ tự tất cả nhân viên theo phòng, sau đó sẽ sắp xếp lần nữa theo thứ tự họ tên trong từng phòng.

SELECT * ;

FROM ***NHAN_VIEN*** **;**
ORDER BY ***MA_PHONG, TEN_NV***

Ngoài ra, mệnh đề ORDER BY có thể dùng để hiển thị thông tin sắp theo thứ tự tăng dần(ASC) hoặc giảm dần(DESC). Khi người sử dụng không nêu rõ thứ tự tăng dần hay giảm dần thì SQL ngầm định là thứ tự tăng dần.

Ví dụ:

Hiển thị thông tin của bảng NHAN_VIEN theo thứ tự mã phòng tăng dần và mức lương giảm dần:

SELECT * **;**
FROM ***NHAN_VIEN*** **;**
ORDER ***MA_PHONG ASC, LUONG*** **DESC**

III.3.4. Sắp xếp thứ tự các cột khi hiển thị

Trong các ví dụ trước, chúng ta dùng dấu hoa thị (*) để liệt kê nội dung của bảng. Bây giờ chúng ta sẽ liệt kê cũng bảng đó nhưng thứ tự các cột sẽ thay đổi.

Ví dụ:

Hiển thị tên, công việc, lương, phụ cấp, mã nhân viên,mã phòng và giới tính của tất cả nhân viên

SELECT TEN_NV, CONG_VIEC, LUONG, PHU_CAP, MA_NV, MA_PHONG, PHAI;

FROM ***NHAN_VIEN***

III.3.5. Giới hạn một số cột khi hiển thị

Nếu không cần thiết hiển thị tất cả các cột trong bảng. Các cột có thể không hiển thị bằng cách không khai báo chúng trong mệnh đề SELECT.

Ví dụ:

Hiển thị tên, công việc và lương của tất cả nhân viên. Sắp xếp kết quả theo thứ tự tên giảm dần .

SELECT TEN_NV, CONG_VIEC, LUONG ;
FROM NHAN_VIEN ;
ORDER BY TEN_NV DESC

III.3.6. Loại bỏ những dòng trùng lắp

Trong trường hợp kết quả của lệnh truy vấn có các dòng trùng lặp, SQL cho phép chúng ta chỉ hiển thị một dòng duy nhất bằng khai báo thông số DISTINCT trong mệnh đề SELECT.

Ví dụ:

Hiển thị tất cả các địa điểm của các phòng trong bảng PHONG và loại bỏ các dòng lặp lại:

SELECT DISTINCT ***VI_TRI*** **;**

FROM ***PHONG***

III.3.7. Sử dụng bí danh cho cột

Ta có thể sử dụng bí danh **AS <tên cột>** cho các ***mục*** trong mệnh đề SELECT, điều này đặc biệt có ích khi các *mục* là một biểu thức hoặc chứa một hàm trên vùng và bạn muốn cho nó một cái tên có nghĩa.

Ví dụ: Có thể thay đặt bí danh cho TEN_PHONG là PHONG như sau:

```
SELECT  TEN_PHONG AS PHONG     ;
FROM      PHONG ;
ORDER BY PHONG
```

III.4. CHỌN CÁC DÒNG TRONG BẢNG

Trong phần trước, chúng ta đã thử dùng lệnh SELECT để tham khảo tất cả hay một số các cột trong bảng. Trong phần này, chúng ta tiếp tục xét xem có thể tham khảo một số dòng nào đó hay không với lệnh SELECT.

Để thực hiện việc này, chúng ta cần sử dụng thành phần WHERE trong lệnh SELECT, thành phần này báo cho CSDL biết cách tìm kiếm thông tin trong bảng và chỉ hiển thị các dòng thỏa điều kiện chọn lựa.

Ví dụ:

Hiển thị các nhân viên đang làm việc trong phòng 40

```
SELECT    *  ;
FROM       NHAN_VIEN    ;
WHERE     MA_PHONG=40
```

Khi chúng ta sử dụng thành phần WHERE, kết quả có thể không có, có một hay nhiều dòng. Điều kiện trong thành phần WHERE được so sánh với nội dung một số cột trong bảng. Chỉ có những dòng thoả tiêu chuẩn được chọn lựa hiển thị.

Vi dụ:

Hiện tất cả thông tin về nhân viên Lê Quỳnh Như

```
SELECT  *    ;
FROM     NHAN_VIEN     ;
WHERE   TEN_NV = “Lê Quỳnh Như”
```

III.4.1. Điều kiện kết hợp

Trong một số trường hợp, chúng ta cần xác định nhiều tiêu chuẩn chọn lựa trong thành phần WHERE. Điều này có thể thực hiện dễ dàng bằng cách dùng từ khóa AND để kết hợp các tiêu chuẩn.

Ví dụ:

Hiển thị các trưởng phòng có mức lương lớn hơn 4000

```
SELECT  *    ;
FROM     NHAN_VIEN     ;
```

```
WHERE   CONG_VIEC = “TPG”    AND    LUONG>4000
```

III.4.2. Điều kiện loại trừ

Trong ví dụ trước, chúng ta đã dùng điều kiện AND để chọn lọc kết quả chỉ hiển thị các dòng thỏa các tiêu chuẩn chọn lựa. Chúng ta cũng có thể chọn lọc thông tin thỏa một trong các tiêu chuẩn chọn lựa bằng cách dùng từ khoá OR

Ví dụ:

Chọn hiển thị các nhân viên làm việc trong phòng 40 hoặc nhân viên có mức lương lớn hơn 4000

```
SELECT     *         ;
FROM       NHAN_VIEN ;
WHERE      MA_PHONG=40  OR  LUONG > 4000
```

III.4.3. Điều kiện phủ định

Chúng ta cũng có thể lựa chọn các dòng không thỏa một tiêu chuẩn chọn lựa nào đó bằng cách dùng toán tử phủ định NOT

Ví dụ:

Hiển thị tất cả các cán bộ trưởng phòng không làm việc trong phòng 30. Nội dung hiển thị bao gồm các cột tên, công việc và phòng.

```
SELECT   TEN_NV, CONG_VIEC, MA_PHONG       ;
FROM     NHAN_VIEN   ;
WHERE    CONG_VIEC= ”TPG” AND    MA_PHONG!=30
```

Chúng ta có thể dùng AND, OR và NOT kết hợp trong cùng một lệnh truy vấn để diễn tả tiêu chuẩn chọn lựa.Ví dụ trên có thể viết lại dùng từ khoá NOT như sau:

```
SELECT   TEN_NV, CONG_VIEC, MA_PHONG       ;
FROM     NHAN_VIEN   ;
WHERE    CONG_VIEC=”TPG” AND NOT   MA_PHONG=30
```

III.4.4. So sánh với một tập dữ liệu

SQL cho phép chúng ta so sánh giá trị cột với một tập các giá trị của một tập dữ liệu (nghĩa là hệ quản trị CSDL sẽ chọn hiển thị các dòng có chứa giá trị nằm trong tập giá trị cho trước). SQL cho phép dùng toán tử **IN** (**NOT IN**) để tìm kiếm giá trị trong một tâp hợp các giá trị.

Ví dụ:

Hiển thị tất cả thông tin về nhân viên đang làm việc trong các phòng 10, 30 và 50.

```
SELECT    * ;
FROM      NHAN_VIEN  ;
WHERE     MA_PHONG   IN    (10, 30, 50)
```

Ví dụ:

Hiển thị tất cả thông tin về nhân viên không làm việc trong các phòng 10, 30 và 50.

```
SELECT    * ;
FROM      NHAN_VIEN  ;
WHERE     MA_PHONG  NOT IN   (10, 30, 50)
```

III.4.5. Tìm kiếm theo phạm vi

SQL cho phép người sử dụng tìm kiếm dễ dàng một giá trị có thuộc trong một vùng xác định nào đó hay không. Toán tử BETWEENcho phép chúng ta chọn lựa hiển thị các dòng có chứa giá trị trong vùng xác định đó.

Cú pháp toán tử BETWEEN là:

```
SELECT   <danh sách tên cột>
FROM     <tên bảng>
WHERE    <tên cột>
[NOT] BETWEEN<giá trị 1>AND<giá trị 2>
```

Ví dụ:

Hiển thị các nhân viên có mức lương nằm trong khoảng từ 3500 đến 4500

```
SELECT   *   ;
FROM     NHAN_VIEN    ;
WHERE    LUONG        BETWEEN    3500 AND 4500
```

III.4.6. Thỏa mẫu dạng chuỗi

Một trong các hình thức so sánh khác là khả năng so sánh giá trị cột với một số phần của một hằng chuỗi. Hàm LIKE của SQL cho phép chúng ta thực hiện điều này.

Cú pháp như sau:

```
SELECT   <danh sách tên cột>
FROM     <tên bảng>
WHERE    <tên cột>
[NOT] LIKE<'chuỗi ký tự'>;
```

Chú ý:

- <*Chuỗi ký tự*> có thể chứa một vài phần của chuỗi ký tự
- Các ký tự đại diện bao gồm ký tự gạch dưới (_) và phần trăm (%). Ký tự _ thay thế một ký tự riêng rẽ. Ký tự % thay thế một chuỗi ký tự bao gồm không có, một hay nhiều ký tự. Hai ký tự đại diện này có thể dùng kết hợp với nhau.

Ví dụ:

- Hiển thị tất cả các nhân viên có tên bắt đầu chữ T

```
SELECT   *   ;
FROM     NHAN_VIEN;
WHERE    TEN_NV  LIKE   'T%'
```

- Hiển thị tất cả các nhân viên có công việc bắt đầu bằng 2 ký tự QL, theo sau là một ký tự bất kỳ

```
SELECT  *  ;
FROM    NHAN_VIEN   ;
WHERE   CONG_VIEC  LIKE  'QL_'
```

III.5. CÁC HÀM NỘI TẠI

Các hàm nội tại trong SQL thường thao tác nhóm dữ liệu theo cột hơn là theo dòng, do đó còn được gọi là Hàm cột (column functions).

Hàm nội tại được dùng trong lệnh SELECT như là một định danh cột. Cú pháp như sau:

SELECT tên-hàm (tên-cột hay *)

FROM …

Bảng các hàm nội tại trong SQL

AVG	Cho biết giá trị trung bình của một tập giá trị
SUM	Cho biết giá trị tổng cộng của một tập giá trị
MIN	Cho biết giá trị tối thiểu của một tập giá trị
MAX	Cho biết giá trị tối đa của một tập giá trị
COUNT	Cho biết số lượng các phần tử của một tập

Ví dụ:

Tìm mức lương trung bình của tất cả các nhân viên

```
SELECT  AVG(LUONG)   ;
FROM    NHAN_VIEN
```

Ví dụ:

Có bao nhiêu nhân viên trong hồ sơ nhân viên

```
SELECT  COUNT(*)       ;
FROM NHAN_VIEN
```

Ví dụ:

Mức lương cao nhất trong hồ sơ nhân viên

```
SELECT   MAX(LUONG);
FROM     NHAN_VIEN
```

Có thể sau đó chúng ta cần biết tên nhân viên có mức lương cao nhất này. Chúng ta không thể thêm tên cột TEN_NV vào lệnh SELECT vì sẽ dẫn đến lỗi sai của SQL (chú ý là các hàm nội tại làm việc với một nhóm dữ liệu). Cách giải quyết đơn giản là dùng truy vấn con (sub-query) sẽ trình bày trong phần sau.

Ngoài ra, cũng nên lưu ý rằng chúng ta có thể dùng cùng lúc nhiều hàm nội tại trong cùng một lệnh SQL như ví dụ sau:

Ví dụ:

Cho biết mức lương cao nhất, thấp nhất và trung bình trong hồ sơ nhân viên

```
SELECT   MAX(LUONG), MIN(LUONG), AVG(LUONG)      ;
FROM     NHAN_VIEN
```

Cuối cùng, hàm nội tại có thể dùng kết hợp với mệnh đề WHERE

Ví dụ:

Tính tổng lương phải trả cho phòng 40

```
SELECT  SUM(LUONG)   ;
FROM    NHAN_VIEN    ;
WHERE   MA_PHONG=40
```

- **Sử dụnh bí danh cho cột:**

Ta có thể sử dụng bí danh **AS <tên cột>** cho các ***mục*** trong mệnh đề SELECT, điều này đặc biệt có ích khi các *mục* là một biểu thức hoặc chứa một hàm trên vùng và bạn muốn cho nó một cái tên có nghĩa.

Ví dụ:

Ví dụ trên có thể viết lại dung bí danh như sau:

```
SELECT  SUM(LUONG) AS tong_luong      ;
FROM    NHAN_VIEN    ;
WHERE   MA_PHONG=40
```

III.6. CÁC TOÁN TỬ SỐ HỌC

SQL cung ứng cho người sử dụng các toán tử số học dùng trong các lệnh xử lý dữ liệu. Các toán tử này được dùng với dữ liệu loại số và bao gồm toán tử cộng (+), trừ (-), nhân (*), chia (/). Các toán tử này có thể được dùng trong thành phần SELECT hay thành phần WHERE.

Ví dụ:

Hiển thị tên, lương, phụ cấp và thu nhập theo năm của tất cả nhân viên, sắp xếp theo thu nhập trong năm giảm dần.

```
SELECT  TEN_NV, LUONG, PHU_CAP, (LUONG+PHU_CAP)*12 ;
FROM    NHAN_VIEN    ;
ORDER BY 4 DESC
```

Chú ý là đối tượng của mệnh đề ORDER BY là số 4. Điều này có nghĩa biểu thức thu nhập theo năm *(LUONG+PHU_CAP)*12* nằm vị trí thứ tư trong mệnh đề SELECT. Giá trị của biểu thức này được hiển thị như là một cột mới trong bảng nhưng chỉ là cột ” Ảo” (không tồn tại thực tế trong CSDL).

Ngoài ra, các phép tính toán còn được dùng trong thành phần WHERE. Khi dùng trong thành phần WHERE, các kết quả tính toán không hiển thị trong kết quả nhưng trở thành một phần của tiêu chuẩn chọn lựa.

Ví dụ:

Hiển thị tất cả nhân viên có phụ cấp nhiều hơn 15% mức lương.

```
SELECT  TEN_NV, LUONG, PHU_CAP;
FROM    NHAN_VIEN    ;
WHERE   PHU_CAP > 0.15 *LUONG
```

III.7. TRUY VẤN CON

Truy vấn con là một trong các đặc thù mạnh của SQL, cho phép người sử dụng kết hợp nhiều truy vấn vào trong cùng một lệnh SELECT. Khái niệm truy vấn con có nghĩa là kết quả của truy vấn thứ nhất (truy vấn con) được tự động chuyển qua truy vấn cấp sau (truy vấn chính) và chính truy vấn này sẽ cho ra kết quả sau cùng. Thành phần WHERE của truy vấn chính sẽ chứa truy vấn con, cú pháp thông thường như sau:

```
Truy vấn    SELECT   <các tên cột>
 chính      FROM     <tên các bảng>
            WHERE    <điều kiện>  AND <tên cột> <toán tử>
             Truy vấn     ( SELECT  <các tên cột>
               con                 FROM    <tên bảng>
                                   [WHERE  <điều kiện>])
```

Các dạng điều kiện nối với truy vấn con:

- <tên cột> <so sánh> ALL (<***truy vấn con***>)
- <tên cột> <so sánh> ANY | SOME (<***truy vấn con***>)
- <tên cột > [NOT] IN (<***truy vấn con***>)
- [NOT] EXISTS (<***truy vấn con***>)

Ví dụ:

Hiển thị các nhân viên làm cùng công việc trùng với công việc mà Lê Quỳnh Như đang làm.

```
SELECT  TEN_NV, CONG_VIEC;
FROM    NHAN_VIEN    ;
WHERE   CONG_VIEC =  ;
        (  SELECT   CONG_VIEC   ;
           FROM     NHAN_VIEN   ;
           WHERE    TEN_NV = 'Lê Quỳnh Như')
```

Chú ý:

1. Một truy vấn con sẽ có cùng dạng như truy vấn chính
2. Nguyên lệnh truy vấn con sẽ được đặt trong dấu ngoặc đơn
3. Một truy vấn con chỉ cho phép ***một tên cột hay một biểu thức trong mệnh đề SELECT*** của nó
4. Không như truy vấn chính, một truy vấn con ***không được phép có mệnh đề ORDER BY***
5. Kết quả của lệnh truy vấn con phải là loại dữ liệu tương thích với loại dữ liệu trong truy vấn chính

6. Các truy vấn con nên dùng điều kiện chọn lựa trong cả hai mệnh đề WHERE và HAVING. Truy vấn con phải được đặt bên phải điều kiện chọn lựa
7. Truy vấn con **không thể có các hàm LIKE và BETWEEN**
8. Trong trường hợp dùng tóan tử = để nối kết với truy vấn con, nếu kết quả của truy vấn con là một tập hợp thì thay toán tử **=** bởi **IN**
9. Các truy vấn trong FoxPro 2.6 không thể lồng nhau quá 2 mức

Ví du:

Hiển thị các nhân viên có mức lương lớn hơn mức lương tối thiểu.

```
SELECT  TEN_NV, LUONG       ;
FROM      NHAN_VIEN     ;
WHERE    LUONG >         ;
          (SELECT  MIN(LUONG)         ;
            FROM     NHAN_VIEN)
```

Ví du:

Hiển thị nhân viên có mức lương cao nhất trong hồ sơ nhân viên.

```
SELECT  TEN_NV  ;
FROM      NHAN_VIEN     ;
WHERE    LUONG =         ;
          (      SELECT    MAX(LUONG)       ;
                  FROM      NHAN_VIEN)
```

III.8. GOM NHÓM CÁC DÒNG

Phần này, chúng ta sẽ làm quen với một đặc thù rất mạnh Của SQL là khái niệm gom nhóm. Đặc tính gom nhóm cho phép chúng ta thực hiện các chức năng trên một nhóm các dòng như là một dòng riêng biệt.

Thành phần GROUP BY cho phép gom nhóm các dòng có liên quan. Cú pháp như sau:

```
SELECT  <các tên cột>, <hàm-nội-tại(thông-số)>
FROM      <tên bảng>
GROUP BY  <các tên cột>
```

Ví du:

Hiển thị mức lương trung bình của từng phòng.

```
SELECT     MA_PHONG, AVG(LUONG)        ;
FROM       NHAN_VIEN  ;
GROUP BY MA_PHONG
```

Trong ví dụ này, MA_PHONG là cột mà căn cứ trên đó chúng ta gom nhóm các dòng liên quan. Khi các phòng đã được gom nhóm lại với nhau, hàm nội tại AVG sẽ tính cột lương trung bình cho từng phòng

Chú ý:

1. Tên cột xác định trong mệnh đề GROUP BY là cột cơ sở để phân loại nhóm

2. Nếu trong lệnh truy vấn có mệnh đề GROUP BY thì các cột xác định trong mệnh đề SELECT phải chứa các cột trong GROUP BY hay có các hàm nội tại ứng dụng trên các cột đó.
3. Tương tự như ORDER BY, chúng ta có thể GROUP BY bởi số thứ tự cột trong lệnh truy vấn đơn.

III.8.1. Mệnh đề HAVING

Trong trường hợp cần xác định *điều kiện chọn lựa cho một nhóm các dòng*, chúng ta có thể dùng thành phần HAVING. Thành phần HAVING có thể dùng để chọn lọc các nhóm có trong kết quả của lệnh truy vấn. Cú pháp như sau:

SELECT <các tên cột>, <hàm-nội-tại(thông-số)>
FROM <tên bảng>
GROUP BY <các tên cột>
HAVING <điều kiện>

Ví dụ:

Chúng ta lại tiếp tục xét ví dụ về mức lương trung bình của một phòng. Giả sử chúng ta muốn tìm xem các phòng có mức lương trung bình lớn hơn 3000.

SELECT *MA_PHONG*, AVG*(LUONG)* ;
FROM *NHAN_VIEN* ;
GROUP BY *MA_PHONG* HAVING AVG*(LUONG)>3000*

Trong trường hợp này, mệnh đề HAVING giúp chọn lọc kết quả của mệnh đề GROUP BY và chỉ hiển thị những nhóm thỏa điều kiện chọn lựa AVG(LUONG)>3000

III.8.2. Sử dụng mệnh đề WHERE

Mệnh đề WHERE có thể dùng chung với các chức năng GROUP BY và HAVING. Trong trường hợp này, nó thực hiện việc chọn lọc đầu tiên bằng cách loại bỏ những dòng không thỏa điều kiện trong mệnh đề WHERE. Sau đó, việc gom nhóm và chọn lọc trên nhóm sẽ thực hiện trên các dòng còn lại.

Cú pháp như sau:

SELECT <các tên cột>, <hàm-nội-tại(thông-số)>
FROM <tên bảng>
[WHERE <điều kiện>]
[GROUP BY <các tên cột>]
[HAVING <điều kiện>];

Ví dụ:

Tiếp tục ví dụ trên, chúng ta chỉ xem các phòng bao gồm 10, 30 và 50. Hiển thị các phòng đó nếu có mức lương trung bình lớn hơn 3000

SELECT *MA_PHONG, AVG(LUONG)* ;
FROM *NHAN_VIEN* ;

WHERE ***MA_PHONG IN (10, 30, 50)*** ***;***
GROUP BY ***MA_PHONG*** **HAVING** ***AVG(LUONG) > 3000***

III.9. NỐI KẾT CÁC BẢNG

Trong các phần trước, chúng ta sẽ thực hiện các truy vấn chỉ trên một bảng duy nhất. Như vậy trong trường hợp cần xử lý thông tin trên nhiều bảng thì sẽ thực hiện như thế nào?

SQL cung cấp một đặc thù gọi là *nối kết* (joining) cho phép chúng ta dễ dàng trích thông tin từ hai hay nhiều bảng để tạo ra kết quả như mong muốn.

Khi thực hiện việc nối kết, đầu tiên, người sử dụng sẽ xác định các tên cột trong mệnh đề SELECT, và các bảng được khai báo trong mệnh đề FROM như các truy vấn khác. Sau đó, điều kiện nối kết sẽ xác định để nối kết các bảng. Điều kiện kết nối là một thành phần của mệnh đề WHERE định nghĩa một điều kiện trên hai cột thuộc hai bảng khác nhau. Các thành phần khác của WHERE được gọi là điều kiện chọn lựa.

Sử dụng đặc thù nối kết:

SELECT ***NHAN_VIEN.MA_PHONG, TEN_PHONG, TEN_NV***

Hình 1

FROM ***NHAN_VIEN,PHONG***

Hình 2

WHERE ***NHAN_VIEN.MA_PHONG = PHONG.MA_PHONG***

Hình 3

AND ***TEN_NV = 'Lê Quỳnh Như'***

Hình 4

Chúng ta đã dùng các bảng NHAN_VIEN và PHONG để truy vấn thông tin riêng rẽ từng bảng. Bây giờ, giả sử chúng ta cần truy vấn thông tin trên cả hai bảng đồng thời, ví dụ như tìm tên của phòng mà Lê Quỳnh Như đang làm việc. Cách giải quyết vấn đề như sau:

- Đầu tiên, như các truy vấn thông thường, các tên cột được hiển thị sẽ được liệt kê trong mệnh đề SELECT (hình 1). Chú ý là số phòng(MA_PHONG) được thêm tiếp đầu ngữ (NHAN_VIEN). Khi nào có tên cột trùng lắp trong lệnh truy vấn nối kết, nó phải được xác định tên bảng như tiếp đầu ngữ.
- Sau đó, danh sách các bảng được xác định trong thành phần FROM và ngăn cách bởi dấu phẩy (,). Thứ tự các tên bảng không quan trọng (hình 2)
- Kế tiếp, xác định điều kiện nối kết. Điều kiện này sẽ nối kết hai vùng chung trong mỗi bảng. SQL sẽ so khớp từng dòng trong hai bảng trên và chỉ hiển thị các cột MA_PHONG, TEN_PHONG và TEN_NV (hình 3)

- Cuối cùng, xác định điều kiện chọn lựa. Một khi các dòng đã được so khớp theo điều kiện nối kết, điều kiện chọn lựa sẽ chọn lọc và hiển thị chỉ những dòng mà TEN_NV = 'Lê Quỳnh Như' (hình 4)

```
SELECT   NHAN_VIEN.MA_PHONG, TENPHONG, TEN_NV
FROM     NHAN_VIEN, PHONG
WHERE    NHAN_VIEN.MA_PHONG = PHONG.MA_PHONG
AND      TEN_NV = 'Lê Quỳnh Như'
```

Hình 5

Hình 5 trình bày toàn bộ lệnh của ví dụ trên. Như vậy, đặc thù nối kết của SQL cho phép người sử dụng trích thông tin chọn lọc từ nhiều bảng thành một bảng kết quả duy nhất

Toán tử bằng nhau(=) dùng để nối kết ***hai cột chung*** là một trong các điều kiện kết nối hay gặp nhất và cũng hiệu quả nhất. Ngoài ra, các toán tử như lớn hơn (>), nhỏ hơn (<) cũng có thể dùng để nối kết.

Chúng ta hãy tiếp tục với một ví dụ khác. Trong ví dụ này, nối kết sẽ phát sinh ra kết quả là một bảng gồm nhiều dòng.

Ví dụ:

Tạo ra bảng báo cáo gồm danh sách mã phòng, tên phòng, tên nhân viên và mức lương. Sắp thứ tự kết quả theo mã phòng.

```
SELECT NHAN_VIEN.MA_PHONG, TEN_PHONG, TEN_NV, LUONG    ;
FROM    NHAN_VIEN, PHONG  ;
WHERE  NHAN_VIEN.MA_PHONG = PHONG.MA_PHONG ORDER BY 1
```

Về mặt lý thuyết, số lượng các bảng có thể nối kết trong cùng một lệnh SELECT là không giới hạn. Tuy nhiên, nhiều hệ quản trị CSDL giới hạn trong mức 16 bảng

Cũng nhắc thêm rằng, việc nối kết có thể kết hợp với các chức năng khác như GROUP BY, HAVING, hàm nội tại và tính toán.

- **Sử dụnh bí danh cho bảng:**

Ta cũng có thể sử dụng bí danh cho các bảng trong mệnh đề WHERE. Nếu bạn đã sử dụng bí danh cho bảng, bạn phải sử dụng bí danh này trong toàn bộ lệnh SELECT

Ví dụ:

Ví dụ trên có thể được viết lại bằng cách sử dụng bí danh như sau:

```
SELECT a.MA_PHONG AS maphong, TEN_PHONG, TEN_NV, LUONG ;
FROMNHAN_VIEN a, PHONG b ;
WHERE      a.MA_PHONG = b.MA_PHONG   ORDER BY maphong
```

III.10. CẬP NHẬT CSDL

III.10.1. Lệnh INSERT

Một trong hình thức khác của lệnh INSERT là sử dụng chúng trong lệnh SELECT để thêm giá trị từ một bảng vào một bảng khác.

Các giá trị của cột sẽ được thêm vào theo thứ tự đã được dùng trong lệnh định nghĩa bảng CREATE TABLE. Trong ví dụ tiếp theo, chúng ta sẽ thêm giá trị vào bảng THANG_CAP như sau:

Ví dụ:

INSERT INTO ***THANG_CAP*** **;**
SELECT ***TEN_NV, MA_PHONG, MA_NV ;***
FROM ***NHAN_VIEN*** **WHERE** ***CONG_VIEC*** **= 'TPG'**

III.10.2. Lệnh UPDATE

Lệnh UPDATE cho phép người sử dụng sửa đổi nội dung của một hay nhiều dòng của một bảng. Ví dụ như chúng ta muốn tăng lương cho tất cả nhân viên thư ký lên 500

Ví dụ:

UPDATE NHAN_VIEN ;
SET LUONG = LUONG + 500 WHERE CONG_VIEC = 'TKY'

III.10.3. Lệnh DELETE

Lệnh DELETE huỷ bỏ một hay nhiều dòng trong một bảng. Cũng như lệnh UPDATE, mệnh đề WHERE xác định các dòng nào sẽ được xử lý.

Ví dụ:

Nhân viên Trần Hồng đã nghỉ việc. Huỷ bỏ thông tin về anh ta trong bảng NHAN_VIEN

DELECT FROM *NHAN_VIEN*
WHERE TEN_NV = 'Trần Hồng';

Chú ý:

Nhiều dòng có thể huỷ bỏ đồng thời nếu các dòng này thoả điều kiện chọn lựa. Hãy cẩn thận khi xác định mệnh đề WHERE trong lệnh DELETE, nếu không chúng ta có thể huỷ nhầm các dòng không muốn huỷ.

III.11. TÌM KIẾM CÓ CHỨA PHÉP TÍNH TẬP HỢP

Ý nghĩa*:*

- UNION: Hợp
- INTERSECT: Giao

- MINUS : Trừ

Cũng giống với các phép tính tập hợp của ĐSQH, các phép tính này cũng đòi hỏi sự tương thích giữa hai quan hệ. Trong Foxpro, hai thuộc tính tương thích nhau nếu chúng có cùng kiểu và độ rộng.

Nhưng trong Foxpro 2.6, phép tính Intersect chạy không ổn định và phép tính Minus không thực hiện được.

Cú pháp:

```
(SELECT ...)  ;
        UNION   ;
              (SELECT ...)
```

Ví dụ:

Cho CSDL sau:

SINHVIEN (MASV, HOTEN, NSINH, QUEQUAN, HOCLUC)

DETAI (MADT, TENDT, KINHPHI, CNHIEM)

SV_DT (MASV, MADT,NOI_AD, KETQUA)

Cho biết danh sách tất cả các sinh viên và giáo viên chủ nhiệm các đề tài.

```
(SELECT Hoten FROM  Sinhvien) ;
UNION ;
(SELECT  Cnhiem FROM Detai)
```

CHƯƠNG IV - RÀNG BUỘC TOÀN VẸN

IV.1. RÀNG BUỘC TOÀN VẸN (Intergrety constraint)

IV.1.1 Khái niệm

Trong một CSDL, luôn luôn tồn tại rất nhiều mối quan hệ ràng buộc giữa các thuộc tính, các bộ với nhau,... Các mối quan hệ này là các điều kiện bất biến mà tất cả các bộ của các quan hệ trong CSDL phải thỏa mãn ở bất cứ thời điểm nào. Các điều kiện này được gọi là ràng buộc toàn vẹn (RBTV).

Ví dụ:

Trong CSDL "Quản Lý Sinh Viên" như sau:

- SV (MASV, HOTEN_SV, NU, NGSINH, ĐCHI_SV, TINH, MAKHOA)
- KHOA (MAKHOA, TENKHOA, SO_CMND)
- MONHOC (MAMH, TENMH, SOTIET)
- KETQUA (MASV, MAMH, LANTHI, ĐIEM)

Có các ràng buộc:

- C1 : Mỗi sinh viên có một mã số riêng biệt không trùng với bất cứ sinh viên nào khác.
- C2 : Mỗi sinh viên phải đăng ký vào một khoa của trường.
- C3 : Mỗi sinh viên được thi tối đa hai lần cho 1 môn.

IV.1.2 Các yếu tố của RBTV

IV.1.2.1 Điều kiện của RBTV:

Điều kiện của RBTV được biểu diễn bằng ngôn ngữ tự nhiên, ngôn ngữ giả, đại số quan hệ, phụ thuộc hàm,...

Ví dụ:

Các điều kiên trên được biểu diễn như sau:

C_1: $\forall$ u $\in$ SV, $\forall$ v $\in$ SV: u<>v $\Leftrightarrow$ u.MASV <> v.MASV

C_2 : SV[MAKHOA] $\subseteq$ KHOA[MAKHOA]

C_3 : $\forall$ sv $\in$ KETQUA Card({k $\in$ KETQUA | k.MASV = sv.MASV}) <= 2

IV.1.2.2 Bối cảnh của một RBTV:

Là những quan hệ mà RBTV đó có hiệu lực; hay nói cách khác, đó là những quan hệ cần phải sử dụng để kiểm tra RBTV đó.

Ví dụ:

Bối cảnh của C_1 là quan hệ SV;

Bối cảnh của C_2 là quan hệ SV và KHOA;

Bối cảnh của C_3 là quan hệ KETQUA

IV.1.2.3 Bảng tầm ảnh hưởng của RBTV:

Khi thực hiện một thao tác cập nhật trên bối cảnh của một RBTV C có thể dẫn đến C bị vi phạm. Vì vậy, người ta lập bảng tầm ảnh hưởng cho từng RBTV để xác định thời điểm cần kiểm tra RBTV đó.

Bảng tầm ảnh hưởng của một RBTV Ci có dạng như sau:

C_i	Thêm	Sửa	Xóa
R_1	+	+	-
R_2	-	-	+
...			
R_n	-	+	-

C_i : có bối cảnh là R1, R_2, ..., R_n.

Dấu + : cần phải kiểm tra C_i.

Dấu - : không cần kiểm tra C_i.

Ví dụ: Bảng tầm ảnh hưởng của C1,C2,C3 như sau:

C_1	Thêm	Sửa	Xóa
SV	+	- (*)	-

C_2	Thêm	Sửa	Xóa
SV	+	+	-
KHOA	-	-(*)	+

C_3	Thêm	Sửa	Xóa
KETQUA	+	- (*)	-

(*): theo quy ước, không được sửa đổi trị của thuộc tính khóa

Trên cơ sở các bảng tầm ảnh hưởng của từng RBTV, người ta đưa ra bảng tầm ảnh hưởng tổng hợp của tất cả các RBTV. Với các cột là các thao tác cập nhật trên từng quan hệ, và các dòng là các RBTV.

Ví dụ Từ các bản tầm ảnh hưởng trên, ta có bảng tầm ảnh hưởng tổng hợp như sau:

	SV			KHOA			KETQUA		
	T	S	X	T	S	X	T	S	X
C_1	+	-	-						
C_2	+	+	-	-	-	+			
C_3							+	-	-

Dựa vào bảng tầm ảnh hưởng tổng hợp này, chúng ta sẽ dễ dàng xác định cần phải tiến hành kiểm tra các RBTV nào khi người sử dụng thực hiện một thao tác cập nhật.

IV.1.3 Phân loại các RBTV:

Trong quá trình phân tích và thiết kế CSDL, người phân tích phải phát hiện tất cả các RBTV tiềm ẩn bên trong CSDL. Việc phân loại các RBTV là một cách tiếp cận giúp người phân tích thiết kế có được một định hướng trong việc phát hiện những RBTV của CSDL. Các RBTV có thể chia làm hai loại chính:

- RBTV có bối cảnh là một quan hệ và
- RBTV có bối cảnh là nhiều quan hệ

IV.1.3.1 RBTV có bối cảnh là một quan hệ:

IV.1.3.1.1 RBTV về miền trị:

RBTV về miền trị liên quan đến miền giá trị của một thuộc tính.

Ví dụ: NGSINH > date()
0 <= ĐIEM <= 10
0 < SOTIET <= 180

IV.1.3.1.2 RBTV liên thuộc tính:

RBTV liên thuộc tính là mối liên hệ giữa các thuộc tính trong cùng một lược đồ quan hệ.

Ví dụ:

CSDL "QLBH" như sau:

KHACH (MAKH, TENKH, ĐCHI_KH, ĐTHOAI_KH, CONGNO)

DATHANG (SO_DDH, MAHH, SL_DAT, NGAY_DH, MAKH)

HOADON (SO_HD, NGAY_HD, SO_DDH, NGAYXUAT, TRIGIA)

PHIEUTHU(SO_PT, NGAYTHU, SOTIEN)

CT_HD (SO_HD, MAHH, GIA_BAN, SL)

Trong quan hệ HOADON có ràng buộc: "*hàng hóa chỉ được xuất sau khi lập hóa đơn*"

$\forall$ hd $\in$ HOADON, hd.NGAY_XUAT $\geq$ hd.NGAY_HD

IV.1.3.1.3 RBTV liên bộ:

Là sự ràng buộc giữa các bộ bên trong một quan hệ, trong đó phổ biến là RBTV *về khóa nội.*

Ví dụ:

- MASV là duy nhất trong quan hệ SV. (MASV là khóa của quan hệ SV).
- Mỗi sinh viên được thi tối đa 2 lần cho một môn.

RBTV về khóa nội là một RBTV rất phổ biến, chúng thường được biểu diễn bằng các phụ thuộc hàm, và thường được các hệ quản trị CSDL hỗ trợ tự động kiểm tra như Visual Fopro, Access, ...

IV.1.3.2 RBTV có bối cảnh gồm nhiều quan hệ:

IV.1.3.2.1 RBTV về phụ thuộc tồn tại (RBTV về khóa ngoài):

Khóa ngoài: Thuộc tính A của một quan hệ R được gọi là khóa ngoài nếu A là thuộc tính khóa của quan hệ R' nào đó. Vì vậy, khi cập nhật dữ liệu cho thuộc tính khóa ngoài này, người ta phải kiểm tra giá trị đó đã tồn tại ở thuộc tính khóa nội của R' chưa?

Ví dụ: Nếu $\exists$ kq $\in$ KETQUA, kq.MASV ='01'

Thì phải $\exists$ sv $\in$ SV: sv.MASV ='01'

IV.1.3.2.2 RBTV liên thuộc tính, liên quan hệ:

Là mối liên hệ giữa các thuộc tính của nhiều quan hệ khác nhau.

Ví dụ: Giữa hai quan hệ DATHANG và HOADON của CSDL "QLBH", có ràng buộc như sau:

Nếu $\exists$ hd $\in$ HOADON, dh $\in$ DATHANG, hd.SO_DDH = dh.SO_DDH thì
dh.NGAY_DH <= hd. NGAY_HD

IV.1.3.2.3 RBTV liên bộ, liên quan hệ:

RBTV loại này có tác dụng trên từng nhóm các bộ của nhiều quan hệ khác nhau (thường là hai quan hệ).

Ví dụ: Giữa hai quan hệ HOADON và CT_HD:

Có ràng buộc: Mỗi hóa đơn phải có ít nhất một mặt hàng

IV.1.3.2.4 RBTV về thuộc tính tổng hợp:

RBTV này được xác định trong trường hợp một thuộc tính A của một quan hệ R được tính toán từ các thuộc tính của các quan hệ khác.

Ví dụ:

- Trị giá của hóa đơn bằng tổng các GIABAN * SL của các mặt hàng trong hóa đơn đó:

$$hd.TRIGIA = \sum_{cthd.SO\_HD = hd.SO\_HD} (cthd.GIABAN * cthd.SL)$$

- Hay số tiền công nợ của một khách hàng A sẽ bằng hiệu số giữa tổng giá trị của các hóa đơn bán cho khách hàng A và tổng số tiền thu của khách hàng đó:

$$\forall kh \in KHACH: \quad kh.CONGNO = \sum_{hd \in H_{kh}} hd.TRIGIA - \sum_{pt \in P_{kh}} pt.SOTIEN$$

với H_{kh} = (HOADON * DATHANG) (MAKH = kh.MAKH)

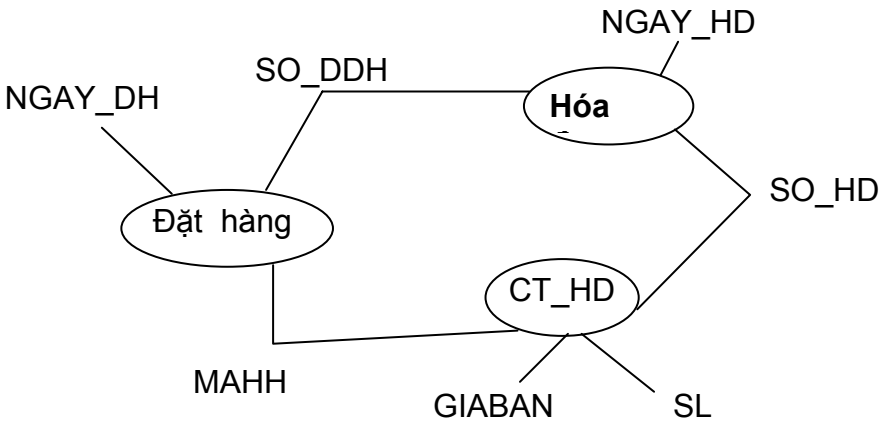
P_{kh} = PHIEUTHU(MAKH = kh.MAKH)

IV.1.3.2.5 RBTV do có chu trình trong đồ thị biểu diễn của lược đồ CSDL:

Một lược đồ CSDL có thể được biểu diễn bằng một đồ thị vô hướng. Trong đồ thị này, ta có 2 loại nút: nút thuộc tính và nút quan hệ. Một cung vô hướng trong đồ thị nối 1 nút thuộc tính A với một nút quan hệ R khi A thuộc R.

Ví dụ:

Một phần đồ thị biểu diễn cho CSDL "QLBH" có dạng như sau:



Trong hình vẽ trên, chúng ta nhận thấy đồ thị biểu diễn có chứa một chu trình gồm 3 quan hệ DATHANG, HOADON, CT_HD. Như vậy, CSDL sẽ phải có một RBTV thuộc 1 trong 3 trường hợp sau:

- Một hóa đơn thực hiện cho một đơn đặt hàng chỉ giao những mặt hàng mà khách yêu cầu và phải gồm đầy đủ tất cả những mặt hàng có trong đơn đặt hàng đó.
- Một hóa đơn thực hiện cho một đơn dặt hàng chỉ giao những mặt hàng mà khách yêu cầu và có thể không giao đầy đủ tất cả các những mặt hàng có trong đơn đặt hàng đó.
- Một hóa đơn thực hiện cho một đơn đặt hàng có thể gồm tùy ý các mặt hàng dù có hay không trong đơn đặt hàng của khách.

RBTV này thể hiện sự tương giao giữa 2 tập hợp A và B với:

A = DATHANG[SO_DDH, MAHH]

B = (HOADON * CT_HD)[SO_DDH, MAHH]

- Trường hợp thứ nhất: A=B
- Trường hợp thứ hai: A $\supset$ B
- Trường hợp thứ ba: A $\not\subset$ B và B $\not\subset$ A.

IV.2. PHỤ THUỘC HÀM

PTH là một công cụ để biểu diễn một số ràng buộc toàn vẹn.

Ví dụ:

- Trong quan hệ SV(MASV, HOTEN,NGSINH, ...),
  - ⇨ Có ràng buộc: không có hai sinh viên trùng MASV
- Trong quan hệ NCC(TEN_NCC, TEN_HG, GIA, DCHI_NCC)
  - ⇨ Có ràng buộc: với mỗi cặp giá trị (TEN_NCC, TEN_HG) ta có một GIA duy nhất. Hay, mỗi NCC chỉ có một địa chỉ duy nhất, ...

IV.2.1 Định nghĩa:

Định nghĩa:

Cho tập thuộc tính U; X,Y là tập con của U. Ta gọi $X \rightarrow Y$ là PTH nếu với mỗi cặp bộ u,v thuộc RBTV, ta có: u.X = v.X thì u.Y = v.Y

Ví dụ:

MASV $\rightarrow$ HOTEN
TEN_NCC, TEN_HG $\rightarrow$ GIA
TEN_NCC $\rightarrow$ DCHI_NCC

IV.2.2 Tính chất của PTH:

1. *F1: tính phản xạ*
   Nếu $X \supseteq Y$ thì $X \rightarrow Y$
2. *F2: tính bắc cầu*
   Nếu $X \rightarrow Y$, $Y \rightarrow Z$ thì $X \rightarrow Z$
3. *F3: tính mở rộng hai vế*:
   Nếu $X \rightarrow Y$ thì $XZ \rightarrow YZ$
4. *F4: tính tựa bắc cầu*
   Nếu $X \rightarrow Y$, $YZ \rightarrow W$ thì $XZ \rightarrow W$
5. *F5: tính phản xạ chặt*
   $X \rightarrow X$
6. *F6: Mỏ rộng vế trái, thu hẹp vế phải*
   Nếu $X \rightarrow Y$ thì mọi Z, W thuộc U, ta có: $XZ \rightarrow Y \setminus W$
7. *F7: Cộng tính đầy đủ*
   Nếu $X \rightarrow Y$, $Z \rightarrow W$ thì $XZ \rightarrow YW$
8. *F8: Mở rộng vế trái*
   Nếu $X \rightarrow Y$ thì $XZ \rightarrow Y$
9. *F9 Cộng tính ở vế phải*
   Nếu $X \rightarrow Y$, $X \rightarrow Z$ thì $X \rightarrow YZ$
10. *F10: Bộ phận ở vế phải*
    Nếu $X \rightarrow YZ$ thì $X \rightarrow Y$ ($X \rightarrow Z$)
11. *F11: tính tích lũy*
    Nếu $X \rightarrow YZ$, $Z \rightarrow AW$ thì $X \rightarrow YAW$.

Chứng minh:

F1: Giả sử X, Y $\subset$ U ; X $\supset$ Y và R là một quan hệ trên U . Nếu u & v là 2 bộ trong R thỏa u.X = v. X thì u.Y = v.Y do Y $\subseteq$ X.
Vậy: R thoả PTH X $\rightarrow$ Y (đpcm).

F2 : Giả sử u,v ε R(U) và u.X = v.X, trong đó R là quan hệ thoả các phụ thuộc hàm X $\rightarrow$ Y và Y $\rightarrow$ Z. khi đó ta có u.Y = v.Y và do đó u.Z = v.Z.
Vậy: R thỏa pth X $\rightarrow$ Z (đpcm).

F3 : Giả sử quan hệ R(U) thỏa phụ thuộc hàm X $\rightarrow$ Y và u, v là hai bộ của R thỏa điều kiện u.XZ = v.XZ => u.X = v.X và u.Z = v.Z (1)
Do R thỏa X $\rightarrow$ Y nên từ u.X = v.X => u.Y = v.Y (2)
Kết hợp (1) và (2) ta được u.YZ = v.YZ.
Vậy quan hệ R thỏa pth XZ $\rightarrow$ YZ (đpcm).

IV.3. BAO ĐÓNG CỦA TẬP THUỘC TÍNH

IV.3.1 Định Nghĩa:

Định nghĩa LĐQH: LĐQH là một bộ đôi α = <U,F> với

U : là tập thuộc tính

F : là tập các PTH trên U.

Định nghĩa bao đóng: Cho một LĐQH α = <U, F>, trong đó:

U = {A_1, A_2, ..., A_n }

F = { $L_i \rightarrow R_i$ | L_i, $R_i \subseteq$ U, i= 1,…, m }

và tập thuộc tính X $\subseteq$ U. Khi đó

Bao đóng của tập X đối với tập PTH F, ký hiệu: X^+, và X^+ = { A $\subseteq$ U | X $\rightarrow$ A}

* **Bổ đề**: Cho các tập thuộc tính X,Y,Z $\subseteq$ U. Khi đó,

X $\rightarrow$ YZ $\Leftrightarrow$ X $\rightarrow$ Y $\wedge$ X $\rightarrow$ Z

Chứng minh:

- **(chiều =>)**

  Ta có: X $\rightarrow$ YZ (1)
  Theo tính phản xạ, ta cũng có: YZ $\rightarrow$ Y (2)
  và YZ $\rightarrow$ Z (3)
  Áp dụng tính chất bắc cầu cho (1) và (2) => X $\rightarrow$ Y
  và (1) và (3) => X $\rightarrow$ Z

- **(chiều <=)**

  Ta có: X $\rightarrow$ Y và X $\rightarrow$ Z

  Áp dụng tính chất cộng tính ở vế phải(F9) ta được: X $\rightarrow$ YZ
  (đpcm)

IV.3.2 Thuật toán tìm bao đóng:

Input: tập thuộc tính U, tập các PTH F và một tập thuộc tính $X \subseteq U$

Output: X^+ thỏa F

Phương pháp:

$X_m = X$;

Repeat

$X_c = X_m$;

For (mỗi phụ thuộc hàm $L_i \rightarrow R_i$ thuộc F) do

If ($L_i \subseteq Xc$) then

$X_m = X_c \cup R_i$

Until ($X_m = X_c$)

Return X_m

Ví dụ:

Cho tập thuộc tính: U ={A,B,C,D,E,F,G,H,I,J}

Và tập các phụ thuộc hàm F = { $AB \rightarrow DG$, $BD \rightarrow EH$,
$C \rightarrow ADE$, $DH \rightarrow BCE$
$E \rightarrow AI$ }

X= AEDB. Hãy tìm X^+

Giải:

Bước 0 : X_0 = AEDB

Bước 1:

Xét PTH $AB \rightarrow DG$, có $AB \in X_0$

Xét PTH $BD \rightarrow EH$, có $BD \in X_0$

Xét PTH $E \rightarrow AI$, có $E \in X_0$

➔ $X_1 = X_0 \cup DG \cup EH \cup AI$ = AEDBGHI

Bước 2:

Xét PTH $DH \rightarrow BCE$, có $DH \in X_1$

➔ $X_2 = X_1 \cup BCE$=AEDBGHIC

Bước 3:

Xét PTH $C \rightarrow ADE$, có $C \in X_2$

➔ $X_3 = X_2 \cup ADE$=AEDBGHIC = X_2 ➔ dừng

Vậy $(AEDB)^+$= AEDBGHIC

IV.3.3 Các tính chất của bao đóng:

1. Tính phản xạ $X^+ \supseteq X$
2. Tính đơn điệu $X \subseteq Y \Rightarrow X^+ \subseteq Y^+$
3. Tính lũy đẳng $X^{++} = X^+$
4. $(XY)^+ \supseteq X^+ Y^+$

Chứng minh:

$XY \supseteq X \Rightarrow (XY)^+ \supseteq X^+$ (i)

$XY \supseteq Y \Rightarrow (XY)^+ \supseteq Y^+$ (ii).

(i) & (ii) $\Rightarrow (XY)^+ \supseteq X^+Y^+$.

Phản ví dụ: Chứng minh không tồn tại $(XY)^+ \subset X^+ Y^+$

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

X = { A }, Y = { B } $\Rightarrow$ XY = { AB }

Ta có: $X^+ = A^+ = A$, $Y^+ = B^+ = B$ $\Rightarrow X^+ Y^+ = AB$

Nhưng: $(XY)^+ = (AB)^+ = ABC = U$

$\Rightarrow (XY)^+ \not\subset X^+ Y^+$

5. $(X^+ Y)^+ = (XY^+)^+ = (XY)^+$

   Chứng minh:

   $X \subseteq X^+$ (Theo tính phản xạ)

   $XY \subseteq X^+ Y$

   $(XY)^+ \subseteq (X^+ Y)^+$ (Theo tính đơn điệu) (1)

   Chứng minh chiều ngược lại

   Do $XY \supseteq X$

   $(XY)^+ \supseteq X^+$, $(XY)^+ \supseteq Y^+ \supseteq Y$ $\Rightarrow (XY)^+ \supseteq X^+Y$

   $\Rightarrow (XY)^{++} \supseteq (X^+Y)^+$

   $\Rightarrow (XY)^+ \supseteq (X^+ Y)^+$ (2)

   Từ (1) & (2) => $(XY)^+ = (X^+Y)^+$

6. $X \rightarrow Y \Leftrightarrow Y \subseteq X+$
7. $X \rightarrow Y \Leftrightarrow Y+ \subseteq X+$
8. $X \rightarrow X+$ và $X+ \rightarrow X$
9. $X+ = Y+ \Leftrightarrow X \rightarrow Y, Y \rightarrow X$

IV.4. BAO ĐÓNG CỦA TẬP CÁC PTH

IV.4.1 Định nghĩa

Cho tập pth F trên tập thuộc tính U. Bao đóng của F, kí hiệu F^+ là tập nhỏ nhất các phụ thuộc hàm trên U thoả 2 tính chất sau:

- $F^+ \supseteq F$
- Khi áp dụng các tính chất F 1, F 2, F3 (của phụ thuộc hàm) cho F^+ thì ta không thu được phụ thuộc hàm mới nào ngoài F^+.

IV.4.2 Tính chất của bao đóng của tập PTH

1. Tính phản xạ: $F^+ \supseteq F$
2. Tính đơn điệu: Nếu $F \subseteq G$ thì $F^+ \subseteq G^+$
3. Tính lũy đẳng: (F+)+ = F+
4. $(FG)^+ \supseteq F^+G^+$

   Chứng minh: Từ $FG \supseteq F$, theo tính chất đơn điệu => $(FG)^+ \supseteq F^+$

   Tương tự $(FG)^+ \supseteq G^+$ => $(FG)^+ \supseteq F^+G^+$

   Nhưng trong trường hợp ngược lại: $(FG)^+ \subseteq F^+G^+$ thì không thể, xét ví dụ sau:

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

   Và giả sử dom(A) = dom(B) = dom(C) = {abc}

   Ta có: FG = { $A \rightarrow B$, $B \rightarrow C$ }.

   Vậy theo tính chất bắc cầu: $A \rightarrow C \in (FG)^+$

   Ta chứng minh $A \rightarrow C \notin F^+ G^+$. Thật vậy R và S như sau:

   R (A B C) S (A B C)

   a b a a b a

   a b c a c c

   c b a

   R thỏa $A \rightarrow B$ và S thỏa $B \rightarrow C$ Nhưng chúng không thỏa $A \rightarrow C$. Vậy

   $A \rightarrow C \notin G^+ \Rightarrow A \rightarrow C \notin F^+ G^+$.
5. $(F^+G)^+ = (FG^+)^+ = (FG)^+$

   Chứng minh: Ta chứng minh $(F^+G)^+ = (FG)^+$

   Theo tính chất phản xạ: $F \subseteq F^+$ do đó $FG \subseteq F^+G$

   Theo tính chất đơn điệu $(FG)^+ \subseteq (F^+G)^+$

   Mặt khác do $F \subseteq FG$, $G \subseteq FG$, theo tính chất đơn điệu phản xạ

   ⇨ $F^+ \subseteq (FG)^+$ và $G \subseteq G^+ \subseteq (FG)^+$ (*đpcm*).

Trên lí thuyết, ta hoàn toàn có thể xác định được 1 thủ tục tính bao đóng F+. Nhưng dù tập thuộc tính U và tập PTH F là hữu hạn, thì bài toán tìm F^+ vẫn khó thực hiện vì tập U và F trong thực tế rất lớn. Do đó, có thể dẫn đến sự bùng nổ tổ hợp .

Thay vào đó người ta thường xét bài toán khác:***" Kiểm tra 1 phụ thuộc hàm f có thuộc F^+ hay không ?"*** Bài toán này có lẽ thiết thực hơn bài toán tìm F^+.

Để giải bài toán thành viên, người ta dựa vào tính chất: $\mathbf{X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X^+}$

Ví dụ:

F = { $A \rightarrow BC$, $B \rightarrow D$

$CD \rightarrow E$, $BE \rightarrow GA$}

Hãy xác định f: $AE \rightarrow DGC$ có thể suy dẫn từ F hay không ?

Hay f : $AE \rightarrow DGC \in F^+$ hay không ?

Ta có: $(AE)^+ = AEBCDG \supseteq DGC$

Vậy, kết luận: AE DGC $\in F^+$

IV.5. TẬP PHỤ THUỘC HÀM TỐI TIỂU

IV.5.1 Định nghĩa

Hai tập phụ thuộc hàm tương đương: Hai tập phụ thuộc hàm F và G được gọi là tương đương nếu $F^+ = G^+$. Khi đó, ta nói F phủ G hay G phủ F. Kí hiệu: $F \equiv G$.

Tập phụ thuộc hàm tối tiểu: Tập F được gọi là tập phụ thuộc hàm tối tiểu nếu:

- Vế phải của các phụ thuộc hàm trong F chỉ chứa một thuộc tính.
- Không tồn tại phụ thuộc hàm thừa. Một phụ thuộc hàm là thừa khi:

  $F - \{X \to A\}$ tương đương với F

  $\Leftrightarrow (F - \{X \to A\})^+$ chứa $X \to A \quad \Leftrightarrow A \in X^+_{F - \{X \to A\}}$
- Không tồn tại các thuộc tính thừa ở vế trái. Một thuộc tính ở vế trái là thừa khi:

  Với $Z \subset X$, $(F - \{X \to A\}) \cup \{Z \to A\}$ tương đương với F

  $\Leftrightarrow$ Với $Z \subset X$, $\{Z \to A\} \in F^+$ hay $A \in Z^+$ thì tập thuộc tính (X - Z) là thừa.

Ví dụ:

Cho F = { $A \to AC$
$B \to ABC$
$D \to ABC$ }

Tìm phủ tối tiểu của F.

Giải:

$F \Leftrightarrow \{A \to A, A \to C, B \to A, B \to B, B \to C, D \to A, D \to B, D \to C\}$

$\Leftrightarrow \{A \to C, B \to A, B \to C, D \to A, D \to B, D \to C\}$

Nhận xét:

$B \to A$, $A \to C$ $\Rightarrow$ $B \to C$ (bắc cầu) ➔ bỏ $B \to C$

$D \to B$, $B \to A$ $\Rightarrow$ $D \to A$ (bắc cầu) ➔ bỏ $D \to A$

$D \to B$, $B \to A$ $\Rightarrow$ $D \to A$ (bắc cầu), $A \to C$ $\Rightarrow$ $D \to C$ (bắc cầu) ➔ bỏ $D \to C$

Ta có tập phụ thuộc hàm tối tiểu như sau:

F = { $A \to C$
$B \to A$
$D \to B$ }

IV.5.2 Định lý:

Mọi tập phụ thuộc hàm F đều có một tập phụ thuộc hàm tối tiểu G tương đương.

IV.5.3 Thuật toán tìm phụ thuộc hàm tối tiểu:

- **Bước 1:** Tách các thuộc tính ở vế phải của các PTH để cho vế phải chỉ còn chứa 1 thuộc tính.
  - *Giải thuật:*

```
H=∅;
For tất cả X → Y trong F
        For tất cả A trong Y
                H = H ∪ {X → A};
        End For;
End For;
```

- **Bước 2:** Loại bỏ phụ thuộc hàm thừa
  - *Giải thuật:*

```
For tất cả  X → A trong H
        J = H – {X → A};
        Xác định X+ trên J;
        If A ∈ X+ then H = H – {X → A};
End For;
```

- **Bước 3:** Loại bỏ thuộc tính thừa ở vế trái, đưa về tập PTH tối tiểu.
  - *Giải thuật:*

```
H0 =H
For tất cả X → A trong H
        For tất cả B ∈ X
                Y=X– {B};
                J=H – {X → A} ∪ {Y → A};
                Tính Y+J, Y+H;
                 If Y+J=Y+H then
                        Cập nhật lại X → A in H;
                        Đặt lại X=Y;
                End If;
        End For;
End for
If A→B trong H0 – H then   //  A, B là các thuộc tính
        Lặp lại bước 2
```

IV.5.4 Ví dụ

Tìm tập PTH tối tiểu của

F= {BCD→ A, BC→ EF, A→ F, F→ G, C→ D, A→ G}

Giải:

Bước 1: Tách các thuộc tính ở vế phải của các PTH để cho vế phải chỉ còn chứa 1 thuộc tính.

Ta có: BC → EF tách thành BC → E và BC → F

H= {BCD → A, BC → E, BC → F, A → F, F → G, C → D, A → G }

Bước 2: Loại bỏ phụ thuộc hàm thừa

Xét BCD → A: J = {BC → E, BC → F, A → F
F → G, C → D, A → G }

$(BCD)^+_J$ = BCDEFG không chứa A nên giữ lại
BCD → A

Xét BC → E: J = {BCD → A, BC → F, A → F
F → G, C → D, A → G }

$(BC)^+_J$ = BCFGD không chứa E nên giữ lại BC → E

Xét BC → F: J = {BCD → A, BC → E, A → F
F → G, C → D, A → G }

$(BC)^+_J$ = BCEDAFG chứa E nên ***xoá*** PTH BC → F trong tập H.

Xét A → F: J = {BCD → A, BC → E
F → G, C → D, A → G }

$(A)^+_J$ = AG không chứa F nên giữ lại A → F

Xét F → G: J = {BCD → A, BC → E, BC → F
A → F, C → D, A → G }

$(F)^+_J$ = F không chứa G nên giữ lại F → G

Xét C → D: J = {BCD → A, BC → E, BC → F
A → F, F → G, A → G }

$(C)^+_J$ = C không chứa D nên giữa lại C → D

Xét A → G: J = {BCD → A, BC → E, BC → F
A → F, F → G, C → D}

$(A)^+_J$ = AFG chứa G nên ***xoá*** A → G trong H.

Vậy tập H ở bước 2 là:

H= {BCD → A, BC → E, A → F, F → G, C → D}

Bước 3: Loại bỏ thuộc tính thừa ở vế trái

- Xét BCD → A:
  - Thử bỏ thuộc tính B:
    J= {CD → A, BC → E, A → F, F → G, C → D}

$(CD)^+_J$=CDAFG
$(CD)^+_H$=CD
} ***$CD^+_J \neq CD^+_H$: giữ lại B***

- o Thử bỏ thuộc tính C:

  J= {BD → A, BC → E, A → F, F → G, C → D}

  $(BD)^+_J$=BDAFG
  $(BD)^+_H$=BD
  } ***$BD^+_J \neq BD^+_H$: giữ lại C***

- o Thử bỏ thuộc tính D:

  J= {BC → A, BC → E, A → F, F → G, C → D}

  $(BC)^+_J$=BCAEFGD
  $(BC)^+_H$=BCEDAFG
  } ***$BC^+_J = BC^+_H$, xoá D***

▪ Xét BC → E:

H= {BC → A, BC → E, A → F, F → G, C → D}

- o Thử bỏ thuộc tính B:

  J= {BC → A, C → E, A → F, F → G, C → D}

  $(C)^+_J$=CED
  $(C)^+_H$=CD
  } ***C^+_J <> C^+_H, giữ lại B***

- o Thử bỏ thuộc tính C:

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

  $(B)^+_J$=BE
  $(B)^+_H$=B
  } ***$B^+_J \neq B^+_H$, giữ lại C***

Vậy tập PTH tối tiểu là:

H= {BC → A, BC → E, A → F, F → G, C → D}

IV.6. TẬP PHỤ THUỘC HÀM RÚT GỌN TỰ NHIÊN

IV.6.1 Định nghĩa

Cho tập phụ thuộc hàm F = $\{L_i \rightarrow R_i \mid L_i, R_i \in U, i=1..m\}$

F ở dạng rút gọn tự nhiên nếu:

- $L_i \cap R_i = \phi$, i=1..m.
- $L_i \neq L_j, \forall i \neq j$, i,j=1..m.

IV.6.2 Cách đưa về dạng rút gọn tự nhiên

- Loại bỏ thuộc tính ở vế phải nếu như thuộc tính đó có mặt ở cả 2 vế trên cùng 1 PTH

  VD: A,B,C → B,D
  => A,B,C → D

- Nếu $L_i \rightarrow R_i$ và $L_j \rightarrow R_j$ trong đó $L_i = L_j$, thì ta sẽ thay 2 PTH này bằng 1 PTH mới là: $L_i \rightarrow R_i \cup R_j$

  VD: A,B → C,D
  A,B → B,C
  } => A,B → C,D

IV.6.3 Ví dụ

Tìm tập rút gọn tự nhiên của F = {

AB → BC
B → D
CD → E
BE → GA
BE → DC }

Nhận xét:

- Phụ thuộc hàm AB→BC có thuộc tính B trùng nhau ở vế phải và vế trái, vì vậy loại bỏ thuộc tính này ở vế phải.
- Hai phụ thuộc hàm BE → GA và BE → DC có vế trái trùng nhau, vì vậy ta thay 2 PTH này bằng PTH BE→ADCG. Ta có kết quả như sau

Tập PHT rút gọn tự nhiên là F={ AB → C
B → D
CD → E
BE → ADCG}

CHƯƠNG V - CHUẨN HÓA LƯỢC ĐỒ CSDL QUAN HỆ

V.1. KHÓA- SIÊU KHÓA

V.1.1 Khái niệm:

Cho lược đồ quan hệ α = <U,F>, K $\subseteq$ U

- K được gọi là siêu khóa của α nếu: K^+ = U (hay K $\rightarrow$ U)
- K được gọi là khóa của α nếu:
  + K là siêu khóa
  + K là siêu khóa nhỏ nhất
    tức là $\forall$ X $\subset$ K, X không là siêu khóa, hay X $\not\rightarrow$ U

Ví dụ: Dữ liệu về các cầu thủ bóng đá gồm các thuộc tính:

U = { TENCT, MAU_AO, SO_AO, DOI, TINH, HLV, DIEM, Km }

F = { TINH $\rightarrow$ Km
MAU_AO $\rightarrow$ TINH, DOI, HLV
DOI, TINH $\rightarrow$ MAU_AO, HLV
MAU_AO, SO_AO $\rightarrow$ U }

Có siêu khóa K: K={DOI, TINH, SO_AO, MAU_AO }

Và có khóa K_1, K_2 : K_1= { MAU_AO, SO_AO}
K_2= {DOI, TINH, SO_AO}

Nhận xét:

- Một lược đồ quan hệ có thể có một hoặc nhiều siêu khóa, và một hoặc nhiều khóa. Các khóa có thể có số lượng thuộc tính khác nhau.
- Hai khóa phân biệt không thể bao nhau, tức là:
  nếu $K_1 \neq K_2$, thì $K_1 \not\subset K_2$ và $K_2 \not\subset K_1$.

V.1.2 . Giải thuật tìm khóa đơn giản

Thuật toán tìm khóa của lược đồ quan hệ α=<U,F>

```
K=U
    For (each attribute A in U) do
      If (K-A)+ =U then
          K=K-A
      Endif
    Endfor
Return K
```

Theo giải thuật trên ta chỉ tìm được duy nhất một khóa. Muốn tìm tất cả các khóa, chỉ cần hoán vị các thuộc tính trong U. Nhưng lúc này giải thuật chỉ chạy được với tập phụ thuộc hàm có số thuộc tính giới hạn. Nguyên nhân là do phải hoán vị **n! lần** tương ứng với n thuộc tính trong U.

V.1.3 Giải thuật tìm tất cả các khóa

V.1.3.1 Phép dịch chuyển lược đồ quan hệ

Giả sử cho lược đồ quan hệ α = <U, F>, X $\subseteq$ U. Phép dịch chuyển lược đồ quan hệ trên X cho ta một lược đồ quan hệ α' = α\X = <U', F'>, trong đó:

U' = U \ X

F' = { L_i \ X $\rightarrow$ R_i \ X, i= 1..m}

Nếu Ri \ X = ϕ thì bỏ đi phụ thuộc hàm đó.

V.1.3.2 Định lý cơ bản

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

Nếu X$\cap$Y =ϕ thì $(XY)^+_\alpha = X\ (Y)^+_{\alpha \setminus X}$

Ví dụ: Giả sử tìm $(AGHI)^+$ trên lược đồ quan hệ α gồm:

U = { ABCDEGHIJ }

F = { AG $\rightarrow$ BC BCE $\rightarrow$ G C $\rightarrow$ DAB }

AD $\rightarrow$ EC IB $\rightarrow$ DJ

Ta có thể tìm $AGH\ (I)^+_{\alpha \setminus \{AGH\}}$

$\alpha' = \alpha \setminus \{AGH\} = <U', F'>$, U' = BCDEIJ

F' = { $\phi \rightarrow$ BC BCE $\rightarrow \phi$

D $\rightarrow$ EC IB $\rightarrow$ DJ }

C $\rightarrow$ DB

$I^+_{\alpha'}$ = IBCDJE

$(AGHI)^+_\alpha = AGH\ (I)^+_{\alpha'}$ = AGHIBCDJE = ABCDEGHIJ

<u>Hệ quả</u>: $X^+_\alpha = X\ (\phi)^+_{\alpha \setminus X}$

V.1.3.3 Định lý

Gọi Kα là tập hợp tất cả các khóa của lược đồ quan hệ α, và:

- $U_0 = \cap$ K α : là tập các thuộc tính nằm trong mọi khóa.
- M = $\cup$ K α : là tập các thuộc tính khóa hay tập thuộc tính nguyên thủy.
- I = U\M: là tập các thuộc tính không nằm trong mọi khóa.

Khi đó: với X $\subseteq$ U thì:

$K_\alpha = X \oplus K_\alpha \setminus X$ <=> X $\subseteq$ U_0

$K_\alpha = K_\alpha \setminus X$ <=> X $\subseteq$ I

Chú ý: phép tóan $\oplus$:

$X \oplus \{ Y_1, Y_2, ... Y_n\} = \{XY_1, XY_2, ... XY_n\}$

V.1.3.4 Bổ đề

Giả sử cho lược đồ quan hệ α = <U, F>, F = {$L_i \rightarrow R_i$ | L_i, $R_i \in$ U, i=1..m} là tập phụ thuộc hàm ở dạng rút gọn tự nhiên. Có hai khẳng định quan trọng sau đây:

U $\setminus \cup R_i \subseteq U_0$ – Các thuộc tính không nằm trong bất kỳ vế phải của phụ thuộc hàm nào sẽ nằm trong mọi khóa.

$\cup R_i \setminus \cup L_i \subseteq$ I – Các thuộc tính được các thuộc tính khác dẫn xuất tới sẽ không nằm trong bất kỳ khóa nào.

Ý nghĩa của bổ đề là thay vì tìm tập tất cả các khóa của lược đồ quan hệ α đã cho chúng ta tìm tất cả các khóa của lược đồ quan hệ $\alpha \setminus U_0 I$ đơn giản hơn (có thể ít thuộc tính hơn và tập phụ thuộc hàm đơn giản hơn).

V.1.3.5 Giải thuật tìm K_α

- Bước1: Đưa F về dạng rút gọn tự nhiên

- Bước 2: Tính $U_0 = U - \bigcup_{i=1}^{m} R_i$

- Bước 3: Tính $I = \bigcup_{i=1}^{m} R_i - \bigcup_{i=1}^{m} L_i$

- Bước 4: Xác định $\alpha' = \alpha \setminus U_0 I$

- Bước 5: Tìm $K_{\alpha'}$

- Bước 6: $K_\alpha = U_0 \oplus K_{\alpha'}$

Ví dụ:

Cho α= <U, F> với U = ABCDEGHIJL

F = { AI $\rightarrow$ ECD
ABJ $\rightarrow$ GH
CDL $\rightarrow$ ABEH }

- F đã ở dạng rút gọn tự nhiên.
- $U_0 = U - \cup R_i$ = IJL
- I = ABCDEGHIJL - ABCDIJL = EGH
- $\alpha' = \alpha \setminus U_0 I$ = <U', F'>
  - U' = U - U_0I = ABCD
  - F' = { A $\rightarrow$ CD
    CD $\rightarrow$ AB }

  Vì A^+ = ABCD = U', $(CD)^+$ = ABCD =>
- $K_{\alpha'}$ = { A, CD }
- $K_\alpha = U_0 \oplus K_{\alpha'}$ = IJL $\oplus$ {A, CD } = {AIJL, CDIJL}

V.2. CÁC DẠNG PHỤ THUỘC HÀM

V.2.1 Phụ thuộc từng phần

Phụ thuộc hàm X → A được gọi là phụ thuộc từng phần nếu X là tập con thật sự của một khóa của quan hệ R.

Ví dụ: Trong quan hệ R(U),

U = SAIP, F = {S→ A, SI → P} => khóa là SI.

A: không là thuộc tính nguyên tố.

S → A: là phụ thuộc hàm từng phần (do S $\subset$ SI)

V.2.2 Phụ thuộc hàm đầy đủ/ phụ thuộc hàm sơ cấp

Phụ thuộc hàm X → A được gọi là phụ thuộc đầy đủ nếu **không tồn tại** tập con thật sự Y nào của X (Y $\subset$ X) để Y → A xảy ra. Như vậy, phụ thuộc đầy đủ vào khóa ngược lại với phụ thuộc từng phần.

V.2.3 Phụ thuộc truyền

Phụ thuộc hàm X → A được gọi là phụ thuộc truyền nếu X **không là tập con** thật sự của bất kỳ khóa nào.

Ví dụ: U= SIDM, F = {SI → D, SD → M}, khóa là SI

M không là thuộc tính nguyên tố.

SD $\not\subset$ SI => SD → M là phụ thuộc hàm truyền.

(Hay tồn tại sơ đồ: SI → SD → M, nên SD → M là phụ thuộc hàm truyền)

V.2.4 Phụ thuộc trực tiếp

Phụ thuộc hàm X → A được gọi là phụ thuộc trực tiếp nếu không tồn tại tập thuộc tính Y, với Y $\neq$ X, Y $\neq$ A thỏa: X → Y và Y → A

V.3. PHÉP TÁCH CÁC SƠ ĐỒ QUAN HỆ

V.3.1 Phép tách một sơ đồ quan hệ

Định nghĩa

Phép tách một sơ đồ quan hệ R ={A_1, A_2, ..., A_n } là thay thế R bằng một tập hợp P = {R_1, R_2, ..., R_k} các tập con của R sao cho

U = $U_1 \cup U_2 \cup ... \cup U_k$, với các R_i xác định trên U_i.

Ví dụ:

R(A,B,C,D) thì P= (AB, BC, CD) là một phép tách trên R.

V.3.2 Phép tách với kết nối không mất thông tin

Nếu R là một sơ đồ quan hệ được tách thành các sơ đồ R_1, R_2, ..., R_k và F là một tập các phụ thuộc hàm, ta nói phép tách có kết nối không mất thông tin nếu với mọi quan hệ r của R thỏa F sao cho:

$$r = r[R_1] * r[R_2] * ... r[R_k]$$

V.3.2.1 Kiểm tra một phép tách có phải là phép tách có kết nối không mất thông tin

Giải thuật:

Input:

Một sơ đồ quan hệ R(A_1, A_2, ..., A_n), một tập phụ thuộc hàm F, và một phép tách P =(R_1, R_2, ..., R_k) của R.

Output:

Kết luận P có phải là phép tách có kết nối không mất thông tin hay không?

Phương pháp:

- Xây dựng một bảng gồm: k dòng (tương ứng với k sơ đồ con)
  j cột (tương ứng với j thuộc tính).

  Tại dòng i cột j ta ghi a_j nếu $A_j \in R_i$, b_{ij} nếu $A_j \notin R_i$.
- Lặp lại việc xét từng phụ thuộc hàm X → Y trong F, cho đến khi không còn thay đổi được bảng.

  Mỗi lần xét, ta tìm các dòng mang trị bằng nhau trên tập thuộc tính X. Nếu tìm thấy hai dòng như vậy, làm cho các ký hiệu trên hai dòng đó giống nhau tại các thuộc tính của Y
- Nếu sau khi sửa bảng, ta thấy có dòng nào đó có dạng a1, a2,..., ak, thì kết nối là không mất thông tin. Ngược lại là không.

Ví du:

Xét phép tách SAIP thành SA và SIP. Các phụ thuộc hàm gồm S → A và SI → P, và bảng ban đầu là:

	S	A	I	P
SA	a_1	a_2	b_{13}	b_{14}
SIP	a_1	b_{22}	a_3	a_4

Do S → A nên bảng trở thành:

	S	A	I	P
SA	a_1	a_2	b_{13}	b_{14}
SIP	a_1	a_2	a_3	a_4

Vì bảng có một dòng chứa toàn a_j, kết nối là không mất thông tin.

V.3.2.2 Định lý

Nếu P=(R_1, R_2) là một phép tách sơ đồ quan hệ R, và F là một tập phụ thuộc hàm, thì P là phép tách có một kết nối không mất thông tin thỏa F nếu và chỉ nếu:

$$(R1 \cap R2) \rightarrow (R1 - R2) \quad \text{hoặc } (R1 \cap R2) \rightarrow (R2 - R1)$$

Chú ý:

- Các phụ thuộc hàm này không cần thuộc F, chỉ cần thuộc F^+.
- Định lý này chỉ áp dụng cho các phép tách R thành 2 sơ đồ quan hệ.

Ví du:

R= ABC và F = {A → B}.

- Phép tách P = (AB,AC) là phép tách không mất thông tin vì:
  - $AB \cap AC = A$
  - $AB - AC = B$
  - Và ta có $A \rightarrow B \in F^+$
- Còn phép tách P = (AB, BC) không phải là phép tách không mất thông tin vì:
  - $AB \cap BC = B$
  - $AB - BC = A$
  - Nhưng ta không có phụ thuộc hàm: $B \rightarrow A \notin F^+$.

V.4. CÁC DẠNG CHUẨN CỦA LĐQH VÀ GIẢI THUẬT CHUẨN HÓA

V.4.1 Giới thiệu

Khi thiết kế một CSDL quan hệ, ta thường phải chọn một trong nhiều sơ đồ quan hệ. Một vài lựa chọn tốt hơn những cái khác vì nhiều lý do khác nhau. Một CSDL được thiết kế không tốt sẽ dẫn đến nhiều vấn đề rắc rối nảy sinh khi sử dụng. Vì thế, vấn đề đặt ra là làm sao thiết kế được một CSDL tốt.

Để giải quyết các vấn đề này, người ta đưa ra các mức chuẩn cho một CSDL quan hệ. Nếu một CSDL quan hệ không ở dạng chuẩn thì sẽ gây khó khăn cho việc cập nhật dữ liệu. Sơ đồ các mức chuẩn hóa có thể được biểu diễn như sau:

V.4.2 Dạng chuẩn thứ nhất (The First Normal Form)

V.4.2.1 Định nghĩa

Một sơ đồ quan hệ được xem là ở dạng chuẩn thứ nhất nếu mọi thuộc tính của R đều khác trống, phụ thuộc hàm vào khóa và không được mạng giá trị kép.

Ví dụ 1:

CUNG_UNG (MA_NSX, MA_HANG, SO_LG, VON_NSX, THANH_PHO, NUOC)

Với các phụ thuộc hàm:

a) MA_NSX, MA_HANG → SO_LG
b) MA_NSX → VON_NSX
c) MA_NSX → TH_PHO
d) MA_NSX → NUOC
e) TH_PHO → NUOC

⇨ Quan hệ CUNG_UNG thỏa dạng chuẩn thứ nhất.

Ví dụ 2:

Cho quan hệ GIAOVIEN như sau:

MAGV	HOTEN	NSINH	KHOA
CNTT001	N.T.N	12/01/76	CNTT
CNTT002	T.T.G	26/3/76	CNTT
QTKD001	T.L.H	10/10/70	QTKD
QTKD002	L.H.P.T	01/02/60	QTKD

Nhận xét: Quan hệ này có thuộc tính MAGV mang giá trị kép nên vi phạm dạng chuẩn thứ nhất.

Ví dụ 3:

Cho quan hệ HOSO như sau:

MAHS	HOTEN	NSINH	KINH	HOA	KHMER	KHAC
001	N.T.N	12/01/76	x			
002	T.T.G	26/3/76	x			
003	T.L.H	10/10/70	x			
004	L.H.P.T	01/02/60			x	
005	L.T.H	01/01/85				x

Nếu ở thời điểm hiện tại, chưa có hồ sơ nào thuộc dân tộc «Hoa», thì quan hệ này vi phạm dạng chuẩn 1.

V.4.2.2 Cách đưa về dạng 1NF

Chia các thuộc tính có thể phân chia thành các thuộc tính thành phần cho đến khi không thể phân chia được nữa.

Ví dụ:

Quan hệ GIAOVIEN trong ví dụ trên có thể tách thành hai quan hệ như sau:

MAGV	HOTEN	NSINH
001	N.T.N	12/01/76
002	T.T.G	26/3/76
003	T.L.H	10/10/70
004	L.H.P.T	01/02/60

MAGV	KHOA
001	CNTT
002	CNTT
003	QTKD
004	QTKD

V.4.3 Dạng chuẩn thứ hai (The Second Normal Form)

V.4.3.1 Định nghĩa

Một sơ đồ quan hệ được xem là ở dạng chuẩn thứ hai nếu nó thỏa dạng chuẩn thứ 1 và không có phụ thuộc hàm từng phần nào.

Nhân xét:

- Để biết một quan hệ R có thỏa dạng chuẩn 2 hay không, ta phải xác định khóa của R. Từ đó, mới có thể xác định các phụ thuộc hàm nào là phụ thuộc hàm từng phần.
- Chỉ có các quan hệ có khóa gồm hai thuộc tính trở lên thì mới có thể không thỏa dạng chuẩn 2.

Ví dụ:

Quan hệ CUNG_ UNG trên có khóa là {MA_NSX, MA_HANG}. Nên quan hệ này không thỏa dạng chuẩn 2 vì có phụ thuộc hàm từng phần:

$$MA\_NSX \rightarrow VON\_NSX$$

V.4.3.2 Nhận xét

Một quan hệ không ở dạng chuẩn 2 sẽ gây khó khăn khi cập nhật dữ liệu.

Giả sử:

- R(U) là quan hệ không ở dạng chuẩn 2
- K : khóa của quan hệ R, $\exists X \in K : X \rightarrow A$, $A \subseteq N$ (N: tập thuộc tính phi nguyên thủy)

Ta muốn thêm bộ t vào quan hệ này, nếu ta chỉ kiểm tra $t.k \neq u.k$ ($\forall u \in R$) thì ta chưa được phép xen t vào R vì có thể $\exists u \in R$ mà $u.x=t.x$ nhưng $u.A \neq t.A$ nhưng vẫn bảo đảm $u.k \neq t.k$

Ví dụ, cho quan hệ SV_DT(MASV,MADT,TEN_DT,KQ) như sau:

MASV	MADT	TEN_DT	KQ
001	1	CNPM	8
002	2	**PTHT**	7

Giả sử ta muốn thêm bộ t=<003,2,**CTD**,6> vào quan SV_DT, rõ ràng ta có khóa của bộ này không trùng với khóa của mọi bộ của quan hệ đã cho, nhưng ta không thể thêm bộ này vào quan hệ vì

- Bộ t có MADT=2,
- Trong quan hệ SV_DT cũng $\exists$ một bộ có MADT=2, nhưng TEN_DT của hai bộ này là khác nhau.

Vì vậy phải đưa quan hệ về dạng chuẩn 2.

V.4.3.3 Cách đưa về dạng 2NF

Nếu R(U) : $\neg$ 2NF

$K \not\rightarrow X$ $(X \subset K, X \cap A=\varnothing)$

$\downarrow$

$A \subseteq N$ (N: tập thuộc tính phi nguyên thủy)

Tách quan hệ R(U) = R[**X**A] * R[U-A], với X là khóa.

- Xét quan hệ R[U-A] nếu thỏa 2NF thì dừng.
- Ngược lại, lặp lại các bước trên cho R[U-A]

Ví dụ:

Cho quan hệ SV_DT(MASV,MADT,TEN_DT,KQ) có khóa là {MASV,MADT},

vì vậy PTH MADT→ TEN_DT là PTH từng phần. Tách quan hệ này thành hai quan hệ

R_1= SV_DT[MADT, TEN_DT], khóa là MADT

R_2=SV_DT[MASV, MADT, KQ], khóa là {MASV, MADT}

Nhận xét: R_1 và R_2 đều thỏa 2NF

V.4.4 Dạng chuẩn thứ ba (The Third Normal Form)

V.4.4.1 Nhận xét

Cho quan hệ SINHVIEN (MASV, TEN, QUE, KM).

Quan hệ này ở dạng chuẩn 2 nhưng vẫn gây khó khăn khi cập nhật dữ liệu. Cụ thể, muốn thêm một bộ t vào quan hệ này ta chỉ kiểm tra trên trường khóa là chưa đủ, mà còn phải kiểm tra phụ thuộc hàm QUE→KM. Nếu không kiểm tra PTH này, có thể dẫn đến sai dữ liệu như sau:

MASV	TEN	QUE	KM
001	BAO	VL	30
002	THU	VL	40

Vì vậy cần phải đưa quan hệ này về dạng chuẩn cao hơn, đó là dạng chuẩn 3.

V.4.4.2 Định nghĩa

Một sơ đồ quan hệ được xem là ở dạng chuẩn thứ ba nếu nó thỏa dạng chuẩn thứ 2 và không có phụ thuộc hàm truyền nào.

Ví dụ:

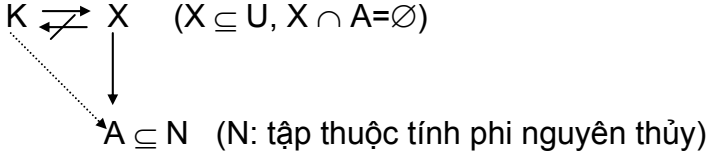
Quan hệ CUNG_UNG ở trên là không thỏa dạng chuẩn 3 vì có phụ thuộc hàm truyền:

MA_NSX → TH_PHO

TH_PHO → NUOC

V.4.4.3 Cách đưa về dạng 3NF

Nếu R: ¬3NF



Tách quan hệ R(U) = R[**X**A] * R[U-A], với X là khóa.

- Xét quan hệ R[U-A] nếu thỏa 3NF thì dừng.
- Ngược lại, lặp lại các bước trên cho R[U-A]

Ví dụ:

Cho quan hệ SINHVIEN(MASV,TEN,QUE,KM). Quan hệ này có khóa là MASV, vì vậy PTH QUE→ KM là PTH truyền. Tách quan hệ này thành hai quan hệ:

R_1= SINHVIEN[QUE, KM], khóa là QUE

R_2= SINHVIEN[MASV,TEN,QUE], khóa là MASV

Nhận xét: R_1 và R_2 đều thỏa 3NF

V.4.5 Dạng chuẩn BCNF(Boyce Codd Normal Form)

V.4.5.1 Định nghĩa

Một sơ đồ quan hệ được xem là ở dạng chuẩn BCNF nếu nó thỏa dạng chuẩn thứ 3 và không tồn tại bất cứ phụ thuộc hàm nào mà vế trái không phải là siêu khóa.

Ví dụ 1:

R = (CSZ), F = { CS → Z, Z → C}, khóa là CS, CZ.

Có phụ thuộc hàm Z → C với Z không là siêu khóa

⇨ R(CSZ) không thỏa BCNF.

Ví dụ 2:

R(MSK), F = {MS → K} khóa MS

MSK thỏa BCNF vì MS là siêu khóa.

V.4.5.2 Giải thuật đưa về dạng chuẩn BCNF bằng phép tách có kết nối không mất thông tin

Input:

Sơ đồ quan hệ R và tập phụ thuộc hàm F. (Với giả thiết F ở dạng tối tiểu và rút gọn tự nhiên).

Output:

Một phép tách R với kết nối không mất thông tin, sao cho mọi sơ đồ quan hệ trong phép tách sẽ ở dạng chuẩn BCNF và thỏa chiếu của F trên đó.

Phương pháp:

Ta lặp đi lặp lại việc xây dựng một phép tách P cho R, và phép tách đó luôn có một kết nối không mất thông tin thỏa F.

- Khởi đầu, ta cho P = (R)
- *Lặp lại* trong khi P còn chứa sơ đồ quan hệ không thỏa dạng chuẩn BCNF.

Gọi S là một sơ đồ quan hệ trong P, và S không ở dạng chuẩn BCNF.

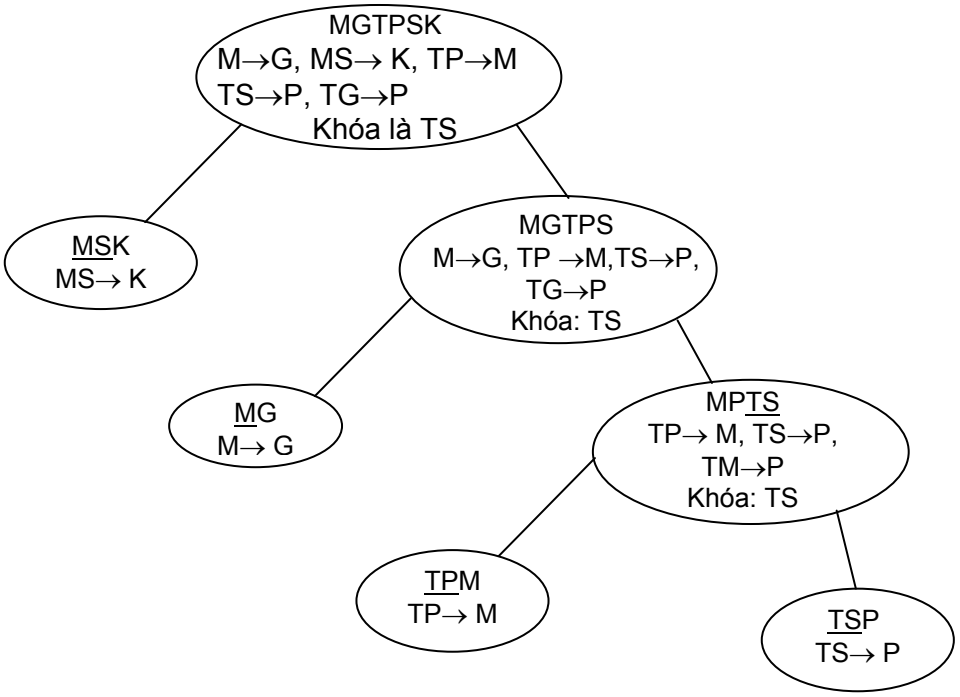
Xét X → A, với X không là siêu khóa của S, và A ∉ X,

⇨ Tách S trong P thành S1 và S2, trong đó:

- S1 chứa A và các thuộc tính của X.
- S2 chứa tất cả các thuộc tính của S trừ đi A.

Hết lặp.

Ví dụ 1:



Lưu ý:

- Phép tách sẽ khác đi nếu ta thay đổi phụ thuộc hàm được chọn để tách.
- Phép tách này có thể là phép tách không bảo tồn phụ thuộc hàm.

Ví dụ 2:

Cho lược đồ quan hệ R(U) và tập phụ thuộc hàm F,

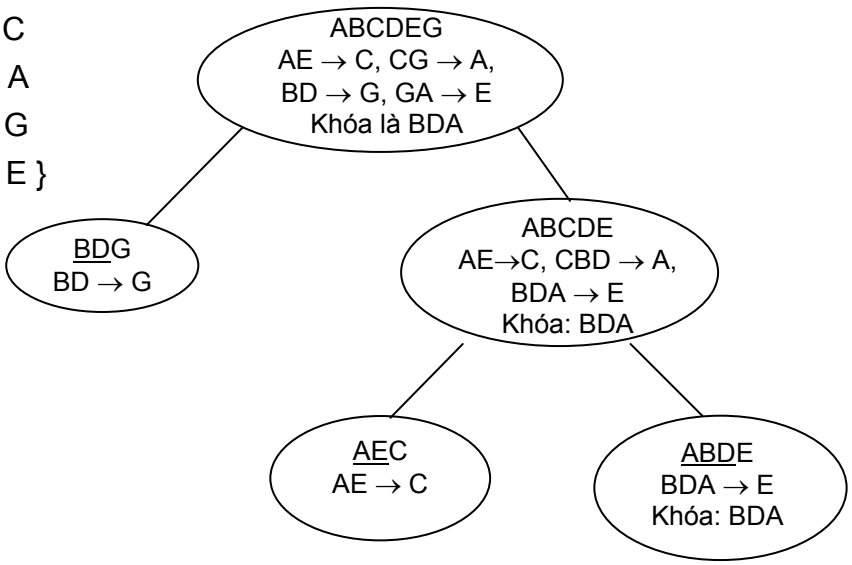
U = ABCDEG

F = { $AE \rightarrow C$

$CG \rightarrow A$

$BD \rightarrow G$

$GA \rightarrow E$ }



Ví dụ 3:

Cho lược đồ quan hệ R(CTHRSG) và tập phụ thuộc hàm F.

U = CTHRSG, trong đó:

C : Giáo trình ; T : Thầy ; H : giờ ; R : Phòng học ; S : Sinh viên ; G : Lớp

F = { $C \rightarrow T$: Mỗi giáo trình có một Thầy dạy.

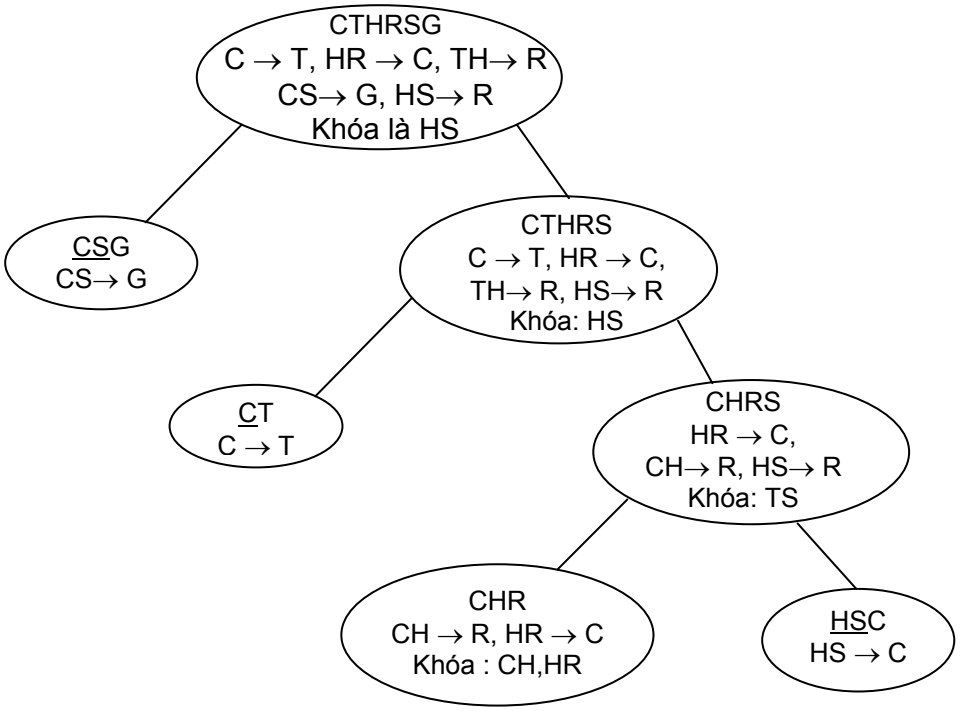
$HR \rightarrow C$: Chỉ một môn học (giáo trình) ở một phòng học tại một thời điểm.

$HT \rightarrow R$: Tại mỗi thời điểm, mỗi Thầy chỉ có thể dạy ở một phòng.

$CS \rightarrow G$: Mỗi sinh viên theo học mỗi giáo trình chỉ ở một lớp.

$HS \rightarrow R$: Mỗi sinh viên chỉ có thể ở một phòng học tại mỗi thời điểm.

}



TÀI LIỆU THAM KHẢO

1. [O'neil 1994] Patrick O'neil, *Database - principles, programming, performance*, Morgan Kaufmann Inc,1994.
2. [Silberschatz et al.1996], Abraham Silberschatz, Henry F.Korth, S.Sudarshan, *Database system concepts,* McGRAW-HILL Inc, 1996
3. [Huy] TS. Nguyễn Xuân Huy, Giáo trình Cơ sở dữ liệu
4. [Lộc 1999] Phạm Thị Xuân Lộc, Bài giảng Cơ sở dữ liệu, 1999

MỤC LỤC