TS. NGUYỄN BÁ TƯỜNG

NHẬP MÔN CƠ SỞ DỮ LIỆU PHÂN TÁN

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT

TS. NGUYỄN BÁ TƯỜNG

Nhập môn
CƠ SỞ DỮ LIỆU PHÂN TÁN

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
HÀ NỘI

LỜI NÓI ĐẦU

Ngày nay công nghệ thông tin (CNTT) đã vượt ra ngoài khuôn khổ một đối tượng riêng của khoa học công nghệ. CNTT trở thành một nhân tố quan trọng trong sản xuất và phát triển kinh tế toàn xã hội với phạm vi toàn cầu. Trong nền kinh tế tri thức CNTT đóng vai trò then chốt. Mạng máy tính trở thành công cụ đắc lực không thể thiếu cho bất kỳ một tổ chức xã hội nào. *Cơ sở dữ liệu phân tán* (CSDLPT) nói riêng và các *hệ phân tán* nói chung là một lĩnh vực được nghiên cứu từ lâu, nhưng gần đây do sự phát triển nhanh chóng của công nghệ truyền tin và sự bành trướng mạnh mẽ của mạng Internet, cùng với xu thế toàn cầu hoá trong mọi lĩnh vực, đặc biệt trong lĩnh vực thương mại, CSDLPT đã trở thành một lĩnh vực thu hút nhiều sự quan tâm của các nhà nghiên cứu trong lĩnh vực CNTT. Vậy CSDLPT là gì ? Để trả lời được câu hỏi này các nhà nghiên cứu đã, đang từng bước tiếp cận và sẽ trả lời chính xác. Tuy nhiên, trong khuôn khổ của cuốn sách này *chúng tôi muốn trình bày các nguyên lý tổng quát nhất của CSDLPT đang được nhiều người quan tâm*. Về mặt trực quan, nghĩa đen của cụm từ CSDLPT chứa hai cụm từ là *cơ sở dữ liệu* (CSDL) và *phân tán* (PT). Như vậy có thể nói CSDLPT là sự hợp nhất của hai hướng tiếp cận nghiên cứu, đó là cơ sở dữ liệu và phân tán.

Khái niệm *phân tán* ở đây chúng ta phải hiểu là *phân tán thông tin* và *các thông tin đó được chứa trên các máy tính của một hệ thống máy tính có liên hệ với nhau được gọi là Mạng Máy Tính* (MMT). Một cách hình ảnh chúng ta có thể nói:

Cơ sở dữ liệu phân tán = Cơ sở dữ liệu + Mạng máy tính.

Như vậy, một CSDLPT là một tập hợp *nhiều CSDL có liên đới logic và được phân bố trên một mạng máy tính.* Trước tiên chúng ta cần nhấn mạnh lại là một hệ CSDLPT không phải là một tập hợp các tập tin rời rạc được lưu riêng rẽ tại mỗi nút của một mạng máy tính. Để có một hệ CSDLPT, các tập tin không chỉ có liên đới logic với nhau mà chúng còn phải có cấu trúc và được truy xuất qua một giao diện chung. Tuy nhiên, gần đây trong thực tiễn đang hình thành dần một hướng *phân tán dữ liệu bán cấu trúc (semi-structured data)*, được lưu trong các tập tin trên Internet kiểu như các trang Web. Tất nhiên, kiểu truy xuất đến dữ liệu bán cấu trúc như trang Web khác với truy xuất dữ liệu của một hệ CSDLPT. Có hai hướng nghiên cứu để tiếp cận CSDLPT đó là các mô hình CSDL, đặc biệt là mô hình CSDL quan hệ và MMT (mạng máy tính), các phương pháp phân tán dữ liệu trên MMT.

Một câu hỏi được đặt ra là, trong một hệ CSDLPT thì những gì được phân tán? Đó là:

(1) Thiết bị xử lý; (2) Chức năng; (3) Dữ liệu; (4) Quyền điều khiển.

Trong một MMT thì thiết bị xử lý ngầm định phân tán, vì các bộ phận của mạng được phân bố tại các vị trí địa lý khác nhau.

Một kiểu phân tán nữa đó là chức năng. Nhiều chức năng của hệ thống máy tính có thể được chuyển giao cho nhiều bộ phận khác nhau.

Kiểu phân tán thứ ba là phân tán dữ liệu. Dữ liệu được dùng bởi một số ứng dụng khác nhau có thể được phân bổ ở một số vị trí khác nhau.

Cuối cùng là phân tán *quyền điều khiển* (control). Quyền điều khiển một số công việc trong hệ thống được phân cấp, chia quyền theo chức năng.

Giáo trình có bảy chương. Chương 1 là một số khái niệm cơ bản của MMT. Chúng tôi coi CSDL quan hệ là một phần cốt yếu của CSDLPT nói riêng và CSDL nói chung nên trong chương 2 chúng tôi cố gắng nêu đầy đủ các khái niệm liên quan đến CSDL quan hệ như các định nghĩa quan hệ, các

phép toán trên quan hệ, khái niệm phụ thuộc hàm, khái niệm khoá, các dạng chuẩn, v.v. Trong chương 3 chúng tôi nêu các lệnh cơ bản và thiết yếu của SQL, nhằm giúp các bạn tiếp cận một cách nhẹ nhàng, đơn giản một ngôn ngữ quan hệ đặc trưng. Đồng thời SQL cũng giúp các bạn một số khái niệm và ứng dụng vấn tin trong các ứng dụng về mạng và các minh hoạ tiếp theo của giáo trình về các bài toán phân tán vấn tin ở các chương sau. Trong chương 4 chúng tôi sẽ trình bày các phương pháp phân mảnh ngang nguyên thuỷ, phân mảnh ngang dẫn xuất, phân mảnh dọc theo tụ lực của các thuộc tính(affinity of attributes), phân mảnh có nối không mất, bảo toàn phụ thuộc và phân mảnh thành BCNF, 3NF ... , Trong chương 4 chúng tôi cũng nêu các vấn đề liên quan đến khái niệm cấp phát dữ liệu như bài toán cấp phát, yêu cầu về thông tin cấp phát, mô hình và các giải pháp cấp phát, v.v. Chương 5 là chương dành cho các vấn đề cơ bản về xử lý vấn tin và kiểm soát dữ liệu ngữ nghĩa như quản lý khung nhìn, an toàn dữ liệu, kiểm soát tính toàn vẹn dữ liệu, v.v. Trong chương 6 chúng tôi nêu một số khái niệm về quản lý giao dịch, các loại khoá chốt, lịch biểu tuần tự, khả tuần tự và các thuật toán kiểm tra một lịch biểu khả tuần tự hay không. Đồng thời trong chương này các bạn được làm quen với các khái niệm điều khiển đồng thời, các cơ chế điều khiển đồng thời, các thuật toán điều khiển đồng thời, các hệ cơ sở dữ liệu song song v.v . Cuối cùng trong chương 7 chúng ta sẽ nghiên cứu một số khái niệm của *cơ sở dữ liệu hướng đối tượng phân tán*. Chương này chúng ta sẽ xét các khái niệm như mô hình hướng đối tượng, thiết kế phân tán đối t-ượng, quản lý đối tượng, v.v. Sau mỗi một chương, nếu có thể chúng tôi có một số bài tập, nhằm giúp các bạn củng cố những kiến thức lý thuyết của mình.

Mặc dù chúng tôi đã rất cố gắng nhưng không thể tránh khỏi những thiếu sót về cách diễn đạt, sự sắp xếp bố cục nội dung và các lỗi cú pháp,

văn phong. Rất mong được bạn đọc góp ý cho chúng tôi. Cuối cùng, chúng tôi xin chân thành cảm ơn Phòng Sau đại học, Phòng Đào tạo Học viện Kỹ thuật Quân sự, PGS.,TS. Phạm Ngọc Phúc - Trưởng phòng Sau đại học, PGS.,TS. Nguyễn Văn Xuất - Chủ nhiệm Khoa Công nghệ Thông tin, TS. Đào Thanh Tĩnh, PGS.,TS. Phạm Văn Ất - Chủ nhiệm Khoa Công nghệ Thông tin, Trường Đại học giao thông Vận tải Hà Nội, PGS.,TS. Đoàn Văn Ban viện Công nghệ Thông tin, TS. Dương Tử Cường, ThS. Nguyễn Văn Thàng cùng các bạn đồng nghiệp trong khoa CNTT - Học viện Kỹ thuật Quân sự. Đặc biệt tác giả xin chân thành cảm ơn Ban biên tập Nhà xuất bản Khoa học và Kỹ thuật, đã có đóng góp và giúp đỡ xác đáng để cuốn sách sớm được ra mắt bạn đọc.

TÁC GIẢ

KÝ PHÁP CHO CÁC QUAN HỆ

Chúng tôi nêu một số ký hiệu và quy ước được sử dụng trong cuốn sách này để mô tả các quan hệ, các lược đồ quan hệ và các bộ của quan hệ:

1. Tên của các lược đồ quan hệ nói chung được biểu thị bằng chữ cái viết hoa, chẳng hạn R, S.
2. Một quan hệ thường được biểu thị bằng chữ cái viết thường tương ứng, chẳng hạn là r, s.
3. Ta dùng các chữ hoa ở đầu bộ chữ cái như A, B, C,...biểu thị các thuộc tính. Các chữ hoa ở cuối bảng chữ cái như X, Y, Z, W, V,...biểu thị các tập thuộc tính.
4. Ta cũng sử dụng *lược đồ quan hệ* (tập các thuộc tính) làm tên của quan hệ: thí dụ như *ABC* là tên của một quan hệ có các thuộc tính *A*, *B* và *C*. Chúng ta cũng xử lý tên quan hệ giống như chúng là các lược đồ: chẳng hạn $R \cap S$ biểu thị cho tập các thuộc tính chung của *R* và *S*, còn $r \cap s$ là tập các bộ chung của của các quan hệ hiện hành của các lược đồ *R* và *S*. Tuy nhiên từng ngữ cảnh cụ thể sẽ giúp phân hai cách sử dụng phép giao, vì thế chúng ta có thể sử dụng $R \cap S$ để nói đến phép giao của các bộ nếu không có gì lầm lẫn.
5. Ta sẽ dùng bảng để biểu thị một quan hệ r, thí dụ:

r

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	d_2	e_2
...				
a_k	b_k	c_k	d_k	e_k

6. Tập các thuộc tính thường được biểu thị bằng *ký hiệu kết chuỗi* (concatenation) chứ không phải ký hiệu tập hợp thông thường. Chẳng hạn *ABC* thay cho tập các thuộc tính {*A*, *B*, *C*} và *XY* thay cho X ∪ Y nếu X và Y là các tập thuộc tính, XA hoặc AX thay cho X ∪ A, với X là tập thuộc tính, A là một thuộc tính.

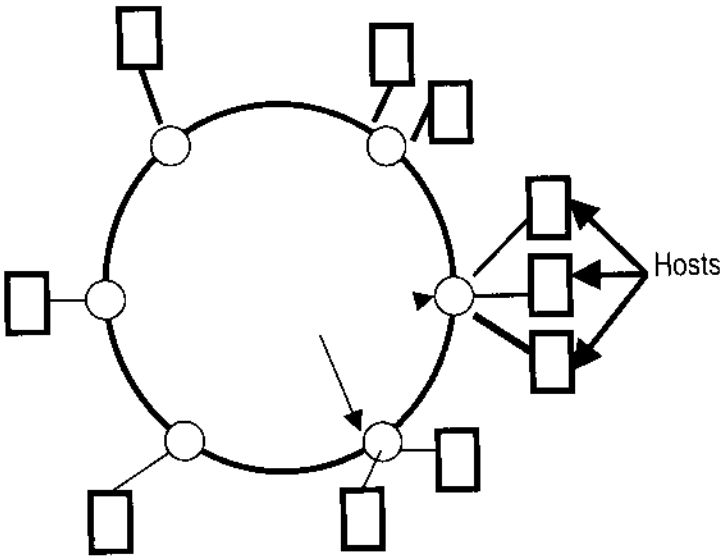
7. Các bộ được đặt tên theo chữ cái Hy Lạp, chẳng hạn μ, α hoặc chữ cái thường t, t' hay t_1, t_2, v.v. Ta thường dùng k để chỉ một khoá của sơ đồ quan hệ và K chỉ họ khoá của sơ đồ quan hệ. Các thành phần của bộ t ứng với tập thuộc tính *X* được biểu thị là t[*X*] hoặc t.X; t(E) chỉ bộ t thoả mệnh đề E

8. Các toán tử đại số quan hệ: +(phép hợp), - (phép lấy hiệu), *(phép giao), σ (phép chọn, ta cũng dùng r(E) để chỉ những bộ của quan hệ r thoả mệnh đề E), π (phép chiếu, ta cũng dùng r.X để chỉ quan hệ r chiếu lên X) và × (tích Descartes), |><| (phép nối tự nhiên), |>< (phép nối nửa).v.v.

CHƯƠNG 1

MẠNG MÁY TÍNH

Trong chương này chúng ta sẽ thảo luận một số vấn đề liên quan đến mạng máy tính, tập trung vào các khái niệm và các vấn đề quan trọng đối với các hệ CSDL phân tán. Vì thế chúng ta sẽ bỏ qua hầu hết các chi tiết về công nghệ và kỹ thuật trong các phần trình bày này.



Hình 1.1 Mạng máy tính.

Chúng ta định nghĩa một mạng máy tính (computer network) là một tập các máy tính tự vận hành, được nối kết lại và có khả năng trao đổi dữ liệu và thông tin giữa chúng (hình 1.1). Có hai ý chính trong định nghĩa này là "*được nối kết lại*" và "*tự vận hành*".

Chúng ta muốn các máy tính *tự vận hành* để mỗi máy có thể cho các chương trình chạy trên chúng và thực hiện được các công việc riêng của chúng. Chúng ta cũng muốn các máy tính *được kết nối lại* để có thể trao đổi thông tin cho nhau. Các máy tính trên một mạng thường được gọi là nút (node), host hoặc trạm (site). Chúng tạo ra các thành phần phần cứng cơ bản của một mạng. Những thành phần cơ bản khác là đường truyền dùng để nối kết các nút. Lưu ý rằng đôi khi thuật ngữ host và node có thể sử dụng để nói đến một thiết bị đơn thuần, còn site được dành để nói đến các thiết bị và các phần mềm chạy trên đó.

1.1. KHÁI NIỆM VỀ TRUYỀN DỮ LIỆU

Trước tiên.chúng ta hãy đưa ra một số định nghĩa cơ bản: *dữ liệu* là các thực thể dùng để truyền tải ý nghĩa. Trong mấy tính *dữ liệu* là tất cả những gì có thể nhớ được trên các thiết bị nhớ của mấy tính. *Thông tin* là kết quả của quá trình xử lý dữ liệu. *Thông tin* nhận được từ dữ liệu. *Tín hiệu (signal)* là sự mã hóa dữ liệu dưới dạng điện hoặc điện từ. *Phát tín hiệu (signaling)* là hành động gây lan truyền tín hiệu qua một vật dẫn truyền thích hợp nào đó. Và cuối cùng, *sự truyền tin (transmission)* là quá trình trao đổi dữ liệu bằng cách làm lan truyền và xử lý các tín hiệu.

Thiết bị (equipment) trong môi trường truyền thông thường được nối kết qua các đường truyền (link), mỗi đường truyền có thể mang một

hoặc nhiều kênh (channel). Đường truyền là một thực thể vật lý còn kênh chỉ là một thực thể logic. Đường truyền có thể mang dữ liệu dưới dạng tín hiệu số (digital signal) hoặc tín hiệu tương tự (analog signal). Chẳng hạn các đường điện thoại có thể mang dữ liệu dưới dạng tương tự, dù rằng chúng đang dần dần được thay thế bởi các đường truyền thích hợp hơn cho việc truyền tải số. Mỗi đường truyền có một sức tải (capacity), được định nghĩa là số lượng dữ liệu có thể được truyền trên đường truyền trong một đơn vị thời gian. Sức tải này thường được gọi là dải thông (bandwidth) của kênh. Trong các kênh truyền tương tự, dải thông được định nghĩa là hiệu số (tính bằng Hertz) giữa tần số thấp nhất và tần số cao nhất có thể truyền được trên kênh mỗi giây. Trong các đường truyền số, dải thông thường được xem là số bit có thể được truyền trong mỗi giây. Dựa theo dải thông chúng ta có thể xác định ba tầm kênh.

1. Kênh điện thoại tương tự (analog telephone channel): Có thể mang đến 33 Kbps với các kỹ thuật điều chế thích hợp.

2. Kênh điện thoại số (digital telephone channel): Có thể mang 56 hoặc 64 Kbps (được gọi là tốc độ ISDN).

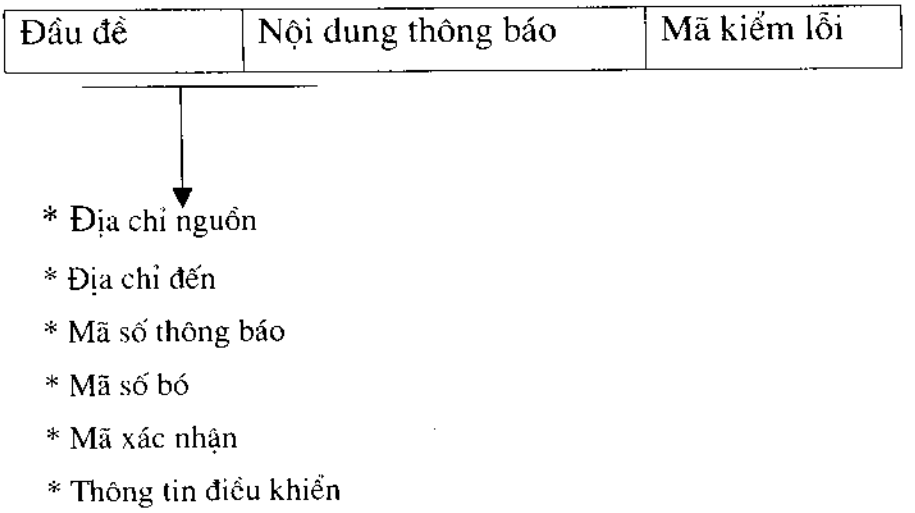
3. Kênh băng rộng (broadband channel): Có thể mang 1,5 Mbps hoặc hơn; chúng tạo ra các thành phần chính cho các mạch điện thoại số.

Nếu dữ liệu được truyền trên các kênh tương tự thì nó phải được điều chế (modulate). Có nghĩa là dữ liệu số được mã hóa thành các tín hiệu mang tương tự (analog carrier signal) bằng cách thay đổi một hoặc nhiều đặc tính cơ bản (biên độ, tần số và pha). Tín hiệu mang đã điều chế sẽ được truyền đến đầu nhận, và tại đó nó lại được tái điều chế thành dạng số. Ưu điểm của việc sử dụng các đường truyền có dải thông cao là dữ liệu truyền có thể được dồn kênh (multiplex), nhờ đó có thể truyền cùng lúc được nhiều tín hiệu. Có hai kiểu dồn kênh cho phép truyền đồng thời nhiều kênh logic

trên một đường truyền vật lý. Một là chia dải thông sao cho mỗi tín hiệu được truyền ở một tần số khác nhau. Dạng dồn kênh này được gọi là dồn kênh phân tần (freauency - division multiplexing, FDM). Một kiểu khác là chia thời gian truyền thành từng khoảng và dành toàn bộ kênh (nghĩa là toàn bộ băng tần) để truyền một tín hiệu. Dạng dồn kênh này được gọi là dồn kênh phân thời (time - division multiplexing, TDM) và được dùng nhiều hơn trong các quá trình truyền dữ liệu.

Từ góc độ hệ CSDL phân tán, một đặc tính khác của đường truyền dữ liệu là chế độ hoạt động của nó. Một đường truyền có thể hoạt động theo chế độ đơn công (simplex), bán song công (half - duplex) hoặc toàn song công (full - duplex). Một đường truyền hoạt động theo chế độ đơn công chỉ truyền tín hiệu và dữ liệu theo một chiều. Đường truyền bán song công có thể truyền dữ liệu theo cả hai chiều nhưng không thực hiện được cùng một lúc. Quá trình truyền trước tiên sẽ tiến hành theo một chiều, sau đó đường truyền phải "quay đầu lại" thì quá trình truyền theo chiều ngược lại mới có thể bắt đầu. Đường truyền toàn song công có thể truyền tín hiệu và dữ liệu theo cả hai chiều đồng thời. Chúng là môi trường linh hoạt nhất và có chi phí cao nhất.

Khi truyền tải giữa các máy tính, dữ liệu thường được truyền theo từng *bó dữ liệu* (frame). Thường thì giới hạn trên của kích thước bó dữ liệu phải được thiết lập cho mỗi mạng và mỗi bó chứa dữ liệu cùng các thông tin điều khiển như nơi đến và địa chỉ nguồn, mã kiểm lỗi cho khối, v.v. (xem hình 1.2). Nếu một thông báo cần phải gửi từ một nút nguồn đến một nút đích nhưng không xếp vừa vào được một bó, nó sẽ được tách ra thành nhiều bó.



Hình 1.2 Dạng thức bó điển hình

Trong chương này, chúng ta sẽ bàn về các *gói (packet)* và kỹ thuật chuyển *mạch gói (packet switching)*. Thuật ngữ *gói* và *bó* đôi khi được dùng lẫn lộn nhưng điều này không hoàn toàn chính xác mặc dù chúng đề cập đến những khái niệm gần giống nhau. Nói theo kiểu các giao thức truyền thông, chúng đề cập đến các thực thể ở những tầng khác nhau. Từ quan điểm thực hành, khác biệt giữa *gói* và *bó* thường được xem xét qua dạng thức của chúng. Một dạng thức gói chứa thông tin tiêu đề cho tầng mạng, nghĩa là thông tin chọn đường (routing), còn một bó chỉ gồm các thông tin liên quan đến các cơ chế khả tín của tầng liên kết dữ liệu.

1.2. CÁC LOẠI MẠNG MÁY TÍNH

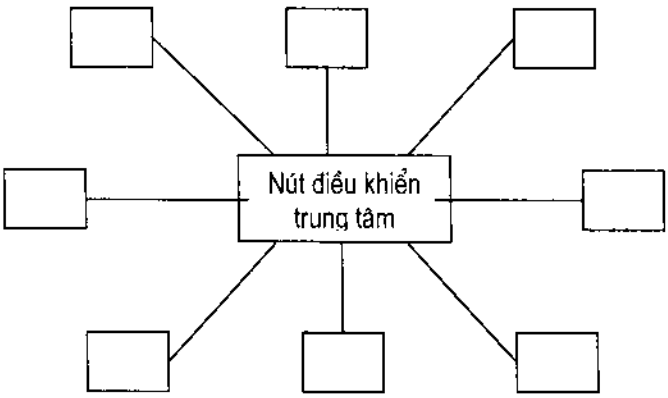
Có rất nhiều chuẩn dùng để phân loại mạng máy tính. Một chuẩn thường dùng là *cấu trúc nối kết* (interconnection structure) của các máy tính

(thường được gọi là topo mạng), một chuẩn khác là *chế độ truyền* và một chuẩn nữa là *sự phân bố địa lý*

1.2.1 CÁC KIỂU CẤU TRÚC KẾT NỐI

Như tên gọi, cấu trúc nối kết muốn nói đến cách nối các máy tính trên một mạng lại với nhau. Một số kiểu thông dụng là mạng hình sao (star), mạng vòng (ring), mạng bus, mạng đầy đủ (meshed) và mạng vô định hình (irregular). . .

Trong các mạng hình sao (hình 1.3), tất cả các máy tính đều được nối với một máy tính trung tâm lo điều phối việc truyền dữ liệu trên mạng. Vì vậy nếu hai máy tính muốn trao đổi với nhau, chúng phải thông qua máy tính trung tâm. Bởi vì mỗi máy tính đều có đường truyền riêng với máy tính trung tâm nên cần phải có một thoả thuận giữa các máy tính "vệ tinh" và máy tính trung tâm khi chúng muốn trao đổi.



Hình 1.3 Mạng hình sao

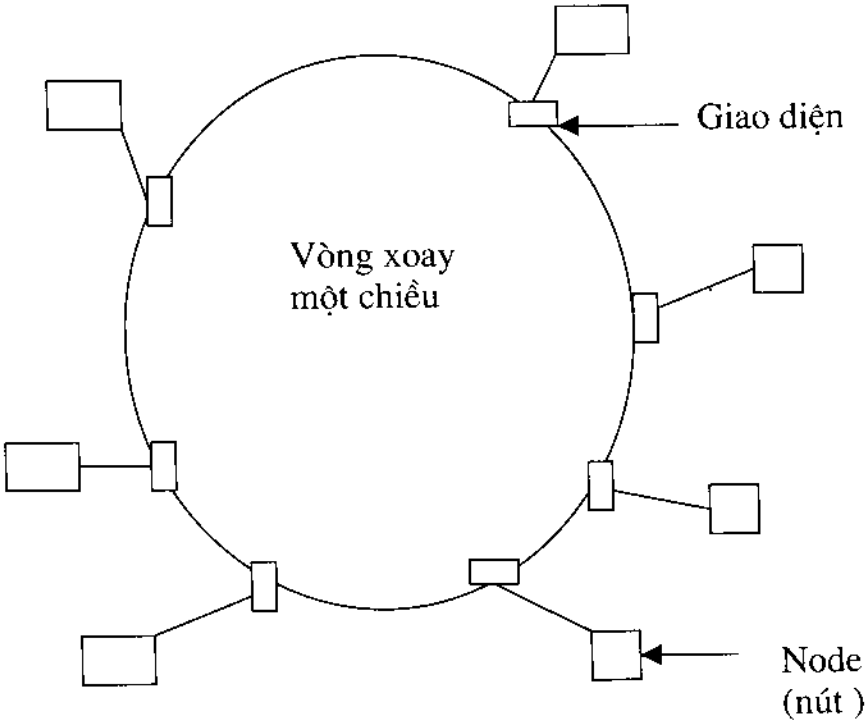
Loại mạng này thường được dùng trong các tổ chức có nhiều chi nhánh nằm ở nhiều vùng khác nhau, máy tính trung tâm được đặt tại văn phòng chính hoặc tại trung tâm vùng. Trong trường hợp này việc xử lý cục bộ được thực hiện tại mỗi nút và dữ liệu cuối cùng sẽ được truyền đến máy trung tâm. Một khuyết điểm của mạng hình sao là độ tin cậy thấp. Vì giao tiếp giữa hai máy tính phụ thuộc vào máy tính trung tâm, một sự cố tại nút này sẽ làm cho việc truyền trên mạng ngừng trệ hoàn toàn. Một khuyết điểm khác là tải trọng quá cao trên máy trung tâm; vì nó phải điều phối việc giao tiếp trên mạng, tải trọng tại đó cao hơn các trạm khác. Vì thế người ta thường dùng một trạm trung tâm mạnh hơn các máy tính vệ tinh. Do những khuyết điểm này, mạng hình sao thường chỉ được dùng khi lượng dữ liệu cần truyền giữa các máy vệ tinh không cao.

Trong các mạng xoay vòng (hình1.4), các máy tính được nối với môi trường truyền (đường truyền) có dạng một vòng khép kín. Truyền dữ liệu quanh vòng thường theo một chiều, và mỗi trạm (thực sự là giao diện tại mỗi trạm) đóng vai trò là một bộ chuyển tiếp (repeater). Khi nhận được một thông báo (message), nó kiểm tra địa chỉ, sao chép thông báo đó nếu là thông báo được gửi cho nó rồi truyền thông báo đi tiếp.

Ngược lại khi nhận được một thông báo (message), nó kiểm tra địa chỉ, sao chép thông báo đó nếu là thông báo không phải được gửi cho nó, nó ghi lại rồi truyền thông báo ngược về nơi đã gửi đến cho nó.

Mạng xoay vòng có nhiều nhược điểm khi một node có sự cố, Vì vậy ngày nay người ta thường dùng các vòng xoay kép để dự phòng khi một node bị sự cố mạng vẫn hoạt động bình thường nhờ vào đường dẫn thứ hai.

Mạng xoay vòng có ưu điểm lớn dễ kiểm soát và lắp đặt.



Hình 1.4 Mạng xoay vòng

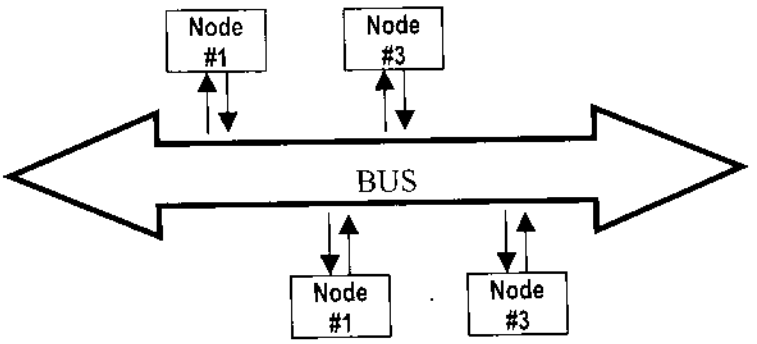
Việc điều khiển truyền tin trên mạng xoay vòng thường được thực hiện bằng thẻ điều khiển (control token). Trong kiểu đơn giản nhất, một thẻ (token) với một mẫu bit chỉ ra rằng mạng hiện đang rảnh và một mẫu bit khác cho biết rằng mạng đang được dùng, được chuyển xoay vòng trên mạng. Mỗi trạm khi muốn truyền thông báo phải đợi đến khi thẻ được truyền đến. Khi đó trạm sẽ kiểm tra mẫu bit của thẻ để xem mạng đang rảnh hay đang được dùng. Nếu mạng rảnh, trạm sẽ thay đổi mẫu bit, chỉ ra rằng mạng đang được dùng rồi đặt các thông báo vào vòng xoay. Thông báo sẽ

được chuyển xoay vòng rồi trở về trạm gửi để được nó đổi lại mẫu bit thành "đang rảnh" và thẻ sẽ được gửi đến trạm kế tiếp.

Các mạng chỉ có một môi trường truyền kiểu xoay vòng thì có độ tin cậy thấp, đơn giản là đường nối chỉ cần bị cắt đứt tại một điểm nào đó là có thể làm ngừng toàn bộ hoạt động của mạng. Để có được độ tin cậy cao hơn, người ta có thể sử dụng loại mạng hai vòng. Trong một mạng như thế, sự cố tại một điểm nối không làm mất khả năng truy xuất đến phần còn lại của mạng bởi vì có thể truyền tắt qua trạm bị hư bằng cách chuyển đường truyền sang vòng thứ hai.

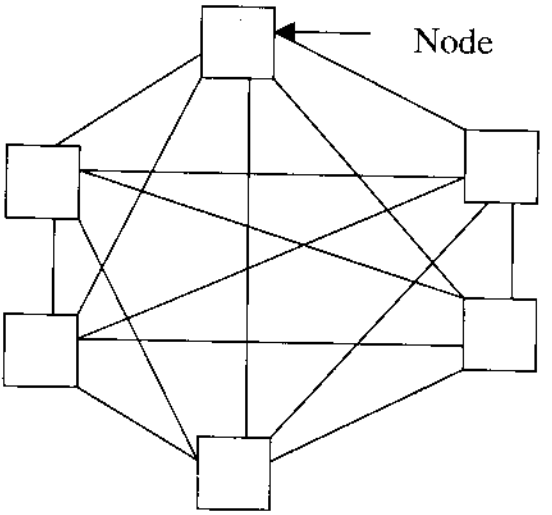
Một thể thức khác nhằm đảm bảo độ tin cậy là sử dụng một nút chuyển mạch trung tâm (central switch). Các nối kết giữa các trạm được thực hiện qua trung tâm chuyển mạch dù hoạt động của mạng có thể ở dạng xoay vòng. Nếu một trạm bị sự cố, hoặc nếu đường nối bị đứt liên lạc thì dễ dàng đi tắt qua phần mạng đó thông qua nút chuyển mạch. Kiến trúc này đã được phát triển tại phòng thí nghiệm của IBM tại Zurich và được cài đặt trên mạng LAN token ring của IBM.

Một loại mạng thông dụng khác là mạng bus (hình 1.5), trong đó có một kênh chung để truyền dữ liệu, các máy tính và các thiết bị đầu cuối sẽ được gắn vào đó. Ở kiểu mạng này, việc điều khiển đường nối được thực hiện bằng hai cách chính . Một là phương pháp CSMA (carrier sense multiple access) và phương pháp thứ hai là CSMA/CD (carrier sense multiple access with collision detection). Ngoài hai phương pháp cơ bản này, bus cũng có thể được điều khiển bằng thẻ. Nếu sử dụng lược đồ này, mạng bus được xem như có một vòng xoay logic.



Hình 1.5 Mạng bus

Cơ chế kiểm soát bus kiểu CSMA có thể được mô tả là lược đồ "lắng nghe trước khi truyền". Điểm cơ bản đó là mỗi trạm sẽ liên tục lắng nghe mọi diễn biến xảy ra trên kênh chung. Khi có một thông báo được gửi đi, trạm sẽ kiểm tra phần header của thông báo xem có phải gửi cho nó hay không, rồi thực hiện một hành động thích hợp. Nếu nó muốn truyền, nó sẽ chờ cho đến khi phát hiện ra không còn hoạt động nào xảy ra trên kênh chung rồi mới đặt thông báo của nó lên mạng. Ngược lại, cơ chế điều khiển bus CSMA/CD có thể được mô tả là lược đồ "lắng nghe trong khi truyền". Loại cơ bản hoạt tác theo cách sau. Các trạm đóng vai trò giống như trong lược đồ CSMA, ngoại trừ chúng tiếp tục lắng nghe trên kênh chung sau khi đã truyền thông báo đi. Mục đích của việc lắng nghe trong khi truyền là phát hiện xem có tương tranh (collision) hay không.



Hình 1.6 Mạng thảm (cấu trúc nối đầy đủ)

Tương tranh xảy ra khi hai trạm truyền thông báo đồng thời (một trạm khởi truyền khi một trạm khác đang truyền). Trong trường hợp như thế, và khi phát hiện ra tương tranh, các trạm sẽ hủy bỏ cuộc truyền, đợi một khoảng thời gian rồi truyền lại thông báo. Lược đồ CSMA/CD cơ bản được dùng trong mạng cục bộ Ethernet.

Một lược đồ nối kết khác là nối kết đầy đủ (mạng thảm), trong đó mỗi nút đều được nối với tất cả mọi nút khác (hình 1.6). Một cấu trúc như thế rõ ràng là cung cấp được độ tin cậy cao hơn và khả năng hoạt động tốt hơn những cấu trúc khác. Tuy nhiên nó cũng là cấu trúc có chi phí cao nhất, nên ít được dùng và không thực tế. Ngay cả khi số lượng máy tính trên mạng khá ít, số nối kết cần có vẫn rất lớn. Thí dụ một nối kết đầy đủ cho 10.000 máy tính cần xấp xỉ $(10.000)^2$ đường nối.

Các mạng truyền thông thường có các đường nối vô định. Nghĩa là các đường nối không có tính hệ thống cũng không tuân theo một khuôn mẫu nào. Chúng ta có thể gặp một nút chỉ nối với một nút khác và cả những nút nối với nhiều nút khác. Các nối kết giữa các máy tính trên Internet thuộc loại này.

1.2.2 CÁC LƯỢC ĐỒ TRUYỀN DỮ LIỆU

Theo các lược đồ truyền thông vật lý được dùng, các mạng có thể thuộc loại điểm đến điểm (point - to - point) hoặc phát tán và còn được gọi là đa điểm (multi - point).

Trong các mạng điểm đến điểm, người ta dùng một hoặc nhiều đường nối giữa mỗi cặp nút. Có thể là không có một đường nối trực tiếp giữa mỗi cặp nhưng thường là một số đường nối gián tiếp. Việc truyền thông (giao tiếp) luôn được thực hiện giữa hai nút, bên nhận và bên gửi được xác định bằng địa chỉ có trong phần header của bó dữ liệu. Truyền dữ liệu từ bên gửi đến bên nhận đi theo một hoặc nhiều đường giữa chúng, một số đường có thể phải đi ngang qua một số nút khác. Các nút trung gian sẽ kiểm tra địa chỉ đích trong phần header và nếu không phải là địa chỉ của nó thì sẽ chuyển cho nút nằm kế cận. Hành động này được gọi là chuyển mạch (switching). Việc chọn các đường nối để truyền các bó dữ liệu được xác định qua các giao thức thích hợp.

Môi trường truyền cơ sở cho mạng điểm đến điểm là cáp đồng trục hoặc cáp quang. Các đường dây điện thoại nối thiết bị của khách hàng thường dùng các dây xoắn đôi, nên tốc độ truyền không cao. Các mạng truyền hình cáp sử dụng đường dây đồng trục dẫn đến tận nhà cho phép kết nối mạng với tốc độ truyền cao. Tương tự nhiều mạng cục bộ đều dùng loại

cáp đồng trục. Tuy nhiên hiện nay người ta đang chuyển sang dùng cáp quang với sức tải và tốc độ cao hơn.

Trong các mạng phát tán, người ta dùng một kênh truyền chung cho tất cả các nút trong mạng. Các bó dữ liệu được truyền qua kênh chung này và như thế tất cả các nút đều nhận được. Mỗi nút sẽ kiểm tra địa chỉ bên nhận trong phần header và nếu bó dữ liệu không gửi cho nó, nó sẽ bỏ qua.

Một trường hợp đặc biệt của mạng phát tán là mạng đa tán (multicast), trong đó thông báo được gửi đến một tập con các nút trong mạng. Địa chỉ bên nhận được mã hóa bằng một cách nào đó để có thể chỉ ra những nút nào là bên nhận.

Mạng phát tán nói chung đều dùng sóng radio hoặc vệ tinh. Trong trường hợp truyền qua vệ tinh, mỗi vị trí phát tín hiệu truyền của nó đến vệ tinh rồi tín hiệu đó được phát trả lại ở một tần số khác . Mỗi vị trí trên mạng đều lắng nghe tần số nhận và phải bỏ qua thông báo không được gửi cho nó. Một mạng có sử dụng kỹ thuật này là mạng SATNET.

Truyền bằng sóng vi ba (microwave) là một cách truyền dữ liệu thông dụng khác, có thể qua vệ tinh hoặc trên mặt đất. Các đường truyền bằng sóng vi ba hiện là phương thức chủ yếu của mạng điện thoại trong phần lớn các quốc gia. Ngoài các dịch vụ công cộng, nhiều công ty còn cho thuê riêng các đường truyền vi ba. Thực sự các thành phố đông dân hiện đang gặp phải vấn đề nhiễu sóng vi ba giữa các đường truyền tư nhân và công cộng. Một thí dụ về mạng dùng sóng vi ba vệ tinh để truyền dữ liệu là hệ thống ALONA.

Một điều cuối cùng cần nói về kiểu topo mạng phát tán là chúng ta dễ dàng phát hiện lỗi, và các thông báo có thể đến được nhiều vị trí hơn so với kiểu điểm đến điểm. Ngược lại do mỗi trạm đều lắng nghe các thông báo trên mạng nên tính an ninh khó duy trì hơn so với kiểu điểm đến điểm.

1.2.3 TẦM ĐỊA LÝ

Theo sự phân phối về mặt địa lý, các mạng có thể được phân loại là mạng diện rộng (wide area network, WAN), mạng liên vùng (metropolitan area network, MAN) và mạng cục bộ (local area network, LAN). Sự phân biệt này thường không rõ ràng mà phân biệt chủ yếu giữa các loại mạng là ở giao thức được dùng để quản lý chúng. Trong phần tiếp theo chúng ta sẽ thảo luận sơ qua về các giao thức mạng diện rộng và mạng cục bộ.

Một mạng diện rộng (WAN) là những mạng có khoảng cách đường nối giữa hai nút xấp xỉ hoặc trên 20 km và có thể dài đến vài ngàn cây số. việc sử dụng các thiết bị chọn đường (router) hoặc/ và các nút chuyển (switch) cho phép truyền thông tin trên những vùng rộng lớn hơn, nhưng sự tăng tầm địa lý lại làm giảm hiệu năng từ những chậm trễ do nhiều nút chuyển/ thiết bị chọn đường được đưa vào giữa hai đầu truyền thông. Các mạng WAN có thể được xây dựng với kiểu topo mạng điểm đến điểm hoặc kiểu phát tán, mặc dù kiểu điểm đến điểm thông dụng hơn. Có một số dạng chuyển mạch (switching) được dùng trong các mạng điểm đến điểm. Dạng đầu tiên là dành hẳn một kênh trong suốt quá trình kết nối giữa bên gửi và bên nhận. Dạng này được gọi là chuyển mạch cứng (circuit switching) và thường được sử dụng trong hệ thống điện thoại. Khi một thuê bao quay số gọi một thuê bao khác, một mạch nối (circuit) được thiết lập giữa hai máy điện thoại qua rất nhiều nút chuyển mạch. Mạch nối này được duy trì trong suốt thời gian điện đàm và chỉ bị cắt khi một bên ngắt máy.

Một dạng chuyển mạch khác thường được sử dụng trong việc truyền thông tin giữa các máy tính là chuyển mạch gói (packet switching), trong đó một thông báo (message) được tách nhỏ thành nhiều gói (Packet) và mỗi gói được truyền đi riêng rẽ. Các gói của cùng một thông báo có thể di chuyển

độc lập và thực sự có thể được truyền trên những tuyến đường khác nhau. Kết quả của việc dùng các đường đi khác nhau trên mạng đó là chúng có thể đến đích một cách lộn xộn. Vì thế phần mềm tại nơi nhận phải có khả năng sắp xếp chúng theo đúng thứ tự, tái tạo lại thông báo ban đầu.

Ưu điểm của chuyển mạch gói thì rất nhiều. Trước tiên các mạng chuyển mạch gói cho phép sử dụng đường truyền tốt hơn bởi vì mỗi đường truyền không phải chỉ dành riêng cho mỗi cặp thiết bị mà có thể được nhiều thiết bị dùng chung. Điều này rất có ích trong việc truyền thông máy tính do bản chất "phong trào" của nó. Thông thường người sử dụng gõ một lệnh, đợi nó được thực thi và trả lời rồi phải cần thời gian suy nghĩ trước khi đưa một lệnh mới vào. Trong một môi trường như thế, việc truyền dữ liệu trên mạng không liên tục nhưng theo từng đợt. Các đường truyền khi đó có thể được dùng cho những người khác khi một người sử dụng trước đó đang đợi trả lời hoặc đang suy nghĩ. Một lý do nữa là việc tách gói cho phép truyền song song dữ liệu. Hệ thống không nhất thiết phải truyền các gói của cùng một thông báo trên cùng một tuyến đường. Như thế chúng có thể được gửi đi song song qua các tuyến đường nhằm cải thiện được tổng thời gian truyền. Như đã nói ở trên, kết quả chuyển dữ liệu theo cách này đó là thứ tự của chúng không được bảo đảm.

Ngược lại, chuyển mạch cứng dành hẳn một kênh giữa bên nhận và bên gửi. Nếu cần truyền một lượng lớn dữ liệu thì kênh dành riêng này rất có ích. Vì thế các lược đồ tương tự như chuyển mạch cứng (nghĩa là các lược đồ đăng ký, reservation - based scheme) rất được ưa chuộng trong các mạng dải rộng (broadband network) có hỗ trợ các ứng dụng cần truyền rất nhiều dữ liệu như các ứng dụng đa phương tiện (multimedia).

Mạng cục bộ (local area network, LAN) thường là mạng truyền gói và hạn chế trong một phạm vi địa lý nhất định (thường dưới 2 km). Chúng

sử dụng môi trường truyền có dải thông cao nhưng chi phí không cao. Topo mạng thông dụng nhất là kiểu bus và kiểu xoay vòng (ring) và các biến thể của chúng như bus chuyển mạch hoặc vòng xoay chuyển mạch. Các môi trường truyền được dùng trong mạng LAN là cáp đồng trục, cáp xoắn đôi hoặc cáp quang. Giữa mạng diện rộng và mạng cục bộ có những khác biệt sau :

1. Trong mạng WAN, chi phí truyền thông rất cao còn ở mạng LAN lại tương đối thấp. Có nhiều lý do nhưng rõ ràng nhất là khoảng cách truyền trong mạng LAN nhỏ hơn nhiều.

2- Mạng WAN truyền thống thường có dải thông bị giới hạn ở khoảng vài megabit mỗi giây (Mbps), trong khi đó ở mạng LAN, dải thông có thể lớn hơn nhiều, và thường vào khoảng 10 - 100 Mbps.

3. Do khoảng cách xa nên trong WAN, độ trễ khi truyền khá lớn. Chẳng hạn qua vệ tinh, độ trễ tối thiểu là khoảng 1/2 giây khi truyền từ nguồn đến đích. Điều này là do tốc độ tín hiệu không thể vượt quá tốc độ ánh sáng và khoảng cách cần truyền quá lớn (khoảng 19.200 miles từ trái đất đến vệ tinh). Ngược lại trong mạng cục bộ, độ trễ này rất nhỏ.

4. Do tính đa chủng loại của môi trường truyền, máy tính, cộng đồng người sử dụng cũng như chất lượng thấp của đường truyền, các giao thức trong mạng WAN phải bảo đảm được độ tin cậy khi truyền. Trong mạng LAN, các đường truyền "Sạch hơn", tính đa chủng của các máy tính nối mạng dễ quản lý hơn và do chúng sử dụng môi trường truyền chung nên thường chỉ cần các giao thức đơn giản là đủ.

5. Mạng LAN thường được quản lý và sử dụng bởi một tổ chức. Tuy nhiên mạng WAN hiếm khi được chính những người sử dụng sở hữu. Nghĩa là người sử dụng mạng LAN mua sản phẩm còn người sử dụng mạng WAN mua dịch vụ.

Các mạng LAN cũng cung cấp một số tiện ích như các ứng dụng tự động hóa công việc văn phòng, các ứng dụng kiểm soát tiến trình phân tán.

Mạng liên vùng (MAN) nằm lưng chừng giữa mạng WAN và LAN về tầm địa lý và thương bao phủ một thành phố hay một phần của nó. Khoảng cách giữa các nút thường khoảng 10 km. MAN có nhiều điểm tương đồng với LAN, và theo một nghĩa nào đó có thể được xem như một phiên bản LAN rộng hơn. Tuy nhiên trong MAN do lượng người dùng nhiều hơn làm nảy sinh nhiều vấn đề mới cần phải giải quyết, chẳng hạn như sự bình đẳng truy xuất cho tất cả mọi người dùng bất kể khoảng cách địa lý. Vì vậy mặc dù về nguyên tắc một số giao thức của mạng LAN có thể được "Nới tầm" để dùng cho MAN nhưng vẫn cần phải có một tập giao thức riêng rẽ và phải xem xét kỹ lưỡng các vấn đề thiết kế.

1.3. CÁC CHUẨN GIAO THỨC

Thiết lập các đường nối vật lý giữa hai máy tính chưa đủ để chúng giao tiếp được với nhau. Truyền thông tin hiệu quả, đáng tin cậy và không có lỗi giữa hai máy tính đòi hỏi phải cài đặt các hệ thống phần mềm thích hợp và thường được gọi là *giao thức (protocol)*. Tính chất phức tạp của những giao thức này đều khác nhau giữa các mạng WAN, MAN và LAN.

Mạng WAN thường phải điều chỉnh thiết bị được sản xuất từ nhiều nhà sản xuất khác nhau. Điều này đòi hỏi môi trường truyền phải có khả năng xử lý tính đa chủng (heterogeneity) của các thiết bị và cách nối kết. Các thiết bị có thể khác nhau về tốc độ, chiều dài từ nhớ (word), lược đồ mã hóa (coding scheme) được dùng để biểu diễn thông tin hoặc các chuẩn khác. Vì thế mạng WAN có nhu cầu về giao thức cấp thiết hơn. Do vậy trước tiên chúng ta sẽ thảo luận về các giao thức trong mạng WAN rồi chuyển sang

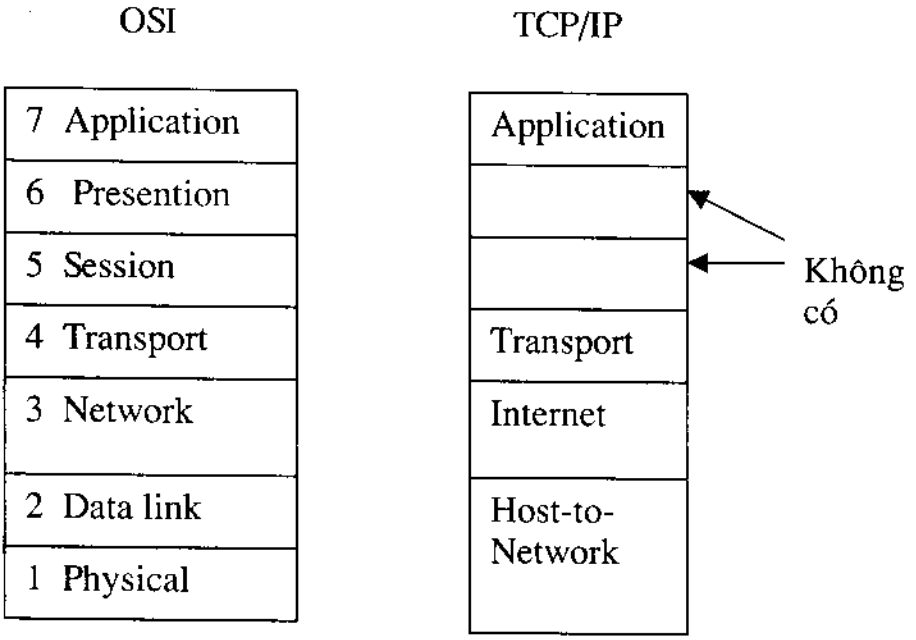
các giao thức cho LAN. Cho đến gần đây, giao thức cho WAN được biết rộng rãi nhất là Kiến trúc giao kết các hệ thống mở (open systems Interconnection Architecture) của Tổ chức tiêu chuẩn quốc tế (International Standards Organization, ISO) và thường được gọi là kiến trúc ISO/OSI (ISO, 1983).

Kiến trúc ISO/OSI mô tả rằng cần xây dựng mạng máy tính theo kiểu phân tầng (vì thế có thuật ngữ chồng giao thức, protocol stack). Giữa các tầng (layer) của một nút cần định nghĩa rõ ràng các giao diện (interface) dùng để trao đổi thông tin giữa các tầng phần mềm và phần cứng. Giữa các tầng tương ứng của các trạm khác nhau, các giao thức (protocol) được định nghĩa và đặc tả cách trình bày thông báo được gửi qua lại giữa hai trạm. Kiến trúc ISO/OSI với cấu trúc gồm có bảy tầng. Khởi đầu từ tầng thấp nhất, lần lượt là tầng vật lý (physical layer), tầng liên kết dữ liệu (data link layer), tầng mạng (network layer), tầng giao vận (transport), tầng phiên (session layer), tầng trình diễn (presentation layer) và tầng ứng dụng (application layer). Ba tầng thấp nhất là tầng vật lý, tầng liên kết dữ liệu và tầng mạng tạo ra tiểu mạng truyền thông (communication subnet). Tiểu mạng truyền thông chịu trách nhiệm cung cấp độ tin cậy vật lý cho việc truyền thông tin giữa hai trạm. Chúng tôi không trình bày chi tiết những tầng này.

Một chồng giao thức WAN thông dụng khác là TCP/IP. Ý tưởng tổng quát giống như ISO/OSI nhưng số lượng tầng chỉ là năm thay vì là bảy. Chồng giao thức này đã "nổi lên" chứ không phải đã phát triển như một mô hình nhất quán.

Mối liên hệ giữa các giao thức ISO/OSI và TCP/IP được mô tả trong hình 1.7

Một khác biệt quan trọng giữa hai chồng giao thức này là tất cả các tầng của ISO/OSI đều được định nghĩa rõ, còn trong TCP/IP tầng host - to - network không được đặc tả.



Hình 1.7 So sánh giữa TCP/IP và ISO/OSI

Kết nối mạng trong mạng cục bộ dường như đơn giản hơn trong mạng WAN bởi vì chúng ta thường chỉ phải quan tâm đến ba tầng thấp nhất trong buồng giao thức và trong LAN thiết bị mạng thường "đồng chủng" hơn. Tuy nhiên như chúng ta sẽ thấy, việc truyền thông trong LAN cũng phải có sự điều hoạt tại tất cả các tầng mạng và cũng thường được thực hiện bằng các giao thức TCP/IP.

1.4. MẠNG DẢI RỘNG VÀ CÁC DỊCH VỤ

Cho đến lúc này, chúng ta đã tập trung vào các "mạng dữ liệu" hoặc các mạng được cấu trúc đặc biệt để mang dữ liệu số, hoặc ở dạng số hoặc ở dạng tương tự đã được điều chế. Vì vậy, ít nhất là về mặt logic, các mạng dữ liệu khác biệt hoàn toàn với các mạng điện thoại (truyền âm thanh). Tuy nhiên nhiều ứng dụng mới (thí dụ các hệ thông tin đa phương tiện) có nhu cầu truyền tải các dạng dữ liệu khác ngoài dữ liệu số, như hình ảnh video hoặc âm thanh với các yêu cầu phân phối theo thời gian thực và những hình ảnh tĩnh với các yêu cầu dải rộng đủ lớn (một hình X-quang số 1024 x 1024 với 8 bit/pixel cần 10 Mbps ở dạng chưa nén). Các mạng dải rộng được thiết kế để đáp ứng những yêu cầu này trong một môi trường mạng duy nhất. Các đặc trưng nhận diện của chúng là sức tải cao (lớn hơn 150 Mbps), khả năng mang nhiều dòng dữ liệu với các đặc tính khác nhau, và khả năng thoả thuận về một mức chất lượng dịch vụ và có thể dành đủ tài nguyên mạng để đáp ứng được mức chất lượng này.

Công nghệ mạng dải rộng thông dụng nhất hiện nay là ATM (Asynchronous Transfer Mode). Mạng ATM đã được phát triển cho những ứng dụng WAN và LAN. Ở mức người dùng, ATM hỗ trợ năm lớp dịch vụ.

1.4.1 DỊCH VỤ CBR

Đây là dịch vụ tốc độ bít cố định (constant bit rate), trong đó mạng truyền dữ liệu ở một tốc độ bit như đã được thoả thuận trước. Dịch vụ này được dùng để truyền video và âm thanh (dòng dữ liệu theo thời gian thực) trong đó nguồn sẽ cung cấp dòng dữ liệu một cách đều đặn với một tốc độ đã thoả

thuận trước. Nó không bao gồm các dịch vụ tương tác, vì thế nó thích hợp hơn cho một số ứng dụng, chẳng hạn như các dịch vụ cung cấp phim theo yêu cầu.

1.4.2 DỊCH VỤ UBR

UBR là một dịch vụ với tốc độ bit không xác định (unspecified bit rate), thích hợp với những ứng dụng cần gửi dữ liệu theo từng đơn vị chứ không cần ở một tốc độ cố định. Phần lớn các giao tiếp máy tính đều theo cách này; không có ràng buộc thời gian thực và dữ liệu được yêu cầu theo từng đợt. Dịch vụ UBR sẽ nỗ lực tối đa để phân phối dữ liệu nhưng không đưa ra bất kỳ một bảo đảm nào.

1.4.3 DỊCH VỤ RT-VBR

Dịch vụ này cũng dành cho dòng dữ liệu theo thời gian thực nhưng tốc độ của nguồn được phép thay đổi. Những thay đổi này cho phép thực hiện các tối ưu hóa bởi vì nguồn với các tốc dộ bit thay đổi có thể được đa hợp nhằm tận dụng tối đa dải thông. Nó cũng thích hợp với các ứng dụng tương tác theo thời gian thực.

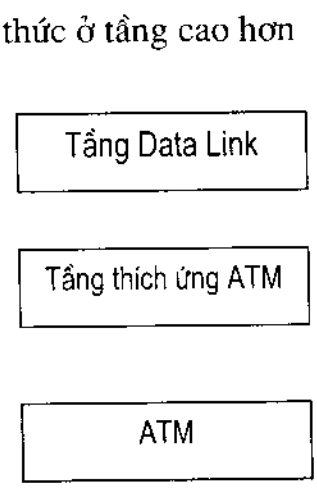
1.4.4 DỊCH VỤ NRT-VBR

Loại dịch vụ có tốc độ bit thay đổi phi thời gian thực, được dùng cho các nhu cầu truyền theo từng đơn vị dữ liệu, tương tự như UBR. Tuy nhiên nó cải thiện các thất lạc và các đặc trưng về độ trễ của UBR bằng cách đưa ra

các tham số QoS như tốc độ cao nhất (peak), tốc độ duy trì (sustainable) và tỷ lệ thất lạc.

1.4.5 DỊCH VỤ ABR

Dịch vụ tốc độ bit sẵn có (available bit rate). Nó gán tốc độ bit hiện có trên mạng cho ứng dụng đang yêu cầu. Mục tiêu là làm giảm các thất lạc bó dữ liệu và hiệu chỉnh các dự trữ của nguồn dựa trên các yêu cầu thay đổi về dòng dữ liệu.



Hình 1.8 Mạng ATM

ATM là mạng chuyển gói với các nút chuyển có mục đích đặc biệt được nối lại bằng các đường cáp quang. Các gói, được gọi là tế bào (cell) trong thuật ngữ ATM, có chiều dài 53 byte (48 byte dữ liệu, 5 byte đầu đề). Công nghệ ATM tương ứng với tầng vật lý của chồng giao thức ISO/OSI và TCP/IP (hình 1.8) và cần phải có một tầng thích ứng ATM (ATM

Adaptation Layer, viết tắt là AAL) để điều chỉnh các khác biệt giữa công nghệ ATM và các công nghệ mạng truyền thống đã được xây dựng cho các tầng giao thức bên trên. AAL chịu trách nhiệm xử lý các tế bào bị thất lạc và bị phân phối sai, chọn thời gian khôi phục, tách các bó dữ liệu từ các tầng giao thức bên trên thành các tế bào ATM ở nguồn và tái hợp lại ở đích. Công việc chọn đường và đa hợp/ giải hợp (multiplex/demultiplex) các tế bào được thực hiện bởi tầng ATM bằng cách dùng các nút chuyển ATM.

Các mạng dải rộng hiện nay hoạt tác với tốc độ khoảng 155 Mbps. Có nhiều hệ thống ATM thử nghiệm cho WAN đang hoạt động và nhiều mạng LAN ATM đã được phát triển. Khả năng mang nhiều loại dữ liệu ở tốc độ rất cao và cơ hội nối kết liên mạng với công nghệ khác đã thu hút nhiều sự quan tâm đối với công nghệ này.

1.5. MẠNG VÔ TUYẾN

Hoạt động không cố định và các xử lý di động đang nổi lên như một lực lượng quan trọng. Các hệ thống điện thoại vô tuyến hiện đang phổ biến rộng rãi ở nhiều nước trên thế giới. Những hệ thống ban đầu đều thuộc loại tương tự và dựa trên phương pháp điều chế tần số. Phần lớn các mạng vô tuyến hiện nay đang được chuyển dần thành mạng số và chúng làm tăng khả năng xử lý di động.

Thuật ngữ "vô tuyến" (wireless) được dùng ở đây không chuẩn lắm. Các truyền thông qua vệ tinh và dùng sóng vi ba đã có từ lâu và thực sự đều là vô tuyến. Các mạng "vô tuyến" hiện nay dành cho việc tính toán di động thực sự là các mạng "tế bào" (cellular network). Những mạng này bao gồm một mạng xương sống hữu tuyến (wireline backbone network) trên đó có

chứa một số trạm điều khiển (control station). Mỗi trạm điều khiển lo điều phối việc giao tiếp từ các máy tính di động trong phạm vi tế bào của nó đến một máy tính di động trong tế bào đó hoặc trong một tế bào khác hoặc với một mấy tính cố định trên mạng hữu tuyến.

Trong các mạng tế bào, mỗi tế bào được tổ chức (về mặt logic) như một topo mạng hình sao với trạm điều khiển được dùng làm nút trung tâm. Thiết lập giao tiếp giữa hai trạm di động trong cùng tế bào hoàn toàn đơn giản. Thiết lập giao tiếp giữa các trạm ở các tế bào khác nhau cần phải được điều phối bởi nhiều trạm điều khiển. Bởi vì các trạm di động có thể di chuyển được, chúng có thể đi ngang qua đường biên của một số tế bào. Điều này đòi hỏi một quá trình (bàn giao) trong đó một trạm điều khiển sẽ bàn giao trạm di động cho một trạm điều khiển khác. Theo dõi sự di động này đòi hỏi phải có một cách nào đó để quản lý thư mục.

Có một số loại trạm di động khác nhau. Thứ nhất là loại bao gồm các máy tính khá đơn giản với khả năng hạn chế. Trong trường hợp này, dữ liệu được lưu trên các máy tính của mạng hữu tuyến và các trạm di động sẽ "tải" dữ liệu xuống khi cần. Bối cảnh này là thực tế đối với một số ứng dụng. Tuy nhiên trong trường hợp này, bài toán quản lý dữ liệu phân tán không bị ảnh hưởng nhiều bởi tính chất di động nhờ dữ liệu nằm chủ yếu trên các máy hữu tuyến. Đáng chú ý hơn là môi trường trong đó các trạm di động có khả năng tính toán mạnh và khả năng lưu trữ dữ liệu của riêng nó và có thể có những máy khác cần dùng dữ liệu đó - chúng được gọi là các "Trạm du mục" (walkstation). Cách tiếp cận này gây ra nhiều khó khăn cho việc quản lý dữ liệu bởi vì các đặc trưng truyền thông, tính chất di động và tính đa tương hợp của môi trường di động.

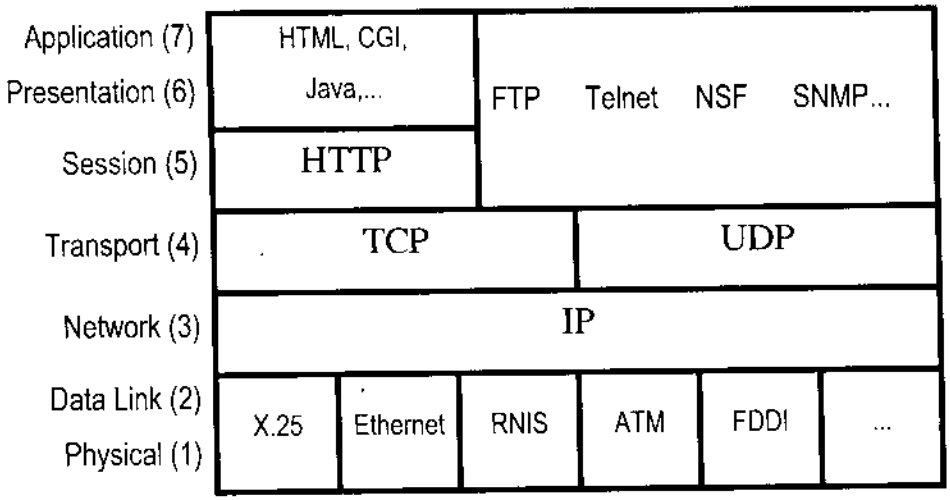
Việc truyền thông trên các mạng vô tuyến rất dễ bị mất liên lạc, nhiễu ồn, tạp âm và dải thông thấp. Tính chất di động của một số thiết bị

trên mạng làm cho các dữ liệu tĩnh trên các trạm cố định bị thay đổi liên tục và dễ bị tổn hại. Tính chất di động làm nảy sinh các vấn đề như thay đổi địa chỉ, duy trì thư mục và khó định vị các trạm. Cuối cùng, tính đa tương hợp buộc phải hạn chế một số loại thiết bị có thể được dùng trong những môi trường này. Thí dụ tính đa tương hợp và yêu cầu ít phải sạc pin (hoạt động được lâu) thường hạn chế loại và kích thước lưu trữ có thể được dùng.

1.6. INTERNET

Mạng Internet là một từ được dùng để chỉ đến một mạng máy tính toàn cầu. Thực sự đó là sự liên hiệp đa chủng của nhiều mạng, mỗi mạng có các đặc tính và giao thức riêng. Nối kết vào Internet là tự nguyện và hầu như không có một tổ chức nào điều khiển hoặc áp đặt các chiến lược và chỉ dẫn việc trao đổi thông tin trên các mạng này. Ngay cả IETF (Internet Engineering Task Force) cũng có ảnh hưởng rất ít đến Internet.

Số nút kết nối vào Internet tăng lên rất nhanh, tăng theo cấp số nhân, dự kiến trong những năm tới là khoảng hàng ngàn triệu máy. Có lẽ động lực chủ yếu cho sự phát triển nhanh chóng và mạnh mẽ của mạng Internet là sự chấp nhận giao thức TCP/IP làm giao thức chính. TCP/IP hiện đã được đưa vào hầu hết các hệ điều hành, tạo dễ dàng cho việc kết nối vào Internet vì nó thích hợp với nhiều giao thức.



Hình 1.9 Bộ giao thức Internet

Mạng Internet đã đặt ra nhiều thách thức mới, đặc biệt là do tính đa chủng của các thiết bị và các mạng tham gia.

Đặc trưng của mạng Internet là cấu trúc quản lý phi tập trung (một số người còn cho rằng không hề được quản lý), thiếu tính an ninh, và nhiều dịch vụ phân tán được cung cấp bởi người dùng và các công ty có kết nối vào Internet. Tuy nhiên, đặc trưng chính của mạng Internet là tất cả các máy tính có kết nối vào nó đều dùng cùng một bộ giao thức (Internet Protocol - IP) và *giao thức TCP/IP hiện đã được hầu hết mọi hệ điều hành cung cấp.*

CÂU HỎI VÀ BÀI TẬP

1.1. Mạng máy tính là gì?

1.2. Dữ liệu và thông tin là gì?

1.3. Trình bày các kiểu kiến trúc vật lý của các mạng máy tính.

1.4. Trình bày các kiểu mạng theo tầm địa lý.

1.5. Trình bày các chuẩn giao thức ISO/OSI và TCP/IP.

1.6. Trình bày các dịch vụ trên mạng giải rộng.

CHƯƠNG 2

CƠ SỞ DỮ LIỆU QUAN HỆ

2.1. MỞ ĐẦU

Mô hình cơ sở dữ liệu quan hệ là một mô hình được sử dụng rộng rãi trong đời sống xã hội của mọi tổ chức, cơ quan, xí nghiệp, doanh nghiệp, nơi nào cần quản lý và xử lý các thông tin. Ta xét một vài ví dụ minh hoạ:

Thí dụ 2.1:

Xét hồ sơ cán bộ của một cơ quan:

TT	MS	TÊN	NS	TĐÔ	QUÊ	GT	LƯƠNG
1	01	Huy	1945	Đại học	Hànội	Nam	300
2	02	Tiến	1950	Cao học	Hảiphòng	Nam	400
3	03	Lan	1960	Trunghọc	Namhà	Nữ	200
4	04	Hiền	1965	Trunghọc	Hảidương	Nữ	250

(TT là thứ tự, MS: mã số, NS: năm sinh,TĐÔ: trình độ, GT: giới tính, . . .).

Thí dụ 2.2:

Xét sổ theo dõi khách của một khách sạn:

MK	ĐẾN	ĐI	NR	SỐNGƯỜI	TIỀN
101	1/10/98	5/10/98	301	2	400
102	5/10/98	20/11/98	302	1	200
103	7/10/98	10/7/98	303	3	600
104	5/12/98	10/12/98	304	2	400
105	15/1/99	10/1/99	304	3	600

Trong đó MK: mã khách, NR: phòng số, TIỀN: tiền thuê phòng. . .

Trong hai thí dụ trên tuy để quản lý các mảng thông tin (dữ liệu) khác nhau nhưng cả hai đều có chung một đặc thù: dữ liệu được mô tả dưới dạng *bảng*, mỗi bảng có một dòng đầu là dòng *thuộc tính*.

Trong thí dụ 2.1 các *thuộc tính* là TT, MS, TÊN, NS, TĐÔ, QUÊ, GT, LƯƠNG.

Trong thí dụ 2.2 *tập thuộc tính* là: {MK, ĐẾN, ĐI, NR, SỐNGƯỜI TIỀN}.

Mỗi thuộc tính có một miền giá trị của nó, thí dụ, thuộc tính năm sinh NS có miền giá trị là các số nguyên: 1945, 1950, 1960, 1965,...thuộc tính TÊN có miền giá trị là các xâu ký tự (string): Minh, Tiến, Lan, Hiền,...

Trong mỗi thí dụ ở trên mỗi bảng đều có một số phần tử, thí dụ, bảng hồ sơ nhân sự của cơ quan có bốn phần tử, bảng theo dõi khách ở khách sạn có năm phần tử, mỗi phần tử là một dòng. Về sau các mảng dữ liệu được mô tả dưới dạng bảng như vậy sẽ được gọi là các *quan hệ*.

Sau đây chúng ta sẽ định nghĩa (mô hình hóa) chính xác mô hình CSDL quan hệ.

2.2. ĐỊNH NGHĨA QUAN HỆ

Cho tập hữu hạn các phần tử $R = \{A_1, A_2, ... A_n\}$- được gọi là tập các *thuộc tính*. Mỗi phần tử A_i của tập R có *miền giá trị (miền trị)* $D(A_i)$.

Mỗi tập con của tích Descartes (Decac) của các miền giá trị $D(A_i)$ với $i = 1, 2, 3,... , n$ được gọi là một *quan hệ* trên R. Về sau ta thường ký hiệu r là quan hệ trên R. Vậy r là *quan hệ* trên tập thuộc tính R nếu:

$r \subset D(A_1) \times D(A_2) \times ... \times D(A_n)$ trong đó $D(A_i)$ là miền giá trị của thuộc tính A_i.

Từ định nghĩa ta cần lưu ý rằng tích Decac $D(A_1) \times D(A_2) ... \times D(A_n)$ có rất nhiều tập con nên trên R ta có nhiều quan hệ khác nhau.

Thí dụ: Giả sử R= {A, B, C }, D(A) = {0,1}, D(B) = {a, b, c}, D(C) ={x,y}.

Tích Decac D(A) × D(B) × D(C) = {(0,a,x), (0,a,y), (0,b,x), (0,b,y), (0,c,x), (0,c,y), (1,a,x), (1,a,y), (1,b,x), (1,b,y), (1,c,x), (1,c,y)}. Như vậy tích D(A) × D(B) × D(C) có 12 phần tử và nó có 2^{12} tập con khác nhau nên trên R = {A,B,C} ta có 2^{12} quan hệ r khác nhau. Thí dụ $r_0 = \{\varnothing\}$ là quan hệ rỗng, r_1= {(0,a,x)}, r_1'= {(0,b,x)} là các quan hệ chứa 1 phần tử, quan hệ r_2= {(0,a,x), (0,b,x) } là quan hệ chứa 2 phần tử... còn quan hệ r = {(0,a,x), (0,a,y), (0,b,x), (0,b,y), (0,c,x), (0,c,y), (1,a,x), (1,a,y), (1,b,x), (1,b,y), (1,c,x), (1,c,y)} là quan hệ chứa 12 phần tử. Qua ví dụ ta thấy mỗi phần tử của quan hệ r là một bộ của tích Decac D(A) × D(B) × D(C).

Đến đây chúng ta lưu ý rằng định nghĩa quan hệ r là tập con của tích Decac D(A) × D(B) × D(C) đã được xét trong toán cơ sở . Trong CSDL quan hệ để cho tiện và dễ hình dung với các bài toán quản lý ta viết mỗi quan hệ r trên R dưới dạng bảng. Dòng đầu của bảng là dòng các thuộc tính, các dòng sau của bảng là các bộ của quan hệ. Thí dụ với quan hệ r_0 không chứa phần tử nào ta viết:

r_0

A	B	C

Hoặc các quan hệ chứa 1 phần tử r_1, r_1' được viết:

r_1		
A	B	C
0	a	x

r_1'		
A	B	C
0	b	x

Hay quan hệ chứa 2 phần tử r_2 ta viết:

r_2		
A	B	C
0	a	x
0	b	x

Tương tự cho những quan hệ khác trên R thí dụ quan hệ chứa 12 phần tử sau dòng thuộc tính ta có 12 dòng, mỗi dòng là một bộ của r.

Một cách tổng quát từ định nghĩa ta thấy nếu cho trước tập thuộc tính R = { A_1, A_2, ... A_n } thì quan hệ r là một bảng hai chiều, trên cột thứ i là các giá trị của $D(A_i)$, trên mỗi dòng của bảng là bộ n giá trị của các miền giá trị của các thuộc tính A_1, A_2,... A_n. Mỗi dòng là một phần tử của quan hệ.

Để ký hiệu một quan hệ nào đó trên tập thuộc tính R = {A_1, A_2, . . . A_n}ta dùng r hoặc nếu không gây nhầm lẫn đôi khi ta viết R(A_1, A_2, . . . A_n).

Ta quay lại thí dụ 2. 1 bảng lưu trữ hồ sơ nhân sự của cơ quan là một quan hệ. Với R = {TT, MS, TÊN, NS, TĐÔ, QUÊ, GT, LƯƠNG}.

D(TT) = {1, 2, 3, 4,... }

D(TÊN) = {Minh, Tiến, Lan, Hiền,... }

D(MS) = {01, 02, 03, 04,... }

D(NS) = {1945, 1950, 1960,1965,... }

D(TĐÔ) = {đạihọc, caohọc, trunghọc,... }

D(QUÊ) = {Hànội, Hảiphòng, Namhà, Hảidương,... }

D(GT) = {nam, nữ}

D(LƯƠNG) = {300, 400, 200, 250,... }

Miền giá trị của thuộc tính GT chỉ có hai giá trị đó là: nam, nữ ; miền giá trị của thuộc tính TT phụ thuộc vào số nhân sự của cơ quan,... Bốn dòng trong bảng ở thí dụ 2. 1 tạo một quan hệ r trên R và quan hệ r có bốn phần tử.

Từ định nghĩa và trực quan của quan hệ r trên R ta thấy rằng khi cho tập thuộc tính R = {A_1, A_2,... A_n} là ta đã cho mẫu, khuôn hoặc lược đồ của quan hệ, và thú vị hơn ta có thể coi như đã cho một quan hệ, chỉ có điều đó là quan hệ rỗng. Vậy khi nói cho tập thuộc tính R = {A_1,... . , A_n} ta coi như cho trước *lược đồ quan hệ* (LĐQH) và cùng với nó ta có quan hệ r = ∅.

Chúng ta cần lưu ý rằng, ký hiệu R(A_1,A_2,. . .A_n) hàm chứa một quan hệ trên R và cho chúng ta biết khung, lược đồ quan hệ R.

Từ định nghĩa ta thấy trên một LĐQH R ={A_1,... , A_n} ta có thể xây dựng được nhiều quan hệ khác nhau, cứ thay đổi giá trị của một dòng hoặc một cột ta được một quan hệ mới. Tuy nhiên chúng ta cần nhớ rằng với cách nhìn của tập hợp thì việc thêm vào một dòng giống với dòng đã có thì quan hệ không thay đổi và trong lý thuyết CSDL ta coi *hai dòng giống nhau đó là một*. Tương tự cho các cột trong một quan hệ *hai cột giống hệt nhau ta coi là một*. Đồng thời với cách biểu diễn quan hệ như một bảng thì thứ tự trước sau của các dòng (cột) không làm thay đổi quan hệ. Vậy tập thuộc tính R = {A_1,... , A_n} đại diện cho các quan hệ trên nó, nên đôi khi thay cho việc nói cho LĐQH R và r là quan hệ trên R ta có thể nói cho quan hệ R(A_1,... , A_n).

Thí dụ 2.3:

Ta xét CSDL quản lý lương của cán bộ.

Cho LĐQH R = {MA, HỌTÊN, ĐƠNVỊ, NS, LƯƠNG, PHỤCẤP, THƯỞNG} và quan hệ r như sau:

MA	HỌTÊN	ĐƠNVỊ	NS	LƯƠNG	PHỤCẤP	THƯỞNG
01	Minh	G1	1965	400	50	50
02	Đông	G1	1946	800	100	100
03	Long	HC	1954	1000	100	100
04	Kiên	K1	1957	600	50	50
05	Đại	G2	1945	1000	200	100

Quan hệ r trong trường hợp này có năm phần tử.

Về sau nếu không cần quan tâm đến bản chất nội tại của mô hình quan hệ, đôi khi để cho tiện ta ký hiệu các thuộc tính bằng các chữ in hoa A, B, C, D,... ; các giá trị cụ thể của miền giá trị của chúng bằng các chữ thường a, b, c,... tương ứng, còn các phần tử của các quan hệ là t, t', ...

Thí dụ 2.4:

Cho quan hệ 5 phần tử r như sau:

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_2	d_2	e_2	f_2
a_3	b_3	c_3	d_3	e_3	f_3
a_4	b_4	c_4	d_4	e_4	f_4
a_5	b_5	c_5	d_5	e_5	f_5

Qua các thí dụ ở trên ta có nhận xét tập các thuộc tính gồm các phần tử khác nhau nhưng miền giá trị của các thuộc tính không nhất thiết phải khác nhau, trong thí dụ 2. 3 các thuộc tính LƯƠNG, PHUCÂP, THƯƠNG đều có miền giá trị là các số nguyên (hoặc thực).

Sau đây chúng ta sẽ xét một số quan hệ của một CSDL mẫu để mô hình hoá cho một công ty máy tính. Các bộ phận (thực thể) chính của công ty là: Nhân viên (EMP-employee) và các Dự án (PROJ-project). Như vậy CSDL của công ty máy tính có hai quan hệ chính là quan hệ EMP (đây là quan hệ chứa các thông tin về các nhân viên như mã nhân viên (ENO), tên nhân viên (ENAME), chức vụ nhân viên (TTLE), lương nhân viên (SAL), dự án nhân viên tham gia (PNO), trách nhiệm của nhân viên trong dự án (RESP-responsibility) và thời gian tham gia dự án của nhân viên (DUR)) và quan hệ PROJ (quan hệ này lưu các thông tin về các dự án như mã dự án (PNO), tên dự án (PNAME) và kinh phí dự án (BUDGET)).

Thí dụ 2.5:

EMP

ENO	ENAME	TITLE	SAL	PNO	RESP	DUR
E1	J.Doe	Elec.Eng	40000	P1	Manager	12
E2	M.Smith	Analist	34000	P1	Analist	24
E2	M.Smith	Analist	34000	P2	Analist	6
E3	A.Lee	Mech.Eng	27000	P3	Consulant	10
E3	A.Lee	Mech.Eng	27000	P4	Engineer	48
E4	J.Miller	Programer	24000	P2	Programer	18
E5	B.Casey	Syst.Analist	34000	P2	Manager	24
E6	L.Chu	Elec.Eng	40000	P4	Manager	48
E7	R.David	Mech.Eng	27000	P3	Engineer	36
E8	J.Jone	Syst.Analist	34000	P3	Maniger	40

PROJ

PNO	PNAME	BUDGET
P1	instrumentation	150000
P2	Database develop.	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

2.3. CÁC PHÉP TOÁN ĐẠI SỐ CÁC QUAN HỆ

2.3.1. PHÉP HỢP

Ta nói hai quan hệ r_1 và r_2 là tương thích nếu chúng có cùng tập thuộc tính R.

Hợp của hai quan hệ tương thích r_1 và r_2 ký hiệu $r_1 + r_2$ là một quan hệ trên R gồm các phần tử thuộc r_1 hoặc r_2. Tức là: $r_1 + r_2 = \{t: t \in r_1$ hoặc $t \in r_2\}$.

Thí dụ 2.6:

Cho hai quan hệ r_1 và r_2 như sau:

Quan hệ r_1: Quan hệ r_2:

r1

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3
a_4	b_4	c_4	d_4

r2

A	B	C	D
x_1	y_1	z_1	v_1
x_2	y_2	z_2	v_2
x_3	y_3	z_3	v_3

Khi đó ta có quan hệ $r_1 + r_2$:

$r_1 + r_2$

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3
a_4	b_4	c_4	d_4
x_1	y_1	z_1	v_1
x_2	y_2	z_2	v_2
x_3	y_3	z_3	v_3

Quan hệ $r_1 + r_2$ có bảy phần tử. Chúng ta chú ý rằng thứ tự trước sau của các phần tử (các dòng) trong các quan hệ là như nhau. Từ định nghĩa ta thấy ngay rằng:

$\forall r_1, r_2$ thì $r_1 + r_2 = r_2 + r_1$

$\forall r$ thì $r + r = r$

Một cách tổng quát có thể lấy hợp của n quan hệ tương thích. Cho m quan hệ tương thích $r_1, r_2, r_3, \ldots r_m$.

Hợp của các quan hệ $r_1, r_2, r_3, \ldots r_m$ là một quan hệ $r_1 + r_2 + \ldots + r_m$ gồm các phần tử thuộc r_1 hoặc r_2 hoặc r_3 hoặc... r_m.

Vậy $r_1 + r_2 + \ldots + r_m = \{t: t \in r_1$ hoặc $t \in r_2 \ldots$ hoặc $t \in r_m\}$

2.3.2 PHÉP GIAO

Cho LĐQH $R = \{A_1, A_2, \ldots, A_n\}$. Cho hai quan hệ tương thích r_1 và r_2 trên R. Giao của hai quan hệ r_1 và r_2 ký hiệu: $r_1 * r_2$ là một quan hệ trên R gồm các phần tử chung của r_1 và r_2.

Vậy: $r_1 * r_2 = \{t: t \in r_1$ và $t \in r_2\}$.

Thí dụ 2.7:

Cho hai quan hệ r_1 và r_2:

r_1

A	B	C	D
a_1	b_1	c_1	d_1
a	b	c	d
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3

r_2

A	B	C	D
a	b	c	d
a_2	b_2	c_2	d_2
x	y	z	v

Khi đó ta có quan hệ giao:

$r_1 * r_2$

A	B	C	D
a	b	c	d
a_2	b_2	c_2	d_2

2.3.3 PHÉP TRỪ (HIỆU)

Cho hai quan hệ r_1 và r_2 tương thích, có tập thuộc tính R. Hiệu của r_1 và r_2 ký hiệu: $r_1 - r_2$ là một quan hệ trên R gồm các phần tử thuộc r_1 và không thuộc r_2. Vậy $r_1 - r_2 = \{t: t \in r_1 \text{ và } t \notin r_2\}$.

Nếu lấy r_1 và r_2 như trong thí dụ 2. 7 ta có:

$r_1 - r_2$:

A	B	C	D
a_1	b_1	c_1	d_1
a_3	b_3	c_3	d_3

$r_2 - r_1$:

A	B	C	D
x	y	z	v

2.3.4 PHÉP CHIẾU

Cho LĐQH R = {A_1, A_2, A_3,... A_n}. Cho r là một quan hệ trên R, X là một tập con của R tức X ⊂ R, ta gọi X là lược đồ con của lược đồ R. Ta xét quan hệ con của quan hệ r chỉ trên tập thuộc tính X, đó là chiếu của r lên X.

Chiếu của r lên tập thuộc tính X là một quan hệ trên lược đồ X ký hiệu r. X gồm các phần tử của r sau khi đã lược bỏ các thuộc tính không thuộc tập X. Tương tự với r. X, các phần tử của r. X là những phần tử ký hiệu là t. X, chính là chiếu của t lên X. Vậy r. X = {t. X: t ∈ r}, t. X là chiếu của phần tử t lên tập thuộc tính X.

Trực quan của r.X là trong bảng quan hệ r ta bỏ các cột với các thuộc tính không thuộc X, bảng còn lai là r.X.

Thí dụ 2.8:

Cho quan hệ r như sau:

r

A	B	C	D	E	F	G
a_1	b_1	c_1	d_1	e_1	f_1	g_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2
a_3	b_3	c_3	d_3	e_3	f_3	g_3
a_4	b_4	c_4	d_4	e_4	f_4	g_4

Giả sử ta có X = {A, B, C}, Y = {F, G}.

Khi đó ta có hai quan hệ con chiếu của r lên X và Y tương ứng:

r. X			r. Y	
A	B	C	F	G
a_1	b_1	c_1	f_1	g_1
a_2	b_2	c_2	f_2	g_2
a_3	b_3	c_3	f_3	g_3
a_4	b_4	c_4	f_4	g_4

Quay lại thí dụ 2.3 CSDL lương cán bộ, ta giả sử X = {MA, HOTEN, THƯƠNG}. Khi đó ta có chiếu của r lên X là quan hệ r. X

r. X		
MA	HOTEN	THƯƠNG
01	Minh	50
02	Đông	100
03	Long	100
04	Kiên	50
05	Đại	100

2.3.5 PHÉP TÍCH DECAC (DESCARTES)

Tích Decac của hai quan hệ ta chỉ xét trên các lược đồ rời nhau. Cho hai lược đồ:

$$R_1 = \{A_1, A_2, \ldots A_n\}$$

$$R_2 = \{B_1, B_2, \ldots B_m\}$$

Với $R_1 \cap R_2 = \varnothing$.

Giả sử r_1, r_2 là hai quan hệ trên R_1 và R_2 tương ứng.

Tích Decac của r_1 và r_2 ký hiệu: $r_1 \times r_2$ là quan hệ trên lược đồ $R_1 \cup R_2$ gồm các phần tử tạo ra từ tích Decac của hai tập r_1 và r_2.

Vậy quan hệ $r_1 \times r_2$ là quan hệ trên lược đồ :

$R = R_1 \cup R_2 = \{A_1, A_2, \ldots A_n, B_1, B_2, \ldots B_m\}$ với

$r_1 \times r_2 = \{< t_1, t_2 >: t_1 \in r_1, t_2 \in r_2\}$.

Thí dụ 2.9:

Cho r_1 và r_2 như sau:

		r_1				r_2
A	B	C	D	E	F	G
a_1	b_1	c_1	d_1	e_1	f_1	g_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2
				e_3	f_3	g_3

Như vậy r_1 có hai phần tử, r_2 có ba phần tử tích Decac $r_1 \times r_2$ sẽ có sáu phần tử.

$r_1 \times r_2$

A	B	C	D	E	F	G
a_1	b_1	c_1	d_1	e_1	f_1	g_1
a_1	b_1	c_1	d_1	e_2	f_2	g_2
a_1	b_1	c_1	d_1	e_3	f_3	g_3
a_2	b_2	c_2	d_2	e_1	f_1	g_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2
a_2	b_2	c_2	d_2	e_3	f_3	g_3

2.3.6 PHÉP NỐI TỰ NHIÊN

Cho hai lược đồ quan hệ R_1 và R_2, r_1 và r_2 là hai quan hệ tương ứng trên R_1 và R_2.

Phép nối (nối tự nhiên) của r_1 và r_2 ký hiệu: $r_1 |><| r_2$ là quan hệ trên lược đồ $R_1 \cup R_2$ gồm các phần tử t mà chiếu của t lên R_1 là phần tử thuộc r_1 còn chiếu của t lên R_2 là phần tử của r_2.

Vậy $r_1 |><| r_2 = \{t : \quad t.R_1 \in r_1$ và $t.R_2 \in r_2\}$

Thí dụ 2.10:

Cho r_1 và r_2 là hai quan hệ sau:

r_1				r_2			
A	B	C	D	C	D	E	F
a_1	b_1	c_1	d_1	c_1	d_1	e_1	f_1
a_2	b_2	c_2	d_2	c_2	d_2	e_2	f_2
a_3	b_3	c_3	d_3	x	y	z	v
1	2	3	4				

Quan hệ nối của r_1 và r_2:

$r_1 |><| r_2$

A	B	C	D	E	F
a_1	b_1	c_1	d_1	e_1	f_1
a_2	b_2	c_2	d_2	e_2	f_2

Thí dụ 2.11:

Xét hai quan hệ cùng tập thuộc tính (tương thích) sau:

r_1					r_2				
A	B	C	D	E	A	B	C	D	E
a_1	B_1	c_1	d_1	e_1	x_1	y_1	z_1	w_1	v_1
a_2	B_2	c_2	d_2	e_2	a_1	b_1	c_1	d_1	e_1
a_3	B_3	c_3	d_3	c_3	x_2	y_2	z_2	w_2	v_2

Ta có:

$r_1 \mid >< \mid r_2$

A	B	C	D	E
a_1	b_1	c_1	d_1	c_1
a_2	b_2	c_2	d_2	e_2
a_3	b_3	c_3	d_3	e_3
a_4	b_4	c_4	d_4	e_4
a_5	b_5	c_5	d_5	e_5
a_6	b_6	c_6	d_6	e_6

Vậy trong trường hợp hai tập thuộc tính như nhau thì $r_1 \mid >< \mid r_2 = r_1 * r_2$

Sau đây ta xét thí dụ mà các tập thuộc tính rời nhau.

Cho hai quan hệ r_1 và r_2 như sau:

r_1

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	d_2	e_2
a_3	b_3	c_3	d_3	e_3

r_2

F	G	H
f_1	g_1	h_1
f_2	g_2	h_2
f_3	g_3	h_3
x	y	z

Trong trường hợp này ta có $r_1 \mid >< \mid r_3$ như sau:

$r_1 \mid >< \mid r_3$

A	B	C	D	E	F	G	H
a_1	b_1	c_1	d_1	e_1	f_1	g_1	h_1
a_1	b_1	c_1	d_1	e_1	f_2	g_2	h_2
a_1	b_1	c_1	d_1	e_1	f_3	g_3	h_3
a_1	b_1	c_1	d_1	e_1	x	y	z
a_2	b_2	c_2	d_2	e_2	f_1	g_1	h_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2	h_2
a_2	b_2	c_2	d_2	e_2	f_3	g_3	h_3
a_2	b_2	c_2	d_2	e_2	x	y	z
a_3	b_3	c_3	d_3	e_3	f_1	g_1	h_1
a_3	b_3	c_3	d_3	e_3	f_2	g_2	h_2
a_3	b_3	c_3	d_3	e_3	f_3	g_3	h_3
a_3	b_3	c_3	d_3	e_3	x	y	z

Vậy trong trường hợp $R_1 \cap R_2 = \varnothing$ thì $r_1 \mid >< \mid r_2 = r_1 \times r_2$. Nói cách khác khi hai tập thuộc tính rời nhau phép nối chính là tích Decac.

Chúng ta cần lưu ý rằng các phép toán tích Decac và phép nối nói chung làm tăng dữ liệu của các quan hệ. Trong thực tiễn khi sử dụng các phép toán trên vào bài toán cụ thể, dựa vào ngôn ngữ sử dụng chúng ta đặt thêm điều kiện để tránh làm cồng kềnh dữ liệu, làm tốn bộ nhớ và dễ đưa đến nhầm lẫn. Khi thực hiện tách, ghép các CSDL ta nên dùng các phép toán đơn giản để tránh sai sót không đáng có.

Tuy nhiên trong hầu hết các bài toán quản lý việc dùng phép nối là không thể thiếu được nếu không nói là rất nhiều.

2.3.7 PHÉP CHIA

Cho lược đồ quan hệ $R = \{A_1, A_2, ... A_n\}$. S là lược đồ con của R tức là $S \subset R$. Giả sử r và s là các quan hệ trên R và S tương ứng.

Phép chia của quan hệ r cho quan hệ s ký hiệu: $r \div s$ là quan hệ trên lược đồ R - S gồm các phần tử t sao cho tồn tại phần tử $u \in s$ và ghép t với u ta được phần tử thuộc r :

Vậy $r \div s = \{t: \exists u \in s \text{ và } < t, u > \in r\}$.

Trong định nghĩa trên ta chú ý rằng ký hiệu < t, u > là sự ghép vào đúng vị trí của hai bộ t và u. Các bạn sẽ thấy rõ trong một vài thí dụ sau.

Thí dụ 2.12:

Cho r là quan hệ trên LĐQH R = {A, B, C, D, E, G}, s là quan hệ trên LĐQH S = {A, E, G} như sau:

r

A	B	C	D	E	G
a_1	b_1	c_1	d_1	e_1	g_1
a_2	b_2	c_2	d_2	e_2	g_2
a_3	b_3	c_3	d_3	e_3	g_3
a_4	b_4	c_4	d_4	e_4	g_4

s

A	E	G
a_1	e_1	g_1
x	y	z
a_3	e_3	g_3

Khi đó

$r \div s$

B	C	D
b_1	c_1	d_1
b_3	c_3	d_3

Từ định nghĩa ta thấy để có thể thực hiện được phép chia r ÷ s thì S phải là lược đồ con thực sự của R. Tất nhiên nếu S rỗng thì r ÷ s = r. Nếu S bao R thực sự thì phép chia không thực hiện được. Còn R = S thì ta được quan hệ rỗng trên tập thuộc tính rỗng.

2.3.8 PHÉP CHỌN

Trong xử lý các CSDL dạng bảng (quan hệ) một phép toán ta thường dùng để xử lý dữ liệu đó là phép chọn. Phép chọn tức là chọn từ bảng quan hệ ra các phần tử thỏa mãn điều kiện nào đó. Trong xử lý CSDL hàng ngày ta luôn làm việc với các phép toán chọn. Thí dụ, khi làm báo cáo ta cần in ra những sinh viên khá giỏi, ta chọn từ bảng quản lý sinh viên các sinh viên (các phần tử của quan hệ) đạt điểm khá giỏi, hoặc ta cần phải in danh sách số cán bộ đến tuổi nghỉ hưu của một cơ quan nào đó. Tất cả đều là phép toán chọn.

Ta sẽ định nghĩa phép chọn trên các quan hệ như sau: Cho quan hệ r trên LĐQH R. Cho E là mệnh đề lôgic. Phần tử t thuộc r thỏa mãn điều kiện E ta ký hiệu t(E). Phép chọn từ quan hệ r theo điều kiện E cho ta một quan hệ ký hiệu r(E) trên đúng lược đồ R và chứa các phần tử của r thỏa mãn điều kiện E.

Vậy r(E) = {t: t ∈ r và t(E)}.

Chú ý: Trong giáo trình này và một số cuốn sách khác đôi khi để biểu thị một phép chọn theo mệnh đề E hoặc công thức E ta sẽ ký hiệu tổng quát là $\sigma_E(r)$ trong đó r là quan hệ và E là điều kiện chọn.

Thí dụ 2.13:

Xét hồ sơ kết quả thi của sinh viên.

Quan hệ này ta gọi là SV. Giả sử ta có quan hệ SV như sau:

TT	HOTEN	NAMSINH	ĐIÊMCSDL	ĐIÊMFOX
1	Tuấn Anh	1974	7	5
2	Huy Công	1974	8	3
3	Th. Hương	1975	8	9
4	Bình Minh	1976	2	3

Giả sử điều kiện E là sinh viên có ít nhất một điểm kém. Vậy r(E):

r(E)

TT	HOTEN	NAMSINH	ĐIÊMCDL	ĐIÊMFOX
2	Huy Công	1974	8	3
4	Bình Minh	1976	2	3

2.3.9 PHÉP KẾT NỐI THEO θ

Như chúng ta đã trình bày trong phép nối, phép nối và tích Decac nói chung làm tăng dữ liệu, trong nhiều trường hợp ta thêm điều kiện để có được dữ liệu như mong muốn.

Chúng ta sẽ xét phép kết nối theo toán tử θ, với θ là một toán tử so sánh số học hai ngôi (= , < , > , ≤ , ≥ , ≠) , ví dụ nếu A , B là các thuộc tính thì kết quả của các phép toán A = B , A > B là các bộ có giá trị của A bằng B hoặc A lớn hơn B tương ứng.

Cho r và s là hai quan hệ tương ứng trên các lược đồ rời nhau R và S, tức $R \cap S = \emptyset$.

Phép kết nối θ của các quan hệ r và s, ký hiệu $r |><|_{i\theta j} s$, là một quan hệ trên lược đồ $R \cup S$ gồm những bộ thuộc tích Decac của r và s sao cho thành

phần thứ i của quan hệ r thoả mãn phép toán θ với thành phần thứ j của quan hệ s.

Vậy kết nối θ: r $|><|_{i\theta j}$s là chọn trong r × s các bộ mà các thành phần thứ i, j của các quan hệ r , s tương ứng thỏa mãn θ, tức là:

r $|><|_{i\theta j}$s = {t ∈ r × s: t (iθj)}.

Đây là phép kết nối gắn với tích Decac.

Thí dụ 2.14:

Giả sử r và s là các quan hệ như sau:

r

A	B	C
1	2	3
4	5	6
7	8	9

s

D	E
3	1
6	2

Khi đó ta có: r $|><|_{2<1}$s

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

Thí dụ 2.15:

Giả sử r và s là các quan hệ:

r

A	B	C
1	2	3

s

D	E	F
1	e	f

	a	b	c		a	e	f
	x	y	z		5	6	7

Khi đó

$r \mid><\mid_{1=1} s$

A	B	C	D	E	F
1	2	3	1	e	f
a	b	c	a	e	f

2.3.10 PHÉP NỐI NỬA

Sau đây ta sẽ xét phép nối nửa (semijoin) gắn với phép nối.

Cho các quan hệ r và s trên các lược đồ R và S tương ứng.

Nối nửa của các quan hệ r và s , ký hiệu r |>< s là một quan hệ trên lược đồ R gồm các bộ của r |><| s chiếu lên R , tức là

$r \mid>< s = \{ t: \quad t \in r \mid><\mid s. R \} = \{t.R : t \in r \mid><\mid s \}.$

Thí dụ 2.16:

Giả sử r và s là các quan hệ:

r

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

s

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

Khi đó ta có:

r |>< s

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

Chúng ta dễ dàng nhận thấy rằng:

1. Nối nửa là nối tự nhiên xong chiếu lên R.

2. Một cách tương đương để tính r |>< s, ta tính chiếu của s lên tập thuộc tính chung của R và S, R∩S rồi lấy nối tự nhiên của r với quan hệ thu được. Nói cách khác, các bạn có thể dễ dàng chứng minh được (xem phần Câu hỏi và bài tập):

r |>< s = r |><| (s. R∩S).

3. r |>< s ≠ s |>< r.

Trên đây là một số phép toán cơ bản thường gặp trong công tác quản lý các CSDL. Trong các phần sau ta xét thêm các phép tách.

Đó là phép toán ngược với phép hợp (phép tách ngang), ngược với phép nối (phép tách dọc). Để giáo trình thêm sáng sủa và nhất quán ta sẽ xét phép tách thành một phần riêng biệt.

2.4. PHỤ THUỘC HÀM

Trong các bảng quan hệ ở phần trước, chúng ta thấy giữa các thuộc tính của quan hệ có một số ràng buộc (phụ thuộc dữ liệu) đóng vai trò quan trọng trong lý thuyết CSDL.

Thí dụ, trong hồ sơ nhân sự của cán bộ (thí dụ 2.1) thuộc tính thứ tự TT có tính chất "quyết định " của bảng quan hệ, chẳng hạn nếu biết số thứ tự ta

có thể suy ra giá trị của các thuộc tính khác, ngược lại nếu biết lương hoặc trình độ chúng ta không thể suy tiếp giá trị của các thuộc tính khác vì có thể nhiều cán bộ có cùng trình độ và lương. Hay xét quan hệ tuyển sinh vào đại học, với các thuộc tính thứ tự (TT), tên (TÊN), năm sinh(NS), quê(QUÊ),số báo danh(SBD), điểm toán(ĐT), điểm lý(ĐL), điểm hoá(ĐH),. . .tức R= {TT, TÊN, NS, QUÊ, SBD, ĐT, ĐL, ĐH, . . .}, ta thấy thuộc tính số báo danh SBD quyết định duy nhất các giá trị của các thuộc tính khác, nói cách khác nếu biết SBD thì biết được giá trị của các các thuộc tính khác như điểm thi, quê quán, . . .tức SBD kéo theo các thuộc tính khác. Như vậy giữa các tập thuộc tính X, Y của lược đồ quan hệ R có những mối ràng buộc kiểu *kéo theo, xác định duy nhất.*

Để đi sâu hiểu rõ bản chất các mối ràng buộc đó, sau đây chúng ta sẽ xét khái niệm phụ thuộc hàm-functional dependence (ta thường viết tắt PTH).

2.4.1 ĐỊNH NGHĨA PHỤ THUỘC HÀM

Khái niệm phụ thuộc hàm là khái niệm quan trọng trong CSDL quan hệ. Trước khi đi vào định nghĩa cụ thể chúng ta cần lưu ý và cố gắng hình dung vấn đề sau: Giả sử ta xét lược đồ quan hệ tuyển sinh R(TT, TÊN, NS, QUÊ, SBD, ĐT, ĐL, ĐH, . . .), với quan hệ tổng quát được quản lý tại Bộ Đại học (quan hệ này chứa tất cả thí sinh dự thi đại học) và các quan hệ con cụ thể được quản lý tại các trường đại học với cùng một lược đồ R. Như vậy thuộc tính SBD có tính chất kéo theo các thuộc tính khác trên toàn bộ lược đồ, tức là trên tất cả các quan hệ. Hiện tượng này khẳng định một loại *ràng buộc trên toàn bộ lược đồ.* Nếu xét trong một quan hệ cụ thể, mặc dù hơi vô lý nhưng ta có thể giả sử một trường nào đó có 100 thí sinh tham gia thi với những tên (TÊN) khác nhau, như vậy ngoài thuộc tính SBD trong trường này

(coi như một quan hệ cụ thể trên R) thuộc tính TÊN cũng có tính chất nếu biết TÊN có thể biết giá trị của các thuộc tính khác, nói cách khác TÊN kéo theo các thuộc tính khác. Hiện tượng này khẳng định một loại *ràng buộc trên một quan hệ cụ thể*. Vậy ta sẽ xét hai khái niệm phụ thuộc hàm, đó là phụ thuộc hàm trên lược đồ R và phụ thuộc hàm trên một quan hệ r trên R.

2.4.1.1 Phụ thuộc hàm trên lược đồ R

Cho lược đồ quan hệ $R = \{A_1, A_2, \ldots, A_n\}$.

Giả sử X, Y là các tập con của R, tức là $X, Y \subset R$.

Ta nói *Y phụ thuộc hàm vào X (hoặc X xác định phụ thuộc hàm Y) trên lược đồ quan hệ R, ký hiệu* $X \rightarrow Y$, *nếu X xác định duy nhất Y*(nói cách khác nếu biết X ta suy ra được Y), một cách chính xác hơn là với mọi quan hệ r trên R mà $\forall$ t, t' $\in$ r nếu t và t' bằng nhau trên tập X thì chúng cũng bằng nhau trên tập Y, tức là :

$$X \rightarrow Y \Leftrightarrow \forall t, t' \in r \text{ nếu } t.X = t'.X \Rightarrow t.Y = t'.Y.$$

Sau này thay cho PTH $X \rightarrow Y$ trên R đôi khi ta nói có PTH f trên R, X là tập thuộc tính xác định (determinant), Y là tập thuộc tính phụ thuộc (dependent).

Trong thực tiễn chúng ta thấy rất nhiều lược đồ quan hệ R mà trên nó mọi quan hệ r đều thỏa mãn một phụ thuộc hàm nào đó. Chẳng hạn xét hồ sơ nhân sự của một nước thì số chứng minh thư là một thuộc tính luôn xác định duy nhất các thuộc tính khác. Hoặc nếu chúng ta xét hồ sơ sỹ quan thì số hiệu sỹ quan cũng là thuộc tính xác định các thuộc tính khác (nói cách khác nếu có hai sỹ quan có cùng số hiệu thì hai sỹ quan đó có cùng những

tham số khác, tức hai sỹ quan đó là một). Trong các bài toán quản lý người ta thường thêm thuộc tính mã của đối tượng ID và thuộc tính này luôn kéo theo các thuộc tính khác.

Như vậy khái niệm phụ thuộc hàm trên lược đồ quan hệ R khẳng định sự ràng buộc mang tính chất nội tại của các tập thuộc tính trong R. Đó là sự ràng buộc của các tập thuộc tính trong R xét với mọi quan hệ r trên R.

Thí dụ 2.17:

Trong các quan hệ r trên R ={TT, A, B, C} với thuộc tính thứ tự được lấy khác nhau trên các quan hệ trong tập các số tự nhiên thì ta có TT $\rightarrow$ {A, B, C} và hiển nhiên TT $\rightarrow$ R vì TT $\rightarrow$ TT.

Hoặc xét hồ sơ nhân sự của tất cả sỹ quan quân đội R = { TT, TÊN, NS , SHSQ, . . .} thì rõ ràng thuộc tính số hiệu sỹ quan kéo theo các thuộc tính khác SHSQ $\rightarrow$ R .

2.4.1.2 Phụ thuộc hàm trong quan hệ r

Ở trên chúng ta đã định nghĩa khái niệm phụ thuộc hàm trên lược đồ R. Sau đây là khái niệm phụ thuộc hàm trên một quan hệ r cụ thể của lược đồ R.

Cho lược đồ quan hệ R và X, Y là các tập con của R, r là một quan hệ trên R.

Ta nói *X xác định phụ thuộc hàm Y, ký hiệu* $X \rightarrow Y$, *trong r nếu với mọi t và t' của r mà t, t' bằng nhau trên tập X thì chúng cũng bằng nhau trên tập Y, tức là* $\forall t, t' \in r$ *nếu* $t.X = t'.X$ *thì* $t.Y = t'.Y$.

Như vậy chúng ta thấy phụ thuộc hàm trên quan hệ r là trường hợp riêng của phụ thuộc hàm trên lược đồ R. Phụ thuộc hàm trên lược đồ R là phụ hàm thỏa mãn mọi quan hệ r trên R còn phụ thuộc hàm trên quan hệ r chỉ đòi hỏi phụ thuộc hàm thỏa mãn một quan hệ r. Tất nhiên $X \rightarrow Y$ là PTH trên lược đồ R thì nó là PTH thỏa mãn mọi quan hệ r bất kỳ trên R. Chúng ta cần lưu ý rằng khái niệm phụ thuộc hàm trên một quan hệ r là khái niệm rất hẹp, nó chỉ đúng cho một quan hệ, chúng ta chỉ cần thay đổi một vài giá trị của các thuộc tính trong quan hệ r là PTH có thể bị biến mất.

Thí dụ đơn giản chẳng hạn xét quan hệ r như sau:

r

A	B	C	D
0	0	0	0
1	1	1	1
0	1	0	1

Rõ ràng trong r thì $A \rightarrow C$ (vì các bộ bằng nhau trong A cũng bằng nhau trong C), tuy nhiên chúng ta chỉ cần thay đổi giá trị của thuộc tính C ở dòng đầu hoặc dòng 3 thì ta vẫn được một quan hệ r' trên R nhưng phụ thuộc hàm $A \rightarrow C$ không còn thỏa mãn trong r'.

2.4.2 CÁC TÍNH CHẤT CỦA PHỤ THUỘC HÀM

Các tính chất của phụ thuộc hàm ta xét trong lược đồ R. Nếu X, Y, Z và W là những tập thuộc tính con của R thì ta có một số tính chất cơ bản của lớp các PTH như sau *(để tiện khi trình bày thay cho tập {A, B, C} về sau ta sẽ viết ABC)*:

A1. Tính phản xạ: $X \rightarrow X$, tổng quát hơn nếu $Y \subset X$ thì $X \rightarrow Y$

A2. Tính bắc cầu: $X \rightarrow Y$ và $Y \rightarrow Z \Rightarrow X \rightarrow Z$.

A3. Tính mở rộng hai vế: $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$. (mở rộng hai vế Z)

4. Tính tựa bắc cầu: $X \rightarrow Y$ và $Y Z \rightarrow W \Rightarrow XZ \rightarrow W$.

5. Tính mở rộng trái và thu hẹp phải: $X \rightarrow Y \Rightarrow XZ \rightarrow Y - W$.

6. Tính cộng đầy đủ: $X \rightarrow Y$ và $Z \rightarrow W \Rightarrow XZ \rightarrow YW$.

7. Tính tích lũy: $X \rightarrow Y$ và $Y \rightarrow ZW \Rightarrow X \rightarrow YZW$.

Có thể chứng minh các tính chất A1, A2, A3, 4, 5, 6, 7 một cách đơn giản. Giả sử t, t' $\in$ r và r là một quan hệ bất kỳ trên R. Chúng ta lần lượt chứng minh các tính chất trên một cách dễ dàng. Thật vậy:

Tính phản xạ: Điều này hiển nhiên vì t và t' đã bằng nhau trong tập X thì chúng phải bằng nhau trong mọi tập con của X, nói cách khác
$t.X = t'.X \Rightarrow t.X = t'.X$ và $t.Y = t'.Y$ với mọi $Y \subset X$.

Tính bắc cầu: Giả sử t. X=t'.X theo giả thiết $X \rightarrow Y$ nên ta có t.Y=t'.Y mà t.Y = t'.Y theo giả thiết $Y \rightarrow Z$ ta lại có t.Z = t'.Z. Như vậy từ t.X = t'.X ta đã suy ra được t.Z = t'.Z, nên ta có $X \rightarrow Z$.

Tính mở rộng hai vế: Giả sử t.XZ = t'.XZ ta phải chứng minh
t.YZ = t'.YZ
Từ t.XZ = t'.XZ ta có t.X = t'.X và t.Z = t'.Z. Theo giả thiết
t.X = t'.X thì t.Y = t'.Y. Như vậy từ t.XZ = t'.XZ ta có t.Y = t'.Y và
t.Z = t'.Z mà t.Y = t'.Y và t.Z = t'.Z thì t.YZ = t'.YZ. Vậy $XZ \rightarrow YZ$.

Các tính chất khác có thể chứng minh tương tự. Tuy nhiên ta thấy rằng các tính chất 4, 5, 6, 7 đều có thể suy ra từ các tính chất A1, A2, A3. Trong lý thuyết CSDL, ba tính chất A1, A2, A3 gọi là *hệ tiên đề Armstrong*. (Armstrong là người đầu tiên nêu ba tính chất A1,A2, A3 của các phụ thuộc hàm).

2.4.3. HỆ TIÊN ĐỀ ARMSTRONG VÀ CÁC PHÉP SUY DẪN

Hệ A bao gồm ba tính chất {A1, A2, A3} đã nêu trong phần các tính chất của PTH ở trên được gọi là *hệ tiên đề Armstrong* của lớp các PTH FD (FD ký hiệu là lớp tất cả các PTH trên R).

Vậy A = {A1, A2, A3} là hệ tiên đề Armstrong.

2.4.3.1. Phép suy dẫn theo hệ tiên đề

Ta thấy hệ tiên đề Armstrong đóng vai trò sinh (generate) của lớp các PTH.

Thật vậy nếu cho trước tập PTH F trên lược đồ R thì ta có thể dùng luật suy dẫn trong các tính chất của PTH (trong đó có cả các tiên đề) để nhận được các PTH mới, lớp các PTH nhận được từ các phép suy dẫn như vậy đóng vai trò quan trọng trong lớp các PTH trên lược đồ quan hệ R. Ta sẽ lần lượt trình bày các vấn đề này trong các phần sau.

Mục này chúng ta chú ý rằng: Các tính chất (thực chất là các PTH) 4, 5, 6, 7 đều được suy dẫn (suy ra) từ hệ tiên đề Armstrong. Thật vậy:

Tính tựa bắc cầu (4) có thể suy ra từ tính mở rộng hai vế và tính bắc cầu vì từ giả thiết $X \rightarrow Y$ mở rộng hai vế Z ta có $XZ \rightarrow YZ$ và vì $YZ \rightarrow W$ theo bắc cầu ta có $XZ \rightarrow W$

Tính mở rộng trái và thu hẹp phải (5) được suy từ tính phản xạ và bắc cầu vì với mọi X, Y, Z, W ta có $XZ \rightarrow X$ và $Y \rightarrow Y - W$ (phản xạ) và từ giả thiết $X \rightarrow Y$ theo tính bắc cầu ta có $XZ \rightarrow Y - W$.

Tính cộng đầy đủ (6) được suy dẫn từ tính bắc cầu và tính mở rộng hai vế, thật vậy từ $X \rightarrow Y$ ta có $XZ \rightarrow YZ$ (mở rộng hai vế lên Z) cũng theo tính chất mở rộng hai vế ta lại có $YZ \rightarrow YW$ (mở rộng hai vế Y)và theo tính bắc cầu ta có $XZ \rightarrow YW$.

Tương tự *tính tích lũy (7)* cũng có thể được suy ra từ tính bắc cầu và tính mở rộng hai vế. Vì $Y \rightarrow ZW$ nên theo tính cộng hai vế ta có $YY \rightarrow YZW$, tức $Y \rightarrow YZW$ và vì $X \rightarrow Y$ nên theo tính bắc cầu ta có $X \rightarrow YZW$.

Như vậy ta thấy rằng mọi PTH f được suy ra từ bảy tính chất của PTH đều có thể được suy ra từ chỉ ba tính chất của hệ tiên đề Armstrong. Từ nay về sau thay cho việc nói PTH f nhận được từ tập PTH F dựa vào các luật suy dẫn trong bảy tính chất của PTH ta sẽ nói: *f được suy dẫn từ F theo hệ tiên đề Armstrong* (suy dẫn theo hệ tiên đề).

Vậy giả sử F là tập các PTH và f là một PTH trên R. Ta nói PTH *f suy dẫn được theo hệ tiên đề Armstrong* từ tập PTH F, ký hiệu $F \models f$, nếu f có thể nhận được từ F sau một số hữu hạn bước áp dụng các luật A1, A2, A3 của hệ tiên đề Armstrong vào các PTH trong F.

Thí dụ 2.18:

Cho R={A, B, C, D}là lược đồ quan hệ.

F = {A → B, A → C, A → D}, f là A → BCD

Ta thấy ngay f có thể nhận được từ phép cộng đầy đủ (suy từ hệ tiên đề Armstrong), tức F |= A →BCD và F |= A→AB (theo tính phản xạ và cộng đầy đủ) hoặc F |= A → ABCD (theo tính cộng đầy đủ),...

2.4.3.2 Phép suy dẫn theo quan hệ

Trên đây chúng ta vừa nêu các phép suy dẫn theo hệ tiên đề.

Sau đây chúng ta sẽ nêu *phép suy dẫn theo quan hệ*.

Cho lược đồ quan hệ R = {A_1, A_2,... A_n}.

F tập PTH trên R, f là một PTH trên R.

Ta nói rằng *f suy dẫn được từ tập PTH F theo quan hệ (hoặc PTH f được suy dẫn theo quan hệ từ tập PTH F)*, ký hiệu F |- f, nếu với mọi quan hệ r trên lược đồ R mà F thỏa mãn quan hệ đó thì f cũng thỏa mãn r. Nói cách khác F |- f nếu với mọi quan hệ r trên R mà tập F là tập PTH thỏa mãn r thì f cũng là một PTH thỏa mãn r. Đây là một phép suy dẫn theo quan hệ, "ở đâu tập F thỏa mãn thì ở đó f thỏa mãn " .

Một vấn đề được đặt ra ở đây là hai luật suy dẫn ở trên có cho ta cùng một tập PTH { f } không?

Sau đây ta sẽ thấy hai luật suy dẫn là tương đương. Nói cách khác hai luật suy dẫn thực chất là một.

Định lý 2.1

Cho tập PTH F và một PTH f trên R khi đó ta có:

$$F \vdash f \quad \text{khi và chỉ khi} \quad F \models f.$$

Thực chất của việc chứng minh định lý 2.1 là ta phải chứng minh hai ý:

1 - Các f được suy dẫn theo hệ tiên đề Armstrong từ tập PTH F, nghĩa là F |= f thì f là một PTH trên R (tính đúng đắn của f hay còn gọi là tính đúng đắn của luật suy dẫn theo hệ tiên đề Armstrong), điều này có nghĩa là chúng ta phải chứng minh f thỏa mãn mọi r mà trong đó tập F thỏa mãn.

2 - Các PTH hàm f được suy dẫn theo quan hệ từ F, nghĩa là F |- f, thì f cũng suy dẫn được từ F theo hệ tiên đề Armstrong (tính đầy đủ của hệ tiên đề Armstrong). Điều này tương đương với việc một hàm f không suy dẫn được theo hệ tiên đề Armstrong thì cũng không suy dẫn được theo quan hệ, nghĩa là nếu f không suy dẫn được theo hệ tiên đề Armstrong thì tồn tại một quan hệ r mà trong nó tập F thỏa mãn nhưng f không thỏa mãn.

Như vậy để chứng minh định lý 2. 1 ở trên ta sẽ chứng minh định lý tương đương thường được nêu trong các tài liệu về DATABASE là định lý 2. 2.

Định lý 2.2

Hệ tiên đề Armstrong là đúng đắn và đầy đủ.

Tính đúng đắn của hệ tiên đề chúng ta đã chứng minh ở trên, trong quá trình chứng minh ba tính chất A1, A2, A3 của hệ tiên đề Armstrong đồng thời chúng ta đã chỉ ra rằng mọi PTH được suy ra từ hệ tiên đề này đều là phụ thuộc hàm thỏa mãn trên lược đồ R, mà trong đó tập PTH F thỏa mãn.

Để chứng minh tính đầy đủ của hệ tiên đề trước tiên chúng ta sẽ chứng minh bổ đề sau:

Bổ đề 2.1

Giả sử $X \subseteq R$. Nếu gọi X^+ là tập tất cả các thuộc tính A của R mà $F \models X \rightarrow A$ (về sau ta sẽ gọi X^+ là bao đóng của X) thì với mọi tập $Y \subseteq R$, $F \models X \rightarrow Y \Leftrightarrow Y \subseteq X^+$.

Ta sẽ chứng minh bổ đề này

a - Chứng minh chiều thuận:

Ta có $F \models X \rightarrow Y$. Giả sử Y = {A,B,C,... } theo tính phản xạ ta có:

$F \models X \rightarrow A$, nên $A \in X^+$

$F \models X \rightarrow B$, nên $B \in X^+$

$F \models X \rightarrow C$, nên $C \in X^+$, ...

Vậy {A, B, C... } = $Y \subseteq X^+$

b - Chứng minh chiều ngược:

Ta có $Y \subseteq X^+$. Theo định nghĩa của tập X^+ thì mọi $A \in Y$ ta có

$F \models X \rightarrow A$, vậy theo tính cộng đầy đủ ta có $F \models X \rightarrow Y$.

Bổ đề đã được chứng minh.

Bây giờ chúng ta sẽ chứng minh tính đầy đủ của hệ tiên đề Armstrong.

Giả sử $f = X \rightarrow Y$ là một PTH trên R không suy dẫn được từ tập PTH F theo hệ tiên đề Armstrong, tức F not $\models X \rightarrow Y$. Ta sẽ xây dựng một quan hệ r trên R mà trên đó tập các PTH F thỏa mãn nhưng $f = X \rightarrow Y$ không thỏa mãn. Ta xét một quan hệ r trên R gồm chỉ hai phần tử t_1 và t_2 như sau: chia tập R thành hai nhóm, một nhóm gồm các thuộc tính của R thuộc tập X^+ và nhóm thứ hai là các thuộc tính còn lại của R; t_2 của quan hệ r chứa toàn giá trị 1; t_1 của r chứa 1 trong X^+ và toàn 0 trong các thuộc tính còn lại:

r

X^+				$R - X^+$			
1	1	1	... 1	0	0	0	... 0
1	1	1	... 1	1	1	1	... 1

Như vậy quan hệ r có hai phần tử t_1, t_2. Phần tử t_1 chứa giá trị 1 trong các thuộc tính của X^+ và giá trị 0 trong những thuộc tính còn lại; còn t_2 chứa toàn giá trị 1 cho mọi thuộc tính.

Ta chứng minh rằng r sẽ thỏa mãn mọi PTH hàm của F. Thật vậy giả sử có một phụ thuộc hàm $W \rightarrow V$ của F không thỏa mãn r, thế thì $W \subseteq X^+$, nếu không sẽ vi phạm tính bằng nhau của hai bộ t_1 và t_2 trên W. Hơn nữa V không thể là tập con của X^+, vì nếu không thì $W \rightarrow V$ sẽ thỏa mãn r. Vậy có một thuộc tính A của V không thuộc X^+.

Theo bổ đề 2.1 thì $W \subseteq X^+ \Leftrightarrow F \models X \rightarrow W$, mà $W \rightarrow V$, nên $W \rightarrow A$ và theo tính bắc cầu (vì $X \rightarrow W$) nên $X \rightarrow A$, tức $F \models X \rightarrow A$, hay A thuộc X^+. Điều này là vô lý vì A không thuộc X^+.

Vậy r thỏa mãn mọi PTH của F. Vấn đề còn lại chúng ta phải chứng minh rằng r không thỏa mãn phụ thuộc hàm $f = X \rightarrow Y$.

Giả sử $X \rightarrow Y$ thỏa mãn r thế thì $X, Y \subseteq X^+$ nếu không thì vi phạm tính bằng nhau của t_1 và t_2 trên X và Y. Lại sử dụng bổ đề 2.1 $Y \subseteq X^+ \Leftrightarrow F \models X \rightarrow Y$. Điều này vô lý, vì F không suy dẫn được theo hệ tiên đề f. Vậy $X \rightarrow Y$ không thỏa mãn r. Định lý được chứng minh.

2.4.4. BAO ĐÓNG F⁺ CỦA TẬP PHỤ THUỘC HÀM F

2.4.4.1 Định nghĩa F⁺

Trong phần trên chúng ta đã nói đến các PTH f được suy dẫn từ tập PTH F cho trước, ta đã có định lý chứng minh các phép suy dẫn theo tiên đề và theo quan hệ là tương đương nên từ nay thay cho nói suy dẫn theo quan hệ hoặc suy dẫn theo tiên đề ta chỉ nói đơn giản là suy dẫn. Tập các PTH f được suy dẫn từ tập PTH F về sau ta sẽ gọi là *bao đóng* của tập F.

Vậy cho lược đồ $R = \{A_1, A_2, ... A_n\}$. F là tập PTH trên R.

Bao đóng của tập PTH F ký hiệu F^+ là tập tất cả các phụ thuộc hàm f được suy dẫn từ tập F. Vậy $F^+ = \{f: F \models f\}$.

Thí dụ 2.19:

Cho lược đồ $R = \{A, B, C, D\}$

Giả sử tập F trên R như sau:

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

Khi đó $F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow D, B \rightarrow D, A \rightarrow BD, A \rightarrow BCD, A \rightarrow C, A \rightarrow CD, A \rightarrow BC, B \rightarrow CD, ...\}$.

Qua thí dụ 2. 19 và cũng từ định nghĩa ta thấy F^+ luôn chứa F.

2.4.4.2 Các tính chất đơn giản của tập F⁺

a. *Tính phản xạ*: với mọi tập PTH F ta luôn có $F \subset F^+$.

b. *Tính đơn điệu*: nếu $F \subset G$ thì $F^+ \subset G^+$.

c. *Tính lũy đẳng*: với mọi tập phụ thuộc hàm F ta luôn có $F^{++} = F^{+}$.

Để giáo trình không bị ảnh hưởng quá nặng về những khái niệm đơn thuần về lý thuyết toán chúng tôi không muốn đi sâu vào khái niệm F^{+}. Các bạn có thể tìm hiểu sâu về phần này trong các tài liệu tham khảo ở cuối giáo trình. Phần chứng minh các tính chất a, b, c của bao đóng của tập F chúng tôi dành cho các bạn như một bài tập nhỏ.

2.4.5. BAO ĐÓNG X^{+} CỦA TẬP THUỘC TÍNH X

2.4.5.1. Định nghĩa bao đóng X^{+}

Cho lược đồ quan hệ $R = \{A_1, A_2, ... A_n\}$. Giả sử F là tập PTH trên R.

X là tập con của tập thuộc tính R.

Bao đóng của tập thuộc tính X đối với F, ký hiệu X^{+} (hoặc X_F^{+} để chỉ bao đóng lấy theo tập F), là tập tất cả các thuộc tính A của R mà $X \rightarrow A$ được suy dẫn từ tập F. Vậy X^{+} là tập:

$X^{+} = \{A: \ A \in R \text{ và } X \rightarrow A \in F^{+}\}$.

Như vậy bao đóng X^{+} của X được định nghĩa qua tập phụ thuộc hàm F, vì thế đôi khi ta ký hiệu X_F^{+}. Tuy nhiên khi không có một tập phụ thuộc hàm nào khác ta hiểu bao đóng X_F^{+} được tính qua F nên ta viết đơn giản X^{+}.

Thí dụ 2.20:

Giả sử $R = \{A, B, C, D, E, G\}$.

$F = \{A \rightarrow C, A \rightarrow EG, B \rightarrow D, G \rightarrow E\}$.

$X = \{A, B\}$, $Y = \{C, G, D\}$.

Khi đó ta sẽ có: $X^+ = \{A, B, C, D, E, G\}$

$Y^+ = \{C, G, D, E\}$.

Tương tự như tập bao đóng của tập PTH F^+ tập bao đóng X^+ cũng chứa các phần tử của tập X, tức là $X \subset X^+$.

2.4.5.2.Các tính chất của tập bao đóng X^+

Nếu X, Y là các tập con của tập thuộc tính R thì ta có các tính chất:

1. Tính phản xạ: $X \subset X^+$
2. Tính đơn điệu: Nếu $X \subset Y$ thì $X^+ \subset Y^+$
3. Tính lũy đẳng: $X^{+\,+} = X^+$
4. $(XY)^+ \supset X^+Y^+$ (bao đóng của tổng chứa tổng các bao đóng)
5. $(X^+Y)^+ = (XY^+)^+ = (X^+ Y^+)^+ = (XY)^+$
6. $X \rightarrow Y \Leftrightarrow Y \subset X^+$
7. $X \rightarrow Y \Leftrightarrow Y^+ \subset X^+$
8. $X \rightarrow X^+$ và $X^+ \rightarrow X$
9. $X^+ = Y^+ \Leftrightarrow X \rightarrow Y$ và $Y \rightarrow X$.

Chúng ta thấy bao đóng của tập PTH F và bao đóng của tập thuộc tính X là những tập liên quan với hệ tiên đề Armstrong. Các tập bao đóng là "kết quả" của các phép suy dẫn dùng luật của tiên đề Armstrong. Sau đây chúng ta sẽ chứng minh các tính chất của tập bao đóng.

Tính chất 1: $X \subset X^+$. Thật vậy theo tính phản xạ của hệ tiên đề Armstrong ta có ngay với mọi thuộc tính A của X thì $X \rightarrow A$. Tất nhiên $X \rightarrow A \in F^+$, vì $X \rightarrow A$ được suy từ hệ tiên đề.

Tính chất 2 (tính đơn điệu): Giả sử $X \subset Y$ ta phải chứng minh $X^+ \subset Y^+$.

Thật vậy lấy $A \in X^+$, theo định nghĩa ta có $X \to A$ mà $X \subset Y$ nên theo tính phản xạ của hệ tiên đề Armstrong, ta có $Y \to A$. Vậy $A \in Y^+$

Tính chất 8: Để chứng minh các tính chất khác trước tiên ta chứng minh tính chất 8 tức là tính chất $\forall X$ thì $X \to X^+$ và $X^+ \to X$.

Thật vậy theo tính phản xạ $X^+ \to X$ vì $X \subset X^+$

Bây giờ ta chứng minh $X \to X^+$. Theo định nghĩa của tập X^+ ta có $X^+ = XZ$ với $Z = \{A: X \to A \in F^+ \ \& \ A \notin X\}$; theo tính cộng đầy đủ (cộng lần lượt 2 vế) ta có $X \to Z$. Hơn nữa theo tính phản xạ ta có tiếp $X \to X$. Theo tính cộng đầy đủ (cộng 2 vế) ta có $X \to XZ$ tức $X \to X^+$.

Tính chất 3: $X^{+\,+} = X^+$

Rõ ràng theo tính phản xạ ta có ngay $X^+ \subset X^{+\,+}$. Bây giờ lấy $A \in X^{++}$ tức là $X^+ \to A \in F^+$ mà $X \to X^+$ nên $X \to A \in F^+$ hay $A \in X^+$. Vậy $X^{+\,+} = X^+$.

Tính chất 4: $(XY)^+ \supset X^+Y^+$

Lấy $A \in X^+Y^+$ tức $A \in X^+$ hoặc $A \in Y^+$ tức là $X \to A$ hoặc $Y \to A \Rightarrow XY \to A \in F^+$. Hay $A \in (XY)^+$.

Tính chất 9: $X^+ = Y^+ \Leftrightarrow X \to Y$ và $Y \to X$

a - Chiều xuôi $\Rightarrow$: ta có $X^+ = Y^+$. Vì $X \to X^+$ và $Y^+ \to Y$ nên $X \to Y$ chứng minh tương tự ta có $Y \to X$.

b - Chiều ngược $\Leftarrow$: Lấy $A \in X^+$ tức là $X \rightarrow A$ vì $Y \rightarrow X$ nên $Y \rightarrow A$ hay $A \in Y^+$. Tương tự lấy $A \in Y^+$ ta chứng minh được $A \in X^+$. Vậy $X^+ = Y^+$. Các tính chất còn lại chứng minh tương tự.

2.4.6. THUẬT TOÁN TÌM BAO ĐÓNG X^+, BÀI TOÁN THÀNH VIÊN

2.4.6.1. Bài toán thành viên

Phần trên chúng ta đã nêu một số khái niệm cơ bản và quan trọng của các tập bao đóng. Ta thấy tập X^+ được định nghĩa thông qua tập F^+. Một vấn đề quan trọng trong lý thuyết về CSDL là: Cho trước tập PTH F và một PTH f, có hay không một khẳng định f thuộc F^+ ? (gọi là *bài toán thành viên*)

Để trả lời được câu hỏi này (*bài toán thành viên xác định xem f có là thành viên của F^+ không ?*) không đơn giản vì mặc dù F là tập nhỏ nhưng tập F^+ có thể rất lớn. Thí dụ, xét tập F: $F = \{A \rightarrow B_1, A \rightarrow B_2, \ldots, A \rightarrow B_n\}$.

Khi đó F^+ chứa tất cả các PTH dạng $A \rightarrow Y$, trong đó Y là tập con bất kỳ của tập $\{B_1, B_2, \ldots, B_n\}$. Vì có tới 2^n những tập con Y như vậy, nên số lượng các phần tử của F^+ sẽ rất lớn, lớn hơn hoặc bằng 2^n.

Vậy để giải *bài toán thành viên* chúng ta có thể dùng tính chất 6 của tập bao đóng X^+ hoặc bổ đề 2.1 đó là tính chất: $X \rightarrow Y \in F^+ \Leftrightarrow Y \subset X^+$. Do vậy chỉ cần tính X^+ và so sánh với tập Y ta có ngay câu trả lời là $X \rightarrow Y$ thuộc F^+ hay không. Việc tính X^+ được giải quyết đơn giản hơn rất nhiều.

Sau đây chúng ta sẽ trình bày một phương pháp tính tập X^+.

2.4.6.2 Thuật toán tìm bao đóng X $^+$

Dưới đây là *thuật toán tìm* X^+ của Beeri và Bernstein.

Cho lược đồ R = {A_1, A_2,... , A_n}. F là tập PTH trên R, X là tập thuộc tính. Tính X^+ =?

Ta sẽ xây dựng dãy X^0, X^1,... , X^k... như sau:

$X^0 = X$

$X^{(i+1)} = X^i Z^i$ với $Z^i = \{A: A \notin X^i$ và $X^i \rightarrow A \in F^+\}$, trong đó i = 0, 1, 2,...

Ta có nhận xét rằng: dãy X^0, X^1,... có thể xây dựng được nhờ tập X tập F và các phép suy dẫn của hệ tiên đề Armstrong. Hơn nữa dãy X^0, X^1, X^2,... là dãy lồng nhau và tăng dần, tức là $X^0 \subset X^1 \subset$... Vì tập thuộc tính R là hữu hạn nên sau một số hữu hạn bước, thuật toán phải kết thúc. Nói cách khác tồn tại số nguyên k (bé nhất) sao cho:

$X^k = X^{(k+1)} = X^{(k+2)}$ =... tất nhiên khi đó $Z^k = Z^{k+1}$ = . . . = rỗng

Tập X^k đó chính là tập X^+

Trước khi chứng minh rằng X^k chính là tập X^+ ta sẽ trình bày thuật toán (bằng ngôn ngữ tựa Pascal) và xét thí dụ minh họa thuật toán.

Thuật toán 2.1:

Input: Lược đồ quan hệ R

Tập PTH F

Tập thuộc tính X

Output: Tập X^+

Thuật toán:

```
Begin
Y: = X;
repeat
Z: = ∅;
for each A in R do
if (A ∉ Y and Y → A ∈ F+) then Z: = Z ∪ A;
Y: = Y ∪ Z;
until Z = ∅;
X+ = Y
 end;
```

Thí dụ 2.21:

Giả sử R = {A, B, C, D, E, G} và tập PTH F như sau:

F = {AB → C, C → A, BC → D, ACD → B, D → EG, BE → C, CG → BD, CE → AG}, X = {B, D}, X^+ = ?

Đầu tiên ta có X^0 = {B, D}. Để tìm X^1 ta tìm những PTH trong F có vế trái nằm trong BD. Ta có PTH D → EG thỏa mãn điều kiện đó.

Vậy Z^0 = {E, G}, nên X^1 = {B, D, E, G}. Tiếp tục để tìm X^2 ta tìm những PTH của F có vế trái nằm trong {B, D, E, G}, đó là D → EG và BE → C, vậy X^2 = {B, C, D, E, G}. Tiếp tục ta có X^3 = {A, B, C, D, E, G}. Đây là tập X^+

Vậy X^+ = {B, D}$^+$ = {A, B, C, D, E, G} = R.

Định lý 2.3:

Trong thuật toán tìm bao đóng X^+, ta có $X^+ = X^k$, với k là số nguyên bé nhất mà $X^k = X^{k+1} = X^{k+2} =...$

Chứng minh:

a - Ta chứng minh $X^+ \subset X^k$. Thật vậy lấy $A \in X^+$. Như ở trên ta đã thấy $X^+ = XZ$ với $Z = \{A: A \notin X$ và $X \rightarrow A \in F^+\}$.

Vậy nếu $A \in X$ thì $A \in X^k$ vì $X \subset X^k$; còn nếu $A \in Z$ thì theo định nghĩa của các tập Z^i, tồn tại một chỉ số i để $A \in Z^i$ vậy $A \in X^k$ vì với mọi i thì $X^i \subset X^k$.

Vậy trong cả hai trường hợp ta đều có $A \in X^k$ và suy ra $X^+ \subset X^k$.

b - Ta có thể chứng minh $X^k \subset X^+$ bằng nhiều cách, thí dụ :

Cách 1:

Ta có $X^{i+1} = X^iZ^i$

$Z^i = \{A: A \notin X^i$ và $X^i \rightarrow A \in F^+\}$.

Vậy $X^0 = X$

$X^1 = X^0Z^0$

$X^2 = X^1Z^1 = X^0Z^0Z^1$

...

$X^{i+1} = X^iZ^i = X^0Z^0Z^1...Z^i$ với i = 0, 1, 2,...

Trước tiên ta sẽ chứng minh bằng phương pháp quy nạp rằng với mọi i thì $X \rightarrow Z^i$. Với i = 0, $Z^0 = \{A: A \notin X^0$ và $X^0 \rightarrow A \in F^+\}$;

Theo tính cộng đầy đủ từ $X^0 = X \rightarrow A$ ta cộng 2 vế theo các phần tử $A \in Z^0$, ta có $X \rightarrow Z^0$.

Bây giờ giả sử bài toán đúng với i, ta chứng minh cho i + 1, với $Z^{i+1} = \{A: A \notin X^{i+1}$ và $X^{i+1} \rightarrow A \in F^+\}$, tương tự cộng 2 vế theo các phần tử A của

tập Z^{i+1}, ta có $X^{i+1} \to Z^{i+1}$, trong đó $X^{i+1} = XZ^0 ... Z^i$, theo giả thiết quy nạp và tính phản xạ ta lại có :

$X \to X^0 = X$

$X \to Z^1$

$X \to Z^2$

...

$X \to Z^i$ cộng lần lượt 2 vế, ta có $X \to X^{i+1}$

Vì $X^{i+1} \to Z^{i+1}$ theo tính bắc cầu ta có $X \to Z^{i+1}$.

Bây giờ ta trở lại chứng minh $X^k \subset X^+$.

Lấy $A \in X^k$ và $X^k = XZ^0 ... Z^{k-1}$. Nếu $A \in X$ thì

$X \to A \in F^+$ nên $A \in X^+$. Còn nếu $A \in Z^i$ thì vì

$X \to Z^i$ nên

$X \to A \in F^+ \Rightarrow A \in X^+$.

Định lý được chứng minh.

Cách 2:

Dễ dàng thấy rằng $X \to X^1$, $X^1 \to X^2$, ... , $X^{k-1} \to X^k$

$\Rightarrow X \to X^k \Leftrightarrow X^k \subset X^+$ (theo tính chất của bao đóng).

Thí dụ 2.22:

Cho R = {A, B, C, D, E, I}.

Tập PTH F = {A → D, AB → E, BI → E, CD → I,

E → C}. Tập thuộc tính X = {A, E}. Tính X^+ = ?

Ta có X^0 = {A, E}

$X^1 = X^0Z^0$ và $Z^0 = \{Y: Y \notin X^0$ và $X^0 \to Y \in F^+\}$

Vậy $Z^0 = \{D, C\}$ và $X^1 = \{A, E, D, C\}$

$Z^1 = \{Y: Y \notin X^1$ và $X^1 \rightarrow Y \in F^+\}$ và

$Z^1 = \{I\}$ nên $X^2 = \{A, E, D, C, I\}$

$Z^2 = \{Y: Y \notin X^2$ và $X^2 \rightarrow Y \in F^+\} = \varnothing$

Vậy $X^3 = X^2 = X^+ = \{A, E, D, C, I\}$.

2.5. KHÓA CỦA SƠ ĐỒ QUAN HỆ

Trong công tác xử lý các CSDL dạng quan hệ ta đã thấy giữa các tập thuộc tính có các mối ràng buộc kiểu PTH. Trong số các tập thuộc tính tham gia vào các PTH ta sẽ thấy có một số tập đóng vai trò quan trọng. Thí dụ, trong hồ sơ nhân sự của cơ quan thuộc tính mã nhân viên (MA-NV) đóng vai trò "xác định" của hồ sơ theo nghĩa ta có thể nhận biết trên mạng toàn bộ số liệu của một cán bộ theo MA-NV, tức là có thể dùng thuộc tính MA-NV như là *khóa*. Hoặc trong lược đồ quản lý tuyển sinh đại học, thuộc tính số báo danh SBD đóng vai trò là *khoá*.

Sau đây để cho tiện khi trình bày và sử dụng ta nêu thuật ngữ *sơ đồ quan hệ* (viết tắt là SĐQH) như sau:

Cho lược đồ quan hệ $R = \{A_1, A_2, \ldots A_n\}$; F là tập phụ thuộc hàm trên R. *Sơ đồ quan hệ* là cặp R , F như trên và ta sẽ ký hiệu SĐQH là $W = < R, F >$. Như vậy nói cho một SĐQH, thực chất chỉ là cho trước lược đồ R, tập PTH F trên R và viết chúng thành cặp $< R, F >$.

2.5.1.ĐỊNH NGHĨA KHÓA CỦA SƠ ĐỒ QUAN HỆ

Tập $k \subset R$ được gọi là *Khóa tối thiểu* của sơ đồ quan hệ W nếu k là tập tối thiểu kéo theo R, tức là k là khoá tối thiểu nếu: $k^+ = R$ ($k \rightarrow R$) và bớt

khỏi k dù một phần tử thì bao đóng của tập còn lại khác R. Vậy tập $k \subset R$ gọi là *khóa tối thiểu* nếu: $k^+ = R$ và $(k - A)^+ \neq R$, với A bất kỳ thuộc k.

Nói cách khác k là một khóa tối thiểu của W nếu k là tập bé nhất có thể và có bao đóng đúng bằng R, hiển nhiên rằng khi k có bao đóng bằng R thì thêm vào k một phần tử ta cũng được tập có bao đóng bằng R. Tuy nhiên khi k đã là khóa thì bớt đi một phần tử ta có tập mà bao đóng không bằng R.

Trực quan từ định nghĩa, ta thấy rằng nếu k là một tập thuộc tính mà $k^+ = R$ thì từ k ta có thể bớt dần các phần tử của k, để nhận được tập k bé nhất và *đó chính là khóa của sơ đồ quan hệ*. Về sau các thuộc tính thuộc một khóa nào đó ta gọi là *thuộc tính khóa*, ngược lại thuộc tính không thuộc khóa nào gọi là *thuộc tính không khóa (hoặc thuộc tính thứ cấp)* và ta ký hiệu là F_n.

Lưu ý rằng trong giáo trình này chỉ xét khoá tối thiểu và vì vậy thay cho khoá tối thiểu ta gọi tắt là khoá.

Thí dụ 2.23 :

Cho sơ đồ quan hệ $W = < R, F >$ với $R = \{A, B, C, D, E, G\}$ $F = \{AB \rightarrow C, D \rightarrow EG, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CG \rightarrow BD, ACD \rightarrow B, CE \rightarrow AG\}$.
Ta sẽ thấy các tập thuộc tính:
$k_1 = \{A, B\}$, $k_2 = \{B, E\}$, $k_3 = \{C, G\}$, $k_4 = \{C, E\}$, $k_5 = \{C, D\}$, $k_6 = \{B, C\}$
đều là các khóa của W và tập các thuộc tính khóa bằng R, và vì vậy $F_n = \varnothing$.

Vậy một sơ đồ quan hệ có thể có nhiều khóa và tập thứ cấp F_n có thể rỗng, và mọi sơ đồ quan hệ luôn có khoá vì ta lấy k = R và tối thiểu dần k để được khoá. Vấn đề còn lại đối với chúng ta là: cho trước một sơ đồ quan hệ $W = < R, F >$, làm thế nào để tìm khóa của nó ?

2.5.2. CÁC THUẬT TOÁN TÌM KHÓA

Ta nhắc lại khóa là tập thuộc tính k mà bao đóng của k đúng bằng R ($k^+ = R$) và nếu bớt khỏi k một phần tử bất kỳ thì bao đóng của nó khác R.

Từ định nghĩa khoá ta thấy có thể tìm khóa bắt đầu từ tập R vì $R^+ = R$ và ta bớt dần các phần tử của R để nhận được tập bé nhất mà bao đóng của nó đúng bằng R.

Sau đây chúng ta sẽ trình bày thuật toán tìm một khóa theo ý tưởng này: Cho sơ đồ quan hệ W = < R, F >, với R - lược đồ quan hệ, F - tập PTH trên R. Tìm một khóa của W.

Thuật toán 2.2:

Input: W= < R,F>;

Output: k - khóa của W;

Thuật toán:

Bước 1: Đặt k = R

Bước 2: Lặp quá trình loại khỏi k

phần tử A mà $(k - A)^+ = R$

Mô tả thuật toán (bằng tựa Pascal):

```
Begin
  k: = R;
  for each A in k do
```

if $(k - A)^+ = R$ then k: = k - A else k:= k

End

Nhận xét: Thuật toán 2.2 trên đây cho ta tìm được một khóa của sơ đồ quan hệ W.

Nếu muốn tìm các khóa khác (nếu có) của sơ đồ quan hệ ta có thể thay đổi thứ tự loại bỏ các phần tử của k.

Thí dụ 2.24:

Cho W = < R, F >, với

R = {A, B, C, D, E, G, H, I}

F = {AC → B, BI → ACD, ABC → D

H → I, ACE → BCG, CG → AE}

Tìm k = ?

Bước 1: k = R = {A, B, C, D, E, G, H, I}

Bước 2: Lần lượt loại các thuộc tính *thừa* có trong k:

Loại phần tử A: Ta có $\{B, C, D, E, G, H, I\}^+ = R$ (vì CG → AE) nên k = {B, C, D, E, G, H, I}

Loại phần tử B: Ta có $\{C, D, E, G, H, I\}^+ = R$ (vì CG → AE và AC → B) nên k = {C, D, E, G, H, I}

Loại phần tử C: Ta có $\{D, E, G, H, I\}^+ \neq R$ nên k = {C, D, E, G, H, I}

Loại phần tử D: Ta có $\{C, E, G, H, I\}^+ = R$ (vì CG → AE, AC → B, ABC → D) nên k = {C, E, G, H, I}

Loại phần tử E: Ta có $\{C, G, H, I\}^+ = R$ (vì CG → AE, AC → B, ABC → D} nên k = {C, G, H, I}

Loại phần tử G: Ta có $\{C, H, I\}^+ \neq R$ nên k = {C, G, H, I}

Loại phần tử H: Ta có $\{C, G, I\}^+ \neq R$ nên k = {C, G, H, I}

Loại phần tử I: Ta có $\{C, G, H\}^+ = R$ (vì CG → AE, AC → B, ABC → D, H → I} nên k = {C, G, H}. Vậy k = {C, G, H} là một khóa của W.

2.5.3.CÁC TÍNH CHẤT CỦA KHOÁ

Từ định nghĩa khoá và thuật toán tìm khóa ta có các *kết luận* sau:

1 - Các thuộc tính không xuất hiện trong cả vế trái và vế phải của tập F phải có trong mọi khóa k.

2 - Các thuộc tính chỉ xuất hiện bên trái của các PTH trong F cũng phải thuộc mọi khóa k.

3 - Trong quá trình tìm khóa ta có thể bỏ tất cả các thuộc tính đơn phía bên phải các PTH của F. Tuy nhiên cần kiểm tra lại vì không phải lúc nào các thuộc tính đó cũng bỏ được. Ví dụ, R = {A,B,C,D,E}

F_1 = {A→ B, C → E, C →D}, khi đó khóa là k = {A,C}.

Nếu F_2 = {A → C, C →ABDE} thì k = {A}, k'={C} là khóa (trong trường hợp này ta không bỏ C được, mặc dù C xuất hiện đơn ở bên phải của PTH của F).

4- Thuật toán 2.2 đã khẳng định *mọi sơ đồ quan hệ W đều có khoá*, tuy nhiên thuật toán *không khẳng định W có bao nhiêu khoá* và *số lượng các phần tử trong mỗi khoá có như nhau không*. Chẳng hạn W = <{A,B,C,D}, {A→BCD, CD→AB}> thì W có hai khoá k_1= {A}, k_2= {C, D}.

5- họ tất cả các khoá của một SĐQH W là hệ Sperner (tức không có hai khoá bao nhau)

Trong [8], tác giả đã định nghĩa khóa là tập thuộc tính k tối thiểu thỏa mãn tính chất: với mọi cặp t_1, t_2 khác nhau của mọi quan hệ r trên R ta luôn có $t_1.k \neq t_2.k$, (xem [8] trang 11). Vậy ta có định lý sau:

Định lý 2.4:

a. Nếu k là khóa của sơ đồ quan hệ W = < R, F >, r là quan hệ trên R thì với mọi cặp phần tử khác nhau t_1, t_2 của r ta luôn có $t_1.k \neq t_2.k$.

Ta có thể chứng minh điều này một cách đơn giản. Thật vậy:
Giả sử K là khoá của W, t_1,t_2 là hai phần tử khác nhau của r mà $t_1.k = t_2.k$. Vì k là khóa nên ta có $k \rightarrow R$ và từ $t_1.k = t_2.k$ ta có $t_1.R = t_2.R$ tức $t_1 = t_2$ điều này mâu thuẫn với giả thiết là $t_1 \neq t_2$.

b. Ngược lại nếu k là tập tối thiểu và với mọi quan hệ r trên R mà mọi cặp t_1, t_2 của r mà $t_1.k \neq t_2.k$ thì k là khoá của sơ đồ quan hệ W = < R, F >.

2.6. CÁC DẠNG CHUẨN CỦA SƠ ĐỒ QUAN HỆ

Trong công tác quản lý và xử lý các hệ cơ sở dữ liệu, chúng ta thường phải đưa CSDL về dạng đơn giản nhất, ít cồng kềnh nhất, tốn ít bộ nhớ nhất, xử lý nhanh nhất và tất nhiên có tính đặc thù nhất. Để thực hiện mục đích đó chúng ta phải tiến hành "chuẩn hóa" các hệ CSDL. Tức là chúng ta sẽ xét một số dạng đặc biệt mà trong CSDL gọi là các dạng chuẩn (Normal Forms) và trong nhiều trường hợp CSDL ở Normal Form này thì tốt hơn ở Normal

Form kia. Sự phân loại các sơ đồ quan hệ W = < R, F > theo các Normal Form thực chất là dựa vào các tập phụ thuộc hàm F để chúng ta phân loại sơ đồ quan hệ thuộc dạng chuẩn nào.

Sau đây chúng ta sẽ xét một số dạng quen thuộc của các sơ đồ quan hệ đã được nhiều tác giả quan tâm.

2.6.1.DẠNG CHUẨN 1NF

Dạng chuẩn 1 (first Norm Form) ký hiệu là 1NF.

Cho lược đồ quan hệ R, F là tập phụ thuộc hàm trên R. SDQH W = <R,F> *được gọi là dạng chuẩn 1* (1NF) nếu và chỉ nếu toàn bộ các miền giá trị có mặt trong R đều chỉ chứa các giá trị nguyên tố (giá trị nguyên tố là giá trị không thể tách thành các giá trị khác - giá trị đơn).

Khi W = <R,F> là 1NF thì mọi quan hệ r trên R cũng được gọi tương ứng là *quan hệ 1NF.*

Thí dụ 2.25:

Xét bảng quản lý học viên cao học theo học một số chuyên đề tại trung tâm đào tạo sau đại học:

a - Quan hệ không là 1NF

MS	TÊN	NGÀNH	MÔNHOC	TIỀN	KẾTTHUC
01	An	Vật lý	Quang	200	30/9/99
02	Anh	Toán	Đại số	200	30/8/99
03	Bình	Hóa	Cao phân tử	300	30/10/99
04	Long	Môi trường	Môi trường	500	30/10/99
05	A,B	Tin	CSDL,SQL	600	30/11/99

Đây là một quan hệ không là 1NF vì các thuộc tính như MÔNHỌC có miền giá trị " Đa trị " (không đơn trị, có thể tách được), hoặc thuộc tính TÊN ở phần tử thứ 5 của quan hệ có hai giá trị là A, B. Vậy quan hệ trên *không là dạng chuẩn 1NF* và tất nhiên W = <R,F> cũng không là 1NF.

Tuy nhiên nếu bỏ qua phần ngữ nghĩa của quan hệ ta luôn có thể đưa dạng chưa chuẩn trên về dạng chuẩn (dạng mà các giá trị của các thuộc tính là đơn) như sau:

b - Quan hệ là 1NF

MS	TÊN	NGÀNH	MÔNHOC	TIỀN	KẾTTHUC
01	An	Vật lý	Quang	200	30/9/99
02	Anh	Toán	Đại số	200	30/8/99
03	Bình	Hóa	Cao phân tử	300	30/10/99
04	Long	Môi trường	Môi trường	500	30/10/99
05	A	Tin	CSDL	300	30/11/99
05	B	Tin	SQL	300	30/11/99

Nhận xét: Hai bảng quan hệ trên đều cùng quản lý một mảng thông tin của một nhóm đối tượng, có cấu trúc lôgic gần như nhau nhưng cấu trúc vật lý khác nhau. Tuy nhiên, trong bảng a - có thể coi r là một quan hệ 5 phần tử, còn bảng b - là một quan hệ sáu phần tử và thông tin đã rõ ràng và xác định hơn. Tất nhiên từ quan hệ đa trị đưa về đơn trị là không duy nhất. Như vậy mọi quan hệ chúng ta đều có thể đưa về dạng đơn trị nên mọi SĐQH W = < R, F> đều có thể coi là chuẩn 1NF. Vậy lớp 1NF chứa tất cả các sơ đồ quan hệ. Từ nay ta chỉ xét sơ đồ quan hệ 1NF.

2.6.2.DẠNG CHUẨN 2NF

Ta nói Y phụ thuộc hoàn toàn vào X nếu trong X không có tập con thực sự X_1 mà $X_1 \rightarrow Y$. Nói cách khác Y phụ thuộc hoàn toàn vào X nếu:

(*) $X \rightarrow Y$ và bớt khỏi X dù một thuộc tính A.

(**) not $(X \setminus A) \rightarrow Y$.

Cho sơ đồ quan hệ $W = < R, F >$.

K là một khóa của W.

Ta nói W là (ở) dạng chuẩn 2, ký hiệu là 2NF, nếu mọi thuộc tính thứ cấp của W phụ thuộc hoàn toàn vào khóa. Nói cách khác W là 2NF nếu: trong W không có PTH dạng $X \rightarrow x \in F_n$ với X là tập con thực sự của khóa K và x là thuộc tính không khóa (thứ cấp).

Từ định nghĩa ta thấy ngay là lớp các sơ đồ quan hệ 2NF là lớp con thực sự của lớp 1NF, vì có nhiều sơ đồ quan hệ không là 2NF.

Thí dụ 2.26:

Ta xét hồ sơ nhân sự cơ quan (quan hệ r) như sau:

TT	HỌ-TÊN	NS	TĐÔ	QUÊ	GT
01	Tuấn Anh	1960	Đại học	Huế	Nam
02	Lan Anh	1977	Đại học	Hà Nội	Nữ
03	Đình Đông	1945	TS	Vĩnh Phú	Nam
05	Đình Đông	1943	TS	Hà Nội	Nam
06	Công Nụ	1960	TS	Vĩnh Phú	Nam
07	Hoa Huệ	1972	Tr. học	Nghệ An	Nữ

Ta thấy ngay rằng tập có một thuộc tính TT là khóa của quan hệ r vì theo định nghĩa của PTH ta có TT →{HO-TÊN, NS, TĐÔ, GT, QUÊ}. Vì r là 1NF và tập khóa chỉ có một phần tử nên không thể có phần tử không khóa phụ thuộc hàm vào tập con thực sự của khóa (tập con thực sự của khóa bằng rỗng), vậy r là 2NF. Ta có kết luận sau:

Kết luận:

W là 2NF nếu mỗi khóa của W chỉ có một phần tử.

Thí dụ 2.27:

Quay lại thí dụ 2. 23 , ta đã thấy khóa của W là:

K_1 = {A, B}, K_2 = {B, E}, K_3 = {C, G}, K_4 = {C, E},K_5 = {C, D}, K_6 = {B, C}.

Trong thí dụ này tất cả các phần tử của R đều là phần tử khóa, tức là tập các phần tử không khóa bằng rỗng nên không có PTH dạng A → x mà x là phần tử thứ cấp (không có phần tử x như vậy). Vậy W là 2NF.

Kết luận:

W là 2NF nếu tập các thuộc tính không khóa F_n của W bằng rỗng.

Thí dụ 2.28:

Sau đây ta sẽ nêu một thí dụ mà sơ đồ quan hệ W không là 2NF.

Cho W = < R, F >, với R = {A, B, C, D, E, H}

$F = \{A \rightarrow E, C \rightarrow D, E \rightarrow DH\}$.

Dễ dàng thấy rằng tập $K = \{A, B, C\}$ là khóa duy nhất của W, D là thuộc tính không khóa và $C \rightarrow D$, vì C là tập con thực sự của khóa nên *W* không là 2NF.

Như vậy khi xét một sơ đồ quan hệ $W = < R, F >$ có là 2NF không ta phải tính tất cả các khóa của sơ đồ và từ đó suy ra tập thuộc tính thứ cấp.

Sau đó xét xem có tập con thực sự nào của khóa kéo theo một thuộc tính thứ cấp không?

Thí dụ $W = < R, F >$; $R = \{A,B,C,D,E\}$, $F = \{A \rightarrow B, C \rightarrow D\}$.

Khi đó mọi khóa K phải chứa E, A, C và ta cũng thấy rằng tập $K = \{A,C,E\}$ là khóa duy nhất. Vậy các thuộc tính thứ cấp là B, D. Ta thấy trong khóa K có chứa các tập con thực sự A, C và chúng kéo theo các thuộc tính thứ cấp B, D tương ứng, nên W không là 2NF.

Vậy vi phạm của 2NF là tập con thực sự của khoá kéo theo thuộc tính thứ cấp và đây là lý do tại sao khi xét dạng 2NF ta phải tìm hết khoá để có tập Fn.

2.6.3. DẠNG CHUẨN 3NF

Cho sơ đồ quan hệ $W = < R, F >$. Ta nói W là (ở) dạng chuẩn 3, ký hiệu là 3NF nếu trong W không tồn tại PTH dạng $X \rightarrow x \notin X$, với mọi tập thuộc tính X mà $X^+ \neq R$ và x là thuộc tính thứ cấp.

Từ định nghĩa ta có nhận xét rằng nếu W là 3NF thì nó là 2NF. Trong dạng chuẩn 2NF ta chỉ cấm các thuộc tính thứ cấp phụ thuộc vào tập con thực sự của khóa (tập có bao đóng khác R), trong 3NF ta cấm thuộc tính thứ cấp phụ thuộc vào mọi tập (trong đó có tập con thực sự của khóa) có bao đóng khác R.

Thí dụ 2.29:

Sau đây là một thí dụ mà W không là 3NF, ta lấy thí dụ 2. 28. Trong thí dụ này ta thấy D là thuộc tính thứ cấp và $C \rightarrow D$, đồng thời $C^+ \neq R$. Vậy W không là 3NF.

Thí dụ 2.30:

Trở lại thí dụ 2. 23, ta thấy rằng W trong ví dụ này là 3NF, vì tập thuộc tính R ở đây không có thuộc tính không khóa (thứ cấp).

Thí dụ 2.31:

Cho sơ đồ quan hệ $W = < R, F >$, với $R = \{A, B, C, D\}$

$F = \{AB \rightarrow C, D \rightarrow B, C \rightarrow ABD\}$

Ta thấy rằng, các tập $K_1 = \{A, B\}$, $K_2 = \{A, D\}$, $K_3 = \{C\}$ là các khóa. Vậy R không có thuộc tính thứ cấp, nên W là 3NF.

Kết luận:

Nếu sơ đồ quan hệ $W = < R, F >$ mà R không chứa thuộc tính thứ cấp thì W là 3NF.

2.6.4 - DẠNG CHUẨN BOYCE – CODD (BCNF)

Dạng chuẩn tiếp theo ta sẽ xét là dạng chuẩn Boyce - Codd ký hiệu là BCNF.

Cho sơ đồ quan hệ W = < R, F >

Ta nói W là BCNF nếu trong W không tồn tại PTH dạng $X \rightarrow x$ với $x \notin X$ và $X^+ \neq R$.

Từ định nghĩa ta thấy rằng nếu W là BCNF thì nó là 3NF. Vì trong 3NF *ta chỉ cấm các thuộc tính thứ cấp* không phụ thuộc vào tập có bao đóng khác R, còn trong BCNF *ta cấm tất cả các thuộc tính* không phụ thuộc vào tập có bao đóng khác R.

Thí dụ 2.32:

Cho W = < R, F >, với R = {A, B, C, D}

F = {AB → C, C → ABD}

Dễ dàng thấy rằng:

Các tập có bao đóng khác R là X = {A} hoặc X = {B} hoặc X = {D} hoặc X = {A, D} hoặc X = {B, D} và trong các tập trên không có PTH dạng $X \rightarrow x$ với $x \notin X$.

Vậy W là BCNF.

Thí dụ 2.33:

Sau đây ta sẽ xét một sơ đồ quan hệ W mà W là 3NF nhưng W không là BCNF.

Cho W = < {A,B,C,D}, {A →BCD, BC → DA, B →C}> . Rõ ràng rằng W là 3NF vì W có hai khoá là {A} và {B,C} nên tập không khoá là D và không có tập nào có bao đóng khác R kéo theo thuộc tính thứ cấp D. Ngược lại W không là BCNF vì có phụ thuộc hàm B→ C mà B^+ khác R.

Nhận xét: Nếu W= <R,F> là1NF,2NF,3NF,BCNF,...thì mọi quan hệ r trên R cũng là 1NF, 2NF, 3NF, BCNF, . . . tương ứng. Tất nhiên mới chỉ vài quan hệ r trên R là 2NF, 3NF, BCNF, ... thì chưa khẳng định được W=< R, F> là các dạng chuẩn tương ứng.

Cũng từ định nghĩa suy ra rằng, cho sơ đồ quan hệ W=< R, F >

a - Muốn xét xem W là 2NF hay không ta phải :

• Tính tất các các khóa K của W và suy ra đồng thời tập thuộc tính thứ cấp F_n

• Xét xem có K'→ x không (K' tập con thực sự của khóa và x thuộc F_n).

Thí dụ, cho W = < R, F > với R = {A, B, C, D, E, G} và F = {A→ BC, C → DE, E → G}. Khi đó ta thấy mọi khóa K phải chứa A và hơn thế nữa tập {A} là khóa nên ta có ngay tập thứ cấp là {B,C,D,E,G} và W là 2NF vì các tập khóa chỉ có một phần tử.

b - Muốn xét xem W là 3NF hay không ta phải:

• Tính tập thuộc tính thứ cấp F_n ={x,... };

• Xét xem có X → x (x∉ X và $X^+ \neq R$).

Thí dụ, W = < R, F >, R = {A, B,C,D,E,G,H}

F = {C→ AB, D → E, B → G}

Ta thấy mọi khóa K đều phải chứa H, C, D và tập ba phần tử này là khóa. Vậy ta có tập các thuộc tính thứ cấp {A,B,E,G}. Từ F ta thấy ngay rằng không có tập X mà bao đóng khác R kéo theo thuộc tính thứ cấp. W là 3NF.

c - Muốn xét xem W là BCNF hay không ta phải:

• Xét xem có X→ a với a∉X và $X^+ \neq R$.

Thí dụ, R= {A, B, C, D, E, G, H}, F= {A →BC, D →E, H → G}. Ta thấy tập con có bao đóng khác R mà kéo theo thuộc tính khác: D → E. Vậy W không là BCNF.

Định lý 2.5:

Các lớp dạng chuẩn của các sơ đồ quan hệ có quan hệ lồng nhau, lớp sau nằm trong lớp trước. Nghĩa là ta có:

!NF ⊃ 2NF ⊃ 3NF ⊃ BCNF . Lồng nhau ở đây là thực sự, nghĩa là lớp sau nằm gọn trong lớp trước.

Thật vậy với R = {A,B,C,D} và F_1 = {AB→C, D→B, C→ABD} thì W_1= < R, F_1 > là 3NF nhưng không là BCNF (vì không có tập X mà có bao đóng khác R nhưng lại kéo theo thuộc tính thứ cấp, tập các thuộc tính thứ cấp bằng rỗng) nên W là 3NF. Ngược lại, W không là BCNF vì nó chứa tập X = D có bao đóng khác R và kéo theo B. Còn W_2 = < R, F_2 > với F_2 = {B → D, A → C, C → ABD} là 2NF, nhưng không là 3NF (vì các thuộc tính thứ cấp B, D phụ thuộc hoàn toàn vào khóa nên nó là 2NF, ngược lại có tập X = {B} có bao đóng khác R nhưng kéo theo thuộc tính thứ cấp D nên nó không là 3NF), và rất nhiều sơ đồ quan hệ là 1NF nhưng không là 2NF.

2.7. PHỤ THUỘC ĐA TRỊ VÀ DẠNG CHUẨN 4NF

Trước khi trình bày các dạng chuẩn tiếp theo, hãy lưu ý rằng lớp các quan hệ chúng ta đã và đang xét rất lớn, một số các quan hệ có ngữ nghĩa (semantic) phức tạp, trong tập các thuộc tính không có PTH hoặc có các phụ

thuộc đặc biệt. Vậy để đi sâu nghiên cứu các đặc thù, tính chất của lớp các quan hệ, chúng ta sẽ trình bày tiếp khái niệm phụ thuộc đa trị.

Thí dụ 2.34:

Ta xét bảng r thông báo chủ và xe như sau :

r

Chủ	Xe	Biển
A	BMW	29F1
A	BMW	29F2
A	BMW	29F3
B	Toyota	29H1
B	Toyota	29H2

Trong quan hệ này không có phụ thuộc hàm giữa Chủ và Biển, nhưng giữa các thuộc tính Chủ, Biển, có mối quan hệ đặc biệt. Ví dụ ta thấy nếu cùng một Chủ (chiếu lên Chủ của bộ t tức t.Chủ = A hoặc B) thì tráo hai biển bất kỳ cho nhau (tráo t.Biển cho nhau)ta vẫn được một xe hợp lệ của chủ đó . Kiểu phụ thuộc đặc biệt này S. Jajda gọi là phụ thuộc đa trị.

Vậy ta có khái niệm phụ thuộc đa trị trong các lược đồ quan hệ như sau.

2.7.1.ĐỊNH NGHĨA PHỤ THUỘC ĐA TRỊ

Trong các công trình của mình S. Jajda đã nêu định nghĩa phụ thuộc đa trị như sau:

Cho lược đồ quan hệ $R = \{A_1, A_2, ... A_n\}$, $n \geq 3$. X và Y là các tập con của R và $Z = R - (X \cup Y)$. Mỗi phần tử t (bộ t) của quan hệ r trên R ta có thể coi như ghép của 3 phần chiếu của t lên các tập thuộc tính X, Y, Z tương ứng tức là t = t.Xt.Yt.Z.

Ta nói trong lược đồ quan hệ R *xác định phụ thuộc đa trị từ X lên Y* (X xác định đa trị Y), ký hiệu là $X \rightarrow\rightarrow Y$, *nếu mọi cặp phần tử* t_1, t_2 *của r bằng nhau trên tập X*, tức:

$$t_1 = t_1.Xt_1.Yt_1.Z \in r$$

$$t_2 = t_1.Xt_2.Yt_2.Z \in r \qquad (1)$$

và $t_1.X = t_2.X$ *thì khi đổi đuôi* t_1*và* t_2 *cho nhau* ($t_1.Z$ *và* $t_2.Z$) *chúng ta vẫn nhận được các phần tử thuộc r*, tức là:

$$t_1' = t_1.Xt_1.Yt_2.Z \text{ và}$$

$$t_2' = t_1.Xt_2.Yt_1.Z \qquad (2)$$

cũng thuộc r với r là một quan hệ bất kỳ trên R.

Nói ngắn gọn lại là ta có $X \rightarrow\rightarrow Y$ *nếu với mọi* t_1,t_2 *như trong (1) thuộc r và* $t_1.X = t_2.X$ *thì ta cũng có* t_1',t_2' *như trong (2) thuộc r.*

Thí dụ 2.35. a

Xét quan hệ Chủ- Xe r như trên ta có ngay:

Chủ $\rightarrow\rightarrow$ Xe và Chủ $\rightarrow\rightarrow$ Biển

Thí dụ 2.35.b:

Một cách tổng quát cho lược đồ quan hệ R và hai tập thuộc tính X, Y.

Nếu $X \rightarrow Y$ thì $X \rightarrow\rightarrow Y$. Thật vậy giả sử t_1 và t_2 thuộc r và $t_1.X = t_2.X$ vì $X \rightarrow Y$, nên $t_1.Y = t_2.Y$. Khi đó $t_1' = t_1.Xt_1.Yt_2.Z = t_2.Xt_2.Yt_2.Z = t_2$ thuộc r. Tương tự ta có $t_2' = t_1$ thuộc r . Nên $X \rightarrow\rightarrow Y$

2.7.2.TÍNH CHẤT CỦA PHỤ THUỘC ĐA TRỊ

Sau đây chúng tôi xin nêu một vài tính chất cơ bản nhất của các phụ thuộc đa trị.

Tính chất d1: Tính bù của phụ thuộc đa trị:

Nếu X, Y, Z là ba tập con rời nhau của lược đồ quan hệ $R = \{A_1, A_2, \ldots, A_n\}$ và $R = X \cup Y \cup Z$ thì

$X \rightarrow\rightarrow Y$ khi và chỉ khi $X \rightarrow\rightarrow Z$.

Tính chất d2: Tính tăng trưởng:

Nếu $X \rightarrow\rightarrow Y$ và $V \subset W$ thì $XW \rightarrow\rightarrow YV$.

Tính chất d3: Tính bắc cầu:

Nếu $X \rightarrow\rightarrow Y$ và $Y \rightarrow\rightarrow V$ thì $X \rightarrow\rightarrow V \setminus Y$.

Tính chất d4: Tính pha trộn:

Nếu $X \rightarrow Y$ thì $X \rightarrow\!\!\!\!\rightarrow Y$.

Nói cách khác phụ hàm là trường hợp riêng của phụ thuộc đa trị.

Tính chất d5: Nếu $X \rightarrow\!\!\!\!\rightarrow Y$ và $W \rightarrow V$ với $V \subset Y$ và $W \cap Y = \varnothing$ thì $X \rightarrow V$.

Và một số tính chất có thể suy dẫn được:

Tính chất 6: (Hợp) $X \rightarrow\!\!\!\!\rightarrow Y$ và $X \rightarrow\!\!\!\!\rightarrow Z \Rightarrow X \rightarrow\!\!\!\!\rightarrow YZ$.

Tính chất 7: (Tựa bắc cầu)

$X \rightarrow\!\!\!\!\rightarrow Y$ và $YW \rightarrow\!\!\!\!\rightarrow V \Rightarrow XW \rightarrow V - YW$.

Tính chất 8: (Tựa hợp)

$X \rightarrow\!\!\!\!\rightarrow Y$ và $XY \rightarrow W$, thì $X \rightarrow W - Y$.

Tính chất 9: Tính phân rã

Nếu $X \rightarrow\!\!\!\!\rightarrow Y$ và $X \rightarrow\!\!\!\!\rightarrow V$ thì

$X \rightarrow\!\!\!\!\rightarrow Y \cap V$;

$X \rightarrow\!\!\!\!\rightarrow Y - V$;

$X \rightarrow\!\!\!\!\rightarrow V - Y$.

Về sau ta thường ký hiệu phụ thuộc đa trị là MD (Multivalued Dependencies), thí dụ cho phụ thuộc đa trị ta viết cho MD $X \rightarrow\rightarrow Y$.

Một MD $X \rightarrow\rightarrow Y$ trên lược đồ R được gọi là *phụ thuộc cơ sở* nếu $Y \neq \emptyset$ và Y không giao với X, tức là $X \cap Y = \emptyset$ và $X \cup Y \neq R$.

Sau đây ta sẽ chứng minh một vài tính chất của phụ thuộc đa trị:

Tính chất d1: Ta có $X \rightarrow\rightarrow Y$ và $Z = R - X - Y$.

Giả sử t_1 và t_2 là hai phần tử của r mà $t_1.X = t_2.X$ và $t_1 = t_1.Xt_1.Yt_1.Z$, $t_2 = t_2.Xt_2.Yt_2.Z$ theo giả thiết $X \rightarrow\rightarrow Y$ nên $t_1' = t_1.Xt_1.Yt_2.Z$, và $t_2' = t_2.Xt_2.Yt_1.Z$ cũng thuộc r, vì $t_1.X = t_2.X$ nên đổi $t_1.X$ và $t_2.X$ cho nhau trong các t_1' và t_2' ta vẫn được các phần tử thuộc r, hay $t_1' = t_2.Xt_1.Yt_2.Z$

$t_2' = t_1.Xt_2.Yt_1.Z$ cũng thuộc r. Theo định nghĩa ta có $X \rightarrow\rightarrow Z$.

Vậy ta có định nghĩa tương đương: $X \rightarrow\rightarrow Y$ nếu t_1 và t_2 mà $t_1.X = t_2.X$ và $t_1 = t_1.Xt_1.Yt_1.Z$ cùng $t_2 = t_2.Xt_2.Yt_2.Z$ thuộc r thì $t_1' = t_1.Xt_2.Yt_1.Z$ và $t_2' = t_2.Xt_1.Yt_2.Z$ cũng thuộc r (thay phần giữa của hai phần tử t_1 và t_2).

Về sau ta sẽ dùng định nghĩa này nhiều hơn. Bạn đọc cần làm quen và nhận dạng nhanh khi nói t_1, t_2 thì ta biết ngay t_1', t_2' tương ứng.

Tính chất d2: Ta có $X \rightarrow\rightarrow Y$ và $V \subset W$, ta phải chứng minh :

$XW \rightarrow\rightarrow YV$.

Thật vậy, giả sử t_1 và t_2 thuộc r mà $t_1.XW = t_2.XW$ và

$t_1 = t_1.XWt_1.YVt_1.Z = t_1.XWt_1.Yt_1.Vt_1.Z$,

$t_2 = t_2.XWt_2.YVt_2.Z = t_2.XWt_2.Yt_2.Vt_2.Z$ và vì $X \rightarrow\rightarrow Y$ nên $t_1' = t_1.XWt_2.Yt_1.Vt_1.Z$ và $t_2' = t_2.XWt_1.Yt_2.Vt_2.Z$ cùng thuộc r (chú ý vì $t_1.XW = t_2.XW$ nên $t_1.X = t_2.X$) mà $V \subset W$ nên $V \subset XW$ và do $t_1.XW = t_2.XW$ nên $t_1.V = t_2.V$, thay $t_1.V$ và $t_2.V$ vào t_1' và t_2' ta có $t_1' = t_1.XWt_2.Yt_2.Vt_1.Z = t_1.XWt_2.YVt_1Z$ và $t_2' = t_2.XWt_1.Yt_1.Vt_2.Z = t_2.XWt_1.YVt_2.Z$ cùng thuộc r.

Vậy $XW \rightarrow\!\!\!\!\rightarrow YV$.

Tính chất d3:

Ta sẽ chứng minh tính chất 3 bằng phản chứng.

Giả sử kết luận của mệnh đề không thỏa mãn, nghĩa là tồn tại một quan hệ r mà trên đó $X \rightarrow\!\!\!\!\rightarrow Y$ và $Y \rightarrow\!\!\!\!\rightarrow V$ thỏa mãn nhưng: $X \rightarrow\!\!\!\!\rightarrow V \setminus Y$ không thỏa mãn. Nghĩa là trên r tồn tại hai bộ t_1, t_2 mà:

$t_1.X = t_2.X$ và $t_1' = t_1.Xt_2.(V \setminus Y)t_1.(R \setminus X \setminus (V \setminus Y)) \notin r$

Theo giả thiết $X \rightarrow\!\!\!\!\rightarrow Y$ và từ $t_1.X = t_2.X$ ta có $t_1'' = t_1.Xt_2.Yt_1.(R \setminus X \setminus Y) \in r$. Ta dễ dàng nhận thấy rằng : $t_1''.Y = t_2.Y$ và vì $Y \rightarrow\!\!\!\!\rightarrow V$ nên ta có thể thay chiếu của V trong t_2 và t_1'' để được các phần tử thuộc r, tức là $t_1''' = t_1.Xt_2.Yt_2.Vt_1.(R \setminus X \setminus Y \setminus V) \in r$.

Ta sẽ chứng minh $t_1' = t_1'''$. Rõ ràng trên tập thuộc tính $X \cup (V \setminus Y)$ thì t_1' và t_1''' bằng nhau. Bây giờ ta chỉ ra rằng t_1' và t_1''' bằng nhau trên phần còn lại, tức trên $R \setminus X \setminus (V \setminus Y)$, mà $t_1'''.(R \setminus X \setminus (V \setminus Y)) = t_2.Yt_2.(V \cap Y)t_1.(R \setminus X \setminus Y \setminus V) = t_2.Yt_1.(R \setminus X \setminus Y \setminus V) = t_1'.(R \setminus (V \setminus Y))$.

Tương tự các bạn có thể tiếp tục chứng minh các tính chất còn lại của phụ thuộc đa trị.

2.7.3.HỆ TIÊN ĐỀ CỦA CÁC RÀNG BUỘC FD (PTH) VÀ MD

Gọi FD là lớp tất cả các phụ thuộc hàm khi đó ta đã có hệ tiên đề Armstrong cho lớp FD:

A1 tính phản xạ: X→ X và nếu Y ⊂ X thì X → Y.

A2 tính bắc cầu: X → Y và Y → V thì X → V.

A3 tính tăng trưởng (mở rộng): X → Y thì XZ → YZ.

Trong lớp MD, hệ năm tính chất d1,d2,d3,d4,d5 gọi là *hệ tiên đề của lớp MD.*

Tổng quát: *Hệ tám tiên đề {A1,A2.A3,d1,d2,d3,d4,d5} gọi là hệ tiên đề của các ràng buộc dạng phụ thuộc hàm và phụ thuộc đa trị FD ∪MD.*

Định lý 2.6:

Hệ tiên đề {A1, A2, A3, d1, d2, d3, d4, d5} *đúng đắn và đầy đủ trong lớp các ràng buộc FD ∪MD.*

Từ nay khi nói cho tập ràng buộc Đ ta ngầm chỉ rằng trong Đ có chứa các ràng buộc kiểu phụ thuộc hàm và các phụ thuộc đa trị. Thí dụ

Đ = {A → B, G →→ C, E → F, G →E}.

Tương tự bao đóng của Đ, ký hiệu $Đ^+$ là tập tất cả các ràng buộc được suy dẫn từ Đ. Thí dụ cho Đ = {A → BCE } thì $Đ^+$ = {A →B,A→C,A →E,

A →BC,... , A →→ B, A →→ C,... }. Vậy chúng ta thấy rằng tập Đ có rất ít phần tử nhưng tập Đ⁺ có thể rất lớn.

2.7.4.DẠNG CHUẨN 4NF

Cho sơ đồ quan hệ W= < R, Đ >. Ta nói *W là 4NF* nếu mọi MD X →→ Y ∈ Đ⁺ là phụ thuộc cơ sở thì X⁺ = R. Nói một cách khác *W là 4NF nếu mọi phụ thuộc đa trị thuộc tập bao đóng của Đ: X →→ Y ∈ Đ⁺ mà Y khác rỗng, Y không nằm trong X, XY ≠ R thì X⁺ = R.*

Thí dụ 2.36:

W = < R, Đ >, với R = {A, B, C, E, G}, Đ = {A →BCEG} thì W là 4NF vì mọi phụ thuộc đa trị X →→ Y ∈ D⁺ đều thỏa mãn định nghĩa của dạng chuẩn 4.

Từ định nghĩa ta có ngay kết luận sau đây:

Kết luận:

• Nếu W là 4NF thì W là BCNF.

Thật vậy giả sử W = < R, Đ > không là BCNF, có nghĩa là trong W có PTH dạng X → A ∉ X và X⁺ ≠ R, vậy trong Đ có X→→ A ∈ Đ⁺ là phụ thuộc cơ sở nhưng X⁺ ≠ R, suy ra vô lý.

Vậy W là BCNF.

• Muốn xét xem sơ đồ quan hệ W= < R, F > có là 4NF hay không ta chỉ cần xét xem trong Đ⁺ có phụ thuộc cơ sở X→→ A ∈ Đ⁺ và X⁺ ≠ R ?

• Nếu Đ' = ∅ thì W là 4NF.

2.8. PHỤ THUỘC KẾT NỐI VÀ DẠNG CHUẨN 5NF

Tính chất nối các quan hệ mà không tổn thất thông tin là một trong các tính chất tạo điều kiện thuận lợi cho việc thiết kế cơ sở dữ liệu. Tuy nhiên, khi nghiên cứu tập các quan hệ, nếu chỉ dùng các công cụ phụ thuộc hàm, phụ thuộc đa trị chúng ta chưa thể xét hết các đặc thù cần thiết của các quan hệ. Để tiếp tục đi sâu nghiên cứu lớp các quan hệ chúng ta sẽ trình bày thêm khái niệm ràng buộc kiểu khác như phụ thuộc kết nối, phụ thuộc suy rộng. Quan hệ r và CSDL liên quan bài toán quản lý chủ và xe sau đây sẽ cho ta một minh hoạ về việc nếu chỉ dừng lại ở các ràng buuọc đã xét chưa đủ để nghiên cứu tiếp các vấn đề liên quan.Ta có bảng theo dõi chủ xe, mác xe cùng màu xe, mỗi chủ có thể có nhiều xe cùng các mác và các màu khác nhau.

Thí dụ 2.37:

Quan hệ chủ, xe cùng các màu và mác khác nhau

CHU	MAU	MAC
Kiệt	Đen	Honda
Kiệt	Đen	Toyota
Kiệt	Đỏ	Toyota
Mười	Đen	Toyota

Quan hệ trên đây là 4NF. Ta có thể tách quan hệ trên thành ba quan hệ sau: r_1 là quan hệ CHU và MAU xe, r_2 là quan hệ CHU và MAC xe, r_3 là quan hệ MAU và MAC xe.

r_1		r_2		r_3	
TÊN	MAU	TÊN	MAC	MAU	MAC
Kiệt	Đen	Kiệt	Honđa	Đen	Honđa
Kiệt	Đỏ	Kiệt	Toyota	Đen	Toyota
Mười	Đen	Mười	Toyota	Đỏ	Toyota

Ta thấy một điều lý thú là nối hai bất kỳ trong ba quan hệ trên không cho ta quan hệ ban đầu. Như vậy phép tách đã làm " tổn thất " thông tin?

Để nghiên cứu và giải quyết những vấn đề của lớp các quan hệ tương tự trên đây, năm 1979, Aho và Nicolas đã nêu được một vài đóng góp với ý tưởng sau:

Nếu chỉ dừng lại ở các phụ thuộc đa trị, chúng ta chưa đủ công cụ để giải quyết được tất cả các các trường hợp dư thừa và tách không tổn thất thông tin trong một lớp khá lớn các quan hệ.

Sau đây chúng ta sẽ xét khái niệm phụ thuộc kết nối.

2.8.1.ĐỊNH NGHĨA PHỤ THUỘC KẾT NỐI

Cho $R = \{A_1, A_2, A_3, \ldots A_n\}$ là lược đồ quan hệ; r là quan hệ trên R; X_1, $X_2, \ldots X_m$ là các tập con của R và $R = X_1 \cup X_2 \cup \ldots \cup X_m$. Ta nói rằng *có phụ thuộc kết nối* giữa các $X_1, X_2, \ldots X_m$ và ta ký hiệu $*\{X_1, X_2, \ldots, X_m\}$ nếu r là nối của các chiếu của r lên các tập $X_1, X_2, \ldots, X_m$ tương ứng. Nói cách khác r có nối không mất tin trên các $X_1, X_2, \ldots X_m$, tức là:

$$r = r.X_1 \mid >< \mid r.X_2 \ldots \mid >< \mid r.X_m.$$

Dạng chuẩn 5 - 5NF gắn với các tập khóa.

2.8.2 .ĐỊNH NGHĨA DẠNG CHUẨN 5NF

Cho lược đồ quan hệ R = {A_1, A_2, A_3,... A_n}.

r là một quan hệ trên R.

Ta nói rằng r là (ở) *dạng chuẩn 5, ký hiệu là 5NF*, khi và chỉ khi tất cả các phụ thuộc kết nối thực hiện bởi các khóa.

Nhận xét:

Trong lớp các quan hệ ở dạng 5NF còn nhiều vấn đề về lý thuyết cũng như thực tiễn chúng ta còn phải thực sự quan tâm và đầu tư thời gian.

Tuy nhiên McFadden và Fred cũng đã nêu định nghĩa dạng chuẩn 5 như sau: Sơ đồ quan hệ W = < R, F > là *dạng chuẩn 5*, ký hiệu là 5NF, nếu W là 4NF và trong W không có phụ thuộc kết nối .

Từ định nghĩa ta thấy ngay rằng nếu *W là 5NF thì W là 4NF.*

2.9. DẠNG CHUẨN DK/NF

Fagin (1981) đã nêu một dạng chuẩn gọi là dạng miền khóa chuẩn. Ta nói quan hệ r là (ở) *dạng chuẩn DK/NF* nếu và chỉ nếu mỗi ràng buộc trong r là kết quả logic của các ràng buộc khóa và ràng buộc miền. Fagin đã chỉ ra rằng nếu r là dạng chuẩn DK/ NF thì r là 5NF, 4NF, ...

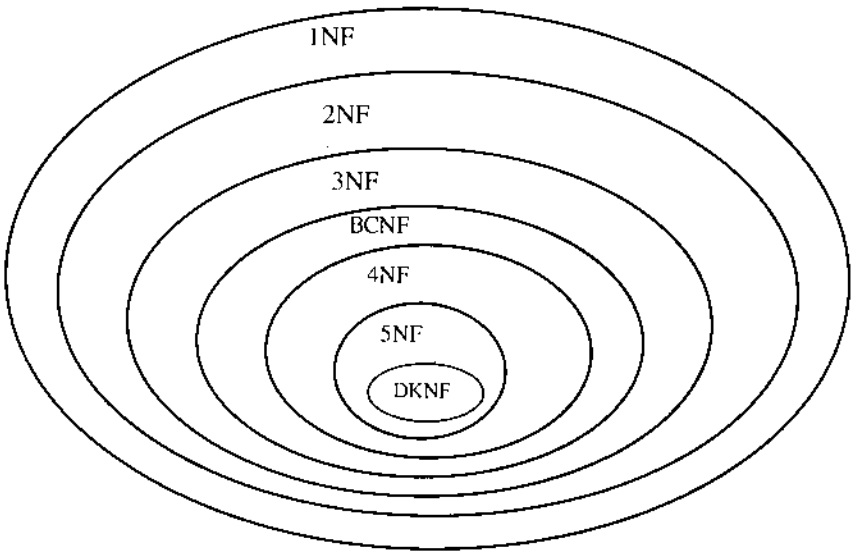
Từ định nghĩa chúng ta thấy rằng điều kiện của các dạng chuẩn được thắt dần vào. Trong quá trình nghiên cứu và xét các dạng chuẩn chúng ta đã xét dần lớp các quan hệ. Đầu tiên là dạng chuẩn1: 1NF đây là lớp quan hệ hầu như chứa hết các quan hệ mà chúng ta quan tâm (mọi quan hệ có thể đư-

a về dạng chuẩn 1). Dạng chuẩn 2 là các quan hệ dạng chuẩn 1 và thêm điều kiện mọi thuộc tính thứ cấp phụ thuộc hoàn toàn vào khóa...

Như vậy lớp các quan hệ ở các dạng chuẩn sau lồng trong các dạng chuẩn trước nó.

Định lý 2.7:

Trong các lớp của các dạng chuẩn ta có mối quan hệ lồng nhau thực sự như sau: DK/NF ⊂ 5NF ⊂ 4NF ⊂ BCNF ⊂ 3NF ⊂ 2NF ⊂ 1NF (lồng nhau thực sự, nghĩa là lớp trong bé hơn lớp ngoài).



Hình 2.1 Sơ đồ biểu thị mối liên hệ của các lớp chuẩn

Trong các phần trước chúng ta đã nêu một số thí dụ để minh họa sự lồng nhau nhưng không bằng nhau của các lớp đó. Tuy nhiên các bạn có thể chứng minh và cho các thí dụ khác.

CÂU HỎI VÀ BÀI TẬP

2.1. Cho hai quan hệ r và s như sau:

r

A	B	C	D
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1

s

A	B	C	D
2	1	1	1
2	2	1	1
1	1	1	0
x	y	z	v

a- Tính r - s và s - r.

b- Tính r + s.

c- Tính r * s.

d- Giả sử X = {A, B, D}, Y = {A, C, D}. Tính các quan hệ chiếu r. X, r. Y và s. X, s. Y, (r+s). X, (r+s). (X∪ Y).

e- Chứng minh rằng với mọi quan hệ r, s, q thì ta luôn có

r * s = s * r và r + s = s + r (tính giao hoán)

r * (q + s) = (r * q) + (r * s) (tính kết hợp)

(r + s). X = r. X + s. X

(r * s). X = r. X * s. X

2.2. Cho hai quan hệ r và s như sau:

a- Hai quan hệ giống nhau:

r

A	B	C	D
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1

s

A	B	C	D
0	0	0	1
0	0	1	1
0	1	1	1
1	1	1	1

Tính r |><| s, r |>< s, s |>< r .

b- Hai quan hệ hoàn toàn khác nhau trên cùng một lược đồ:

r

A	B	C	D
0	0	0	0
0	0	1	1
0	1	1	1

s

A	B	C	D
a	b	c	d
x	y	z	v

Tính r |><| s.

c- Hai quan hệ có tập các thuộc tính lồng nhau:

r

A	B	C	D
0	1	1	1
x	y	z	v

s

C	D
1	1
0	0
z	v
v	z

Tính r |><| s, r |>< s, s |>< r.

d- Hai quan hệ lồng nhau:

r

A	B	C	D
0	0	1	1
1	1	1	1
1	1	0	1

s

C	D
1	1
0	1

Tính r |><| s, Tính r |>< s và tính r $|><|_{2=2}$ s, r $|><|_{1<2}$ s, s $|><|_{1<2}$ r.

e- Cho hai quan hệ r(TT, TÊN, NS, GT) và s(QUÊ, NH, ĐIÊM) như sau:

r

TT	TÊN	NS	GT	QUÊ	NH	ĐIÊM
1	Linh	77	Nữ	HN	Anh	18
2	Quyên	76	Nữ	HF	Hóa	20
3	Nam	75	Nam	SG	Toán	22
4	Tuấn	74	Nam	VF	Tin học	22

s

Hãy dùng các thủ thuật nhỏ và sử dụng các phép toán quan hệ để có quan hệ kết quả DS như sau:

DS

TT	TÊN	NS	GT	QUÊ	NH	ĐIÊM
1	Linh	77	Nữ	HN	Anh	18
2	Quyên	76	Nữ	HF	Hóa	20
3	Nam	75	Nam	SG	Toán	22
4	Tuấn	74	Nam	VF	Tin học	22

Một cách tổng quát cho hai quan hệ r và s như sau:

r

A	B	C	D
a_1	b_1	c_1	d_1
a_2	b_2	c_2	d_2
a_3	b_3	c_3	d_3
a_4	b_4	c_4	d_4

s

E	F	G
e_1	f_1	g_1
e_2	f_2	g_2
e_3	f_3	g_3
e_4	f_4	g_4

Hãy tìm cách sử dụng các phép toán quan hệ và các thủ thuật để đưa về quan hệ kq:

kq

A	B	C	D	E	F	G
a_1	b_1	c_1	d_1	e_1	f_1	g_1
a_2	b_2	c_2	d_2	e_2	f_2	g_2
a_3	b_3	c_3	d_3	e_3	f_3	g_3
a_4	b_4	c_4	d_4	e_4	f_4	g_4

2.3. Cho quan hệ r như sau:

r

A	B	C	D	E
0	0	1	1	0
1	1	0	0	0
1	1	1	0	0
2	2	0	0	0
1	1	1	1	1

Mệnh đề E: tổng giá trị của dòng < 3 (nhỏ hơn 3). Viết các quan hệ chọn r (E) và r (không E).

2.4. Cho hai quan hệ r và s như sau:

r

A	B	C
0	0	0
1	1	1
1	1	0

s

D	E
5	6
0	6

Tính tích Decac của r và s: r x s.

2.5. Cho hai quan hệ r và s như sau:

r					s	
A	B	C	D	E	D	E
0	0	0	0	1	0	1
0	0	1	1	0	1	1
1	1	1	1	1	1	0
0	0	0	1	1		

Tính r ÷ s.

2.6*

a- Chứng minh rằng: năm (5) phép toán cơ bản của đại số quan hệ hợp, hiệu, Decac, chiếu, chọn là độc lập với nhau, nghĩa là không một phép toán nào trong chúng có thể biểu diễn qua các phép còn lại.

b- Chứng minh rằng các phép toán còn lại của đại số quan hệ như giao, nối, nối nửa, nối theo θ, thương, đều có thể nhận được từ các phép toán cơ bản trên (thí dụ xem bài tập 2.7).

2.7. Cho r và s là hai quan hệ trên các lược đồ tương ứng

$R = \{A_1, A_2, \ldots, A_n\}$, $S = \{A_1, A_2, \ldots, A_k\}$ với $k < n$.

Giả sử $X = R - S = \{A_{k+1}, \ldots, A_n\}$. Chứng minh rằng:

$r \div s = r.X - ((r.X \times s) - r).X$.

2.8.

a- Cho lược đồ quan hệ R và tập PTH

$F = \{AB \rightarrow E, AG \rightarrow I, BE \rightarrow I, E \rightarrow G, GI \rightarrow H\}$ trên R.

Chứng minh rằng $AB \rightarrow GH$.

b- Tương tự cho tập PTH F như sau :

$F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH, G \rightarrow A\}$.

Chứng minh rằng : $AB \rightarrow E$, $AB \rightarrow G$.

2.9. Cho sơ đồ quan hệ W = < R, F >. Chứng minh (giải thích) rằng: với mọi tập con X bất kỳ của R và mọi phần tử A thuộc tập X thì $X \rightarrow A \in F^+$. Tức là:

$\forall A \in X \subset R \Rightarrow X \rightarrow A \in F^+$.

2.10. Cho sơ đồ quan hệ W = < R, F >, với R = {A, B, C, D} và F = {A $\rightarrow$ B, A $\rightarrow$ C}. Hãy tìm các PTH suy được từ các quy tắc của PTH trong các ràng buộc sau :

a- $A \rightarrow D$.

b- $C \rightarrow D$.

c- $AB \rightarrow B$.

d- $BC \rightarrow A$.

e- $A \rightarrow BC$.

2.11. Cho sơ đồ quan hệ W = < R, F >, với R = {A, B, C, D } và F = {A $\rightarrow$ B, BC $\rightarrow$ D}. PTH nào trong dãy sau là suy được từ F bằng các quy tắc của PTH:

a- $C \rightarrow D$.

b- $A \rightarrow D$.

c- $AD \rightarrow C$.

d- $BC \rightarrow A$.

e- $B \rightarrow CD$.

2.12. Cho bảng quan hệ r:

r

A	B	C	D
x	u	x	y
y	x	z	x

z	y	y	y
y	z	w	z

Trong các PTH sau đây PTH nào không thỏa mãn r:

$A \to B$, $A \to C$, $B \to A$, $C \to D$, $D \to C$, $D \to A$.

2.13. Cho quan hệ r như sau:

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_1	b_2	c_2	d_2	e_1
a_2	b_1	c_3	d_3	e_1
a_2	b_1	c_4	d_3	e_1
a_3	b_2	c_5	d_1	e_1

Tìm tập PTH F thỏa mãn r.

2.14. Cho hai lược đồ quan hệ R_1 và R_2, $R_1 \cap R_2 = X$.

Chứng minh rằng nếu quan hệ r trên tập thuộc tính $R_1 \cup R_2$ thỏa mãn $X \to R_2$ thì $r = r.R_1 |><| r.R_2$.

2.15. Cho sơ đồ quan hệ $W = < R, F >$,

với $R = \{A, B, C, D, E, G, H\}$ và tập các PTH F:

$F = \{A \to D, AB \to DE, CE \to G, E \to H\}$. Tính $(AB)^+$.

2. 16 Trong thuật toán tìm bao đóng X^+ ta đã xây dựng dãy

$X^0 \subset X^1 \subset X^2 ... \subset X^i ... \subset$ với

$X^0 = X$

$X^{i+1} = X^i Z^i$ và

$Z^i = \{A \in R: A \notin X^i$ và $X^i \to A \in F^+\}$.

a- Xét giao của các Z^i: $Z^i \cap Z^k = ?$

b- Chứng minh rằng: $\forall i\ Z^i \subset (X^i)^+$.

2.17.

a- Tìm các khóa còn lại của W = < R, F > trong thí dụ 2. 21 và tìm các khóa của sơ đồ quan hệ W= < R, F > trong bài tập 2. 16.

b- Cho sơ đồ quan hệ W = < R, F >, với R = {A,B,C,D,E,H} và

F = {A→ E, C → D, E →DH}. Chứng minh rằng K = {A,B,C} là khóa duy nhất của W.

c- Cho sơ đồ quan hệ W = < R, F >, với R = {A,B,C,D} và

F = {AB → C, D → B, C → ABD}. Tìm các khóa của W.

d- Cho sơ đồ quan hệ W = < R, F >, với R = {A,B,C,D,E,G} và

F = {AB → C, C → A, BC → D, ACD → B, D → EG,BE →C, CG → BD, CE → CG}. Tìm các khóa của W.

2.18. Cho lược đồ quan hệ R = $\{A_1, A_2, \ldots A_n\}$. Họ S các tập con của R được gọi là *hệ Sperner* nếu trong S không có hiện tượng lồng nhau, nghĩa là không có $X \subseteq Y$ với $X, Y \in S$. Gọi Γ là họ tất cả các khóa của sơ đồ quan hệ W = < R, F >.

Chứng minh rằng: Γ là hệ Sperner.

2.19 * Cho hệ Sperner không rỗng Γ trên R. Chứng minh rằng tồn tại một sơ đồ quan hệ W để Γ là tập các khóa của W. Vậy ta có thể chứng minh mệnh đề Γ là họ khoá của W= < R, F > khi và chỉ khi Γ là hệ Sperner.

2.20. Cho lược đồ quan hệ R = $\{A_1, A_2, \ldots, A_n\}$.

K là hệ Sperner trên R (tức là K là tập các khóa của một sơ đồ quan hệ trên lược đồ R). Ta gọi tập *phản khóa* của K, ký hiệu là K^{-1}, là tập:

K^{-1} = {X ⊂ R: (∀ k ∈ K) ⇒ (k ⊄ X) và nếu có Z mà (X ⊂ Z) thì

∃ k ∈ K để k ⊆ Z (tức là phản khóa gồm các tập con của R mà không có tập nào của nó chứa trọn một khóa, đồng thời nếu có một tập Z nào đó

chứa thực sự một phần tử của phản khóa thì cũng có một phần tử của họ khóa K nằm gọn trong Z). Chứng minh rằng: K^{-1} là hệ Sperner.

2.21. Giả sử K là hệ Sperner trên R. Chứng minh rằng:

$$\cup K = R - \cap K^{-1}$$

ở đây $\cup$ K và $\cap K^{-1}$ là hợp và giao của các tập trong K và K^{-1} tương ứng.

2.22. Thuật toán tìm khóa của một quan hệ.

Cho lược đồ quan hệ $R = \{A_1, A_2, ... A_n\}$.

r là một quan hệ có m phần tử ta ký hiệu là: $t_1, t_2, ... t_m$; tức là $r = \{t_1, t_2, ... t_m\}$ và mỗi t_i là một dòng.

Nội dung thuật toán:

Input:

$r = \{t_1, t_2, ... t_m\}$ là quan hệ trên R.

Output:

K là tập tất cả thuộc tính khóa của r (K là một khoá của r).

Thuật toán:

Bước 1: Xây dựng họ $E = \{E_{ij}: 1 \le i < j \le m\}$.

Với $E_{ij} = \{A \in R: t_i.A = t_j.A\}$.

Bước 2: Xây dựng họ $M = \{B \in E:$ với mọi $B' \in E: B \not\subset B'\}$.

(Về sau họ M ta gọi là hệ bằng nhau cực đại của r)

a- Chứng minh rằng tập các phần tử khóa $K = R - \cap M$.

b- Tìm các thuộc tính khóa của các quan hệ sau:

r

A	B	C	D	E
1	1	0	0	0
0	1	0	1	1
0	0	1	0	0
2	0	2	0	1
1	0	0	1	0

s

A	B	C	D	E
0	0	1	1	1
0	1	1	2	3
1	1	1	2	1
1	1	3	1	1

2.23. Ta nói tập PTH F là *Phủ* của tập PTH G nếu $F^+ \supset G^+$. Hai tập PTH F và G là *tương đương*, ký hiệu là F ~ G nếu:

$F^+ = G^+$. Chứng minh rằng: F ~ G khi và chỉ khi F phủ G và G phủ F.

2.24. Ký hiệu $X_G{}^+$, $X_F{}^+$ là các tập bao đóng đối với các tập PTH G và F tương ứng. Chứng minh rằng nếu F ~ G thì $X_G{}^+ = X_F{}^+$.

2.25. Cho sơ đồ quan hệ W = < R, F >. Ta nói *W là chính tắc* nếu:

a- Vế phải của mỗi PTH trong F là thuộc tính đơn.

b- Trong tập PTH F không có PTH f (thừa) mà: F - f ~ F.

c- Trong F không có PTH $X \rightarrow A$ mà có $Z \subset X$ và $(F - (X \rightarrow A)) \cup (Z \rightarrow A) \sim F$. Chứng minh rằng với mọi sơ đồ quan hệ W = < R, F > luôn tồn tại một sơ đồ quan hệ chính tắc W' = < R, G > tương đương với W(hai sơ đồ quan hệ là tương đương nếu các tập phụ thuộc hàm của chúng tương đương).

2.26. Thuật toán tìm phủ chính tắc của một sơ đồ quan hệ.

Input: W = < R, F >, với $F = \{f_1, f_2, ... f_m\}$

Output: W' = < R, G > chính tắc và tương đương với W.

Thuật toán:

Bước 1:

$F_0 = F$

$F_i = F_{i-1}$ - fi nếu F_{i-1} - fi tương đương với F_{i-1}, ngược lại $F_i = F_{i-1}$.

i = 1, 2,... , m.

Bước 2:

Loại bỏ các thuộc tính thừa trong vế trái của các PTH của F_m.

Chứng minh rằng tập F_m nhận được chính là tập G.

2.27. Cho sơ đồ quan hệ W = < R, F >, với R = {a, b, c, d, e, g} và F = {ab → c, c → a, bc → d, acd → d, d → eg, be → c, cg → bd, ce → ag}.

Dùng thuật toán trong bài 2. 26 tìm W' = < R, F > chính tắc và tương đương với W.

2.28. Giả sử W = < R, F > là sơ đồ quan hệ, K là họ khóa của W. Đặt M = {k - A: A ∈ k và k là một khóa, tức k ∈ K}.

F_n là tập tất cả các thuộc tính thứ cấp (thuộc tính không khóa).

Đặt $L = \{C^+ : C \in M\}$.

Chứng minh rằng khi đó ta có 3 mệnh đề sau là tương đương:

1- W là 2NF.

2- $\forall X \in L$ thì $X \cap F_n = \emptyset$.

3- $\forall X \in L$ và $B \in F_n$ thì $(X - B)^+ = X - B$

Gợi ý: Độc giả có thể chứng minh thí dụ 1⇒ 2, 2 ⇒ 3, 3 ⇒ 1

Trong phần 3 ⇒ 1 chúng ta lưu ý rằng với mọi tập thuộc tính X mà $X^+ = X$ tức là X không kéo theo một thuộc tính nào không thuộc X. Vậy $\forall X \in L$ và $A \in F_n$. Nếu F_n rỗng thì W - 2NF. Giả sử $F_n \neq \emptyset$. Khi đó

$(X - B)^+ = X - B$ có nghĩa là not ((X - B) → x ∉ (X - B)) tức là not ((k - A)$^+$ - B → x ∉ ((k - A)$^+$ - B)) ⇒ not ((k - A) - B → x ∉ ((k - A) - B)) ⇒ not (k - A) → x ∉ (k - A) ⇒ not (k - A) → B vì B là thứ cấp nên B không thuộc (k - A). Vậy W là 2NF.

2.29. Cho sơ đồ quan hệ W = < R, F >, F_n là tập tất cả các thuộc tính thứ cấp, K là tập các khóa. Đặt Y = {k - F_n: k ∈ K^{-1}}. Chứng minh rằng: W là 2NF khi và chỉ khi ∀ X ∈ Y thì $X^+ = X$.

2.30. Cho sơ đồ quan hệ W = < R, F >. Ta nói trong W có *quan hệ bắc cầu* nếu giữa 2 (hoặc nhiều hơn) thuộc tính thứ cấp có ràng buộc PTH. Chứng minh rằng: W là 3NF nếu trong W không có quan hệ bắc cầu.

2.31. Cho lược đồ quan hệ R = {C, I, D, B, K, F, G, L, M} và tập PTH = {C → IDBKF, D → B, K → F}. Xét xem W thuộc dạng chuẩn nào, 2NF, 3NF, BCNF, 4NF ?

2.32. Xét xem sơ đồ quan hệ sau đây thuộc dạng chuẩn nào

W = < R, F >, với R = {A,B,C,D,E,G,H,I} và

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

2.33. Xét xem sơ đồ quan hệ sau thuộc dạng chuẩn nào: W = <R, F>, với R = {A, B, C, D, E, G, H, I, M} và F = {CB →GH, DE → IMH, CI → CBDH, H → I}.

2.34. Cho danh sách các môn học của lớp học dưới dạng bảng thông báo mỗi môn học một danh sách như sau:

HVKTQS

DANH SACH LOP CAO HOC

Học kỳ 1 - 1999

Cua số (course no.): 350

Tên cua học: Ngoại ngữ

Giáo viên:

Chỗ ở giáo viên:

MSSV	TÊN	MÔN	BẰNG
38214	Hoa	Anh	A
40875	Mơ	Đức	B
51893	Tuấn	Anh	A

Hãy biến đổi (có thể tách) bảng thông báo trên thành các quan hệ ở dạng chuẩn 3NF với điều kiện ngữ nghĩa (semantic):

a- Mỗi giáo viên chỉ có một chỗ ở.

b- Mỗi sinh viên chỉ có một môn học.

c- Mỗi cua chỉ có một tên.

2.35. Cho sơ đồ quan hệ $W = < R, F >$. F_n là các phần tử thứ cấp. Chứng minh rằng: W là 3NF khi và chỉ khi $\forall B \in K^{-1}$, $a \in F_n$ thì $(B - a)^+ = B - a$

2.36. Cho sơ đồ quan hệ $W = < R, F >$. Chứng minh rằng W là 3NF khi và chỉ khi với mọi tập thuộc tính $X \neq R$:

$X^+ = X$, a là thuộc tính thứ cấp và $a \in X$ thì

$(X - a)^+ = X - a$

Gợi ý:

• Chiều thuận, tức là ta có W là 3NF.

Giả sử rằng có a thuộc X và a là thuộc tính thứ cấp mà $(X - a)^+ \neq X - a$, tức có phần tử $b \notin X - a$ và $X - a \rightarrow b$. Có hai trường hợp: nếu b = a thì ta có ngay vô lý vì trong W có tập X - a kéo theo phần tử thứ cấp b mà bao đóng khác R (bao đóng của $X - a \neq R$ vì $X - a \subset X = X^+ \neq R$).

Nếu $b \neq a$ thì vì X - a không chứa b nên X không chứa b và $X \rightarrow b \notin A$, điều này vô lý vì $X^+ = X$, nghĩa là X không kéo theo phần tử nào không thuộc nó.

Vậy điều giả sử là sai nên $(X - a)^+ = X - a$.

• Chiều ngược lại, ta chứng minh tương tự.

2.37. Giả sử r là một quan hệ trên R. Chứng minh rằng r là 3NF khi và chỉ khi với mọi A thuộc E (E là hệ bằng nhau của quan hệ r - xem bài tập 2. 24), a là phần tử thứ cấp thuộc A thì $(A - a)^+ = A - a$

2.38. Cho sơ đồ quan hệ $W = < R, F >$. F_n là các thuộc tính thứ cấp. Chứng minh rằng W là BCNF khi và chỉ khi $\forall B \in K^{-1}$, $a \in B$ thì $(B - a)^+ = (B - a)$.

2.39. Cho lược đồ quan hệ R = {A, B, C, D, E, G}, và tập PTH F = {AB $\rightarrow$ C, C $\rightarrow$ B, ABD $\rightarrow$ E, G $\rightarrow$ A}.

Xét xem $W = < R, F >$ có là BCNF không ?

2. 40 * Cho quan hệ r, E là hệ bằng nhau của r.

Từ E ta lập hệ M gọi là hệ bằng nhau cực đại của r gồm các tập lớn nhất của E, nghĩa là trong E nếu có các tập lồng nhau thì tập lớn thuộc M: $M = \{B \in E: (\text{not } \exists) B' \in E \text{ mà } B \subset B'\}$. Chứng minh rằng r là BCNF khi và chỉ khi với mọi A thuộc M và a thuộc A thì $(A - a)^+ = A - a$

2.41. * Cho sơ đồ quan hệ $W = < R , F >$, trong F không có PTH dạng tầm thường ($X \rightarrow Y$ mà $Y \subset X$). Chứng minh rằng: W là BCNF khi và chỉ khi $A \rightarrow B \in F$ thì $A^+ = R$.

2.42. Hãy xét xem các quan hệ r_1 và r_2 trong thí dụ 2.32 thuộc dạng chuẩn nào ? Thử lại xem r_1, r_2 có là 4NF không ?

2.43* Chứng minh các tính chất 1, 2, 3, 4, 5, 6 của phụ thuộc đa trị MD.

2.44. Cho lược đồ quan hệ $R = \{A_1, A_2, \ldots A_n\}$, $X, Y \subset R$. Chứng minh rằng Nếu $X \rightarrow Y$ thì $X \rightarrow\!\!\!\rightarrow Y$ (nói một cách khác PTH là trường hợp riêng của MD).

2.45. * Ta nói MD $X \rightarrow\!\!\!\rightarrow Y$ trên R là không tầm thường (non trivial) nếu $Y \neq \varnothing$, Y not $\subseteq X$ và $X \cup Y \neq R$.

Cho lược đồ quan hệ R; X, Y, Z là các tập rời nhau và $X \cup Y \cup Z = R$.

Nếu $X \to Y$ hoặc $X \to Z$ thì ta có $X \to\to Y$ (hoặc Z). Khi đó MD $X \to\to Y$ (hoặc Z) ta sẽ gọi là MD **kề** của PTH $X \to Y$ (hoặc Z).

Ta nói rằng phụ thuộc đa trị $X \to\to Y$ là thuần nhất nếu nó không tầm thường và không là kề của bất kỳ PTH nào trên R.

Cho sơ đồ quan hệ $W = < R, F >$. $K = \{K_1, K_2, \ldots, K_m\}$ là tập khóa của W. Chứng minh rằng nếu $Y \cap (\cap K_i) = \emptyset$ và $X \to\to (Y - K_i)$ với i = 1, 2,... m thì MD $X \to\to Y$ không là một MD thuần nhất .

2.46* Giả sử MD $X \to\to Y$ là MD thuần nhất trên R.

$K = \{K_1, K_2, \ldots, K_m\}$ là tập các khóa của R.

Chứng minh rằng nếu $X \to\to (Y - K_i)$, i = 1, 2,... m thì

$Y \cap (\cap K_i) \neq \emptyset$.

2.47. Giả sử $X \to\to Y$ là MD không tầm thường trên lược đồ R và $K_1, K_2, \ldots K_m$ là tập các khóa của R mà: $Y - K_i \neq \emptyset$, i = 1, 2,... m. Chứng minh rằng nếu $Y \cap (\cap K_i) = \emptyset$ và $X \to\to Y \cap K_i$, i = 1, 2,... m thì $X \to\to Y$ là MD không thuần nhất.

2.48* Giả sử $X \to\to Y$ là MD thuần nhất trong lược đồ quan hệ R; $K = \{K_1, K_2, \ldots, K_m\}$ là tập các khóa của R. Chứng minh rằng nếu $X \to\to (Y - \cap K_j)$ thì $Y \cap (\cap K_i) \neq \emptyset$.

2.49* Giả sử $X \to\to Y$ là MD không tầm thường trên lược đồ quan hệ R. $K_1, K_2, \ldots K_m$ là tập các khóa của R mà $Y - K_i \neq \emptyset$, i = 1, 2... m. Chứng minh rằng $X (Y \cap K_i) \to X (Y \cap K_j)$ với $i \neq j$.

2.50* Giả sử $X \to\to Y$ là MD không tầm thường trên R.

K là khóa của R, $Y \cap K \neq \emptyset$.

Chứng minh rằng $X (Y \cap K) \to Y$.

2.51* Giả sử $X \rightarrow\rightarrow Y$ (hoặc Z) là MD thuần nhất trên R.

Chứng minh rằng với khóa K của R thì $K - Y \neq \emptyset$ và $K - Z \neq \emptyset$.

2.52* Chứng minh rằng nếu $X \rightarrow\rightarrow Y$ (hoặc Z) là MD thuần nhất trên R, K là một khóa của R thì K có ít nhất là 3 phần tử, tức là $|K| \geq 3$.

2.53* Chứng minh rằng nếu $X \rightarrow\rightarrow Y$ là MD không tầm thường thì $X \rightarrow Y$ khi và chỉ khi tồn tại khóa K mà $X \rightarrow Y \cap K$.

2.54. Cho lược đồ quan hệ R = {B, D, I, O, S, Q}, tập các ràng buộc {$S \rightarrow\rightarrow D$, $I \rightarrow B$, $IS \rightarrow Q$, $B \rightarrow Q$}. Xét xem W có là 4NF không?

2.55. Cho lược đồ R = {A, B, C, D, E, I} và tập ràng buộc {$A \rightarrow\rightarrow BCD$, $B \rightarrow\rightarrow AC$, $C \rightarrow D$}. W = < R, F > có là 4NF không?

2.56. Gọi U là tập thuộc tính và D là tập phụ thuộc (thuộc một loại bất kỳ) trên tập thuộc tính U. Chúng ta hãy định nghĩa SAT (D) là tập các quan hệ r trên U sao cho r thoả mỗi phụ thuộc trong D. Hãy chứng minh.

a) $SAT(D_1 \cup D_2) = SAT(D_1) \cap SAT(D_2)$

b) Nếu D_1 suy diễn logic được tất cả các phụ thuộc trong D_2 thì

$$SAT(D_1) \supseteq SAT(D_2)$$

2.57. Gọi F là một tập hợp phụ thuộc với các vế phải chỉ có một thuộc tính.

a) Chứng minh rằng nếu lược đồ R có một phụ thuộc vi phạm BCNF $X \rightarrow A$, trong đó $X \rightarrow A$ thuộc F^+ thì tồn tại một phụ thuộc $Y \rightarrow B$ trong chính tập F vi phạm dạng BCNF của R.

b) Chứng minh giống như trên cho dạng chuẩn cấp ba.

2.58. Chứng minh nhận xét sau : Nếu R là một lược đồ quan hệ và $X \subseteq R$ là một khoá của R ứng với tập phụ thuộc F thì X không thể có một vi phạm dạng 3NF ứng với tập phụ thuộc chiếu của F lên X, $\pi_X(F)$.

2.59. Chứng minh rằng không thể có một phụ thuộc được gọi là "*Phụ thuộc hàm gắn kết*" (embedded functional dependency). Nghĩa là nếu $S \subseteq R$ và $X \rightarrow Y$ đúng trong $\pi_S(R)$ thì $X \rightarrow Y$ đúng trong R.

2.60. *Phụ thuộc bao hàm đơn ngôi (*unary inclusion dependency*)* $A \subseteq B$ trong đó A, B là các thuộc tính (có thể từ các quan hệ khác nhau) khẳng định rằng trong những giá trị hợp lệ của các quan hệ, mỗi giá trị xuất hiện trong cột của A cũng xuất hiện trong cột của B. Chứng tỏ rằng các tiên đề sau là đúng đắn và đầy đủ đối với các phụ thuộc bao hàm đơn ngôi.

a) $A \subseteq A$ với mọi A

b) Nếu $A \subseteq B$ và $B \subseteq C$ thì $A \subseteq C$

2.61. Giả sử với số chẵn *n* chúng ta có các thuộc tính $A_1,\ldots,A_n$. Cũng giả sử rằng $A_i \subseteq A_{i+1}$ với i lẻ, nghĩa là i = 1,3,..., n-1. Cuối cùng giả sử rằng với i = 3,5,..., n-1 chúng ta có $A_i \rightarrow A_{i+1}$ và $A_1 \rightarrow A_n$.

a) Chứng minh rằng nếu các quan hệ được giả định là hữu hạn thì tất cả các phụ thuộc trên có thể đảo ngược lại nghĩa là:

$A_2 \subseteq A_1$, $A_2 \rightarrow A_3$, $A_4 \subseteq A_3$, $A_4 \rightarrow A_5,\ldots$, $A_n \subseteq A_{n-1}$, $A_n \rightarrow A_1$

b) Chứng minh rằng tồn tại những quan hệ vô hạn mà (*a*) không đúng; nghĩa là chúng thoả tất cả các phụ thuộc đã cho nhưng không thoả các phụ thuộc đảo ngược.

2.62. Chứng minh rằng nếu D chỉ là một tập phụ thuộc hàm thì quan hệ R có dạng BCNF ứng với D nếu và chỉ nếu R có dạng 4NF ứng với D.

2.63. Chứng minh rằng nếu $X \rightarrow A_1, \ldots, X \rightarrow A_n$ là các phụ thuộc hàm trong một phủ cực tiểu thì lược đồ $XA_1, \ldots A_n$ có dạng 3NF.

Câu hỏi và bài tập ôn thi (tham khảo)

2.64. Định nghĩa quan hệ, cho thí dụ.

2.65. Nêu định nghĩa các phép toán trên quan hệ.

2.66. Nêu định nghĩa phụ thuộc hàm, bao đóng của tập PTH F, tập thuộc tính X.

2.67. Nêu định nghĩa khóa của sơ đồ quan hệ, cho thí dụ.

2.68. Trình bày thuật toán tìm khóa, cho thí dụ.

2.69. Nêu định nghĩa dạng chuẩn 2 (2NF), cho thí dụ.

2.70. Nêu định nghĩa dạng chuẩn 3 (3NF), cho thí dụ.

2.71. Nêu định nghĩa dạng chuẩn BCNF, cho thí dụ.

2.72. Nêu định nghĩa dạng chuẩn 4 (4NF), cho thí dụ.

2.73. Nêu định nghĩa hệ Sperner, chứng minh rằng họ khóa của sơ đồ quan hệ W là một hệ Sperner và ngược lại.

2.74. Cho hai quan hệ r và s như sau:

r

A	B	C	D
1	0	0	0
1	1	0	0
1	1	1	0
1	1	1	1

s

A	B	C	D
2	1	1	1
2	2	1	1
1	1	1	0
x	y	z	v

a- Tính r - s và s - r.

b- Tính r + s.

c- Tính r * s.

d- Giả sử X = {A, B, D}, Y = {A, C, D}.

Tính các quan hệ chiếu r. X, r. Y và s. X, s. Y, (r+s). X, (r+s). (X∪ Y)

e- Chứng minh rằng với mọi quan hệ r, s, q thì ta luôn có :

e. 1 r * s = s * r và r + s = s + r (tính giao hoán).

e. 2 r * (q + s) = (r * q) + (r * s) (tính kết hợp).

e. 3 (r + s). X = r. X + s. X.

e. 4 (r * s). X = r. X * s. X.

2.75. Cho hai quan hệ r và s như sau:

r

A	B	C
0	0	0
1	1	1
1	1	0

s

D	E
5	6
0	6

Tính tích Decac của r và s: r x s.

2.76. Cho hai quan hệ r và s như sau:

r

A	B	C	D	E
0	0	0	0	1
0	0	1	1	0
1	1	1	1	1
0	0	0	1	1

s

D	E
0	1
1	1
1	0

Tính r ÷ s.

2.77. Cho sơ đồ quan hệ W = < R, F >. Chứng minh (giải thích) rằng với mọi tập con X bất kỳ của R và mọi phần tử A thuộc tập X thì $X \rightarrow A \in F^+$. Tức là $\forall A \in X \subset R \Rightarrow X \rightarrow A \in F^+$.

2.78. Cho sơ đồ quan hệ W = < R, F >, với R = {A, B, C, D } và F = {A → B, BC → D}.

PTH nào trong dãy sau là suy được từ F bằng các quy tắc của PTH :

a- C → D.

b- A → D.

c- AD → C.

d- BC → A.

e- B → CD.

2.79. Cho sơ đồ quan hệ W = < R, F >,với R = {A, B,... } và F = {AB → E, AG → I, BE → I, E → G, GI → H}. Chứng minh rằng AB → GH.

2.80. Cho sơ đồ quan hệ W = < R, F >, với R = {A, B, C, D, E, G, H}, tập các PTH F: F = {A → D, AB → DE, CE → G, E → H}. Tính $(AB)^+$.

2.81.

a - Tìm các khóa còn lại của W = < R, F > trong thí dụ 2.21 và tìm các khóa của sơ đồ quan hệ W= < R, F > trong bài tập 2.15.

b - Cho sơ đồ quan hệ W = < R, F >, với R = {A,B,C,D,E,H} và

F = {A→ E, C → D, E →DH}. Chứng minh rằng K = {A,B,C} là khóa duy nhất của W.

c - Cho sơ đồ quan hệ W = < R, F >, với R = {A,B,C,D} và

F = {AB → C, D → B, C → ABD}. Tìm các khóa của W.

d - Cho sơ đồ quan hệ W = < R, F >, với R = {A,B,C,D,E,G} và

F = {AB → C, C → A, BC → D, ACD → B, D → EG,BE →C, CG → BD, CE → CG}. Tìm các khóa của W.

2.82. Xét lược đồ quan hệ có các thuộc tính S (stor), D (department), I (item) và M (manager) với các phụ thuộc hàm SI→D và SD→M.

a) Tìm tất cả các khoá của SĐQH W= <{S, D, I, M}, F >.

b) Chứng minh rằng W là 2NF nhưng không là 3NF.

2.83. Cho SĐQH W = < R, F > ; k là một khoá W.

Chứng minh rằng, với mọi tập con X của k ta có: $X^+ \cap k = X$.

2.84. Cho SĐQH W = < R, F >; Gọi X là giao của tất cả các khoá của W.

Chứng minh rằng:

a. $X = R - \cup \{Z - Y\}$ với $Y \rightarrow Z$ thuộc F.

b. W có một khoá duy nhất khi và chỉ khi $X^+ = R$.

2.85. Cho SĐQH W = < R, F >. Xét xem các mệnh đề sau có đúng không? vì sao:

a. Hai khoá khác nhau của W không có thuộc tính chung.

b. Số lượng các thuộc tính trong các khoá phải bằng nhau.

c. Số khoá của W có thể lớn hơn số thuộc tính của R.

d. W có thể không có khoá nào.

e. W có thể có một khoá k = R.

2.86. Giả sử W = < R, F > là 3NF và W có một khoá duy nhất W có luôn là BCNF không? vì sao?

2.87. Cho W = < R , F > với R = ABCDEGH và tập F = { CD → H, E → B, D → G, BH → E, CH → DG, C → A}.

a. Tìm các khoá của W, tìm giao của các khoá.

b. Tính Z = ($X^+Y)^+ \cap (k^+ - Y$) với X = CD và k là một khoá của W.

2.88. Cho SĐQH W = < R, F > với R = ABCDEH và F { BC → E, D → A, C → A, AE → D, BE → CH }.

a. Tìm các khoá của W.

b. Tính Z = ($X^+Y)^+ \cap k^+$ - (XY) với X = AB, Y = D, k là một khoá của W.

2.89. Cho W = < R, F > với R = ABCDEGH và F { AB → R, CD → R, EG → R, H → R }.

a. Tìm các phần khoá k^{-1}.

b. Tìm giao của các phản khoá.

c. Tìm giao của các khoá

2.90. Cho SĐQH W = < R, F >.

a. Có hay không một tập con thực sự của khoá k kéo theo k? vì sao?

b. Có hay không một tập con thực sự của khoá k kéo theo một khoá khác?

c. Nếu R có bốn thuộc tính thì W có nhiều nhất mấy khoá ? cho thí dụ.

d. Nếu R có 10 thuộc tính thì W có ít nhất mấy khoá ? cho thí dụ.

e. Có bao giờ số khoá của sơ đồ quan hệ W bằng số thuộc tính của R không? cho ví dụ.

f. Số khoá của W có thể nhiều hơn số thuộc tính của R không?

2.91. Cho W1 = < R, F1 >; W2 = < S, F2 >.

a. Cho ví dụ các tập R, S, F1, F2 để tập các khoá của

W = < R ∪ S , F1 ∪ F2 > là X ∪ Y với X, Y là các tập khoá của W1 và W2 tương ứng.

b. Cho ví dụ W1, W2, để W = < R ∪ S , F1 ∪ F2 > có ít nhất một khoá k = k1 ∪ k2 với k1 là khoá của W1 ; k2 là khoá của W2.

c. Cho thí dụ W1 và W2 để các khoá của W = < R ∪ S , F1 ∪ F2 > là hợp của hai khoá của W1 và W2.

d. Mệnh đề sau có luôn đúng không: W1 và W2 đều có khoá nhưng W = < R ∪ S , F1 ∪ F2 > không có khoá?

e. Cho ví dụ W1, W2 để số khoá của W1 bằng số khoá của W= < R ∪ S, F1 ∪ F2 >.

f. Có hay không một quan hệ r trên R có khoá trùng với khoá của một quan hệ s trên S?

2.92. Cho lược đồ quan hệ R = {H, G, S, T}; với H là hàng; G là giá; S là số lượng; T là tiền. Coi R như một hoá đơn. Hãy thực hiện:

a. Tìm tập phụ thuộc hàm F.

b. Tìm khoá của W = < R, F >.

c. Xét xem W thuộc dạng chuẩn nào?

d. Nếu thêm vào R thuộc tính mã hàng # ta có R = {#, H, G, S, T}, tìm tập phụ thuộc hàm F.

e. Xét xem W bây giờ thuộc dạng chuẩn nào?

2.93. Cho một cơ sở dự liệu của một cơ quan gồm các quan hệ như sau:

a. Quan hệ về nhân viên r (#, T, N, Q, V) với # là mã nhân viên; T là tên nhân viên; N là năm sinh của nhân viên; Q là quê của nhân viên; V là năm vào cơ quan của nhân viên.

b. Bảng học hàm học vị của nhân viên s (#, H, B) với H là học hàm (phó giáo sư hoặc giáo sư); B là học vị (đại học, cao học , tiến sỹ, tiến sỹ khoa học).

c. Bảng lương l ($, M) với $ là hệ số lương (đại học hệ số 1, cao học hệ số 2, tiến sỹ hệ số 3, tiến sỹ khoa học hệ số 4); M là tiền tương ứng với hệ số.

Hãy dùng các phép toán đại số quan hệ để lập các bảng báo cáo như sau:

(1) Bảng gồm T(tên) của những người Q (quê) Hà Nội có học hàm H là giáo sư và có số năm công tác trên 30.

(2) Bảng lương của những người quê Nghệ An.

(3) Bảng những người dưới 30 tuổi có lương hệ số 4.

Môn thi: Lý thuyết CSDL.
120'

Bài 1. Khoá của một sơ đồ quan hệ là gì ? Khoá của một quan hệ là gì? Cho ví dụ sơ đồ quan hệ có duy nhất một khoá, có 6 và chỉ 6 khoá. Cho ví dụ quan hệ có 3 khoá.

Bài 2. CMR, 5 phép toán quan hệ không cơ bản có thể biểu thị qua các phép toán cơ bản.

Bài 3. Cho ví dụ một quan hệ không có khoá, một sơ đồ quan hệ không có khoá.

Bài 4. Giả sử trong CSDL quản lý Điểm của một trường THPT ta có các bảng sau:

T1: Bảng điểm.

Field	Tên field	Giải thích
1	MA_MH	Mã môn học
2	KOCKY	Học kỳ
3	MA_NH	Mã năm học
4	MIENG	Kiểm tra miệng
5	15PHUT	Kiểm tra 15 phút
6	VIET	Kiểm tra viết
7	THIHOCKY	Thi học kỳ
8	MA_HS	Mã học sinh
9	TB	Điểm trung bình

T2: Hồ sơ học sinh.

Field	Tên field	Giải thích
1	MA_HS	Mã học sinh
2	TEN_HS	Tên học sinh
3	NGAYS_HS	Ngày sinh học sinh
4	ĐCHI_HS	Địa chỉ học sinh
5	NGàYNHPHOC	Ngày nhập học

T3: Lên lớp.

Field	Tên field	Giải thích
1	MA_LOPCU	Mã lớp cũ
2	MA_LOPMOI	Mã lớp mới
3	LOP_LUUBAN	Lớp lưu ban

T4: Cac lớp.

Field	Tên field	Giải thích
1	MA_LOP	Mã lớp
2	TÊNLOP	Tên lớp

T5:Môn.

Field	Tên field	Chú thích
1	MA_MH	Mã môn học
2	TEN_MH	Tên môn học

T6: Năm học.

Field	Ten field	Chú thích
1	MA_NH	Mã năm học
2	TEN_NH	Tên năm học
3	NKHOA	Niên khoá

T7: Phân lớp.

Field	Tên field	Chú thích
1	MA_NH	Mã năm học
2	MA_HS	Mã học sinh
3	MA_LOP	Mã lớp
4	HANH_KIEM_KI	Hạnh kiểm kỳ1
5	TBKYI	Trung bình kỳ 1
6	HANH_KIEMKII	Hạnh kiểm kỳ 2
7	TBHKII	Trung bình học kỳ 2
8	TB_NAM	Trung bình năm
9	HANH_KIEM	Hạnh kiểm năm
10	LOAI_KYI	Xếp loại học kỳ 1
11	LOAI_KYII	Xếp loại học kỳ 2
12	LOAI_NAM	Xếp loại cả năm

a. Ta nói một bảng là dư thừa dữ liệu nếu giá trị của một thuộc tính nào đó được lặp lại nhiều lần không cần thiết, thí dụ xét quan hệ nhà cung cấp r(Tên, Địachỉ, Hàng, Phôn, Giá, Sốlượng) thì Địachỉ và Phôn là dư thừa dữ liệu. Trong các bảng trên có bảng nào dư thừa dữ liệu kiểu đó không?

b.Các bảng trên thuộc dạng chuẩn nào?

Bài 5.

a. Hãy dùng những kiến thức về cơ sở dữ liệu của bạn để đánh giá xem với các bảng cơ sở dữ liệu như trên chúng ta đã đủ để đánh giá và quản lý, thống kê báo cáo khi cần thiết của một trường THPT?

b. Bạn có thể thiết kế một csdl tốt hơn không, vì sao?

c. Nếu bạn được giao thiết kế và quản lý điểm cho một trường học bạn phải làm thế nao?

CHƯƠNG 3

NGÔN NGỮ VẤN TIN SQL

Ở chương 2, chúng ta đã làm quen với các phép toán đại số trên các quan hệ. Trong chương này chúng ta sẽ trình bày tiếp một trong các ngôn ngữ thao tác (Data Manipulation Language - DML) trên các CSDL quan hệ, đó là ngôn ngữ SQL. SQL hay còn gọi là ngôn ngữ vấn tin có cấu trúc (Structured Query Language) để diễn tả các câu lệnh kiểu như:

1 - Truy xuất từ CSDL số lượng ghế còn trống trên một chuyến bay.

2 - Giảm một số chỗ trống trên chuyến bay 747 vào ngày 3 tháng 9.

3 - Tìm tất cả các chuyến bay từ Hà Nội đi Sài Gòn ngày 20 tháng 9.

4 - Nhập 100 chỗ từ Hà Nội đi Sài Gòn vào chuyến bay 456 ngày 20 tháng 9.

Với CSDL cài trong máy là quan hệ chuyến bay, ký hiệu là r trên tập thuộc tính {number, data, seats, from, to}. Qua các mục thao tác ở trên ta thấy các phép toán ở đây là các phép toán một ngôi, phép toán thực hiện trên một quan hệ. Các phép toán kiểu này minh họa cho một yêu cầu vấn tin CSDL ta thường gọi là các phép cập nhật (update), chèn (insert) trong các ngôn ngữ vấn tin.

Thí dụ trong ngôn ngữ SQL để diễn tả câu lệnh số 2 với giảm 4 chỗ trống:

```
Update r
Set seats = seats - 4
Where number = 747 And Date = '3/9';
```

Hoặc để thực hiện mục 4 - ta có thể viết:

Insert into r

Values (456,'20/9',100,'Hanoi','Saigon');

Thuật ngữ "ngôn ngữ vấn tin" thường được dùng như một từ đồng nghĩa với DML. Nói tóm lại chỉ một số câu lệnh của DML là vấn tin, chúng là những câu lệnh 1 và 3 ở trên. Những câu lệnh này chỉ vấn tin không làm thay đổi CSDL. Các câu lệnh 2 và 4 có làm thay đổi CSDL nên không phải là những câu vấn tin mặc dù chúng được dùng trong các ngôn ngữ vấn tin.

Như vậy ngôn ngữ vấn tin có thể hiểu là những ngôn ngữ hàm chứa những câu vấn tin.

Sau đây chúng ta sẽ xét một "đại diện" của các ngôn ngữ vấn tin, đó là ngôn ngữ SQL, một ngôn ngữ được sử dụng khá phổ biến trong CSDLPT trên các Mạng Máy Tính.

Ngôn ngữ SQL (Structured Query Language) là sự phát triển tiếp theo của SEQUEL, là một ngôn ngữ được IBM phát triển ở San Jose, mục đích để sử dụng trong hệ thống CSDL thử nghiệm là System R. Ngày nay nó được sử dụng trong rất nhiều hệ thống CSDL thương mại, trong một số trường hợp, toàn bộ hệ thống CSDL được thương mại hóa với tên SQL.

Trong giáo trình này chúng ta sẽ nêu các nguyên lý cơ bản của các câu lệnh trong SQL. Nội dung chương này chưa đủ để thao tác với tư cách là ngôn ngữ lập trình. Để thao tác tốt với SQL các bạn có thể tham khảo thêm trong các tài liệu về SQL 7.0 hiện có nhiều trong các quầy sách.

Phiên bản đặc biệt chúng ta sẽ sử dụng trong phần này là SQL/RT, được cài đặt trên máy IBM PC/RT của công ty Oracle.

Bởi SQL là một ngôn ngữ vấn tin quan hệ được cài đặt thông dụng nhất nên độc giả nên xét nó một cách chi tiết hơn.

Sau đây chúng ta sẽ xét các câu lệnh vấn tin tiêu biểu của SQL.

Chúng ta nhắc lại rằng SQL là một *ngôn quan hệ* nên kết quả của mọi câu lệnh là một quan hệ.

3.1. ĐỊNH NGHĨA DỮ LIỆU TRONG SQL

Trước khi thực hiện truy vấn trên các quan hệ, chúng ta phải khai báo và tạo các quan hệ.

Sau đây chúng ta sẽ xét cách thiết lập, tạo một bảng cho các dữ liệu quan hệ.

3.1.1 CÁCH TẠO MỘT QUAN HỆ

Chúng ta thấy rằng các miền giá trị của các quan hệ có thể là các đại lượng số, các ký tự, các chuỗi ký tự,... Vậy khi tạo bảng quan hệ chúng ta phải thông báo luôn cả các kiểu của các thuộc tính.

Cú pháp tạo bảng quan hệ T:

CREATE TABLE T (A_1 k_1, A_2 k_2,... , A_n k_n) ;

Trong đó T là tên quan hệ, A_1,... , A_n là các thuộc tính của T, k_1,... , k_n là các kiểu tương ứng của A_1,... , A_n

Thí dụ 3.1:

Ta tạo quan hệ supplies cho siêu thị Tomek :

CREATE TABLE supplies

(NAME CHAR(20) NOT NULL, ITEM CHAR(10)NOT NULL,
PRICE NUMBER(6,2)) ;

Trong câu lệnh trên NOT NULL nhằm chỉ các chuỗi ký tự phải khác rỗng, các số (20), (10) chỉ độ dài của các chuỗi ký tự. Còn (6,2) là số có sáu chữ số với hai chữ số lẻ.

3.1.2 HỦY MỘT QUAN HỆ

Ngược với phép tạo một quan hệ là phép hủy một quan hệ. Muốn xóa một quan hệ ta dùng DROP. Ví dụ, DROP TABLE supplies.

Xóa toàn bộ bảng quan hệ supplies.

Vậy cú pháp xoá quan hệ T là:

DROP TABLE T

3.2. CÂU LỆNH SELECT

Bạn đọc cần lưu ý rằng trong các CSDL tập trung cũng như phân tán chúng ta thường có nhiều quan hệ và nhiều lúc chúng ta cần chọn trong các quan hệ ấy những đối tượng có một số thuộc tính nào đó và thoả một điều kiện nào đó. Các thuộc tính đó có thể xuất hiện trong một hoặc nhiều quan hệ khác nhau. Để thực hiện công việc này trong SQL ta dùng câu lệnh SELECT.

Dạng thông dụng nhất của câu vấn tin trong ngôn ngữ SQL là câu lệnh SELECT.

Cho các quan hệ r trên R = {A_1, A_2, ...A_n}; s trên S = {B_1, B_2, ...B_m}. Giả sử ta cần tạo ra bảng mới gồm các đối tượng có các thuộc tính A_1, B_1, ... thoả điều kiện E. Chú ý ta có thể có nhiều quan hệ.

Cú pháp của câu lệnh SELECT:

SELECT r. A_1,... , s. B_1

FROM r,..., s { Giữa r và s có thể có các quan hệ khác}

WHERE E ;

Trong đó: r.A_1... , s.B_1 là các chiếu của các quan hệ r và s lên A_1,...,B_1 tương ứng. (cách viết r. A là chỉ thuộc tính A của quan hệ r). E là một biểu thức logic có các phép so sánh số học, các tân từ lôgic AND, OR, NOT.

Nếu sau từ FROM chỉ có một quan hệ thì thay cho việc viết r. A ở dòng đầu ta chỉ cần viết A, thí dụ :

SELECT A, B

FROM r

WHERE E

Nội dung của câu vấn tin SELECT trên có thể viết trong đại số quan hệ như sau:

$$((r |><| \ldots |><| s)(E)).\ A_1B_1.$$

Nghĩa là lấy trong nối tự nhiên (hoặc tích Decac khi các quan hệ rời nhau) của các quan hệ r, . . ., s có trong mệnh đề FROM các bộ thỏa mãn E rồi chiếu quan hệ thu được lên các thuộc tính A_1,. . ., B_1.

Nếu sau FROM chỉ có một quan hệ r thì câu lệnh vấn tin có nội dung:

$$(r(E)).\ AB$$

Các bạn nên lưu ý rằng kết quả cuối cùng của câu lệnh SELECT là chiếu chứ không phải là chọn. Các bước thực hiện trong câu vấn tin này là: *nối- chọn - chiếu.*

Thí dụ 3.2:

Cho CSDL về các nhà cung cấp và các khách hàng lớn của một siêu thị gồm các quan hệ như sau:

(1) Nhà cung cấp : supplies(NAME, ITEM, PRICE, QUANTITY). Quan hệ cho biết tên nhà cung cấp, hàng, giá và số lượng.

(2) Khách hàng : customers(NAME, ITEM, PRICE, QUANTITY, MONEY, BALANCE). Quan hệ có các thuộc tính tên khách hàng, hàng, giá, số lượng, tiền và số dư tương ứng của khách hàng.

(3) Địa chỉ của nhà cung cấp : adsup(NAME, ADDR, TELEPHON). Quan hệ này cho biết tên, địa chỉ, telêphon của nhà cung cấp.

(4) Địa chỉ của khách hàng : adcus(NAME, ADDR, TELEPHON). Quan hệ cho biết tên, địa chỉ, telêphon của khách hàng.

(5) Đơn đặt hàng của khách hàng: order(O#, NAMECUS, DATE, ITEM, QUANTITY). Quan hệ cho biết mã đơn, tên khách hàng, ngày, hàng và số lượng tương ứng. Sau đây giả sử ta có một vài quan hệ cụ thể như sau:

supplies

NAME	ITEM	PRICE	QUANTITY
Acme	rượu nho	3.49	100
Acme	rượu lê	4.19	1000
Acme	đồng hồ	1,06	10000
Acme	Tivi	10,25	15
Ajax	rượu nho	3.98	1000
Ajax	ruọu lê	1.09	1005
Ajax	tủ lạnh	10.69	104
Tomek	tivi	10.05	100
Tomek	rau sạch	0.5	100

Customers

NAME	ITEM	PRICE	QUANTITY	MONEY	BALANCE
Lan	rau sạch	0,5	5	0,75	-200
Kiệt	ruọu lê	1,09	10	10,9	-50
Mười	tivi	10,05	2	20.1	+43
Hương	rượu nho	3,98	2	7,96	+50
Hoa	rượu nho	3,98	3	11,94	+200
Hoa	đồng hồ	1,06	10	10,6	+189,4
Hoa	tivi	10,05	1	10,05	+178,9
Hoa	rau sạch	0,5	2	1,0	+177,9
Linh	rượu nho	3,98	6	23,88	+100
Linh	rượu lê	1,09	7	7,63	+92,37

Order

O#	NAMECUS	DATE	ITEM	QUANTITY
1024	Linh	03/07	rượu nho	6
1024	Linh	03/07	rượu lê	7
1025	Vinh	04/07	rượu nho	5
1025	Vinh	05/07	tivi	2
1026	Quang	06/07	tulạnh	2

Xét câu vấn tin "liệt kê những khách hàng có số dư âm" của siêu thị. Khi đó trong SQL ta có:

```
SELECT NAME
FROM customers
Where BALANCE < 0 ;
```

Trong thí dụ này chỉ có một quan hệ khách hàng customers trong mệnh đề FROM nên không có sự nhầm lẫn nào liên quan đến các thuộc tính. Tuy nhiên chúng ta có thể viết:

```
SELECT customers. NAME
FROM customers
WHERE  customers. BALANCE < 0 ;
```

Kết quả của câu lệnh chọn trên là:

NAME	BALANCE
Lan	-200
Kiệt	-50

Kết quả của việc thực hiện câu lệnh này cho ta một quan hệ chỉ một cột, một thuộc tính là NAME

Nếu muốn lấy các thông tin về những khách hàng có số dư âm, địa chỉ, tên của họ, chúng ta có thể viết:

```
SELECT customer.NAME, adcus.ADDR, customer.BALANCE
FROM customers, adcus
WHERE customer.BALANCE < 0 ;
```

Và kết quả là một quan hệ có ba thuộc tính (ba cột) và hai dòng.

Trong SQL có thể dùng dấu thay thế * để chỉ tất cả các thuộc tính:

```
SELECT *
FROM customers
WHERE BALANCE < 0 ;
```

Thí dụ 3.3:

Xét mệnh đề "in ra tất cả các nhà cung cấp các mặt hàng mà khách hàng Hoa mua" (các nhà cung cấp có ít nhất một mặt hàng Hoa mua).

Khi đó trong SQL ta có thể viết:

```
SELECT supplies.NAME
FROM  supplies, customers
WHERE (customers.NAME = Hoa) AND
        (customers.ITEM = supplies.ITEM)
```

Kết quả của câu lệnh:

NAME

Ajax

Acme

Tomek

Tomek

Một đặc thù của các ngôn ngữ vấn tin nói chung và SQL nói riêng là không tự động loại trừ các trùng lặp, vì vậy trong câu vấn tin trên nhà cung cấp A được in ra nhiều lần, vì A cung cấp nhiều mặt hàng Hoa đặt. Để loại trừ các trùng lặp kiểu như vậy ta dùng từ khóa DISTINCT theo sau SELECT, nghĩa là trong câu lệnh vấn tin khi đó có dạng:

```
SELECT  DISTINCT r. A,... , s. B
FROM r,... , s
WHERE E ;
```

3.3. BIẾN BỘ

Đôi khi điều kiện E cần liên hệ đến hai bộ hoặc nhiều hơn hai bộ trong cùng một quan hệ. Để thực hiện việc này chúng ta định nghĩa các biến bộ cho quan hệ sau mệnh đề FROM và dùng biến bộ này làm " bí danh " cho quan hệ:

Thí dụ 3.4:

Xét câu "in ra tên của những khách hàng có số dư nhỏ hơn của Kiệt" khi đó câu lệnh sẽ là:

SELECT NAME .
FROM customer
WHERE BALANCE < - 50 ;(nếu biết số dư của Kiệt là -50).

Vấn đề ở đây là chúng ta không biết cụ thể số dư của Kiệt là bao nhiêu khi đó câu lệnh phải được viết như thế nào ? nghĩa là tham chiếu đến số dư của Kiệt bằng cách nào ? Trong trường hợp này ta dùng " bí danh ".

Khi đó câu lệnh có dạng:

SELECT c_1. NAME
FROM customer c_1, customer c_2
WHERE c_1. BALANCE < c_2. BALANCE
AND c_2. NAME = 'Kiệt' ;

Ở đây chúng ta cần nhớ thêm một cách viết của SQL là: các tên đi sau các tên quan hệ không có dấu phân cách là các *bí danh bộ* của các quan hệ đó, ví dụ customer c_1, customer c_2 thì cả c_1 và c_2 đều là bí danh bộ của quan hệ customer, chúng trở thành các bộ chạy trên customer.

3.4. MỞ RỘNG KHẢ NĂNG CỦA WHERE E BẰNG TỪ KHOÁ LIKE

Nhằm mở rộng thêm khả năng của các biểu thức, công thức E trong mệnh đề WHERE ta dùng từ khóa like và hai ký hiệu thay thế phụ: %, _

- từ *like* ở đây dùng theo nghĩa " là " .
- dấu % được dùng để thế chân " chuỗi ký tự bất kỳ " .
- dấu _ thay thế " một ký tự bất kỳ " .

Thí dụ 3.5:

Xét câu vấn tin "in ra các mặt hàng của các nhà cung cấp có tên gọi bắt đầu bằng ký tự A " , khi đó ta có câu lệnh:

```
SELECT ITEM
FROM supplies
WHERE ITEM like 'A%' ;
```

Kết quả của câu lệnh là một quan hệ một cột gồm các mặt hàng có tên gọi bắt đầu bằng ký tự A của các nhà cung cấp .

Thí dụ 3.6:

Xét câu vấn tin "in ra tên của những nhà cung cấp bắt đầu bằng TO và tên đó có 5 chữ cái", khi đó ta có câu lệnh:

```
SELECT  NAME
FROM  supplies
WHERE NAME like 'TO_ _ _ ' ;
```

Tất nhiên ba dấu _ _ _ đại diện cho các dấu bất kỳ nhưng máy sẽ hiểu bất kỳ trong các tên của các nhà cung cấp thí dụ TOMEK, TOMIH, TOAMI,...

3.5. PHÉP TOÁN TẬP HỢP TRONG WHERE

Trong phần này chúng ta tiếp tục xét tiếp các khả năng rộng hơn so với phép toán chọn trong đại số quan hệ của biểu thức E trong WHERE E.

Trước tiên nhắc lại trong WHERE ta có thể viết:

WHERE A > B + C - 10

Một khả năng mở rộng của biểu thức E trong mệnh đề WHERE E, trong E được dùng các toán hạng, các toán hạng đó đôi khi là một câu vấn tin con dạng SELECT - FROM - WHERE. Chúng ta thấy kết quả của các câu vấn tin SELECT - FROM - WHERE là một quan hệ có được từ các phép nối-chọn - chiếu. Vậy để có thể dùng (SELECT - FROM - WHERE) như một toán hạng cho một phép nối-chọn-chiếu khác ta phải dùng các phép toán tập hợp như:

- IN (thuộc).
- NOT IN (không thuộc).
- ANY (tồn tại).
- ALL (với mọi).

Thí dụ 3.7:

Ta quay lại câu vấn tin đã xét ở phần trên "Hãy in tên các nhà cung cấp có ít nhất một mặt hàng Hoa mua". Để thực hiện công việc này ta có thể viết

```
SELECT NAME
FROM supplies
    WHERE ITEM IN (SELECT ITEM FROM customer  WHERE
    NAME = 'Hoa') ;
```

Hoặc in tên những nhà cung cấp không bán tivi cho Hoa ta có câu lệnh:

```
    SELECT NAME
```

```
FROM supplies
WHERE (ITEM = tivi) and ITEM  NOT IN (SELECT ITEM
FROM customer  WHERE NAME = 'Hoa') ;
```

Từ khóa ANY được sử dụng như một lượng tử tồn tại, nếu S là một biểu thức biểu thị một tập hợp thì điều kiện: A θ ANY S tương đương với biểu thức logic: tồn tại X thuộc S sao cho X, A thỏa mãn toán tử θ, tức là ta có biểu thức: (∃ X) (X∈ S and Aθ X), với A là một thuộc tính được lấy giá trị từ một quan hệ nào đó. Tương tự như thế A θ ALL S tương đương với biểu thức logic: (∀ X) (nếu X∈ S thì Aθ X).

Thí dụ 3.8:

Xét câu "in ra mặt hàng có giá lớn hơn mọi giá trong quan hệ supplies".

Ta có thể thực hiện công việc bằng một câu vấn tin con để tạo ra tập tất cả các giá, từ đây ta biết giá lớn nhất rồi đưa ra tiếp yêu cầu của công việc.

Ta có câu lệnh:

```
SELECT ITEM
FROM supplies
WHERE PRICE >= ALL
   (SELECT  PRICE  FROM supplies) ;
```

Thí dụ 3.9:

Nếu chúng ta biết chắc chắn rằng Mạnh đã đặt đúng một đơn đặt hàng thì chúng ta có thể lấy được mã số đơn đặt hàng đó qua một câu vấn tin con và dùng nó để chọn tất cả các mặt hàng được đặt bởi Mạnh từ order.

```
SELECT ITEM
FROM order
WHERE O# = (SELECT O#   FROM  order   WHERE NAMECUST = 'Mạnh');
```

3.6. CÁC PHÉP TOÁN GỘP NHÓM

3.6.1. GỘP THEO MỘT THUỘC TÍNH

Trong SQL ta dùng năm phép toán gộp: tính giá trị trung bình AVG, tính số lượng COUNT, tính tổng SUM, tính MIN, MAX của một thuộc tính, cùng các toán tử STDDEV và VARIANCE để tính độ lệch chuẩn và phương sai của một danh sách các số.

Giả sử ta ký hiệu G là một trong năm phép gộp. Khi muốn thực hiện phép gộp lên thuộc tính A ta viết G(A), thí dụ AVG(A), COUNT(A),... Nếu trước thuộc tính A có thêm từ khóa DISTINCT thì sự trùng lặp được loại bỏ trước khi thực hiên phép gộp. Tất nhiên phép tính gộp phải được đặt trong SELECT - FROM - WHERE.

Thí dụ 3.10:

Ta cần in ra giá trị trung bình của các số dư trong quan hệ customers:

Khi đó ta có câu lệnh:

```
SELECT AVG (BALANCE)
FROM customers ;
```

Thí dụ 3.11:

Ta cần tính và in ra số nhà cung cấp: (phải tránh sự trùng lặp)

```
SELECT COUNT (DISTINCT NAME) # KH
FROM supplies;
```

Câu lệnh sẽ in ra số lượng các nhà cung cấp khác nhau nhớ vào # KH

Kết quả ứng dụng cho quan hệ cụ thể trên ta có #KH =3

Thí dụ 3.12:

Ta sẽ in ra số các nhà cung cấp rượu nho nhớ vào #CCnho:

Ta có câu lệnh:

```
SELECT COUNT (DISTINCT NAME) # CCnho
FROM supplies
WHERE ITEM = ' rượu nho' ;
```

3.6.2 GỘP THEO NHÓM THUỘC TÍNH

Trong SQL cho phép chúng ta thực hiện các phép gộp theo nhóm bằng cách chia các bộ của quan hệ thành các nhóm và thực hiện phép gộp theo các nhóm riêng biệt. Chúng ta sẽ thực hiện các phép gộp như vậy qua từ khóa GROUP BY (nhóm lại bởi) và danh sách các thuộc tính cần nhóm.

Nghĩa là mệnh đề: GROUP BY A_1, A_2,... , A_k chia quan hệ thành các nhóm sao cho hai bộ sẽ cùng một nhóm nếu và chỉ nếu chúng giống nhau ở các thuộc tính A_1,... , A_k.

Để kết quả của câu vấn tin như thế có nghĩa, tất nhiên các thuộc tính A_1,... , A_k phải xuất hiện trong câu lệnh SELECT.

Thí dụ 3.13:

Trở lại CSDL về siêu thị trên và xét câu vấn tin "in ra một bảng tất cả các mặt hàng và giá trung bình của chúng trong quan hệ các nhà cung cấp" . Khi đó ta có câu lệnh:

```
SELECT ITEM, AVG(PRICE) AP
FROM supplies
GROUP BY ITEM ;
```

Bí danh AP cho AVG(PRICE) được dùng theo nghĩa như kết quả sau

ITEM	AP
nho	3. 74
lê	2. 64
Đồng hồ	10. 06
Tivi	10. 15
Tủlạnh	10. 69
Rau sạch	0.5

Nghĩa là ta được bảng các mặt hàng và giá trung bình của chúng.

Với các phép gộp nhóm, mệnh đề WHERE vẫn có thể xuất hiện, nhưng chúng ta sẽ dùng thêm từ khóa HAVING để đưa ra một tập con các nhóm độc lập với các điều kiện chọn lọc trong WHERE trước khi xây dựng các nhóm.

Mệnh đề HAVING E được thực hiện: HAVING E phải đi sau GROUP BY, nghĩa là muốn dùng HAVING E ta viết:

GROUP BY A_1,... , A_k

HAVING E

và khi đó điều kiện E được sử dụng cho mỗi quan hệ r, chứa nhóm các bộ giá trị a_1,... ,a_k tương ứng với các thuộc tính A_1,... , A_k thỏa mãn điều kiện E và cũng chỉ những nhóm này được xét trong câu lệnh.

Thí dụ 3.14:

Ta quay lại thí dụ 3. 13 và giả sử ta chỉ in ra những mặt hàng được nhiều nhà cung cấp. Khi đó ta có thể viết:

```
SELECT ITEM, AVG(PRICE) AP
FROM supplies
GROUP BY ITEM
HAVING COUNT(*) < 4 ;
```

Ta cần lưu ý rằng dấu * đại diện cho tất cả các thuộc tính của quan hệ được tham chiếu, trong trường hợp này ta chỉ có một quan hệ supplies, do đó COUNT(*) đếm các bộ theo các nhóm riêng biệt và chỉ xét những nhóm ITEM có giá trị nhỏ hơn 4, ví dụ ta có kết quả thực hiện câu lệnh là:

ITEM	AP
Rượu nho	3. 74
Rượu lê	2. 64
Rau sạch	0.5

Hoặc ta lấy thí dụ in tên của khách hàng đã giả nhiều tiền nhất cho siêu thị. Khi đó ta có câu lệnh:

```
SELECT NAME
FROM customers
    WHERE NAME IN (SELECT DISTINCT NAME, SUM(TIEN) T
    FROM customer  Group by NAME  HAVING T = MAX(T)) ;
```

3.7. PHÉP CHÈN

Để chèn một bộ vào một quan hệ ta dùng lệnh chèn INSERT INTO r. Cú pháp của lệnh chèn là:

```
INSERT INTO r
VALUES (v1,... , vn) ;
```

Trong đó r là quan hệ trên tập thuộc tính R = {A_1,... , A_n} và v_1,... , v_n là một bộ giá trị cụ thể của các thuộc tính A_1,... , A_n tương ứng.

Thí dụ 3.15:

Thêm vào quan hệ order giả sử mặt hàng sơmi, có mã là 1027 và số lượng là 100, tên khách hàng là Linh đặt ngày 3 tháng 7. Khi đó ta có câu lệnh:

```
INSERT INTO order
 VALUES (1027, Linh, 06/07, sơmi, 100) ;
```

Giả sử mặt hàng sơmi được cung cấp bởi ông Tàu và giá 8, 5[đ] và số lượng 1000 ta có câu lệnh:

INSERT INTO supplies
VALUES (Tàu, sơmi, 8.5, 1000) ;

3.8. PHÉP XÓA

Ngược với phép chèn là phép xóa một bộ hoặc một nhóm bộ ra khỏi một quan hệ r, trong phép xóa thay cho việc liệt kê bộ giá trị cần xóa ta dùng điều kiện E để chỉ điều kiện các giá trị đó phải thỏa mãn.

Cú pháp của phép xóa:

DELETE FROM r
WHERE E;

Trong đó r là quan hệ, E là biểu thức logic. Kết quả thực hiện của câu lệnh là những bộ của r thỏa mãn E sẽ bị xóa.

Thí dụ 3.16:

Xóa khỏi order các mặt hàng có mã số là 1025:

DELETE FROM order
WHERE O# = '1025' ;

Thí dụ 3.17:

Ta cần xóa các đơn đặt hàng có chứa rượu nho:

DELETE FROM orders
WHERE ITEM = ' rượu nho ';

3.9. PHÉP CẬP NHẬT

Một phép toán ta thường gặp khi sử dụng các CSDL là cần sửa đổi các bộ dữ liệu, để thực hiện công việc đó trong SQL ta dùng lệnh cập nhật UPDATE r. Cú pháp:

UPDATE r

SET $A_1 = v_1, \ldots , A_n = v_n$

WHERE E ;

Trong đó r là quan hệ

$v_1, \ldots , v_n$ là các giá trị cần sửa đổi cho các thuộc tính

$A_1, \ldots , A_n$ tương ứng.

E là điều kiện để các bộ cần cập nhật thỏa mãn.

Thí dụ 3.18:

Trong quan hệ supplies, ta cần sửa giá của rượu lê của ông A thành một đôla. Trong SQL ta có câu lệnh:

UPDATE supplies
SET PRICE = 1. 00
WHERE NAME = 'A' AND ITEM = 'rượu lê ' ;

Sau đó hạ giá tất cả các mặt hàng của nhà cung cấp A
xuống 10%. Trong SQL ta viết:

```
UPDATE supplies
SET PRICE =. 9*PRICE
WHERE NAME = ' A' ;
```

3.10. TÍNH ĐẦY ĐỦ CỦA SQL

Sau đây chúng ta sẽ xét tính đầy đủ của SQL, tức là xét xem SQL có thực hiện được tất cả các biểu thức của các phép toán đại số quan hệ hay không? Để thực hiện một biểu thức của đại số quan hệ ta thực hiện theo thứ tự từ các biểu thức con dần ra đến biểu thức toàn bộ. Vậy để xét xem SQL có thực hiện được các biểu thức của đại số quan hệ ta chỉ cần xét lần lượt cho các phép toán cơ bản của đại số quan hệ.

Trong chương 2 chúng ta đã có bài tập nói về mối liên hệ giữa các phép toán quan hệ (xem bài tập 2.6*). Bây giờ chúng ta phát biểu kết quả đó dưới dạng định lý được công nhận như sau:

Định lý 3.1:

a. Năm phép toán quan hệ cơ bản là hợp, hiệu, tích Decac, chọn, chiếu của đại số quan hệ độc lập với nhau, nghĩa là không một phép nào trong chúng được biểu thị qua các phép còn lại.

b. Các phép toán khác của đại số quan hệ như nối tự nhiên, giao, nối nửa, nối theo teta, chia có thể nhận được từ các phép cơ bản trên.

Vậy để SQL thực hiện được các phép toán đại số quan hệ ta chỉ cần cài đặt cho SQL năm phép toán cơ bản là hợp, hiệu, tích Decac, chọn, chiếu.

• *Phép hợp:* Giả sử ta có hai quan hệ r và s có cùng lược đồ R = $\{A_1,\ldots, A_n\}$. Khi đó để tính T = r + s ta viết:

```
INSERT INTO T
SELECT *
FROM r ;
```

Tiếp theo là

```
INSERT INTO T
SELECT *
FROM s ;
```

• *Phép trừ:* Để tính T = r - s trước tiên chúng ta chèn r vào T như trên. Sau đó dùng câu lệnh xóa như sau:

```
DELETE FROM T
WHERE (A1,... , An) IN
        (SELECT * FROM s) ;
```

• *Tích Decac:* Giả sử ta đã có r trên $R = \{A_1, A_2, \ldots, A_n\}$ và s trên $S = \{B_1, B_2, \ldots, B_m\}$; với điều kiện $R \cap S = \emptyset$; thực hiện $T = r \times s$, ta có câu lệnh:

```
INSERT INTO T
SELECT r. A1,... , r. An, s. B1,... , s. Bm
FROM r, s ;
```

• *Phép chọn:* T = r(E).

```
INSERT INTO T
SELECT *
FROM r
WHERE E ;
```

• *Phép chiếu:* Giả sử X là tập con của R và X = $\{A_1,... A_k\}$

T = r. X, ta có câu lệnh :

```
INSERT INTO T
SELECT A1,... , Ak
FROM r ;
```

• Các phép toán còn lại của đại số quan hệ đều có thể nhận được từ các phép toán trên. Thí dụ:

• *Phép giao:* Ta chú ý rằng T = r ∩ s = r - (r - s). Các câu lệnh của phép trừ ta dùng như trên.

• *Phép nối tự nhiên:* T = r |><| s của r trên R và s trên S. Ta lưu ý rằng nếu Rvà S rời nhau thì nối tự nhiên của r và s là tích Decac của r và s. Nếu R và S chung nhau tập thuộc tính X thì ta chỉ nối những bộ của r và s có giá trị bằng nhau trên X. Vậy câu lệnh để tính T sẽ là:

```
INSERT INTO T
SELECT r.A1, . . . . ,r.An , s.B1 , . . . s.Bm
```

FROM r, s

WHERE r.X = s.X

Ở đây chúng ta cần lưu ý rằng trong các thuộc tính $r.A_1$, ...$s.B_m$ có những thuộc tính trong X chung nên chỉ được viết một lần.

Một cách tiếp cận khác chúng ta có thể biểu diễn phép nối qua tích Decac như sau: r |><| s = (r.(R- S) × s)(E). Với E là những t mà t.R∈r hoặc r |><| s = (r × s.(S-R))(E). Với E là điều kiện t.S ∈ s. Kết quả của phép toán này là chiếu, tích Decac, chọn.

Insert into r1

Select r.(R-S) ** r.(R-S) viết ngắn cho các thuộc tính trong R-S **

From r

Insert into r2

Select r.(R-S) s.S

From r1, s

Insert into T

Select *

From r2

Where t.R ∈r

• *Phép chia:* T = r ÷ s với r trên R và s trên S, S là tập con của R. Thực chất kết quả của phép chia là những bộ r.(R-S) thoả mãn r.S = s.S. Vậy ta có câu lệnh tính T:

INSERT INTO T

SELECT r.(R-S)

FROM r

WHERE r.S = s.S

- *Phép nối theo θ :* chọn trong tích Decac r×s những bộ thoả θ
- *Phép nối nửa:* T = r |>< s Thực chất là chiếu của r |><| s lên R. Tức là

r |>< s = (r |><| s). R

3.11. TẠO KHUNG NHÌN

Trong SQL có một nhóm lệnh đóng vai trò như ngôn ngữ định nghĩa dữ liệu DDL (Data Definition Language), nghĩa là từ một quan hệ, nhóm lệnh này định nghĩa cho ta một quan hệ con, tạo cho ta một khung nhìn (view) có tên mà không tốn thêm bộ nhớ.

Cú pháp của câu lệnh tạo khung nhìn:

CREATE VIEW V(A_1,... , A_k) AS Q ;

Trong đó V là tên của khung nhìn, A_1,... , A_k là các thuộc tính của nó. Q là câu vấn tin định nghĩa khung nhìn.

Thí dụ 3.19:

Từ quan hệ các nhà cung cấp ta có thể tạo ra một khung nhìn: quan hệ gồm các mặt hàng và giá của chúng do A cung cấp:

```
CREATE VIEW V (ITEM, PRICE) AS
SELECT ITEM, PRICE
FROM supplies
WHERE NAME = 'A' ;
```

Tương tự như hủy bảng, muốn hủy khung nhìn ta dùng câu lệnh DROP. Thí dụ, DROP V.

Trên đây là những khái niệm cơ bản của ngôn ngữ vấn tin SQL đủ dùng cho các minh hoạ trong cuốn sách này. Để sử dụng được SQL chúng ta cần tham khảo những qui định cụ thể và sự gắn kết của SQL với các ngôn ngữ chủ khác. Các bạn có thể tìm thấy các phiên bản khác của SQL trong nhiều tài liệu khác hiện đang được bày bán trên thị trường.

CÂU HỎI VÀ BÀI TẬP

3.1. Giả sử chúng ta có CSDL thống kê về mối liên hệ của các quán (BAR) bia (BEER) và những người uống (DRINKER) bia như sau:

R(BAR, DRINKER) là quan hệ cho biết các quán bia và những khách hàng của quán. S(BAR, BEER) là quan hệ cho biết các quán và các loại bia thường bán của quán. Quan hệ T(DRINKER, BEER) cho biết những loại bia mà một khách hàng ưa thích.

Hãy viết các câu lệnh bằng SQL thực hiện các mệnh đề tương ứng:

a- Các quán có loại bia Long ưa thích.

b- Những khách hàng thường đi uống ít nhất một quán có loại bia họ ưa thích.

c- Những khách hàng không đến uống ít nhất một quán có loại bia họ ưa thích.

d- Xoá tất cả loại bia Tiger trong T(DRINKER, BEER).

e- Chèn thông tin Long thích Tiger.

f- Chèn thông tin Long thích tất cả loại bia bán ở quán A2.

3.2. Giả sử trong CSDL bia ở trên ta thêm quan hệ BAN(BAR, BEER, SL) là quan hệ cho biết Số Lượng từng loại bia đã bán ở các quán.

Hãy viết các câu vấn tin bằng SQL thực hiện các mệnh đề sau:

a- Tổng số bia của một loại đã bán.

b- Số trung bình mỗi loại bia được bán ở các quán.

c- Số lượng loại bia được bán ra nhiều nhất.

CÂU HỎI VÀ BÀI TẬP

3.1. Giả sử chúng ta có CSDL thống kê về mối liên hệ của các quán (BAR) bia (BEER) và những người uống (DRINKER) bia như sau:

R(BAR, DRINKER) là quan hệ cho biết các quán bia và những khách hàng của quán. S(BAR, BEER) là quan hệ cho biết các quán và các loại bia thường bán của quán. Quan hệ T(DRINKER, BEER) cho biết những loại bia mà một khách hàng ưa thích.

Hãy viết các câu lệnh bằng SQL thực hiện các mệnh đề tương ứng:

a- Các quán có loại bia Long ưa thích.

b- Những khách hàng thường đi uống ít nhất một quán có loại bia họ ưa thích.

c- Những khách hàng không đến uống ít nhất một quán có loại bia họ ưa thích.

d- Xoá tất cả loại bia Tiger trong T(DRINKER, BEER).

e- Chèn thông tin Long thích Tiger.

f- Chèn thông tin Long thích tất cả loại bia bán ở quán A2.

3.2. Giả sử trong CSDL bia ở trên ta thêm quan hệ BAN(BAR, BEER, SL) là quan hệ cho biết Số Lượng từng loại bia đã bán ở các quán.

Hãy viết các câu vấn tin bằng SQL thực hiện các mệnh đề sau:

a- Tổng số bia của một loại đã bán.

b- Số trung bình mỗi loại bia được bán ở các quán.

c- Số lượng loại bia được bán ra nhiều nhất.

3.3. Từ CSDL bia ở trên tạo khung nhìn trong SQL : W(DRINKER, BEER, BAR), khung nhìn này chứa những bộ (d,b,q) cho biết khách hàng d thích loại bia b và hay uống ở quán q.

3.4. Giả sử chúng ta có quan hệ R(F, S, O) cho biết tập tin F của chủ nhân O có kích thước S và quan hệ FTD(F, T, D) với ý nghĩa F có kiểu T trong thư mục D. Hãy dùng SQL để thực hiện các câu sau:

a- In ra chủ nhân và kiểu tin của tất cả các kiểu tin có kích thước tối thiểu là 1000.

b- In ra tất cả các tập tin của ông long.

c- In ra kích thước trung bình của các tập tin trong thư mục BIN.

d- In ra tất cả các tập tin có trong thư mục BIN với tên có chứa chuỗi ký tự abc.

3.5. Với dữ liệu như bài 3.4, hãy viết biểu thức đại số quan hệ tương ứng của câu vấn tin sau:

```
SELECT O
FROM R
 WHERE FILE IN
     ( SELECT  FILE FROM FTD WHERE T = 'number').
```

3.6. Hãy dùng SQL tạo bảng sinh viên vừa thi vào trường của bạn với các thuộc tính như sau: SBD (số báo danh), HT (họ tên), NS (năm sinh), Q (quê), Đ (điểm).

a- In những sinh viên đậu vào trường (trên 20 điểm)

b- In những sinh viên quê Lai Châu, Hà Nội.

c- In những sinh viên đậu vào trường quê VINH.

3.7. Giả sử tại bộ giáo dục và đào tạo có CSDL của thí sinh thi vào 5 trường đại học T1, T2, T3, T4, T5 với các thuộc tính như bài tập 3.6. Bạn hãy dùng SQL để tạo bảng in danh sách thí sinh trúng tuyển của tất cả 5 trường theo mẫu : số thứ tự, họ tên, số báo danh, điểm, tên trường. Viết các câu lệnh thực hiện các mệnh đề sau:

a- Điểm số trung bình của thí sinh thi vào cả 5 trường.

b- Tính tỷ số phần trăm của học sinh đạt loại giỏi (trên 25 điểm), loại kém (dưới 5 điểm).

3.8. Bạn hãy biểu thị các biểu thức đại số của các phép toán không cơ bản qua 5 phép toán cơ bản hợp, hiệu, tích decac, chiếu, chọn và từ đó viết các câu lệnh SQL tương ứng.

3.9. Giả sử chúng ta có CSDL tuyển sinh đại học như sau: quan hệ cho biết các khu vực kv (QUE, KV), quan hệ cho biết danh sách thí sinh tham gia thi ds (TT, HOTEN, GT, QUE, NS, SBD, T, L, H) gồm các thuộc tính TT-thứ tự; HOTEN- họ tên; GT- giới tính; QUE- quê; NS- năm sinh; SBD- số báo danh; T, L, H là các điểm toán lý hoá tương ứng. Giả sử chúng ta có 4 khu vực : 0, 1, 2, 3; Điểm được cộng thêm cho tong khu vực tương ứng là 0,5 điểm cho khu vực 0; 1 điểm cho khu vực 1; 1,5 điểm cho khu vực 2; 2 điểm cho khu vực 3. Hãy tìm cách (có thể xây dựng lai các quan hệ) để thực hiện:

a- In danh sách thí sinh đậu đại học, biết rằng điểm sàn trúng tuyển là 25.

b- In điểm trung bình của từng môn thi theo khu vực.

c- In điểm trung bình theo giới tính.

3.10. Hãy xây dung CSDL nhân sự của một cơ quan và tìm cách thực hiện:

a- In danh sách họ tên những người có lương trên 5 triệu.

b- In lương trung bình của những người ở độ tuổi dưới 30, độ tuổi trên 50.

c- In danh sách họ tên các GS. TSKH (giáo sư tiến sỹ khoa học) có lương dưới 3 triệu.

3.11. Hãy dùng CSDL trong thí dụ 3.2 và thực hiện:

a- Tính giá trung bình của các mặt hàng có tên bắt dầu bằng ru.

b- In mặt hàng có giá thấp nhất của các nhà cung cấp.

c- In những nhà cung cấp có ít nhất hai mặt hàng Hoa mua.

d- In những nhà cung cấp Hoa chỉ mua dưới 3 mặt hàng.

e- In những nhà cung cấp mà Hoa đã mua hàng với tổng tiền trên 100 triệu.

3.12. Hãy xây dựng một CSDL nhân sự cho một cơ quan thí dụ gồm các quan hệ: NV- nhân viên, L- lương, DV- đảng viên, DOV- đoàn viên.

a- In những đảng viên có lương trên 6 triệu.

b- In những đảng viên là đoàn viên.

c- In những nhân viên không đảng viên và đoàn viên.

d- In những đảng viên có lương cao hơn tất cả các đoàn viên.

e- Tính lương trung bình của các đoàn viên.

f- Tính lương trung bình của các đảng viên.

g- In số lượng các đoàn viên mà đảng viên.

CHƯƠNG 4

CÁC PHƯƠNG PHÁP PHÂN TÁN DỮ LIỆU

Thiết kế hệ thống máy tính phân tán bao gồm:

- Phân tán và chọn những vị trí đặt dữ liệu.
- Các chương trình ứng dụng trên mạng máy tính đó và
- Thiết kế tổ chức khai thác hệ thống đó trên mạng.

Trong chương này chúng ta chỉ tập trung nghiên cứu mục thứ nhất là *các phương pháp phân tán dữ liệu và cấp phát dữ liệu.*

Các quan hệ được dùng làm cơ sở phân tán trong cuốn sách này. Việc phân tán các quan hệ, chúng ta thường chia chúng thành nhiều quan hệ nhỏ hơn gọi là các mảnh (fragment) và chính các mảnh sẽ được phân tán tiếp. Vì vậy hoạt động phân tán gồm có hai bước :

- *Phân mảnh (fragmentation).*
- *Cấp phát (allocation) các mảnh cho các vị trí .*

4.1. KHÁI NIỆM VỀ PHÂN TÁN DỮ LIỆU

Trong mạng máy tính nói chung dữ liệu phải phân mảnh. Tại sao phải làm như vậy?

Tập các câu hỏi dưới đây sẽ bao quát toàn bộ vấn đề của phân mảnh.

* Tại sao lại cần phải phân mảnh?

* Làm thế nào để thực hiện phân mảnh?

* Phân mảnh nên thực hiện đến mức độ nào?

* Có cách gì kiểm tra tính đúng đắn của việc phân mảnh này hay không?

* Chúng ta sẽ cấp phát như thế nào trên mạng khi đã phân thành các mảnh ?

* Những thông tin nào sẽ cần thiết cho việc phân mảnh và cấp phát?

4.1.1. CÁC LÝ DO PHÂN MẢNH

Trước tiên việc phân tán (hoặc phân mảnh) được thực hiện dựa trên cơ sở cấp phát các tập tin cho các nút trên một mạng máy tính. Sự thực là trên một mạng máy tính diện rộng, mỗi nút ở một vị trí khác nhau nên CSDL không thể để tập trung mà phải phân tán trên các nút của mạng.

Hơn nữa một quan hệ không phải là một đơn vị truy xuất dữ liệu tốt. Thí dụ nếu các ứng dụng được thực hiện trên một phần của quan hệ mà quan hệ đó lại nằm tại những vị trí khác nhau thì có thể gây ra những truy xuất thừa và hơn thế việc nhân bản (sao) các quan hệ làm tốn không gian nhớ.

Ngoài ra việc phân rã một quan hệ thành nhiều mảnh, mỗi mảnh được xử lý như một đơn vị, sẽ cho phép thực hiện nhiều giao dịch đồng thời. Vậy việc phân mảnh các quan hệ sẽ *cho phép thực hiện song song một tập câu vấn tin* bằng cách chia nó thành một tập các câu vấn tin con hoạt tác trên các mảnh. *Vì thế việc phân mảnh sẽ làm tăng mức độ hoạt động đồng thời (song song)* và như thế làm tăng lưu lượng hoạt động của hệ thống.

Tuy nhiên chúng ta cũng sẽ gặp những rắc rối của việc phân mảnh, ví dụ nếu các ứng dụng có những "xung đột" sẽ ngăn cản hoặc gây khó khăn cho việc truy xuất dữ liệu. Phân rã thành các mảnh nói chung làm tăng chi

phí trong việc truy xuất dữ liệu. Một vấn đề nữa liên quan đến việc *kiểm soát dữ liệu ngữ nghĩa (semantic data control), đặc biệt là vấn đề kiểm tra tính toàn vẹn dữ liệu.*

4.1.2.CÁC KIỂU PHÂN MẢNH

Thể hiện của các quan hệ chính là các bảng, vì thế vấn đề là tìm những cách khác nhau để *chia một bảng thành nhiều bảng nhỏ hơn.* Rõ ràng có hai phương pháp khác nhau : Chia bảng theo chiều dọc và chia bảng theo chiều ngang. Chia dọc ta được các quan hệ con mà mỗi quan hệ chứa một tập con các thuộc tính (các cột) của quan hệ gốc- gọi là *phân mảnh dọc.* Chia ngang một quan hệ ta được các quan hệ con mà mỗi quan hệ chứa một số bộ của quan hệ gốc- gọi là *phân mảnh ngang.*

Ngoài ra còn có một khả năng hỗn hợp, đó là phân mảnh kết hợp hai cách ngang và dọc. Tất nhiên quá trình phân mảnh gắn liền với vấn đề cấp phát và bài toán cụ thể, vì vậy những vấn đề chúng ta bàn luận dưới đây chỉ có tính chất minh hoạ cho một cách phân mảnh. Cần lưu ý một điều quan trọng trong phần này là hiện tại chúng ta chưa có một thuật toán tổng quát tối ưu cho bài toán phân mảnh tổng quát và cấp phát dữ liệu trên mạng.

Thí dụ 4.1:

Trong thí dụ này chúng ta sử dụng một CSDL của một công ty máy tính thực hiện các dự án phần mềm gồm các quan hệ PROJ(dự án), EMP(nhân viên), ASG(trách nhiệm) và bảng lương PAY.

Cụ thể quan hệ PROJ(PNO,PNAME,BUDGET,LOC) có các thuộc tính : PNO, PNAME, BUDGET, LOC để chỉ mã, tên, kinh phí và vị trí của

dự án. Quan hệ nhân viên EMP(ENO,ENAME,TITLE) có các thuộc tính mã, tên, chức vụ của nhân viên. Quan hệ ASG(ENO,PNO,RESP,DUR) có các thuộc tính mã nhân viên, mã dự án nhân viên tham gia, trách nhiệm nhân viên trong dự án, thời gian tham gia dự án của nhân viên (xem hình 4.1). Hình 4.2 trình bày quan hệ của hình 4.1 được chia ngang thành hai quan hệ. Quan hệ con $PROJ_1$ chứa các thông tin về các dự án có ngân sách dưới 200.000 đô la còn quan hệ con $PROJ_2$ lưu các thông tin về các dự án có ngân sách lớn hơn 200.000đô la.

EMP

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	M. Smith	Syst. Anal
E3	A. Lee	Mech. Eng.
E4	J. Miler	Programmer
E5	B. Casey	Syst, Anal
E6	L. Chu	Elect. Eng
E7	R. David	Mech. Eng.
E8	J. Jones	Syst. Anal.

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal
P2	Database Develop	135000	New york
P3	CAD/CAM	250000	New york
P4	Maintenance	310000	Paris

PAY

TITLE	SAL
Elect. Eng	40000
Syst. Anal	34000
Mech. Eng	27000
Programmer	24000

Hình 4.1 CSDL của công ty máy tính

$PROJ_1$

PNO	PNAME	GUDGET	LOC
P1	Intrumentation	150000	Montreal
P2	Database Develop	135000	New York

$PROJ_2$

PNO	PNAME	GUDGET	LOC
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

Hình 4.2 Thí dụ về phân mảnh ngang

Chú ý rằng trong hình 4.1 ta có bốn quan hệ : nhân viên EMP với tập thuộc tính (ENO, ENAME, TITLE), ASG với tập thuộc tính (ENO, PNO, RESP;

DUR), PROJ với tập thuộc tính (PNO, PNAME, BUDGET, LOC), PAY với tập thuộc tính (TITLE, SAL).

Thí dụ 4.2:

Hình 4.3 trình bày quan hệ PROJ của Hình 4.1 được phân mảnh dọc thành hai quan hệ con, $PROJ_1$ và $PROJ_2$. $PROJ_1$ chỉ chứa thông tin về ngân sách các dự án còn $PROJ_2$ chứa tên và vị trí dự án. Điều quan trọng cần chú ý là khóa của quan hệ (PNO) có mặt trong cả hai mảnh.

$PROJ_1$

PNO	BUDGET
P1	150000
P2	135000
P3	250000
P4	310000

$PROJ_2$

PNO	PNAME	LOC
P1	Instrumentation	Montreal
P2	Database Develop	NewYork
P3	CAD/CAM	NewYork
P4	Maintenance	Paris

Hình 4.3 Thí dụ về phân mảnh dọc

Dĩ nhiên việc phân mảnh có thể lồng ghép với nhau. Nếu có lồng các loại phân mảnh khác nhau, chúng ta thu được một phân mảnh hỗn hợp (hybrid fragmentation). Trong thực tế có rất nhiều phân mảnh thuộc loại hỗn hợp.

4.1.3.MỨC ĐỘ PHÂN MẢNH

Phân mảnh CSDL đến mức độ nào là một quyết định rất quan trọng, có ảnh hưởng đến hiệu năng thực hiện vấn tin. Mức độ phân mảnh có thể là từ thái cực không phân thành mảnh nào đến thái cực phân mảnh thành từng bộ (trường hợp phân mảnh ngang) hoặc thành từng thuộc tính (trường hợp phân mảnh dọc).

Chúng ta sẽ nói đến "Các tác dụng phụ" của các đơn vị phân mảnh quá lớn và quá nhỏ. Vậy *điều chúng ta cần là tìm ra được một mức độ phân mảnh thích hợp*. Một mức độ như thế chỉ được định nghĩa với các ứng dụng cụ thể trên CSDL. Vấn đề là sẽ thực hiện như thế nào? Nói chung, các ứng dụng cần được đặc trưng qua một số tham số. Theo giá trị của những tham số này mà chúng ta có thể xác định được từng mảnh. Để minh hoạ cho phần này ta sẽ xét một phương pháp phân mảnh theo tụ lực (affinity) của các thuộc tính trong phần tiếp theo.

4.1.4.QUY TẮC PHÂN MẢNH ĐÚNG ĐẮN

Chúng ta sẽ tuân thủ ba quy tắc trong khi phân mảnh mà chúng bảo đảm rằng CSDL sẽ không có thay đổi nào về ngữ nghĩa khi phân mảnh.

1- *Tính đầy đủ (completeness).* Nếu một quan hệ R được phân rã thành các mảnh R1, R2, ... Rn, thì mỗi mục dữ liệu có trong R phải có mặt trong một hoặc nhiều mảnh R_i.

2- *Tính tái thiết được (reconstruction).* Nếu một quan hệ R được phân rã thành các mảnh $P_R = \{R_1, R_2, ..., R_n\}$thì cần phải định nghĩa một toán tử Θ tái thiết quan hệ gốc R sao cho :

$$R = \Theta R_i, \quad \forall R_i \in P_R$$

toán tử Θ thay đổi tùy theo từng loại phân mảnh; tuy nhiên điều quan trọng là phải xác định được nó. Thông thường khi phân mảnh ngang thì Θ là phép hợp còn trong phân mảnh dọc Θ là phép nối.

3. *Tính tách biệt (disjointness).* Nếu quan hệ R được phân rã ngang thành các mảnh R_1, R_2, ... R_n và mục dữ liệu t_j nằm trong mảnh R_j, thì nó sẽ không nằm trong một mảnh R_k khác ($k \neq j$). Tiêu chuẩn này bảo đảm rằng các mảnh ngang sẽ tách biệt (rời nhau). Nếu quan hệ được phân rã dọc, các thuộc tính khóa chính phải được lập lại trong mỗi mảnh. Vì thế trong trường hợp phân mảnh dọc, tính tách biệt chỉ được định nghĩa trên các trường không phải là khóa chính của một quan hệ.

4.1.5. CÁC KIỂU CẤP PHÁT

Giả sử rằng CSDL đã được phân mảnh thích hợp và cần phải quyết định cấp phát các mảnh cho các vị trí trên mạng. Khi dữ liệu được cấp phát, nó có thể được *nhân bản* hoặc chỉ duy trì *một bản duy nhất*.

Lý do cần phải nhân bản là nhằm bảo đảm được độ tin cậy và hiệu quả cho các câu vấn tin chỉ đọc. Nếu có nhiều bản sao của một mục dữ liệu thì chúng ta vẫn có cơ hội truy xuất được dữ liệu đó ngay cả khi hệ thống xảy ra sự cố. Hơn nữa các câu vấn tin chỉ đọc truy xuất đến cùng một mục dữ liệu có thể cho thực hiện song song bởi vì các bản sao có mặt tại nhiều vị trí. Ngược lại câu vấn tin cập nhật có thể gây ra nhiều rắc rối bởi vì hệ thống phải bảo đảm rằng tất cả các bản sao phải được cập nhật chính xác. Vì vậy quyết định nhân bản cần phải được cân nhắc và phụ thuộc vào tỷ lệ giữa các câu vấn tin chỉ đọc và các câu vấn tin cập nhật. Quyết định này hầu như đều

có ảnh hưởng đến tất cả các thuật toán của DBMS phân tán và các chức năng kiểm soát khác.

4.1.6.CÁC YÊU CẦU THÔNG TIN

Một điều cần lưu ý trong việc thiết kế phân tán là quá nhiều yếu tố có ảnh hưởng đến một thiết kế tối ưu. Tổ chức logic của CSDL, vị trí các ứng dụng, đặc tính truy xuất của các ứng dụng đến CSDL, và các đặc tính của hệ thống máy tính tại mỗi vị trí đều có ảnh hưởng đến các quyết định phân tán. Điều này khiến cho việc diễn đạt bài toán phân tán trở nên hết sức phức tạp.

Các thông tin cần cho thiết kế phân tán có thể phân chia thành bốn loại:

- Thông tin CSDL, đây là tập các quan hệ dữ liệu.
- Thông tin ứng dụng, đây là tập các câu vấn tin trên các quan hệ.
- Thông tin về mạng
- Thông tin về hệ thống máy tính.

4. 2. PHÂN MẢNH NGANG

Trong phần này chúng ta sẽ bàn đến các khái niệm liên quan đến phân mảnh ngang (phân tán ngang). Có hai chiến lược phân mảnh ngang cơ bản :

- *Phân mảnh ngang nguyên thủy.*
- *Phân mảnh ngang dẫn xuất.*

Phân mảnh ngang nguyên thủy (primary horizontal fragmentation) của một quan hệ được thực hiện dựa trên các vị từ được định nghĩa trên quan hệ

đó. Ngược lại phân mảnh ngang dẫn xuất (derived horizontal fragmentation) là phân mảnh một quan hệ dựa vào các vị từ được định nghĩa trên một quan hệ khác.

4.2.1. HAI KIỂU PHÂN MẢNH NGANG

Như đã giải thích ở phần trên, phân mảnh ngang chia một quan hệ r theo các bộ. Vì vậy mỗi mảnh là một tập con các bộ t của quan hệ r.

Phân mảnh ngang nguyên thủy (primary horizontal fragmentation) của một quan hệ được thực hiện dựa trên các vị từ được định nghĩa trên quan hệ đó. Ngược lại *phân mảnh ngang dẫn xuất* (derived horizontal fragmentation) là phân mảnh một quan hệ dựa vào các vị từ được định nghĩa trên một quan hệ khác. Như vậy trong phân mảnh ngang *tập các vị từ* đóng vai trò quan trọng.

Trong phần này sẽ xem xét các thuật toán thực hiện các kiểu phân mảnh ngang. Trước tiên chúng ta nêu các thông tin cần thiết để thực hiện phân mảnh ngang.

4.2.2.YÊU CẦU THÔNG TIN CỦA PHÂN MẢNH NGANG

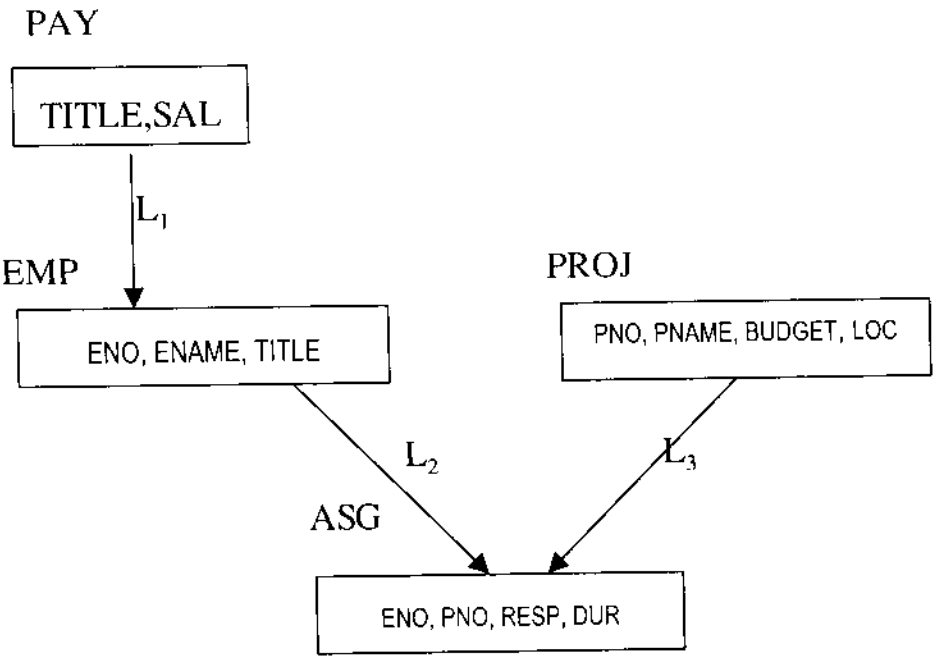
4.2.2.1 Thông tin về cơ sở dữ liệu

Thông tin về CSDL muốn nói đến là lược đồ toàn cục và quan hệ gốc, các quan hệ con. Trong ngữ cảnh này, chúng ta cần biết được các quan hệ sẽ kết lại với nhau bằng phép nối hay phép tính khác. Với mục đích phân mảnh dẫn xuất, các vị từ được định nghĩa trên quan hệ khác. Để xác định "quan hệ khác" ta thường dùng mô hình thực thể - liên hệ (entity - relationship

model), vì trong mô hình này các mối liên hệ được biểu diễn bằng các đường nối có hướng (các cung) giữa các quan hệ có liên hệ với nhau qua một nối .

Thí dụ 4.3:

Hình 4.4 trình bày một cách biểu diễn các đường nối giữa các quan hệ. Chú ý rằng hướng của đường nối cho biết mối liên hệ một - nhiều. Chẳng hạn với mỗi chức vụ (title) có nhiều nhân viên giữ chức vụ đó, vì thế chúng ta vẽ một đường nối từ quan hệ PAY hướng đến EMP. Đồng thời mối liên hệ nhiều - nhiều giữa EMP và PROJ được biểu diễn bằng hai đường nối đến quan hệ ASG.



Hình 4.4 Biểu diễn mối liên hệ giữa các quan hệ nhờ các đường nối.

Quan hệ nằm tại đầu (không mũi tên) của đường nối được gọi là chủ nhân (owner) của đường nối và quan hệ tại cuối đường nối (phía mũi tên) gọi là thành viên (member). Một số thuật ngữ được sử dụng thông dụng trong mô hình mạng là quan hệ nguồn (source relation) cho chủ nhân và quan hệ đích (target relation) cho thành viên. Chúng ta hãy định nghĩa hai hàm : owner và member, chúng sẽ trả về quan hệ thành viên hoặc quan hệ chủ nhân của đường nối.

Thí dụ 4.4:

Cho đường nối L_1 của hình 4.4, các hàm owner và member có các giá trị sau :

owner (L_1) = PAY

member (L_1) = EMP

Thông tin định lượng cần có về CSDL là lực lượng (cardinality) của mỗi quan hệ R, đó là số bộ có trong R, được ký hiệu là card (R)

4.2.2.2 Thông tin về ứng dụng

Để phân tán ngoài thông tin định lượng Card(R) ta còn cần thông tin định tính cơ bản gồm các vị từ được dùng trong các câu vấn tin. Lượng thông tin này phụ thuộc bài toán cụ thể.

Nếu không thể phân tích được hết tất cả các ứng dụng để xác định những vị từ này thì ít nhất cũng phải nghiên cứu được các ứng dụng "quan trọng" nhất.

Vậy chúng ta phải tìm cách xác định *các vị từ đơn giản (simple predicate)*. Cho lược đồ quan hệ R (A_1, A_2, ... A_n), trong đó mỗi A_j là một thuộc tính có miền trị $D(A_j)$ (hay D_j).

Một vị từ đơn giản P được định nghĩa trên R có dạng:

P : A_j θ Value

Trong đó θ ∈ {=, <, ≠ , ≤, > , ≥ }, A_j là một thuộc tính và Value ∈ $D(A_j)$.

Như vậy cho trước lược đồ R, nếu các miền trị D_j hữu hạn chúng ta có thể xác định được tập tất cả các vị từ đơn giản Pr trên R.

Vậy Pr = { P : A_j θ Value }. Tuy nhiên trong thực tế ta chỉ cần những tập con thực sự của Pr.

Thí dụ 4.5:

Cho quan hệ PROJ của hình 4.1

P_1: PNAME = "Maintenance"

P_2: BUDGET ≤ 200000

là các vị từ đơn giản..

Hay ta có quan hệ r

r

A	B	C	D
1	2	3	4
1	3	2	5
0	1	1	0

Khi đó P_1 : A = 0

P_2 : B < 2

P_3 : C ≠ 3 là những vị từ đơn giản trên R.

Trong các bài toán thực tế các câu vấn tin thường chứa nhiều vị từ phức tạp hơn, là tổ hợp của các vị từ đơn giản. Thí dụ *vị từ hội sơ cấp (conjunction)* của các vị từ đơn giản. Bởi vì chúng ta luôn có thể biến đổi một biểu thức boole thành dạng chuẩn hội (conjuntive normal form), nên ta chỉ sử dụng vị từ hội sơ cấp trong các thảo luận về sau.

Cho một tập $Pr = \{p_1, p_2, \ldots .p_m\}$ là tập các vị từ đơn giản trên R.

Tập các vị từ hội sơ cấp $M = \{m_1, m_2, \ldots m_z\}$ được định nghĩa là

$m_i = \quad \wedge \ p^*_{ik}$ với $1 \le ik \le m$.

Trong đó $p^*_{ik} = p_{ik}$ hoặc $p^*_{ik} = \neg p_{ik}$(với $\neg p_{ik}$ là phủ định của p_{ik}).

Vì thế mỗi vị từ đơn giản có thể xuất hiện trong vị từ hội sơ cấp dưới dạng tự nhiên hoặc dạng phủ định.

Một điểm quan trọng cần lưu ý ở đây là phép lấy phủ định không phải lúc nào cũng thực hiện được. Thí dụ xét hai vị từ đơn giản.

Cận - dưới $\le$ A ; A $\le$ Cận - trên.

Tức là thuộc tính A có miền trị nằm trong cận dưới và cận trên, khi đó phần bù của chúng là :

$\neg$ (Cận - dưới $\le$ A)

$\neg$ (A $\le$ Cận - trên) không xác định được. Giá trị của A trong các phủ định này đã ra khỏi miền trị của A.

hoặc hai vị từ đơn giản trên có thể được viết lại là

(Cận - dưới $\le$ A $\le$ Cận - trên)

Với phần bù là :

$\neg$ (Cận - dưới $\le$ A $\le$ Cận - trên) không định nghĩa được. Vì vậy khi nghiên cứu những vấn đề này ta chỉ xem xét các vị từ đẳng thức đơn giản.

Thí dụ 4.6:

Xét quan hệ PAY của hình 4.1. Dưới đây là một số vị từ đơn giản có thể định nghĩa được trên PAY.

P_1 : TITLE = "Elect. Eng"

P_2 : TITLE = "Programmer"

P_3 : SAL ≤ 30000

P_4 : SAL > 30000

Và một số các vị từ hội sơ cấp được định nghĩa dựa trên các vị từ đơn giản này.

m_1 : TITLE = "Elect.Eng" ∧ SAL ≤ 30000

m_2 : TITLE = "Elect.Eng" ∧ SAL > 30000

m_3 : ¬(TITLE = "Elect.Eng")∧ SAL ≤ 30000

m_4 : ¬(TITLE = "Elect.Eng")∧ SAL > 30000

m_5 : TITLE = "Programmer" ∧ SAL ≤ 30000

m_6 : TITLE = "Programmer" ∧ SAL > 30000

Vậy có hai điều cần chú ý: Thứ nhất, như ta đã thấy, không phải tất cả các vị từ hội sơ cấp đều có thể định nghĩa được.

Thứ hai, một số trong chúng có thể vô nghĩa đối với ngữ nghĩa của quan hệ PAY. Ngoài ra cần chú ý rằng m_3 có thể được viết lại như sau :

m_3 : TITLE ≠ "Elect. Eng" ∧ SAL ≤ 30000

Theo những thông tin định tính về các ứng dụng, chúng ta cần biết hai tập dữ liệu.

1. Độ tuyển hội sơ cấp (minterm selectivity): số lượng các bộ của quan hệ sẽ được truy xuất bởi câu vấn tin được đặc tả theo một vị từ hội sơ

cấp đã cho. Chẳng hạn độ tuyển của m_1 trong Thí dụ 4.6 là zero bởi vì không có bộ nào trong PAY thoả vị từ này. Độ tuyển của m_2 là 1. Ký hiệu độ tuyển của một hội sơ cấp m_i là sel (m_i).

2. Tần số truy xuất (access frequency) và trong một số trường hợp là số truy xuất. Nếu $Q = \{q_1, q_2, \ldots, q_q\}$ là tập các câu vấn tin, acc (q_i) biểu thị cho số truy xuất của q_i trong một khoảng thời gian đã cho, hoặc trong quan hệ cụ thể. Ta sẽ xem xét vấn đề này trong một bài toán cụ thể sau.

Chú ý rằng mỗi hội sơ cấp là một câu vấn tin. Ký hiệu tần số truy xuất của một hội sơ cấp là acc(m_i).

4.2.3. PHÂN MẢNH NGANG NGUYÊN THỦY

Trước khi trình bày thuật toán hình thức cho phân mảnh ngang, chúng ta sẽ thảo luận một cách trực quan về quá trình phân mảnh cho cả phân mảnh ngang nguyên thủy và phân mảnh ngang dẫn xuất. Phân mảnh ngang nguyên thủy được định nghĩa bằng một phép toán chọn trên các quan hệ chủ nhân của một lược đồ CSDL. Vì thế cho biết quan hệ R, các mảnh ngang của R là các R_i và $R_i = R(E_i), \quad 1 \le i \le z$

Trong đó E_i là công thức chọn (hội sơ cấp) được sử dụng để có được mảnh R_i. Chú ý rằng nếu E_i có dạng chuẩn hội, nó là một vị từ hội sơ cấp (m_j). R_i là các bộ của R thoả mãn E_i

Thí dụ 4.7:

Phân rã quan hệ PROJ thành các mảnh ngang $PROJ_1$ và $PROJ_2$ trong thí dụ 4.1 được định nghĩa như sau:

$PROJ_1$= PROJ(BUDGET $\le$ 200000)

$PROJ_2$= PROJ(BUDGET $>$ 200000)

Thí dụ 4.8:

Xét quan hệ PROJ của Hình 4.1.

Chúng ta có thể định nghĩa các mảnh ngang sau đây dựa vào vị trí dự án. Các mảnh thu được, được trình bày trong Hình 4.5.

$PROJ_1$= PROJ(LOC = "Montreal")

$PROJ_2$= PROJ(LOC = "New York")

$PROJ_3$= PROJ(LOC = "Pari")

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P_1	Instrumentation	150000	Montreal

$PROJ_2$

PNO	PNAME	BUDGET	LOC
0P2	Database Develop	135000	New York
P3	CAD/CAM	250000	New York

$PROJ_3$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

Hình 4.5 Phân mảnh ngang nguyên thủy cho quan hệ PROJ

Bây giờ chúng ta có thể định nghĩa một mảnh ngang chặt chẽ và rõ ràng hơn.

Một mảnh ngang R_i của quan hệ R có chứa tất cả các bộ R thoả vị từ hội sơ cấp m_i.

Vì vậy cho một tập các vị từ hội sơ cấp M, số lượng các mảnh ngang cũng bằng số lượng các vị từ hội sơ cấp (tất nhiên một mảnh ngang có thể rỗng nếu vị từ không truy xuất đến bộ nào của quan hệ, các mảnh ngang có thể bằng nhau nếu hai vị từ tương đương). Tập các mảnh ngang này cũng thường được gọi là tập các mảnh hội sơ cấp (minterm fragent).

Như đã thảo luận ở phần trước, rõ ràng là việc định nghĩa các mảnh ngang phụ thuộc vào các vị từ hội sơ cấp. Vì thế bước đầu tiên của mọi thuật toán phân mảnh là phải xác định tập các vị từ đơn giản sẽ tạo ra các vị từ hội sơ cấp. Tập vị từ này nói chung phụ thuộc mục tiêu và yêu cầu của bài toán.

Một khái niệm quan trọng của các vị từ đơn giản là *tính đầy đủ (completeness) và tính cực tiểu (minimality)* của tập các vị từ.

Tập các vị từ đơn giản Pr *được gọi là đầy đủ nếu và chỉ nếu xác suất của mỗi bộ của R (hoặc của các mảnh Ri) được truy xuất bởi các p của Pr đều bằng nhau.*

Thí dụ 4.9:

Xét phân mảnh quan hệ PROJ được đưa ra trong thí dụ 4.5. Nếu tập ứng dụng Pr = {LOC = "Montreal", LOC = "New York", LOC = "Paris",

BUDGET ≤ 200000} thì Pr không đầy đủ vì có một số bộ của PROJ không được truy xuất bởi vị từ BUDGET ≤ 200000. Để cho tập vị từ đầy đủ, chúng ta cần phải thêm vị từ BUDGET > 200000 vào Pr. Vậy

Pr = {LOC = "Montreal", LOC = "New York", LOC = "Paris",

BUDGET ≤ 200000, BUDGET > 200000} là đầy đủ bởi vì mỗi bộ được truy xuất bởi đúng 2 vị từ p của Pr. Tất nhiên nếu ta bớt đi một vị từ bất kỳ trong Pr thì tập còn lại không đầy đủ.

Lý do cần phải bảo đảm tính đầy đủ là vì các mảnh thu được theo tập vị từ đầy đủ sẽ nhất quán về mặt logic do tất cả chúng đều thoả vị từ hội sơ cấp. Chúng cũng đồng nhất và đầy đủ về mặt thống kê theo cách mà ứng dụng truy xuất chúng.

Vì thế chúng ta sẽ dùng một tập hợp gồm các vị từ đầy đủ làm cơ sở của phân mảnh ngang nguyên thủy.

Đặc tính thứ hai của tập các vị từ là tính cực tiểu. Đây là một đặc tính cảm tính. Vị từ đơn giản phải có liên đới (relevant) trong việc xác định một phân mảnh. Một vị từ mà không tham gia vào một phân mảnh nào thì có thể coi vị từ đó là thừa. *Nếu tất cả các vị từ của tập Pr đều có liên đới và không có các vị từ tương đương thì Pr là cực tiểu.*

Thí dụ 4.10:

Tập Pr được định nghĩa trong thí dụ 4.9 là đầy đủ và cực tiểu. Tuy nhiên nếu chúng ta thêm vị từ

PNAME = "Instrumentation" vào Pr, tập kết quả sẽ không còn cực tiểu bởi vì vị từ mới thêm vào không có liên đới ứng với Pr. Vị từ mới thêm vào không chia thêm mảnh nào trong các mảnh đã được tạo ra.

Chúng ta cần lưu ý khái niệm đầy đủ gắn chặt với mục tiêu của bài toán. Số các vị từ phải đầy đủ theo yêu cầu của bài toán chúng ta mới thực hiện được những vấn đề đặt ra. Khái niệm cực tiểu liên quan đến vấn đề tối ưu của bộ nhớ, tối ưu của các thao tác trên tập các câu vấn tin. Vậy khi cho trước một tập các vị từ Pr, để xét tính cực tiểu chúng ta có thể kiểm tra bằng

cách vứt bỏ những vị từ thừa để có tập vị từ Pr' là cực tiểu và tất nhiên Pr' cũng là tập đầy đủ đối với Pr. Sau đây là một thí dụ về việc tìm một tập vị từ đầy đủ và cực tiểu Pr' từ tập Pr. Thuật toán này gọi là thuật toán COM-MIN.

Ta tạm quy ước :

Quy tắc 1 : Quy tắc cơ bản về tính đầy đủ và cực tiểu của Pr'là một quan hệ hoặc một mảnh được phân hoạch "thành ít nhất hai phần bởi mỗi vị từ của Pr' ".

Thuật toán 4.1 COM - MIN

Input R : quan hệ ; Pr : tập các vị từ đơn giản ;

Output Pr' : Tập các vị từ đơn giản đầy đủ và cực tiểu;

declare

F : tập các mảnh hội sơ cấp;

Begin

Pr' = Ø; F = Ø;

For each vị từ p ∈ Pr if p phân hoạch R theo Quy tắc 1 then

Begin

Pr':= Pr' ∪ p ;

Pr: = Pr – P ;

F : = F ∪ f ;

{f là mảnh hội sơ cấp theo p}

End;{chúng ta đã chuyển các vị từ có phân mảnh R vào Pr'};

Repeat

begin

For each p ∈ Pr if p phân hoạch một mảnh f_k của Pr' theo quy tắc 1 then

Pr': = Pr' ∪ p ;

Pr: = Pr – p ;

F: = F ∪ f

End ;

Until Pr' đầy đủ {không còn p nào phân mảnh f_k của Pr' };

For each p ∈ Pr', if ∃ p' mà p <=> p' then

begin

Pr'; = Pr' – p ;

F : = F - f

end {đã kiểm tra tính cực tiểu};

end. (COM- MIN)

Thuật toán bắt đầu bằng cách tìm tất cả các vị từ có liên đới và phân hoạch quan hệ đã cho. Vòng lặp repeat-until thêm các vị từ có phân hoạch các mảnh vào tập này, bảo đảm tính đầy đủ của Pr'. Đoạn cuối kiểm tra lại tính cực tiểu của Pr'.Vì thế cuối cùng ta có tập Pr' là cực tiểu và đầy đủ. Các bạn cần lưu ý rằng thực chất của thuật toán COM-MIN ẩn chứa Pr đã đầy đủ. Bởi vì nếu chúng ta có tập Pr tuỳ tiện ví dụ nó chỉ chứa một vị từ A= value thì không việc gì phải bàn luận tiếp mà quan hệ gốc được phân làm hai mảnh một mảnh R_1 thoả A = value và mảnh còn lai là quan hệ R- R_1 .

Bước thứ hai của thiết kế phân mảnh ngang nguyên thủy là suy dẫn ra tập các vị từ hội sơ cấp có thể được định nghĩa trên các vị từ trong tập Pr'. Các vị từ hội sơ cấp này xác định các mảnh "ứng cử viên" cho bước cấp phát.

Bước thứ ba của quá trình thiết kế là loại bỏ một số mảnh vô nghĩa. Điều này được thực hiện bằng cách xác định những vị từ mâu thuẫn với tập các phép kéo theo (implication) I. Chẳng hạn, nếu Pr' = (P_1, P_2), trong đó

P_1 : A = value _ 1

P_2 : A = value _ 2

và miền biến thiên của A là {valuc _1, value _2}, rõ ràng I chứa hai phép kéo theo với khẳng định :

i_1 : (A = value _1) $\Rightarrow$ $\neg$(A =value _ 2)

i_2 : $\neg$ (A = value _1) $\Rightarrow$ (A =value _ 2)

Bốn vị từ hội sơ cấp sau đây được định nghĩa theo Pr'

m_1 : (A = value _1) $\wedge$ (A =value _ 2)

m_2 : (A = value _1) $\wedge$ $\neg$(A =value _ 2)

m_3 : $\neg$ (A = value _1) $\wedge$ (A =value _ 2)

m_4 : $\neg$ (A = value _1) $\wedge$ $\neg$(A =value _ 2)

Trong trường hợp này các vị từ hội sơ cấp m_1 và m_4 mâu thuẫn với các phép kéo theo I và vì thế bị loại ra khỏi M

Thuật toán phân mảnh ngang nguyên thủy được trình bày trong thuật toán 4.2 được gọi là thuật toán PHORIZONTAL .

Thuật toán 4.2 PHORIZONTAL

Input : R : quan hệ; Pr : Tập các vị từ đơn giản ;

Output : M : Tập các vị từ hội sơ cấp ;

begin

Pr' := COM _ MIN (R, Pr);

Xác định tập M các vị từ hội sơ cấp ;

Xác định tập I các phép kéo theo giữa các $P_i \in$ Pr' ;

for each $m_i \in M$ if m_i mâu thuẫn với I then

$M := M - m_i$;

end (PHORIZONTAL)

Thí dụ 4.11:

Bây giờ chúng ta hãy xét thiết kế của lược đồ CSDL được cho trong hình 4.4. Điều đầu tiên cần chú ý là có hai quan hệ cần phải phân mảnh ngang nguyên thủy : quan hệ PAY và PROJ.

Giả sử rằng chỉ có một ứng dụng truy xuất PAY. Ứng dụng này kiểm tra thông tin lương và xác định số lương sẽ tăng. Giả sử rằng các mẫu tin nhân viên được quản lý ở hai nơi, một nơi xử lý các mẫu tin có lương thấp hơn hay bằng 30.000 đô la và một nơi khác xử lý các mẫu tin của những nhân viên có lương cao hơn 30.000 đô la. Vì thế câu vấn tin được đưa ra ở cả hai nơi.

Tập vị từ đơn giản được sử dụng để phân hoạch quan hệ PAY là

P_1 : SAL $\leq$ 30000

P_2 : SAL $>$ 30000

Vì vậy cho ra tập các vị từ đơn giản khởi đầu là $Pr = (P_1, P_2)$. Áp dụng thuật toán COM _ MIN với i = 1 làm giá trị khởi đầu sinh ra $Pr' = (P_1)$. Đây là tập đầy đủ và cực tiểu vì P_2 không phân hoạch f_1 (là mảnh hội sơ cấp được tạo ra ứng với P_1) theo quy tắc 1. Ta có thể tạo ra các vị từ hội sơ cấp sau đây làm các phần tử của M.

m_1 : (SAL $\leq$ 30000)

m_2 : $\neg$ (SAL $\leq$ 30000) = SAL $>$ 30000

Sau đó chúng ta định nghĩa hai mảnh $F_s = (PAY_1, PAY_2)$ theo M (Hình 4.6)

PAY_1

TITLE	SAL
Mech. Eng.	27000
Programmer	24000

PAY_2

TITLE	SAL
Elect.Eng.	40000
Syst.Anal.	34000

Hình 4.6 Phân mảnh ngang cho quan hệ PAY

Xét quan hệ PROJ. Giả sử rằng có hai ứng dụng, ứng dụng đầu tiên được đưa ra tại ba vị trí và cần tìm tên và ngân sách của các dự án khi cho biết vị trí. Theo ký pháp SQL câu vấn tin được viết là:

SELECT PNAME, BUDGET

FROM PROJ

WHERE LOC = Value

Đối với ứng dụng này, các vị từ đơn giản có thể được dùng là:

P_1 : LOC = "Montreal"

P_2 : LOC = "New York"

P_3 : LOC = "Paris"

Ứng dụng thứ hai được đưa ra tại hai vị trí và phải do Ban điều hành dự án đưa ra. Những dự án có ngân sách dưới 200.000 đô la được quản lý tại một vị trí, còn những dự án có ngân sách lớn hơn được quản lý tại vị trí thứ hai. Vì thế các vị từ đơn giản phải được sử dụng để phân mảnh theo ứng dụng thứ hai là:

P_4 : BUDGET $\leq$ 200000

P_5 : BUDGET > 200000

Nếu kiểm tra bằng thuật toán COM _ MIN, tập Pr' = (P_1, P_2, P_3, P_4, P_5) rõ ràng là đầy đủ và cực tiểu.

Dựa trên Pr' chúng ta có thể định nghĩa sáu vị từ hội sơ cấp sau đây tạo ra M:

m_1 : (LOC = "Montreal") $\wedge$ (BUDGET $\leq$ 200000)

m_2 : (LOC = "Montreal") $\wedge$ (BUDGET > 200000)

m_3 : (LOC = "New York") $\wedge$ (BUDGET $\leq$ 200000)

m_4 : (LOC = "New York") $\wedge$ (BUDGET > 200000)

m_5 : (LOC = "Paris") $\wedge$ (BUDGET $\leq$ 200000)

m_6 : (LOC = "Paris") $\wedge$ (BUDGET > 200000)

Đây không phải là các vị từ hội sơ cấp duy nhất có thể được tạo ra. Chẳng hạn vẫn có thể định nghĩa các vị từ

$P_1 \wedge P_2 \wedge P_3 \wedge P_4 \wedge P_5$

Tuy nhiên các phép kéo theo hiển nhiên là:

i_1 : $P_1 \Rightarrow \neg P_2 \wedge \neg P_3$ ($\neg$ là dấu phủ định)

i_2 : $P_2 \Rightarrow \neg P_1 \wedge \neg P_3$

i_3 : $P_3 \Rightarrow \neg P_1 \wedge \neg P_2$

i_4 : $P_4 \Rightarrow \neg P_5$

i_5 : $P_5 \Rightarrow \neg P_4$

i_6 : $\neg P_4 \Rightarrow P_5$

i_7 : $\neg P_5 \Rightarrow P_4$

Cho phép loại bỏ những vị từ hội sơ cấp này và chúng ta còn lại m_1 đến m_6. Xem xét thể hiện của CSDL trong hình 4.1, chúng ta có thể chứng tỏ rằng các phép kéo theo dưới đây đúng:

i_8 : (LOC = "Montreal") $\Rightarrow \neg$ (BUDGET > 200000)

i_9 : (LOC = "Paris") $\Rightarrow \neg$ (BUDGET $\leq$ 200000)

i_{10} : ¬ (LOC = "Montreal") ⇒ (BUDGET ≤ 200000)

i_{11} : ¬ (LOC = "Paris") ⇒ (BUDGET > 200000)

Tuy nhiên cần nhớ rằng các phép kéo theo phải được định nghĩa theo ngữ nghĩa của CSDL, không phải theo các giá trị hiện tại. Một số mảnh được định nghĩa theo M = (m_1, ... m_5) có thể rỗng nhưng chúng vẫn là các mảnh. Qua ngữ nghĩa của CSDL không có bằng chứng nào cho thấy rằng các phép kéo theo từ i_8 đến i_{11} là đúng.

Kết quả của phân mảnh ngang nguyên thủy cho PROJ là tạo ra sáu mảnh ($PROJ_1$, $PROJ_2$, $PROJ_3$, $PROJ_4$, $PROJ_5$, $PROJ_6$,) của quan hệ PROJ theo các vị từ hội sơ cấp M (hình 4.7). Chúng ta cũng cần chú ý rằng có một số mảnh rỗng ($PROJ_2$, $PROJ_5$,) và vì thế không được trình bày trong hình 4.7.

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

$PROJ_3$

PNO	PNAME	BUDGET	LOC
P2	Database Develop	135000	New York

$PROJ_4$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York

$PROJ_6$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	31000	Paris

Hình 4.7 Phân hoạch ngang cho quan hệ PROJ.

4.2.4.PHÂN MẢNH NGANG DẪN XUẤT

Phân mảnh ngang dẫn xuất được định nghĩa trên một quan hệ thành viên của một đường nối dựa theo phép toán chọn trên quan hệ chủ nhân của đường nối đó. Như thế nếu cho trước một đường nối L, trong đó owner (L) = S và member (L) = R, các mảnh ngang dẫn xuất của R được định nghĩa là :

$$R_i = R \ |>< S_i, \ 1 \leq i \leq w$$

Trong đó w là số lượng các mảnh được định nghĩa trên R, và S_i = S(Ei) với E_i là công thức định nghĩa mảnh ngang nguyên thủy S_i

Thí dụ 4.12:

Xét đường nối L_1 trong hình 4.4, trong đó owner (L_1) = PAY và member (L_1) = EMP. Thế thì chúng ta có thể nhóm các kỹ sư (engineer) thành hai nhóm tùy theo lương : Nhóm có lương từ 30.000 đô la trở xuống và nhóm lương trên 30.000 đô la. Hai mảnh EMP_1 và EMP_2 được định nghĩa như sau :

$$EMP_1 = EMP \ |>< \ PAY_1$$

$$EMP_2 = EMP \ |>< \ PAY_2$$

Trong đó

$$PAY_1 = PAY(SAL \leq 30000)$$

$$PAY_2 = PAY(SAL > 30000)$$

Kết quả phân mảnh được trình bày trong hình 4.8.

EMP_1

ENO	ENAME	TITLE
E3	A.lee	Mech. Eng.
E4	J. Miller	Programmer
D7	R. David	Mech. Eng.

EMP_2

ENO	ENAME	TITLE
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst. Anal.
E5	B. Casey	Syst. Anal.
E6	L. Chu	Elect. Eng.
E8	J. Jones	Syst. Anal.

Hình 4.8 Phân mảnh ngang dẫn xuất của quan hệ EMP.

Muốn thực hiện phân mảnh ngang dẫn xuất, chúng ta cần ba nguyên liệu (input) : tập các phân hoạch của quan hệ chủ nhân (chẳng hạn PAY_1 và PAY_2 trong thí dụ 4.12), quan hệ thành viên, và tập các vị từ nối nửa giữa chủ nhân và thành viên (chẳng hạn EMP.TITLE = PAY.TITLE trong thí dụ 4.12). Thuật toán phân mảnh khi đó hoàn toàn tầm thường, vì thế chúng ta không trình bày ở đây.

Cũng có một vấn đề phức tạp cần phải chú ý. Trong lược đồ CSDL, chúng ta hay gặp nhiều đường nối đến một quan hệ R (thí dụ như trong hình 4.4, ASG có hai đường nối đến). Như thế có thể có nhiều cách phân mảnh ngang dẫn xuất cho R. Quyết định chọn cách phân mảnh nào cần dựa trên hai tiêu chuẩn sau:

1- Phân mảnh có đặc tính nối tốt hơn.

2- Phân mảnh được sử dụng trong nhiều ứng dụng hơn.

Tuy nhiên, việc áp dụng các tiêu chuẩn trên còn là một vấn đề rắc rối.

Thí dụ 4.13:

Chúng ta hãy tiếp tục với thiết kế phân tán cho CSDL đã bắt đầu từ Thí dụ 4.11. Chúng ta đã quyết định phân mảnh quan hệ EMP theo phân mảnh của PAY (thí dụ 4.12). Bây giờ hãy xét ASG. Giả sử rằng có hai ứng dụng sau :

1. Ứng dụng thứ nhất tìm tên các kỹ sư có làm việc tại một nơi nào đó. Ứng dụng này chạy ở cả ba trạm và truy xuất cao hơn các kỹ sư của các dự án ở những vị trí khác.

2- Tại mỗi trạm quản lý, nơi quản lý các mẫu tin nhân viên, người dùng muốn truy xuất đến các dự án đang được các nhân viên này thực hiện và cần biết xem họ sẽ làm việc với dự án đó trong bao lâu.

ứng dụng thứ nhất dẫn đến việc phân mảnh ASG theo các mảnh $PROJ_1$, $PROJ_3$, $PROJ_4$, và $PROJ_6$, của PROJ đã thu được trong thí dụ 4.11. Cần nhớ lại rằng

$PROJ_1$= PROJ(LOC = "Montreal" $\wedge$ BUDGET $\leq$ 200000)

$PROJ_3$= PROJ(LOC = "New York" $\wedge$ BUDGET $\leq$ 200000)

$PROJ_4$= PROJ(LOC = "New York" $\wedge$ BUDGET > 200000)

$PROJ_6$= PROJ(LOC = "Paris" $\wedge$ BUDGET > 200000)

Vì thế phân mảnh dẫn xuất của ASG theo $PROJ_1$, $PROJ_3$, $PROJ_4$, $PROJ_6$ được định nghĩa như sau :

ASG_1 = ASG |>< $PROJ_1$

ASG_2 = ASG |>< $PROJ_3$

ASG_3 = ASG |>< $PROJ_4$

ASG_4 = ASG |>< $PROJ_6$

Thể hiện của các mảnh này được trình bày trong hình 4.9

Câu vấn tin thứ hai có thể được viết bằng SQL như sau

```
SELECT    RESP, DUR
FROM      ASG, EMP_i
WHERE     ASG.ENO = EMP_i.ENO
```

Trong đó i = 1 hoặc i = 2, tùy thuộc vào nơi đưa ra câu vấn tin. Phân mảnh dẫn xuất của ASG theo phân mảnh của EMP được định nghĩa dưới đây và được trình bày trong hình 4.10.

$ASG_1 = ASG \ |>< \ EMP_1$

$ASG_2 = ASG \ |>< \ EMP_2$

ASG_1

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24

ASG_3

ENO	PNO	RESP	DUR
E3	P3	Consultant	10
E6	P3	Engineer	36
E7	P3	Manager	40

ASG_2

ENO	PNO	RESP	DUR
E2	P2	Analyst	6
E4	P2	Programmer	18
E5	P2	Manager	24

ASG_4

ENO	PNO	RESP	DUR
E3	P4	Engineer	48
E6	P4	Manager	48

Hình 4.9 Phân mảnh dẫn xuất của ASG ứng với PROJ

ASG_1

ENO	PNO	RESP	DUR
E3	P3	Consultant	02
E3	P4	Engineer	48
E4	P2	Programmer	18
E7	P3	Engineer	36

ASG_2

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E3	P2	Analyst	6
E4	P2	Manager	24
E5	P4	Manager	48
E6	P3	Manager	40

Hình 4.10 Phân mảnh dẫn xuất của ASG ứng với EMP

Thí dụ 4.13 minh hoạ hai điều :

1. Phân mảnh dẫn xuất có thể xảy ra dây chuyền trong đó một quan hệ được phân mảnh như hệ quả của một phân mảnh cho một quan hệ khác, và đến lượt nó lại làm cho các quan hệ khác phải phân mảnh (chẳng hạn như dây chuyền PAY - EMP - ASG).

2- Thường thì một quan hệ sẽ có nhiều cách phân mảnh dẫn xuất (thí dụ như quan hệ ASG). Chọn một lược đồ phân mảnh là một loại bài toán quyết định và sẽ được phân tích trong phần cấp phát.

4.2.5. KIỂM TRA TÍNH ĐÚNG ĐẮN CỦA PHÂN MẢNH NGANG

Bây giờ chúng ta cần phải kiểm tra các thuật toán phân mảnh đã được thảo luận ứng với ba tiêu chuẩn đúng đắn đã trình bày trong phần 4.1.

4.2.5.1.Tính đầy đủ

Tính đầy đủ (Completeness) của một phân mảnh ngang nguyên thủy dựa vào các vị từ chọn được dùng. Với điều kiện các vị từ chọn là đầy đủ, phân mảnh thu cũng được bảo đảm là đầy đủ, bởi vì cơ sở của thuật toán phân mảnh là một tập các vị từ cực tiểu và đầy đủ Pr', tính đầy đủ sẽ được bảo đảm với điều kiện là không có sai sót xảy ra khi định nghĩa Pr'.

Tính đầy đủ của một phân mảnh ngang dẫn xuất có khác chút ít. Khó khăn là do vị từ định nghĩa phân mảnh có liên quan đến hai quan hệ. Trước tiên chúng ta hãy định nghĩa qui tắc đầy đủ một cách hình thức rồi xem thử một thí dụ.

Gọi R là quan hệ thành viên của một đường nối mà chủ nhân là quan hệ S được phân mảnh thành $F_s = (S_1, S_2, \ldots S_w)$. Cũng gọi A là thuộc tính nối giữa R và S. Thế thì đối với mỗi bộ t của R, phải có một bộ t' của S sao cho

$$t.A = t'.A$$

Chẳng hạn không có bộ ASG nào có mã số dự án không nằm trong quan hệ PROJ. Tương tự không có bộ nào của EMP có giá trị TITLE không là một trong các giá trị TITLE có ở trong quan hệ PAY. Quy tắc này được gọi là ràng buộc toàn vẹn hay toàn vẹn tham chiếu (referential integrity), bảo đảm rằng mọi bộ trong các mảnh của quan hệ thành viên đều nằm trong quan hệ chủ nhân.

4.2.5.2.Tính tái thiết được

Tái thiết một quan hệ toàn cục từ các mảnh được thực hiện bằng toán tử hợp trong cả phân mảnh ngang nguyên thủy lẫn dẫn xuất. Vì thế một quan hệ R với phân mảnh $F_R = (R_1, R_2, \ldots R_w)$, chúng ta có

$$R = \cup R_i \qquad \forall R_i \in R_F$$

4.2.5.3.Tính tách biệt

Chúng ta dễ dàng thấy rằng tính tách rời của phân mảnh nguyên thủy là rõ ràng theo cách phân mảnh của ta.Với phân mảnh dẫn xuất, tính tách rời sẽ được bảo đảm nếu các vị từ hội sơ cấp xác định phân mảnh có tính loại trừ lẫn nhau (mutually exclusive).

Tuy nhiên, phân mảnh dẫn xuất có hàm chứa các nối nửa khiến vấn đề có phức tạp hơn. Tính tách rời có thể được bảo đảm nếu đồ thị nối thuộc loại đơn giản. Nếu nó không đơn giản thì cần phải xem xét các giá trị thực sự của các bộ. Nói chung chúng ta không muốn một bộ của một quan hệ thành viên nối với hai hoặc nhiều bộ của quan hệ chủ nhân khi những bộ này thuộc các mảnh khác nhau của chủ nhân.

Thí dụ 4.14:

Khi phân mảnh quan hệ PAY (thí dụ 4.11), các vị từ hội sơ cấp

$M = (m_1, m_2)$ là :

m_1 : SAL $\leq$ 30000

m_2 : SAL $>$ 30000

Vì m_1 và m_2 loại trừ lẫn nhau, nên phân mảnh của PAY là tách biệt. Tuy nhiên với quan hệ EMP chúng ta đã yêu cầu là :

1. Mỗi kỹ sư có một chức vụ duy nhất.

2. Mỗi chức vụ có một giá trị lương duy nhất đi kèm với nó.

Vì hai quy tắc này suy ra từ ngữ nghĩa của CSDL, phân mảnh của EMP ứng với PAY cũng tách rời.

4.3. PHÂN MẢNH DỌC

Cần nhớ rằng một phân mảnh dọc cho một quan hệ r sinh ra các mảnh r_1, r_2, ... r_r, mỗi mảnh chứa một tập con thuộc tính của R và cả khóa của r. Mục đích của phân mảnh dọc là phân hoạch một quan hệ thành một tập các quan hệ nhỏ hơn để nhiều ứng dụng có thể chỉ chạy trên một quan hệ. Trong ngữ cảnh này, một phân mảnh "tối ưu" là một phân mảnh sinh ra một lược đồ phân mảnh cho phép giảm đến tối đa thời gian thực thi các ứng dụng chạy trên các mảnh đó.

Phân mảnh dọc đã được nghiên cứu trong ngữ cảnh của các hệ CSDL tập trung lẫn phân tán. Lý do chính trong ngữ cảnh tập trung là sử dụng nó làm một công cụ thiết kế, cho phép các vấn tin ảnh hưởng đến các quan hệ nhỏ hơn, vì thế giảm bớt số truy xuất và tiết kiệm bộ nhớ. Phân mảnh dọc tất nhiên là phức tạp hơn so với phân mảnh ngang. Điều này là do tổng số chọn lựa có thể có của phân hoạch dọc rất lớn.

Vì vậy để có được các lời giải tối ưu cho bài toán phân mảnh dọc thực sự rất khó;

Vì thế lại phải dùng các phương pháp khám phá (heuristic). Chúng ta đưa ra hai loại heuristic cho phân mảnh dọc các quan hệ toàn cục.

1. Nhóm thuộc tính : Bắt đầu bằng cách gán mỗi thuộc tính cho một mảnh, và tại mỗi bước, nối một số mảnh lại cho đến khi thoả một tiêu chuẩn nào đó. Kỹ thuật nhóm thuộc tính được đề xuất lần đầu trong [Hammer and Niamir, 1979] cho các CSDL tập trung và về sau được dùng trong [Sacca and Wiederhold, 1985] cho các CSDL phân tán.

2. Tách mảnh : Bắt đầu bằng một quan hệ và quyết định cách phân mảnh có lợi dựa trên hành vi truy xuất của các ứng dụng trên các thuộc tính. Kỹ thuật này được thảo luận lần đầu tiên cho thiết kế CSDL tập trung trong [Hoffer and Severance, 1975]. Sau đó được mở rộng cho môi trường phân tán trong [Navathe et al., 1984].

Trong phần tiếp theo, ta chỉ thảo luận về kỹ thuật tách mảnh. Trước tiên chúng ta hãy nhắc lại rằng việc nhân bản khoá của quan hệ toàn cục trong các mảnh là một đặc trưng của phân mảnh dọc cho phép tái thiết quan hệ toàn cục và duy trì tính toàn vẹn ngữ nghĩa. Vì thế phương pháp tách mảnh chỉ đề cập đến các thuộc tính không tham gia vào khóa chính.

Một chọn lựa khác đơn giản với vấn đề nhân bản các thuộc tính khóa là sử dụng một mã định bộ TID (Tuple IDentifier), là một giá trị duy nhất được hệ thống gán cho mỗi bộ của một quan hệ. Bởi vì TID được chính hệ thống duy trì, các mảnh đều tách biệt ở góc độ người sử dụng.

4.3.1. PHÂN MẢNH DỌC THEO TỤ LỰC

Bởi vì phân hoạch dọc đặt vào một mảnh các thuộc tính thường được truy xuất chung với nhau, ta cần có một giá trị "đo" để định nghĩa chính xác hơn về khái niệm "Chung với nhau". Số đo này được gọi là *tụ lực* hay *lực hút (affinity)* của các thuộc tính, nó chỉ ra mức độ liên đới giữa các thuộc tính. Bây giờ chúng ta trình bày một cách để thu được các giá trị này từ những dữ liệu nguyên thủy hơn.

Gọi $Q = (q_1, q_2, \ldots . q_q)$ là tập các vấn tin của người dùng (các ứng dụng) sẽ chạy trên quan hệ $R (A_1, A_2, \ldots . A_n)$. Thế thì với mỗi câu vấn tin q_i và mỗi thuộc tính A_j, chúng ta đưa ra một giá trị sử dụng thuộc tính (attribute usage value), ký hiệu là use (q_i, A_j) và được định nghĩa như sau :

$$\text{use}(q_i, A_j) = \begin{cases} 1 \text{ nếu thuộc tính } A_j \text{ được vấn tin } q_i \text{ tham chiếu} \\ 0 \text{ trong trường hợp ngược lại} \end{cases}$$

Các vectơ use (q_i, •) cho mỗi ứng dụng rất dễ định nghĩa nếu nhà thiết kế biết được các ứng dụng sẽ chạy trên CSDL.

Thí dụ 4.15:

Xét quan hệ PROJ của hình 4.1. Giả sử rằng các ứng dụng sau đây chạy trên quan hệ đó. Trong mỗi trường hợp chúng ta cũng đặc tả bằng SQL.

q_1 : Tìm ngân sách của một dự án, khi biết mã số dự án đó là Value

```
SELECT    BUDGET
FROM      PROJ
WHERE     PNO=Value
```

q_2 : Tìm tên và ngân sách của tất cả mọi dự án.

```
SELECT    PNAME, BUDGET   FROM   PROJ
```

q_3 : Tìm tên của các dự án được thực hiện tại một thành phố đã cho

```
SELECT    PNAME
FROM      PROJ
WHERE     LOC = Value
```

q_4 : Tìm tổng ngân sách dự toán cho thành phố

```
SELECT    SUM (BUDGET)
FROM      PROJ
WHERE     LOC = Value
```

Dựa theo bốn ứng dụng này, ta có thể định nghĩa được giá trị sử dụng thuộc tính. Để cho tiện về mặt ký pháp, ta gọi A_1 = PNO, A_2 = PNAME, A_3 = BUDGET, và A_4 = LOC. Giá trị sử dụng được định nghĩa dưới dạng ma trận (hình 4.11), trong đó mục (i, j) biểu thị use (q_i, A_j).

4.3.1.1 tụ lực của các thuộc tính

Giá trị sử dụng thuộc tính không đủ để làm cơ sở cho việc tách và phân mảnh. Điều này là do chúng không biểu thị cho độ lớn của tần số ứng dụng. Số đo lực hút (affinity) của các thuộc tính aff (A_i, A_j), biểu thị cho cầu nối (bond) giữa hai thuộc tính của một quan hệ theo cách chúng được các ứng dụng truy xuất, sẽ là một đại lượng cần thiết cho bài toán phân mảnh.

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Hình 4.11 Ma trận mẫu về giá trị sử dụng thuộc tính

Trong [1] các tác giả đã nêu một công thức để đo lực hút (affinity) của hai thuộc tính A_i, A_j .

Gọi k là số các mảnh cuả quan hệ R được phân tán . Tức $R = R_1 \cup \ldots \cup R_k$

$Q = \{ q_1, q_2, \ldots q_m \}$ là tập các câu vấn tin (tức tập các ứng dụng trên quan hệ R) . Đặt Q(A,B) là tập các ứng dụng q của Q mà use(q,A).use(q,B) = 1. Nói cách khác

$Q(A,B) = \{q \in Q : use(q,A) = use(q,B) = 1\}$. Thí dụ dựa vào ma trận ở trên (hình 4.11) ta thấy $Q(A_1,A_1) = \{q_1\}$, $Q(A_2,A_2) = \{q_2,q_3\}$, $Q(A_3,A_3) = \{q_1,q_2,q_4\}$, $Q(A_4,A_4) = \{q_3,q_4\}$, $Q(A_1,A_2)$ = rỗng, $Q(A_1,A_3)= \{q_1\}$, $Q(A_2,A_3) =\{q_2\}$, . .

Số đo lực hút (affinity đôi khi có thể gọi là ái lực hay tụ lực) giữa hai thuộc tính A_i và A_j của quan hệ R (A_1, A_2, ... , A_n) ứng với tập ứng dụng Q = (q_1, q_2, ... q_m) được định nghĩa là :

$$aff(A_i, A_j) = \sum_{q \in Q(Ai, Aj)} \sum_{l \in Rl} ref_l(q) acc_l(q)$$

Trong đó $ref_l(q)$ là số truy xuất đến các thuộc tính (A_i, A_j) cho mỗi ứng dụng q tại vị trí R_l và $acc_l(q)$ là số đo tần số truy xuất ứng dụng q đến các thuộc tính A_i, A_j tại vị trí l. Chúng ta cần lưu ý rằng trong công thức tính $aff(A_i,A_j)$ chỉ xuất hiện các ứng dụng q mà cả A_i và A_j đều sử dụng.

Kết quả của tính toán này là một ma trận đối xứng n x n, mỗi phần tử của nó là một số đo được định nghĩa ở trên. Chúng ta gọi nó là ma trận lực tụ (lực hút hoặc ái lực) thuộc tính (AA) (attribute affinity matrix).

Thí dụ 4.16:

Chúng ta hãy tiếp tục với trường hợp đã xem xét trong thí dụ 4.15. Để cho đơn giản, chúng ta hãy giả sử rằng $ref_l(q) = 1$ cho tất cả q và S_l và số mảnh sẽ là 3. Nếu tần số ứng dụng cho mọi cặp thuộc tính là

$acc_1(q_1) = 15$, $acc_2(q_1) = 20$, $acc_3(q_1) = 10$

$acc_1(q_2) = 5$, $acc_2(q_2) = 0$, $acc_3(q_2) = 0$

$acc_1(q_3) = 25$, $acc_2(q_3) = 25$, $acc_3(q_3) = 25$

$acc_1(q_4) = 3 \quad acc_2(q_4) = 0 \quad acc_3(q_1) = 0.$

Số đo lực tụ giữa các thuộc tính A_1 và A_3 là:

$$aff(A_1, A_3) = \sum_{k}^{4} \sum_{l=1}^{3} acc_l(q_k) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

Bởi vì ứng dụng duy nhất truy xuất đến cả hai thuộc tính này là q_1, tức $Q(A_1,A_3) = \{q_1\}$ và số mảnh là 3. Vậy ma trận lực hút thuộc tính đầy đủ được trình bày trong Hình 4.12. Chú ý rằng, Ma trận AA là đối xứng và các giá trị ở đường chéo chính tuy cũng được tính nhưng chúng hoàn toàn không cần cho chúng ta.

	A_1	A_2	A_3	A_4
A_1	45	0	45	0
A_2	0	80	5	75
A_3	45	5	53	3
A_4	0	75	3	78

Hình 4.12 Ma trận lực hút thuộc tính

4.3.1.2 Thuật toán năng lượng nối BEA (Bond Energy Algorithm)

Đến đây ta có thể phân R làm các mảnh của các thuộc tính dựa vào sự liên đới (lực hút) giữa các thuộc tính, thí dụ tụ lực của A_1, A_3 là 45, của A_2, A_4= là 75, còn của A_1,A_2 là 0, của A_3,A_4 là 3. . . Tuy nhiên, phương pháp tuyến tính sử dụng trực tiếp từ ma trận này ít được mọi người quan tâm và sử dụng. Sau đây chúng ta xét một phương pháp gọi là phương pháp dùng thuật

toán năng lượng nối BEA (Bond Energy Algorithm) của Hoffer and Severance, 1975 và Navathe et al., 1984.

1- Nó được thiết kế đặc biệt để xác định các nhóm gồm các mục tương tự, khác với một sắp xếp thứ tự tuyến tính của các mục.

2- Các kết quả tụ nhóm không bị ảnh hưởng bởi thứ tự đưa các mục vào thuật toán.

3- Thời gian tính toán của thuật toán có thể chấp nhận được là $O(n^2)$ với n là số lượng thuộc tính.

4- Mối liên hệ qua lại giữa các nhóm thuộc tính tụ có thể xác định được.

Thuật toán năng lượng nối BEA nhận nguyên liệu là một ma trận lực hút thuộc tính (AA), hoán vị các hàng và cột rồi sinh ra một ma trận lực tụ (CA) (clustered affinity matrix). Hoán vị được thực hiện sao cho số đo lực hút chung AM (global affinity measure) là lớn nhất. Trong đó AM là đại lượng:

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j) [aff(A_i, A_{j-1}) + aff(A_i, A_{j+1}) +$$

$$aff(A_{i-1}, A_j) + aff(A_{i+1}, A_j)]$$

Với $aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0$ cho mọi i,j .

Tập các điều kiện cuối cùng đề cập đến những trường hợp một thuộc tính được đặt vào CA ở về bên trái của thuộc tính tận trái hoặc ở về bên phải của thuộc tính tận phải trong các hoán vị cột, và bên trên hàng trên cùng và bên dưới hàng cuối cùng trong các hoán vị hàng. Trong những trường hợp này, chúng ta cho 0 là giá trị lực hút aff giữa thuộc tính đang được xét và các lân cận bên trái hoặc bên phải (trên cùng hoặc dưới đáy) của nó hiện chưa có trong CA.

Hàm cực đại hóa chỉ xét những lân cận gần nhất, vì thế nó nhóm các giá trị lớn với các giá trị lớn, giá trị nhỏ với giá trị nhỏ. Vì ma trận lực hút thuộc tính AA có tính đối xứng nên hàm số vừa được xây dựng ở trên thu lại thành

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j) [aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})]$$

Chi tiết của thuật toán năng lượng nối được cho trong thuật toán 4.3. Quá trình sinh ra ma trận tụ lực (CA) được thực hiện qua ba bước

1- Khởi gán : Đặt và cố định một trong các cột của AA vào trong CA. Thí dụ cột 1 được chọn trong thuật toán này.

2- Thực hiện lặp. Lấy lần lượt một trong n - i cột còn lại (trong đó i là số cột đã được đặt vào CA) và thử đặt chúng vào i + 1 vị trí còn lại trong ma trận CA. Chọn nơi đặt sao cho ái lực chung AM lớn nhất. Lặp đến khi không còn cột nào để đặt.

3- Sắp thứ tự hàng. Một khi thứ tự cột đã được xác định, các hàng cũng cần được đặt lại để các vị trí tương đối của chúng phù hợp với các vị trí tương đối của các cột.

Thuật toán 4.3 BEA

Input : AA : Ma trận lực tụ thuộc tính ;

Output : CA : Ma trận lực tụ sau khi đã sắp xếp lại các hàng cột;

begin

(Khởi gán : Cần nhớ rằng AA là một ma trận n x n)

CA (•, 1) ← AA (•, 1)

CA (•, 2) ← AA (•, 2)

index := 3

While index <= n do(Chọn vị trí tốt nhất cho thuộc tính A_{index})
begin
for i := 1 to index - 1 do
tính cont (A_{i-1}, A_{index}, A_i);

Tính cont ($A_{index-1}$, A_{index}, $A_{index+1}$) ; (Điều kiện biên)
Loc ← nơi đặt, được cho bởi giá trị cont lớn nhất ;
for i := index downto loc do (Xáo trộn hai ma trận)
CA (•, j)← CA (•, j - 1);

CA (•, loc) ←AA (•, index) ;
index := index + 1;
end - while
- Sắp thứ tự các hàng theo thứ tự tương đối của các cột
end . (BEA)

Để hiểu rõ thuật toán chúng ta cần biết cont(*,*,*). Cần nhắc lại rằng số đo tự lực chung AM đã được định nghĩa là :

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} aff(A_i, A_j)\,[aff(A_j, A_{j-1}) + aff(A_i, A_{j+1})]$$

và có thể viết lại :

$$AM = \sum_{i=1}^{n} \sum_{j=1}^{n} [aff(A_i, A_j)\, aff(A_i, A_{j-1}) + aff(A_i, A_j) aff(A_i, A_{j+1})]$$

$$= \sum_{j=1}^{n} \left[\sum_{i=1}^{n} aff(A_i, A_j)\, aff(A_i, A_{j-1}) + \sum_{i=1}^{n} aff(A_i, A_j) aff(A_i, A_{j+1}) \right]$$

Ta định nghĩa cầu nối (bond) giữa hai thuộc tính A_x và A_y là

$$\text{bond}(A_x A_y) = \sum_{z=1}^{n} \text{aff}(A_z, A_x)\,\text{aff}(A_z, A_y)$$

Thế thì có thể viết lại AM là

$$AM = \sum_{j=1}^{n} [\text{bond}(A_j, A_{j-1}) + \text{bond}(A_j, A_{j+1})]$$

Bây giờ xét n thuộc tính sau :

$$A_1\ A_2 \ldots A_{i-1} \quad A_i A_j \quad A_{j+1} \ldots A_n$$

Với $A_1, A_2, \ldots A_{i-1}$ thuộc nhóm AM' và $A_{j+1}, \ldots A_n$ thuộc nhóm AM"

Số đo lực hút chung cho những thuộc tính này có thể được viết là

$AM_{old} = AM' + AM'' + \text{bond}(A_{i-1}, A_i) + \text{bond}(A_i, A_j) + \text{bond}(A_j, A_i) +$

$\text{bond}(\text{bond}(A_j, A_{j+1}) = \sum_{l=1}^{n} [\text{bond}(A_l, A_{l-1}) + \text{bond}(A_i, A_{l+1})] + \sum_{l=i+1}^{n} [\text{bond}$

$(A_l, A_{l-1}) + \text{bond}(A_i, A_{l+1})] + 2\,\text{bond}(A_i, A_i)$

Bây giờ xét đến việc đặt một thuộc tính mới A_k giữa các thuộc tính A_i và A_j trong ma trận lực hút tụ. Số đo lực hút chung mới có thể được viết tương tự như :

$AM_{new} = AM' + AM'' + \text{bond}(A_i, A_k) + \text{bond}(A_k, A_i) + \text{bond}(A_k, A_j) +$

$\text{bond}(A_j, A_k) = AM' + AM'' + 2\text{bond}(A_i, A_k) + 2\text{bond}(A_k, A_j)$

Vì thế đóng góp thực (net contribution) cho số đo lực hút chung khi đặt thuộc tính A_k giữa A_i và A_j là :

$\text{Cont}(A_i, A_k, A_j) = AM_{new} - AM_{old} = 2\text{bond}(A_i, A_k) + 2\text{bond}(A_k, A_j) - 2\text{bond}(A_i, A_j)$

Thí dụ 4.17:

Ta xét ma trận AA được cho trong hình 4.12 và tính toán phần đóng góp khi di chuyển thuộc tính A_4 vào giữa các thuộc tính A_1 và A_2, được cho bằng công thức :

$$\text{Cont}(A_1, A_4, A_2) = 2\text{bond}(A_1, A_4) + 2\text{bond}(A_4, A_2) - 2\text{bond}(A_1, A_2)$$

Tính mỗi số hạng, chúng ta được

$\text{bond}(A_1, A_4) = 45^*0 + 0^*75 + 45^*3 + 0^*78 = 135$

$\text{bond}(A_4, A_2) = 11865$

$\text{bond}(A_1, A_2) = 225$

Vì thế $\text{cont}(A_1, A_4, A_2) = 2^*135 + 2^*11865 - 2^*225 = 23550$

Lưu ý rằng việc tính toán cầu nối giữa hai thuộc tính cần phải nhân các phần tử của hai cột biểu diễn những thuộc tính này và lấy tổng theo hàng.

Thuật toán và thảo luận của chúng ta cho đến lúc này chỉ chú ý đến các cột của ma trận ái lực thuộc tính AA. Ta có thể lập luận tương tự và thiết kế lại thuật toán thao tác trên các hàng. Vì ma trận AA đối xứng, cả hai lối tiếp cận đều sinh ra cùng một kết quả.

Một điểm nữa cần ghi nhận về thuật toán 4.3 đó là để cải thiện hiệu quả, cột thứ hai cũng được cố định và đặt cạnh cột thứ nhất trong bước khởi gán. Điều này là chấp nhận được bởi vì theo thuật toán, A_2 có thể được đặt ở bên trái hoặc bên phải của A_1. Tuy nhiên cầu nối giữa hai thuộc tính này độc lập với vị trí tương đối của chúng đối với nhau.

Cuối cùng, cần phân tích vấn đề khi tính cont tại các đầu tận. Nếu thuộc tính A_i đang được xét tìm chỗ đặt ở bên trái thuộc tính tận trái, một trong các phương trình cầu nối được tính giữa phần tử không có ở bên trái và A_k (nghĩa là bond (A_0, A_k)). Vì thế ta cần dùng đến những điều kiện được đặt ra cho định nghĩa của số đo ái lực chung AM, trong đó CA(0,k) = 0. Ở

thái cực kia là trường hợp A_j là thuộc tính tận phải đã được đặt trong ma trận CA và chúng ta đang kiểm tra đóng góp khi đặt thuộc tính A_k vào bên phải của A_j. Trong trường hợp này bond (A_k, A_{k+1}) cần được tính. Tuy nhiên, vì chưa có thuộc tính nào được đặt ở cột k + 1 của ma trận CA, số đo ái lực này chưa được định nghĩa. nên theo điều kiện đầu tận, giá trị bond này cũng là 0.

Thí dụ 4.18:

Ta hãy xét quá trình gom tụ các thuộc tính của quan hệ PROJ và dùng ma trận lực hút thuộc tính AA của hình 4.12.

Theo bước đầu, chúng ta chép các cột 1 và 2 của ma trận AA vào ma trận CA (hình 4.13a) và bắt đầu thực hiện từ cột 3 (nghĩa là thuộc tính A_3). Có ba nơi có thể đặt cột 3 : ở bên trái của cột 1, tạo ra thứ tự (3 - 1 - 2), giữa cột 1 và 2, cho ra (1 - 3 - 2) và ở bên phải cột 2, kết quả là (1 - 2 - 3). Chú ý rằng để tính đóng góp của lối sắp xếp cuối cùng chúng ta phải tính cont(A_2, A_3, A_4) chứ không phải cont (A_1, A_2, A_3). Hơn nữa trong ngữ cảnh này, A_4 chỉ cột thứ tư trong ma trận CA, đó là vị trí rỗng (hình 4.13b), không phải là cột thuộc tính A_4 của ma trận AA. Ta tính đóng góp cho số đo ái lực chung của mỗi khả năng này.

Thứ tự (0 - 3 - 1).

cont (A_0, A_3, A_1) = 2 bond (A_0, A_3) + 2 bond (A_3, A_1) - 2 bond (A_0, A_1)

Chúng ta biết rằng

bond (A_0, A_1) = bond (A_0, A_3) = 0

bond (A_3, A_1) = 45* 48 + 5*0 + 53* 45 + 3*0 = 4410

Vì thế:

cont (A_0, A_3, A_1) = 8820.

	A_1	A_2
A_1	45	0
A_2	0	80
A_3	45	5
A_4	0	75

(a)

	A_1	A_3	A_2
A_1	45	45	0
A_2	0	5	80
A_3	45	53	5
A_4	0	3	75

(b)

	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_2	0	5	80	75
A_3	45	53	5	3
A_4	0	3	75	78

(c)

	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_3	45	53	5	3
A_2	0	5	80	75
A_4	0	3	75	78

(d)

Hình 4.13 Tính toán ma trận ái lực tụ CA

Thứ tự (1 - 3 - 2) :

cont (A_1, A_3, A_2) = 2bond (A_1, A_3) + 2bond (A_3, A_2)- 2bond (A_1, A_2)

bond (A_1, A_3) = bond (A_3, A_1) = 4410

bond (A_3, A_2) = 890

bond (A_1, A_2) = 225

Vì thế :

cont (A_1, A_3, A_2) = 10150

Thứ tự (2 - 3 - 4)

cont (A_2, A_3, A_4) = 2bond (A_2, A_3) + 2bond (A_3, A_4) - 2bond (A_2, A_4)

bond (A_1, A_4) = 890

bond (A_3, A_4) = 0

bond (A_2, A_4) = 0

Vì thế :

cont (A_2, A_3, A_2) = 1780

Bởi vì đóng góp của thứ tự (1 - 3 - 2) là lớn nhất, chúng ta đặt A_3 vào bên phải của A_1 (hình 4.13b). Tính toán tương tự cho A_4 chỉ ra rằng cần phải đặt nó vào bên phải của A_2 (hình 4.13c).

Cuối cùng các hàng được tổ chức với cùng thứ tự như các cột và kết quả được trình bày trong hình 4.13d. Trong hình 4.13d ta thấy quá trình tạo ra hai tụ : Một ở góc trên trái chứa các giá trị ái lực nhỏ, còn tụ kia ở góc dưới phải chứa các giá trị ái lực cao. Quá trình phân tụ này chỉ ra cách thức tách các thuộc tính của quan hệ PROJ. Tuy nhiên, nói chung ranh giới giữa các phần tách không hoàn toàn rõ ràng. Khi ma trận CA lớn, thường thường sẽ có nhiều tụ hơn được tạo ra và có nhiều phân hoạch để chọn hơn. Do vậy cần phải tiếp cận với bài toán một cách có hệ thống hơn. (Bạn đọc có thể tìm hiểu thuật toán phân hoạch trong [1]).

4.3.2. PHÂN MẢNH DỌC CÓ NỐI KHÔNG MẤT THÔNG TIN

4.3.2.1.Khái niệm phân mảnh dọc Có nối không mất thông tin

Ta cần nhắc lại phân rã dọc lược đồ quan hệ $R = \{A1, A_2, ...A_n\}$ là thay nó bằng một tập $\rho = \{R_1, R_2, ... R_k\}$ trong đó R_i là các tập con của R sao cho $R = R_1 \cup R_2 \cup ... \cup R_k$

Các tập R_i không nhất thiết phải tách biệt mà các R_i có thể chứa các thuộc tính khoá .

Trong phần này, chúng ta sẽ minh hoạ bằng các lược đồ quan hệ của các nhà cung cấp hàng cho siêu thị (xem chương 2, 3), thí dụ suplies (SNAME, ITEM, PRICE, ADDR) chứa các thông tin về các nhà cung cấp (SUPPLIERS), với các thuộc tính là tên, hàng hoá, giá, địa chỉ tương ứng của nhà cung cấp. Để cho gọn ta dùng các chữ viết tắt cho các thuộc tính S (SNAME), A (ADDR), I (ITEM) và P(PRICE). Trước khi tiến hành các thuật toán phân mảnh lưu ý rằng lược đồ trên không tối ưu do dư thừa dữ liệu mỗi khi nhập thêm một mặt hàng mới chúng ta phải nhập địa chỉ của nhà cung cấp, mặc dù địa chỉ của mỗi nhà cung cấp đáng lẽ chỉ phải nhập một lần. Nếu lược đồ trên được chia làm hai lược đồ SUP1(S,A) và SUP2(S,I,P) thì ta được một CSDL tốt hơn, các nhược điểm trên đã biến mất. Chúng ta dễ dàng chứng minh được rằng SUP1và SUP2 là những quan hệ thuộc dạng chuẩn BCNF.

Vậy có những phân rã quan hệ thành các dạng chuẩn sẽ tối ưu cho phần thiết kế CSDL tập trung và phân tán.

Ta nhắc lại trong lược đồ SUP(S,A,I,P) với các phụ thuộc hàm chúng ta giả sử S → A, SI →P. Trên đây chúng ta vừa đã thấy rằng khi thay lược đồ SUP bởi hai lược đồ SUP1 và SUP2 có thể loại bỏ một số vấn đề rắc rối như dư thừa dữ liệu.Vấn đề đặt ra ở đây là việc phân rã SUP thành SUP1 và SUP2 có đảm bảo cho chúng ta tái thiết được SUP hay không? quá trình phân rã có làm mất thông tin không?

Giả sử r là một quan hệ trên tập thuộc tính{ S,A,I,P}. Nếu CSDL sử dụng SUP1 và SUP2 thay cho SUP(S,A,I,P) khi đó quan hệ r được tách trên hai lược đồ { S,A} và {S,I,P}, nghĩa là r.SA và r.SIP. Vấn đề được đặt ra là quan hệ r có thể tái thiết được từ r.SA và r.SIP bằng phép nối tự nhiên không? Sau đây ta sẽ trình bày một số vấn đề liên quan đến khẳng định là r = r.SA |><| r.SIP, lý do là nếu gọi s = r.SA |><| r.SIP mà s ≠ r thì r không tái thiết được .

Bây giờ chúng ta sẽ nêu định nghĩa phân rã có nối không mất.

Cho sơ đồ quan hệ W = < R, F >. R là lược đồ quan hệ được phân rã thành các lược đồ con $R_1, R_2, \ldots, R_k$

Tức là ta có phân rã $\rho = \{R_1, R_2, \ldots R_k\}$.

Ta nói phân rã ρ có nối không mất thông tin (gọi tắt là có nối không mất) ứng với F, nếu mọi quan hệ r trên R thoả mãn F ta luôn có:

$$r = r.R_1 \ |><| \ r.R_2 \ \ldots \ |><| \ r.R_k$$

nghĩa là r là nối tự nhiên của các hình chiếu của nó lên các R_i. Nói cách khác nếu gọi $m_\rho(r) = r.R_1 \ |><| \ r.R_2 \ \ldots \ |><| \ r.R_k$ thì phân rã có nối không mất ứng với F nếu mọi r thoả F thì $r = m_\rho(r)$.

Bổ đề 4.1:

Gọi R là một lược đồ quan hệ, $\rho = (R_1, R_2 ... , R_k)$ là một phân rã của R và r là một quan hệ trên R. Gọi $r_i = r.R_i$. Khi đó ta dễ dàng chứng minh được các hệ thức sau:

a. $r \subseteq m_\rho(r)$.

b. Nếu $s = m_\rho(r)$ thì $s.R_i = r_i$.

c. $m_\rho(m_\rho(r)) = m_\rho(r)$.

4.3.2.2 Kiểm tra tính nối không mất

Chúng ta thấy rằng có thể khẳng định một phân rã có tính chất nối không mất ứng với một tập các phụ thuộc hàm F.

Thuật toán 4.4a- thuật toán Chase:

Kiểm tra tính chất nối không mất của một phân rã

Input: $W = < R, F >$: Với $R = \{A_1 ... A_n\}$ là lược đồ quan hệ.
F là tập PTH.
$\rho = (R_1, ... R_k)$ là một phan rã của R;

Qutput: Một khẳng định ρ có phải là một phân rã có nối không mất hay không;

Phương pháp: chúng ta xây dựng một bảng gồm n cột và k hàng: cột j

tương ứng với thuộc tính A_j, hàng i tương ứng với lược đồ quan hệ R_i. Ở vị trí hàng i và cột j, chúng ta đặt ký hiệu a_j nếu A_j thuộc R_i; Ngược lại, ta đặt ký hiệu b_{ij} vào vị trí đó.

Coi bảng là một quan hệ r trên R và ép cho F thoả bảng (quan hệ r)bằng cách:

Xét lặp nhiều lần mỗi phụ thuộc $X \rightarrow Y$ trong F cho đến khi không xét được nữa, tức toàn bảng r đã thoả F. Mỗi lần xét $X \rightarrow Y$, chúng ta tìm những hàng giống nhau ở tất cả các cột của các thuộc tính trong X, nếu thấy hai hàng như thế, hãy làm cho các ký hiệu của hai hàng này bằng nhau ở các thuộc tính của Y.

Khi làm cho hai ký hiệu bằng nhau trong Y, ta lưu ý nếu một trong hai ký hiệu là a_j thì cho ký hiệu kia trở thành a_j. Nếu hai ký hiệu là b thì có thể cho chúng trở thành b một cách tuỳ ý. Điều quan trọng cần phải nhớ là chúng ta phải sửa các a,b nhiều lần cho đến khi cả bảng thoả tập F. Nếu sau khi sửa đổi (hoặc sau vài sửa đổi) chúng ta thu được một hàng toàn a thì phân rã có nối không mất, ngược lại có nối mất.

Thí dụ 4.19:

Xét lại phân rã R= {S,A,I,P} thành R_1= {S,A} và R_2 = {S,I,P} , các phụ thuộc là:

F = { $S \rightarrow A$ và $SI \rightarrow P$}.

Bảng khởi đầu là:

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

Bởi vì S → A và hai hàng giống nhau ở cột S, chúng ta có thể làm cho các ký hiệu của cột A bằng nhau, cho b_{22} thành a_2, bảng kết quả là:

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

Bởi vì hàng thứ hai đều là a, đây là nối không mất.

Xét một thí dụ khác phức tạp hơn, gọi R = ABCDE, R_1 = AD, R_2 = AB, R_3 = BE, R_4 = CDE và R_5 = AE. Giả sử có tập các phục thuộc hàm:

$$A \rightarrow C \qquad DE \rightarrow C$$
$$B \rightarrow C \qquad CE \rightarrow A$$
$$C \rightarrow D$$

Bảng khởi đầu được trình bày trong Hình 4.14 (a). Chúng ta có thể áp dụng A → C để có các ký hiệu b_{13}, b_{23} và b_{53} bằng nhau. Sau đó chúng ta dùng B → C để có các ký hiệu này bằng với b_{33}; kết quả được trình bày trong hình 4.14 (b), trong đó b_{13} được chọn làm ký hiệu đại diện. Bây giờ chúng ta dùng C → D để cho các ký hiệu a_4, b_{24}, b_{34} và b_{54} bằng nhau; ký hiệu kết quả phải là a_4. Từ DE → C cho phép chúng ta cho b_{13} bằng với a_3,

và CE → A cho b_{31} và b_{41} bằng với a_1. Kết quả được trình bày trong hình 4.14(c). Bởi vì hàng giữa đều là a nên phân rã này có tính chất nối không mất .

Định lý 4.1:

Thuật toán 4.4a xác định chính xác phân rã đã cho có hay không có tính chất nối không mất .

Chứng minh:

Giả sử bảng cuối cùng sinh ra từ Thuật toán không có hàng toàn a. Chúng ta có thể xem bảng này như một quan hệ r của lược đồ R; các hàng là các bộ và a_j, b_{ij} là các ký hiệu riêng biệt, mỗi ký hiệu được lấy từ miền trị của các thuộc tính A_j. Quan hệ r thoả phụ thuộc F bởi vì Thuật toán 4.4a sửa đổi bảng mỗi khi tìm thấy một vi phạm.

Chúng ta sẽ chứng minh rằng $r \neq m_\rho(r)$. Thật vậy

Rõ ràng r không chứa bộ $a_1a_2...a_n$. Nhưng đối với mỗi lược đồ R_i có một bộ μ_i trong $r.R_i$, chứa toàn a. Vì vậy nối của các $r.R_i$ (tức $m_\rho(r)$) chứa bộ có toàn a, vì rằng bộ $t = (a_1,a_2,a_3, \ldots, a_n)$ có $t.R_i$ thuộc R_i. Vậy $r \neq m_\rho(r)$ và chúng ta kết luận rằng nếu bảng cuối cùng của Thuật toán 4.4a không có hàng nào chứa toàn a thì phân rã ρ không có tính chất nối không mất. Vì có một quan hệ r trên R thoả mãn F mà $m_\rho(r) \neq r$.

Ngược lại, giả sử rằng cuối cùng có một hàng toàn a. Phần chứng minh này các bạn có thể tham khảo trong [2].

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

(a)

A	B	C	D	E
a_1	b_{12}	b_{13}	a_4	b_{15}
a_1	a_2	b_{13}	b_{24}	b_{25}
b_{31}	a_2	b_{33}	b_{34}	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{54}	b_{13}	b_{54}	a_5

(b)

A	B	C	D	E
a_1	b_{12}	a_3	a_4	b_{15}
a_1	a_2	a_3	a_4	b_{25}
a_1	a_2	a_3	a_4	a_5
a_1	b_{42}	a_3	a_4	a_5
a_1	b_{52}	a_3	a_4	a_5

(c)

Hình 4.14 Minh hoạ Thuật toán 4.4a

Thuật toán 4.4a có thể áp dụng cho các phân rã với số lượng các mảnh bất kỳ. Tuy nhiên, đối với các phân rã thành chỉ hai lược đồ, chúng ta có một phép kiểm tra đơn giản hơn, đó là nội dung của định lý sau đây.

Định lý 4.2:

Nếu $\rho = (R_1, R_2)$ là một phân rã của R, và F là tập hợp các phụ thuộc hàm, thì ρ có nối không mất ứng với F nếu và chỉ nếu :

$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ hoặc $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$.

Chú ý rằng những phụ thuộc này không nhất thiết thuộc tập F, chỉ cần chúng thuộc F^+.

	$R_1 \cap R_2$	$R_1 - R_2$	$R_2 - R_1$
Hàng cho R_1	aa... a	aa...a	bb...b
Hàng cho R_2	aa ...a	bb... b	aa... a

Hình 4.15 Một bảng hai hàng tổng quát

Chứng minh: Bảng khởi đầu được sử dụng trong Thuật toán 4.4a được trình bày trong Hình 4.15 ta đã bỏ những chỉ số trên a và b vì không quan trọng. Giả sử rằng $(R_1 \cap R_2) \rightarrow R_1 - R_2$. Khi đó theo Thuật toán 4.4a chúng ta sửa các giá trị trong bảng theo phụ thuộc hàm này, kết quả ta được hàng thứ hai toàn a, nên phân rã có nối không mất.

Tương tự nếu $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$, sau khi sửa bảng cho phù hợp với phụ thuộc hàm này chúng ta có hàng thứ nhất toàn a và cũng có ngay kết luận phân rã có nối không mất.

Thí dụ 4.20:

Giả sử R = ABC và F = {A → B}. Khi đó phân rã của R thành AB và AC có nối không mất bởi vì AB ∩ AC = A, AB - AC = B, và A → B đúng. Tuy nhiên, nếu chúng ta phân rã R thành R_1 = AB và R_2 = BC, chúng ta có $R_1 \cap R_2$ = B, nhưng cả R_1 - R_2 lẫn R_2 - R_1 đều không phụ thuộc vào B. Vì vậy phân rã AB và BC không có tính chất nối không mất ứng với F = {A →B}.

Chúng ta cần lưu ý rằng thuật toán 4.4a của Chase cho ta một phương pháp kiểm tra một phân rã có nối không mất hay không? thuật toán không cho ta biết cách phân rã R thành các lược đồ con có nối không mất.

Bổ đề 4.2:

Giả sử R là lược đồ quan hệ thoả mãn các phụ thuộc hàm F.

Gọi $\rho = (R_1, \ldots, R_k)$ là một phân rã R có nối không mất ứng với F, và gọi

$\sigma = (S_1, S_2)$ là một phân rã có nối không mất của R_1 ứng với $\pi_{R1}(F)$. Khi đó phân rã của R thành $(S_1, S_2, R_2, \ldots, R_k)$ cũng có nối không mất ứng với F.

Chứng minh:

Đặt $r_i = r.R_i$; với i = 1,2,...k vì phân rã ρ có nối không mất nên ta có

$r = r_1 |><| r_2 |><| r_3 \ldots |><| r_k$ (*)

và vì phân rã $\sigma = (S_1, S_2)$ có nối không mất trên R_1 nên nếu đặt $r_{11} = r_1.S_1$ và $r_{12} = r_1.S_2$ thì ta có $r_1 = r_{11} |><| r_{12}$. Thay $r_1 = r_{11} |><| r_{12}$ vào (*) và vì phép nối có tính kết hợp nên ta có $r = r_{11} |><| r_{12} |><| r_2 \ldots |><| r_k$.

4.3.2.3 Thuật toán phân mảnh có nối không mất thông tin

Thuật toán 4.4b. *Phân rã W thành các W_i có nối không mất*

Input $W = < R, F >$;

Output $W_1, W_2, \ldots W_k$ có nối không mất.

Phương pháp:

Phân rã W thành từng cặp có nối không mất. Đầu tiên phân rã W thành W_1 và phần còn lại W_{c1}. Coi phần còn lại W_{c1} là W. Sau đó lại phân rã W thành W_2 và W_{c1} còn lại có nối không mất. Quá trình tiếp tục cho đến khi không phân rã được nữa, thêm W còn lại cuối cùng vào phân rã. Theo định lý 4.2 và bổ đề 4.2 ta có phân rã có nối không mất. Chẳng hạn, giả sử F = { $X \rightarrow Y$ }là tập các phụ thuộc hàm.

Đặt $W_1 = < XY, X \rightarrow Y >$; $R_{c1} = R - Y$; $F_{c1} = \pi_{Rc1}(F)$; $W_{c1} = < R_{c1}, F_{c1} >$.Theo định lý 4.2 khi đó $R_1 = XY$, $R_{c1} = R-Y$ có nối không mất ứng với F. Coi

$< R_{c1}, F_{c1} >$ là W tiếp tục phân rã cho đến khi không phân rã được nữa thêm W cuối cùng vào phân rã.

Một cách chính xác hơn ta mô tả thuật toán như sau:

```
Begin
k:= 1;
R:= R;          ** R là lược đồ ban đầu**
```

F:= { X→ Y} * F là tập phụ thuộc hàm*;

Repeat

W_k := < R_k, F_k> := < XY, X→ Y>;

R := R- Y; F := $\pi_R(F)$; W còn lại := < R, F >;

k:= k+1;

Until $\pi_R(F) = Ø$;

W_k = < R, Ø >

** phân rã $\rho = (R_1, ..., R_k)$ có nối không mất ứng với F **

End.

Thí dụ 4.21:

Xét W= < {G, C, H, P,S,T}, {C →T, HP→C, HT→P, CS→G, HS→P}>.

Khi đó ta có phân rã theo thuật toán 4.4b:

W_1 = <R_1, F_1> = < CT, C→ T>; W_{c1} = < CGHPS, { HP→C, CS→G, HS→P}>.

W_2 = < R_2, F_2> = < HPC, HP→C>; W_{c1} = < GHPS, HS→P >

W_3 = < R_3, F_3> =< HSP, HS→P>; W_{c1} = < GHS,Ø>

W_4 = <R_4, F_4> = < GHS, Ø>. Khi đó $\rho = (R_1, ..., R_4)$ có nối không mất ứng với F.

Kiểm tra lại tính nối không mất bằng thuật toán Chase:

Để cho tiện trong các bảng kiểm tra ta thêm cột các R_i.

Bảng khởi đầu (các ô rỗng chứa các b_{ij})

	C	G	H	P	S	T
R_1	a					a
R_2	a		a	a		
R_3			a	a	a	
R_4		a	a		a	

Bảng sau khi sửa cho phù hợp với các phụ thuộc hàm

	C	G	H	P	S	T
R_1	a					a
R_2	a		a	a		a
R_3	a		a	a	a	a
R_4	a	a	a	a	a	a

Bảng có một dòng toàn a nên phân rã có nối không mất.

Chú ý với một sơ đồ quan hệ W = < R, F>, ta có thể có nhiều cách phân rã R thành các lược đồ con có nối không mất ứng với F. Thí dụ xét sơ đồ quan hệ sau:

W= < {G, C, H, P,S,T}, {C →T, HP→C, HT→P, CS→G, HS→P}>.

Khi đó ta có thể phân rã R thành các lược đồ có nối không mất:

R_1 = {C,T};

R_2 = {H,S,P};

R_3 = {C,S,G};

R_4 = {C,S,H};

Bảng khởi đầu để kiểm tra phép nối không mất:

	C	T	H	S	P	G
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	b_{33}	a_4	b_{35}	a_6
R_4	a_1	b_{42}	a_3	a_4	b_{45}	b_{46}

Bảng sau khi ép để quan hệ r (bảng trên) thoả các PTH trong W:

	C	T	H	S	P	G
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	a_2	b_{33}	a_4	b_{35}	a_6
R_4	a_1	a_2	a_3	a_4	a_5	a_6

Vậy bảng có một hàng toàn a nên phép tách có nối không mất.

Hoặc ta có thể phân rã R thành:

R_1= {C,T};

R_2= {H, P, C};

R_3= {H, S, P};

R_4= {H, S, G};

Và chúng ta dễ dàng kiểm tra lại bằng các bảng sau (vì trong thuật toán Chase ta chỉ chỉnh sửa các giá trị b nên trong bảng khởi đầu ta chỉ cần liệt kê a còn các ô chứa các b_{ij} để trống):

Bảng r khởi đầu để kiểm tra phép nối không mất:

	C	T	H	S	P	G
R_1	a	a				
R_2	a		a		a	
R_3			a	a	a	
R_4			a	a		a

Bảng sau khi ép để quan hệ r (bảng trên) thoả các PTH trong W:

	C	T	H	S	P	G
R_1	a	a				
R_2	a	a	a		a	
R_3	a	a	a	a	a	
R_4	a	a	a	a	a	a

Vậy bảng có một hàng toàn a nên phép tách có nối không mất.

Hoặc R_1 = {C,T}; R_2 = {C, H, S, P, G} hay R_1 = {H,P,C}, R_2= {H,P,T,S,G} vân vân là những phép tách có nối không mất của lược đồ R ứng với tập phụ thuộc hàm F.

Kết luận:

a. *Phân rã W thành các sơ đồ con có nối không mất không duy nhất.*

b. *Mọi sơ đồ quan hệ W= <R, F> đều có thể phân rã R thành các lược đồ có nối không mất.*

Lưu ý lại rằng thuật toán Chase là tìm một dòng toàn a, nên các a_j mới là những đại lượng quan trọng, vì thế trong bảng ta có thể bỏ qua các b_{ij} . Hơn nữa ta có thể bỏ qua chỉ số j của a_j vì j đã được xác định qua cột tương ứng trong bảng. Một lưu ý nữa là thuật toán chỉ chọn sửa cùng lắm các b_{ij} nên trong quá trình chỉnh sửa ta chỉ việc thêm a vào các ô trống nếu có. Thí dụ để kiểm tra phép tách có nối không mất của R_1={C,T}, R_2= {C,G,H,P,S}ở trên ta lập hai bảng như sau:

Bảng khởi đầu:

	C	G	H	P	S	T
R_1	a					a
R_2	a	a	a	a	a	

Bảng sau khi sửa cho bảng khởi đầu thoả một phụ thuộc hàm C→ T:

	C	G	H	P	S	T
R_1	a					a
R_2	a	a	a	a	a	a

4.3.3. PHÂN MẢNH DỌC BẢO TOÀN PHỤ THUỘC

4.3.3.1. Định nghĩa phân rã dọc bảo toàn phụ thuộc

Chúng ta đã hiểu được rằng một phân rã cần phải có đặc tính nối không mất thì nó cho phép khôi phục lại quan hệ gốc bằng nối các chiếu của nó.

Tuy nhiên trong các phân tích ở trên chúng ta ít quan tâm đến tập phụ thuộc hàm F sau khi phân rã R. Một đặc tính quan trọng khác của phân rã $\rho = (R_1, ..., R_k)$ của lược đồ quan hệ R là có thể suy ra tập phụ thuộc F từ những hình chiếu của F trên các R_i.

Về trực quan, hình chiếu của F trên tập các thuộc tính Z, ký hiệu là $\pi_Z(F)$

(hoặc $F_{.Z}$), là tập các phụ thuộc $X \rightarrow Y$ thuộc F^+ sao cho $XY \subseteq Z$ (chú ý rằng $X \rightarrow Y$ không nhất thiết thuộc F; chỉ cần thuộc F^+).

Ta nói phân rã $\rho = (R_1, R_2, ...R_k)$ của lược đồ R có *bảo toàn phụ thuộc (bảo toàn tập phụ thuộc hàm F)* nếu hợp của tất cả phụ thuộc trong các $\pi_{R_i}(F)$, với i = 1, 2, ... k xác định được tất cả các phụ thuộc trong F.

Lý do ρ cần bảo toàn tập F là vì các phụ thuộc trong F là các ràng buộc toàn vẹn cho lược đồ quan hệ R. Tập F đóng vai trò quan trọng và bản chất của sơ đồ quan hệ. Nếu các phụ thuộc hình chiếu không suy ra được F thì khi biểu diễn R bằng $\rho = (R_1 ... R_k)$, chúng ta có thể thấy rằng giá trị hiện hành của các R_i biểu diễn một lược đồ quan hệ R không thoả F, ngay cả khi ρ có nối không mất ứng với F. Khi đó mỗi thao tác cập nhật trên các R_i sẽ phải kiểm tra lại các ràng buộc và tính toàn vẹn ngữ nghĩa.

Thí dụ 4.22:

Chúng ta xét lược đồ R có các thuộc tính là C, S và Z. Giả sử có các phụ thuộc là CS → Z và Z → C. Phân rã của lược đồ quan hệ R(C,S,Z) thành R_1(S,Z) và R_2(C,Z) có nối không mất, bởi vì:

$(SZ \cap CZ) \rightarrow (CZ - SZ)$

Tuy nhiên, chiếu của F = {CS → Z, Z → C } lên R_1 là rỗng và lên R_2 là Z → C. Vậy hợp của các chiếu của F lên các R_i không suy ra được CS → Z , vì thế phân rã này không bảo toàn các phụ thuộc mặc dù nó có nối không mất.

4.3.3.2 Kiểm tra tính bảo toàn phụ thuộc

Về nguyên tắc, chúng ta có thể kiểm tra xem một phân rã $\rho = (R_1 \ldots R_k)$ có bảo toàn tập phụ thuộc F hay không. Chúng ta chỉ cần tính F^+ rồi chiếu nó lên các R_i. Sau đó lấy hợp của các tập phụ thuộc kết quả kiểm tra xem tập này có tương đương với F hay không.

Nhắc lại hai tập phụ thuộc hàm F và G xác định trên cùng lược đồ R được gọi là tương đương nếu và chỉ nếu $F^+ = G^+$.

Chúng ta dễ dàng thấy rằng F tương đương với G nếu và chỉ nếu

$$F \subseteq G^+ \,\&\, G \subseteq F^+.$$

Trong chương hai ta đã thấy, việc tính trực tiếp tập F^+ là một công việc hết sức khó khăn. Nhưng có một cách để kiểm tra tính bảo toàn này mà không cần phải tính F^+. Đó là thuật toán sau:

Thuật toán 4.5a: *Kiểm tra tính bảo toàn các phụ thuộc*

Input: Một phân rã $\rho = (R_1, ... R_k)$ và tập các phụ thuộc F.

$G = \cup \pi_{R_i}(F)$.

Output: Một khẳng định là ρ có bảo toàn F hay không.

Phương pháp:

Chúng ta gọi G là hợp của các $\pi_{R_i}(F)$. Chúng ta xét G có tương đương với F hay không? Vì theo định nghĩa của tập G ta đã có $G \subseteq F^+$ nên ta chỉ cần kiểm tra xem $F \subseteq G^+$ không? tức mỗi phụ thuộc $X \rightarrow Y$ của F có thuộc G^+ không?

Theo các tính chất của phụ hàm ta có $X \rightarrow Y \in G^+ \Leftrightarrow Y \subseteq X^+_G$ .Vậy ta chỉ cần xác định xem X^+_G được tính ứng với G, có chứa Y hay không? Nếu tất cả các phụ thuộc hàm $X \rightarrow Y$ của F mà X^+_G chứa Y thì $F \subseteq G^+$ và phân rã có bảo toàn phụ thuộc, ngược lại phân rã không bảo toàn phụ thuộc.

Thí dụ 4.23:

Xét tập các thuộc tính ABCD với phân rã thành ba tập {AB, BC, CD} và tập các phụ thuộc F ={A → B, B → C, C →D, D →A }. Nghĩa là trong F^+, mỗi thuộc tính đều xác định tất cả các thuộc tính còn lại. Lúc đầu chúng ta có thể lầm tưởng rằng khi chiếu F lên AB, BC, và CD, chúng ta không nhận được phụ thuộc D → A, nhưng điều đó không đúng. Khi chiếu F, thực sự chúng ta đã chiếu F^+ trên các lược đồ con, vì thế chiếu F trên AB chúng ta không chỉ thu được A → B mà cả B → A. Vậy $\pi_{AB}(F)$= { A → B, B → A}. Tương tự ta có $\pi_{BC}(F)$= { B → C, C → B}, $\pi_{CD}(F)$= { C → D, D → C} và rõ ràng

$G^+ = \pi_{AB}(F) \cup \pi_{BC}(F) \cup \pi_{CD}(F) = \{ A \rightarrow B, B \rightarrow A , B \rightarrow C, C \rightarrow B, C \rightarrow D, D \rightarrow C\} = F^+$

Vậy chúng ta kết luận rằng phân rã này bảo toàn tập phụ thuộc F.

4.3.3.3 Thuật toán phân rã dọc bảo toàn phụ thuộc

Thuật toán 4.5b: *Phân rã W thành các W_i có bảo toàn phụ thuộc*

Input W = < R, F >;

Output W_1, W_2, ... có bảo toàn phụ thuộc;

Phương pháp:

Giả sử F ={ X→ Y }. Khi đó ta đặt vào mỗi Wi = < XY, X→ Y>.

Nếu có tập các thuộc tính không xuất hiện trong F ta đặt một sơ đồ chứa các thuộc tính đó với tập phụ thuộc rỗng.

Hiển nhiên phép phân rã theo thuật toán 4.5b bảo toàn phụ thuộc vì tập các phụ thuộc trong F bảo toàn trong các W_i và $R = R_1 \cup R_2 \cup R_3 \ldots \cup R_k$.

Thí dụ 4.24:

Xét W = < {A, B, C, D, E, G, H, I, M, N}, {AB → CD, DE→ GB, HA→ EG, EG→ D, G→ I}>. Khi đó ta có phân rã W thành các Wi con:

W_1 = < ABCD, AB → CD >;

W_2 = < DEGB, DE → GB > ;

W_3 = < HAEG, HA → EG >;

W_4 = < EGD, EG → D >;

W_5 = < GI, G → I >;

W_6 = < MN, Ø >. Rõ ràng phân rã bảo toàn phụ thuộc vì $\cup \pi_{R_i}(F) = F$

Chú ý: có nhiều cách phân rã có bảo toàn phụ thuộc của một sơ đồ quan hệ.

Thật vậy với sơ đồ quan hệ trong thí dụ 4.23 ta có thể phân rã R thành hai lược đồ con R_1= {A,B,C} và R_2 = {C,D}. Khi đó ta có:

$\pi_{ABC}(F)$= { A → B, B → A, B → C, C → B, C → A, A → C}.

$\pi_{CD}(F)$= { C → D, D → C, } và rõ ràng $G^+ = F^+$

Vậy phân rã này bảo toàn tập phụ thuộc F.

Kết luận:

a. *Phân rã có bảo toàn phụ thuộc là không duy nhất.*

b. *Với mọi sơ đồ quan hệ W = <R, F> luôn tồn tại phân rã có bảo toàn phụ thuộc.*

4.3.4. PHÂN MẢNH DỌC CÓ NỐI KHÔNG MẤT THÔNG TIN VÀ BẢO TOÀN PHỤ THUỘC

4.3.4.1. Định nghĩa phân mảnh dọc có nối không mất thông tin và bảo toàn phụ thuộc

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

Phép phân rã p = (R1,R2, . .., Rk) của lược đồ R ứng với F được gọi là *có nối không mất bảo toàn phụ thuộc* nếu nó bảo toàn phụ thuộc và có nối không mất.

Để kiểm tra phép phân rã p = (R_1,R_2, . .., R_k) có bảo toàn phụ thuộc và có nối không mất, ta phải lần lượt kiểm tra hai điều kiện:

(1) Tính bảo toàn phụ thuộc.

(2) Tính nối không mất.

Thí dụ xét lược đồ quan hệ R = {A,B,C,D} và F = { A → B, B → C, C → D,

D → A}, và phân rã p = (AB,BC,CD).

Rõ ràng phép phân rã này bảo toàn phụ thuộc. Để kiểm tra tính nối không mất ta dùng thuật toán 4.4a:

Bảng khởi đầu

	A	B	C	D
R_1	a_1	a_2		
R_2		a_2	a_3	
R_3			a_3	a_4

Bảng sau khi cho thoả nãn B→C và C→D

	A	B	C	D
R_1	a_1	a_2	a_3	a_4
R_2		a_2	a_3	
R_3			a_3	a_4

Ta có một dòng toàn a nên phân rã có nối không mất. Kết luận phân rã có nối không mất và bảo toàn phụ thuộc.

Sau đây ta sẽ xét tiếp thuật toán kiểm tra phân rã có nối không mất và bảo toàn phụ thuộc.

4.3.4.2 kiểm tra phân rã dọc có nối không mất và bảo toàn phụ thuộc

Thuật toán 4. 6a : *Kiểm tra phân rã W có bảo toàn phụ thuộc và nối không mất*

Input :W = < R, F >;

p = (R_1, R_2, ... R_k) là một phân rã của R;

Output: Khẳng định p có nối không mất và bảo toàn phụ thuộc?

Phương pháp:

Lần lượt kiểm tra từng tính chất của phân rã:

Tính nối không mất kiểm tra bằng thuật toán Chase. Tính bảo toàn phụ thuộc kiểm tra bằng thuật toán 4.5a.

Chú ý: có nhiều cách để phân rã một sơ đồ quan hệ thành các sơ đồ con có nối không mất và bảo toàn phụ thuộc .

Thí dụ hai phép phân rã lược đồ R ={A,B,C,D} thành:

ρ1= (AB,BC,CD) và ρ1 = (ABC,CD) cuả sơ đồ quan hệ trong thí dụ 4.23 đều có nối không mất và bảo toàn phụ thuộc.

Kết luận:

Với mỗi W có thể có nhiều phân rã khác nhau bảo toàn phụ thuộc và có nối không mất của W.

4.3.4.3 Thuật toán phân rã dọc có nối không mất và bảo toàn phụ thuộc

Thuật toán 4.6b: *Phân rã W thành các W_i bảo toàn phụ thuộc và có nối không mất*

Input : W = < R, F >; K là khoá của W.

Output: W_1, W_2, ... W_k có nối không mất và bảo toàn phụ thuộc.

Phương pháp:

Để đảm bảo tính bảo toàn phụ thuộc ta lấy tất cả các phụ thuộc hàm của F :

X→ Y cho vào các sơ đồ con Wi = < XY, X→ Y>; vậy Ri = XY;

Nếu có các thuộc tính không thuộc F ta lập sơ đồ con gồm các thuộc tính đó với tập phụ thuộc rỗng.

Để đảm bảo tính nối không mất ta đưa vào phân rã một sơ đồ con < K, $\pi_K(F)$>

với tập thuộc tính là khoá K, tập phụ thuộc là chiếu của F lên K.

Về sau ta sẽ chứng minh phân rã như thế có nối không mất.

Thí dụ 4.25:

Xét W = < {A, B, C, D, E, G, H, I, M, N}, {AB → CD, DE→ GB, HA→ EG, EG→ D, G→ I}>. Khoá của W là K = {A,H,M,N}

Khi đó ta có phân rã W thành các Wi con bảo toàn phụ thuộc:

W_1 = < ABCD, AB → CD >;

W_2 = < DEGB, DE → GB > ;

W_3 = < HAEG, HA → EG >;

$W_4 = < EGD, EG \rightarrow D >$;

$W_5 = < GI, G \rightarrow I >$;

Để đảm bảo tính nối không mất ta thêm vào $W_6 = < AHMN, Ø >$

Kiểm tra lại : tính bảo toàn phụ thuộc hiển nhiên, tính nối không mất bằng thuật toán Chase.

Bảng khởi đầu

	A	B	C	D	E	G	H	I	M	N
R_1	a	a	a	a						
R_2		a		a	a	a				
R_3	a				a	a	a			
R_4				a	a	a				
R_5						a		a		
R_6	a						a		a	a

Bảng sau khi sửa cho hợp với các phụ thuộc hàm trong F

	A	B	C	D	E	G	H	I	M	N
R_1	a	a	a	a						
R_2		a		a	a	a				
R_3	a				a	a	a			
R_4				a	a	a				
R_5						a		a		
R_6	a	a	a	a	a	a	a	a	a	a

Bảng có một hàng (hàng chứa các thuộc tính khoá) toàn a nên phân rã có nối không mất.

4.3.5. PHÂN MẢNH DỌC THÀNH CÁC BCNF CÓ NỐI KHÔNG MẤT THÔNG TIN VÀ BẢO TOÀN PHỤ THUỘC

Trong các phần trên chúng ta đã có các thuật toán và các phép kiểm tra tương ứng cho các phân rã W thành :

(1) Nối không mất.

(2) Bảo toàn phụ thuộc.

(3) Bảo toàn phụ thuộc và nối không mất.

Bây giờ chúng ta lần lượt xét các phân rã W thành các sơ đồ con BCNF có kèm theo các tính chất (1) hoặc (2) hoặc (3).

Chúng ta cảm nhận rằng mọi lược đồ quan hệ đều có một phân rã có nối không mất thành dạng chuẩn Boyce – Codd (BCNF).

Mọi lược đồ cũng có thể phân rã thành dạng BCNF bảo toàn phụ thuộc.

Một vấn đề đặt ra nữa là có hay không một phân rã W thành các dạng Boyce – Codd(BCNF) mà vẫn bảo toàn phụ thuộc và có nối không mất.

Trong mục này ta sẽ xét và nghiên cứu các vấn đề này.

Bổ đề 4.3:

a. Mỗi sơ đồ W= < R, F> với R có hai thuộc tính đều có dạng BCNF.

b. Nếu R không có dạng BCNF thì tồn tại hai thuộc tính A và B trong R sao cho (R - AB) $\rightarrow$ A.

Chứng minh:

Đối với trường hợp (a), gọi AB là một lược đồ, ở lược đồ này chỉ có hai phụ thuộc không tầm thường có thể đúng: A $\rightarrow$ B và B $\rightarrow$ A. Nếu không

4.3.5. PHÂN MẢNH DỌC THÀNH CÁC BCNF CÓ NỐI KHÔNG MẤT THÔNG TIN VÀ BẢO TOÀN PHỤ THUỘC

Trong các phần trên chúng ta đã có các thuật toán và các phép kiểm tra tương ứng cho các phân rã W thành :

(1) Nối không mất.

(2) Bảo toàn phụ thuộc.

(3) Bảo toàn phụ thuộc và nối không mất.

Bây giờ chúng ta lần lượt xét các phân rã W thành các sơ đồ con BCNF có kèm theo các tính chất (1) hoặc (2) hoặc (3).

Chúng ta cảm nhận rằng mọi lược đồ quan hệ đều có một phân rã có nối không mất thành dạng chuẩn Boyce – Codd (BCNF).

Mọi lược đồ cũng có thể phân rã thành dạng BCNF bảo toàn phụ thuộc.

Một vấn đề đặt ra nữa là có hay không một phân rã W thành các dạng Boyce – Codd(BCNF) mà vẫn bảo toàn phụ thuộc và có nối không mất.

Trong mục này ta sẽ xét và nghiên cứu các vấn đề này.

Bổ đề 4.3:

a. Mỗi sơ đồ W= < R, F> với R có hai thuộc tính đều có dạng BCNF.

b. Nếu R không có dạng BCNF thì tồn tại hai thuộc tính A và B trong R sao cho (R - AB) $\rightarrow$ A.

Chứng minh:

Đối với trường hợp (a), gọi AB là một lược đồ, ở lược đồ này chỉ có hai phụ thuộc không tầm thường có thể đúng: A $\rightarrow$ B và B $\rightarrow$ A. Nếu không

có phụ thuộc nào đúng thì chắc chắn không có vi phạm BCNF. Nếu chỉ A → B đúng thì A là một khoá, vì thế không có vi phạm của BCNF. Nếu chỉ B → A đúng thì B là khoá, và nếu cả hai đúng, cả A và B đều là khoá, vì thế không bao giờ có vi phạm BCNF.

Đối với trường hợp (b), giả sử có một vi phạm X → A trong R. Thế thì R phải có một thuộc tính B không thuộc tập XA, nếu không thì $X^+ = R$, và X → A không phải là vi phạm. Vì B và A không thuộc X nên tập R – AB vẫn chứa X mà X → A, do đó (R - AB) → A chính là điều phải chứng minh.

Bổ đề 4.4:

Nếu chúng ta có một tập phụ thuộc F trên R và chúng ta chiếu các phụ thuộc trên $R_1 \subseteq R$ để được F_1, rồi lại chiếu F_1 trên $R_2 \subseteq R_1$ để được F_2 thì

$$F_2 = \pi_{R2}(F)$$

Nghĩa là ta có thể giả sử rằng F là tập phụ thuộc của R_1, dù rằng F có thể có những thuộc tính không có trong R_1.

Chứng minh:

Nếu $XY \subseteq R_2$ thì X →Y thuộc F^+ nếu và chỉ nếu nó thuộc F_1^+.

Bổ đề 4.5 :

Với mọi sơ đồ quan hệ W = < R, F >, luôn tồn tại sơ đồ quan hệ

W' = < R, F'> sao cho F tương đương với F'và trong F' chỉ gồm những phụ thuộc hàm mà phía bên phải của nó chỉ có một thuộc tính. Chứng minh bổ đề này hiển nhiên.

Trong các thuật toán phân rã sau đây ta hay xét những sơ đồ quan hệ W mà bên phải các phụ thuộc hàm F chỉ gồm một thuộc tính.

4.3.5.1 Phân rã dọc có nối không mất và BCNF

Chúng ta có thể sử dụng các bổ đề trên để xây dựng thuật toán phân rã một sơ đồ quan hệ $W = < R, F >$ thành các sơ đồ con $W_i = < R_i, F_i >$ BCNF có nối không mất. Ý tưởng chính của thuật toán:

Bắt đầu từ W nếu có một vi phạm BCNF trong W, giả sử là $X \rightarrow A$, chúng ta phân rã R thành các lược đồ XA và R-A . Cả hai đều nhỏ hơn R, bởi vì XA không thể bằng R (nếu bằng R thì $X^+ = R$, nên $X \rightarrow A$ sẽ không vi phạm BCNF). Theo định lý 4.2 phân rã R thành $R_1 = XA$ và $R_2 = R - A$ có nối không mất, bởi vì $R_1 \cap R_2 = X$, $R_1 - R_2 = A$ và $X \rightarrow A$. Chúng ta tính F_1, F_2 là chiếu tương ứng của các phụ thuộc F trên XA và $R_2 = R - A$, và được hai sơ đồ quan hệ con $W_1 = < R_1, F_1 > = < XA, X \rightarrow A >$, $W_2 = < R_2, F_2 >$, với W_1 là BCNF, coi W_2 là W lại áp dụng tương tự như trên tìm một vi phạm trong W cho bước phân rã tiếp theo. Tiếp tục phân rã cho đến khi không thể phân rã được nữa, nghĩa là sơ đồ quan hệ con cuối cùng cần phân rã hoặc có nhỏ hơn hai thuộc tính hoặc có tập phụ thuộc bằng rỗng, hoặc một sơ đồ quan hệ BCNF. Thêm lược đồ cuối cùng vào phân rã. Khi đó tất cả các sơ đồ đều có dạng BCNF sẽ là một phân rã có nối không mất.

***Thuật toán 4.7**: Phân rã W thành các BCNF có nối không mất*

Input: $W = <R, F>$; $F = \{X \rightarrow A\}$

Output : Một phân rã R có nối không mất, sao cho mỗi sơ đồ quan hệ con $W_i = < R_i, F_i >$ có dạng BCNF ;

Phương pháp:

Trọng tâm của thuật toán là lấy lược đồ quan hệ R rồi phân rã nó thành hai lược đồ. Một lược đồ có tập các thuộc tính XA; và sơ đồ quan hệ W = < XA, $X \rightarrow A$ > là BCNF. Lược đồ thứ hai sẽ là R - A, do đó nối của R - A với XA là nối không mất. Thực hiện đệ qui thủ tục phân rã, với R - A ở vị trí của R cho đến khi chúng ta đạt được kết quả mong muốn.

Z : = R; {* bất kỳ lúc nào Z cũng là lược đồ phân rã mà có thể W không có dạng BCNF *}.

K:= 1;

Repeat

If $X \rightarrow A$ vi phạm BCNF trong W= < Z, F> then

begin

$R_k := XA$; $F_k := X \rightarrow A$; $W_k := < R_k, F_k >$; F := F – F_k và loại khỏi F những phụ thuộc hàm có chứa A; Z:= Z- A ;

W:= < Z,F>;

{Phân rã W thành W_k và W, trong đó W_k có dạng PCNF}

end;

k:= k + 1;

until không thể phân rã Z tiếp;

thêm Z vào phân rã;

Thí dụ 4.26:

Xét lược đồ quan hệ R(C,T,H,P,S,G), với C - course (lớp, khoá học), T = teacher (giảng viên), H - hour (giờ học), P = phòng học, S = student (sinh viên) và G = grade (điểm số). Các phụ thuộc hàm F theo ngữ nghĩa của bài toán chúng ta giả sử tồn tại là:

$C \rightarrow T$ Mỗi khoá có một giảng viên.

$HP \rightarrow C$ Tại một giờ học, một phòng chỉ có một lớp.

$HT \rightarrow P$ Tại một giờ học, một giảng viên chỉ có mặt trong một phòng học.

$CS \rightarrow G$ Mỗi sinh viên có một điểm số cho mỗi khoá học.

$HS \rightarrow P$ Tại một giờ học, một sinh viên chỉ có mặt tại một phòng học.

Vậy chúng ta có $W = < R, F >$; với R= { C,T,H,P,S,G } và tập phụ thuộc hàm $F = \{ C \rightarrow T, HP \rightarrow C, HT \rightarrow P, CS \rightarrow G, HS \rightarrow P \}$. Ta thấy ngay $C \rightarrow T$ vi phạm BCNF của sơ đồ quan hệ nên $W_1 = < \{C, T\}, C \rightarrow T >$ và W còn lại (sau khi đã bỏ thuộc tính T và những phụ thuộc hàm có chứa T) cần phân rã tiếp là:

$W = < \{C,H,P,S,G\}, \{HP \rightarrow C, CS \rightarrow G, HS \rightarrow P\} >$. Tiếp tuc ta có

$W_2 = < \{H,P,C\}, HP \rightarrow C >$ và W lại còn là $W = < \{H,P,S,G\}, HS \rightarrow G >$. Tiếp tuc ta có $W_3 = < \{H,S, G\}, HS \rightarrow G >$ và $W = < \{H,P,S\}, \emptyset >$; đặt $W_4 = < \{H,P,S\}, \emptyset >$. Vậy W_1, W_2, W_3, W_4 là những sơ đồ con có dạng BCNF và phân rã $p = (R_1,R_2,R_3,R_4) = (CT, HPC, HSG, HPS)$ của R có nối không mất.

Trong phép phân rã này phụ thuộc ham $HS \rightarrow G$ không thuộc F mà thuộc F^+.

Cần lưu ý rằng trong thực hành có thể thực hiện nhanh hơn bằng cách sau mỗi lần tách được một Wi =< XA, X → A> ta loại bỏ khỏi W phụ thuộc X → A và những phụ thuộc có chứa A. Và loại A ra khỏi tập thuộc tính. Thí dụ quay lại thí dụ trên ta thực hiện như sau:

W_1 = < CT, C → T >; W = < CHPSG, { HP → C, HS → P}>

W_2 = < HPC, HP → C>; W = < HPSG, HS → P>

W_3 = < HSP, HS → P >; W = < HSG, Ø > và

W_4 = < HSG, Ø>.

Phép phân rã này không bảo toàn phụ thuộc vì đã bỏ mất CS → G của F.

Tuy nhiên, vẫn có những sơ đồ quan hệ W được phân rã thành các sơ đồ con dạng BCNF có nối không mất và bảo toàn phụ thuộc.

Thí dụ xét SĐQH W và phép phân rã sau:

W = < ABCDEHI, { A → B, C → D, E → H}>;

W_1 = < AB, A → B >; W = <ACDEHI, { C → D, E → H}>;

W_2 = < CD, C → D>; W= <ACEHI, E → H >;

W_3 = < EH, E → H >; W = < ACEI, Ø >;

W_4 = < ACEI, Ø >;

Tính bảo toàn phụ thuộc và BCNF của phân rã là hiển nhiên. Dựa vào định lý 4.2 và bổ đề 4.2 ta có thể kết luận được tính nối không mất của phân rã.

Tuy nhiên chúng ta có thể kiểm tra lại phép phân rã có nối không mất bằng thuật toán Chase như sau:

Bảng khởi đầu

	A	B	C	D	E	H	I
R_1	a_1	a_2					
R_2			a_3	a_4			
R_3					a_5	a_6	
R_4	a_1		a_3		a_5		a_7

Bảng sau khi sửa cho thoả A→ B, C→ D và E→H

	A	B	C	D	E	H	I
R_1	a_1	a_2					
R_2			a_3	a_4			
R_3					a_5	a_6	
R_4	a_1	a_2	a_3	a_4	a_5	a_6	a_7

Bảng có một hàng toàn a nên phân rã có nối không mất.

Một vấn đề cần phải quan tâm là mọi sơ đồ quan hệ W có luôn phân rã thành các sơ đồ con BCNF và có nối không mất không?

Chúng ta thấy rằng nếu W có những vi phạm BCNF thì thuật toán trên cho ta một phân rã thành BCNF và có nối không mất. Ngược lại nếu W đã là BCNF thì ta có thể lấy $W_1 = W$, $W_2 = W$,...

Kết luận:

a. *Mọi sơ đồ quan hệ W đều có thể phân rã thành các BCNF có nối không mất.*

b. *Phân rã W thành các sơ đồ quan hệ con BCNF có nối không mất không duy nhất.*

Kết luận này dễ dàng suy ra từ các phân tích vừa xong.

4.3.5.2 Phân mảnh dọc thành các BCNF có bảo toàn phụ thuộc

Thuật toán 4.7 trên đây, ta đã phân rã sơ đồ quan hệ thành các sơ đồ con có dạng BCNF và có nối không mất. Tuy nhiên thuật toán không đảm bảo phân rã có bảo toàn phụ thuộc. Và sau đây ta sẽ phân rã sơ đồ thành các BCNF có bảo toàn phụ thuộc. Tất nhiên với W bất kỳ chúng ta luôn có thể tìm được một phân rã thành các BCNF bảo toàn phụ thuộc.

***Thuật toán 4.8**: Phân rã W thành các BCNF có bảo toàn phụ thuộc*

Input: W = < R, F > ;

Output: Một phân rã thành BCNF bảo toàn phụ thuộc ;

Phương pháp:

Nếu có những thuộc tính của R không nằm trong một phụ thuộc nào của F , thì mọi thuộc tính như thế đều có thể tạo ra một sơ đồ con BCNF với tập phụ thuộc rỗng. Nếu một trong những phụ thuộc trong F có chứa tất cả các thuộc tính của R thì kết quả có một mảnh chính là R. Còn lại, phân rã kết quả ρ sẽ chứa các lược đồ XA cho mỗi phụ thuộc X → A trong F.

Thí dụ 4.27:

Xét lại sơ đồ quan hệ W = <R,F> ; với R= {C,T,H,P,S,G} và các phụ thuộc của F:

C → T; CS → G; HP → C; HS → P; HT → P

Thuật toán 4.7 sinh ra tập sơ đồ quan hệ :

W_1 = < CT, C→T>,

W_2 = < CHP, HP→ C>,

W_3 = < THP, HT→P >,

W_4 = < CSG, CS →G >,

W_5 = < HPS, HS→ P >.

Định lý 4.3:

Thuật toán 4.7 tạo ra một phân rã bảo toàn phụ thuộc thành các dạng chuẩn BCNF .

Chứng minh:

Bởi vì các phụ thuộc hình chiếu là những phụ thuộc của F, nên phân rã rõ ràng là bảo toàn phụ thuộc. Còn mỗi sơ đồ con dạng < XA, X→A > rõ ràng là BCNF .

***Kết luận** : Mọi sơ đồ quan hệ đều có thể phân rã thành các BCNF (và tất nhiên là 3NF) có bảo toàn phụ thuộc.*

4.3.5.3 Phân mảnh dọc thành các BCNF và có bảo toàn phụ thuộc có nối không mất thông tin

Thuật toán 4.8 trên đây ta xét phân rã với 2 điều kiện: thành BCNF và bảo toàn phụ thuộc. Sau đây chúng ta sẽ xét phân rã với 3 điều kiện : thành BCNF, bảo toàn phụ thuộc , có nối không mất.

Như đã phân tích và kết luận trước đây, ta có thể phân rã R của sơ đồ quan hệ W = < R,F> bất kỳ thành một tập các lược đồ con:

$$\rho = (R_1, \ldots, R_k)$$

sao cho ρ có nối không mất và mỗi $<R_i, F_i>$ có dạng BCNF . Chúng ta cũng có thể phân rã R thành $\sigma = (S_1, \ldots, S_m)$ sao cho σ bảo toàn tập phụ thuộc F, và mỗi $<S_j, F_i>$ có dạng BCNF . Vấn đề bây giờ là liệu chúng ta có thể phân rã W thành các BCNF mà vẫn có cả hai đặc tính bảo toàn phụ thuộc và nối không mất? Chúng ta chỉ cần nối vào σ một lược đồ quan hệ k, với k là khoá của W = < R, F > như thuật toán sau:

Thuật toán 4.9: *Phân rã W thành các BCNF, bảo toàn phụ thuộc, nối không mất.*

Input W = < R, F >; k là khoá của W.

Output $W_1, W_2, \ldots$ Ws BCNF, nối không mất, bảo toàn phụ thuộc.

Phương pháp:

Nếu có các thuộc tính không thuộc tập F thì nhóm các thuộc tính đó thành một sơ đồ con với tập phụ thuộc rỗng.

$W_i = < XA, X \rightarrow A >$; với mọi phụ thuộc của F.

$W_k = < k, \pi_k(F) >$; k là khoá của W và $\pi_k(F)$ là chiếu của F lên k.

Định lý 4.4 Thuật toán 4.9 phân rã một sơ đồ quan hệ thành các BCNF, bảo toàn phụ thuộc có nối không mất.

Chứng minh: Rõ ràng phép phân rã trên có bảo toàn phụ thuộc và thành các BCNF . Để đảm bảo tất cả các sơ đồ quan hệ con đều là BCNF ta chỉ cần kiểm tra lại W_k . Giả sử trong W_k có một vi phạm BCNF, tức có $Y \rightarrow A$ và $Y^+ \neq k$, A là thuộc tính khoá, Y là tập con thực sự của khoá K. Điều này vô lý vì k là khoá nên theo tính tối thiểu không thể có một tập con của khoá kéo theo một thuộc tính của khoá nếu có như vậy ta có thể bỏ thuộc tính A đi.

Trước khi chứng minh tính nối không mất của phân rã ta lấy một vài thí dụ minh hoạ.

Thí dụ 4.28:

Chúng ta xem lại thí dụ $W = < R, F >$; với R= { C,T,H,P,S,G } và tập phụ thuộc hàm F = { $C \rightarrow T$, $HP \rightarrow C$, $HT \rightarrow P$, $CS \rightarrow G$, $HS \rightarrow P$ }.

Khoá của W là k = { H, S} và chiếu của F lên k là rỗng. Phân rã theo thuật toán 4.8 ta được:

$W_1 = < CT, C \rightarrow T >$;

$W_2 = < HPC, HP \rightarrow C >$;

$W_3 = < HPT, HT \rightarrow P >$;

$W_4 = < CS, CS \rightarrow G >$;

$W_5 = < HSP, HS \rightarrow P >$;

$W_6 = < HS, \emptyset > = W_k$;

Hiển nhiên phép phân rã có bảo toàn phụ thuộc và thành các BCNF.

Kiểm tra lại tính nối không mất bằng thuật toán Chase:

Bảng khởi đầu

	C	G	H	P	S	T
R_1	a					a
R_2	a		a	a		
R_3			a	a		a
R_4	a	a			a	
R_5			a	a	a	
R_6			a		a	

Sau khi sửa cho bảng khởi đầu trên đây thoả một số phụ thuộc trong F ta được một bảng chứa toàn a ở hàng thứ 6, hàng chứa các thuộc tính khoá.

Bảng có hàng $R_6 = k = \{H, S\}$ toàn a.

Vậy phân rã có nối không mất.

	C	G	H	P	S	T
R_1	a					a
R_2	a		a	a		a
R_3	a		a	a		a
R_4	a	a			a	a
R_5	a		a	a	a	a
R_6	a	a	a	a	a	a

Thí dụ 4.29:

Xét sơ đồ quan hệ W = < R, F >; với

R = {A, B, C, D, E, G, H, I }; F = {AB →CD, DA→BG, EG→H, H→IE, IA→BD}. Ta thấy K = {A, H} là khoá của W và chiếu của tập F lên K là rỗng.

Bây giờ ta phân rã theo thuật toán 4.8

W_1 = < ABCD, AB→CD >;

W_2 = < DABG, DA→BG >;

W_3 = < EGH, EG→H >;

W_4 = < HIE, H→IE >;

W_5 = < IABD, IA→BD >;

W_6 = < AH, Ø > = W_k.

Ta dễ dàng thấy rằng các Wi là những sơ đồ quan hệ BCNF. Phép phân rã báo toàn phụ thuộc.

Các bạn cần lưu ý rằng ở đây một số sơ đồ quan hệ có bên phải phụ thuộc hàm là hai thuộc tính. Mỗi sơ đồ như vậy có thể tách thành hai sơ đồ.

Sau đây là bảng kiểm tra tính nối không mất.

Bảng khởi đầu (không làm sai lệch thuật toán ta viết a thay cho a_j)

	A	B	C	D	E	G	H	I
R_1	a	a	a	a				
R_2	a	a		a		a		
R_3					a	a	a	
R_4					a		a	a
R_5	a	a		a				a
R_6	a						a	

Bảng sau khi đã sửa cho thoả một số phụ thuộc hàm của tập phụ thuộc hàm F

	A	B	C	D	E	G	H	I
R_1	a	a	a	a		a		
R_2	a	a	a	a		a		
R_3					a	a	a	a
R_4					a		a	a
R_5	a	a	a	a	a	a		a
R_6	a	a	a	a	a	a	a	a

Như vậy hàng thứ 6, R_6 = k = {A, H} là khoá chứa toàn a nên phân rã có nối không mất. Qua hai ví dụ ta thấy rằng cuối cùng hàng chứa khoá k luôn chứa toàn a.

Bây giờ ta phải chứng minh tính chất nối không mất của phân rã theo thuật toán 4.9 và cũng là hoàn chỉnh chứng minh định lý 4.4 .

Qua hai ví dụ trên ta thấy sau một số lần sửa cho bảng khởi đầu thoả các phụ thuộc trong F thì hàng chứa khoá k sẽ chứa toàn a.

Phần này của định lý xem trang 473 trong [2].

4.3.6. LỜI KẾT VỀ VẤN ĐỀ PHÂN MẢNH DỌC

Ta cần nhắc lại rằng bài toán phân rã luôn gắn với mục tiêu của bài toán. Rất nhiều vấn đề của phân tán chúng ta vẫn phải khám phá và sáng tạo heuristic. Những vấn đề ta đã phân tích và nghiên cứu ở đây chung cho cả phân rã tập trung và phân rã theo tầm địa lý trên mạng.

Trong phần phân rã tập trung, với những công cụ đã cho trong các Thuật toán chúng ta chỉ dùng để tham khảo cho bài toán cụ thể của mình. Điều quan trọng cần nhớ là không phải mọi phân rã có nối không mất đều hữu ích, thậm chí một số còn có hại. Sai lầm hay gặp nhất là đi phân rã một lược đồ đã có dạng BCNF chỉ vì một phân rã có nối không mất và bảo toàn phụ thuộc.

Chẳng hạn chúng ta có thể có một quan hệ cung cấp thông tin về nhân viên bán hàng, thí dụ như mã số nhân viên I, tên nhân viên N, gian hàng D, lương S. Bởi vì I là khoá duy nhất trong trường hợp này, chúng ta có $I \rightarrow A$ với mọi thuộc tính A. Vì vậy hoàn toàn có thể phân rã lược đồ này thành IN, ID và IS. Dễ thấy rằng phân rã này có nối không mất vì I, thuộc tính duy nhất trong phần giao của mỗi cặp , xác định tất cả các thuộc tính; cũng rõ ràng là nó bảo toàn phụ thuộc $I \rightarrow NDS$.

Tuy nhiên, chính lược đồ INDS lại có dạng BCNF, và có những ưu điểm khi phải trả lời những câu vấn tin có liên quan đến những thuộc tính

khác I. Chẳng hạn nếu chúng ta muốn biết tên và lương của tất cả nhân viên của gian hàng đồ chơi trẻ em, ta phải nối IN |><| ID |><| IS của các lược đồ đã được phân rã, nhưng có thể trả lời ngay được câu hỏi mà không cần lấy bất cứ nối nào nếu chúng ta để nguyên lược đồ quan hệ cũ. Hơn nữa lược đồ được phân rã đòi hỏi mã số nhân viên phải được lặp lại ở nhiều vị trí mặc dù nó không phải là dư thừa nhưng tốn kém bộ nhớ.

Vậy khi áp dụng lý thuyết phân rã, cần phải nhớ rằng phân rã là cứu cánh được sử dụng để giải quyết các vấn đề dư thừa, bất thường và mạng chứ không phải là mục đích. Khi áp dụng Thuật toán 4.6b chúng ta nên tránh phân rã nếu lược đồ đã có dạng BCNF. Khi sử dụng Thuật toán 4.8 chúng ta nên xem xét đến việc tổ hợp các lược đồ quan hệ được tạo ra nếu không có phạm vi BCNF.

4.4. CẤP PHÁT

Cấp phát tài nguyên cho các nút của một mạng máy tính là một bài toán đã được nhiều người quan tâm và nghiên cứu rộng rãi. Tuy nhiên, phần lớn các nghiên cứu này đều không tập trung vào bài toán thiết kế CSDL phân tán, mà vào cách thức đặt các tập tin trên một mạng máy tính. Chúng ta sẽ xem xét một cách tổng quan những khái niệm này. Trước tiên, cần phải định nghĩa bài toán cấp phát một cách chính xác hơn.

4.4.1.BÀI TOÁN CẤP PHÁT

Giả sử rằng có một tập các mảnh $F = \{F_1, F_2, \ldots . F_k\}$ và một mạng máy tính bao gồm các vị trí $S = \{S_1, S_2, \ldots . S_m\}$ trên đó có một tập các ứng dụng $Q = \{q_1, q_2, \ldots . q_q\}$ đang chạy. Bài toán cấp phát (allocation problem)

là tìm một phân phối "tối ưu" của F cho S.

Tính "tối ưu" (optimality) có thể được định nghĩa ứng với hai số đo [Dowdy and Foster, 1982] :

1- Chi phí nhỏ nhất : Hàm chi phí gồm có chi phí lưu mỗi mảnh F_i tại vị trí S_j, chi phí vấn tin F_i tại vị trí S_j, chi phí cập nhật F_i tại tất cả mọi vị trí có chứa nó, và chi phí truyền dữ liệu. Vì thế bài toán cấp phát cố gắng tìm một lược đồ cấp phát với hàm chi phí tổ hợp thấp nhất.

2- Hiệu quả. Chiến lược cấp phát được thiết kế nhằm duy trì một hiệu quả lớn đó là hạ thấp thời gian đáp ứng và tăng tối đa lưu lượng hệ thống tại mỗi vị trí.

Nói chung bài toán cấp phát tổng quát là một bài toán phức tạp và có độ phức tạp là NP- đầy đủ (NP-complete). Vì thế các nghiên cứu đã được dành cho việc tìm ra các giải thuật heuristic tốt để có thể có được các lời giải gần tối ưu. Để phân biệt bài toán cấp phát tập tin truyền thống với cấp phát mảnh trong các thiết kế CSDL phân tán, chúng ta gọi bài toán thứ nhất là bài toán cấp phát tập tin FAP (file allocation problem) và bài toán sau là bài toán cấp phát CSDL DAP (database allocation problem).

Hiện không có một mô hình heuristic tổng quát nào nhận một tập các mảnh và sinh ra một chiến lược cấp phát gần tối ưu ứng với các loại ràng buộc đã thảo luận ở đây.

Các mô hình đã được phát triển chỉ mới đưa ra một số giả thiết đơn giản hóa và dễ áp dụng cho một số cách đặt vấn đề cụ thể. Vì thế thay vì trình bày một hoặc nhiều thuật toán cấp phát, chúng tôi trình bày một mô hình tương đối tổng quát và thảo luận một số heuristic có thể được dùng để giải quyết chúng.

4.4.2.THÔNG TIN CHO CẤP PHÁT

Ở giai đoạn cấp phát, chúng ta cần các thông tin định lượng về CSDL, về các ứng dụng chạy trên đó, về cấu trúc mạng, khả năng xử lý và giới hạn lưu trữ của mỗi vị trí trên mạng.

4.4.2.1.Thông tin về CSDL

Để thực hiện phân mảnh ngang, chúng ta đã định nghĩa độ tuyển hội sơ cấp. Bây giờ chúng ta cần mở rộng định nghĩa đó cho các mảnh, và định nghĩa độ tuyển của một mảnh F_j ứng với câu vấn tin q_i. Đây là số lượng các bộ của F_j cần được truy xuất để xử lý q_i. Giá trị này được ký hiệu là $sel_i(F_j)$.

Một loại thông tin khác trên các mảnh là kích thước của chúng. Kích thước một mảnh F_j được cho bởi:

$$size(F_j) = card(F_j) * Length(F_j)$$

trong đó length (F_j) là chiều dài (tính theo byte) của một bộ trong mảnh F_j.

4.4.2.2 Thông tin về ứng dụng

Phần lớn các thông tin có liên quan đến ứng dụng đều đã được biên dịch trong khi thực hiện phân mảnh nhưng cũng cần một số ít nữa cho mô hình cấp phát. Hai số liệu quan trọng là số truy xuất đọc do câu vấn tin q_i thực hiện trên mảnh F_j trong mỗi lần chạy của nó (ký hiệu là RR_{ij}), và tương ứng là các truy xuất cập nhật (UR_{ij}). Thí dụ chúng có thể đếm số truy xuất khối cần phải thực hiện theo yêu cầu vấn tin.

Chúng ta cũng cần định nghĩa hai ma trận UM và RM với các phần tử tương ứng u_{ij} và r_{ij} được đặc tả như sau :

$$u_{ij} = \begin{cases} 1 \text{ nếu vấn tin } q_i \text{ có cập nhật mảnh } F_j \\ 0 \text{ trong trường hợp ngược lại} \end{cases}$$

$$r_{ij} = \begin{cases} 1 \text{ nếu vấn tin } q_i \text{ cần đọc mảnh } F_j \\ 0 \text{ trong trường hợp ngược lại} \end{cases}$$

Một vectơ O gồm các giá trị 0 (i) cũng được định nghĩa, với 0 (i) đặc tả vị trí đưa ra câu vấn tin q_i. Cuối cùng để định nghĩa ràng buộc thời gian đáp ứng, thời gian đáp ứng tối đa được phép của mỗi ứng dụng cũng cần phải được đặc tả.

4.4.2.3 Thông tin về vị trí

Với mỗi vị trí (trạm) chúng ta cần biết về khả năng lưu trữ và xử lý của nó. Hiển nhiên là những giá trị này có thể tính được bằng các hàm thích hợp hoặc bằng các phương pháp đánh giá đơn giản. Chi phí đơn vị để lưu dữ liệu tại vị trí S_k sẽ được ký hiệu là USC_k. Cũng cần phải đặc tả số đo chi phí LPC_k, là chi phí xử lý một đơn vị công việc tại vị trí S_k. Đơn vị công việc cần phải giống với đơn vị của RR và UR.

4.4.2.4 Thông tin về mạng

Trong mô hình đang xét, chúng ta giả sử có tồn tại một mạng đơn giản, trong đó chi phí truyền mỗi bó giữa hai vị trí S_i và S_j. Để có thể tính

được số lượng thông báo, chúng ta dùng fsize làm kích thước (tính theo byte) của một bó dữ liệu. Chắc chắn là có các mô hình mạng phù hợp hơn có xem xét đến khả năng truyền của các kênh, khoảng cách giữa các vị trí, chi phí giao thức, vân vân. Tuy nhiên xét phương trình như thế vượt ra khỏi phạm vi cuốn sách này.

4.4.3. MÔ HÌNH CẤP PHÁT

Chúng ta sẽ thảo luận một mô hình cấp phát có mục tiêu là giảm thiểu tổng chi phí xử lý và lưu trữ .

Mô hình của chúng ta có hình thái như sau :

min (Total Cost)

ứng với ràng buộc thời gian đáp ứng, ràng buộc lưu trữ, ràng buộc xử lý

Trong những đoạn còn lại, chúng ta khai triển những thành phần của mô hình này dựa trên biến quyết định x_{ij}, được định nghĩa là

$$x_{ij} = \begin{cases} 1 \text{ nếu mảnh } F_i \text{ được lưu tại vị trí } S_j \\ 0 \text{ trong trường hợp ngược lại} \end{cases}$$

TỔNG CHI PHÍ

Hàm tổng chi phí có hai thành phần : Phần xử lý vấn tin và phần lưu trữ, vì thế nó có thể được biểu diễn là:

$$TOC = \sum_{q_i \in Q} QPC_i + \sum_{\forall S_j} \sum_{F_j \in F} STC_{jk}$$

với QPC_i là chi phí xử lý câu vấn tin của ứng dụng q_i, và STC_{jk} là chi phí lưu mảnh F_j tại vị trí S_k được tính bởi:

$$STC_{jk} = USC_k * size(F_j) * x_{jk}$$

Chi phí xử lý vấn tin khó xác định hơn. Hầu hết các mô hình cho bài toán cấp phát tập tin FAP tách nó thành hai thành phần : Chi phí xử lý chỉ đọc và chi phí xử lý cập nhật. Chúng tôi chọn một lối tiếp cận khác trong mô hình cho bài toán DAP và xác định nó như chi phí xử lý là PC và chi phí truyền là TC. Vì thế chi phí xử lý vấn tin QPC cho ứng dụng q_i là

$$QPC_i = PC_i + TC_i$$

Thành phần xử lý PC gồm có ba hệ số chi phí, chi phí truy xuất AC, chi phí duy trì toàn vẹn IE và chi phí điều khiển đồng thời CC :

$$PC_i = AC_i + IE_i + CC_i$$

Mô tả chi tiết cho mỗi hệ số chi phí phụ thuộc vào thuật toán được dùng để hoàn tất các tác vụ đó. Tuy nhiên để minh hoạ, chúng tôi sẽ mô tả chi tiết về AC:

$$AC_i = \sum_{S_k \in S} \sum_{F_j \in F} (u_{ij} * UR_{ij} + r_{ij} * RR_{ij}) * x_{jk} * LPC_k$$

Hai số hạng đầu trong công thức trên tính số truy xuất của vấn tin q_i đến mảnh F_j. Chú ý rằng $(UR_{ij} + RR_{ij})$ là tổng số các truy xuất đọc và cập nhật. Chúng ta giả thiết rằng các chi phí xử lý chúng là như nhau. Ký hiệu tổng cho biết tổng số các truy xuất cho tất cả mọi mảnh được q_i tham chiếu. Nhân với LPC_k cho ra chi phí của truy xuất này tại vị trí S_k. Chúng ta lại dùng x_{jk} để chỉ chọn các giá trị chi phí cho các vị trí có lưu các mảnh.

Hàm chi phí truyền có thể được đưa ra giống như cách của hàm chi phí truy xuất. Tuy nhiên tổng chi phí truyền dữ liệu cho cập nhật và cho yêu cầu chỉ đọc sẽ khác nhau hoàn toàn. Trong các vấn tin cập nhật, chúng ta cần cho tất cả mọi vị trí biết nơi có các bản sao, còn trong vấn tin chỉ đọc thì

chỉ cần truy xuất một trong các bản sao là đủ. Ngoài ra, vào lúc kết thúc yêu cầu cập nhật thì không cần phải truyền dữ liệu ngược lại cho vị trí đưa ra vấn tin ngoài một thông báo xác nhận, còn trong vấn tin chỉ đọc có thể phải có nhiều thông báo truyền dữ liệu.

Thành phần cập nhật của hàm truyền dữ liệu là

$$TCU_i = \sum_{S_k \in S} \sum_{F_j \in F} u_{ij} * x_{jk} * g_{o(i),k} + \sum_{S_k \in S} \sum_{F_j \in F} u_{ij} * x_{jk} * g_{k,o(i)}$$

Số hạng thứ nhất để gửi thông báo cập nhật từ vị trí gốc 0 (i) của q_i đến tất cả bản sao cần cập nhật. Số hạng thứ hai dành cho thông báo xác nhận.

Thành phần chi phí chỉ đọc có thể được đặc tả là

$$TCR_i = \sum_{F_j \in F} \sum_{S_k \in S} \min (u_{ij} * x_{jk} * g_{o(i),k} + r_{ij} * x_{jk} * \frac{sel_i(Fj) * lengt(Fj)}{fsize} * g_{k,o(i)})$$

Số hạng thứ nhất trong TCR biểu thị chi phí truyền yêu cầu chỉ đọc đến những vị trí có bản sao của mảnh cần truy xuất. Số hạng thứ hai để truyền các kết quả từ những vị trí này đến vị trí yêu cầu. Phương trình này khẳng định rằng trong số các vị trí có bản sao của cùng một mảnh, chỉ vị trí sinh ra tổng chi phí truyền thấp nhất mới được chọn để thực hiện thao tác này.

Bây giờ hàm chi phí truyền cho vấn tin q_i có thể được tính là

$$TC_i = TCU_i + TCR_i$$

RÀNG BUỘC

Các hàm ràng buộc có thể được đặc tả tương tự. Tuy nhiên thay vì mô tả những hàm này kỹ lưỡng, chúng tôi sẽ mô tả xem hình thái của chúng sẽ như thế nào. Ràng buộc thời gian đáp ứng cần được đặc tả là

Thời gian thực thi của $q_i \leq$ thời gian đáp ứng lớn nhất của q_{ji}, $\forall q_i \in Q$

Người ta thích đặc tả số đo chi phí của hàm theo thời gian bởi vì nó đơn giản hóa đặc tả về ràng buộc thời gian thực thi.

Ràng buộc lưu trữ là:

$$\sum_{\forall Fj \in F} STC_{jk} \leq \text{khả năng lưu trữ tại vị trí } S_k \quad \forall S_k \in S$$

Trong đó ràng buộc xử lý là :

$$\sum_{\forall qi \in Q} \text{tải trọng xử lý của } q_i \text{ tại vị trí } S_k \leq \text{khả năng xử lý của } S_k, \forall S_k \in S$$

Như thế chúng ta kết thúc phần phân tích về mô hình cấp phát ở đây. Mặc dù chúng ta đã không xây dựng nó một cách đầy đủ, song sự chính xác qua một số thuật ngữ đã cho thấy làm thế nào để trình bày một bài toán như thế. Ngoài khía cạnh này, chúng ta cũng đã chỉ ra các vấn đề quan trọng cần chú ý trong các mô hình cấp phát.

4.4.4. KẾT LUẬN CHO CẤP PHÁT

Trong phần trước chúng ta đã nhấn mạnh một mô hình cấp phát tổng quát có tính phức tạp và rắc rối đáng kể. Mô hình FAP và mô hình DAP thuộc loại NP - complete. Chính vì lẽ đó mà người ta phải tìm kiếm các phương pháp heuristic có thể cho ra những lời giải gần tối ưu. Kiểm tra "chất lượng"

trong trường hợp này tất nhiên là xem kết quả của thuật toán heuristic tiệm cận đến mức độ nào so với cấp phát tối ưu.

Một số heuristic đã được áp dụng làm giải pháp cho các mô hình FAP và DAP. Từ lâu người ta đã nhận ra rằng có sự tương ứng giữa bài toán FAP và bài toán chọn vị trí đặt thiết bị đã được khám phá trong các nghiên cứu về quá trình điều hành sản xuất. Thực sự người ta đã chứng tỏ sự đẳng cấu giữa bài toán FAP đơn giản với bài toán định vị kho lưu hàng [ramamoorthy and Wah, 1983]. Vì thế các heuristic đã được phát triển bởi các nhà nghiên cứu về điều hành thường được dùng để giải các bài toán FAP và DAP. Một số thí dụ đã được nêu trong [Ceri et al., 1982b], [Fischer and Hochbaum, 1980] và [Chang and liu, 1982].

Vậy thiết kế CSDL trên mạng hiện tại vẫn còn nhiều vấn đề buộc các chuyên gia tin học phải quan tâm.

BÀI TẬP

4.1*. Cho quan hệ EMP như trong Hình 4.1, gọi p_1 : TITLE < "Programmer" và P_2 : TITLE > "Programmer" là hai vị từ đơn giản. Giả sử rằng chuỗi ký tự được sắp theo thứ tự chữ cái.

(a) Thực hiện phân mảnh ngang cho quan hệ EMP ứng với $\{p_1, p_2\}$

(b) Giải thích tại sao phân mảnh kết quả $\{EMP_1, EMP_2\}$ không đáp ứng được quy tắc đúng đắn của phân mảnh.

(c) Sửa lại các vị từ p_1 và p_2 để chúng phân hoạch EMP tuân theo các qui tắc đúng đắn của phân mảnh. Để làm điều đó, hãy sửa lại các vị từ, xây dựng tất cả các vị từ hội sơ cấp và suy ra các phép kéo theo tương ứng rồi thực hiện phân mảnh ngang của EMP dựa trên các vị từ hội sơ cấp này. Cuối cùng chứng tỏ rằng kết quả có tính đầy đủ, tính tái thiết được và tính tách biệt.

4.2. Xét quan hệ ASG trong hình 4.1. Giả sử có hai ứng dụng truy xuất ASG. ứng dụng thứ nhất được đưa ra tại năm vị trí và muốn tìm thời gian phân công các nhân viên khi biết mã số nhân viên. Giả sử rằng nhà quản lý, nhà tư vấn, kỹ sư và lập trình viên được lưu tại bốn vị trí khác nhau. Ứng dụng thứ hai được đưa ra tại hai vị trí trong đó các nhân viên có thời gian phân công dưới 20 tháng được quản lý tại một vị trí còn những người trên 20 tháng được quản lý tại vị trí thứ hai. Hãy phân mảnh ngang nguyên thủy cho ASG với các thông tin trên.

4.3. Xét quan hệ EMP và PAY trong Hình 4.1. EMP và PAY đều được phân mảnh ngang như sau :

EMP_1 = EMP(TITLE = 'Elect.Eng').

EMP_2 = EMP(TITLE = 'Syst.Anal').

EMP_3 = EMP(TITLE = 'Mech.Eng').

EMP_4 = EMP(TITLE = 'Programmer').

PAY_1 = PROJ(Sal ≥30000).

PAY_2 = PROJ(Sal < 30000).

Tính nối nửa EMP_2 |>< PAY_2 và nối tự nhiên của các EMP_i với PAY

4.4. Cho một thí dụ về ma trận CA trong đó điểm tách không duy nhất và phân hoạch này nằm ngay giữa ma trận. Cho biết số lượng các thao tác xê dịch cần thực hiện để có được một điểm tách duy nhất.

4.5. Cho quan hệ PAY như trong hình 4.3 gọi p_1 : SAL < 30000 và p_2 : SAL ≥ 30000 là hai vị từ đơn giản. Hãy thực hiện phân mảnh ngang cho PAY ứng với những vị từ này để có được hai mảnh PAY_1 và PAY_2. Sử dụng phân mảnh của PAY, thực hiện phân mảnh ngang dẫn xuất cho EMP. Chứng tỏ tính đầy đủ, tính tái thiết được và tính tách biệt của phân mảnh cho EMP.

4.6. Gọi Q = $\{q_1, q_2, q_3, q_4, q_5\}$ là một tập vấn tin, A = $\{A_1, A_2, A_3, A_4, A_5\}$ là tập thuộc tính và S = $\{S_1, S_2, S_3\}$ là tập vị trí. Ma trận của hình 4.21a dưới đây mô tả giá trị sử dụng thuộc tính còn ma trận trong hình 4.16b cho biết tần số truy xuất ứng dụng. Giả sử rằng $ref_i(q_k) = 1$ cho mọi q_k và S_i, A_1 là thuộc tính khóa. Sử dụng các thuật toán năng lượng nối và phân hoạch dọc để có được một phân mảnh dọc cho tập thuộc tính trong A.

	A_1	A_2	A_3	A_4	A_5
q_1	1	1	1	0	1
q_2	1	1	1	0	1
q_3	1	0	0	1	1
q_4	0	0	1	0	0
q_5	1	1	1	0	0

(a)

	S_1	S_2	S_3
q_1	10	20	0
q_2	5	0	10
q_3	0	35	5
q_4	0	10	0
q_5	0	15	0

(b)

Hình 4.16 Giá trị sử dụng thuộc tính và tần số truy xuất ứng dụng

4.7.** Viết một thuật toán cho phân mảnh ngang dẫn xuất.

4.8 **. Giả sử định nghĩa khung nhìn sau đây

```
CREATE    VIEW   EMPVIEW (ENO, ENAME, PNO, RESP)
AS        SELECTEMP.ENO, EMP.ENAME, ASG.PNO,ASG.RESP
          FROM        EMP, ASG
          WHEREEMP. ENO = ASG.ENO
          AND         DUR = 24
```

được truy xuất bởi ứng dụng q_1 nằm tại các vị trí 1và 2 với tần số lần lượt là 10 và 20. Giả sử tiếp rằng có một vấn tin q_2 khác được định nghĩa là

```
SELECT    ENO, DUR
FROM      ASG;
```

chạy tại các vị trí 2 và 3 với tần số tương ứng là 20 và 10. Dựa trên những thông tin này hãy xây dựng ma trận use (q_i, A_j) cho các thuộc tính của cả hai

quan hệ EMP và ASG. Cũng xây dựng ma trận tụ lực để có thể dùng nó khi tách quan hệ này thành hai mảnh dọc bằng cách dùng một heuristic hoặc thuật toán BEA.

4.9.** Hãy định nghĩa một cách hình thức ba tiêu chuẩn đúng đắn cho phân mảnh ngang dẫn xuất.

4.10.** Chứng tỏ rằng thuật toán năng lượng nối sinh ra các kết quả như nhau dù sử dụng thao tác hàng hay cột.

4.11.** Sửa lại thuật toán PARTITION để có thể phân hoạch k dòng và tính độ phức tạp của thuật toán thu được.

4.12.** Hãy định nghĩa một cách hình thức ba tiêu chuẩn đúng đắn cho phân mảnh hỗn hợp.

4.13.** Mô tả xem làm sao để có thể mô hình hóa chính xác các yêu cầu dưới đây trong bài toán cấp phát CSDL.

(a) Mối liên hệ giữa các mảnh.

(b) Xử lý vấn tin

(c) Duy trì tính toàn vẹn.

(d) Cơ chế điều khiển đồng thời.

4.14.** Xét những thuật toán heuristic khác nhau cho bài toán cấp phát CSDL.

(a) cho biết một số tiêu chuẩn hợp lý để so sánh những heuristic này.

Phân tích các tiêu chuẩn này.

(b) So sánh thuật toán heuristic ứng với các tiêu chuẩn đã đưa ra.

4.15. Giả sử chúng ta có cơ sở dữ liệu của một công ty hoạt động đầu tư với các thuộc tính sau: *B* (broker, ngời môi giới), *O* (office of a broker, văn phòng của người môi giới), *I* (investor, nhà đầu t), *S* (stock, cổ phần), *Q* (quantily of stock owned by an investor, số lượng cổ phần của nhà đầu tư) và *D* (divident aid by a stock, lãi của mỗi cổ phần) với các phụ thuộc hàm: $S \rightarrow D$, $I \rightarrow B$, $IS \rightarrow Q$ và $B \rightarrow O$.

a) Hãy tìm một khoá cho lược đồ quan hệ R = BOSQID

b) Lược đồ R có bao nhiêu khoá? Hãy chứng minh

c) Hãy tìm một phân rã thành dạng Boyce – Codd có nối không mất.

d) Hãy tìm một phân rã R thành dạng chuẩn cấp ba có nối không mất và bảo toàn phụ thuộc.

4.16. Giả sử chúng ta biểu diễn lược đồ quan hệ R của bài tập 4.15 bằng hai lược đồ ISQD và IBO. Hãy đoán xem có dư thừa và bất thường nào không?

4.17. Nếu ta biểu diễn R của bài 4.15 bằng các lược đồ: SDQ, IB, IQD và BI thì phân rã này có đặc tính nối không mất hay không?

4.18. Giả sử chúng ta biểu diễn R bàI 4.15 bằng các lược đồ ISQ, IB, SD và ISO. Hãy tìm các phủ cực tiểu của các phụ thuộc được chiếu lên mỗi lược đồ quan hệ trên. Tìm mỗi phủ cực tiểu cho hợp của các phụ thuộc được chiếu. Phân rã này có bảo toàn phụ thuộc hay không?

4.19.** Trong CSDL của Bài tập 4.15 hãy thay phụ thuộc hàm S → D bằng phụ thuộc đa trị S →→ D. Nghĩa là D bây giờ biểu thị cho các phần lãi đã nhận từ mỗi cổ phần.

a) Hãy tìm cơ sở phụ thuộc của I.

b) Hãy tìm cơ sở phụ thuộc của BS.

c) Hãy tìm một phân rã đạt dạng chuẩn cấp 4 cho R.

4.20. Xét cơ sở dữ liệu của Ship Voyages (tạm dịch là công ty vận tải bằng đường thuỷ) có các thuộc tính sau: S (ship name, tên tàu), T (type of ship, loại tàu), V (voyage indentifier, mã số chuyến tàu), C (cargo carried by one ship on one vogage, khối lượng hàng hoá được vận chuyển), P (port, bến cảng) và D (day, ngày). Giả sử rằng một chuyến tàu (voyage) có một chuỗi các sự kiện: lấy một loại hàng và phân phối hàng này cho các bến cảng. Một tàu trong một ngày chỉ được ghé qua một cảng, vì thế chúng ta có thể giả định có các phụ thuộc hàm sau: S→T, V→SC, SD→PV.

a) Hãy tìm một phân rã thành dạng BCNF có nối không mất.

b) Hãy tìm phân rã thành dạng 3NF có nối không mất và bảo toàn phụ thuộc.

c) Hãy giải thích tại sao không có phân rã BCNF vừa có nối không mất vừa bảo toàn phụ thuộc cho CSDL này.

4.21. Gọi U là tập thuộc tính và D là tập phụ thuộc (thuộc một loại bất kỳ) trên tập thuộc tính U. Chúng ta hãy định nghĩa SAT (D) là tập các quan hệ r trên U sao cho r thoả mỗi phụ thuộc trong D. Hãy chứng minh.

a) SAT $(D_1 \cup D_2)$ = SAT $(D_1) \cap$ SAT (D_2)

b) Nếu D_1 suy diễn logic được tất cả các phụ thuộc trong D_2 thì

$$SAT(D_1) \supseteq SAT(D_2)$$

4.22. Gọi F là một tập phụ thuộc hàm.

a) Chứng tỏ rằng phụ thuộc hàm $X \rightarrow A$ trong F là dư thừa nếu và chỉ nếu X^+ chứa A khi bao đóng này được tính ứng với $F - \{X \rightarrow A\}$.

b) Chứng minh rằng thuộc tính B trong vế trái X của mỗi phụ thuộc hàm $X \rightarrow A$ là dư thừa nếu và chỉ nếu A thuộc $(X - \{B\})^+$ khi bao đóng được lấy ứng với F.

4.23. Trong thuật toán xây dựng sơ đồ quan hệ chính tắc tương đương với sơ đồ quan hệ đầu, chúng ta đã sử dụng hai *phép biến đổi* trên các tập phụ thuộc hàm để nhận được một phủ cực tiểu là:

i) Loại bỏ phụ thuộc dư thừa

ii) Loại bỏ thuộc tính dư thừa ở vế trái.

Hãy chứng tỏ

a) Nếu trước tiên chúng ta áp dụng (ii) cho đến khi không còn áp dụng được nữa rồi sau đó mới áp dụng (i) cũng cho đến khi không còn áp dụng được nữa thì chúng ta luôn luôn nhận được một phủ cực tiểu (chính tắc).

b) Ngược lại nếu chúng ta áp dụng (i) trước cho đến khi không còn áp dụng được nữa rồi mới áp dụng (ii) cũng cho đến khi không còn áp dụng được nữa thì không chắc chắn chúng ta thu được một phủ cực tiểu.

4.24. Cho SĐQH W= < R, F >, với R = { A, B, C, D }, F ={A→B, C→D }
CMR phân rã ρ = (AB,CD) là một phân rã bảo toàn phụ thuộc nhưng không phải là phân rã có nối không mất của lược đồ ABCD.

4.25. Gọi F = {AB→C, A→D, BD→C}

a) Tìm phủ cực tiểu của F

b) Hãy đưa ra một phân rã của ABCD thành hai lược đồ dạng 3NF và bảo toàn phụ thuộc (ứng với tập phụ thuộc hàm F).

c) Trình bày những phụ thuộc hình chiếu cho mỗi lược đồ tìm được.

d) Kết quả của câu (b) có phải là một phân rã có nối không mất hay không? Nếu không, chúng ta có thể sửa đổi lược đồ như thế nào để phân rã có nối không mất và vẫn bảo toàn phụ thuộc.

4.26. Gọi F = {AB→C, A→B}

a) Tìm một phủ cực tiểu của F.

b) Khi câu (a) được cho trong một kỳ kiểm tra ở một trường đại học, hơn một nửa số sinh viên đã trả lời G = {A→B, B→C}. Chứng tỏ rằng câu trả lời này là sai bằng cách đưa ra một quan hệ thoả F nhưng vi phạm G hoặc chứng minh bằng cách khác .

4.27. Giả sử chúng ta có một lược đồ quan hệ ABCD với các phụ thuộc hàm {A→B, B→C, A→D, D→C}. Gọi ρ là phân rã (AB, AC, BD).

a) Tìm các phụ thuộc hình chiếu cho mỗi lược đồ của ρ.

b) ρ có phải là một phân rã có nối không mất ứng với các phụ thuộc đã cho hay không?

c) ρ có bảo toàn các phụ thuộc đã cho hay không?

4.28. Chứng minh rằng (AB, ACD, BCD) không phải là một phân rã có nối không mất của ABCD ứng với tập phụ thuộc hàm {A→C, D→C, BD→A}.

4.29. Xét lược đồ quan hệ ABCD với các phụ thuộc

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

Giả sử chúng ta muốn tìm một phân rã BCNF có nối không mất.

a) Nếu ở bước đầu tiên chúng ta phân rã ABCD thành ACD và BD thì những phụ thuộc hình chiếu trong hai lược đồ này là những phụ thuộc nào?

b) Những lược đồ này có dạng BCNF không? Nếu không thì cần phải phân rã tiếp như thế nào?

4.30. Đối với những tập phụ thuộc khác nhau, phân rã.

ρ = (AB, BC, CD) của lược đồ R = {A, B, C, D } có thể nối không mất hoặc không. Đối với mỗi tập phụ thuộc dưới đây, hãy chứng minh rằng phân rã trên có nối không mất hoặc cho một phản thí dụ bằng một quan hệ.

a) {A→B, B→C}

b) {B→C, C→D}

c) {B→C}

4.31. Thuật toán kiểm tra tính bảo toàn phụ thuộc cần thực hiện tối đa bao nhiêu vòng nếu F là tập gồm *n* phụ thuộc hàm trên *m* thuộc tính (chỉ cần ước lượng độ lớn là đủ).

4.32. Chứng minh nhận xét sau : Nếu R là một lược đồ quan hệ và $X \subseteq R$ là một khoá của R ứng với tập phụ thuộc F thì X không thể có một vi phạm dạng 3NF ứng với tập phụ thuộc $\pi_X(F)$.

4.33. *Phụ thuộc bao hàm đơn ngôi (*unary inclusion dependency*)* $A \subseteq B$ trong đó A, B là các thuộc tính (có thể từ các quan hệ khác nhau) khẳng định rằng trong những giá trị hợp lệ của các quan hệ, mỗi giá trị xuất hiện trong cột của A cũng xuất hiện trong cột của B. Chứng tỏ rằng các tiên đề sau là đúng đắn và đầy đủ đối với các phụ thuộc bao hàm đơn ngôi.

a) $A \subseteq A$ với mọi A

b) Nếu $A \subseteq B$ và $B \subseteq C$ thì $A \subseteq C$

4.34. Chứng minh rằng nếu $X \rightarrow A_1, \ldots, X \rightarrow A_n$ là các phụ thuộc hàm trong một phủ cực tiểu thì lược đồ $XA_i \ldots A_n$ có dạng 3NF.

434. Cho SĐQH W = < R, F >; với R = { C, T, H, P, S, G }; tập phụ thuộc

F = { $C \rightarrow T$, $HP \rightarrow C$, $HT \rightarrow P$, $CS \rightarrow G$, $HS \rightarrow P$}. Hãy phân rã W thành các dạng chuẩn BCNF có nối không mất. Dùng thuật toán Chase để kiểm tra lại tính nối không mất của phân rã.

4.36 Cho SĐQH W = < R, F >; với R = {A,B,C,D,E,G,H,I,J}, tập phụ thuộc hàm là F = {$AB \rightarrow C$, $DE \rightarrow G$, $DE \rightarrow AB$, $HI \rightarrow JDE$}. Hãy phân rã W thành các dạng chuẩn BCNF có bảo toàn phụ thuộc.

4.37. Cho SĐQH W = < R, F >; với R = { A, B,C,D,E,G,I,J,H}, tập phụ thuộc hàm là F ={BC→A, EG→D, EG→BC, IJ→H, IJ→ EG}. Hãy phân rã W thành các sơ đồ con có nối không mất và bảo toàn phụ thuộc. Hãy dùng thuật toán Chase để kiểm tra lại tính nối không mất của phép phân rã.

4.38. Cho SĐQH W = < R, F >; với R = { A, B,C,D,E,G,I,J,H}, tập phụ thuộc hàm F ={AC→B, DG→E, DG→AC, IJ→H, IJ→ EG}. Hãy phân rã W thành các BCNF có nối không mất và bảo toàn phụ thuộc.

Kiểm tra lại bằng thuật toán Chase tính nối không mất của phép tách.

4.39. Cho lược đồ quan hệ về các nhà cung cấp :

S(TÊNS, HANGS, GIÁ, SỐLƯỢNG, Đ/CS, PHONE); với TÊNS là tên nhà cung cấp, HANGS là hàng của các nhà cung cấp, GIÁ là giá của hàng, Đ/C và PHONE là địa chỉ và phone của các nhà cung cấp. SỐLƯỢNG là số lượng hàng của một mặt hàng do một nhà cung cấp nào đó cung cấp.

a) Hãy xét xem các quan hệ trên lược đồ S trên thuộc dạng chuẩn nào?

b) Dùng tập phụ thuộc hàm F trong câu a phân rã W= <R,F> thành các sơ đồ con có bảo toàn phụ thuộc và có nối không mất.

c) Phân rã W thành các sơ đồ con BCNF có nối không mất và bảo toàn phụ thuộc.

4.40. Cho lược đồ R dưới dạng DESIGN VIEW (xem trong Access) như sau:

R: Bảng danh sách cán bộ.

Field	Tên field	Giải thích
1	MA_CB	Mã cán bộ
2	HOTEN_CB	Họ tên các bộ
3	GT	Giới tính
4	NOISINH_CB	Nơi sinh
5	QUE_CB	Quê quán cán bộ
6	CHOHIEN_CB	Chỗ ở hiện nay của CB
7	DT	Dân tộc
8	THANHPHAN_CB	Thành phần gia đình
9	NGAYVĐOAN	Ngày vào đoàn
10	NOIVĐOAN	Nơi vào doàn
11	SHĐLĐ_CB	Số hợp đồng lao động
12	CHUCDANH_CB	Chức danh CB
13	HH_CB	Học hàm CB
14	BACLUONG_CB	Bậc lương CB
15	HSLUONG_CB	Hệ số lương CB
16	TOTNGHIEP_TR	Tốt ngiệp trường
17	NAMTOTNGIEP	Năm tốt nghiệp
18	LOAITOTNGHIEP	Loại tốt nghiệp
19	CHUYEN_NGANH	Chuyên ngành
20	MA_KH	Mã khoa
21	MA_PH	Mã phòng

Như vậy R có 20 thuộc tính. Hãy phân tìm tập phụ thuộc hàm F và phân rã sơ đồ quan hệ tương ứng thành các BCNF có nối không mất tin.

CHƯƠNG 5

KIỂM SOÁT DỮ LIỆU NGỮ NGHĨA VÀ XỬ LÝ VẤN TIN

Một trong những yêu cầu quan trọng của một hệ quản trị CSDL tập trung hoặc phân tán là khả năng kiểm soát dữ liệu ngữ nghĩa và xử lý vấn tin. Tuy nhiên, những khái niệm này cho đến nay vẫn đang được nhiều người quan tâm ở mức lý thuyết, mức khái niệm. Trong chương này chúng ta sẽ xét một số khía cạnh có tính khái niệm của vấn đề. Các phần 5.1, 5.2, 5.3 được dành cho các khái niệm liên quan đến kiểm soát dữ liệu ngữ nghĩa. Các phần còn lại của chương chúng ta sẽ thảo luận về các vấn đề có liên quan đến xử lý vấn tin và tối ưu hoá vấn tin.

Kiểm soát dữ liệu gồm quản lý khung nhìn (view management), kiểm soát tính an toàn, bảo mật (security) và kiểm soát tính toàn vẹn ngữ nghĩa. Các chức năng này phải bảo đảm để những người được phép sử dụng sẽ thực hiện đúng đắn các thao tác trên CSDL nhằm duy trì tính toàn vẹn dữ liệu. Trong CSDL quan hệ, kiểm soát dữ liệu ngữ nghĩa có thể đạt được bằng một phương pháp quy định thống nhất: Hệ thống tự duy trì. Vi phạm một quy tắc do một chương trình của người sử dụng (một tập các thao tác trên CSDL) hệ thống sẽ tự động vô hiệu hóa tác dụng của chương trình đó.

Định nghĩa các quy tắc nhằm kiểm soát các thao tác dữ liệu là một phần trong công việc quản trị CSDL, là một chức năng thường được nhân viên quản trị CSDL thực hiện (DBA). Người quản trị CSDL cũng phải chịu

trách nhiệm áp dụng các chiến lược tổ chức hoạt động, các giải pháp kiểm soát dữ liệu ngữ nghĩa đã được đề xuất cho môi trường tập trung. Trong chương này, chúng ta chỉ xem qua giải pháp cho môi trường tập trung và trình bày những vấn đề sẽ nảy sinh trong một môi trường phân tán.

5.1. QUẢN LÝ KHUNG NHÌN

Một trong những ưu điểm của mô hình quan hệ là nó cung cấp được tính độc lập dữ liệu logic. Trong một hệ thống quan hệ, như trong chương 3 chúng ta đã thấy một khung nhìn là một quan hệ ảo (virtual relation), được định nghĩa như là kết quả vấn tin trên quan hệ cơ sở hoặc quan hệ thực (base relation, real relation) nhưng không được vật chất hóa như một quan hệ cơ sở, nghĩa là không được lưu thực sự trong CSDL. Một khung nhìn là một cửa sổ động (dynamic window) theo nghĩa là nó phản ánh tất cả mọi cập nhật trên CSDL. Một lược đồ ngoài có thể được định nghĩa là một tập các khung nhìn. Bên cạnh việc sử dụng chúng trong các lược đồ ngoài, các khung nhìn còn có tác dụng bảo đảm được tính an toàn dữ liệu bằng một cách rất đơn giản. Nhờ chọn ra một tập con của CSDL, các khung nhìn đã che dấu đi một số dữ liệu. Nếu người sử dụng chỉ được phép truy xuất CSDL qua các khung nhìn, họ không thể nhìn thấy hoặc thao tác trên các dữ liệu ẩn, vì thế chúng được bảo vệ.

Cần chú ý rằng trong môi trường phân tán, một khung nhìn có thể lấy dữ liệu từ các quan hệ phân tán, và việc truy xuất một khung nhìn đòi hỏi phải thực thi câu vấn tin phân tán tương ứng với định nghĩa khung nhìn.

5.1.1. KHUNG NHÌN TRONG QUẢN LÝ TẬP TRUNG

Phần lớn các hệ quản trị CSDL quan hệ đều sử dụng cơ chế khung nhìn. Một khung nhìn là một quan hệ được dẫn xuất từ các quan hệ cơ sở như kết quả của một vấn tin quan hệ. Nó được định nghĩa bằng cách gán tên của khung nhìn cho câu vấn tin truy xuất mô tả nó.

Thí dụ 5.1:

Khung nhìn SYSAN (system analyst - phân tích viên hệ thống) được dẫn xuất từ quan hệ EMP (ENO, ENAME, TITLE) có thể được định nghĩa bởi câu vấn tin SQL sau:

```
CREATE   VIEW SYSAN (ENO, ENAME)
IN       (SELECT   ENO, ENAME
          FROM     EMP
          WHERE    TITLE  =  "Syst. Anal. ")
```

Tác dụng duy nhất của câu lệnh này là lưu định nghĩa khung nhìn vào hồ sơ cơ cấu. Không có thông tin nào khác cần phải ghi nhận. Vì thế kết quả của câu vấn tin định nghĩa khung nhìn (nghĩa là một quan hệ có các thuộc tính ENO và ENAME cho các phân tích viên hệ thống như được trình bày trong hình 5.1) không được tạo ra. Tuy nhiên, khung nhìn SYSAN có thể được xử lý như một quan hệ cơ sở.

SYSAN

ENO	ENAME
E2	M.Smith
E5	B. Casey
E8	J. Jones

Hình 5.1 Quan hệ tương ứng với khung nhìn SYSAN.

Thí dụ 5.2:

Câu vấn tin:

"Tìm tên của các phân tích viên hệ thống , mã số dự án và nhiệm vụ của họ".

Liên quan đến khung nhìn SYSAN và quan hệ ASG (ENO, PNO, RESP, DUR) có thể được diễn tả là

```
SELECT    ENAME, PNO, RESP
FROM      SYSAN, ASG
WHERE     SYSAN. ENO = ASG.ENO
```

Thí dụ 5.3:

Tuy nhiên, câu vấn tin ở thí dụ 5.2 có thể không dùng quan hệ SYSAN.

```
SELECT    ENAME, PNO, RESP
FROM      EMP, ASG
WHERE     EMP. ENO = ASG .ENO
AND       TITLE = "Syst. Anal".
```

Kết quả của câu vấn tin này được trình bày trong hình 5.2.

ENAME	PNO	RESP
M. Smith	P1	Analyst
M. Smith	P2	Analyst
B. Casey	P3	Manager
J. Jones	P4	Manager

Hình 5.2 Kết quả câu vấn tin có dùng đến khung nhìn SYSAN

Câu vấn tin không dùng SYSAN được diễn tả theo các quan hệ cơ sở và vì thế có thể xử lý được vấn tin. Điều quan trọng cần chú ý là xử lý khung nhìn có thể được thực hiện vào lúc biên dịch. Cơ chế khung nhìn cũng có thể được sử dụng nhằm điều chỉnh việc kiểm soát truy xuất để có thể chứa cả các tập con các đối tượng. Khi muốn che dấu dữ liệu không cho bất kỳ người sử dụng nào nhìn thấy, người ta dùng từ khóa USER để nói đến tên người sử dụng khi nhập vào hệ thống.

Thí dụ 5.4:

Khung nhìn ESAME hạn chế người sử dụng chỉ cho truy xuất đến những nhân viên có cùng chức vụ.

```
CREATE    VIEW ESAME
IN        (SELECT    *
          FROM       EMP E1,  EMP E2
          WHERE      E1. TITLE  = E2. TITLE
          AND        E1. ENO  =  USER)
```

Trong định nghĩa khung nhìn ở trên, * đại diện cho "tất cả mọi thuộc tính", và hai biến bộ (E1 và E2) biến thiên trên quan hệ EMP để diễn tả nối của một bộ của EMP (bộ tương ứng với người sử dụng nhập vào hệ thống) với tất cả các bộ của EMP có chức vụ (title) như nhau. Chẳng hạn câu vấn tin dưới đây được đưa ra bởi người sử dụng J. Doe,

```
SELECT    *
FROM ESAME
```

trả về quan hệ của hình 5.3. Chú ý rằng người sử dụng J. Doe cũng xuất hiện trong kết quả. Nếu người sử dụng tạo ra ESAME là một kỹ sư điện như trong trường hợp này thì khung nhìn biểu thị tập tất cả các kỹ sư điện.

ENO	ENAME	TITLE
E1	J. Doe	Elect. Eng.
E2	L. Chu	Elect. Eng.

Hình 5.3 Kết quả của vấn tin trên khung nhìn ESAME

5.1.2. CẬP NHẬT QUA CÁC KHUNG NHÌN

Khung nhìn có thể được định nghĩa bằng các câu vấn tin phức tạp với các phép chọn, chiếu, nối các hàm gộp nhóm, v.v. Tất cả các khung nhìn có thể được truy vấn như một quan hệ cơ sở, nhưng không phải tất cả chúng đều được thay đổi như thế. Cập nhật qua khung nhìn chỉ được xử lý tự động nếu chúng có thể được lan truyền chính xác đến các quan hệ cơ sở. Chúng ta có thể phân các khung nhìn thành hai loại: loại cập nhật được và loại không

cập nhật được. Một khung nhìn gọi là cập nhật được nếu các thao tác cập nhật trên khung nhìn lan truyền chính xác đến các quan hệ cơ sở mà không gây ra sự nhầm lẫn nào. Khung nhìn SYSAN ở trên là cập nhật được; chẳng hạn, chèn một phân tích viên mới < 201, Smith > sẽ được ánh xạ thành thao tác chèn một nhân viên mới < 201, Smith, Syst. Anal. >. Nếu các thuộc tính khác ngoài TITLE ra bị che khuất khỏi khung nhìn, chúng có thể được gán các giá trị null. Thế nhưng khung nhìn sau đây không cập nhật được.

```
CREATE    VIEW EG (ENAME, RESP)
IN        (SELECT    ENAME, RESP
          FROM       EMP , ASG
          WHERE      EMP. ENO  = ASG. ENO)
```

Chẳng hạn như phép xóa bộ < Smith, Syst. Anal. > không thể thực hiện được bởi vì nó không rõ ràng. Xóa Smith trong quan hệ EMP hoặc xóa phân tích viên trong quan hệ ASG đều có nghĩa nhưng hệ thống không biết trường hợp nào là đúng.

Các hệ thống hiện tại hỗ trợ cập nhật qua khung nhìn rất hạn chế. Các khung nhìn chỉ có thể được cập nhật nếu chúng được dẫn xuất từ một quan hệ duy nhất bằng phép chọn hoặc chiếu. Điều này loại trừ các khung nhìn được định nghĩa bởi phép toán nối, phép gộp nhóm v.v. Một điều đáng chú ý là các khung nhìn được dẫn xuất từ phép nối có thể cập nhật được nếu chúng có chứa khóa của các quan hệ cơ sở.

5.1.3. KHUNG NHÌN TRONG CƠ SỞ DỮ LIỆU PHÂN TÁN

Định nghĩa khung nhìn đều giống nhau trong các hệ quản trị CSDL tập trung cũng như phân tán. Tuy nhiên khung nhìn trong các hệ thống phân tán có thể được dẫn xuất từ các quan hệ đã phân mảnh được lưu ở nhiều vị trí khác nhau. Khi một khung nhìn được định nghĩa, tên và câu vấn tin truy xuất của nó sẽ được lưu vào hồ sơ cơ cấu.

Bởi vì khung nhìn có thể được sử dụng làm quan hệ cơ sở trong các ứng dụng, đinh nghĩa của chúng phải được lưu trong thư mục giống như các mô tả của quan hệ cơ sở. Tuỳ thuộc vào mức độ tự trị của vị trí được đưa ra bởi hệ thống, các định nghĩa khung nhìn có thể được tập trung tại một vị trí, được nhân bản một phần hoặc toàn bộ. Trong mỗi trường hợp, thông tin liên kết tên khung nhìn với vị trí định nghĩa của nó phải được nhân bản. Nếu định nghĩa khung nhìn không có tại vị trí định nghĩa của nó phải được nhân bản. Nếu định nghĩa khung nhìn không có tại vị trí đưa ra câu vấn tin thì sẽ phải truy xuất từ xa đến vị trí có định nghĩa khung nhìn đó.

Trong chương 4 chúng ta đã trình bày nhiều cách phân mảnh các quan hệ cơ sở. Định nghĩa phân mảnh thực sự rất giống định nghĩa của các khung nhìn cụ thể. Adiba (Adiba, 1981) đã đề xuất một cơ chế hợp nhất để quản lý khung nhìn và các mảnh. Nó dựa trên nhận xét là các khung nhìn trong một hệ quản trị CSDL phân tán có thể được định nghĩa bằng các quy tắc tương tự như các quy tắc định nghĩa mảnh. Các đối tượng được quản trị viên CSDL tác động đến có thể được xem như một cây phân cấp trong đó các nút lá là các mảnh mà từ đó các quan hệ và các khung nhìn được dẫn xuất. Vì thế DBA có thể tăng cường tính cục bộ tham chiếu bằng cách tạo ra các khung nhìn SYSAN như trong thí dụ 5.1 bằng một mảnh tại vị trí đã cho,

với điều kiện là phần lớn người sử dụng sẽ truy xuất đến khung nhìn này tại cùng vị trí đó.

5.2. AN TOÀN DỮ LIỆU

An toàn dữ liệu (Data security) là một nhiệm vụ quan trọng của hệ thống CSDL nhằm bảo vệ dữ liệu không bị truy xuất "bất hợp pháp". An toàn dữ liệu bao gồm hai vấn đề: bảo vệ dữ liệu và kiểm soát cấp quyền.

Bảo vệ dữ liệu nhằm tránh cho những người "không có phận sự" hiểu được nội dung vật lý của dữ liệu. Chức năng này do hệ thống tập tin đảm trách trong các hệ điều hành tập trung và phân tán. Phương pháp chính là mã hóa dữ liệu, được dùng cho cả các thông tin được lưu trên đĩa lẫn thông tin trao đổi trên mạng. Dữ liệu đã mã hóa chỉ có thể được "giải mã" bởi những người sử dụng được quyền, là những người "biết" được mã. Hai lược đồ chính được sử dụng là chuẩn DES (Data Encryption Standard) và lược đồ mã hóa khóa công cộng (public-key encryption schemes). Trong phần này chúng ta chỉ đề cập đến vấn đề thứ hai vì nó đặc trưng cho CSDL. Trình bày đầy đủ về các kỹ thuật bảo vệ có thể đọc trong (Fernandez, 1981).

Kiểm soát cấp quyền phải đảm bảo rằng chỉ những người được phép mới được thực hiện các thao tác trên CSDL. Những người sử dụng khác nhau có thể có quyền truy xuất đến một lượng lớn dữ liệu dưới sự kiểm soát thống nhất của một hệ thống tập trung hay phân tán. Vì thế DBMS phân tán hay tập trung phải có khả năng hạn chế truy xuất một phần dữ liệu đối với một tập con những người sử dụng. Kiểm soát cấp quyền từ lâu đã được hệ điều hành đảm nhiệm. Các đối tượng có thể được xác định nhờ tên ngoài của chúng. Trong các hệ thống quan hệ, việc cấp quyền có thể được kiểm soát

một cách thống nhất bởi quản trị viên CSDL (DBA) qua các kết cấu (construct) bậc cao. Thí dụ các đối tượng cần kiểm soát có thể được xác định bằng các vị từ giống như cách lượng từ hóa câu vấn tin.

5.2.1. KIỂM SOÁT CẤP QUYỀN TẬP TRUNG

Ba tác nhân chính có liên quan đến việc kiểm soát cấp quyền là: người sử dụng, là người kích hoạt các chương trình ứng dụng; các thao tác được gắn vào ứng dụng; và các đối tượng CSDL sẽ được các thao tác tác động.

Kiểm soát cấp quyền bao gồm việc kiểm tra xem bộ ba (người sử dụng, thao tác, đối tượng) có được phép tiến hành hay không (nghĩa là người sử dụng có thể thực hiện thao tác trên đối tượng đó hay không). Một cấp quyền (authorization) có thể được xem như một bộ ba (người sử dụng, loại thao tác, định nghĩa đối tượng), nó xác định rằng người sử dụng có quyền thực hiện một thao tác thuộc loại này trên một đối tượng. Để có thể kiểm soát hữu hiệu, DBMS đòi hỏi phải định nghĩa người sử dụng, các đối tượng và các quyền.

Khai báo một người sử dụng (hay nhóm người sử dụng) với hệ thống thường được thực hiện bằng một cặp (tên người sử dụng, mật khẩu- user name, password). Tên người sử dụng (user name) sẽ xác định một người sử dụng duy nhất có tên đó trong hệ thống, còn mật khẩu (password), chỉ có người sử dụng biết, dùng để xác nhận người sử dụng. Cả tên và mật khẩu đều phải trình ra khi đăng nhập vào hệ thống. Điều đó ngăn chặn những

người không biết mật khẩu "xâm nhập" vào hệ thống khi họ chỉ biết tên người sử dụng.

Các đối tượng cần bảo vệ là các tập con của CSDL. Hệ thống quan hệ có thể bảo vệ các đối tượng ở nhiều mức chi tiết (độ mịn) hơn so với các hệ thống trước đó. Trong hệ thống tập tin, đơn vị được bảo vệ là tập tin, còn trong các hệ quản trị CSDL mạng và phân cấp đó là kiểu đối tượng (thí dụ mẫu tin, tập tin). Trong hệ thống quan hệ, đối tượng có thể được định nghĩa bằng kiểu của chúng (khung nhìn, quan hệ, bộ, thuộc tính) cũng như bằng nội dung của chúng qua các vị từ chọn. Hơn nữa, cơ chế khung nhìn như đã thảo luận trong phần 5.1 cho phép bảo vệ các đối tượng chỉ đơn giản bằng cách che khuất các tập con của quan hệ (thuộc tính hoặc bộ) đối với những người sử dụng không được phép.

Quyền hạn (right) biểu thị mối liên hệ giữa người sử dụng và một đối tượng ứng với một tập các thao tác cụ thể. Trong các hệ quản trị CSDL dựa trên SQL, một thao tác là một câu lệnh bậc cao như SELECT, INSERT, UPDATE, hoặc DELETE, và các quyền được định nghĩa (trao quyền hoặc thu hồi) bằng các câu lệnh sau:

GRANT < kiểu thao tác > ON < đối tượng > TO < người sử dụng>

REVOKE < kiểu thao tác > FROM < đối tượng > TO < người sử dụng>

Từ khóa public (công cộng) có thể được dùng để chỉ tất cả mọi người sử dụng. Kiểm soát cấp quyền có thể được đặc trưng dựa vào người nào có thể trao quyền (grantor, người trao quyền). Dạng đơn giản nhất, việc kiểm soát được tập trung: một người hoặc một nhóm người (DBA) có tất cả quyền hạn trên các đối tượng CSDL và là người duy nhất được phép sử dụng các câu lệnh GRANT và REVOKE.

Một dạng phức tạp nhưng linh hoạt hơn là kiểm soát phi tập trung người tạo ra đối tượng trở thành chủ nhân (owner) của nó và được trao tất cả

mọi quyền trên đối tượng đó. Đặc biệt, có thêm một loại thao tác GRANT. Trao cho quyền GRANT có nghĩa là tất cả mọi quyền của người thực hiện câu lệnh đó đều được trao cho những người sử dụng đã được mô tả. Vì thế người nhận quyền (grantee) có thể tiếp tục trao quyền trên đối tượng đó. Khó khăn chính trong cách tiếp cận này là quá trình thu hồi quyền phải thực hiện đệ quy. Chẳng hạn nếu A, là người đã trao cho B và B là người đã trao cho C quyền GRANT trên đối tượng O thì khi muốn thu hồi tất cả các quyền của B trên O, thì tất cả các quyền của C cũng phải được thu hồi. Để thực hiện quá trình thu hồi, hệ thống phải duy trì một cây phân cấp chứa các hành động trao quyền cho mỗi đối tượng, trong đó chủ nhân của đối tượng chính là gốc.

Quyền hạn của các chủ thể (subject) trên các đối tượng được ghi nhận trong hồ sơ cơ cấu (thư mục) dưới dạng các qui tắc cấp quyền. Có nhiều cách để lưu các cấp quyền này. Cách thuận tiện nhất là xem tất cả các quyền như một ma trận cấp quyền (authorization matrix), trong đó hàng định nghĩa một chủ thể, cột định nghĩa đối tượng, và một mục ghi trong ma trận (cho cặp < chủ thể, đối tượng>), là các thao tác được phép. Những thao tác này được xác định bằng kiểu thao tác (thí dụ SELECT, UPDATE). Thông thường kèm với mỗi kiểu thao tác có một vị từ hạn chế thêm khả năng truy xuất đến đối tượng. Việc chọn này được cung cấp với các đối tượng là các quan hệ cơ sở, không dùng cho các khung nhìn. Chẳng hạn, một thao tác được phép cho cặp

< Jones, quan hệ EMP> có thể là:

SELECT WHERE TITLE = "Syst. Anal."

nó cho phép Jones chỉ được phép truy xuất đến các bộ của các phân tích viên hệ thống. Hình 5.4 trình bày một thí dụ mẫu về ma trận cấp quyền trong đó

đối tượng là các quan hệ (EMP và ASG) hoặc Công ty là các thuộc tính (ENAME).

	EMP	ENAME	ASG
Casey	UPDATE	UPDATE	UPDATE
Jones	SELECT	SELECT	SELECT WHERE RESP ≠ "Manager"
Smith	NONE	SELECT	NONE

Hình 5.4 Thí dụ về ma trận cấp quyền

Ma trận cấp quyền có thể được lưu theo ba cách: theo cột, theo hàng hoặc theo phần tử. Khi ma trận được lưu theo hàng, mỗi chủ thể được liên kết với một danh sách các đối tượng được phép truy xuất cùng với các quyền truy xuất tương ứng. Lối tiếp cận này cho phép duy trì các cấp quyền một cách hiệu quả, bởi vì tất cả các quyền của một người sử dụng khi nhập vào hệ thống được cất cùng nhau trong hồ sơ cá nhân (profile) của người sử dụng. Tuy nhiên việc thao tác trên các quyền truy xuất (thí dụ khi cần cho phép mọi người truy xuất đến đối tượng) sẽ không hiệu quả bởi vì phải truy xuất tất cả mọi hồ sơ cá nhân. Khi ma trận được lưu theo cột, mỗi đối tượng được liên kết với một danh sách những người sử dụng được phép truy xuất. Ưu điểm và khuyết điểm của phương pháp này là khuyết điểm và ưu điểm của phương pháp lưu theo hàng.

Các ưu điểm của cả hai cách tiếp cận trên được tổ hợp lại trong cách tiếp cận thứ ba, trong đó ma trận được lưu theo phần tử, nghĩa là theo quan hệ (chủ thể, đối tượng, quyền). Quan hệ này có thể có chỉ mục trên cả chủ

thể và đối tượng, qua đó cho phép truy xuất nhanh chóng đến các quyền của mỗi chủ thể và mỗi đối tượng.

5.2.2. KIỂM SOÁT CẤP QUYỀN PHÂN TÁN

Các vấn đề mới của kiểm soát cấp quyền trong môi trường phân tán có nguồn gốc từ sự kiện là các đối tượng và các chủ thể đều phân tán. Những vấn đề này bao gồm: cấp quyền cho người sử dụng ở xa, quản lý các quy tắc cấp quyền phân tán và việc xử lý các khung nhìn và các nhóm người sử dụng. Có hai giải pháp cho vấn đề này:

1. Thông tin xác nhận người sử dụng (tên và mật khẩu) được nhân bản tại tất cả các vị trí trong mạng. Các chương trình cục bộ cũng phải chỉ rõ tên và mật khẩu của người sử dụng.

2. Tất cả các vị trí trong hệ thống phân tán cũng nhận diện và xác nhận nhau tương tự như cách người sử dụng thực hiện. Giao tiếp giữa các vị trí được bảo vệ bằng cách sử dụng mật khẩu của vị trí. Một khi vị trí khởi hoạt đã được xác nhận thì không cần phải xác nhận người sử dụng của chúng nữa.

Các quy tắc cấp quyền phân tán được diễn tả theo cùng phương thức như trong hệ tập trung. Giống như các định nghĩa khung nhìn, chúng phải được lưu vào trong hồ sơ cơ cấu. Chúng có thể được nhân bản hoàn toàn tại mỗi vị trí hoặc lưu tại các vị trí của các đối tượng cần truy xuất. Ưu điểm chính của lối tiếp cận nhân bản hoàn toàn là việc cấp quyền có thể được xử lý bằng kỹ thuật hiệu chỉnh vấn tin vào lúc biên dịch. Tuy nhiên việc quản lý thư mục sẽ tốn kém hơn do việc nhân bản này. Giải pháp thứ hai tốt hơn trong trường hợp tính chất cục bộ của tham chiếu rất cao. Tuy nhiên việc cấp quyền phân tán không thể kiểm soát được vào lúc biên dịch.

Khung nhìn có thể được xem như các đối tượng của cơ chế cấp quyền. Khung nhìn là những đối tượng phức tạp, nghĩa là nó được cấu tạo bởi những đối tượng cơ sở khác. Vì thế trao quyền truy xuất đến một khung nhìn được dịch thành trao quyền truy xuất đến các đối tượng cơ sở. Nếu định nghĩa khung nhìn và các quy tắc cấp quyền được nhân bản hoàn toàn (như trong nhiều hệ thống) thì việc biên dịch này khá đơn giản và có thể được thực hiện tại chỗ.

Nhóm các người sử dụng dùng cấp quyền chung làm đơn giản công việc quản lý CSDL phân tán. Trong các DBMS tập trung, khái niệm "mọi người sử dụng" có thể được xem là nhóm công cộng (public). Trong môi trường phân tán, nhóm công cộng biểu thị cho tất cả mọi người sử dụng của hệ thống. Tuy nhiên, người ta thường đưa ra một mức trung gian nhằm mô tả nhóm công cộng tại một vị trí cụ thể, được biểu thị là public@ site. Nhóm công cộng là một nhóm đặc biệt. Các nhóm cụ thể hơn có thể được định nghĩa bằng lệnh.

DEFINE GROUP < GROUP _ id > AS <danh sách các id chủ thể >

Quản lý các nhóm trong môi trường phân tán đặt ra một số vấn đề phải giải quyết bởi vì các chủ thể của một nhóm có thể cư ngụ tại nhiều vị trí khác nhau và quyền truy xuất đến một đối tượng có thể được trao cho nhiều nhóm, mà bản thân chúng lại phân tán. Nếu thông tin của nhóm và các quy tắc cấp quyền được nhân bản hoàn toàn tại tất cả mọi vị trí thì việc duy trì quyền truy xuất tương tự như trong hệ thống tập trung. Tuy nhiên việc duy trì các bản sao này hết sức tốn kém. Bài toán sẽ khó hơn khi phải duy trì hoạt động tự trị vị trí (với việc kiểm soát phi tập trung).

5.3. KIỂM SOÁT TÍNH TOÀN VẸN NGỮ NGHĨA

Một vấn đề quan trọng và khó khăn cho một hệ CSDL là bảo đảm tính nhất quán CSDL (database consistency). Một trạng thái CSDL được gọi là nhất quán nếu CSDL thoả một tập các ràng buộc, được gọi là ràng buộc toàn vẹn ngữ nghĩa (semantic integrity constraint). Duy trì CSDL nhất quán đòi hỏi phải sử dụng đến nhiều cơ chế khác nhau như điều khiển hoạt động đồng thời, độ tin cậy, bảo vệ và kiểm soát tính toàn vẹn ngữ nghĩa.

5.3.1. KIỂM SOÁT TOÀN VẸN NGỮ NGHĨA TẬP TRUNG

Một tiểu hệ thống kiểm soát toàn vẹn ngữ nghĩa có hai thành phần chính: một ngôn ngữ cho phép diễn tả và thao tác các phán đoán toàn vẹn, và một định chế chịu trách nhiệm thực hiện các hành động cụ thể nhằm cưỡng chế tính toàn vẹn khi có cập nhật.

5.3.1.1. Khái niệm các ràng buộc toàn vẹn

Ràng buộc toàn vẹn phải do quản trị viên CSDL thiết lập nhờ vào một ngôn ngữ cấp cao. Trong phần này chúng ta sẽ minh hoạ một ngôn ngữ khai báo để đặc tả ràng buộc toàn vẹn. Ngôn ngữ này rất giống ngôn ngữ chuẩn SAL (ANSI, 1986), nhưng tổng quát hơn. Nó cho phép chúng ta đặc tả, ghi nhận hoặc tháo bỏ ràng buộc toàn vẹn. Những ràng buộc này được định nghĩa vào lúc tạo quan hệ hoặc vào một lúc nào đó, ngay cả khi quan

hệ đã chứa dữ liệu. Tuy nhiên trong cả hai trường hợp, cú pháp hầu như giống nhau.

Trong các hệ CSDL quan hệ, ràng buộc toàn vẹn được định nghĩa như là các phán đoán. Một phán đoán (assertion) là một biểu thức đặc biệt được viết bằng phép tính quan hệ bộ, trong đó mỗi biến được lượng từ hóa phổ dụng hoặc tồn tại. Vì vậy một phán đoán có thể được xem như là một lượng từ hóa (qualification) câu vấn tin, mang giá trị đúng hoặc sai cho mỗi bộ trong tích Descartes của các quan hệ được xác định bởi các biến bộ. Chúng ta có thể phân biệt ba loại ràng buộc toàn vẹn: ràng buộc *tiền định*, ràng buộc *tiền dịch* hoặc ràng buộc *tổng quát*.

Qua CSDL dưới đây, chúng ta sẽ đưa ra một số thí dụ về các loại ràng buộc toàn vẹn.

EMP (ENO, ENAME, TITLE)

PROJ (PNO, PNAME, BUDGET)

ASG (ENO, PNO, RESP, DUR)

Ràng buộc tiền định (predefined constraint) dựa trên các từ khóa đơn giản, nhờ đó có thể diễn tả chính xác các ràng buộc thông dụng trong mô hình quan hệ, chẳng hạn như thuộc tính không null, khoá duy nhất, khóa ngoại hoặc phụ thuộc hàm. Các thí dụ 5.5 đến 5.8 minh hoạ cho các ràng buộc tiền định.

Thí dụ 5.5:

Thuộc tính không nhận giá trị null

Mã số nhân viên trong quan hệ EMP không được nhận giá trị null.

ENO NOT NULL IN EMP

Thí dụ 5.6:

Khóa duy nhất

Cặp (ENO, PNO) là khóa duy nhất của quan hệ ASG.

(ENO, PNO) UNIQUE IN ASG

Thí dụ 5.7:

Khóa ngoại

Mã số dự án PNO trong quan hệ ASG là khóa ngoại tương ứng với khóa chính PNO của quan hệ PROJ. Nói cách khác, một dự án được tham chiếu trong quan hệ ASG phải tồn tại trong quan hệ PROJ.

PNO IN ASG REFERENCES PNO IN PROJ

Thí dụ 5.8:

Phụ thuộc hàm

Mã số nhân viên xác định hàm tên nhân viên.

ENO IN EMP DETERMINES ENAME

Ràng buộc tiền dịch (precompiled constraint) diễn tả các điều kiện đầu phải được thoả bởi tất cả các bộ trong một quan hệ đối với một kiểu cập nhật đã cho. Kiểu cập nhật, có thể là INSERT, DELETE hoặc MODIFY, cho phép giới hạn công việc kiểm soát ràng buộc. Để xác định các bộ cần cập nhật trong định nghĩa ràng buộc, chúng ta đưa ra hai biến NEW và OLD. Tương ứng chúng biến thiên trên các bộ mới (được chèn vào) và các bộ cũ (được xóa khỏi). Ràng buộc tiền dịch có thể được diễn tả với câu lệnh CHECK của SAL được bổ sung thêm khả năng đặc tả kiểu cập nhật. Cú pháp của câu lệnh CHECK là:

CHECK ON < tên quan hệ> WHEN <kiểu cập nhật>
(<lượng từ hóa trên tên quan hệ>)

Một số thí dụ về các ràng buộc tiền dịch:

Thí dụ 5.9:

Ràng buộc miền

Ngân sách của một dự án trong khoảng từ 500K đến 1000K.

CHECK ON PROJ (BUDGET ≥ 500000 AND BUDGET ≤ 1000000).

Thí dụ 5.10:

Ràng buộc miền khi xóa

Chỉ có các bộ với ngân sách là 0 mới được xóa.

CHECK ON PROJ WHEN DELETE (BUDGET = 0)

Thí dụ 5.11:

Ràng buộc di chuyển

Ngân sách của dự án chỉ được tăng.

CHECK ON PROJ (NEW.BUDGET > OLD. BUDGET
AND NEW. PNO = OLD.PNO)

Ràng buộc tổng quát (general constraint) là các công thức của phép tính quan hệ bộ trong đó tất cả các biến đều được lượng từ hóa. Hệ thống CSDL phải bảo đảm rằng những công thức này phải luôn đúng. Ràng buộc tổng quát chuẩn xác hơn ràng buộc tiền dịch bởi vì chúng có thể liên quan đến

nhiều quan hệ. Chẳng hạn chúng ta phải dùng ít nhất ba ràng buộc tiền dịch để diễn tả một ràng buộc tổng quát trên ba quan hệ. Một ràng buộc tổng quát có thể được diễn tả với cú pháp sau:

CHECK ON danh sách <tên biến>: <tên quan hệ>, (<lượng từ hóa>)

Dưới đây là những thí dụ về ràng buộc tổng quát.

Thí dụ 5.12:

Phụ thuộc hàm

Ràng buộc của Thí dụ 6.8 cũng có thể diễn tả

CHECK ON e1: EMP, e2: EMP

(e1. ENAME = e2. ENAME IF e1. ENO = e2. ENO)

Thí dụ 5.13:

Ràng buộc có kèm hàm gộp nhóm

Tổng thời gian của các nhân viên trong dự án CAD/CAM phải nhỏ hơn 100.

CHECK ON g: ASG, J: PROJ (SUM (g.DUR WHERE g.PNO=J.PNO) < 100

IF J. PNAME = "CAD/CAM"

5.3.1.2 Cưỡng chế thi hành ràng buộc

Cưỡng chế thi hành ràng buộc toàn vẹn bao gồm việc phế bỏ các chương trình cập nhật vi phạm một số ràng buộc nào đó. Một ràng buộc bị vi phạm khi ở trạng thái CSDL mới được sinh ra do hành động cập nhật, các phán đoán ràng buộc trở thành sai. Khó khăn chính khi thiết kế một tiểu hệ thống kiểm soát ràng buộc là tìm ra một thuật toán cưỡng chế hiệu quả. Hai phương pháp cơ bản cho phép phế bỏ các cập nhật sinh mâu thuẫn. Phương pháp đầu tiên dựa vào việc phát hiện mâu thuẫn (không nhất quán). Một thao tác cập nhật u được thực thi sẽ biến đổi CSDL từ trạng thái D sang trạng thái D_u. Thuật toán cưỡng chế phải xác nhận lại rằng tất cả các ràng buộc liên đới vẫn đúng trong D_u bằng cách áp dụng các kiểm tra có được từ những ràng buộc này. Nếu trạng thái D_u không nhất quán, DBMS sẽ cố gắng chuyển sang một trạng thái D_u khác bằng cách hiệu chỉnh lại D_u qua một số hành động "bù trừ", hoặc phải khôi phục lại trạng thái D. Bởi vì những kiểm tra này được áp dụng sau khi trạng thái của CSDL đã thay đổi, chúng được gọi là kiểm tra sau (posttest). Cách tiếp cận này sẽ không hiệu quả nếu hệ thống phải hồi lại rất nhiều thao tác khi ràng buộc bị vi phạm.

Phương pháp thứ hai dựa trên việc ngăn chặn mâu thuẫn. Một thao tác cập nhật chỉ được thực hiện nếu nó chuyển CSDL sang một trạng thái nhất quán khác. Các bộ cần cập nhật đã có sẵn (trong trường hợp chèn) hoặc phải truy xuất trong CSDL (trường hợp xóa hoặc hiệu chỉnh). Thuật toán cưỡng chế xác nhận rằng tất cả các ràng buộc có liên đới đều đúng sau khi cập nhật các bộ. Nói chung nó được thực hiện bằng cách áp dụng các kiểm tra có được từ các ràng buộc cho các bộ. Như thế các kiểm tra được áp dụng trước khi trạng thái của CSDL bị thay đổi, do vậy chúng được gọi là các kiểm tra trước (pretest). Phương pháp ngăn chặn hiệu quả hơn phương pháp

phát hiện bởi vì chúng ta không bao giờ phải hồi lại các thao tác cập nhật do ràng buộc bị vi phạm.

Thuật toán hiệu chỉnh vấn tin là một thí dụ về phương pháp ngăn chặn, có hiệu quả đặc biệt trong việc cưỡng chế các ràng buộc miền biến thiên. Nó đưa thêm lượng từ hóa phán đoán vào lượng từ hóa vấn tin bằng toán tử AND, vì thế câu vấn tin được hiệu chỉnh có thể được cưỡng chế toàn vẹn.

Thí dụ 5.14:

Câu vấn tin làm tăng ngân sách dự án CAD/CAM lên 10% :

```
UPDATE    PROJ
SET       BUDGET = BUDGET * 1. 1
WHERE     PNAME  = "CAD/CAM"
```

sẽ được biến đổi thành câu vấn tin sau, có thêm điều kiện nhằm cưỡng chế ràng buộc miền giá trị như trong thí dụ 5.9.

```
UPDATE    PROJ
    SET       BUDGET  = BUDGET * 1.1
    WHERE     PNAME  = "CAD/CAM"
    AND       NEW. BUDGET ≥ 500000
    AND       NEW. BUDGET ≤ 1000000
```

Thuật toán hiệu chỉnh vấn tin đơn giản, nó tạo ra các kiểm tra trước vào lúc thực thi bằng cách lấy hội các vị từ phán đoán với các vị từ cập nhật của mỗi chỉ thị của giao dịch. Tuy nhiên, thuật toán chỉ được áp dụng cho các công thức phép tính bộ và có thể được đặc tả như sau:

Xét phán đoán $(\forall x \in R)\ F(x)$ trong đó R là biểu thức quan hệ bộ, x là biến bộ. Thao tác cập nhật của R có thể được viết là $(\forall x \in R)(Q(x) \Rightarrow$

update(x); trong đó Q là biểu thức phép tính bộ có biến là x. Nói đơn giản, hiệu chỉnh vấn tin cần sinh ra thao tác cập nhật $(\forall x \in R)(Q(x)$ and $F(x) \Rightarrow$ update(x). Vì thế x cần phải được lượng từ hóa phổ dụng.

Thí dụ 5.15:

Phán đoán cho khóa ngoại của thí dụ 5.7 khi được viết là

$\forall g \in ASG, \exists j \in PROJ: g.PNO = j.PNO$

Không thể được xử lý bởi phép hiệu chỉnh vấn tin bởi vì biến j không được lượng từ hóa phổ dụng.

Để có thể xử lý được các phán đoán tổng quát hơn, các hành động kiểm tra trước có thể được xây dựng vào lúc định nghĩa phán đoán, và được thực hiện vào lúc thực thi khi có cập nhật xảy ra. Phương pháp này chấp nhận các phán đoán đa quan hệ, các phán đoán đơn biến, có thể có cả các phép gộp nhóm. Nguyên tắc là thay thế các biến bộ trong phán đoán bằng các hằng từ một bộ được cập nhật. Mặc dù đây là một đóng góp quan trọng về mặt lý thuyết, song phương pháp này rất khó sử dụng trong thực tế do những hạn chế trên các cập nhật.

Định nghĩa các phán đoán biên dịch dựa trên khái niệm quan hệ vi phân (differential relation). Gọi u là một cập nhật trên quan hệ R. R^+ và R^- là các quan hệ vi phân của R do u, với R^+ chứa các bộ được chèn vào R và R^- chứa các bộ bị xóa khỏi R. Nếu u là thao tác chèn, R^- sẽ rỗng. Nếu u là thao tác xóa, R^+ rỗng. Cuối cùng nếu u là thao tác sửa đổi, quan hệ R sau khi sửa đổi là $R^+ \cup (R - R^-)$.

Một phán đoán biên dịch là bộ ba (R, T, C) trong đó R là một quan hệ, T là kiểu cập nhật và C là một phán đoán biến thiên trên các quan hệ vi phân có mặt trong một cập nhật kiểu T. Khi một ràng buộc I được định nghĩa, một

tập phán đoán biên dịch có thể được sinh ra cho các quan hệ được I tác động. Mỗi khi một quan hệ có trong I được cập nhật bởi một chương trình u, các phán đoán biên dịch cần phải được thẩm tra chỉ là các phán đoán được định nghĩa trên I cho kiểu cập nhật của u. Cách tiếp cận này có hai ưu điểm về hiệu năng. Thứ nhất, số lượng các phán đoán cần cưỡng chế được hạ thấp tối đa bởi vì chỉ có các phán đoán thuộc kiểu u cần phải được thẩm tra. Thứ hai, chi phí cưỡng chế thi hành một phán đoán biên dịch rõ ràng nhỏ hơn chi phí cưỡng chế I bởi vì nói chung các quan hệ vi phân nhỏ hơn nhiều so với các quan hệ cơ sở.

Phán đoán biên dịch có thể thu được bằng cách áp dụng các quy tắc biến đổi cho phán đoán gốc. Những quy tắc này dựa trên việc phân tích ngữ nghĩa của phán đoán và các hoán vị lượng từ. Chúng cho phép thay các quan hệ cơ sở bằng các quan hệ vi phân. Bởi vì các phán đoán biên dịch đơn giản hơn các phán đoán gốc, quá trình tạo ra chúng được gọi là quá trình đơn giản hóa (simplification).

Thí dụ 5.16:

Xét biểu thức ràng buộc khóa ngoại trong thí dụ 5.15. Phán đoán biên dịch đi kèm với ràng buộc này là:

(ASG, INSERT, C_1), (PROJ, DELETE, C_2) và (PROJ, MODIFY, C_3)

Trong đó C_1 là:

$$\forall NEW \in ASG^+, \exists j \in PROJ: NEW.PNO = j.PNO$$

C_2 là:

$$\forall g \in ASG, \forall OLD \in PROJ^-, g.PNO \neq OLD.PNO$$

và C_3 là:

$$\forall g \in ASG, \forall OLD \in PROJ^-, \exists NEW \in PROJ^+:$$

g.PNO ≠ OLD.PNO OR OLD.PNO = NEW.PNO

Ưu điểm của các phán đoán biên dịch là hiển nhiên. Chẳng hạn thao tác xóa trên quan hệ ASG không gây ra bất kỳ một kiểm tra nào.

Thuật toán cưỡng chế được mô tả trong [Simon and Valduriez, 1984] sử dụng các phán đoán biên dịch và được chuyên biệt hóa theo từng lớp phán đoán. Chúng ta phân biệt ba lớp phán đoán: phán đoán đơn quan hệ, đa quan hệ và các phán đoán có các hàm gộp nhóm.

Bây giờ chúng ta hãy tóm tắt thuật toán cưỡng chế. Chúng ta biết rằng một chương trình cập nhật sẽ hiệu chỉnh tất cả các bộ của quan hệ R. Thuật toán được thực hiện qua hai bước.

Bước 1: tạo ra các quan hệ vi phân R^+ và R^- từ R.

Bước 2: truy xuất các bộ trong R^+ và R^-, lấy những bộ không thoả các phán đoán biên dịch. Nếu không lấy được bộ nào, phán đoán là tốt.

Thí dụ 5.17:

Giả sử có một thao tác xóa trên PROJ. Cưỡng chế (PROJ,DELETE,C_2) tạo ra các câu lệnh sau:

result ← truy xuất tất cả các bộ của $PROJ^-$ với phủ định của (C_2) đúng.

Nếu result rỗng, phán đoán này đã được xác nhận bởi phép cập nhật.

5.3.2. KIỂM SOÁT TOÀN VẸN NGỮ NGHĨA PHÂN TÁN

Trong phần này chúng ta sẽ trình bày các thuật toán nhằm bảo đảm tính toàn vẹn ngữ nghĩa của các CSDL phân tán, chúng là những mở rộng của các phương pháp đơn giản hóa đã được thảo luận trước đây.

Trong những đoạn sau, chúng ta giả thiết là có tính tự trị vị trí, nghĩa là mỗi vị trí có thể xử lý các câu vấn tin cục bộ và thực hiện việc kiểm soát dữ liệu như một hệ quản trị CSDL tập trung.

Hai vấn đề chính của việc thiết kế một tiểu hệ thống kiểm soát toàn vẹn trong một DBMS phân tán là: *định nghĩa, lưu trữ các phán đoán*, và *cưỡng chế thi hành những phán đoán này.*

5.3.2.1 Định nghĩa các phán đoán toàn vẹn phân tán

Giả thiết rằng một phán đoán toàn vẹn được diễn tả bằng phép tính quan hệ bộ. *Mỗi phán đoán được xem là một lượng từ hóa vấn tin một vấn tin theo vị từ).* Các phán đoán có thể liên quan đến dữ liệu được lưu ở nhiều vị trí khác nhau. Phân loại các phán đoán toàn vẹn :

1- Phán đoán riêng: Là các phán đoán đơn biến, đơn quan hệ. Chúng chỉ đề cập đến các bộ được cập nhật, độc lập với phần còn lại của CSDL. Chẳng hạn ràng buộc miền giá trị của thí dụ 5.9 là một phán đoán riêng.

2- Phán đoán hướng tập hợp: Bao gồm các ràng buộc đa biến đơn quan hệ như phụ thuộc hàm (thí dụ 5.8) và đa biến đa quan hệ như các ràng buộc khóa ngoại (thí dụ 5.7).

3. Phán đoán có các hàm gộp: đòi hỏi phải được xử lý đặc biệt do chi phí ước lượng hàm gộp. Phán đoán trong thí dụ 5.13 là một đại diện cho lớp phán đoán này.

Định nghĩa một phán đoán toàn vẹn có thể được bắt đầu tại một vị trí có lưu các quan hệ có mặt trong phán đoán. Cần nhớ rằng các quan hệ có thể bị phân mảnh. Một vị từ phân mảnh là một trường hợp đặc biệt của phán đoán thuộc lớp 1. Các mảnh khác nhau của cùng một quan hệ có thể nằm tại

nhiều vị trí khác nhau. Vì thế việc định nghĩa một phán đoán toàn vẹn trở thành một thao tác phân tán và được thực hiện qua hai bước. Bước đầu tiên là biến đổi các phán đoán ở cấp cao thành các phán đoán biên dịch nhờ các kỹ thuật đã trình bày trong phần trước. Bước kế tiếp là lưu các phán đoán này tùy theo lớp phán đoán. Các phán đoán thuộc lớp 3 được xử lý giống như các phán đoán thuộc lớp 1 hoặc 2, tùy thuộc vào đặc tính của chúng là riêng hay hướng tập hợp.

PHÁN ĐOÁN RIÊNG

Định nghĩa phán đoán được gửi đến tất cả mọi vị trí lưu trữ các mảnh của quan hệ có mặt trong phán đoán. Phán đoán phải tương thích với dữ liệu của quan hệ tại mỗi vị trí. Tính tương thích có thể được kiểm tra ở hai cấp: cấp vị từ và cấp dữ liệu. Trước tiên, tương thích có thể được xác nhận bằng cách so sánh vị từ phán đoán với vị từ mảnh. Một phán đoán C được xem là không tương thích với vị từ mảnh p nếu "C đúng" dẫn đến "P sai", và ngược lại thì được xem là tương thích. Nếu có tình huống không tương thích tại một trong các vị trí, định nghĩa phán đoán phải bị phế bỏ ở mức toàn cục bởi vì các bộ của mảnh đó không thoả ràng buộc toàn vẹn này. Thứ hai là nếu có tương thích vị từ, phán đoán sẽ được thẩm tra ứng với thể hiện của mảnh. Nếu thể hiện đó không thoả phán đoán thì nó cũng bị phế bỏ ở mức toàn cục. Nếu có tương thích, phán đoán sẽ được lưu lại tại mỗi vị trí. Chú ý rằng việc thẩm tra tính tương thích được thực hiện cho các phán đoán biên dịch với kiểu cập nhật là "chèn" (các bộ trong các mảnh xem như là "được chèn vào").

Thí dụ 5.18:

Xét quan hệ EMP, được phân mảnh ngang cho ba vị trí bằng các vị từ

P_1: $0 \leq ENO < "E4"$

P_2: $"E3" \leq ENO \leq "E6"$

P_3: $ENO > "E6"$

và phán đoán miền biến thiên C: ENO < "E3". Phán đoán C tương thích với P_1 (nếu C đúng thì P_1 đúng), nhưng không tương thích với P_2, P_3 (nếu C đúng thì P_2, P_3 sai). Vì thế phán đoán C buộc phải bị phế bỏ bởi vì các bộ tại vị trí 2, 3 không thể thoả C, vì thế quan hệ EMP không thoả C.

PHÁN ĐOÁN HƯỚNG TẬP HỢP

Phán đoán hướng tập hợp thuộc loại đa biến; nghĩa là chúng chứa vị từ nối. Mặc dù vị từ phán đoán có thể thuộc loại đa quan hệ nhưng một phán đoán biên dịch chỉ được liên kết với một quan hệ. Vì thế định nghĩa phán đoán có thể được gửi đến tất cả các vị trí có chứa các mảnh được các biến này tham chiếu. Việc thẩm tra tính tương thích cũng gồm cả các mảnh của quan hệ được sử dụng trong vị từ nối. Tương thích vị từ không có tác dụng gì ở đây vì chúng ta không thể suy ra rằng một vị từ mảnh P sai nếu phán đoán C (dựa trên vị từ nối) đúng. Vì thế chúng ta cần phải thẩm tra C theo dữ liệu. Điều này đòi hỏi phải nối mỗi mảnh của quan hệ R với tất cả các mảnh của quan hệ S, là hai quan hệ có trong vị từ nối. Như thế rõ ràng là chi phí sẽ rất cao, và như mọi phép nối khác, nó cần được tối ưu hóa bằng thể xử lý vấn tin phân tán. Ba tình huống sau có thể xảy ra được liệt kê theo mức chi phí.

1- Phân mảnh của R được dẫn xuất (xem chương 4) từ phân mảnh của S dựa vào một nối nửa trên thuộc tính được dùng trong vị từ nối.

2- S được phân mảnh trên thuộc tính nối.

3- S không được phân mảnh trên thuộc tính nối.

Trong trường hợp đầu tiên, việc kiểm tra tương thích có chi phí thấp bởi vì bộ của S đối sánh được với một bộ của R sẽ ở cùng một vị trí. Trong trường hợp thứ hai, mỗi bộ của R phải được so sánh với tối đa một mảnh của S, bởi vì giá trị thuộc tính nối của bộ thuộc R có thể được dùng để tìm vị trí mảnh tương ứng của S. Trong trường hợp thứ ba, mỗi bộ của R phải được so sánh với tất cả các mảnh của S. Nếu tất cả các bộ của R đều bảo đảm tương thích thì phán đoán sẽ được lưu tại mỗi vị trí.

Thí dụ 5.19:

Xét phán đoán biên dịch hướng tập hợp (ASG, INSERT, C_1) được định nghĩa trong thí dụ 5.16, trong đó C_1 là:

$$\forall NEW \in ASG^+, \exists j \in PROJ: NEW.PNO = j.PNO$$

Chúng ta xét ba trường hợp sau:

1. ASG được phân mảnh theo vị từ

$$ASG\ X_{PNO}\ PROJ_i$$

Trong đó $PROJ_i$ là một mảnh của quan hệ PROJ. Trong trường hợp này, mỗi bộ NEW của ASG đã được đặt tại cùng vị trí với bộ j sao cho NEW.PNO = j. PNO. bởi vì vị từ phân mảnh giống với vị từ của C_1, việc thẩm tra tương thích không mất chi phí truyền dữ liệu.

2- PROJ được phân mảnh ngang theo hai vị từ

P_1: PNO < P3 $\qquad$ P_2: PNO $\geq$ P3

Trong trường hợp này, mỗi bộ NEW của ASG được so sánh với mảnh $PROJ_1$ nếu NEW. PNO < P3, hoặc với mảnh $PROJ_2$ nếu NEW.PNO ≥ P3.

3. PROJ được phân mảnh ngang theo hai vị từ

P_1: PNAME = "CAD/CAM"

P_2: PNAME ≠ "CAD/CAM"

Trong trường hợp này, mỗi bộ của ASG phải được so với cả hai mảnh $PROJ_1$ và $PROJ_2$.

5.3.2.2 Cưỡng chế thi hành các phán đoán toàn vẹn phân tán

Cưỡng chế thi hành các phán đoán toàn vẹn phân tán rõ ràng phức tạp hơn so với các DBMS tập trung. Vấn đề chính là quyết định xem vị trí nào sẽ thực hiện cưỡng chế. Lựa chọn phụ thuộc vào lớp phán đoán, kiểu cập nhật, và bản chất của vị trí đưa ra yêu cầu cập nhật (được gọi là vị trí vấn tin chính). Vị trí này có thể có hoặc không có liên quan với quan hệ cần cập nhật hoặc một số quan hệ có mặt trong phán đoán toàn vẹn. Tham số quan trọng cần xem xét là chi phí truyền dữ liệu, kể cả các thông báo, từ vị trí này đến vị trí khác. Bây giờ chúng ta sẽ phân tích một số chiến lược khác nhau theo những tiêu chuẩn này.

PHÁN ĐOÁN RIÊNG

Chúng ta xem xét hai trường hợp.

Nếu cập nhật là yêu cầu chèn, tất cả các bộ cần chèn đã được người sử dụng cung cấp. Trong trường hợp này, phán đoán riêng có thể được cưỡng

chế thi hành tại vị trí đưa ra cập nhật. Nếu đây là xóa hoặc sửa đổi, nó sẽ được gửi đến các vị trí có lưu quan hệ cần cập nhật. Thể xử lý vấn tin sẽ cho thực hiện lượng từ hóa cập nhật cho mỗi mảnh. Các bộ được tạo ra tại mỗi vị trí sẽ được tổ hợp lại thành một quan hệ tạm thời trong trường hợp câu lệnh xóa, hoặc hai trong trường hợp hiệu chỉnh (nghĩa là R^+ và R^-). Mỗi vị trí có mặt trong cập nhật đều thẩm tra các phán đoán có liên quan đến vị trí đó (thí dụ ràng buộc miền khi xóa).

PHÁN ĐOÁN HƯỚNG TẬP HỢP

Trước tiên chúng ta nghiên cứu các ràng buộc đơn quan hệ qua một thí dụ. Xét phụ thuộc hàm của thí dụ 5.8. Phán đoán biên dịch kèm với kiểu cập nhật INSERT là

(EMP, INSERT, C)

Trong đó C là

$(\forall E \in EMP)\ (\forall NEW1 \in EMP)\ (\forall NEW2 \in EMP)$ (1)

(NEW 1. ENO = c.ENO => NEW1. ENAME = e.ENAME) ^

(2)

(NEW1. ENO = NEW2.ENO => NEW 1. ENAME = NEW2. ENAME) (3)

Dòng thứ hai trong định nghĩa của C kiểm tra ràng buộc giữa các bộ được chèn (NEW 1) và các bộ hiện có (c), còn dòng thứ ba kiểm tra giữa các bộ được chèn. Điều này cho thấy tại sao chúng ta khai báo hai biến (NEW 1 và NEW 2) ở dòng thứ nhất.

Bây giờ chúng ta xét một thao tác cập nhật EMP. Trước tiên lượng từ hoá cập nhật được thực hiện bởi thể xử lý vấn tin và trả về một hoặc hai quan hệ tạm thời như trong trường hợp trước. Sau đó những quan hệ tạm thời được

gửi đến tất cả mọi vị trí lưu EMP. Giả sử rằng cập nhật này là một câu lệnh INSERT. Mỗi vị trí có chứa mảnh của EMP sẽ phải thi hành phán đoán C được mô tả ở trên. Bởi vì e trong C được lượng từ hoá phổ dụng, dữ liệu cục bộ tại mỗi vị trí phải thoả C. Điều này do sự kiện $\forall x \in \{a1, \ldots, a_n\} f(x)$ tương đương với $[f(a_1) \wedge \ldots \wedge f(a_n)]$. Vì vậy vị trí đưa ra yêu cầu được cập nhật phải nhận được các thông báo từ mỗi vị trí, cho biết rằng phán đoán này được thoả và là một điều kiện cho tất cả mọi vị trí. Nếu phán đoán không được thoả tại một vị trí, vị trí này sẽ gửi thông báo lỗi cho biết rằng phán đoán bị vi phạm. Do vậy cập nhật sẽ không có giá trị, và khi đó nhiệm vụ của tiểu hệ thống kiểm tra toàn vẹn là quyết định xem toàn bộ chương trình phải được loại bỏ hay không.

Bây giờ chúng ta xem xét các phán đoán đa quan hệ. Để cho đơn giản, chúng ta giả sử rằng các phán đoán toàn vẹn không có quá một biến bộ biến thiên trên cùng một quan hệ. Có lẽ đây là trường hợp hay gặp nhất. Giống như với các phán đoán đơn quan hệ, cập nhật được tính toán tại vị trí đưa ra yêu cầu này. Cưỡng chế thi hành được thực hiện tại vị trí vấn tin chính nhờ thuật toán ENFORCE dưới đây.

Thuật toán 5.1 ENFORCE

Nguyên liệu: T: Kiểu cập nhật; R: quan hệ

begin

Truy xuất tất cả các phán đoán biên dịch (R,T,C_i)

inconsistency ←false

for mỗi phán đoán biên dịch do

begin

result ← truy xuất tất cả các bộ mới (tương ứng là bộ cũ) của R với $\neg (C_i)$

if card (result) ≠ 0 then
begin
inconsistency ← true
exit
end-if
enf-for
if not (inconsistency) then
gửi các bộ cập nhật đến tất cả các vị trí đang có các mảnh của R
else phế bỏ cập nhật
end-if
end. {ENFORCE }

Thí dụ 5.20:

Chúng ta minh hoạ thuật toán này bằng một thí dụ dựa trên phán đoán khóa ngoại của Thí dụ 5.7. Gọi u là một thao tác chèn một bộ mới vào ASG. Thuật toán trên dùng phán đoán biên dịch (ASG, **INSERT**, C), trong đó C là:

$$\forall NEW \in ASG^{+}, \exists j \in PROJ: NEW.PNO = j.PNO$$

Đối với phán đoán này, câu lệnh lấy dữ liệu là truy xuất tất cả các bộ mới trong ASG^{+} không thoả C.

Câu lệnh này có thể được diễn tả bằng SQL như sau:

SELECTNEW. *
FROM ASG + NEW, PROJ
WHERE COUNT(PROJ. PNO WHERE NEW.PNO=PROJ.PNO)=0

Chú ý rằng NEW. * biểu thị tất cả các bộ của ASG^{+}

Vì vậy chiến lược của chúng ta là gửi các bộ mới đến các vị trí có lưu quan hệ PROJ để thực hiện nối rồi tập trung tất cả mọi kết quả về vị trí vấn

tin chính. Với mỗi vị trí có lưu một mảnh của PROJ, nó sẽ nối mảnh đó với ASG^+ và gửi kết quả về vị trí vấn tin chính. Nơi đây sẽ lấy hợp tất cả mọi kết quả. Nếu hợp rỗng thì CSDL nhất quán. Bằng không cập nhật này sẽ dẫn đến một trạng thái không nhất quán. Phế bỏ chương trình khi đó phụ thuộc vào chiến lược được chọn bởi bộ quản lý chương trình của DBMS phân tán.

PHÁN ĐOÁN CÓ HÀM GỘP

Loại phán đoán này có chi phí kiểm tra cao nhất vì chúng đòi hỏi phải tính các hàm gộp.

Hàm gộp thường được dùng là MIN, MAX, SUM và COUNT. Mỗi hàm gộp chứa một phần chiếu và một phần chọn. Để cưỡng chế các phán đoán này một cách hiệu quả, chúng ta có thể tạo ra các phán đoán biên dịch nhằm cô lập dữ liệu thừa được lưu tại mỗi vị trí có chứa quan hệ đi kèm. Dữ liệu này được gọi là các khung nhìn cụ thể (concrete view).

5.3.3. TÓM TẮT VỀ KIỂM SOÁT TOÀN VẸN PHÂN TÁN

Vấn đề chính của kiểm soát toàn vẹn phân tán đó là chi phí truyền và chi phí xử lý việc cưỡng chế thi hành các phán đoán phân tán. Hai vấn đề chính trong khi thiết kế một tiểu hệ thống toàn vẹn phân tán là định nghĩa các phán đoán phân tán và các thuật toán cưỡng chế nhằm hạ thấp tối đa chi phí kiểm tra toàn vẹn phân tán. Chúng ta đã chứng minh rằng việc kiểm soát ràng buộc phân tán có thể đạt được bằng cách mở rộng một phương pháp ngăn cản dựa trên quá trình biên dịch các phán đoán toàn vẹn ngữ nghĩa. Đây là phương pháp tổng quát vì tất cả mọi kiểu phán đoán diễn tả bằng

logic vị từ bậc nhất đều có thể xử lý được. Nó cũng tương thích với định nghĩa phân mảnh và hạ thấp tối đa việc truyền thông qua lại giữa các vị trí. Chúng ta có thể có được một hiệu năng cưỡng chế toàn vẹn phân tán tốt hơn nếu các mảnh được định nghĩa một cách cẩn thận. Vì thế đặc tả ràng buộc toàn vẹn phân tán là một vấn đề quan trọng trong quá trình thiết kế CSDL phân tán.

5.4. TỔNG QUAN VỀ XỬ LÝ VẤN TIN

Trong những phần còn lại của chương này chúng ta sẽ giới thiệu tổng quan về việc xử lý vấn tin trong các hệ quản trị CSDL phân tán. Ngữ cảnh được chọn ở đây là phép tính quan hệ và đại số quan hệ vì chúng được sử dụng rộng rãi trong các DBMS phân tán. Như đã thấy trong chương 4, các quan hệ phân tán được cài đặt qua các mảnh. Thiết kế CSDL phân tán có vai trò hết sức quan trọng đối với việc xử lý vấn tin vì định nghĩa các mảnh có mục đích là làm tăng tính cục bộ tham chiếu, và đôi khi là tăng khả năng thực hiện song song đối với những câu vấn tin quan trọng nhất. Vai trò của thể xử lý vấn tin phân tán là *ánh xạ câu vấn tin cấp cao* trên một CSDL phân tán vào *một chuỗi các thao tác của đại số quan hệ* trên các mảnh. Một số chức năng quan trọng biểu trưng cho ánh xạ này. Trước tiên câu vấn tin phải được phân rã thành một chuỗi các phép toán quan hệ được gọi là vấn tin đại số. Thứ hai, dữ liệu cần truy xuất phải được cục bộ hóa để các thao tác trên các quan hệ được chuyển thành các thao tác trên dữ liệu cục bộ (các mảnh). Cuối cùng câu vấn tin đại số trên các mảnh phải được mở rộng để bao gồm các thao tác truyền thông và được tối ưu hóa để hàm chi phí là thấp nhất.

Hàm chi phí muốn nói đến các tính toán như các thao tác xuất nhập đĩa, tài nguyên CPU, và mạng truyền thông.

Có hai phương pháp tối ưu hoá cơ bản được dùng trong các bộ xử lý vấn tin: *phương pháp biến đổi đại số* và *chiến lược ước lượng chi phí* .
Phương pháp biến đổi đại số đơn giản hoá các câu vấn tin nhờ các phép biến đổi đại số nhằm hạ thấp chi phí trả lời câu vấn tin, độc lập với dữ liệu thực và cấu trúc vật lý của dữ liệu.
Nhiệm vụ chính của thể xử lý vấn tin quan hệ là biến đổi *câu vấn tin cấp cao thành một câu vấn tin tương đương ở cấp thấp hơn được diễn đạt bằng đại số quan hệ*. Câu vấn tin cấp thấp thực sự sẽ cài đặt chiến lược thực thi vấn tin. Việc biến đổi này phải đạt được cả tính đúng đắn lẫn tính hiệu quả. Một biến đổi được xem là đúng đắn nếu câu vấn tin cấp thấp có cùng ngữ nghĩa với câu vấn tin gốc, nghĩa là cả hai đều cho ra cùng một kết quả. Một câu vấn tin có thể có nhiều cách biến đổi tương đương thành đại số quan hệ. Bởi vì mỗi chiến lược thực thi tương đương đều sử dụng tài nguyên máy tính rất khác nhau, khó khăn chính là chọn ra được một chiến lược hạ thấp tối đa việc tiêu dùng tài nguyên.

Thí dụ 5.21:

Chúng ta hãy xét một tập con của lược đồ CSDL của công ty máy tính đã được xét trong các chương 2 và 4 cho hai quan hệ:

EMP (ENO, ENAME, TITLE)

ASG (ENO, PNO, RESP, DUR)

và một câu vấn tin đơn giản sau:

"Cho biết tên của các nhân viên hiện đang quản lý một dự án"

Biểu thức (câu) vấn tin bằng phép tính quan hệ theo cú pháp của SQL là:

```
SELECT    ENAME
FROM      EMP, ASG
WHERE     EMP. ENO = ASG. ENO
AND       RESP = "Manager"
```

Hai biểu thức tương đương trong đại số quan hệ (xem chương 2) do biến đổi chính xác từ câu vấn tin trên là:

(1) Vì hai quan hệ trên có chung thuộc tính ENO nên ta lấy nối tự nhiên của hai quan hệ rồi chiếu lên thuộc tính ENAME:

$$(EMP \mid\!><\!\mid ASG).ENAME$$

(2) Do điều kiện bài toán yêu cầu cần ENAME của các nhân viên quản lý một dự án nên ta có thể lấy nối (nối bằng) theo teta của hai quan hệ trên và chiếu lên ENAME (tất nhiên phép nối theo teta ở đây phải được mở rộng trên các lược đồ không rời nhau):

$$(EMP \underset{I=I}{\mid\!><\!\mid} ASG).ENAME$$

Hiển nhiên là trong câu vấn tin thứ hai, chúng ta tránh sử dụng nối tự nhiên, vì thế tiêu dùng ít tài nguyên máy tính hơn câu vấn tin thứ nhất và vì vậy nên được lưu lại.

Trong bối cảnh tập trung, chiến lược thực thi vấn tin có thể được diễn tả chính xác bằng một mở rộng của đại số quan hệ. Nhiệm vụ của thể xử lý vấn tin tập trung là đối với một câu vấn tin đã cho, nó phải chọn ra được một câu vấn tin đại số tốt nhất trong số những câu vấn tin tương đương. Bởi vì đây là một bài toán phức tạp về mặt tính toán khi số lượng các quan hệ khá lớn, nên nói chung nó thường được rút lại ở yêu cầu là chọn được một lời giải gần tối ưu.

Trong các hệ phân tán, đại số quan hệ không đủ để diễn tả các chiến lược thực thi. Nó phải được cung cấp thêm các phép toán trao đổi dữ liệu giữa các vị trí.

Bên cạnh việc chọn thứ tự cho các phép toán đại số quan hệ, thể xử lý vấn tin phân tán cũng phải chọn các vị trí tốt nhất để xử lý dữ liệu, và có thể cả cách biến đổi dữ liệu. Kết quả là không gian lời giải các chiến lược thực thi sẽ tăng lên, làm cho việc xử lý vấn tin phân tán khó khăn hơn rất nhiều.

Thí dụ 5.22:

Thí dụ này minh hoạ tầm quan trọng của việc chọn lựa vị trí và cách truyền dữ liệu của một câu vấn tin đại số. Chúng ta xét lại câu vấn tin của thí dụ 5.21:

$$(EMP \underset{I=I}{|><|} ASG).ENAME$$

Chúng ta giả sử rằng các quan hệ EMP và ASG được phân mảnh ngang như sau:

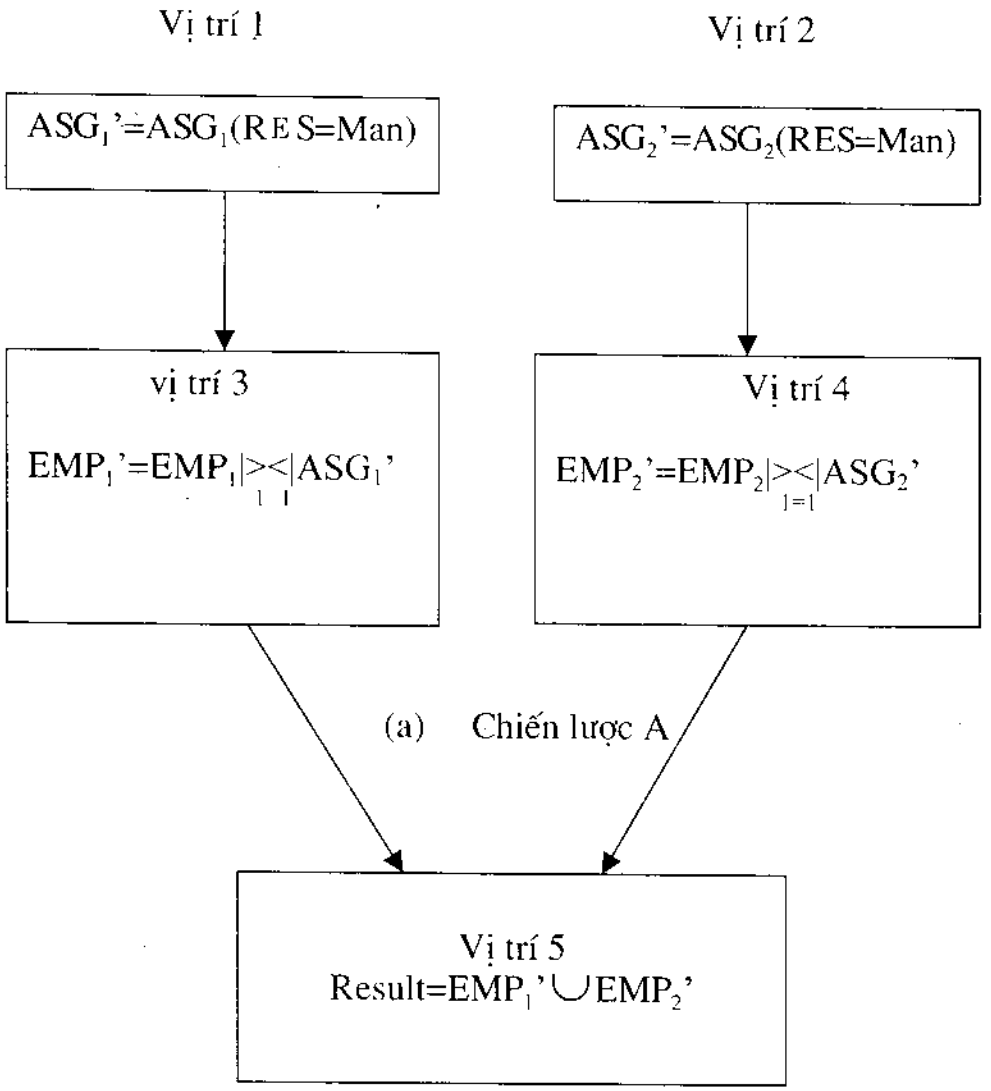
$EMP_1 = EMP(ENO \leq E3)$

$EMP_2 = EMP(ENO > E3)$

$ASG_1 = ASG(ENO \leq E3)$

$ASG_2 = ASG(ENO \geq E3)$

Các mảnh ASG_1, ASG_2, EMP_1, EMP_2 theo thứ tự được lưu tại các vị trí 1, 2, 3 và 4 kết quả được lưu tại vị trí 5. Ta sẽ xét hai chiến lược A và B tương ứng bằng hai chuỗi biểu thức đại số quan hệ như sau (xem hình 5.5).



Vị trí 5

Result = (EMP$_1$ ∪ EMP$_2$)|><|(ASG$_1$ ∪ ASG$_2$)(RES=Manager)

Vị trí 1 Vị trí 2 Vị trí 3 Vị trí 4

(b) Chiến lược B

Hình 5.5 Các chiến lược thực thi vấn tin tương đương

Hai chiến lược thực thi phân tán tương đương cho câu vấn tin trên được trình bày trong hình 5.5. Mũi tên từ vị trí i đến vị trí j chỉ ra rằng quan hệ R được chuyển từ vị trí i đến vị trí j. Chiến lược A sử dụng sự kiện là các quan hệ EMP và ASG được phân mảnh theo cùng một cách để thực hiện song song các phép toán chọn và nối. Chiến lược B tập trung tất cả các dữ liệu tại vị trí lưu kết quả trước khi xử lý câu vấn tin.

Để đánh giá được việc tiêu dùng tài nguyên của hai chiến lược này, chúng ta sử dụng một mô hình chi phí đơn giản sau. Chúng ta giả sử rằng thao tác truy xuất một bộ (tuple access), được ký hiệu là tupacc, là 1 đơn vị và thao tác truyền một bộ (tuple transfer) tuptrans là 10 đơn vị, Đồng thời chúng ta cũng giả sử là các quan hệ EMP và ASG tương ứng có 400 và 1000 bộ, và có 20 giám đốc dự án (manager) trong quan hệ ASG. Chúng ta cũng giả sử rằng dữ liệu được phân tán thống nhất cho các vị trí. Cuối cùng chúng ta giả sử rằng các quan hệ ASG và EMP được gom tụ cục bộ tương ứng theo các thuộc tính RESP và ENO. Vì vậy có thể truy xuất trực tiếp đến các bộ của ASG dựa trên giá trị của thuộc tính RESP (tương ứng là ENO cho EMP).

Tổng chi phí của chiến lược A có thể được tính như sau:

1. Tạo ra ASG' bằng cách chọn trên ASG cần (10 + 10) * tupacc = 20
2. Truyền ASG' đến các vị trí của EMP cần (10 + 10) * tuptrans = 200
3. Tạo EMP' bằng cách nối ASG' và EMP' cần (10 + 10)* tupacc*2 = 40
4. Truyền EMP' đến vị trí nhận kết quả cần (10 + 10) * Tuptrans = 200

Tổng chi phí 460

Tổng chi phí của chiến lược B có thể được tính như sau:

1. Truyền EMP đến vị trí 5 cần 400 * tuptrans = 4.000
2. Truyền ASG đến vị trí 5 cần 1000 * tuptrans = 10.000
3. Tạo ra ASG' bằng cách chọn trên ASG cần 1000 * tupacc = 1.000
4. Nối EMP và ASG' cần 400 * 20* tupacc = 8.000

Tổng chi phí là: 23.000

Trong chiến lược B, chúng ta đã giả sử rằng các phương pháp truy xuất các quan hệ EMP và ASG dựa trên các thuộc tính RESP và ENO bị mất tác dụng do việc truyền dữ liệu. Đây là một giả thiết hợp lý trong thực tế. Theo bảng thống kê trên ta thấy chiến lược A tốt hơn chiến lược B 50 lần. Hơn nữa nó đưa ra một cách phân phối công việc giữa các vị trí. Khác biệt còn cao hơn nữa nếu chúng ta giả thiết là tốc độ truyền chậm hơn và/ hoặc mức độ phân mảnh cao hơn.

Sau đây chúng ta sẽ xét các vấn đề về phân rã các câu vấn tin.

5.5. PHÂN RÃ VẤN TIN

Phân rã vấn tin là giai đoạn đầu tiên của quá trình xử lý câu vấn tin. Nó biến đổi câu vấn tin ở dạng phép tính quan hệ thành câu vấn tin đại số quan hệ. Các vấn tin nhập và xuất đều tham chiếu đến các quan hệ toàn cục và không dùng đến các thông tin về phân bố dữ liệu. Vì thế phân rã vấn tin đều giống nhau trong cả hệ tập trung lẫn phân tán. Trong phần này, chúng ta giả sử là câu vấn tin nhập đã đúng về cú pháp. Khi kết thúc giai đoạn này, câu vấn tin xuất sẽ đúng về ngữ nghĩa và đạt chất lượng, theo nghĩa là đã loại bỏ các hành động không cần thiết. Các bước phân rã vấn tin là:

(1) Chuẩn hóa.

(2) Phân tích.

(3) Loại bỏ dư thừa.

(4) Viết lại vấn tin.

Các bước 1, 3 và 4 dựa vào các sự kiện là đối với một câu vấn tin sẽ có nhiều biến đổi tương đương, và một số trong chúng có hiệu năng tốt hơn. Chúng ta trình bày ba bước đầu tiên trong ngữ cảnh của phép tính trong

quan hệ bộ. Chỉ có bước cuối cùng sẽ viết lại câu vấn tin thành dạng đại số quan hệ.

5.5.1.CHUẨN HÓA

Câu vấn tin nhập có thể khá phức tạp, tuỳ thuộc vào các tiện ích có sẵn của ngôn ngữ. Mục đích của chuẩn hóa (normalization) là biến đổi câu vấn tin thành một dạng chuẩn để xử lý tiếp. Với các ngôn ngữ quan hệ như SQL, biến đổi quan trọng nhất là lượng từ hóa vấn tin (query qualification) (mệnh đề WHERE), có thể đó là một vị từ phi lượng từ với độ phức tạp nào đó, với tất cả các lượng từ cần thiết ($\forall$ hoặc $\exists$) được đặt phía trước. Có hai dạng chuẩn có thể có cho vị từ, một có thứ bậc cao cho AND ($\wedge$) và loại còn lại có thứ bậc cao cho OR ($\vee$). Dạng chuẩn hội (conjunctive normal form) là hội (vị từ $\wedge$) của các tuyển (các vị từ $\vee$):

$$(p_{11} \vee p_{12} \vee \ldots \vee p_{1n}) \wedge \ldots \wedge (p_{m1} \vee p_{m2} \vee \ldots \vee p_{mn})$$

Trong đó P_{ij} là một vị từ đơn giản. Ngược lại, một lượng từ hóa ở dạng chuẩn tuyển (disjunctive normal form) như sau:

$$(p_{11} \wedge p_{12} \wedge \ldots \wedge p_{1n}) \vee \ldots \vee (p_{m1} \wedge p_{m2} \wedge \ldots \wedge p_{mn})$$

Biến đổi các vị từ phi lượng từ là tầm thường bằng cách sử dụng các quy tắc tương đương đã biết cho các phép toán logic ($\wedge$, $\vee$ và $\neg$):

1. $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$
2. $p_1 \vee p_2 \Leftrightarrow p_2 \vee p_1$
3. $p_1 \wedge (p_2 \wedge p_3) \Leftrightarrow (p_1 \wedge p_2) \wedge p_3$
4. $p_1 \vee (p_2 \vee p_3) \Leftrightarrow (p_1 \vee p_2) \vee p_3$
5. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$

6. $p_1 \vee (p_2 \wedge p_3) \Leftrightarrow (p_1 \vee p_2) \wedge (p_1 \vee p_3)$

7. $\neg (p_1 \wedge p_2) \Leftrightarrow \neg p_1 \vee \neg p_2$

8. $\neg (p_1 \vee p_2) \Leftrightarrow \neg p_1 \wedge \neg p_2$

9. $\neg (\neg p) \Leftrightarrow p$

Trong dạng chuẩn tuyển, câu vấn tin có thể được xử lý như các câu vấn tin con hội độc lập, được nối lại bằng phép hợp (tương ứng với các tuyển mệnh đề). Tuy nhiên dạng này có thể dẫn đến các vị từ nối và chọn trùng nhau như được trình bày trong thí dụ sau. Lý do là các vị từ rất hay được nối với các vị từ khác qua phép AND logic. Sử dụng quy tắc 5 ở trên, với p_1 là một vị từ nối hoặc chọn, sẽ phải lập lại p_1. Dạng chuẩn hội thường được sử dụng trong thực tế bởi vì các lượng từ hóa vấn tin thường chứa nhiều vị từ AND hơn là OR. Tuy nhiên nó cũng dẫn đến việc lập lại vị từ cho các câu vấn tin có chứa nhiều tuyển và ít hội, một trường hợp hiếm gặp.

Thí dụ 5.23:

Chúng ta hãy xét câu vấn tin trên CSDL của Công ty điện toán đã được đề cập nhiều lần.

"Tìm tên các nhân viên đang làm việc ở dự án P1 trong 12 hoặc 24 tháng"

Câu vấn tin được diễn tả bằng SQL như sau:

```
SELECT    ENAME
FROM      EMP, ASG
WHERE     EMP.ENO = ASG.ENO
  AND ASG.PNO = "P1"
  AND DUR = 12 OR DUR = 24
```

Lượng từ hóa ở dạng chuẩn hội là

EMP.ENO = ASG.ENO ^ ASG.PNO = "P1" ^ (DUR = 12 ∨ DUR = 24)

Còn lượng từ hóa ở dạng chuẩn tuyển là:

(EMP.ENO = ASG.ENO ^ ASG.PNO = "P1" ^ DUR=12) ∨

(EMP.ENO = ASG.ENO ^ ASG.PNO = "P1" ^ DUR =24)

Trong dạng sau, xử lý hai hội độc lập có thể là một công việc thừa nếu các biểu thức con chung không được loại bỏ.

5.5.2. PHÂN TÍCH

Phân tích câu vấn tin cho phép phế bỏ các câu vấn tin đã chuẩn hóa nhưng không thể tiếp tục xử lý được hoặc không cần thiết. Những lý do chính là do chúng sai kiểu hoặc sai nghĩa. Khi phát hiện ra một trong những trường hợp này, hệ thống chỉ đơn giản trở về và đưa ra một thông báo giải thích. Nếu không, xử lý vấn tin sẽ được tiếp tục. Dưới đây chúng tôi trình bày các kỹ thuật phát hiện các vấn tin không đúng này.

Một câu vấn tin gọi là sai kiểu (type incorrect) nếu có một thuộc tính hoặc tên quan hệ chưa được khai báo trong lược đồ toàn cục, hoặc nếu nó áp dụng các phép toán cho các thuộc tính có kiểu không thích hợp. Kỹ thuật được sử dụng để phát hiện các câu vấn tin sai kiểu ở đây giống như kỹ thuật kiểm tra kiểu trong ngôn ngữ lập trình. Tuy nhiên, các khai báo kiểu là thành phần của lược đồ toàn cục chứ không phải nằm trong câu vấn tin bởi vì một câu vấn tin quan hệ không tạo ra một kiểu nào mới.

Thí dụ 5.24:

Câu vấn tin SQL sau đây trên CSDL về các dự án

```
SELECT    E#
FROM      EMP
WHERE     ENAME > 200
```

là sai kiểu vì hai lý do: thứ nhất, thuộc tính E# chưa được khai báo trong lược đồ. Thứ hai, phép so sánh ">200" không phù hợp với kiểu chuỗi của ENAME.

Một câu vấn tin gọi là sai nghĩa (semantically incorrect) nếu các thành phần của nó không tham gia vào việc tạo ra kết quả. Trong ngữ cảnh của phép tính quan hệ, chúng ta không thể xác định được tính đúng đắn ngữ nghĩa cho câu vấn tin tổng quát. Tuy nhiên vẫn có thể thực hiện điều đó cho một lớp vấn tin quan hệ khá rộng rãi, đó là các vấn tin không chứa các tuyển và phủ định bằng cách biểu diễn câu vấn tin như một đồ thị, được gọi là đồ thị kết nối (connection graph) hay đồ thị vấn tin (query graph). Chúng ta định nghĩa đồ thị này cho các loại vấn tin hữu ích nhất có chứa phép chọn, chiếu và nối. Trong một đồ thị vấn tin, một nút biểu thị cho một quan hệ kết quả, và các nút khác biểu thị cho các quan hệ toán hạng. Một cạnh giữa hai nút không phải là quan hệ kết quả biểu diễn cho một nối, còn một cạnh mà nút đích là kết quả sẽ biểu thị cho phép chiếu. Còn các nút không phải là kết quả có thể được gắn nhãn là một vị từ chọn hoặc một vị từ tự nối (nối của một quan hệ với chính nó). Một đồ thị con quan trọng của đồ thị vấn tin là đồ thị nối (join graph) chỉ có các nối. Đồ thị nối rất có ích cho giai đoạn tối ưu hóa vấn tin.

Thí dụ 5.25:

Chúng ta hãy xét câu vấn tin sau:

"Tìm tên và nhiệm vụ của các lập trình viên đã làm việc cho dự án CAD/CAM trong hơn ba năm".

Câu vấn tin này được diễn tả trong SQL như sau:

```
SELECT    ENAME , RESP
FROM      EMP, ASG , PROJ
WHERE     EMP. ENO = ASG.ENO
 AND ASG.PNO = PROJ. PNO
 AND PNAME = "CAD/CAM"
 AND DUR ≥ 36
 ANDTITLE = "Programmer"
```

Ta có thể biểu thị các câu vấn tin dưới dạng đồ thị. Đồ thị vấn tin rất có ích cho việc xác định tính đúng đắn về ngữ nghĩa của một câu vấn tin hội đa biến không có phủ định. Một câu vấn tin như thế sẽ sai ngữ nghĩa nếu đồ thị vấn tin của nó không liên thông. Trong trường hợp này một hoặc nhiều đồ thị con (tương ứng với câu vấn tin con) bị tách rời khỏi đồ thị chứa quan hệ kết quả. Câu vấn tin sẽ được xem là đúng (một số hệ thống sẽ thực hiện) bằng cách xét các nối kết bị thiếu như một tích Descartes. Nhưng nói chung, vấn đề chính là các vị từ nối bị thiếu và câu vấn tin cần được loại bỏ.

Thí dụ 5.26:

Chúng ta hãy xét câu vấn tin SQL

```
SELECT    ENAME , RESP
FROM      EMP, ASG , PROJ
WHERE     EMP. ENO = ASG.ENO
```

AND PNAME = "CAD/CAM"
AND DUR ≥ 36
ANDTITLE = "Programmer"

Đồ thị vấn tin của nó không liên thông, cho chúng ta biết rằng câu vấn tin này sai về ngữ nghĩa. Có ba giải pháp cơ bản cho vấn đề này:

(1) loại bỏ câu vấn tin, hoặc

(2) giả định rằng có một tích Descartes ngầm định giữa các quan hệ ASG và PROJ, hoặc

(3) suy ra (sử dụng lược đồ) vị từ nối bị thiếu ASG.PNO = PROJ.PNO để biến đổi nó thành câu vấn tin của thí dụ trên.

5.5.3. LOẠI BỎ DƯ THỪA

Như chúng ta đã biết, các ngôn ngữ quan hệ có thể được dùng một cách thống nhất nhằm kiểm soát dữ liệu ngữ nghĩa. Đặc biệt một câu vấn tin của người sử dụng thường được diễn tả trên một khung nhìn có thể được bổ sung thêm nhiều vị từ để có được sự tương ứng khung nhìn - quan hệ, bảo đảm được tính toàn vẹn ngữ nghĩa và bảo mật. Thế nhưng lượng từ hóa vấn tin đã được sửa đổi này có thể chứa các vị từ dư thừa. Một ước lượng ngây thơ cho lượng từ hóa có dư thừa có thể khiến phải lặp lại một số công việc. Các vị từ và các công việc thừa có thể được loại bỏ bằng cách giảm ước lượng từ hóa đó bằng các quy tắc lũy đẳng sau.

1. $P \wedge P \Leftrightarrow P$
2. $P \vee P \Leftrightarrow P$
3. $P \wedge true \Leftrightarrow P$
4. $P \vee false \Leftrightarrow P$

5. $P \wedge false \Leftrightarrow false$

6. $P \vee true \Leftrightarrow true$

7. $P \wedge \neg P \Leftrightarrow false$

8. $P \vee \neg P \Leftrightarrow true$

9. $P_1 \wedge (P_1 \vee P_2) \Leftrightarrow P_1$

10. $P_1 \vee (P_1 \wedge P_2) \Leftrightarrow P_1$

Thí dụ 5. 27:

Câu vấn tin SQL

```
SELECT    TITLE
FROM      EMP
WHERE     (NOT (TITLE = "Programmer")
 AND TITLE = "Programmer"
 OR        TITLE = "ELect. Eng. ")
  ANDNOT (TITLE = "Elect. Eng. ")
  OR      ENAME = " J. Doe"
```

Có thể được đơn giản hóa nhờ các quy tắc trên và trở thành

```
SELECT    TITLE
FROM      EMP
WHERE     ENAME = "J. Doe"
```

Cách đơn giản hóa được tiến hành như sau: Gọi p_1 là < TITLE = "Programmer">, p_2 là <TITLE = "ELect. Eng. ">, và p_3 là <ENAME = "J.Doe">. Lượng từ hóa vấn tin là:

$(\neg p_1 \wedge (p_1 \vee p_2) \wedge \neg p_2) \vee p_3$

Dạng chuẩn tuyển cho lượng từ hóa này thu được bằng cách áp dụng các quy tắc 5 được định nghĩa trong phần trước, cho kết quả:

$$(\neg p_1 \wedge ((p_1 \wedge \neg p_2) \vee (p_2 \wedge \neg p_2))) \vee p_3$$

rồi bằng quy tắc 3, ta được :

$$(\neg p_1 \wedge p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2 \wedge \neg p_2) \vee p_3$$

Bằng cách áp dụng quy tắc 7, ta được :

$$(\text{false} \wedge \neg p_2) \vee (\neg p_1 \wedge \text{false}) \vee p_3$$

và áp dụng quy tắc 5 hai lần, chúng ta có

$$\text{false} \vee \text{false} \vee p_3$$

biểu thức trên tương đương với p_3 theo quy tắc 4.

5.5.4. VIẾT LẠI CÂU VẤN TIN

Bước cuối cùng của phân rã vấn tin là viết lại câu vấn tin bằng đại số quan hệ. Điển hình, bước này được chia thành hai bước nhỏ: (1) biến đổi câu vấn tin từ phép tính quan hệ thành đại số quan hệ, (2) cấu trúc lại câu vấn tin đại số nhằm cải thiện hiệu năng. Để cho dễ hiểu, chúng ta sẽ trình bày câu vấn tin đại số quan hệ một cách hình ảnh bằng cây toán tử (operator tree). Một cây toán tử là một cây với mỗi nút lá biểu thị cho một quan hệ được lưu trong CSDL, và các nút không phải nút lá biểu thị cho một quan hệ trung gian được sinh ra bởi các phép toán quan hệ. Chuỗi các phép toán đi theo hướng từ lá đến gốc biểu thị cho kết quả vấn tin.

Biến đổi câu vấn tin phép tính quan hệ bộ thành một cây toán tử có thể thu được dễ dàng bằng cách sau: trước tiên chúng ta tạo ra một nút lá cho mỗi biến bộ khác nhau (tương ứng với một quan hệ). Trong SQL, các nút lá có sẵn trong mệnh đề FROM. Thứ hai, nút gốc được tạo ra như một phép chiếu chứa các thuộc tính kết quả. Các thuộc tính này nằm trong mệnh đề SELECT của câu vấn tin SQL. Thứ ba, lượng từ hóa (mệnh đề WHERE

của SQL) được dịch thành chuỗi các phép toán quan hệ thích hợp (phép chọn, nối, hợp, v.v.) đi từ các nút lá đến gốc. Chuỗi này có thể được cho trực tiếp qua thứ tự xuất hiện của các vị từ và các toán tử.

Thí dụ 5.28:

Câu vấn tin "Tìm tên các nhân viên trừ J.Doe đã làm cho dự án CAD/CAM trong một hoặc hai năm".

Biểu thức SQL là:

```
SELECT    ENAME
FROM      PROJ,  ASG, EMP
WHERE     ASG. ENO  = EMP. ENO
AND       ASG. PNO  = PROJ. PNO
AND       ENAME ≠ "J. Doe"
AND       PROJ. NAME  = "CAD/CAM"
AND       (DUR  = 12 OR DUR  = 24).
```

Sáu quy tắc tương đương hữu ích nhất và được xem là các phép toán đại số quan hệ cơ bản sẽ được trình bày dưới đây. Tính đúng đắn của chúng được chứng minh trong Ullman (1982).

Trong những đoạn còn lại của phần này r, s và T là những quan hệ, E là mệnh đề logic, với quan hệ r được định nghĩa trên các thuộc tính $\{A_1, A_2, \ldots A_n\}$ và quan hệ s được định nghĩa trên các thuộc tính $\{B_1, B_2, \ldots B_m\}$ và X là một tập thuộc tính.

1. TÍNH GIAO HOÁN CỦA CÁC PHÉP TOÁN HAI NGÔI

Tích Descartes của hai quan hệ r và s có tính giao hoán:

$r \times s \Leftrightarrow s \times r$

Tương tự, nối của hai quan hệ cũng có tính giao hoán.

$r \mid\!><\!\mid s \Leftrightarrow s \mid\!><\!\mid r$

Quy tắc này cũng áp dụng được cho hợp nhưng không áp dụng cho hiệu tập hợp hay nối nửa.

2. TÍNH KẾT HỢP CỦA CÁC PHÉP TOÁN HAI NGÔI

Tích Descartes và nối là các phép toán có tính kết hợp.

$(r \times s) \times T \Leftrightarrow r \times (s \times T)$

$(r \mid\!><\!\mid s) \mid\!><\!\mid T \Leftrightarrow r \mid\!><\!\mid (s \mid\!><\!\mid T)$

3. TÍNH LŨY ĐẲNG CỦA CÁC PHÉP TOÁN ĐƠN NGÔI

Nhiều phép chiếu liên tiếp trên cùng một quan hệ có thể được nhóm lại. Đảo lại, một phép chiếu trên nhiều thuộc tính có thể được tách ra thành nhiều phép chiếu liên tiếp nhau. Nếu r được định nghĩa trên tập thuộc tính A và $A' \subseteq A$, $A'' \subseteq A'$ thì:

$$(r.(A')).A'' = r.A''$$

Nhiều phép chọn liên tiếp theo thứ tự $p_1, p_2, \ldots p_k$ trên cùng một quan hệ r, trong đó p_i là một vị từ và đầu tiên chọn trong r các bộ thoả p_1, sau đó chọn trong r các bộ thoả $p_2, \ldots$ có thể được nhóm lại như sau: $r(p_k)$

Đảo lại một phép chọn qua một hội các vị từ có thể được tách ra thành nhiều phép chọn liên tiếp nhau.

4. GIAO HOÁN CHỌN VỚI CHIẾU

Phép chọn và chiếu trên cùng một quan hệ có thể được giao hoán như sau: Chiếu xong chọn bằng chọn xong chiếu $(r.X)(E) = (r(E)).X$

Chú ý rằng nếu E là mệnh đề logic có các thuộc tính ra khỏi tập thuộc tính X thì phép chọn đầu ở vế trái của hệ thức không có tác dụng.

5. GIAO HOÁN PHÉP CHỌN VỚI CÁC PHÉP TOÁN HAI NGÔI

Phép chọn và tích Descartes có thể giao hoán bằng các quy tắc sau (cần nhớ rằng thuộc tính A_i thuộc quan hệ r):

$(r \times s)(E) \Leftrightarrow r(E) \times s(E)$ nếu E thoả trong r và s

Chọn và nối cũng có thể giao hoán

$(r \mid\!><\!\mid s)(E) \Leftrightarrow r(E) \mid\!><\!\mid s(E)$

Chọn và hợp cũng có thể giao hoán nếu r và T có lược đồ giống nhau.

$(r + T)(E) \Leftrightarrow r(E) + T(E)$

Chọn và hiệu cũng có thể được giao hoán tương tự.

6. GIAO HOÁN CHO PHÉP CHIẾU VỚI PHÉP TOÁN HAI NGÔI

Chiếu và tích Descartes có thể được giao hoán. Nếu $C = A' \cup B'$,

trong đó A' $\subseteq$ A, B' $\subseteq$ B và A, B là các tập thuộc tính tương ứng của các quan hệ r và s, chúng ta có:

$(r \times s).X \Leftrightarrow r.X \times s.X$

Chiếu và nối cũng có thể giao hoán:

$(r \mid\!><\!\mid s).X \Leftrightarrow r.X \mid\!><\!\mid s.X$

Để có đẳng thức đúng, chúng ta giả sử rằng các điều kiện cần thiết của X thoả mãn. Chiếu và hợp có thể được giao hoán như sau:

$(r + s).X \Leftrightarrow r.X + s.X$

Chiếu và hiệu có thể được giao hoán tương tự.

Áp dụng sáu quy tắc này cho phép tạo ra nhiều cây tương đương. Trong giai đoạn tối ưu hóa, người ta thử so sánh tất cả các cây dựa trên chi phí dự đoán. Tuy nhiên, số lượng cây quá lớn làm cho cách tiếp cận này không thực tế. Các quy tắc trên có thể được sử dụng để cấu trúc lại cây một cách có hệ thống nhằm loại bỏ các cây "xấu". Chúng ta có thể sử dụng các quy tắc này bằng bốn cách khác nhau. Trước tiên chúng cho phép tách các phép toán đơn ngôi, làm đơn giản hóa biểu thức vấn tin. Thứ hai, các phép toán đơn ngôi trên cùng một quan hệ có thể được nhóm lại để chỉ cần thực hiện truy xuất đến quan hệ một lần. Thứ ba, các phép toán đơn ngôi có thể giao hoán với các phép toán hai ngôi để một số phép toán (chẳng hạn phép chọn) có thể được thực hiện trước. Thứ tư, các phép toán hai ngôi có thể được sắp xếp lại. Quy tắc cuối cùng được sử dụng rộng rãi trong kỹ thuật tối ưu hóa vấn tin.

Một thuật toán tái cấu trúc đơn giản được trình bày trong Ullman (1982) có sử dụng một heuristic trong đó áp dụng các phép toán đơn ngôi (chọn/ chiếu) càng sớm càng tốt nhằm giảm bớt kích thước của các quan hệ trung gian.

Thí dụ 5.29:

Tái cấu trúc cây trong hình 5.3 sinh ra cây trong hình 5.5. Cây kết quả được xem là đạt chất lượng theo nghĩa là nó tránh truy xuất nhiều lần đến cùng một quan hệ và các phép toán chọn lựa nhiều nhất được thực hiện trước tiên. Tuy nhiên cây này còn xa mới đạt được tối ưu. Thí dụ phép toán chọn trên EMP không hữu ích lắm trước phép nối bởi vì nó không làm kích thước của quan hệ toán hạng giảm đi nhiều.

5.6. CỤC BỘ HÓA DỮ LIỆU PHÂN TÁN

Trong phần 5.1 chúng ta đã trình bày các kỹ thuật tổng quát để phân rã và tái cấu trúc các câu vấn tin được diễn tả bằng phép tính quan hệ. Các kỹ thuật này áp dụng cho cả hệ thống phân tán lẫn các hệ thống tập trung và không xét đến sự phân bố dữ liệu. Đây là vai trò của tầng cục bộ hóa. Như đã được trình bày trong lược đồ phân tầng tổng quát, tầng cục bộ hóa dữ liệu chịu trách nhiệm dịch câu vấn tin đại số trên quan hệ toàn cục sang câu vấn tin đại số trên các mảnh vật lý. Cục bộ hoá có sử dụng các thông tin được lưu trong lược đồ phân mảnh

Phân mảnh được định nghĩa qua các quy tắc phân mảnh, và chúng có thể được diễn tả như các vấn tin quan hệ. Như đã thảo luận trong chương trước, một quan hệ toàn cục có thể được xây dựng lại bằng cách áp dụng các quy tắc tái thiết (hoặc phân mảnh đảo) và dẫn xuất ra một chương trình đại số quan hệ mà các toán hạng của nó là các mảnh. Chúng ta gọi đây là chương trình cục bộ hóa (localization program). Để cho đơn giản, chúng ta không xét tình huống các mảnh được nhân bản, mặc dù điều này có thể làm tăng hiệu năng. Một cách cục bộ hóa một vấn tin phân tán khá thô sơ là tạo ra một câu vấn tin trong đó mỗi quan hệ toàn cục được thay bằng chương

trình cục bộ hóa của nó. Điều này có thể xem như việc thay các nút lá của cây toán tử của câu vấn tin phân tán bằng các cây con tương ứng với các chương trình cục bộ hóa. Câu vấn tin thu được theo cách này được gọi là câu vấn tin gốc (generic query). Nói chung thì phương pháp này không hiệu quả bởi vì nhiều tái cấu trúc và đơn giản hóa quan trọng cho các vấn tin gốc vẫn có thể được thực hiện (Ceri and Pelagatti, 1983 và Ceri et al., 1986). Trong những đoạn còn lại của phần này, đối với mỗi kiểu phân mảnh, chúng ta sẽ trình bày các kỹ thuật rút gọn (reduction technique) để tạo ra các câu vấn tin tối ưu và đơn giản hơn. Chúng ta sẽ sử dụng các quy tắc biến đổi và các heuristic, chẳng hạn như đẩy các phép toán đơn ngôi xuống thấp như đã được giới thiệu ở trên.

5.6.1. RÚT GỌN CHO PHÂN MẢNH NGANG NGUYÊN THỦY

Việc phân mảnh ngang phân tán một quan hệ dựa trên các vị từ chọn (select predicate). Thí dụ dưới đây được dùng cho các phần bàn luận tiếp theo.

Thí dụ 5.30:

Quan hệ EMP (ENO, ENAME, TITLE) của hình 4.1 có thể được tách thành ba mảnh ngang EMP_1, EMP_2, và EMP_3 được định nghĩa như sau:

EMP_1 = EMP(ENO ≤ "E3")

EMP_2 = EMP("E3" <ENO ≤ "E6")

EMP_3 = EMP(ENO > "E6")

Chú ý rằng phân mảnh cho quan hệ EMP ở đây khác với phân mảnh được thảo luận trong phần trước.

Chương trình cục bộ hóa cho quan hệ phân mảnh ngang là hợp của các mảnh.

Trong thí dụ này chúng ta có:

$EMP = EMP_1 \cup EMP_2 \cup EMP_3$

Vì vậy dạng vấn tin gốc được xác định trên EMP sẽ thu được nhờ thay EMP bằng ($EMP_1 \cup EMP_2 \cup EMP_3$).

Rút gọn vấn tin trên các quan hệ phân mảnh ngang chủ yếu là xác định, sau khi tái cấu trúc lại các cây con, xem cây con nào tạo ra các quan hệ rỗng rồi loại bỏ chúng. Phân mảnh ngang có thể được dùng để đơn giản hóa phép chọn và phép nối.

5.6.2.RÚT GỌN VỚI PHÉP CHỌN

Chọn trên các mảnh có lượng từ hóa mâu thuẫn với lượng từ hóa của quy tắc phân mảnh sẽ sinh ra các quan hệ rỗng. Cho một quan hệ r được phân mảnh ngang thành $r_1, r_2, \ldots, r_w$, trong đó $r_j = r(p_j)$, quy tắc trên được phát biểu một cách hình thức như sau:

QUY TẮC 1:

$r_j = \varnothing$ nếu $\forall t$ thuộc r : $\neg (t(p_i) \wedge t(p_j))$

Trong đó p_i và p_j là các vị từ chọn, t biểu thị cho một bộ, và t(p) biểu thị "vị từ p đúng với t".

Chẳng hạn, vị từ chọn ENO = "E1" mâu thuẫn với các vị từ của các mảnh EMP_2 và EMP_3 của thí dụ trên đây (nghĩa là không có các bộ nào trong EMP_2 và EMP_3 thoả vị từ này). Xác định các vị từ mâu thuẫn đòi hỏi phải sử dụng các kỹ thuật chứng minh định lý nếu các vị từ này hoàn toàn tổng quát (Hunt and Rosenkrantz, 1979). Tuy nhiên các DBMS nói chung sẽ

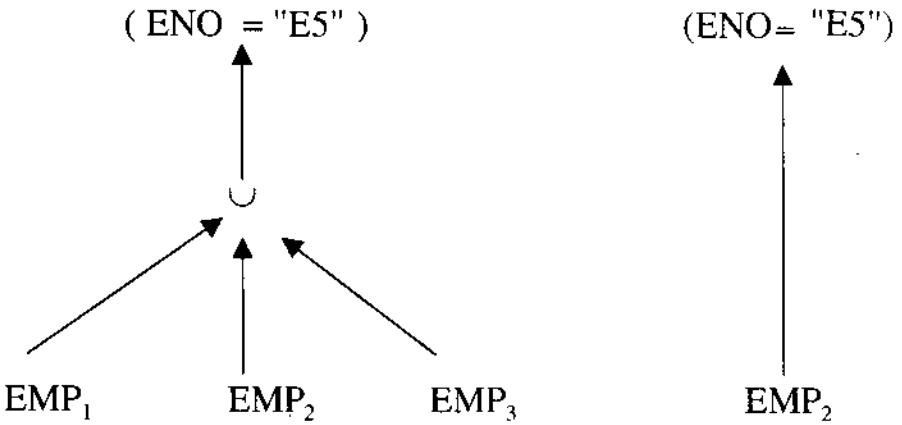
đơn giản hóa việc so sánh vị từ bằng cách chỉ hỗ trợ những vị từ đơn giản khi định nghĩa các quy tắc phân mảnh (bởi quản trị viên CSDL, DBA).

Thí dụ 5.31:

Chúng ta sẽ minh hoạ phép rút gọn cho phân mảnh ngang bằng cách dùng câu vấn tin mẫu sau đây:

```
SELECT    *
FROM      EMP
WHERE     ENO = "E5"
```

Áp dụng cách tiếp cận thô sơ để cục bộ hóa EMP từ EMP_1, EMP_2 và EMP_3, cho ra câu vấn tin gốc của hình 5.6a. Bằng cách hoán vị phép chọn với phép hợp, chúng ta dễ dàng phát hiện ra rằng vị từ chọn mâu thuẫn với các vị từ của EMP_1 và EMP_3, do vậy tạo ra các quan hệ rỗng. Câu vấn tin đã rút gọn chỉ áp dụng cho một mình EMP_2 được trình bày như trong Hình 5.6b.



(a) Vấn tin gốc (b) Vấn tin đã rút gọn

Hình 5.6 Rút gọn cho phân mảnh ngang (với phép chọn)

5.6.3.RÚT GỌN VỚI PHÉP NỐI

Nối trên các quan hệ phân mảnh ngang có thể được đơn giản khi các quan hệ nối được phân mảnh theo thuộc tính nối. Đơn giản hóa gồm có phân phối các nối trên các hợp rồi bỏ đi các nối vô dụng. Phân phối nối trên hợp có thể được phát biểu:

$$(r_1 \cup r_2) |><| s = (r_1 |><| s) \cup (r_2 |><| s)$$

trong đó r_i là các mảnh của r còn s là một quan hệ.

Bằng phép biến đổi này, các hợp có thể được di chuyển lên trên cây toán tử để tất cả các nối có thể có các mảnh đều được lộ ra.

Các nối vô dụng có thể xác định được khi lượng từ hóa của các mảnh nối có mâu thuẫn. Giả sử rằng các mảnh r_i và r_j được định nghĩa tương ứng theo các vị từ p_i và p_j trên cùng một quan hệ, quy tắc đơn giản hóa có thể được phát biểu như sau:

QUI TẮC 2:

$$r_i = \varnothing \text{ nếu } \forall t,t' \text{ thuộc } r : \neg (t(p_i) \wedge t'(p_j))$$

Vì thế việc xác định các nối vô dụng có thể được thực hiện bằng cách chỉ xét các từ của các mảnh. Áp dụng quy tắc này cho phép nối hai quan hệ được cài đặt như các nối từng phần song song cho các mảnh (Ceri et al., 1986). Câu vấn tin rút gọn không phải lúc nào cũng tốt hơn (nghĩa là đơn giản hơn) câu vấn tin gốc. Câu vấn tin gốc sẽ tốt hơn khi có quá nhiều nối từng phần trong câu vấn tin rút gọn. Điều này nảy sinh khi có rất ít các vị từ phân mảnh mâu thuẫn. Trường hợp xấu nhất xảy ra khi mỗi mảnh của quan hệ này phải được nối với mảnh của quan hệ kia. Như thế nó giống như lấy tích Descarrtes của hai tập mảnh, với mỗi tập tương ứng với một quan hệ.

Câu vấn tin rút gọn sẽ tốt hơn khi có ít các nối từng phần. Thí dụ, nếu cả hai quan hệ được phân mảnh nhờ cùng một vị từ, số lượng các nối từng phần bằng với số lượng các mảnh của mỗi quan hệ. Ưu điểm của câu vấn tin rút gọn là các nối từng phần có thể được thực hiện song song, và vì vậy làm giảm thời gian đáp ứng.

Thí dụ 5.32:

Giả sử rằng quan hệ EMP được phân mảnh thành EMP_1, EMP_2 và EMP_3 như trên và quan hệ ASG được phân mảnh như sau:

$$ASG_1 = ASG(ENO \leq "E3")$$
$$ASG_2 = ASG(ENO > "E3")$$

EMP_1 và ASG_1 được định nghĩa bởi cùng một vị từ. Ngoài ra vị từ định nghĩa ASG_2 là hợp của các vị từ định nghĩa EMP_2 và EMP_3. Bây giờ hãy xét câu vấn tin nối:

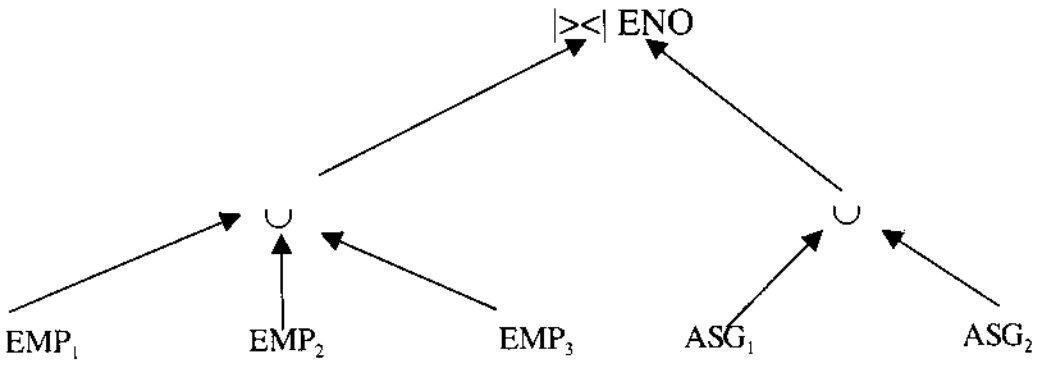
```
SELECT *
FROM    EMP, ASG
WHERE   EMP. ENO = ASG. ENO
```

Câu vấn tin gốc tương đương được trình bày trong hình 5.7a. Câu vấn tin được rút gọn bằng cách phân phối các nối trên các hợp và việc áp dụng quy tắc 2 có thể được cài đặt như hợp của ba nối từng phần được thực hiện song song (hình 5.7b).

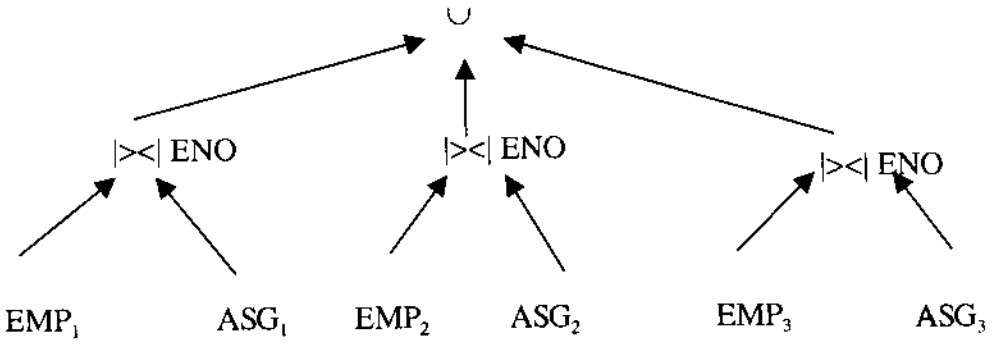
5.6.4,RÚT GỌN CHO PHÂN MẢNH DỌC

Công việc phân mảnh dọc phân tán một quan hệ dựa trên các thuộc tính chiếu. Bởi vì toán tử tái xây dựng các mảnh dọc là nối, chương trình cục

bộ hóa cho một quan hệ phân mảnh dọc gồm có nối của các mảnh theo thuộc tính chung. Đối với phân mảnh dọc, chúng ta sử dụng thí dụ sau:



(a) Vấn tin gốc



(b) Vấn tin đã rút gọn

Hình 5.7 Rút gọn cho phân mảnh ngang (với nối).

Thí dụ 5.33:

Quan hệ EMP có thể được phân chia thành hai mảnh dọc, trong đó thuộc tính khóa ENO được nhân đôi.

EMP_1 = EMP.{ENO, ENAME}

EMP_2 = EMP.{ENO, TITTLE}

Chương trình cục bộ hóa là

$$EMP = EMP_1 \underset{ENO}{|><|} EMP2$$

Tương tự như phân mảnh ngang, các vấn tin trên các mảnh dọc có thể được rút gọn bằng cách xác định các quan hệ trung gian vô dụng và loại bỏ các cây con đã sinh ra chúng. Phép chiếu trên một mảnh dọc không có thuộc tính chung với các thuộc tính chiếu (trừ khóa của quan hệ) sinh ra các quan hệ vô dụng, mặc dù không phải là quan hệ rỗng. Cho trước quan hệ r được định nghĩa trên các thuộc tính $A = \{A_1, \ldots A_n\}$ và được phân mảnh dọc thành $r_i = r(A')$, trong đó $A' \subseteq A$, quy tắc này có thể được phát biểu một cách hình thức như sau:

QUY TẮC 3:

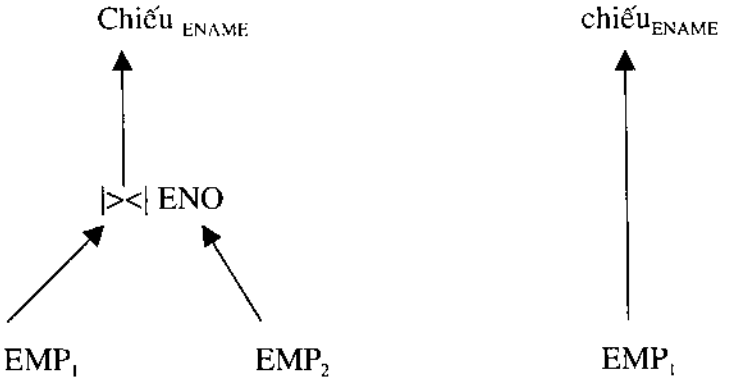
$r_i(D,A')$ là vô dụng nếu tập các thuộc tính chiếu D không thuộc A'

Thí dụ 5.34:

Chúng ta hãy minh hoạ cách áp dụng quy tắc này bằng cách sử dụng câu vấn tin mẫu trong SQL

```
SELECT      ENAME
FROM        EMP
```

Câu vấn tin gốc tương đương trên EMP_1 và EMP_2 (như trong thí dụ 5. 31) được cho trong hình 5.8a. Bằng cách hoán vị phép chiếu với phép nối (nghĩa là chiếu trên ENO, ENAME), chúng ta có thể thấy rằng chiếu trên EMP_2 là vô dụng bởi vì ENAME không thuộc EMP_2. vì thế phép chiếu chỉ cần thực hiện trên EMP_1 như được trình bày trong hình 5.8b.



Hình 5.8 Rút gọn cho phân mảnh dọc

5.6.5.RÚT GỌN CHO PHÂN MẢNH DẪN XUẤT

Như chúng ta đã thấy trong các phần trước rằng phép nối, có lẽ là phép toán quan trọng nhất vì nó vừa thường gặp lại vừa có chi phí cao, có thể được tối ưu hóa bằng cách dùng phân mảnh ngang nguyên thủy khi các quan hệ nối được phân mảnh theo các thuộc tính nối. Trong trường hợp này, nối của hai quan hệ được cài đặt như hợp của các nối từng phần. Tuy nhiên phương pháp này ngăn cản không cho một trong các quan hệ phân mảnh

theo một phép chọn trên một thuộc tính khác. Phân mảnh ngang dẫn xuất là một cách khác để phân phối hai quan hệ mà nhờ đó có thể cải thiện khả năng xử lý các điểm giao nhau giữa phép chọn và nối. Điển hình, nếu quan hệ r phải phân mảnh dẫn xuất theo quan hệ s, các mảnh của r và s có giá trị giống nhau ở thuộc tính nối sẽ nằm cùng vị trí. Ngoài ra s có thể được phân mảnh theo một vị từ chọn.

Bởi vì các bộ của r được đặt tùy theo các bộ của s, phân mảnh dẫn xuất chỉ được sử dụng cho mối liên hệ một - nhiều (phân cấp) từ s đến r, trong đó một bộ của s có thể khớp với nhiều bộ của r, nhưng một bộ của r chỉ khớp với đúng một bộ của s. Chú ý rằng phân mảnh dẫn xuất có thể được sử dụng cho các mối liên hệ nhiều - nhiều, với điều kiện là các bộ của s (là các bộ khớp với nhiều bộ của r) được nhân bản. Nhân bản như thế gây nhiều khó khăn cho việc duy trì tính nhất quán. Để cho đơn giản, chúng tôi giả sử và khuyên độc giả rằng phân mảnh dẫn xuất chỉ nên dùng cho mối liên hệ phân cấp.

Thí dụ 5.35:

Cho mối liên hệ một - nhiều từ EMP đến ASG, quan hệ ASG (ENO, PNO, RESP, DUR) có thể được phân mảnh gián tiếp theo các quy tắc sau:

$ASG_1 = ASG \mid\!><_{ENO} EMP_1$

$ASG_2 = ASG \mid\!><_{ENO} EMP_2$

độc giả chắc còn nhớ rằng từ chương 4, các mảnh ngang của EMP là:

$EMP_1 = EMP(TITLE = \text{"Programmer"})$

$EMP_2 = EMP(TITLE \neq \text{"Programmer"})$

Chương trình cục bộ hóa cho quan hệ phân mảnh ngang là hợp của các mảnh.

Trong thí dụ đang xét, chúng ta có:

$ASG = ASG_1 \cup ASG_2$

Các câu vấn tin trên các mảnh dẫn xuất cũng có thể được rút gọn. Bởi vì phân loại phân mảnh này rất hữu ích trong việc tối ưu hóa các câu vấn tin nối, một biến đổi có ích là phân phối các nối trên các hợp (được dùng trong chương trình cục bộ hóa) và áp dụng quy tắc 2 đã được trình bày ở trên. Bởi vì quy tắc phân mảnh chỉ rõ các bộ sẽ khớp với nhau, một số nối sẽ sinh ra các quan hệ rỗng nếu các vị từ phân mảnh có mâu thuẫn. Thí dụ, các vị từ của ASG_1 và EMP_2 mâu thuẫn; vì thế chúng ta có:

$ASG_1 |><| EMP_2 = \varnothing$

Ngược lại với rút gọn bằng nối được thảo luận trước đây, câu vấn tin rút gọn luôn được ưa chuộng hơn so với câu vấn tin gốc bởi vì số lượng các nối từng phần thường bằng với số mảnh của R.

Thí dụ 5.36:

Rút gọn theo các mảnh dẫn xuất được minh họa bằng cách áp dụng nó cho câu vấn tin SQL truy xuất tất cả các thuộc tính từ EMP và ASG có cùng giá trị của ENO và chức vụ là "Mech.Eng."

```
SELECT    *
FROM      EMP, ASG
WHERE     ASG.ENO = EMP.ENO
AND       TITLE = "Mech. Eng."
```

Câu vấn tin gốc trên các mảnh EMP_1, EMP_2, ASG_1, ASG_2 đã định nghĩa trước đây. Bằng cách đẩy phép chọn xuống các mảnh EMP_1 và EMP_2, câu vấn tin rút gọn lại do vị từ chọn mâu thuẫn với vị từ của EMP_1, và vì thế có thể loại bỏ EMP_1. Để phát hiện các vị từ nối mâu thuẫn, chúng ta phân

phối các nối trên các hợp. Kết quả là một cây . Cây con bên trái nối hai mảnh, ASG_1 và EMP_2 với các lượng từ hóa mâu thuẫn do vị từ

TITLE = "Programmer" trong ASG_1 và TITLE ≠ "Programmer" trong EMP_2. Vì thế chúng ta có thể loại bỏ cây bên trái và thu được câu vấn tin rút gọn . Thí dụ này minh hoạ cho giá trị của phân mảnh trong việc cải thiện hiệu năng của các câu vấn tin phân tán.

5.6.6. RÚT GỌN CHO PHÂN MẢNH HỖN HỢP

Phân mảnh hỗn hợp là tổ hợp của các phân mảnh được thảo luận ở trên. Mục tiêu là hỗ trợ hiệu quả các câu vấn tin có chứa phép chiếu, chọn và nối. Chú ý rằng tối ưu hóa một phép toán hoặc tổ hợp của các phép toán luôn đi kèm với việc tăng chi phí của các phép toán khác. Chẳng hạn như phân mảnh hỗn hợp dựa trên tổ hợp chọn - chiếu sẽ làm cho phép chọn hoặc phép chiếu kém hiệu quả hơn so với phân mảnh ngang (hoặc phân mảnh dọc). Chương trình cục bộ hóa cho một quan hệ phân mảnh hỗn hợp có sử dụng hợp và nối của các mảnh.

Thí dụ 5.37:

Đây là một thí dụ về phân mảnh hỗn hợp cho quan hệ EMP:

EMP_1 = (EMP.{ENO, ENAME}(ENO ≤ "E4")

EMP_2 = (EMP.{ENO, ENAME})(ENO > "E4")

EMP_3 = EMP.{ ENO, TITLE}

Trong thí dụ trên, chương trình cục bộ hóa là:

$EMP = (EMP_1 \cup EMP_2) \ |><|_{ENO} \ EMP_3$

Câu vấn tin trên các mảnh hỗn hợp có thể được rút gọn bằng cách tổ hợp các quy tắc tương ứng đã được dùng trong các phân mảnh ngang nguyên thủy, phân mảnh dọc và phân mảnh ngang dẫn xuất. Những quy tắc này có thể được tóm tắt như sau:

1- Loại bỏ các quan hệ rỗng được tạo ra bởi các phép chọn mâu thuẫn trên các mảnh ngang.

2- Loại bỏ các quan hệ vô dụng được tạo ra bởi các phép chiếu trên các mảnh dọc.

3- Phân phối các nối cho các hợp nhằm cô lập và loại bỏ các nối vô dụng.

Thí dụ 5.38:

Câu vấn tin SQL sau minh hoạ cách áp dụng các quy tắc 1- và 2- cho phân mảnh ngang - dọc của quan hệ EMP thành EMP_1, EMP_2, EMP_3 được cho ở trên:

```
SELECT    ENAME
FROM      EMP
WHERE     ENO = "E5"
```

Câu vấn tin gốc có thể được rút gọn bằng cách trước tiên đẩy chọn đi xuống, loại bỏ mảnh EMP_1 rồi đẩy phép chiếu xuống, loại bỏ mảnh EMP_3.

BÀI TẬP

5.1. Định nghĩa bằng cách dùng cú pháp tựa SQL một khung nhìn V(ENO, ENAME, PNO, RESP) của CSDL mẫu, trong đó thời gian (DUR) là 24. Khung nhìn V có cập nhật được không? Giả sử rằng quan hệ EMP và ASG được phân mảnh ngang dựa vào tần số truy xuất như sau:

Vị trí 1	Vị trí 2	Vị trí 3
EMP_1	EMP_2	
	ASG_1	ASG_2

Trong đó:

$EMP_1 = \sigma_{TITLE \neq "Engineer"}(EMP)$

$EMP_2 = \sigma_{TITLE \neq "Engineer"}(EMP)$

$ASG_1 = \sigma_{0 < DUR < 36}(ASG)$

$ASG_2 = \sigma_{DUR > 36}(ASG)$

Chúng ta nên lưu định nghĩa của V không nhân bản hoàn toàn tại những vị trí nào để làm tăng tính cục bộ tham chiếu?

5.2. Diễn tả câu vấn tin sau: Tìm tên các nhân viên trong khung nhìn V có làm việc cho dự án CAD/CAM.

5.3. Giả sử quan hệ PROJ được phân mảnh ngang như sau:

$PROJ_1 = \sigma_{PNAME="CAD/CAM"}(PROJ)$

$PROJ_2 = \sigma_{PNAME\neq"CAD/CAM"}(PROJ)$

Hãy sửa lại câu vấn tin thu được trong Bài tập 5.2 thành câu vấn tin được diễn tả trên các mảnh.

5.4*. Hãy tìm một thuật toán cập nhật hiệu quả một khoảnh khắc được dẫn

xuất bởi phép chiếu từ các quan hệ được phân mảnh ngang.

5.5. Hãy tìm một lược đồ quan hệ để lưu các quyền truy xuất đi kèm với các nhóm người sử dụng vào một hồ sơ cơ cấu của CSDL phân tán và đưa ra một lược đồ phân mảnh cho quan hệ đó, giả thiết rằng tất cả thành viên của một nhóm ở tại cùng một vị trí.

5.6.** Hãy tìm một thuật toán thực hiện câu lệnh REVOKE trong một DBMS phân tán, giả thiết rằng quyền GRANT chỉ có thể trao cho một người sử dụng với tất cả mọi thành viên đều ở cùng một vị trí.

5.7. Sử dụng ngôn ngữ đặc tả phán đoán của chương này, hãy biểu diễn một ràng buộc toàn vẹn khẳng định rằng thời gian được phân công cho một dự án không vượt quá 48 tháng.

5.8*. Định nghĩa các phán đoán biên dịch đi kèm với các ràng buộc toàn vẹn được đề cấp đến trong các thí dụ 5.5 đến 5.8

5.9. Giả sử có phân mảnh dọc của các quan hệ EMP, ASG và PROJ:

Vị trí 1	Vị trí 2	Vị trí 3	Vị trí 4
EMP_1	EMP_2		
	$PROJ_1$	$PROJ_2$	
		ASG_1	ASG_2

Trong đó:

EMP_1 = EMP.{ ENO,ENAME }

EMP_2 = EMP.{ ENO,TITLE}

$PROJ_1$ = PROJ.{ PNO,PNAME}

$PROJ_2$ = PROJ.{ PNO,BUDGET}

ASG_1 = ASG.{ ENO,PNO,RESP}

ASG_2 = ASG.{ ENO,PNO,DUR }

Cần lưu các phán đoán biên dịch thu được từ bài tập 5.8 ở đâu, giả thiết rằng có tính tự trị vị trí.

5.9. Hãy đơn giản hóa câu vấn tin được diễn tả trong SQL sau trên CSDL mẫu bằng cách sử dụng các quy tác lũy đẳng:

```
SELECT    ENO
FROM      ASG
WHERE     RESP = "Analyst"
AND       NOT (PNO = "P2" OR DUR = 12)
AND       PNO ≠ "P2"
AND       DUR = 12
```

5.10 Trình bày đồ thị vấn tin cho câu vấn tin SQL trên CSDL mẫu:

```
SELECT    ENAME, PNAME
FROM      EMP , ASG, PROJ
WHERE     DUR > 12
  AND EMP. ENO = ASG. ENO
```

và ánh xạ nó thành một cây toán tử.

5.11*. Đơn giản hóa câu vấn tin sau đây:

```
SELECT    ENAME, PNAME
FROM      EMP, ASG, PROJ
WHERE     DUR > 12
AND       EMP. ENO = ASG. ENO
AND       (TITLE = "Elect. Eng. "
OR        ASG.PNO < "P3")
AND       ASG. PNO = PROJ. PNO
```

và biến đổi nó thành một cây toán tử tối ưu bằng thuật toán tái cấu trúc.

5.12*. Biến đổi cây toán tử của hình 5.5 trở lại cây của hình 3.3 bằng cách dùng thuật toán tái cấu trúc. Mô tả mỗi cây trung gian và cho biết phép biến đổi dựa trên quy tắc nào.

5.13.** Xét câu vấn tin sau trên CSDL mẫu:

```
SELECT    ENAME, SAL
FROM      EMP, PROJ, ASG, PAY
WHERE     EMP,ENO = ASG. ENO
AND       EMP. TITLE = PAY. TITLE
AND       (BUDGET > 200000 OR DUR > 24)
AND       ASG. PNO = PROJ. PNO
```

Xây dựng vị từ chọn tương ứng với mệnh đề WHERE và biến đổi nó, bằng cách dùng các quy tắc lũy đẳng, thành một dạng tương đương đơn giản nhất. Ngoài ra hãy xây dựng một cây toán tử tương ứng với câu vấn tin này và biến đổi nó, bằng cách dùng quy tắc biến đổi đại số quan hệ, thành một dạng tối ưu ứng với tổng thời gian thực thi bằng cách chỉ xét các hệ số chọn của các phép toán.

5.14. Giả sử quan hệ PROJ của CSDL mẫu được phân mảnh ngang thành:

$PROJ_1$ = PROJ(PNO ≤ "P2")

$PROJ_2$ = PROJ(PNO > "P2")

Biến đổi câu vấn tin sau thành một vấn tin rút gọn trên các mảnh.

```
SELECT    BUDGET
FROM      PROJ, ASG
WHERE     PROJ. PNO  ASG. PNO
AND       ASG. PNO = "P4"
```

5.15*. Giả sử rằng quan hệ PROJ được phân mảnh ngang như trong bài tập 5.14 và quan hệ ASG được phân mảnh ngang thành:

ASG_1 = ASG(PNO ≤ "P2")

ASG_2 = ASG("P2" < PNO ≤ "P3")

ASG_3 = ASG(PNO > "P3")

Hãy biến đổi câu vấn tin sau thành một vấn tin rút gọn trên các mảnh và xác định xem nó có tốt hơn câu vấn tin gốc hay không.

```
SELECT    RESP, BUDGET
FROM      ASG , PROJ
WHERE     ASG. PNO = PROJ. PNO
AND       APNAME = "CAD/CAM".
```

5.16*. Giả sử rằng quan hệ PROJ được phân thành mảnh như bài tập 5.14. Ngoài ra quan hệ ASG được phân mảnh gián tiếp thành:

ASG_1 = ASG I><I $_{PNO}$ $PROJ_1$

ASG_2 = ASG I><I_{PNO} $PROJ_2$

và quan hệ EMP được phân mảnh dọc thành:

EMP_1 = EMP(ENO, ENAME)

EMP_2 = EMP(ENO, TITLE)

Biến đổi câu vấn tin sau thành một câu vấn tin rút gọn trên các mảnh.

```
SELECT    ENAME
FROM      EMP, ASG, PROJ
WHERE     PROJ. PNO = ASG. PNO
AND       PNAME = "Instrumentation"
AND       EMP. ENO = ASG. ENO.
```

CHƯƠNG 6

QUẢN LÝ GIAO DỊCH VÀ ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN

Như chúng ta đã biết trên mạng máy tính có nhiều ứng dụng đòi hỏi phải cho thực hiện nhiều chương trình cùng một lúc, hoặc nhiều tiến trình của cùng một chương trình. Chẳng hạn xét hệ thống đặt chỗ trên các chuyến bay của một hãng hàng không. Tại một thời điểm, nhiều đại lý có thể cùng bán vé, vì thế danh sách hành khách và số ghế của một chuyến bay sẽ bị thay đổi. Vấn đề mấu chốt là nếu chúng ta không thận trọng khi cho phép hai hay nhiều truy xuất CSDL, cùng một thời điểm chúng ta có thể bán một ghế hai lần. Trong hệ thống đặt chỗ, hai hoặc nhiều tiến trình đọc và thay đổi giá trị của cùng một đối tượng không được phép thực hiện cùng một lúc bởi vì chúng có thể cho ra kết quả không sát với thực tế.

Tuy nhiên trong một thí dụ khác về CSDL thống kê, chẳng hạn như dữ liệu điều tra dân số, trong đó có thể có nhiều người cùng vấn tin CSDL đồng thời. Trong trường hợp này vì không ai cần thay đổi dữ liệu, chúng ta thực sự không quan tâm đến thứ tự đọc dữ liệu; chúng ta có thể để cho hệ điều hành lo việc phân phối các yêu cầu đọc đồng thời. Ở tình huống chỉ có yêu cầu đọc, chúng ta muốn càng nhiều thao tác đọc cùng một lúc càng tốt, nhằm tiết kiệm thời gian. Ngược lại, trong trường hợp của hệ thống đặt chỗ, trong đó xảy ra cả hai quá trình đọc và ghi, chúng ta cần có phương pháp giải quyết riêng, khống chế không cho hai tiến

trình có thể thực hiện đồng thời, gây hậu quả sai về ngữ nghĩa và tính toàn vẹn của CSDL.

Trong chương này chúng ta sẽ xét các mô hình cho các *tiến trình xảy ra đồng thời* (concurrent processes) khi chúng cùng thao tác trên CSDL. Đối với mỗi mô hình, chúng ta sẽ giữ được tính toàn vẹn của CSDL bằng cách ngăn cản các thao tác đồng thời có khả năng phá huỷ tính toàn vẹn của CSDL. Như một quy tắc, mô hình càng chi tiết sẽ càng bảo đảm an toàn khi cho phép hoạt động đồng thời.

6.1. CÁC KHÁI NIỆM CƠ BẢN GIAO DỊCH

Giao dịch (transaction) là một lần thực hiện chương trình (một thực hiện). Đôi khi để biểu thị một giao dịch T ta viết T: begin . . . end. Giữa begin và end là những bước cơ bản của giao dịch. Chương trình này có thể là một câu vấn tin hoặc một chương trình ngôn ngữ chủ với các lời gọi được gắn vào một ngôn ngữ vấn tin. Có nhiều thực hiện độc lập của cùng một chương trình T được tiến hành đồng thời ở nhiều vị trí khác nhau trên mạng; mỗi thực hiện này là một giao dịch khác nhau.

Một giao dịch sẽ đọc dữ liệu và ghi dữ liệu vào CSDL(ta ngầm hiểu CSDL chung), qua một *vùng làm việc tư hữu* (private worrkspace), là một vùng của bộ nhớ để thực hiện tất cả các tính toán. Cụ thể là các tính toán do giao dịch thực hiện sẽ không có tác dụng trên CSDL cho đến khi các giá trị mới được ghi vào trong CSDL.

Thí dụ ta có giao dịch T :

begin read A; A:= A+100; read A; A:= A+2; WRITE A end

Khi đó trong CSDL A được tăng lên 2, vì phép toán A:= A+100 chỉ làm việc trong vùng tư hữu.
Ngược lại giao dịch T_1:

begin read A; A:= A+100; WRITE A; read A; A:=A+2; WITE A end
Khi đó trong CSDL A được tăng lên 102.

6.1.1. TÍNH NGUYÊN TỬ

Trên quan điểm để "quản lý được", *quản lý giao dịch* (transaction management) là *một cố gắng nhằm làm cho các thao tác phức tạp xuất hiện dưới dạng các nguyên tử (atomic).* Nghĩa là thao tác xảy ra trọn vẹn hoặc không xảy ra. Nếu xảy ra, không có biến cố hay giao dịch nào cùng xảy ra trong suốt thời gian tồn tại của nó. Mỗi nguyên tử về sau ta sẽ gọi là *một bước cơ bản hoặc một thao tác cơ bản.* Cách thông dụng nhằm đảm bảo được tính nguyên tử của các giao dịch là *phương pháp tuần tự hoá* (serialization). Phương pháp này là làm cho các giao dịch được thực hiện tuần tự. Một giao dịch không có tính nguyên tử nếu:

1. Trong một hệ thống phân chia thời gian (time shared system), lát thời gian (time - slice) cho giao dịch T có thể kết thúc trong khi T đang tính toán, và các hoạt động của một giao dịch khác sẽ được thực hiện trước khi T hoàn tất. Hoặc

2. Một giao dịch có thể không hoàn tất được. Chẳng hạn có khi nó phải chấm dứt giữa chừng, có thể vì nó thực hiện một phép tính không hợp lệ (ví dụ chia cho 0), hoặc có thể do nó đòi hỏi những dữ liệu không được quyền truy xuất. Bản thân hệ thống CSDL có thể buộc giao

dịch này ngừng lại vì nhiều lý do. Chẳng hạn giao dịch đó có thể bị kẹt trong một *khoá gài* (deadlock), v.v.

Trong trường hợp (1), nhiệm vụ của hệ thống CSDL là phải bảo đảm rằng cho dù có bất kỳ điều gì xảy ra ngay giữa một giao dịch, tác dụng của giao dịch trên CSDL không bị ảnh hưởng của những biến cố "bất ngờ" này. Trong trường hợp (2), hệ thống phải bảo đảm rằng giao dịch bị huỷ bỏ không có ảnh hưởng gì trên CSDL hoặc các giao dịch khác.

Trong thực tế, mỗi giao dịch đều có một chuỗi các bước cơ bản, như : *đọc* hay *ghi* một *mục dữ liệu* (item) vào CSDL và *thực hiện các tính toán số học đơn giản trong vùng làm việc riêng*, hoặc những bước sơ đẳng khác như các bước *khoá chốt* (lock) và *giải phóng khoá (unlock), uỷ thác (hoàn tất-commit)* giao dịch và có thể những bước khác nữa. Chúng ta sẽ luôn giả sử rằng những bước sơ đẳng này là nguyên tử. Thậm chí thao tác kết thúc *lát thời gian* xảy ra khi đang tính toán cũng có thể xem là nguyên tử, bởi vì nó xảy ra trong vùng làm việc cục bộ và không có gì có thể ảnh hưởng đến vùng làm việc đó cho đến khi giao dịch đang thực hiện dở phép tính số học được tái hoạt động trở lại.

6.1.2. MỤC DỮ LIỆU

Để quản lý các hoạt động đồng thời, CSDL phải được phân nhỏ thành các *mục dữ liệu* (item), đó là những đơn vị dữ liệu cần được truy xuất có điều khiển. Bản chất và kích thước các mục dữ liệu do nhà thiết kế hệ thống chọn lựa. Chẳng hạn trong mô hình dữ liệu quan hệ, chúng

ta có thể chọn các mục lớn như các quan hệ, hoặc các mục nhỏ như các bộ hay thành phần của các bộ. Chúng ta cũng có thể chọn lựa các mục có kích thước trung gian, như một khối của quan hệ (quan hệ con). Kích thước của các mục dữ liệu được hệ thống sử dụng gọi là độ *mịn* (granularity) của hệ thống. Một hệ thống được gọi là *hạt mịn* (fine - grained), nếu nó sử dụng các mục dữ liệu nhỏ và hệ thống là *hạt thô* (coarse - grained), nếu nó sử dụng các mục dữ liệu lớn.

Phương pháp thông dụng nhất để điều khiển việc truy xuất các mục là sử dụng *khoá chốt* (*lock*). Nói ngắn gọn, bộ *quản lý khoá chốt* (lock manager) là thành phần của DBMS chịu trách nhiệm theo dõi xem một mục *I* hiện có giao dịch nào đang đọc ghi vào các phần của *I* hay không. Nếu có thì bộ quản lý khoá chốt sẽ ngăn cản không cho các giao dịch khác truy xuất *I* trong trường hợp truy xuất (đọc hay ghi) có thể xảy ra xung đột, chẳng hạn như việc bán một ghế trên một chuyến bay hai lần.

Chọn độ hạt thô sẽ làm giảm đi tổng chi phí cần để duy trì các khoá chốt, bởi vì chúng ta cần ít chỗ để lưu các khóa, và chúng ta tiết kiệm được thời gian bởi vì hệ thống chỉ phải thực hiện rất ít hành động mở đóng khóa. Tuy nhiên, độ hạt mịn cho phép nhiều giao dịch hoạt động song song, bởi vì xác xuất các giao dịch yêu cầu khoá chốt trên cùng một mục sẽ thấp.

Tuỳ mục tiêu của bài toán chúng ta có thể chọn lựa kích thước thích hợp cho một mục dữ liệu sao cho các giao dịch trung bình truy xuất được một vài mục. Vì vậy trong một hệ thống quan hệ người ta hay lấy bộ làm mục dữ liệu. Nếu giao dịch cần lấy nhiều quan hệ, và như thế cần phải truy xuất tất cả các bộ của những quan hệ này thì tốt hơn chúng ta xem tất cả các bộ của một quan hệ như một mục dữ liệu.

Trong những phần sau chúng ta sẽ giả sử rằng khi một thành phần của một mục được hiệu chỉnh thì toàn bộ mục đều được hiệu chỉnh, nhận một giá trị duy nhất và không bằng với giá trị do những hiệu chỉnh khác thực hiện. Chúng ta đưa ra giả định này không chỉ để làm đơn giản việc mô hình hoá các giao dịch. Trong thực hành, chúng ta cần phải thực hiện rất nhiều công việc trên các thành phần của hệ thống để suy ra được những sự kiện như: kết quả của một lần hiệu chỉnh một mục làm cho mục đó có giá trị giống như một lần hiệu chỉnh trước đó. Hơn nữa, nếu hệ thống phải ghi nhớ xem phần nào của mục chưa bị thay đổi sau khi nó được hiệu chỉnh thì chúng ta có thể chia mục này thành nhiều mục nhỏ hơn.

6.1.3 KHOÁ CHỐT (LOCK)

Như chúng ta đã khẳng định, *khoá chốt* (lock) là một đặc quyền truy xuất trên một mục dữ liệu mà bộ quản lý khoá chốt có thể trao cho một giao dịch hay thu hồi lại. Mặc dù mô hình giao dịch khởi đầu của chúng ta chỉ có một loại khoá chốt, chúng ta sẽ phân tích nhiều mô hình phức tạp hơn với nhiều loại khoá chốt hơn. Thông thường tại mỗi thời điểm, chỉ có một tập con các mục bị khoá chốt, vì vậy bộ quản lý khoá chốt có thể lưu các khoá chốt hiện hành trong một *bảng khóa* (lock table) với các mẫu tin sau: *(I, L, T)*.

Ý nghĩa của mẫu tin *(I, L, T)* là giao dịch T có một khoá chốt kiểu L, trên mục I. Chúng ta sẽ thấy trong phần sau cùng một lúc có thể có nhiều giao dịch với các khoá chốt thuộc nhiều kiểu trên cùng một mục.

6.1.4. KIỂM SOÁT HOẠT ĐỘNG ĐỒNG THỜI BẰNG KHOÁ CHỐT

Để thấy được nhu cầu phải sử dụng khoá chốt (hay một cơ chế tương tự),ta hãy xét thí dụ sau đây.

Thí dụ 6.1:

Xét hai giao dịch T_1 và T_2. Mỗi giao dịch truy xuất một mục dữ liệu A được giả sử là mang giá trị số nguyên, rồi cộng thêm 1 vào A. Hai giao dịch này là các thực hiện của chương trình P dưới đây:

P: READ A: A: = A + 1; WRITE A;

Giá trị của A tồn tại trong CSDL. Với mỗi giao dịch P đọc A vào vùng làm việc tư hữu, cộng 1 vào giá trị này rồi ghi kết quả vào trong CSDL. Trong hình 6.1 chúng ta thấy hai giao dịch đang thực hiện theo kiểu xen kẽ, và chúng ta ghi nhận giá trị của A trong CSDL tại một bước.

Chúng ta nhận ra rằng mặc dù hai giao dịch đều đã cộng thêm 1 vào A, giá trị của A trong CSDL chỉ tăng 1. Vấn đề sẽ nghiêm trọng nếu A biểu thị số ghế đã bán của một chuyến bay.

Giải pháp thông dụng nhất cho vấn đề được trình bày trong thí dụ 6.1 là cung cấp một khoá chốt trên A (LOCK A). Trước khi đọc A, một giao dịch T phải khoá A lại, ngăn cản không cho giao dịch khác truy xuất A cho đến khi T hoàn tất xong thao tác trên A. Hơn nữa T khoá được mục A chỉ khi trước đó A không bị khoá bởi một giao dịch khác. Nếu A đã bị khoá, T phải đợi đến khi giao dịch kia mở khoá cho A.

A trong csdl	5	5	5	5	6	6
T_1	READA		A:= A+1			WRITE A
T_2		READ A		A:=A+1	WRITE A	
A trong vùng làm việc T_1	5	5	6	6	6	6
A trong vùng làm việc T_2	5	5	5	6	6	

Hình 6.1 Các giao dịch cho thấy nhu cầu phải khoá chốt mục dữ liệu A

Vậy để ngăn cản những trường hợp đáng tiếc xẩy ra ta phải dùng khoá chốt(LOCK). Như vậy trong những chương trình giao dịch phải có khoá chốt và mở khoá chốt (UNLOCK). Ta giả sử rằng một khoá chốt phải được đặt trên một mục trước khi đọc hay ghi nó, và các thao tác khóa chốt hành động như một *hàm đồng bộ hoá* (synchronization primitive). *Nghĩa là nếu một giao dịch khoá một mục đã được khoá, nó không thể tiến hành cho đến khi khoá này được giải phóng bằng một lệnh mở khoá, được thực hiện bởi giao dịch đang giữ khoá.* Ta cũng giả sử rằng mỗi giao dịch đều có thể mở được mọi khoá do chính nó khoá. Một *lịch biểu* (schedule) chứa các thao tác cơ bản của nhiều giao dịch tuân theo các quy tắc của khoá chốt được gọi là *hợp lệ* (legal).

Thí dụ 6.2:

Chương trình P của thí dụ 6.1 được viết với các khoá sau:

P: LOCK A; READ A; A: = A + 1; WRITE A; UNLOCK A;

Lại giả sử rằng T_1 và T_2 là hai thực hiện của P. Nếu T_1 bắt đầu trước, nó yêu cầu khoá chốt trên A. Giả sử rằng không có giao dịch nào

đang khoá A, bộ quản lý khoá chốt sẽ cho nó khoá chốt này. Bây giờ chỉ có T_1 mới có thể truy xuất A. Nếu T_2 bắt đầu trước khi T_1 chấm dứt thì khi T_2 thực hiện lệnh LOCK A, hệ thống buộc T_2 phải đợi. Chỉ khi T_1 thực hiện lệnh UNLOCK A, hệ thống mới cho phép T_2 tiến hành. Kết quả là điều bất thường trong thí dụ 6.1 không thể xảy ra; T_1 hoặc T_2 sẽ hoàn tất trước khi giao dịch kia bắt đầu, và kết quả chung của chúng là cộng 2 vào A.

6.1.5. KHOÁ SỐNG

Hệ thống trao và buộc khoá chốt các mục dữ liệu không thể hoạt động một cách tuỳ tiện, nếu không thì các hiện tượng không mong muốn có thể sẽ xảy ra. Giả sử như trong thí dụ 6.2 khi T_1 giải phóng khoá chốt trên A, trong khi T_2 đang đợi nhận khoá, một giao dịch T_3 khác cùng xin một khoá trên A, và T_3 được trao khoá này trước T_2. Rồi sau khi T_3 được trao khoá trên A thì lại có một giao dịch T_4 xin khoá trên A, v.v. Rất có thể T_2 phải đợi mãi và chẳng bao giờ nhận được khoá trên A.

Tình huống này được gọi là *khoá sống* (livelock). Vậy *khoá sống (livelock) trên mục A của giao dịch T là T không khoá được A vì A luôn bị khoá bởi một giao dịch khác.* Rất nhiều giải pháp đã được các nhà thiết kế hệ điều hành đề xuất để giải quyết vấn đề khoá sống, ví dụ chúng ta có thể yêu cầu hệ thống khi trao khoá phải ghi nhận tất cả các yêu cầu chưa được đáp ứng, và khi mục A được mở khoá thì trao khoá này cho giao dịch đã xin đầu tiên trong số những giao dịch đang đợi khoá A. Chiến lược "*đến trước được phục vụ trước*" loại bỏ được các khoá sống.

6.1.6. KHOÁ GÀI

Một vấn đề khác có thể xảy ra trong điều khiển các hoạt động đồng thời, đó là vấn đề *khoá gài* (deadlock). Một mục trong giao dịch này bị *gài* bởi giao dịch khác và ngược lại.

Thí dụ 6.3:

Giả sử chúng ta có hai giao dịch đồng thời T_1 và T_2 như sau:

T_1: LOCK A; LOCK B; UNLOCK A; UNLOCK B;

T_2: LOCK B; LOCK A; UNLOCK B; UNLOCK A;

T_1 và T_2 cùng có thể thực hiện một số tác vụ nào đó trên A và B Giả sử T_1 và T_2 được thực hiện cùng lúc. T_1 yêu cầu và được trao khoá trên A, còn T_2 yêu cầu và được trao khoá trên B. Do đó khi T_1 yêu cầu khoá trên B nó sẽ phải đợi vì T_2 đã khoá B. Tương tự khi T_2 yêu cầu khoá trên A, nó cũng buộc phải đợi vì T_1 đã khoá A. Kết quả là không một giao dịch nào tiếp tục hoạt động được: mỗi giao dịch đều phải đợi cho giao dịch kia mở khoá, và chúng đều phải đợi nhưng chẳng bao giờ nhận khoá như yêu cầu.

Như thế *khoá gài* (deadlock) là tình huống mà trong đó mỗi thành viên T_i của tập giao dịch T= { T_1, T_2, ... } đang đợi nhận khoá của một mục hiện đang bị khoá bởi một giao dịch khác trong tập T. Bởi vì mỗi giao dịch trong tập T đều đang đợi, nó không thể mở khoá cho mục mà một giao dịch khác đang cần, vì vậy tất cả vẫn tiếp tục đợi mãi mãi. Giống như khoá sống, ngăn cản khoá gài là một vấn đề quan trọng. Một số giải pháp cho vấn đề khoá gài là:

1. Yêu cầu các giao dịch phải đưa ra tất cả mọi yêu cầu khoá cùng một lúc, và bộ quản lý khoá trao tất cả các khoá cho chúng nếu được, hoặc không trao và cho giao dịch này đợi nếu một hay nhiều khoá đang được giữ bởi một giao dịch khác. Để ý rằng quy tắc này sẽ ngăn được khoá gài trong thí dụ 6.3 hệ thống sẽ trao các khoá trên A và B cho T_1 nếu giao dịch này yêu cầu trước; khi T_1 hoàn tất, T_2 sẽ được trao cả hai khoá.

2. Gán một thứ tự tuyến tính cho các mục, buộc tất cả các giao dịch phải xin khoá theo thứ tự này.

Rõ ràng, cách tiếp cận thứ nhất tránh được khoá gài và cách thứ hai cũng thế, mặc dù với cách tiếp cận thứ 2 lý do có vẻ không hiển nhiên.

Trong tình huống tổng quát cho cách giải quyết thứ 2, *hiện tượng khoá gài không xảy ra nếu các mục dữ liệu đã được gán thứ tự tuyến tính*. Thật vậy giả sử chúng ta có một tập T các giao dịch bị khoá gài và mỗi giao dịch T_i trong T đang đợi một giao dịch khác trong T mở khoá một mục A_i. Chúng ta cũng giả sử rằng mỗi T_i trong T giữ ít nhất một (và chỉ một) trong các mục A_i nếu không chúng ta có thể loại T_i khỏi T để có một khoá gài ít phần tử hơn. Gọi A_k là mục đầu tiên trong số A_i theo thứ tự đã nêu. Thế thì T_k, đang đợi A_k, và theo thứ tự ưu tiên tuyến tính T_k không phải đợi suy ra vô lý.

Một cách khác nhằm xử lý các khoá gài là không cần ngăn cản chúng. Và cứ sau mỗi khoảng thời gian nhất định, hệ thống sẽ kiểm tra yêu cầu khoá và tìm xem có xảy ra khoá gài hay không. Nếu chúng ta sử dụng *đồ thị chờ đợi* (waits - for graph), với các nút là các giao dịch và các cung $T_1 \rightarrow T_2$ biểu thị cho T_2 đang đợi nhận khoá trên một mục đang được T_1 giữ. Thế thì *mỗi chu trình trong đồ thị chờ sẽ biểu thị cho*

một khoá gài, và nếu không có chu trình thì kết luận không có khoá gài. Nếu một khóa gài được phát hiện, thì hệ thống phải khởi động lại, và tác dụng trên CSDL của giao dịch khoá gài đó phải được bỏ đi. Quá trình huỷ bỏ và tái khởi động có thể gặp rắc rối nếu chúng ta không biết được cách thức các giao dịch ghi vào CSDL trước khi chúng hoàn thành. Nội dung này được dành cho phần 6.8.

6.1.7. TÍNH KHẢ TUẦN TỰ CỦA CÁC LỊCH BIỂU

Bây giờ chúng ta chuyển sang vấn đề xử lý hoạt động đồng thời có liên quan đến các nhà thiết kế CSDL.

Giả sử chúng ta có một tập các giao dịch T = { T_1, T_2, T_3 . . .}. Chúng ta thấy ngay rằng nếu các giao dịch thực hiện *tuần tự* theo một thứ tự nào đó, giao dịch nọ sau giao dịch kia (*lịch biểu tuần tự*) thì các sự cố tranh chấp chắc chắn không xảy ra và trong CSDL chúng ta có một kết quả nào đó. Trong thí dụ 6.1 nếu cho thực hiện T_1 (hoặc T_2) trước rồi đến T_2 (hoặc T_1) thì chúng ta có một *lịch biểu tuần tự* và nhận được một kết quả là: giá trị 2 được thêm vào A.

Giả sử chúng ta có tập k giao dịch T = {T_1, T_2, ...,T_k}. Tương ứng với T ta có k! (k giai thừa) các lịch biểu tuần tự khác nhau. Bởi vậy chúng ta sẽ giả sử rằng hoạt động của các giao dịch đồng thời là *đúng đắn* nếu và chỉ nếu *tác dụng của nó giống như tác dụng có được của một lịch biểu tuần tự.*

Chúng ta định nghĩa một *lịch biểu* S (*schedule*) cho một tập các giao dịch T *là thứ tự (có thể xen kẽ) các bước cơ bản của các giao dịch (khoá, đọc, ghi, gán, v. v.) được thực hiện.*

Các bước của một giao dịch đã cho phải xuất hiện trong lịch biểu theo đúng thứ tự xảy ra trong giao dịch đó.

Vậy *một lịch biểu S của tập giao dịch T được gọi là hợp lệ và đúng đắn nếu các bước cơ bản của một giao dịch T_i đã cho phải xuất hiện trong lịch biểu S theo đúng thứ tự xảy ra trong giao dịch T_i và các bước cơ bản của các giao dịch tuân theo các quy tắc của khoá chốt.*

Giả sử T = { T_1, T_2, ..., T_k }là tập các giao dịch.

Gọi $T_i = \{t^i_1, t^i_2, \ldots t^i_{n1}\}$ với i = 1,2,...k là các bước cơ bản của giao dịch T_i. Khi đó số các bước cơ bản của tất cả các giao dịch trong T là n = n1+n2+...nk. Tất nhiên ni là số các bước cơ bản của T_i

Một lịch biểu S của các giao dịch trong T là *một hoán vị* của các bước cơ bản của các T_i. Trong tập T có n bước cơ bản nên ta có n! hoán vị khác nhau của các bước cơ bản. Vậy với tập giao dịch T ta có n! lịch biểu S khác nhau. Tuy nhiên, trong số đó có nhiều lịch biểu vô nghĩa vì không đúng đắn hoặc không hợp lệ.

Việc quản lý các giao dịch là quản lý các lịch biểu đúng đắn và hợp lệ tương đương với một lịch biểu tuần tự nào đó.

Lịch biểu được gọi là *khả tuần tự* (serializable) *nếu tác dụng của nó giống với tác dụng (tương đương) của một lịch biểu tuần tự.*

Hai lịch biểu được gọi là *tương đương* nếu chúng có kết quả giống nhau.

Chúng ta sẽ định nghĩa chính xác khái niệm về "tác dụng giống nhau" trong phần tiếp theo sau khi xem xét vài thí dụ.

Thí dụ 6.4:

Xét hai giao dịch T_1 và T_2 là thành phần của một chương trình kế toán. T_1 thực hiện chuyển 10 tiền từ tài khoản A sang tài khoản B và T_2 chuyển 20 tiền từ tài khoản B sang tài khoản C.

T_1: READ A; A: = A - 10; WRITE A; READ B; B: = B + 10; WRITE B;

T_2: READ B; B: = B - 20; WRITE B; READ C; C: = C + 20; WRITE C;

Rõ ràng mọi lịch biểu tuần tự đều có đặc tính bảo toàn tổng A + B + C.

Trong hình 6.2(a) chúng ta có một lịch biểu tuần tự, và trong hình 6.2(b) là một lịch biểu khả tuần tự, hình 6.2(c) trình bày một lịch biểu bất khả tuần tự vì kết quả của B không giống với 6.2(a). Chú ý rằng hình 6.2(c) thêm 10 vào B, chứ không phải trừ 10 khỏi B như kết quả thực sự, bởi vì T_1 đọc B trước khi T_2 ghi giá trị mới của B. Chúng ta có thể ngăn không cho lịch biểu của hình 6.2(c) xảy ra bằng cách yêu cầu tất cả giao dịch khoá B trước khi đọc nó.

T_1	T_2
READ A	
A:= A- 10	
WRITE A	
READ B	
B:=B+10	
WRITE B	
	READ B
	B:=B-20
	WRITE B
	READ C
	C:=C+20
	WRITE C

(a)

T_1	T_2
READ A	
	READB
A:=A-10	
	B:=B-20
WRITEA	
	WRITEB
READ B	
	READ C
B:=B+10	
	C:=C+20
WRITE B	
	WRITEC

(b)

T_1	T_2
READ A	
A: = A-10	
	READ B
WRITEA	
	B:=B-20
READ B	
	WRITE B
B:=B+10	
	READ C
WRITE B	
	C:=C+20
	WRITEC

(c)

Hình 6.2 ba lịch biểu tuần tự, khả tần tự, bất khả tuần tự.

Bạn đọc nên nhớ rằng chúng ta đã định nghĩa một lịch biểu là khả tuần tự nếu tác dụng của nó tương đương với tác dụng của một lịch biểu tuần tự. Trong trường hợp tổng quát, chúng ta làm thế nào để khẳng định được một lịch biểu là khả tuần tự ? chúng ta phải thực hiện tất cả các lịch biểu tuần tự rồi so sánh với lịch biểu đã cho. Ngoài ra với một lịch biểu đã cho ta có khẳng định được rằng nó luôn có một tác dụng cho các giá trị đầu vào khác nhau. Không thể kiểm chứng được hai lịch biểu có cùng tác dụng đối với tất cả mọi giá trị ban đầu của các mục vì có thể có vô số các giá trị ban đầu. Vậy trong thực hành, chúng ta sẽ phải đưa ra một số giả thiết để đơn giản hoá cho các thao tác được phép thực hiện đối với các mục.

Trong thực tế, qua các tính chất của đại số đơn thuần chúng ta có thể gặp một *lịch biểu là bất khả tuần tự nhưng nó cho cùng kết quả so với một lịch biểu tuần tự.* Tuy nhiên, những lỗi này không bao giờ tạo ra một kết quả không chính xác như một lỗi tai hại có thể gây ra. Trong những phần tiếp theo chúng ta sẽ sử dụng các mô hình chi tiết, cho phép chúng ta suy ra được nhiều lớp lịch biểu khả tuần tự và do đó cho phép nhiều hoạt động đồng thời hơn nhưng vẫn bảo đảm được tính đúng đắn.

6.1.8. BỘ XẾP LỊCH

Chúng ta đã nhận thấy rằng khi thực hiện hoạt động đồng thời, các giao dịch có thể gây ra tình trạng *khoá sống, khoá gài và vấn đề bất khả tuần tự.*

Để loại bỏ những vấn đề này, chúng ta sẽ dùng *bộ xếp lịch (scheduler) và các nghi thức (protocol).*

Bộ xếp lịch là thành phần của hệ thống CSDL, có vai trò làm trọng tài phân xử các yêu cầu đang có xung đột. Chẳng hạn chúng ta đã biết cách loại bỏ khoá sống của một bộ xếp lịch "đến trước phục vụ trước". Một bộ xếp lịch cũng có thể xử lý các khoá gài và tính bất khả tuần tự bằng cách:

1. Buộc một giao dịch phải đợi, chẳng hạn cho đến khi khoá đang được yêu cầu giải phóng.

2. Buộc một giao dịch ngừng lại và tái khởi động.

Có lẽ (2) là phương pháp không nên dùng bởi vì nó sẽ làm mất các kết quả đã được giao dịch thực hiện cho đến lúc đó. Tuy nhiên, khi bắt nhiều giao dịch phải đợi trong một khoảng thời gian dài có thể làm cho quá nhiều khoá trở thành "không sẵn sàng", bởi vì các giao dịch đang chờ đợi có thể đã giữ một số khoá nào đó. Điều đó lại làm cho xác suất xảy ra khoá gài cao hơn, và có thể làm chậm trễ nhiều giao dịch đến nỗi tác dụng của nó có thể nhận ra được, chẳng hạn đối với một người sử dụng đang đứng đợi tại một máy rút tiền tự động. Trong trường hợp đã xảy ra khoá gài, thường không có cách lựa chọn nào ngoại trừ phải bắt ít nhất một trong các giao dịch đang vướng vào khoá gài phải ngưng hoạt động (huỷ bỏ giao dịch abort).

6.1.9.NGHI THỨC

Chúng ta cũng có thể sử dụng một công cụ khác để xử lý khoá gài và tính bất khả tuần tự. Công cụ đó chính là các nghi thức (protocol) mà tất cả các giao dịch phải tuân theo.

Một nghi thức, theo nghĩa tổng quát nhất, chỉ là một hạn chế trên chuỗi các bước cơ bản mà một giao dịch có thể thực hiện.

Chiến lược tránh khoá gài bằng cách yêu cầu khoá chốt trên các mục theo một thứ tự tuyến tính nào đó chính là một nghi thức.

6.2. MÔ HÌNH GIAO DỊCH ĐƠN GIẢN

Chúng ta sẽ bắt đầu bằng một mô hình giao dịch đơn giản, nó giúp chúng ta hình dung về các khoá chốt và tính tuần tự. Trong mô hình này, một giao dịch được xem như là một chuỗi các câu lệnh khoá chốt (lock) và mở khoá (unlock). Mỗi mục được khoá phải được mở khoá sau đó. Giữa một bước LOCK A và UNLOCK A kế tiếp, giao dịch sẽ được coi là đang giữ một khoá trên A. Chúng ta giả sử rằng một giao dịch sẽ không yêu cầu khoá một mục nếu nó hiện đang giữ khoá của mục đó, hoặc mở một mục mà nó hiện không giữ khoá trên mục đó.

Ngoài ra chúng ta còn giả sử rằng khi một giao dịch khoá một mục A, nó sẽ đọc và ghi A. Nghĩa là mỗi bước LOCK kéo theo thao tác đọc và mỗi bước UNLOCK kéo theo thao tác ghi.

6.2.1.Ý NGHĨA CỦA GIAO DỊCH- HÀM ĐẶC TRƯNG

Về nguyên tắc, ý nghĩa của giao dịch T_i chính là tác dụng của chương trình P tương ứng trên CSDL.

Về hình thức, chúng ta gán một *hàm f đặc trưng* cho mỗi cặp LOCK A và UNLOCK A.

Hàm f nhận đối là các giá trị của tất cả các mục bị khoá bởi giao dịch T trước khi mở khoá cho A, và giá trị của f là giá trị mới của A sau khi mở khoá A. Chú ý rằng một giao dịch có thể có nhiều hàm

như thế đối với một mục A, bởi vì chúng ta có thể khoá và mở khoá một mục A nhiều lần.

Vậy gọi A_0 là giá trị ban đầu của A trước khi các giao dịch bắt đầu thực hiện, như vậy giá trị của hàm $f(A_0, ...)$ sau khi UNLOCK A là giá trị mới cuả A. Một cách tổng quát gọi A là giá trị của mục A trước mô men lock A thì giá trị mới của A sau mô men mở khoá A là f(A,...),ta ký hiệu: lock A...unlock A f(A,...).

Các giá trị mà A có thể nhận (trong CSDL) trong khi thực hiện giao dịch là những công thức (biểu thị giá trị của f) được xây dựng bằng cách áp dụng những hàm này cho các giá trị ban đầu của các mục.

Hai công thức khác nhau được coi là những giá trị khác nhau.

Hai lịch biểu là *tương đương* nếu các công thức cho giá trị cuối cùng của mỗi mục giống nhau trong cả hai lịch biểu. Ta xét thí dụ :
Cho ba giao dịch T_1, T_2, T_3 như sau:

T_1	T_2	T_3
LOCK A	LOCK B	LOCK A
LOCK B	LOCK C	LOCK C
UNLOCK A f_1 (A, B)	UNLOCK B f_3 (B, C)	UNLOCK C f_6 (A, C)
UNLOCK B f_2 (A, B)	LOCK A	UNLOCK A f_7 (A, C)
	UNLOCK C f_4 (A, B, C)	
	UNLOCK A f_5 (A, B, C)	

Hình 6.3 Ba giao dịch T_1,T_2,T_3

Thí dụ 6.5:

Trong hình 6.3 chúng ta có ba giao dịch và những hàm đặc trưng có liên quan với mỗi cặp LOCK - UNLOCK; là những hàm xuất hiện trên cùng một dòng với UNLOCK. Chẳng hạn f_1, nhận A và B làm đối số,

bởi vì đây là những mục trong LOCKA – UNLOCKA của T_1 . Hàm f_3 chỉ nhận B và C làm đối số bởi vì T_2 mở khoá B, và trong LOCK – UNLOCKB có B và C. . .

Hình 6.4 trình bày một lịch biểu của những giao dịch này và tác dụng của chúng trên các mục A, B và C. Chúng ta có thể nhận xét rằng lịch biểu này không có đặc tính khả tuần tự. Thật vậy, giả sử rằng nó khả tuần tự. Thế thì nếu T_1 thực hiện trước T_2 trong lịch biểu tuần tự, giá trị cuối cùng của B sẽ là: $f_3(f_2(A_0,B_0),C_0)$ chứ không phải $f_2(A_0, f_3(B_0,C_0))$ như trong hình 6.5 và ta dễ dàng kiểm tra lại rằng không có lịch biểu tuần tự nào có công thức tính B cuối cùng giống $f_2(A_0, f_3(B_0,C_0))$.

Chúng ta thấy rằng giả thiết các công thức của hàm f sinh ra một giá trị duy nhất là mấu chốt của lập luận trong thí dụ 6.5. Thật vậy điều gì sẽ xảy ra chẳng hạn nếu có thể tồn tại:

$$f_3(f_2(A_0, B_0), C_0) = f_2(A_0, f_1(B_0, C_0)) \qquad (6.1)$$

Trong trường hợp này chúng ta không thể kết luận lịch biểu bất khả tuần tự được.

Trong hình 6.5 để cho gọn chúng ta sẽ qui ước như sau:

(i) $= f_4(f_1(A_0,f_3(B_0,C_0)),B_0,C_0)$.

(ii) $= f_5(f_1(A_0,f_3(B_0,C_0)),B_0,C_0)$.

(iii) $= f_6((ii), (i))$.

(iv) $= f_7((ii), (i))$.

Các giá trị này đã được tính trước.

Để hiểu thêm về cách sử dụng hàm đặc trưng các bạn có thể tự tính lại các giá trị mới của A, B, C và những công thức này như một bài tập thực hành.

Bước	Thao tác	A	B	C
(1)	T_1: LOCK A	A_0	B_0	C_0
(2)	T_2: LOCK B	A_0	B_0	C_0
(3)	T_2: LOCK C	A_0	B_0	C_0
(4)	T_2: UNLOCK B	A_0	$f_3(B_0,C_0)$	C_0
(5)	T_1: LOCK B	A_0	$f_3(B_0,C_0)$	C_0
(6)	T_1: UNLOCK A	$F_1(A_0,f_3(B_0,C_0))$	$f_3(B_0,C_0)$	C_0
(7)	T_2: LOCK A	$F_1(A_0,f_3(B_0,C_0))$	$f_3(B_0,C_0)$	C_0
(8)	T_2: UNLOCK C	$F_1(A_0,f_3(B_0,C_0))$	$f_3(B_0,C_0)$	(i)
(9)	T_2: UNLOCK A	(ii)	$f_3(B_0,C_0)$	(i)
(10)	T_3: LOCK A	(ii)	$f_3(B_0,C_0)$	(i)
(11)	T_3: LOCK C	(ii)	$f_3(B_0,C_0)$	(i)
(12)	T_1: UNLOCK B	(ii)	$f_2(A_0,f_3(B_0,C_0))$	(i)
(13)	T_3: UNLOCK C	(ii)	$f_2(A_0,f_3(B_0,C_0))$	(iii)
(14)	T_3: UNLOCK A	(iv)	$f_2(A_0,f_3(B_0,C_0))$	(iii)

Hình 6.4 Một lịch biểu của 3 giao dịch T_1,T_2,T_3

Cần lưu ý rằng phép kiểm tra tính khả tuần tự bằng hàm đặc trưng là một *phương pháp phức tạp* với những tập nhiều gao dịch và nhiều thao tác. Hơn nữa phép kiểm tra này là một *phương pháp yếu* vì hai công thức của các hàm đặc trưng có thể khác nhau nhưng giá trị của chúng có thể giống nhau. Hai công thức bằng nhau khi chúng giống nhau là một hạn chế rất lớn.

Thí dụ xét lịch biểu S của hai giao dịch T_1 và T_2 như sau:

Giả sử A_0, B_0, C_0 là các giá trị của các mục A, B, C trước khi thực hiện các giao dịch T_1, T_2.

Bước	T_1	T_2
(1)	Lock A	
(2)	Unlock A $f_1(A)$	
(3)		Lock A
(4)	Lock B	
(5)	Lock C	
(6)	Unlock C $f_2(C)$	
(7)		LockC
(8)	Unlock B $f_3(B,C)$	
(9)		Unlock A $f_4(A,C)$
(10)		Unlock C $f_5(A,C)$

Ta dễ dàng thấy rằng S là lịch *biểu khả tuần tự* vì đồ thị không có chu trình (xem phần tiếp theo). Tuy nhiên, ở đây sau khi thực hiện S ta có giá trị mới của A, B, C tương ứng là:

$A = f_4(A,C) = f_4(A,f_2(C_0)) = f_4(f_1(A_0),f_2(C_0))$.

$B = f_3(B,C) = f_3(B,f_2(C_0)) = f_3(B_0,f_2(C_0))$.

$C = f_5(A,C) = f_5(f_4(f_1(A_0),f_2(C_0)),C) = f_5(f_4(f_1(A_0),f_2(C_0)),f_2(C_0))$.

Sau khi thực hiện lịch biểu tuần tự T theo thứ tự T_2 trước T_1 ta có:

$A = f_4(A,C) = f_4(A_0,C_0)$.

$B = B_0$.

$C = f_5(A,C) = f_5(f_4(A_0,C_0),C_0)$.

Đến đây chúng ta chưa thể kết luận được điều gì cho lịch biểu S, vì T và S không tương đương. Muốn kết luận cho lịch biểu S chúng ta phải thực hiện tiếp các lịch biểu tuần tự còn lại.

Sau khi thực hiện lịch biểu S' tuần tự T_1 trước T_2 ta có:

$A = f_4(A,C) = f_4(A,f_2(C_0)) = f_4(f_1(A_0),f_2(C_0))$.

$B = f_3(B,C) = f_3(B,f_2(C_0)) = f_3(B_0,f_2(C_0))$.

$C = f_5(A,C) = f_5(f_4(f_1(A_0),f_2(C_0)),C) = f_5(f_4(f_1(A_0),f_2(C_0)),f_2(C_0))$.

Như vậy giá trị của các mục dữ liệu giống nhau sau khi thực hiện hai lịch biểu S' và S. Đến đây ta có thể khẳng định S khả tuần tự.

Thật may là chúng ta chỉ có hai giao dịch, nếu ta có nhiều giao dịch, thí dụ số các giao dịch là10 thì số các lịch biểu tuần tự là 10! Và vì vậy để khẳng định được S có khả tuần tự không chúng ta không thể thực hiện theo phương pháp hàm đặc trưng này được, vì chúng ta phải tìm cho được một lịch biểu tuần tự tương đương. Sau đây chúng ta sẽ xét một phương pháp đơn giản hơn. *Phương pháp kiểm tra tính khả tuần tự bằng đồ thị có hướng.*

6.2.2.KIỂM TRA TÍNH KHẢ TUẦN TỰ BẰNG ĐỒ THỊ CÓ HƯỚNG

Để xác định rằng một bộ xếp lịch nào đó là đúng, chúng ta phải chứng minh rằng mỗi lịch biểu S được nó cho phép đều có đặc tính khả tuần tự. Vì vậy chúng ta cần có một phép kiểm tra đơn giản về tính khả tuần tự của một lịch biểu. Sau đây là thuật toán kiểm tra tính khả tuần tự bằng đồ thị có hướng.

Thuật toán 6.1: *Kiểm tra tính khả tuần tự một lịch biểu*

Input: Một lịch biểu S cuả một tập các giao dịch T_1 ... T_k

Output: Khẳng định S có khả tuần tự hay không. Nếu có thì đưa ra một lịch biểu tuần tự tương đương với S.

Phương pháp:

Tạo một đồ thị có hướng G (được gọi là đồ thị tuần tự hoá-serialization graph), có các nút là các giao dịch T_i. Để xác định các cung của đồ thị G, gọi { a_1; a_2; ... a_n } là tập các bước cơ bản trong S. Trong đó mỗi a_i là một thao tác có dạng (xem lịch biểu hình 6.5)

T_j: LOCK A hoặc T_j: UNLOCK A

Nếu a_j là thao tác kiểu:

T_j: UNLOCK A

Thì tìm thao tác a_p kế tiếp sau a_j có dạng T_s: LOCK A. Nếu có một cặp thao tác như thế và s ≠ j, chúng ta vẽ một cung từ T_j đến T_s ($T_j \rightarrow T_s$). Cung này có ý nghĩa là trong một lịch biểu tuần tự tương đương với S, T_j phải đi trước T_s.

Nếu G có một chu trình thì S là bất khả tuần tự, ngược lại nếu G không có chu trình thì S là khả tuần tự và chúng ta tìm một thứ tự tuyến tính cho các giao dịch cho T_i bằng một quá trình gọi là sắp xếp topo của đồ thị G như sau:

6.2.3. SẮP XẾP TOPO ĐỒ THỊ CÓ HƯỚNG G KHÔNG CÓ CHU TRÌNH

Ta biết rằng trong G phải có một nút T_i nào đó không có cung đến, nếu không G có một chu trình. Liệt kê T_i rồi loại T_i ra khỏi G. Sau đó lặp lại quá trình này trên đồ thị còn lại cho đến khi không còn nút nào nữa. Thứ tự các nút được liệt kê trong danh sách là một thứ tự tuần

tự của các giao dịch. Thứ tự đó tạo nên lịch biểu tuần tự tương đương với S.

Thí dụ 6.6 a:

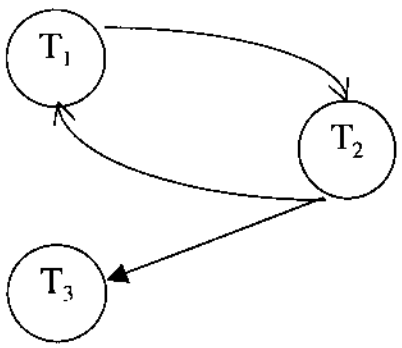
Xét lịch biểu của hình 6.5. Đồ thị G, được trình bày trong hình 6.6 có các nút T_1, T_2 và T_3. Để tìm các cung, chúng ta xét mỗi bước ULNLOCK trong hình 6.4. Chẳng hạn bước (4).

T_2 : UNLOCK B

đi theo sau là T_1 : LOCK B. Trong trường hợp này, thao tác khoá xảy ra ở bước kế tiếp. Do đó chúng ta vẽ một cung $T_2 \rightarrow T_1$. Thí dụ khác, hành động tại bước (8).

T_2 : UNLOCK C

theo sau là T_3 : LOCK C tại bước (11), và không có bước LOCK C nào nằm giữa hai bước này. Vì vậy chúng ta vẽ một cung từ $T_2 \rightarrow T_3$. Bước (6) và (7) đặt một cung $T_1 \rightarrow T_2$. Bởi vì G có một chu trình, lịch biểu của hình 6.4 bất khả tuần tự.



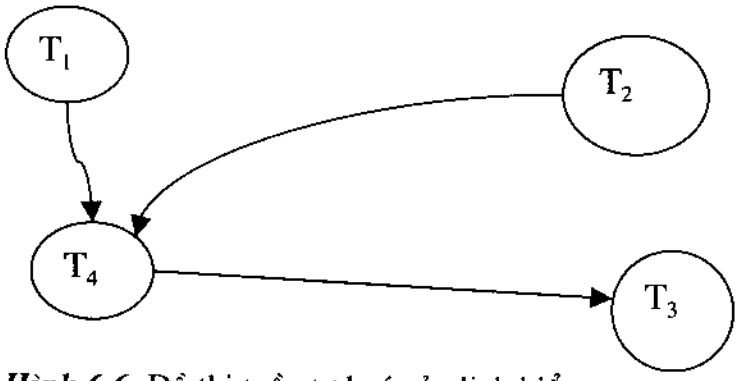
Hình 6.5 Đồ thị trong thuật toán 6.1, ứng với hình 6.4

Thí dụ 6.6b:

Cho lịch biểu S của tập các giao dịch T = {T_1,T_2, T_3, T_4}

Bước	T_1	T_2	T_3	T_4
(1)	lock A			
(2)		lock B		
(3)			lock C	
(4)			lock D	
(5)	unlock A			
(6)		unlock B		
(7)				lock A
(8)				lock B
(9)				unlock A
(10)				unlock B
(11)			lock B	
(12)			unlock D	
(13)			unlock C	
(14)			unlock B	

Đồ thị tuần tự hoá của lịch biểu là:

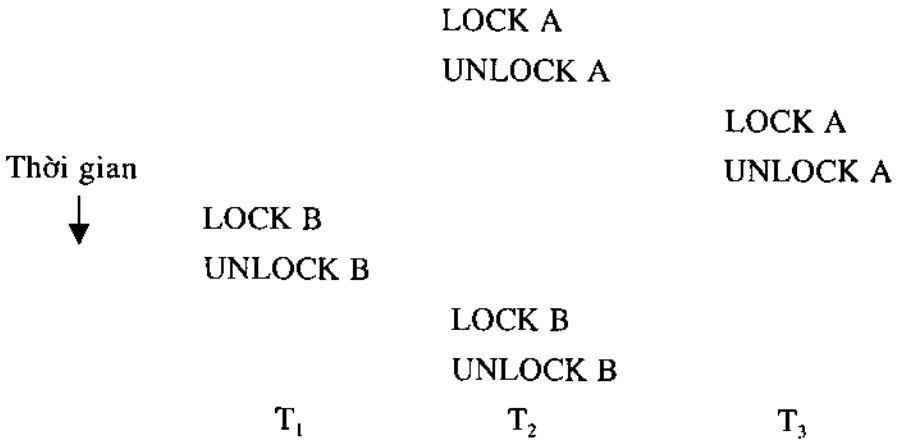


Hình 6.6 Đồ thị tuần tự hoá của lịch biểu

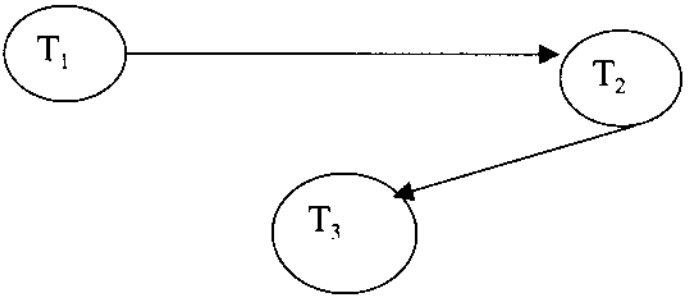
Đồ thị không có chu trình nên lịch biểu là khả tuần tự.

Thí dụ 6.7:

Trong hình 6.7 chúng ta có một lịch biểu cho ba giao dịch và hình 6.8 trình bày đồ thị tuần tự hoá của nó. Bởi vì không có chu trình, lịch biểu của hình 6.7 khả tuần tự, và thuật toán 6.1 cho chúng ta biết rằng thứ tự tuần tự này là T_1, T_2, T_3. Điều đáng chú ý là theo thuật toán 6.1 cho chúng ta biết rằng thứ tự tuần tự này là T_1, T_2, T_3 và theo thứ tự tuần tự này, T_1 đi trước T_3 dù rằng trong hình 6.7 T_1 sẽ không bắt đầu khi T_3 chưa chấm dứt (hai hành động của T_1 sau hai hành động của T_3).



Hình 6.7 Một lịch biểu khả tuần tự



Hình 6.8 đồ thị tuần tự hoá của 3 giao dịch

Định lý 6.1:

Thuật toán 6.1 xác định đúng một lịch biểu có khả tuần tự hay không.

Chứng minh của định lý này các bạn có thể thực hiện như một bài tập hoặc xem phần chứng minh trang 553 tập 2 trong [2].

Thí dụ 6.8:

Giả sử ta có một giao dịch T_0 khoá các mục A và B, và trong một lịch biểu S, mục A bị khoá lần lượt bởi T_1 và T_2 rồi đến T_0, còn mục B bị khoá bởi T_3, T_1, T_4 và T_0 theo thứ tự đó (các giao dịch khác có thể khoá A và B sau T_0). Hình 6.8 trình bày một cách thay đổi các giá trị của A và B, trong cả S và một lịch biểu tuần tự R tương đương.

$A_0 \rightarrow \ldots T_1 \rightarrow T_2 \rightarrow \ldots T_0$

$B_0 \rightarrow T_3 \rightarrow T_1 \rightarrow \ldots T_4 \rightarrow T_0$

Hình 6.8 Các giao dịch làm thay đổi giá trị của A và B

Chúng ta có thể chắc chắn rằng trong R, T_2 đi trước T_0, và không có giao dịch nào khoá A có thể nằm giữa T_2 và T_0. Lý luận tương tự cho B, chúng ta thấy rằng T_4 đi trước T_0 và không có giao dịch nào khoá B nằm giữa T_4 và T_0. Giá trị được ghi bởi T_0 cho A chỉ phụ thuộc vào những giá trị của A và B ngay trước đó, và chúng ta biết rằng những giá trị này là những giá trị được ghi bởi T_2 và T_4 trong cả R và S.

6.3. NGHI THỨC KHOÁ CHỐT HAI PHA

Chúng ta cần phải hiểu rõ những điều kiện để một lịch biểu là khả tuần tự nhằm tìm được một bộ xếp lịch và một nghi thức, đảm bảo rằng mọi lịch biểu mà chúng ta cho phép đều khả tuần tự. Trong phần này chúng ta giới thiệu một cách tiếp cận đơn giản và phổ dụng nhất.

Nghi thức, *được gọi là nghi thức hai pha (two phase protocol)*, nếu mọi giao dịch thực hiện tất cả các thao tác khoá trước tất cả mọi thao tác mở khoá. Các *giao dịch tuân theo nghi thức này được gọi là các giao dịch hai pha; pha đầu là pha khoá, và pha thứ hai là pha mở khoá.* Chẳng hạn, trong các hình 6.4 và 6.7, T_1 và T_3 là các giao dịch hai pha còn T_2 thì không.

Nghi thức hai pha có đặc điểm là mọi tập giao dịch tuân theo nghi thức này đều có các lịch biểu khả tuần tự. Trước tiên chúng ta sẽ chứng minh rằng nghi thức hai pha bảo đảm được tính khả tuần tự.

Định lý 6.2.

Nếu S là một lịch biểu của các giao dịch hai pha thì S khả tuần tự.

Chứng minh:

Giả sử khẳng định trên không đúng. Theo Định lý 6.1. đồ thị tuần tự hoá G của S có một chu trình.

$$T_{i1} \rightarrow T_{i2} \rightarrow \ldots \rightarrow T_{ip} \rightarrow T_{i1}$$

Do đó có một thao tác khoá do T_{i2} sẽ đi sau một thao tác mở khoá do T_{i1}; một thao tác khoá do T_{i3} sẽ đi sau một thao tác mở khoá T_{i2}, v.v. Cuối cùng một thao tác khoá do T_{i1} sẽ đi sau một thao tác mở khoá do T_{i1}, mâu thuẫn với giả thiết T_{i1} là giao dịch hai pha.

6.4. MÔ HÌNH KHOÁ ĐỌC VÀ KHOÁ GHI

Trong phần 6.2 chúng ta đã giả sử rằng mỗi lần giao dịch khoá một mục thì nó sẽ thay đổi mục đó. Trong thực hành, nhiều khi một giao dịch chỉ cần lấy giá trị của một mục nhưng không thay đổi giá trị đó. Nếu chúng ta phân biệt một truy xuất chỉ đọc (read - only) và một truy xuất đọc - ghi (read - write), chúng ta có thể phát triển một mô hình chi tiết hơn cho các giao dịch loại đọc ghi này. Trong một số trường hợp mô hình này cho phép sử dụng một số hoạt động đồng thời đã bị cấm trong mô hình của phần 6.2 ví dụ như nhiều giao dịch có thể cùng giữ khoá đọc một mục A. Chúng ta hãy phân biệt hai loại khoá.

1. Khoá đọc RLOCK (read - lock). Một giao dịch T chỉ đọc một mục A sẽ thực hiện lệnh RLOCK A, ngăn không cho bất kỳ giao dịch khác ghi giá trị mới của A trong khi T đã khoá A. Tuy nhiên các giao dịch khác vẫn có thể giữ một khoá đọc trên A cùng lúc với T.

2. Khoá ghi WLOCK (write - lock). Đây là những khoá theo nghĩa của phần 6.2. Một giao dịch muốn thay đổi giá trị của mục A đầu tiên sẽ lấy khoá ghi bằng cách thực hiện lệnh WLOCK A. Khi một giao

dịch đang giữ một khoá ghi trên một mục, những giao dịch khác không thể lấy được khoá đọc hoặc khoá ghi trên mục đó.

Như vậy khoá *đọc* mục A chỉ cấm các giao dịch khác *ghi* dữ liệu vào A, còn khoá *ghi* trên mục A cấm các giao dịch khác cả *ghi hoặc đọc* trên A.

Cả hai khoá đọc và khoá ghi đều được mở bằng lệnh UNLOCK. Giống như trong phần 6.2. Chúng ta cũng giả sử không có giao dịch nào cố mở khoá một mục mà nó hiện không khoá mục đó(đọc hay ghi). Hơn nữa một giao dịch cũng không cố khoá ghi một mục nếu nó đã giữ một khoá ghi trên mục đó, tuy nhiên nó có thể khoá ghi một mục mà nó hiện giữa khoá đọc.

6.4.1.Ý NGHĨA CỦA GIAO DỊCH VỚI KHOÁ ĐỌC VÀ KHÓA GHI

Giống như trong phần 6.2, chúng ta giả sử rằng mỗi lần áp dụng khóa ghi cho một mục A sẽ có một hàm duy nhất đi kèm với khoá đó, và nó tạo ra một giá trị mới cho A; hàm đó phụ thuộc vào tất cả các mục bị khoá trước khi mở khoá A. Tuy nhiên chúng ta cũng giả sử rằng một khoá đọc trên A không làm thay đổi A.

Cũng giống như trong phần 6.2, chúng ta giả sử rằng mỗi mục A có một giá trị ban đầu là A_0, và tác dụng của một lịch biểu trên CSDL có thể được diễn tả bởi những công thức của các hàm đặc trưng f, là những giá trị của mỗi mục được ghi ít nhất là một lần bởi các giao dịch. Tuy nhiên vì có thể có một giao dịch đọc các mục nhưng không ghi gì hoặc đọc một số mục chỉ sau khi ghi vào lần cuối cùng, vì thế những giá trị mà mỗi mục đang có khi một giao dịch chỉ đọc nó cũng được xử lý

như giá trị cũ. Vì vậy chúng ta có thể nói hai *lịch biểu là tương đương* nếu.

1. Chúng sinh ra cùng một giá trị cho mỗi mục và

2. Mỗi khoá đọc được áp dụng bởi mỗi giao dịch (ở vị trí tương ứng) trong cả hai lịch biểu vào những lúc mục bị khoá có cùng giá trị.

6.4.2.ĐỒ THỊ TUẦN TỰ HOÁ TRONG CÁC GIAO DỊCH RLOCK & WLOCK

Chúng ta hãy xét những điều kiện mà trong đó, từ ý nghĩa của các giao dịch và các lịch biểu, ta có thể suy ra được khi nào một giao dịch phải đi trước một giao dịch khác trong một lịch biểu tuần tự tương đương. Giả sử rằng có một lịch biểu S trong đó một khoá ghi mục A bởi giao dịch T_1, và gọi f là hàm đi kèm với khoá ghi đó. Sau khi T_1 mở khoá A, gọi T_2 là một trong những giao dịch kế tiếp nhận khoá đọc A trước khi một giao dịch khác nhận khoá ghi A. Chắc chắn rằng T_1 phải đi trước T_2 trong một lịch biểu tuần tự tương đương với S. Nếu không thì T_2 đọc một giá trị của A không chứa hàm f, và một giá trị như thế không thể đồng nhất với một giá trị có chứa f. Tương tự, nếu T_3 là giao dịch kế tiếp sau T_1, nhận khoá ghi A, thì T_1 phải đi trước T_3. Đây chính là lập luận trong định lý 6.1.

Bây giờ ta giả sử rằng T_4 là một giao dịch nhận khoá đọc A trước khi T_1 khoá ghi A. Nếu T_1 xuất hiện trước T_4 trong lịch biểu tuần tự thì T_4 đọc một giá trị của A có chứa hàm f, còn trong lịch biểu S, giá trị được đọc bởi T_4 không chứa f. Vì vậy, T_4 phải đi trước T_1 trong một lịch biểu tuần tự. Suy luận duy nhất không thể thực hiện được là nếu trong S hai giao dịch cũng nhận khoá đọc một mục A theo một thứ tự nào đó thì những giao dịch này phải xuất hiện theo thứ tự đó trong một lịch biểu

tuần tự. Đúng ra thứ tự tương đối này của các khoá đọc không tạo ra sự khác biệt nào trên các giá trị được tạo ra bởi các giao dịch thực hiện đồng thời. Những nhận xét này gợi ý rằng một cách tiếp cận tương tự như trong phần 6.2 sẽ cho phép khẳng định một lịch biểu có khả tuần tự hay không.

Thuật toán 6.2: *Kiểm tra tính khả tuần tự của các lịch biểu với các khoá đọc / ghi*

Input: Một lịch biểu S cho một tập các giao dịch $T_1, \ldots, T_k$

Output: Khẳng định S có khả tuần tự hay không nếu được sẽ đưa ra một lịch biểu tuần tự tương đương.

Phương pháp:

Chúng ta xây dựng một đồ thị tuần tự hoá G như sau. Các nút tương ứng là các giao dịch . Các cung được xác định bằng quy tắc sau.

1. Giả sử trong S, giao dịch T_i nhận khoá đọc hoặc khoá ghi mục A, T_j là giao dịch kế tiếp khoá ghi A, và $j \neq i$. Khi đó chúng ta đặt một cung từ T_i đến T_j.

2. Giả sử trong S, giao dịch T_i khoá ghi A. Gọi T_m, với $m \neq i$ là một giao dịch khoá đọc A sau khi T_i mở khoá A nhưng trước các giao dịch khác khoá ghi A. Chúng ta vẽ một cung từ $T_i \rightarrow T_m$.

Nếu G có chu trình thì S bất khả tuần tự. Nếu G không có chu trình thì một sắp xếp topo của G là thứ tự tuần tự cho các giao dịch này.

Thí dụ 6.9:

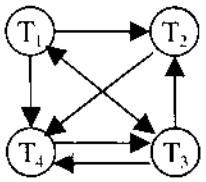
Trong hình 6.9, chúng ta có một lịch biểu của bốn giao dịch:T_1, T_2, T_3, T_4. Hình 6.10 là một đồ thị tuần tự hoá cho lịch biểu này. Lệnh

UNLOCK đầu tiên là bước 3, ở đó T_3 mở khoá ghi cho A. Sau bước 3 là các khoá đọc A và T_1 và T_2 (bước 4 và 7) và một khoá ghi A do T_4 tại bước 12. Như vậy T_1, T_2 và T_4 phải theo sau T_3. Chúng ta vẽ các cung từ T_3 đến các nút này. Chú ý rằng không có gì sai khi cả T_1 và T_2 đều giữ các khoá đọc trên A sau bước 7. Tuy nhiên, T_4 không thể nhận khoá ghi A cho đến khi T_1 và T_2 đều giải phóng các khoá đọc của chúng. Tiếp tục, T_4 giải phóng khoá đọc trên B ở bước 5, và khoá ghi kế tiếp trên B là do T_3 vì thế chúng ta vẽ một cung từ T_4 đến T_3.

	T_1	T_2	T_3	T_4
(1)			WLOCK A	
(2)				RLOCK B
(3)			UNLOCK A	
(4)	RLOCK A			
(5)				UNOCK B
(6)			WLOCK B	
(7)		RLOCK A		
(8)			UNLOCK B	
(9)	WLOCK B			
(10)		UNLOCK A		
(11)	UNLOCK A			
(12)				WLOCKA
(13)	UNLOCK B			
(14)		RLOCK B		
(15)				UNLOCK A
(16)		UNLOCK B		

Hình 6.9 Một lịch biểu của T_1,T_2,T_3,T_4

Lịch biểu trong hình 6.9 bà bất khả tuần tự vì có chu trình. Đồ thị tuần tự hoá được trình bày trong hình 6.10.



Hình 6.10 Đồ thị tuần tự hoá của lịch biểu 6.9

Định lý 6.3:

Thuật toán 6.2 xác định chính xác lịch biểu S có khả tuần tự hay không.

Chứng minh:

Đơn giản là khi chúng ta vẽ một cung T_i đến T_j thì trong lịch biểu tuần tự tương đương T_i phải đi trước T_j. Do đó nếu G có một chu trình, chúng ta có thể chứng minh như trong định lý 6.1 rằng không thể có một lịch biểu tuần tự như thế tồn tại. Ngược lại, giả sử rằng G không có chu trình thì bằng cách lập luận như định lý 6.1, giá trị cuối cùng của mỗi mục là như nhau trong S và trong lịch biểu tuần tự R được xây dựng từ phép sắp xếp topo của G. Chúng ta cũng phải chứng minh rằng các khoá đọc tương ứng trên A thu được cùng giá trị trong R và S. Phần chứng minh này hoàn toàn dễ dàng vì các cung của G bảo đảm rằng các khoá ghi trên A đi trước một khoá đọc cho trước phải như nhau trong R và S và phải xảy ra theo cùng một thứ tự.

Cuối cùng chúng ta lưu ý rằng nghi thức hai pha của các giao dịch với khoá đọc, ghi giống như với mô hình trong phần trước, nghi thức hai

pha, trong đó tất cả các khoá đọc hay ghi đều đi trước các bước mở khoá, là đủ để đảm bảo tính khả tuần tự.

6.5. CÁC THỂ THỨC KHOÁ CHỐT

Như chúng ta đã thấy trong các phần trước, các khoá có thể được đưa ra theo nhiều "hứng thú" khác nhau, chúng được gọi là các thể thức khóa chốt (lock mode) và mỗi thể thức đều có các đặc tính khác nhau khi cần quyết định xem có thể trao khoá hay không khi một giao dịch khác đã có một khoá thuộc cùng thể thức hoặc một thể thức khác trên cùng một mục. Trong phần này chúng ta có hai thể thức là "đọc" và "ghi", với những quy tắc liên quan là:

1. Một khoá đọc trên một mục có thể được trao nếu không có khoá ghi nào của một giao dịch khác giữ trên mục đó.

2. Một khoá ghi trên một mục có thể được trao chỉ khi không có khóa đọc hay khoá ghi nào đang được một giao dịch khác giữ trên mục đó.

Chú ý rằng những quy tắc này không áp dụng được cho tình huống ở đó một giao dịch đang yêu cầu một khoá đồng thời lại đang giữ một khoá khác trên mục đó.

Chúng ta có thể tóm tắt các quy tắc cho các khoá đọc và ghi bằng một *ma trận tương thích khoá* (lock compatibility matrix). Các hàng tương ứng với thể thức của khoá được yêu cầu và các cột tương ứng với thể thức của khoá đang được một giao dịch khác giữ. Các khoản ghi là Y (yes, khoá được yêu cầu có thể trao được), và N (no, khoá không thể trao được).

Thí dụ 6.10:

Ma trận tương thích khoá đọc và ghi được trình bày:

		Khoá được giữ bởi một giao dịch khác	
		Read	Write
Khoá được yêu cầu	Read	Y	N
	Write	N	N

Hình 6.11 Ma trận tương thích khoá đọc và ghi

Phép kiểm tra tính khả tuần tự của thuật toán 6.2 được tổng quát hoá cho những ma trận tương thích. Chúng ta xây dựng một đồ thị tuần tự hoá bằng cách đặt một cung từ $T_1 \rightarrow T_2$ nếu trong lịch biểu đã cho, một khoá bởi T_1 trên mục A ở thể thức L đi trước một khoá bởi T_2 trên A ở thể thức M, giá trị của mục trong M và cột L và "N". Giống như trước kia, phép kiểm tra tính khả tuần tự chính là kiểm tra sự không có chu trình trong đồ thị.

	Read	Write	Incr
Read	Y	N	N
Write	N	N	N
Incr	N	N	Y

Hình 6.12 Tính tương thích của thao tác đọc, ghi

6.6. THỂ THỨC CHỈ ĐỌC, CHỈ GHI

Trong phần 6.2 và 6.4 chúng ta đã biết, khi một giao dịch ghi một giá trị mới vào một mục A thì trước đó nó phải đọc giá trị của A và giá trị mới của A phụ thuộc vào giá trị cũ. Trong thực tiễn chúng ta chấp nhận khả năng là một giao dịch *đọc một tập các mục (tập đọc)* và *ghi vào một tập các mục (tập ghi)*. Mục A có thể xuất hiện ở một trong hai tập hoặc ở trong cả hai tập.

Thí dụ 6.11:

Giao dịch T sau đây vấn tin một CSDL nhưng không làm thay đổi các giá trị trong nó. Khi đó sẽ có một tập ghi rỗng.

T: Begin READ A; READ B; C: = A + B ; A : = A – 1 end .

Thí dụ 6.12:

Giao dịch T' sau đây có làm thay đổi các giá trị trong nó.

T': Begin READ A; READ B; C: = A + B ; A : = A - 1; WRITE C; WRITE A end.

Tập đọc là {A, B} và tập ghi là {A, C}.

6.6.1.Ý NGHĨA CỦA CÁC GIAO DỊCH VÀ LỊCH BIỂU

Các giao dịch ở đây chỉ khác với mô hình của phần 6.4 ở một điểm. Chúng ta *không giả sử rằng khoá ghi một mục hàm chứa thao tác đọc mục đó* (xem thí dụ 6.12). Vì vậy đi kèm với mỗi khoá ghi trên một mục

A là một hàm để tính giá trị mới A theo tập đọc của giao dịch này. Đặc biệt giá trị mới này không phụ thuộc vào giá trị cũ của A nếu A không thuộc tập đọc (giá trị của C trong thí dụ 6.12 không phụ thuộc giá trị cũ của nó).

Trong phần này *hai lịch biểu là tương đương* nếu và chỉ nếu chúng sinh ra những giá trị giống nhau cho mỗi mục CSDL được ghi, như là các hàm theo giá trị ban đầu của các mục là được đọc.

6.6.2.HAI KHÁI NIỆM KHẢ TUẦN TỰ :TRỰC QUAN VÀ TƯƠNG TRANH

Theo khuôn mẫu của các phần 6.2 và 6.4, chúng ta có thể định nghĩa một lịch biểu là "khả tuần tự" nếu nó tương đương với một lịch biểu tuần tự nào đó. Định nghĩa này gây ra nhiều khó khăn, cụ thể là trong mô hình này không tồn tại phép kiểm tra bằng đồ thị đơn giản như trong những mô hình trước. Vì vậy tính tương đương với một lịch biểu tuần tự ở đây chỉ là một định nghĩa thường được gọi là:

Tính khả tuần tự trực quan (view-serializability). Hoặc

Tính khả tuần tự tương tranh (conflict - serializability).

6.6.3.ĐỒ THỊ TUẦN TỰ HOÁ CHO CÁC GIAO DỊCH CHỈ ĐỌC, CHỈ GHI

Giả sử (theo mô hình của phần 6.4) rằng trong lịch biểu S đã cho, giao dịch T_1 ghi một giá trị cho mục A, và sau đó T_2 ghi một giá trị cho A. Vậy theo giả thiết trong phần 6.4 T_2 khoá ghi A sau khi T_1 mở khoá A, và chúng ta suy ra T_2 đã dùng giá trị của A được ghi bởi T_1 trong việc tính giá trị mới. Do đó khi giải quyết tính khả tuần tự, chúng ta thừa nhận rằng trong một lịch biểu tuần tự R tương đương với S, T_1

xuất hiện trước T_2, và rõ ràng là không có một giao dịch T nào khác khoá ghi A xuất hiện giữa T_1 và T_2 .

Tất nhiên nếu chúng ta giả sử rằng T_2 ghi giá trị cho A mà không đọc A thì giá trị mới của A sẽ độc lập với giá trị cũ: nó chỉ phụ thuộc vào những giá trị của các mục thực sự được đọc bởi T_2. Vì vậy nếu giữa những lần T_1 và T_2 ghi giá trị cho A không có giao dịch nào đọc A, chúng ta thấy rằng giá trị được viết bởi T_1 không có tác dụng nào trên CSDL. Kết quả là, trong một lịch biểu tuần tự, không cần thiết T_1 phải xuất hiện trước T_2 (ít nhất khi xét đến các tác dụng trên A). Yêu cầu duy nhất đối với T_1 là nó phải được thực hiện vào một thời điểm trước khi một giao dịch T_3 khác ghi A, và giữa những lần T_1 và T_2 ghi A không có giao dịch nào đọc A.

Bây giờ chúng ta có thể thiết lập một định nghĩa mới cho đồ thị tuần tự hoá, dựa trên ý nghĩa là các giá trị được ghi bởi một giao dịch là những hàm chỉ phụ thuộc vào những giá trị được đọc, và những giá trị được đọc khác nhau sinh ra những giá trị được ghi khác nhau. Những điều kiện yêu cầu một giao dịch phải đi trước một giao dịch khác được quy định (và không hoàn toàn chính xác) như sau:

Nếu trong lịch biểu S, giao dịch T_2 đọc giá trị của mục A được ghi bởi T_1 thì

1. T_1 phải đi trước T_2 : trong một lịch biểu tuần tự tương đương với S.

2. Nếu T_3 là một giao dịch ghi A, thì trong một lịch biểu tuần tự tương đương với S, T_3 có thể đi trước T_1 hay đi sau T_2 nhưng không thể xuất hiện giữa T_1 và T_2.

6.6.4.KHẢ TUẦN TỰ TƯƠNG TRANH

Phép kiểm tra đồ thị tuần tự đơn giản của những phần trước không dùng ở đây được. Hãy nhớ rằng có hai loại ràng buộc trên một lịch biểu tuần tự tương đương với một lịch biểu S cho trước.

1. *Ràng buộc kiểu 1* : Nếu T_2 đọc một giá trị của A được ghi bởi T_1 trong S thì T_1 phải đi trước T_2 trong một lịch biểu tuần tự. Kiểu ràng buộc này có thể diễn tả bằng một cung từ T_1 đến T_2 .

2. *Ràng buộc kiểu 2*: Nếu T_2 đọc một giá trị A được ghi bởi T_1 trong S thì một giao dịch T_3 ghi A phải xuất hiện trước T_1 hoặc sau T_2 . Điều này diễn tả bằng một cung trong cặp cung:

$T_3 \rightarrow T_1$ hoặc $T_2 \rightarrow T_3$, và chọn một trong hai cung này.

Vậy lịch biểu S được gọi là *khả tuần tự tương tranh (conflict - serializable)* nếu có một lịch biểu tuần tự theo các ràng buộc kiểu 1 và 2 được sinh ra từ S. Như đã thấy trong định lý 6.1 và 6.3, các khái niệm của tính khả tuần tự trực quan (trong các phần 6.2, 6.4) và khả tuần tự tương tranh là tương đương nhau trong các mô hình đơn giản của phần 6.2 và 6.4. Tuy nhiên chúng ta thấy rằng tính khả tuần tự tương tranh kéo theo tính khả tuần tự trực quan, nhưng ngược lại sẽ không đúng trong mô hình hiện tại.

6.6.5.KIỂM TRA ĐỒ THỊ CHO KHẢ TUẦN TỰ TƯƠNG TRANH

Trong phần này ta sẽ dùng đa đồ thị. Đa đồ thị khác đơn đồ thị ở chỗ trong đa đồ thị giữa một cặp đỉnh (nút) có thể có hơn hai cung. Một đa đồ thi được gọi là *không chu trình (acyclic)* nếu có một đơn đồ thị (chọn lựa một cung từ mỗi cặp để tạo ra một đồ thị đơn) không có chu trình theo nghiã thông thường.

Kiểm tra tính khả tuần tự tương tranh là xây dựng một đa đồ thị thích hợp và xác định xem nó có phải là không chu trình hay không.Tuy nhiên, việc kiểm tra tính chất có chu trình hay không của một đa đồ thị là một bài toán khó, đây là bài toán dạng NP - complete.

Thuật toán 6.3: *Kiểm tra tính khả tuần tự tương tranh cho các giao dịch với các khoá chỉ đọc, chỉ ghi*

Input: một lịch biểu S cho tập giao dịch $T_1, T_2, \ldots T_k$.

Output: Một kết luận cho biết S có khả tuần tự tương tranh hay không, nếu có thì đưa ra một lịch biểu tuần tự tương đương.

Phương pháp:

1. Mở rộng S bằng cách ghi thêm vào vị trí bắt đầu một chuỗi các bước của một giao dịch giả T_0 ghi vào mỗi mục xuất hiện trong S và ghi thêm vào vị trí cuối một giao dịch giả T_f đọc mỗi mục như thế.

2. Tạo một đa đồ thị P có các nút là các giao dịch, bao gồm cả các giao dịch T_0 và T_f. Tạm thời chúng ta đặt một cung T_i đến T_j, nếu T_j đọc một mục A được T_i ghi cuối cùng trong lịch biểu S mở rộng.

3. Tìm các giao dịch vô dụng. Một giao dịch T là vô dụng nếu không có đường đi từ T đến T_f.

4. Đối với mỗi giao dịch vô dụng T, loại bỏ mọi cung đến T.

5. Đối với mỗi cung $T_i \rightarrow T_j$ còn lại, và đối với mỗi mục A sao cho T_j đọc giá trị của A được ghi bởi T_i, chúng ta xét mỗi giao dịch $T \neq T_0$ cũng ghi A. Nếu $T_i = T_0$ và $T_j = T_f$, chúng ta không vẽ thêm một cung nào. Nếu $T_i = T_0$ nhưng $T_j \neq T_f$, chúng ta vẽ thêm một cung $T_j \rightarrow T$. Nếu $T_j = T_f$ nhưng $T_i \neq T_0$ thì thêm cung $T \rightarrow T_i$. Nếu $T_i \neq T_0$ và $T_j \neq T_f$ thì đưa ra một cặp cung ($T \rightarrow T_i$, $T_j \rightarrow T$).

6. Xác định xem đa đồ thị P có phải là một đa đồ thị không chu trình hay không. Đối với bước này, không có phương pháp nào tốt hơn phương pháp vét cạn (exhaustive method). Nếu P không chu trình, gọi G là một đơn đồ thị không chu trình được tạo từ P bằng cách chọn một cung từ mỗi cặp. Như vậy một sắp xếp topo của G, loại bỏ T_0 và T_f, biểu diễn một lịch biểu tuần tự tương đương với S. nếu P có chu trình thì không tồn tại một lịch biểu tuần tự nào tương đương với S.

Thí dụ 6.13:

Xét lịch biểu của hình 6.13. Các cung được xây dựng bằng bước (2) của thuật toán 6.3 được trình bày trong hình 6.14 để cho rõ ràng, các cung được gắn nhãn bằng các trụ tương ứng. Để hiểu được quá trình xây dựng hình 6.14, trước tiên chúng ta nhận xét rằng lịch biểu của hình 6.13 là hợp lệ, theo nghĩa là hai giao dịch không đồng thời giữ các khoá ghi, hoặc một khoá đọc - ghi. Vì vậy giả sử rằng tất cả mọi thao tác đọc và ghi xảy ra ngay tại lúc nhận khoá, và chúng ta có thể bỏ qua các bước UNLOCK.

Hãy xét lần lượt mỗi bước khoá đọc. Các khoá đọc trên A tại bước (1) và (2) đọc giá trị "được ghi" bởi giao dịch giả T_0. Vì vậy ta vẽ

các cung từ T_0 đến T_1 và T_2. Tại bước (5), T_3 đọc giá trị của C được ghi bởi T_1 tại bước (3), vì vậy chúng ta có cung $T_1 \rightarrow T_3$. Tại bước (8), T_4 đọc giá trị đo T_1 ghi tại bước (6) nên chúng ta có cung $T_1 \rightarrow T_4$, vân vân. Cuối cùng lúc kết thúc, T_f đọc A, B, C và D, những giá trị của chúng được ghi cuối cùng bởi T_4, T_4, T_1 và T_2 giải thích ba cung đến T.

Bây giờ chúng ta đi tìm các giao dịch vô dụng, là những giao dịch không có đường đi đến T_f trong hình 6.14; T_3 là giao dịch duy nhất như thế. Do đó chúng ta loại cung $T_1 \rightarrow T_3$, ra khỏi hình 6.14.

Trong bước (5) của thuật toán 6.3, chúng ta xét các cung hoặc các cặp cung cần để ngăn cản tác động của một thao tác ghi đối với một thao tác khác. Một mục như C hoặc D được ghi bằng một giao dịch "không giả" duy nhất nên không được nói đến trong bước (5). Tuy nhiên, A được ghi bởi cả T_3 và T_4 và giao dịch giả T_0. Giá trị được ghi bởi T_3 không được giao dịch nào đọc, vì vậy, T_4 không cần xuất hiện ở một vị trí đặc biệt nào so với T_3. Giá trị được ghi bởi T_4 được T_f đọc. Vì T_3 không thể xuất hiện sau T_f nên phải xuất hiện trước T_4. Trong trường hợp này, chúng ta không cần đến một cặp cung nào; ta chỉ thêm vào P cung $T_3 \rightarrow T_4$. Giá trị của A được ghi bởi T_0 và được đọc bởi T_1 và T_2. Hơn nữa, vì T_3 và T_4 không thể xuất hiện trước T_0, ta đặt các cung từ T_1 và T_2 đến T_3 và T_4; một lần nữa cũng không cần sử dụng thêm một cặp cung nào.

Mục B được ghi bởi T_1 và T_4. Giá trị của B được ghi bởi T_4 chỉ được T_f đọc, vì vậy ta có một cung $T_1 \rightarrow T_4$. Giá trị của B do T_1 ghi được đọc bởi T_2 và T_4. Việc ghi B bởi T_4 không ảnh hưởng đến việc đọc B bởi T_4. Do đó không cần đòi hỏi "T_4 phải đi trước T_1 hoặc đi sau T_4".

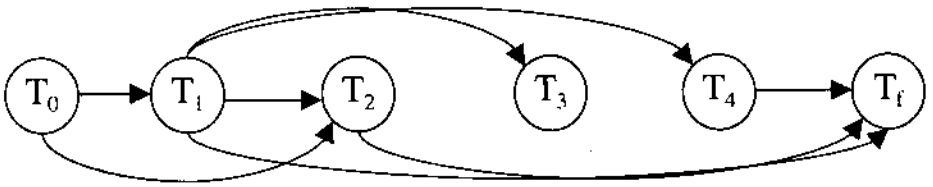
	T_1	T_2	T_3	T_4
(1)		RLOCK A		
(2)	RLOCK A			
(3)	WLOCK C			
(4)	UNLOCK C			
(5)			RLOCKC	
(6)	WLOCK B			
(7)	UNLOCK B			
(8)				RLOCK B
(9)	UNLOCK A			
(10)		UNLOCK A		
(11)			WLOCK A	
(12)				RLOCK C
(13)		WLOCKD		
(14)				UNLOCK B
(15)			UNLOCK C	
(16)		RLOCK B		
(17)			UNLOCKA	
(18)				WLOCK A
(19)		UNLOCK B		
(20)				WLOCK B
(21)				UNLOCK B
(22)		UNLOCK D		
(23)				UNLOCK C
(24)				UNLOCK A

Hình 6.13 Một lịch biểu

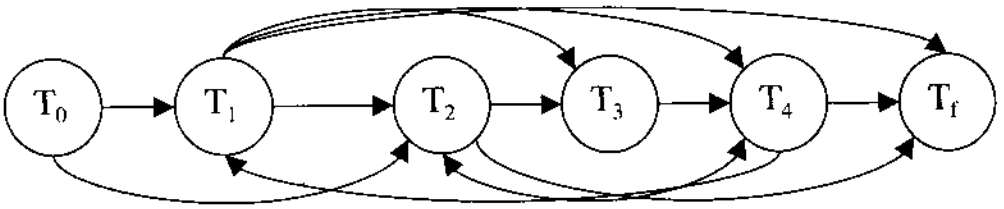
Tuy nhiên T_4 không được xen vào giữa T_1 và T_2, nên ta thêm một cặp cung ($T_4 \rightarrow T_1$, $T_2 \rightarrow T_4$). Đa đồ thị cuối cùng được trình bày trong

Hình 6.15. Chú ý rằng cung $T_1 \rightarrow T_3$ đã bị loại bỏ ở bước (4) lại được trả về trong bước (5).

Nếu chọn cung $T_4 \rightarrow T_1$ thì ta có một chu trình. Tuy nhiên, chọn $T_2 \rightarrow T_4$ ta được một đồ thị không chu trình, từ đó chúng ta có thể có được một thứ tự tuần tự T_1, T_2, T_3, T_4 vì thế lịch biểu trong hình 6.13 khả tuần tự.



Hình 6.14 Bước đầu tiên trong quá trình xây dựng một đa đồ thị



Hình 6.15 Đa đồ thị của lịch biểu.

Thuật toán 6.1 và 6.2 đúng đắn theo nghĩa là chúng sinh ra một lịch biểu tuần tự tương đương nếu và chỉ nếu nó tồn tại, theo ngữ nghĩa của các giao dịch được dùng tương ứng trong các phần 6.2 và 6.4. Thuật toán 6.3 không hoàn toàn đúng đắn theo nghĩa chặt chẽ như vậy. Nếu nó tìm ra một lịch biểu tuần tự thì lịch biểu đó hoàn toàn tương đương với

lịch biểu đã cho, theo ngữ nghĩa của các giao dịch chỉ đọc, chỉ ghi trong phần này. Tuy nhiên nếu không tìm được một lịch biểu tuần tự, nó chỉ có nghĩa là không có lịch biểu nào phù hợp với các ràng buộc kiểu 1 và 2 trên thứ tự giao dịch. Vì vậy ta có định lý:

Định lý 6.4:

a. Nếu thuật toán 6.3 thành công đối với lịch biểu S cho trước, thì có một lịch biểu tuần tự tương đương với S theo ngữ nghĩa chỉ đọc, chỉ ghi.

b. Nếu thuật toán 6.3 thất bại đối với S thì không có lịch biểu tuần tự nào phù hợp với các kiểu ràng buộc 1 và 2 đối với S.

Chứng minh:

a. Đầu tiên giả sử rằng đa đồ thị thu được không có chu trình. Nghĩa là có một cách chọn trong các cung của mỗi cặp để tạo ra một đồ thị G không chu trình. Phương pháp xây dựng P trong thuật toán 6.3 bảo đảm rằng mỗi giao dịch không vô dụng, bao gồm cả T_f, sẽ đọc cùng một bản sao của mỗi mục trong S giống như trong lịch biểu tuần tự sinh ra từ một cách sắp xếp topo của G. Vì thế, những giá trị tương ứng được sinh ra cho mỗi mục là như nhau trong cả hai lịch biểu.

b. Ngược lại, giả sử có một lịch biểu tuần tự R thoả các ràng buộc kiểu 1 và 2 được sinh ra từ lịch biểu S đã cho. Theo thuật toán 6.1, nếu $T_i \rightarrow T_j$ là cung được tạo ra ở bước (2) và không bị loại trong bước (4), T_i phải đi trước T_j trong R. Giả sử cặp cung ($T_n \rightarrow T_i$ $T_j \rightarrow T_n$) được tạo ra trong bước (5). Vậy T_n không thể xuất hiện giữa T_i và T_j trong R.

Ta chọn cung $T_n \rightarrow T_i$ nếu T_n đi trước T_i trong R và ngược lại chọn cung $T_j \rightarrow T_n$. Thứ tự tuyến tính được suy ra bởi R sẽ nhất quán với cách chọn này. Tương tự, một cung được thêm vào trong bước (5) cũng phải nhất quán với thứ tự tuyến tính này, vì thế chúng ta có một cách xây dựng một đồ thị không chu trình dựa trên R từ đa đồ thị P.

6.6.6. NGHI THỨC HAI PHA

Giống như những mô hình trước, nghi thức hai pha đòi hỏi mỗi giao dịch thực hiện tất cả mọi thao tác khoá chốt trước tất cả mọi thao tác mở khoá nhằm bảo đảm được tính khả tuần tự tương tranh của một lịch biểu hợp lệ. Muốn hiểu tại sao, chúng ta hãy giả sử S là một lịch biểu hợp lệ của những giao dịch tuân theo nghi thức hai pha. Giả sử ($T_3 \rightarrow T_1$, $T_2 \rightarrow T_3$) là một cặp cung trong đa đồ thị P. khi đó có mục A sao cho T_2 đọc A được ghi bởi T_1.Nếu trong S, T_3 mở khoá A trước khi T_1 khoá đọc A thì chọn cung $T_3 \rightarrow T_1$. Nếu T_3 khoá ghi A sau khi T_2 mở khoá A thì chúng ta chọn $T_2 \rightarrow T_3$. Không có khả năng khác tồn tại, bởi vì cặp cung này được đặt trong P bằng thuật toán 6.3.

Bây giờ chúng ta có một đồ thị G được xây dựng từ P. Giả sử G có một chu trình $T_1 \rightarrow T_2 \rightarrow \ldots T_n \rightarrow T_1$. Chắc chắn rằng không có giao dịch giả nào có thể tham gia vào chu trình này. Kiểm tra thuật toán 6.3 và các quy tắc xây dựng G từ P cho thấy đối với mỗi cung $T_i \rightarrow T_{i+1}$ (với $T_{n+1} = T_1$) trong chu trình này, có một mục A_i sao cho trong S, T_i mở khoá A_i trước khi T_{i+1} khoá A_i. Theo nghi thức hai pha, T_{i+1} chỉ mở khoá A_{i+1} sau khi đã khóa A_i. Vì vậy T_1 mở khóa A_1 trước khi T_{n+1} khoá A_n. Nhưng T_{n+1} chính là T_1, và nghi thức hai pha cấm T_1 mở khoá A_1

trước khi khoá A_n. Vì vậy chúng ta đã chứng minh xong định lý dưới đây.

Định lý 6.5:

Nếu các giao dịch tuân theo nghi thức hai pha thì mọi lịch biểu hợp lệ đều khả tuần tự. Chú ý định lý này chỉ dùng cho mô hình trong phần này.

6.6.7. KHẢ TUẦN TỰ TRỰC QUAN

Trong mô hình đang xét có một số lịch biểu *không khả tuần tự tương tranh nhưng khả tuần tự trực quan.* Nghĩa là kiểm tra bằng thuật toán 6.3

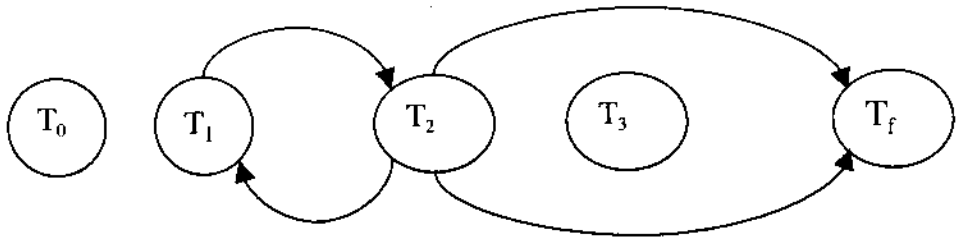
Ta có một đồ thị có chu trình, tuy nhiên, nhìn vào lịch biểu ta thấy lịch biểu khả tuần tự (trực quan) vì nó có kết quả giống với một lịch biểu tuần tự.

Thí dụ xét lịch biểu của ba giao dịch T_1, T_2, T_3 sau:

Bước	T_1	T_2	T_3
(1)	Wlock A		
(2)	Unlock A		
(3)		Wlock C	
(4)		Unlock C	
(5)		Rlock A	

(6)		Wlock B	
(7)		Unlock A	
(8)		Unlock B	
(9)	Rlock C		
(10)	Wlock D		
(11)	Unlock C		
(12)	Unlock D		
(13)			Wlock B
(14)			Wlock D
(15)			Unlock B
(16)			Unlock D

Đa đồ thị của lịch biểu là:



Đây là lịch biểu có đa đồ thi có chu trình vậy theo thuật toán 6.3 lịch biểu bất khả tuần tự tương tranh. Tuy nhiên theo trực quan ta thấy lịch biểu S tương đương với lịch biểu tuần tự T_2, T_1, T_3. Trong thực tiễn chúng ta ít gặp các trường hợp khả tuần tự trực quan nên *chúng ta chỉ xét những lịch biểu với khả tuần tự tương tranh.*

6.7. XỬ LÝ SỰ CỐ GIAO DỊCH

Cho đến bây giờ chúng ta đã giả sử rằng mỗi giao dịch đều thực hiện một cách trọn vẹn, không bị trục trặc. Trong thực hành, rất nhiều lý do khiến cho một giao dịch cuối cùng phải bị huỷ bỏ (kết thúc nhưng chưa hoàn tất) do gặp phải một trong các trường hợp sau:

1. Chẳng hạn do người sử dụng ngắt, do một phép toán sai lầm như chia cho 0 hoặc do giao dịch cố gắng truy xuất một mục của CSDL mà nó không có quyền truy xuất.

2. Bộ xếp lịch phát hiện ra một *khoá gài* (deadlock), và quyết định huỷ bỏ giao dịch này để có thể giải phóng các khoá của nó và cho phép các giao dịch khác tiếp tục.

3. Một số thuật toán xếp lịch đôi khi cần phải huỷ bỏ một giao dịch nhằm bảo đảm được tính khả tuần tự.

4. Một lỗi phần mềm hoặc phần cứng làm cho hệ thống phải ngừng lại.

Trường hợp đơn giản nhất là các sự cố của một giao dịch chỉ xẩy ra như (1 - 3). Việc xử lý khó khăn hơn đối với những sự cố của hệ thống phần mềm(như (4)), bởi vì thông thường các giao dịch đang hoạt động tại thời điểm xảy ra sự cố sẽ phải thực hiện lại. Nếu không cẩn thận trong việc quản lý các giao dịch, chúng ta sẽ mất các thông tin quan trọng về những sự việc đang xảy ra lúc có sự cố, và không thể khởi hoạt lại chính xác các hoạt động kể từ lúc đó. Nghiêm trọng hơn là sự cố về vị trí lưu dữ liệu, ở đó các dữ liệu của CSDL bị mất. Cách duy nhất để khôi phục hoạt động do sự cố thiết bị là duy trì một bản dự phòng cho tất cả CSDL ở tình trạng mới nhất.

Các phương pháp để khôi phục những sự cố phần cứng và phần mềm sẽ được thảo luận trong phần sau (6.9). Ở đây chúng ta chỉ xét những vấn đề xảy ra do các sự cố của một giao dịch hoặc do quyết định huỷ bỏ một giao dịch của bộ xếp lịch.

6.7.1.HOÀN THÀNH GIAO DỊCH VÀ DỮ LIỆU RÁC

Hoàn thành (hoặc uỷ thác - commit) giao dịch là chỉ thời điểm giao dịch đã hoàn tất mọi công việc, và *uỷ thác* cho hệ thống ghi kết quả lao động của nó vào CSDL.

Ngược lại, một giao dịch không kết thúc được do sự cố thì tại thời điểm đó (thời điểm sự cố) phải *huỷ bỏ (abort) giao dịch (giao dịch huỷ bỏ).*

Trong phần sau chúng ta sẽ thấy rằng có một số hành động đặc biệt phải được thực hiện khi đạt đến điểm uỷ thác, nhưng tạm thời chúng ta chỉ xem hành động COMMIT như một thao tác đánh dấu điểm uỷ thác của một giao dịch.

Dữ liệu được ghi vào trong CSDL bởi một giao dịch trước khi giao dịch đó hoàn thành gọi là *dữ liệu rác* (dirty data).

Chúng ta sẽ gặp nhiều phiền phức do đọc các dữ liệu rác trong tình huống giao dịch ghi bị huỷ bỏ. Chẳng hạn xét tình huống tại một thời điểm nào đó giao dịch T ghi một mục A vào CSDL (A là dữ liệu rác) nhưng cuối cùng chính giao dịch T bị huỷ bỏ, chúng ta phải làm gì với dữ liệu A ? Ta xét một thí dụ cụ thể :

Thí dụ 6.14:

Xét hai giao dịch của hình 6.16. Về cơ bản, những giao dịch này tuân theo mô hình của phần 6.2. Chúng ta sẽ trình bày các thao tác uỷ thác, đọc, ghi, và các phép tính số học được thực hiện trong vùng làm việc của mỗi giao dịch. Giả sử rằng hành động WRITE lưu một giá trị vào trong CSDL, còn các bước tính toán số học như (3) chẳng hạn, được thực hiện trong vùng làm việc riêng và không có tác dụng trên CSDL.

Giả sử rằng sau bước (14) giao dịch T_1 gặp sự cố chia cho 0, hoặc do giao dịch này bị kẹt trong một khoá gài khiến bộ xếp lịch quyết định huỷ bỏ T_1. Chúng ta phải thực hiện các hành động sau.

1. T_1 vẫn còn giữ một khóa trên mục B. Khoá đó phải được hệ thống lấy lại.

2. Giá trị của A được T_2 ghi ở bước (4) phải được khôi phục về lại giá trị A ở trước bước (1). Ở đó có lẽ không có một mẫu tin nào ghi lại giá trị cũ của A; khi chúng ta thảo luận về quá trình khôi phục từ những sự cố hệ thống ở phần sau chúng ta sẽ thấy rằng điều quan trọng trong trường hợp này là chúng ta cần ghi giá trị cũ của A vào trong một *nhật ký* (journal hay log).

3. Giá trị của A được đọc bởi T_2 ở bước (8) bây giờ không còn đúng nữa. T_2 đã thực hiện một phép tính không đúng và ghi kết quả vào A. Vì vậy không những chúng ta phải khôi phục lại giá trị của A từ trước bước (1) mà chúng ta còn phải hồi lại tất cả các kết quả của T_2 và thực hiện lại giao dịch đó.

4. Giả sử rằng một giao dịch T_3 nào đó đọc giá trị của A giữa các bước (13) và (14). Như vậy T_3 cùng sử dụng dữ liệu vô giá trị và cũng

phải thực hiện lại, ngay cả nếu nó đã đạt đến điểm commit. Hơn nữa, một giao dịch đọc một giá trị được viết bởi T_3 cũng thực hiện lại v.v.

	T_1	T_2
(1)	LOCK A	
(2)	READ A	
(3)	A: = A – 1	
(4)	WRITE A	
(5)	LOCK B	
(6)	UNLOCK A	
(7)		LOCK A
(8)		READ A
(9)		A: = A * 2
(10)	READ B	
(11)		WRITE A
(12)		COMMIT
(13)		UNLOCK A
(14)	B: = B/A	

Hình 6.16 Một lịch biểu

Hiện tượng được minh hoạ ở điểm (4) của thí dụ 6.14 được gọi là *cuộn ngược dây chuyền* (cascading rollback). Đây là hậu quả của quyết định cho phép T_2 đọc các dữ liệu rác. Nghĩa là một khi chúng ta cho phép dù chỉ một giao dịch đọc dữ liệu rác thì việc hoàn tất một giao dịch T không đảm bảo rằng một lúc nào đó trong tương lai sẽ phát hiện ra rằng T đã đọc một giá trị đã không còn ở đó, và vì vậy phải thực hiện lại T.

6.7.2. KHOÁ HAI PHA NGHIÊM NGẶT

Do đọc những dữ liệu rác không thể chấp nhận được nên thông thường người ta sử dụng một phiên bản của nghi thức khoá hai pha, được gọi là *khoá hai pha nghiêm ngặt* (strict two - phase locking). Đó là

1. Một giao dịch không thể ghi vào CSDL trước khi nó đạt đến điểm commit.

2. Một giao dịch không thể giải phóng các khoá trước khi nó hoàn tất việc ghi vào trong CSDL; vì thế các khoá được giải phóng sau điểm commit.

Thí dụ 6.15:

T_2 của hình 6.16 không hoàn toàn là một giao dịch hai pha nghiêm ngặt. Bởi vì nếu nó nghiêm ngặt, lệnh COMMIT tại dòng (12) sẽ phải di chuyển đến trước lệnh WRITE ở dòng (11).

Rõ ràng điều kiện (2) là cần thiết nhằm tránh hiện tượng *cuộn ngược dây chuyền*, bởi vì nếu không có nó, một giao dịch khác có thể đọc một giá trị được ghi bởi một giao dịch mà về sau bị buộc phải huỷ bỏ. Điều kiện (1) cũng quan trọng. Thật vậy, cho đến khi khoá được giải phóng không giao dịch nào khác có thể đọc các dữ liệu rác, nhưng nếu thấy rằng giao dịch ghi bị huỷ bỏ, chúng ta không có cách nào để hồi phục lại giá trị cũ của mục đã được ghi.

Nếu tất cả các giao dịch đều tuân theo nghi thức khoá hai pha nghiêm ngặt, chúng đảm bảo cho quá trình hoạt động nhất quán và nó không bao giờ phải thực hiện lại.

6.8. NGHI THỨC BẢO TOÀN VÀ TÍCH CỰC

Có hai vấn đề về hiệu quả hoạt động của các giao dịch cần phải đề cập đến:

1. Ta muốn có được một hiệu năng tối ưu, nghĩa là khả năng hoàn tất giao dịch nhanh nhất đối với một tổ hợp các giao dịch đã cho.

2. Ta muốn cho một giao dịch cụ thể hoàn tất mà không có nhiều chậm trễ.

Đôi khi (1) và (2) không tương thích với nhau, và có nhiều tình huống không có người sử dụng nào đang đợi giao dịch kết thúc, vì thế (2) không quan trọng. Cũng có thể có tình huống ở đó (1) không còn là vấn đề tối quan trọng bởi vì khả năng tính toán quá đủ cho các giao dịch cần được xử lý.

Chọn lựa nghi thức và bộ xếp lịch ảnh hưởng đến (1) có nhiều cách, tuy nhiên chúng ta cần tránh huỷ bỏ các giao dịch càng nhiều càng tốt, đặc biệt bằng cách sử dụng một nghi thức và bộ xếp lịch tránh khoá gài, hay chọn tính đơn giản của bộ xếp lịch tiết kiệm được nhiều thời gian quản lý các khoá hơn thời gian tốn kém khi phải giải quyết vấn đề khoá gài?.

Chúng ta sẽ phân loại các nghi thức sau

1. *Kiểu tích cực (aggressive)*, nghĩa là chúng ta cố gắng tiến hành càng nhanh càng tốt, dù rằng có thể dẫn đến tình huống chúng phải huỷ bỏ.

2. *Kiểu bảo toàn (conservative)*, nghĩa là nghi thức tránh thực hiện ngay từ đầu giao dịch nếu nó không bảo đảm rằng giao dịch có thể hoàn tất.

6.8.1.NGHI THỨC BẢO TOÀN

Phiên bản *bảo toàn của khoá hai pha nghiêm ngặt* là *buộc mỗi giao dịch T_i phải đưa ra tất cả yêu cầu về khoá sẽ cần dùng ngay từ lúc bắt đầu giao dịch.* Bộ xếp lịch sẽ trao các khoá và cho phép T_i tiến hành nếu tất cả các khoá đều đã sẵn sàng. Nếu một hoặc nhiều khoá không sẵn sàng, T_i được đặt vào một hàng đợi.

Lược đồ này rõ ràng tránh được *khoá gài* do tranh chấp các khoá.

Để ngăn *khoá sống* bộ xếp lịch không thể cho phép một giao dịch trong hàng đợi đang đợi một trong những khoá mà T cần. Hơn nữa, một khi giao dịch được đưa vào hàng đợi, nó không thể tiến hành, ngay cả khi tất cả các khoá của nó yêu cầu đều sẵn sàng, nếu có một giao dịch khác đứng trước nó trong hàng cũng yêu cầu một trong các khoá của nó.

Thí dụ 6.16:

Giả sử có một chuỗi các giao dịch

$$U_0, V_1, U_1, V_2, U_2...$$

Với đặc tính là mỗi U_i khoá mục A, còn mỗi V_i khoá B và C. Cũng giả sử rằng một giao dịch T_1 được khởi đầu ngay sau U_0 và T_1 cần khoá A và B. Như vậy T_1 phải đợi khoá trên.A và T_1 được đặt vào trong hàng đợi. Tuy nhiên, trước khi U_0 kết thúc, V_1 lại bắt đầu và được trao các khoá B và C. Do vậy khi U_0 kết thúc nó giải phóng khoá A nhưng bây giờ B lại không sẵn sàng, và T_1 không thể tiến hành được. Trước khi V_1 kết thúc, U_1 lại bắt đầu, nó lại ngăn T_1 tiến hành khi V_1 giải phóng khoá B. Theo cách này, T_1 có thể phải đợi mãi.

Tuy nhiên, theo chiến lược ngăn chặn khoá sống được mô tả ở trên, bộ xếp lịch sẽ không trao khoá B cho V_1, bởi vì T_1 đang đợi trong hàng đợi và T_1 cần B. Vì vậy hành động đúng của bộ xếp lịch khi V_1 yêu cầu khoá là đặt V_1 vào hàng đợi sau T_1. Do đó khi U_0 kết thúc, cả hai khoá A và B đều sẵn sàng và được trao cho T_1 . Khi T_1 kết thúc, nó trả lại khóa B và được trao lại cho V_1 cùng với khoá C mà chúng ta giả sử rằng vẫn còn.

Đinh lý 6.6.

Nếu chúng ta dùng một nghi thức trong đó tất cả các khoá được nhận ngay từ lúc bắt đầu giao dịch, và chúng ta sử dụng một bộ xếp lịch để cho phép một giao dịch T (có thể ở trong hàng đợi hoặc không) nhận các khoá theo yêu cầu nếu và chỉ nếu:

1. Tất cả các khoá đều sẵn sàng, và

2. Không có giao dịch nào ở trước T trong hàng đợi cần một trong những khoá mà T yêu cầu.

Khi đó các khoá sống và khoá gài không thể xảy ra.

Chứng minh: Các khoá gài không thể xảy ra là hệ quả trực tiếp của (1); không có một giao dịch nào giữ một khoá nhưng lại đang phải đợi một khoá khác.

Để chứng tỏ rằng không có khoá sống nào xảy ra, giả sử rằng giao dịch T được đặt vào trong hàng đợi. Lúc này, chỉ có một số hữu hạn các giao dịch ở phía trước T, gọi các giao dịch đó là U_1,..., U_k. Các khoá được yêu cầu bởi giao dịch đầu tiên trong số này là U_1 sẽ được giải phóng trong một khoảng thời gian hữu hạn, và không có khoá nào trong chúng được trao cho một giao dịch khác. Vì vậy U_1 sẽ bắt đầu sau một

khoảng thời gian hữu hạn và sẽ được lấy ra khỏi hàng đợi. Chúng ta có thể lập luận tương tự cho $U_2,\ldots,U_k$. Vì vậy cuối cùng T sẽ đến được vị trí đầu hàng đợi, và nó sẽ được trao các khoá theo yêu cầu sau một khoảng thời gian hữu hạn nào đó. Do đó khoá sống không xảy ra.

Cần lưu ý rằng nghi thức bảo toàn có vẻ hấp dẫn, nhưng chúng vẫn gây ra hai trở ngại:

1. Các giao dịch có thể bị chậm trễ một cách không cần thiết, bởi vì chúng có thể cần một khoá mà mãi về sau mới được sử dụng nhưng lại phải yêu cầu ngay từ khi bắt đầu. Nếu khoá đó chưa sẵn sàng, giao dịch không thể bắt đầu, dù rằng nó có thể thực hiện một số bước mà chưa cần đến khoá này.

2. Các giao dịch phải khoá tất cả các mục chúng cần thao tác dù rằng chúng có thể phải làm một số phép tính để xác định xem chúng có thực sự cần đến mục đó hay không.

Thí dụ 6.17:

Trong hình 6.17 ta có một câu vấn tin SQL tìm một khách hàng đã đặt pho mat (Brie). Giả sử rằng các mục là các khối, mỗi mục chứa một số bộ từ một trong các quan hệ ORDERS hoặc INCLUDES, hoặc từ thành phần của một cấu trúc chỉ mục. Chúng ta cũng giả sử rằng có một chỉ mục thứ cấp trên ITEM của INCLUDES và chỉ mục chính trên O# của ORDERS (có thể có những chỉ mục khác nữa).

Cách hiệu quả nhất để trả lời câu vấn tin này là sử dụng chỉ mục ITEM, khoá đọc các khối cần để tìm các bộ của INCLUDES có "Brie" trong thành phần ITEM. Sau đó với tập các mã số đơn đặt hàng có chứa Brie, chúng ta sử dụng chỉ mục O# trên ORDERS để tìm các bộ cho

các đơn đặt hàng này, rồi lại khoá các khối chỉ mục và các khối của quan hệ ORDERS.

```
SELECT CUST
FROM   ORDERS, INCLUDES
WHERE  ORDERS. O# = INCLUDES.O#
       AND ITEM = "Brie";
```

Hình 6.17 Tìm khách hàng đặt Brie

Nếu ngay từ đầu đã phải khoá mỗi khối có thể sẽ dùng đến trong khi thực hiện câu vấn tin của hình 6.17, chúng ta phải yêu cầu một khoá chốt cho khối của hai quan hệ và hai chỉ mục.

Ưu điểm khi giới hạn số lượng các khối cần phải khoá là chúng ta có thể cho phép các thao tác cập nhật, chèn, và xoá tiến hành song song với quá trình vấn tin. Hơn nữa, bằng cách lấy các khoá khi cần, câu vấn tin vẫn được phép tiến hành ngay cả khi một bộ của ORDERS mà ta cần đang được ghi trong thời gian chúng ta truy xuất quan hệ INCLUDES.

6.8.2.NGHI THỨC TÍCH CỰC

Hình thức *tích cực* nhất của khoá hai pha là *chỉ yêu cầu khoá trên một mục ngay trước khi đọc hoặc ghi mục này*. Nếu một mục được ghi sau khi đọc, khoá đọc được lấy trước, rồi được nâng cấp thành khoá ghi khi cần. Dĩ nhiên các khoá chỉ có thể được giải phóng sau khi tất cả các khoá đã được lấy, nếu không chúng ta đã vượt ra khỏi phạm vi của khoá hai pha, và có thể xảy ra các hành động bất khả tuần tự. Cũng vậy,

các khóa vẫn phải được giải phóng ở cuối giao dịch nếu chúng ta muốn tuân theo nghi thức nghiêm ngặt.

Tuy nhiên, hành vi tích cực này có thể dẫn đến các khoá gài. Khả năng các khoá được nâng cấp từ đọc lên ghi làm tăng khả năng gây khoá gài. Chẳng hạn, mỗi giao dịch T_1 và T_2 đều giữ một khoá đọc trên A và không thể tiến hành mà không nâng cấp chúng thành khoá ghi bởi vì mỗi giao dịch đều muốn ghi giá trị mới vào A. Đó là khoá gài, T_1 hoặc T_2 phải bị huỷ bỏ và thực hiện lại.

Chúng ta có thể tránh được khoá gài bằng cách xếp thứ tự các mục và yêu cầu các giao dịch khóa các mục theo đúng thứ tự đó.

6.9. KHÔI PHỤC SAU SỰ CỐ

Trong phần 6.7 ta đã xem xét những điều cần phải thực hiện để xử lý các giao dịch đơn độc gặp sự cố. Bây giờ ta xét những trường hợp phức tạp hơn của các sự cố phần mềm và phần cứng. Những sự cố như thế có hai mức độ nghiêm trọng, phụ thuộc vào dữ liệu nào bị mất. Bộ nhớ (memory) có thể phân chia thành bộ *nhớ không tự duy trì* (volatilestorage), trên đó dữ liệu đang lưu sẽ bị mất khi có sự cố, chẳng hạn như khi bị mất điện, và *bộ nhớ ổn định* (stablestorage), trên đó dữ liệu vẫn còn tồn tại trừ những trường hợp đặc biệt . Bộ nhớ là những thí dụ về bộ *nhớ không tự duy trì*, còn đĩa và băng từ là bộ nhớ ổn định. Trong những phần sau, chúng ta sẽ sử dụng thuật ngữ *"bộ nhớ thứ cấp"* (secondary memory) đồng nghĩa với "bộ nhớ ổn định" và *"bộ nhớ chính"* (main memory) được dùng cho "bộ nhớ không tự duy trì".

Chúng ta sẽ xem việc mất dữ liệu trên bộ nhớ chính là sự cố *hệ thống* (system failure), còn việc hư hại thiết bị lưu trữ thứ cấp là sự *cố vật liệu* (media failure).

Một CSDL không làm mất dữ liệu khi có sự cố (hệ thống hoặc vật liệu) xảy ra gọi là *thích ứng được* đối với loại sự cố đó.

6.9.1. NHẬT KÝ

Nhật ký (log hay Journal) là công cụ thông dụng nhất dùng để chống lại tình trạng bị mất dữ liệu khi xảy ra sự cố hệ thống. *Nhật ký* là bản ghi chép các thay đổi được thực hiện trên CSDL và tình trạng của mỗi giao dịch. Nghĩa là các sự kiện sau đây được ghi nối đuôi vào cuối nhật ký:

1. Khi giao dịch T bắt đầu, chúng ta cần ghi mẫu tin (T, begin)

2. Khi giao dịch T yêu cầu ghi một giá trị mới v vào mục A, chúng ta ghi nối đuôi vào nhật ký mẫu tin (T, A, v). Nếu có khả năng giao dịch này bị huỷ bỏ thì mẫu tin này cũng chứa giá trị cũ của A. Cũng vậy nếu mục A là một đối tượng có kích thước lớn, chúng ta nên để v dưới dạng "mã" nào đó thay cho đổi trong A toàn bộ giá trị mới của A.

3. Nếu giao dịch T uỷ thác, chúng ta ghi (T, commit)

4. Nếu giao dịch T bị huỷ bỏ, chúng ta ghi (T, abort)

Thí dụ 6.18:

Đây là một thí dụ về cách sử dụng nhật ký để xử lý những giao dịch bị huỷ bỏ nhưng nó cũng minh họa nhiều vấn đề về nhật ký và sự

cố hệ thống. Giả sử thực hiện 14 bước của hình 6.16, sau đó T_1 bị huỷ bỏ. Bởi vì một hệ thống cho phép dùng lịch biểu của hình 6.16 rõ ràng không sử dụng nghi thức khoá hai pha nghiêm ngặt, chúng ta phải cho phép thực hiện cuộn ngược các giao dịch. Vì thế, khi chúng ta ghi giá trị mới v cho mục A có giá trị cũ u, chúng ta ghi mẫu tin (T, A, u, v). Để có thể tính được các giá trị thực sự, chúng ta giả sử rằng giá trị ban đầu của mục A là 10 trong lịch biểu hình 6.16.

Hình 6.18 trình bày các mẫu tin được ghi vào nhật ký và chỉ ra các bước ghi chép các *khoản nhật ký* (log entry). Như chúng ta sẽ thấy, điều quan trọng là các khoản nhật ký phải được ghi trước khi hành động mà nó mô tả thực sự xảy ra trong CSDL.

Bước	Khoảng nhật ký
Trước bước (1)	(T_1, begin)
(4)	(T_1, A, 10, 9)
Trước bước (7)	(T_2, begin)
(11)	(T_2, A, 9, 18)
(12)	(T_2, commit)
Sau bước (14)	(T_1, abort)

Hình 6.18 Các khoản nhật ký của hình 6.16

Thí dụ 6.18 đã đưa ra cách sử dụng nhật ký để khôi phục lại sau sự cố mà chúng ta đã giả sử là xảy ra sau bước (14) của hình 6.16. Nó cũng đưa ra một số vấn đề sẽ nảy sinh khi chúng ta không sử dụng khoá hai pha nghiêm ngặt. Trước tiên chúng ta kiểm tra nhật ký và phát hiện ra rằng T_1 đã bị huỷ bỏ, và vì vậy chúng ta phải cuộn ngược CSDL về tình trạng trước khi T_1 bắt đầu. Chúng ta không gặp khó khăn khi tìm

mẫu tin (T_1, begin) nhờ quét ngược lại từ cuối nhật ký. Chúng ta cũng có thể tìm ra mẫu tin (T_1, A, 10, 9) và phát hiện ra rằng 10 là giá trị phải được hồi phục lại cho A.

Điều không hiển nhiên là T_2 phải được thực hiện lại. Dù rõ ràng là T_2 ghi A, nhật ký không ghi nhận rằng T_2 đọc A khi T_1 ghi A. Vì vậy, nếu chúng ta muốn giải quyết vấn đề cuộn ngược, thì cần phải ghi nhận trong nhật ký các hành động đọc cũng như các hành động ghi. Và bây giờ chúng ta phải thực hiện lại T_2 với giá trị của A là 10 ở vị trí của giá trị 9 đã dùng trước đó. Tuy nhiên, dù có mẫu tin (T_2, begin) trong nhật ký, và T_2 có thể là dấu nhận dạng duy nhất cho giao dịch đó, ta cũng không biết chính xác phần chương trình nào T_2 biểu diễn. Vì vậy nếu định thực hiện T_2, chúng ta cần phải có một dấu biểu thị thủ tục dùng để thực hiện T_2. Hơn nữa, chúng ta cần giữ phần chương trình đó vô hạn định, bởi vì không có giới hạn về thời gian giữa lúc thực hiện một giao dịch và lúc phát hiện ra rằng nó có mặt trong một quá trình cuộn ngược.

6.9.2. NGHI THỨC THÍCH ỨNG

Sau đây chúng ta nêu một phương pháp đơn giản, bàn về một nghi thức *thích ứng được* ngay cả khi có sự cố liên hệ thống. Có nhiều phương pháp khác nhau đang được sử dụng, nhưng đây có lẽ là một phương pháp dễ hiểu, dễ cài đặt và đơn giản nhất. Nghi thức này được gọi là *nghi thức tái thực hiện (redo protocol).*

Nghi thức tái thực hiện là một dạng tinh chỉnh của khoá chốt hai pha nghiêm ngặt. Khi đạt đến điểm uỷ thác của một giao dịch T, những điều sau đây phải xảy ra theo đúng thứ tự:

1. Đối với mỗi mục A có một giá trị mới v được ghi bởi giao dịch T, chúng ta ghi nối đuôi mẫu tin (T, A, v) vào nhật ký.

2. Ghi nối đuôi (T, commit) vào nhật ký.

3. Ghi vào bộ nhớ thứ cấp các khối ở cuối nhật ký nhưng chưa được ghi vào đó. Ở điểm này, T gọi là *đã uỷ thác* (committed).

4. Đối với mỗi mục A, ghi giá trị mới v vào vị trí của A trong CSDL. Thao tác ghi này có thể thực hiện bằng cách đưa khối chứa A vào bộ nhớ chính và cập nhật nó ở đây. Chúng ta có thể ghi khối của A trở lại vào trong bộ nhớ thứ cấp ngay sau đó.

Thí dụ 6.19:

Trong hình 6.19 chúng ta có một giao dịch T tuân theo *nghi thức tái thực hiện*, và bên cạnh T chúng ta có các mục nhật ký tương ứng. Điểm uỷ thác đạt được giữa bước (3) và (4).

(1)		(T, begin)
(2)	LOCK A	
(3)	LOCK B	
(4)		(T, A, v)
(5)		(T, B, u)
(6)		(T, commit)
(7)	WRITE A	
(8)	WRITE B	
(9)	UNLOCK A	
(10)	UNLOCK B	

Hình 6.19 Giao dịch tuân theo nghi thức tái thực hiện

Giả sử rằng tất cả các phép tính trong vùng làm việc được thực hiện giữa bước (3)và (4). Nhật ký được ghi vào bộ nhớ thứ cấp giữa các bước (6) và (7).

6.9.3.THUẬT TOÁN KHÔI PHỤC "TÁI THỰC HIỆN"

Khi một sự cố hệ thống xảy ra, chúng ta thực hiện một *thuật toán khôi phục* (recovery algorithm) để kiểm tra nhật ký và hoàn trả lại CSDL về một *trạng thái nhất quán*. Đồng thời hệ thống cũng cần phải giải phóng các khoá đã bị giữ tại thời điểm xảy ra sự cố, bởi vì giao dịch đã giữ chúng sẽ được thực hiện lại và xin lại các khoá này, hoặc giao dịch đã uỷ thác nhưng chưa giải phóng chúng. Trong trường hợp sau, giao dịch sẽ không được tái hoạt động, và vì vậy không có cơ hội để tự giải phóng các khoá.

Nếu các giao dịch tuân theo nghi thức tái thực hiện, chúng ta có thể sửa chữa CSDL bằng thuật toán tái thực hiện đơn giản sau đây:

Chúng ta bắt đầu kiểm tra nhật ký từ mẫu mới nhất trở ngược lại, và xác định xem những giao dịch T nào có mẫu tin nhật ký (T, commit). Đối với mỗi giao dịch T như thế, chúng ta kiểm tra mỗi mẫu tin (T, A, v) và ghi giá trị v của mục A vào CSDL đang tồn tại trong bộ nhớ thứ cấp. Chú ý rằng v có thể đã là giá trị của A lúc đó, bởi vì hoạt động của T có thể đã tiến hành các thay đổi và đã được ghi vào bộ nhớ thứ cấp.

Các giao dịch chưa ghi mẫu tin commit vào nhật ký, hoặc đã ghi mẫu tin abort sẽ bị thuật toán "tái thực hiện" bỏ qua; vì ta có các mẫu tin (T, begin) trong nhật ký, chúng ta có thể biết rằng giao dịch T không hoàn tất nếu chúng ta không tìm thấy (T, commit), hoặc ta tìm thấy

(T,Abort). Một khi xác định những giao dịch nào đã uỷ thác, chúng ta có thể quét qua nhật ký theo hướng tới trước. Mỗi khi chúng ta đọc được một mẫu tin (T, A, v) trong đó T là một giao dịch đã uỷ thác, chúng ta ghi v vào mục A. Chú ý rằng không có cách nào để từ nhật ký chúng ta biết được bản sao của A trong CSDL đã được cập nhật theo mẫu tin nhật ký này hay chưa. Có thể là:

1. Sự cố đã xảy ra ngay sau khi mẫu tin (T, **commit**) được đặt vào trong nhật ký mà không đủ thời gian để cập nhật bản sao của A trong bộ nhớ chính, hoặc

2. Bản sao của A trong bộ nhớ chính được cập nhật, nhưng sự cố xảy ra trước khi khối đang chứa A được ghi vào bộ nhớ thứ cấp.

6.9.4.BẢO VỆ DỮ LIỆU DO SỰ CỐ VẬT LIỆU

Những sự cố vật liệu, trong đó chính CSDL ổn định bị huỷ hoại. Sự cố vật liệu có thể xẩy ra do"vỡ đầu đọc" của một đĩa, hoặc hoả hoạn hay do tác động của một nam châm cực lớn...; sự tiêu huỷ vũ khí hạt nhân hàng loạt cũng gây ra những tổn thất về sự cố vật liệu. Rõ ràng chúng ta không thể ngăn được hậu quả mất dữ liệu trong tất cả những tình huống như vậy, nhưng chúng ta có thể làm cho hệ thống đưa ra thông báo khi sắp có sự cố.

Trong nhiều hệ thống máy tính, phương pháp chính để ngăn chặn tình trạng mất dữ liệu do sự cố vật liệu là chép dự phòng theo đúng định kỳ. Chẳng hạn vào mỗi buổi tối, mỗi tập tin có thay đổi kể từ lần sao chép dự phòng cuối cùng sẽ được sao chép dự phòng lại. Nếu một tập tin bị thay đổi trong quá trình chép dự phòng, có thể nó không được ghi nhận, và không ai quan tâm đến điều đó ngoại trừ chủ nhân của tập tin

đó. Hành động thiếu chu đáo này là chấp nhận được đối với một số hệ thống CSDL nhưng cũng có một số ứng dụng CSDL quan trọng mà trong đó chúng ta phải làm cho xác xuất bị mất dữ liệu do sự cố vật liệu càng thấp càng tốt; hệ thống ngân hàng và hệ thống đặt chỗ là hai thí dụ thuộc loại này.

Các công cụ chủ yếu để đảm bảo tính thích ứng được khi có sự cố môi trường là tạo ra liên tục một bản sao lưu cho toàn bộ CSDL. Nếu chúng ta giữ bản sao lưu trên một tập đĩa riêng, không phải là tập đĩa chúng ta dùng cho CSDL thực sự, và chúng ta giữ cả hai tập đĩa đủ xa để chúng không thể bị huỷ cùng lúc, thì chúng ta đã có một biện pháp bảo vệ tuyệt vời chống lại tình trạng mất dữ liệu. Chúng ta có thể sửa đổi kỹ thuật ghi nhật ký và đánh dấu điểm kiểm tra sao cho phù hợp với việc tạo một bản sao lưu cho CSDL bằng các nhận xét sau:

1. Bản thân nhật ký cũng phải được nhân bản. Khi chúng ta ghi một khối của nhật ký vào bộ nhớ thứ cấp, chúng ta cũng phải sao chép khối nhật ký đó vào bản lưu. Nếu một sự cố vật liệu xảy ra giữa những lần thực hiện hai bản sao thì một phần nhật ký có thể hoặc không thể hồi phục được, tuỳ thuộc vào thiết bị lưu bản sao nào bị hư. Giao dịch mà thao tác uỷ thác của nó làm cho hệ thống phải ghi nhật ký vào bộ nhớ thứ cấp có thể không xuất hiện như đã uỷ thác, nhưng đó là trường hợp xấu nhất có thể xảy ra. Trong quá trình khôi phục, chúng ta thông báo rằng giao dịch đã bị thất lạc bởi vì chúng ta tìm thấy mẫu tin begin trong một nhật ký.

2. Khi đánh dấu điểm kiểm tra, chúng ta phải bảo đảm rằng các khối đã được sửa đổi từ điểm kiểm tra cuối cùng đã được chép vào bản lưu. Những khối này bao gồm những khối đang nằm trong bộ nhớ chính tại thời điểm bắt đầu đánh dấu điểm kiểm tra, và vì thế là những khối sẽ

được sao chép vào CSDL trong bộ nhớ thứ cấp trong khi đánh dấu điểm kiểm tra. Đồng thời trong bản lưu cũng phải chứa các khối đã được ghi trước kia vào trong bộ nhớ thứ cấp của CSDL trong quá trình hoạt động bình thường của bộ quản lý phân trang. Những mục có các giá trị được ghi ít nhất một lần thì bằng cách kiểm tra nhật ký kể từ điểm kiểm tra cuối cùng, chúng ta có thể tìm ra những khối cho các mục đó.

6.10. ĐIỀU KHIỂN HOẠT ĐỘNG ĐỒNG THỜI BẰNG NHÃN THỜI GIAN

Trong những phần trước đã giả sử rằng cách duy nhất để bảo đảm được tính khả tuần tự của các giao dịch là duy trì và gán các khoá trên các mục. Tuy nhiên, cũng có những phương pháp khác để đạt được tính khả tuần tự. Chẳng hạn, để cho bộ xếp lịch xây dựng một *đồ thị tương tranh* dựa trên tất cả thao tác đọc và ghi đã trao cho các giao dịch đang hoạt động và chỉ cho phép một truy xuất được thực hiện nếu nó không tạo ra một chu trình trong đồ thị đó. Ở đây chúng ta sẽ nêu một cách tiếp cận khác đã được dùng trong thực hành, và nó thích hợp với kiểu khoá chốt tích cực.

Ý tưởng của phương pháp này là gán cho mỗi giao dịch một *nhãn thời gian* (timestamp) cho biết thời điểm bắt đầu của giao dịch. Chẳng hạn mỗi khi một giao dịch bắt dầu, hệ thống sẽ gọi bộ xếp lịch và giao dịch sẽ được trao cho một nhãn thời gian, hoặc giả bộ xếp lịch sẽ trao nhãn thời gian khi lần đầu tiên giao dịch yêu cầu truy xuất một mục nào đó của CSDL.

Với phương pháp dán nhãn thời gian, thứ tự tuần tự tương đương đơn giản chỉ là thứ tự nhãn thời gian của các giao dịch. Hai cách tiếp cận khoá và nhãn thời gian có đặc điểm khác nhau: một lịch biểu có thể tương đương với một lịch biểu tuần tự theo khoá nhưng có thể không tương đương với lịch biểu tuần tự theo nhãn thời gian, hoặc ngược lại. Thí dụ sau sẽ minh họa điều này.

Thí dụ 6.20:

Xét các giao dịch của hình 6.20. Bởi vì T_2 bắt đầu trước T_1, thứ tự tuần tự theo nhãn thời gian là T_2, T_1. Ngược lại, nếu khoá chốt được dùng để tuần tự hoá những giao dịch này thì rõ ràng, T_1 nhận một khoá trên C trước khi T_2 nhận. Vì vậy T_1 đạt điểm khoá ở trước bước (3), trong khi T_2 không thể đạt đến điểm này cho đến sau bước (3). Điều này chứng tỏ rằng nếu phương pháp khoá chốt được dùng, thứ tự tuần tự tương đương là T_1, T_2. Nói một cách khác, nếu chúng ta dùng phương pháp khoá hai pha, lịch biểu trong hình 6.20 là một lịch biểu khả hữu, nhưng nếu chúng ta dùng nhãn thời gian để điều khiển các hoạt động đồng thời, chúng ta không thể cho phép một chuỗi các sự kiện như thế xảy ra.

	T_1	T_2
(1)		READ B
(2)	READ A	
(3)	WRITE C	
(4)		WRITE C

Hình 6.20 Lịch biểu khả tuần tự theo khoá, không phải theo nhãn thời gian

Ngược lại, hãy xét lịch biểu của hình 6.21. Chúng ta có thể nói rằng T_2 không đạt đến điểm khoá cho đến sau bước (7) bởi vì T_1 đã có một khoá trên B, và như thế T_2 không thể có khoá B cho đến lúc đó. Tuy nhiên T_3 đã kết thúc kể từ bước (6), và vì thế đạt được điểm khoá của nó trước T_2. Vì vậy trong lịch biểu tuần tự dựa trên khoá, T_3 đi trước T_2. Nhưng rõ ràng là trong lịch biểu dựa trên nhãn thời gian T_2 đi trước T_3. Thứ tự nào là đúng? Chỉ có thứ tự T_2, T_3 mới có thể xuất hiện trong một lịch biểu tuần tự tương đương, bởi vì trong hình 6.21, T_3 ghi một giá trị A sau khi T_2 đọc A và nếu thứ tự tuần tự có T_3 trước T_2 thì T_2 có thể đọc sai giá trị được ghi bởi T_3. Vì thế hình 6.21 là thí dụ về một lịch biểu chúng ta có thể dùng nhãn thời gian để điều khiển đồng thời nhưng không thể dùng khoá chốt.

	T_1	T_2	T_3
(1)	READ A		
(2)		READ A	
(3)			READ D
(4)			WRITE D
(5)			WRITE A
(6)		READ C	
(7)	WRITE B		
(8)		WRITE B	

Hình 6.21 Lịch biểu khả tuần tự theo nhãn thời gian, không phải theo khoá

Mấu chốt của thí dụ vừa nêu là khẳng định rằng với hai phương pháp khoá và phương pháp nhãn thời gian, chúng ta không thể nói loại

nào là ưu việt hơn loại nào. Mỗi loại cho phép một số lịch biểu và cấm một số khác.

6.10.1.THIẾT LẬP NHÃN THỜI GIAN

Trong phương pháp dùng nhãn thời gian chúng ta có thể bảo đảm rằng không có hai giao dịch lấy cùng nhãn thời gian, và thứ tự tương đối của nhãn thời gian tương ứng với thứ tự trong đó giao dịch bắt đầu. Một cách tiếp cận khác là dùng trị số của đồng hồ hệ thống tại thời điểm một tiến trình khởi đầu làm nhãn thời gian của tiến trình.

Nếu có nhiều tiến trình có thể gán nhãn thời gian, chẳng hạn do:

1. Hệ thống CSDL đang chạy trên một máy có nhiều bộ xử lý, và có thể tồn tại nhiều bộ xếp lịch, hoặc

2. CSDL được phân tán trên nhiều máy, như sẽ thảo luận trong phần sau thì chúng ta phải chọn một *hậu tố* (suffix) duy nhất cho mỗi nhãn thời gian được sinh ra từ bộ xử lý đó. Chẳng hạn, nếu không có hơn 256 bộ xử lý, chúng ta có thể gán vào đuôi nhãn một chuỗi 8 bit để nhận dạng bộ xử lý. Chúng ta cũng phải sắp xếp để bộ đếm hoặc đồng hồ được dùng bởi mỗi bộ xử lý luôn đồng bộ; nhưng làm thế nào để thực hiện được? Điều này được giải thích trong phần sau.

6.10.2.BẢO ĐẢM TÍNH KHẢ TUẦN TỰ BẰNG NHÃN THỜI GIAN

Trong lược đồ đọc - ghi, chúng ta đã liên kết mỗi mục trong CSDL với hai thời điểm, *thời điểm đọc* (read - time), là nhãn thời gian cao nhất có được do một giao dịch đã đọc mục đó, và *thời điểm ghi* (write - time), là nhãn thời gian cao nhất do một giao dịch đã ghi mục

đó. Bằng cách thực hiện như thế, chúng ta có thể cho rằng mỗi giao dịch sẽ thực hiện ngay lập tức, tại thời điểm được chỉ ra bằng nhãn thời gian.

Chúng ta sử dụng nhãn thời gian đi kèm với giao dịch, và các thời điểm đọc- ghi các mục để kiểm tra xem những tình huống nào không thể xảy ra được.

1. Không thể xảy ra trường hợp một giao dịch đọc được giá trị của mỗi mục nếu giá trị đó chưa được ghi cho đến sau lúc thực hiện giao dịch đó. Nghĩa là, một giao dịch có nhãn thời gian t_1 không thể đọc một mục có thời điểm ghi là t_2 nếu $t_2 > t_1$. Nếu một trường hợp như thế xảy ra thì giao dịch có nhãn thời gian t_1 phải bị huỷ bỏ và được khởi động lại với một nhãn thời gian mới.

2. Không thể xảy ra trường hợp giao dịch ghi một mục nếu giá trị cũ của mục đó được đọc tại một thời điểm sau đó. Nghĩa là một giao dịch có nhãn thời gian t_1 không thể ghi một mục có thời điểm đọc t_2 nếu $t_2 > t_1$. Giao dịch có nhãn thời gian t_1 phải bị huỷ bỏ và được khởi động lại với một nhãn thời gian mới.

Chúng ta có quy tắc duy trì thứ tự bằng nhãn thời gian như sau:

Giả sử có một giao dịch có nhãn thời gian t đang muốn thực hiện thao tác X trên một mục có thời điểm đọc t_r và thời điểm ghi t_w:

a) Cho thực hiện thao tác này nếu X = READ và $t \geq t_w$ hoặc nếu X = WRITE, $t \geq t_r$ và $t \geq t_w$ trong trường hợp trước, đặt thời điểm đọc là t nếu $t > t_r$ và trong trường hợp sau, đặt thời điểm ghi là t nếu $t > t_r$.

b) Không thực hiện điều gì nếu X = WRITE và $t_r \leq t < t_w$

c) Huỷ bỏ giao dịch này nếu X = READ và $t < t_w$ hoặc

X = WRITE và $t < t_r$.

Thí dụ 6.21:

Chúng ta hãy xem lại các giao dịch của hình 6.1 được trình bày lại trong hình 6.22. với thời điểm đọc (read - time, RT), và thời điểm ghi (write - time, WT) của mục A được chỉ ra khi nó bị thay đổi. Giả sử T_1 được cho nhãn thời gian là 150 và T_2 là 160. Chúng ta cũng giả sử thời điểm đọc và thời điểm ghi của A ban đầu đều là 0. Vậy A được cho RT 150 khi T_1 đọc nó và 160 ở bước kế tiếp khi T_2 đọc nó. Tại bước thứ (5) khi T_2 đọc A, nhãn thời gian của T_2 là 160, không nhỏ hơn RT của A, cũng là 160, cũng không nhỏ hơn WT của A là 0. Vậy thao tác ghi này được phép, và WT của A được dặt là 160. Khi T_1 cố gắng ghi vào A ở bước cuối cùng, nhãn thời gian của nó là 150 nhỏ hơn RT của A (160), vì vậy T_1 bị huỷ bỏ, ngăn được kết quả bất thường của hình 6.1.

Một chuỗi sự kiện tương tự cũng xảy ra nếu nhãn thời gian của T_1 lớn hơn T_2 . Do đó T_2 bị huỷ bỏ ở bước (5).

Thí dụ 6.22:

Hình 6.23 minh hoạ ba giao dịch, với các nhãn thời gian lần lượt là 200, 150 và 175, hoạt tác trên ba mục A, B, C, với thời điểm ghi và đọc đều được giả sử là 0 vào lúc khởi đầu. Ba cột cuối cùng chỉ ra các thay đổi đối với các thời điểm ghi và đọc của các mục.

Tại bước (6) có một thao tác ghi C được thực hiện. Tuy nhiên, giao dịch đang thực hiện thao tác ghi là T_2 có nhãn thời gian 150, và thời điểm đọc của C là 175. Vì thế T_2 không thể thực hiện được thao tác ghi này và bị huỷ bỏ. Sau đó tại bước (7), T_3 thực hiện ghi A. Bởi vì T_3 có nhãn thời gian 175 lớn hơn thời điểm đọc của A là 150. T_3 không

cần phải huỷ bỏ. Tuy nhiên thời điểm ghi của A là 200, vì thế giá trị của A được T_3 ghi không được đưa vào CSDL.

	T_1	T_2	T_3
	150	160	RT = 0 WT = 0
(1)	READ A		RT = 0
(2)		READ A	RT = 160
(3)	A: = A + 1		
(4)		A: = A + 1	
(5)		WRITE A	WT = 160
(6)	WRITE A T_1 bị huỷ bỏ		

Hình 6.22 Các giao dịch của hình 6.1 sử dụng nhãn thời gian

	T_1	T_2	T_3	A	B	C
	200	150 WT = 0	175 WT = 0	TR = 0 WT = 0	RT = 0	RT = 0
(1)	READ B				RT = 200	
(2)		READ A		RT = 150		
(3)			READ C			RT = 75
(4)	WRITE B				WT = 200	
(5)	WRITE A			WT = 200		
(6)		WRITE C T_2 huỷ bỏ				
(7)			WRITE A			

Hình 6.23 Các giao dịch được điều khiển bằng nhãn thời gian

6.10.3.NHẬT KÝ VÀ CUỘN NGƯỢC DÂY CHUYỀN

Trong mọi tình huống để giải quyết các vấn đề rắc rối do sự cố chúng ta phải duy trì một nhật ký.

Khi cho phép ghi vào CSDL trước khi giao dịch đạt đến điểm uỷ thác, chúng ta thường gặp những vấn đề rắc rối nếu chúng ta phải khôi phục lại do sự cố hệ thống. Bất kể công việc điều khiển đồng thời được duy trì nhờ nhãn thời gian, khoá hay một cơ chế nào khác, chúng ta vẫn cần đặt các mẫu tin (T, **begin**), và (T, **commit**) hoặc (T, **abort**) hay (T,A,v), ... vào trong nhật ký cho mỗi giao dịch T.

6.10.4.ĐIỀU KHIỂN HOẠT ĐỘNG ĐỒNG THỜI THEO NHÃN THỜI GIAN NGHIÊM NGẶT

Khi dùng nhãn thời gian, tính nghiêm ngặt gây ra một rắc rối nhỏ. Chúng ta huỷ bỏ một giao dịch T nếu nó cố gắng ghi một mục A và thấy thời điểm đọc của A vượt quá nhãn thời gian của T. Vì vậy thao tác kiểm tra nhãn thời gian phải được thực hiện trước điểm uỷ thác.

Chẳng hạn, giả sử rằng T có nhãn thời gian là 100, và T quyết định ghi vào A. Nó phải kiểm tra để bảo đảm thời điểm đọc của A nhỏ hơn 100, và cũng phải thay thời điểm ghi của A bằng 100; nếu nó không thay đổi thời điểm ghi của A vào lúc này, một giao dịch khác, giả sử có nhãn thời gian là 110, có thể đọc A giữa lúc này và thời điểm T đạt điểm uỷ thác. Trong trường hợp đó, ta sẽ phải kiểm tra lại thời điểm đọc của A (bây giờ là 110) và phải huỷ bỏ sau khi T đã đạt đến điểm uỷ thác rồi.

Tuy nhiên bây giờ chúng ta lại gặp một tình huống trong đó T đã thay đổi thời điểm ghi của A, nhưng thực sự chưa ghi vào CSDL giá trị được xem là đã ghi vào lúc đó; T thực sự không thể ghi giá trị này, bởi vì T vẫn có thể bị huỷ bỏ, và chúng ta muốn tránh tình trạng cuộn ngược dây chuyền. Điều duy nhất chúng ta có thể thực hiện là cho giao dịch T một khoá trên A, và cho A giữ từ lúc T thay đổi thời điểm ghi của A đến lúc T cung cấp giá trị tương ứng. Nếu T phải huỷ bỏ trong khoảng thời gian này, khoá phải được giải phóng và thời điểm ghi của A được hồi phục lại .

Có hai cách tiếp cận khác nhau nhằm thực hiện việc kiểm tra khi một giao dịch T đọc hay ghi một mục A:

1. Kiểm tra thời điểm ghi của A lúc T đọc A, và kiểm tra thời điểm đọc của A lúc T ghi giá trị của A vào vùng làm việc, hoặc

2. Kiểm tra thời điểm đọc của A (nếu T đã ghi A) và thời điểm ghi của A (nếu T đã đọc A) lúc T uỷ thác.

Trong mỗi trường hợp, khi ghi A, chúng ta phải duy trì một khoá trên A từ lúc kiểm tra đến lúc giá trị được ghi. Tuy nhiên theo cách (1), các khoá được giữ trong một thời gian dài, trong khi theo cách (2), các khoá được giữ trong một thời gian ngắn, đủ để các mục khác được ghi bởi T cũng được kiểm tra về thời điểm đọc của chúng. Ngược lại, chiến lược (2), thường được gọi là *điều khiển đồng thời lạc quan* (optimistic concurrency control), kiểm tra nhãn thời gian trễ hơn (1), và vì thế sẽ phải huỷ bỏ nhiều giao dịch hơn (1).

Tóm lại, các bước để uỷ thác một giao dịch theo chiến lược lạc quan, là mục (2) ở trên, được thực hiện như sau.

i) Khi một giao dịch T kết thúc công việc tính toán, chúng ta kiểm tra thời điểm đọc của tất cả các mục mà T muốn ghi vào CSDL;

các khoá được lấy trên tất cả các mục này. Nếu có một mục có thời điểm đọc trễ hơn nhãn thời gian của T, chúng ta buộc phải huỷ bỏ T. Chúng ta cũng cần kiểm tra thời điểm ghi của các mục được T đọc, và nếu có thời điểm nào quá trễ thì hãy huỷ bỏ T. Ngược lại T đã đạt đến điểm uỷ thác.

ii) Ghi các giá trị của T vào trong nhật ký.

iii) Ghi nối đuôi một mẩu tin **commit** cho T vào nhật ký và chép phần cuối của nhật ký vào bộ nhớ thứ cấp.

iv) Ghi các giá trị của T vào CSDL.

v) Giải phóng các khoá đã lấy ở bước (i).

Nếu chúng ta dùng chiến lược (1) thì khác biệt duy nhất là bước (i) không được thực hiện.

6.10.5.CÁCH TIẾP CẬN ĐA PHIÊN BẢN

Mỗi lần một mục được ghi, chúng ta tạo ra một phiên bản mới và gán cho nó thời điểm ghi bằng với nhãn thời gian của giao dịch đang ghi. Khi một giao dịch với nhãn thời gian t muốn đọc một mục A, nó tìm phiên bản của A có thời điểm ghi cao nhất không vượt quá t và đọc phiên bản này.

Điều duy nhất gây rắc rối là nếu một giao dịch T với nhãn thời gian t muốn ghi A, và chúng ta thấy có một phiên bản A_i của A có thời điểm ghi nhỏ hơn t và thời điểm đọc lớn hơn t. Bởi vì giao dịch đã đọc A_i có nhãn thời gian lớn hơn t, do vậy đã đọc giá trị được ghi bởi T với nhãn thời gian t chứ không phải phiên bản cũ hơn. Cách duy nhất để hiệu chỉnh vấn đề này là huỷ bỏ T.

Thí dụ 6.23:

Gọi A là phiên bản của A tồn tại trước khi T_1 và T_2 chạy. Tại các bước (1) và (2) (xem hình 6.22), thời điểm đọc của A đầu tiên được tăng lên 150, rồi 160. Tại bước (5), T_2 ghi một phiên bản mới của A, gọi nó là A_1 với thời điểm ghi là 160. Tại bước (6), T_1 cố tạo ra một phiên bản mới của A, nhưng thấy rằng có một phiên bản khác là A_0, với thời điểm đọc (160) lớn hơn nhãn thời gian T_1 (150) và thời điểm ghi (0) nhỏ hơn nhãn thời gian của T_1.Vì vậy T_1 vẫn bị huỷ bỏ.

Thí dụ 6.24:

Xét lịch biểu của hình 6.24. Hai giao dịch, T_1 có nhãn thời gian 100, T_2 có nhãn thời gian 200 truy xuất các mục A và B. Các phiên bản khởi đầu của các mục này là A_0 và B_0 và chúng ta giả sử rằng chúng có các thời điểm đọc và ghi 0. T_2 tạo ra một phiên bản mới của B có thời điểm ghi là 200 và T_1 tạo ra một phiên bản mới của A có thời điểm ghi là 100; chúng ta gọi chúng tương ứng là A_1 và B_1. Ưu điểm của đa phiên bản được thấy ở bước (4), khi T_1 đọc B. Bởi vì T_1 có nhãn thời gian là 100, nó cần biết giá trị của B đã có lúc đó. Dù rằng T_2 ghi B tại bước (3), giá trị B_0, đã tồn tại từ thời điểm 0 đến 199 vẫn còn sẵn sàng cho T_1, và giá trị này là giá trị được bộ xếp lịch trả về cho T_1.

	T_1	T_2	A_0	A_1	B_0	B_1
	100	200	RT = 0		RT = 0	
			WT = 0		WT = 0	
(1)	READ A		RT = 100			
(2)		READ A	RT = 200			
(3)		WRITE B				WT= 200
(4)	READB				RT= 100	
(5)	WRITEA			WT = 100		

Hình 6.24 Một lịch biểu đa phiên bản

Xếp lịch đa phiên bản là dạng điều khiển đồng thời bằng nhãn thời gian bảo toàn. Rõ ràng nó tạo ra ít tình huống phải huỷ bỏ hơn những cách tiếp cận khác đã được nghiên cứu trong phần này, mặc dù nó tạo ra một số tình huống phải huỷ bỏ mà khoá chốt hai pha bảo toàn không gây ra (phương pháp sau không gây ra bất kỳ tình huống nào). Khuyết điểm của phương pháp xếp lịch đa phiên bản là:

1. Chúng ta tốn thêm không gian lưu trữ.

2. Kỹ thuật truy xuất phức tạp hơn so với phương pháp đơn phiên bản.

3. DBMS phải phát hiện ra khi nào thì mọi giao dịch đang hoạt động không còn truy xuất một phiên bản cũ được nữa, vì vậy nó có thể bị xoá.

Chúng ta có thể xây dựng một thuật toán để đạt được (3).

6.11. GIAO DỊCH PHÂN TÁN

Một CSDL phân tán bao gồm một tập các nút (node), mỗi nút biểu diễn cho một máy tính và thiết bị lưu trữ thứ cấp kèm theo. Có thể một số nút có thiết bị lưu trữ nhỏ hoặc không có, và một số nút khác chỉ là thiết bị lưu trữ kèm với khả năng tính toán tối thiểu cần thiết cho việc lưu và truy xuất dữ liệu. Một số cặp máy tính có thể nối với nhau bằng các đường liên lạc (link), cho phép dữ liệu hoặc thông tin được gửi trực tiếp từ máy này sang máy khác theo một trong hai chiều.

Cần lưu ý rằng, các nối kết giữa các nút có thể thông qua đường dây điện thoại, tốc độ truyền dữ liệu thấp hơn rất nhiều so với tốc độ thực hiện lệnh trong máy tính. Do vậy, chúng ta phải giữ liên lạc ở mức tối

thiểu khi thực hiện các giao dịch, quản lý các khoá hoặc nhãn thời gian, và thực hiện uỷ thác giao dịch.

6.11.1.TÍNH THÍCH ỨNG CỦA CÁC MẠNG

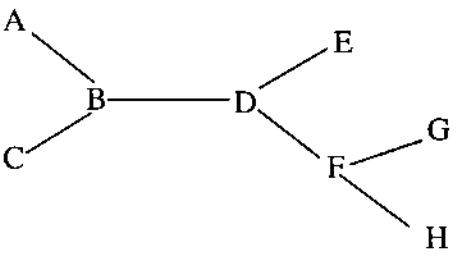
Chúng ta muốn hệ CSDL phân tán vẫn tiếp tục hoạt động khi một đường liên lạc hay một nút bị sự cố; nghĩa là hệ thống phải có khả năng tự thích ứng khi có sự cố về mạng.

Một phương pháp nhằm tăng cường tính tự thích ứng của hệ thống là duy trì nhiều bản sao của mỗi mục và các bản sao được giữ ở những vị trí khác nhau. Khi một nút N có chứa một bản sao của A bị sự cố, chúng ta có thể truy xuất các bản sao khác của A ở những nút khác. Cuối cùng khi N được khôi phục thì điều cần thiết là những thay đổi cho A tại những nút khác cùng được thực hiện tại N. Tất nhiên, việc duy trì nhiều bản sao phải trả giá đắt khi khôi phục sự cố.

Một loại sự cố phức tạp hay xảy ra khi các đường liên lạc bị cắt đứt, và vì vậy mạng sẽ bị phân làm nhiều mảnh không thể giao tiếp được với nhau.

Thí dụ 6.25:

Sự cố của nút D trong cây của hình 6.25 tách cây này thành ba phần {A, B, C}, {E} và {F, G, H}. Sự cố trên đường liên lạc (B, D) tách cây thành hai phần {A, B, C} và {D, E, F, G, H}.



Hình 6.25 Mạng cây

Mất liên lạc trong mạng gây nhiều khó khăn cho việc duy trì hoạt động của hệ thống CSDL. Vấn đề là, tất cả các bản sao của một mục có thể ở trong một khối của phần mạng bị tách rời, và những khối khác không thể truy xuất đến mục này được. Một vấn đề khác là trong các khối khác nhau, những thay đổi khác nhau có thể được thực hiện cho cùng một mục, và những thay đổi này cần phải được tích hợp lại khi mạng đã thông suốt trở lại. Vì vậy khi thiết kế mạng, chúng ta cần tạo ra một số đường liên lạc đủ để mạng không bị gián đoạn thường xuyên. Chẳng hạn, nối các nút theo kiểu vòng tròn sẽ giảm bị mất liên lạc trong mạng khi có sự cố xảy ra tại bất kỳ nút nào hay đường liên lạc nào.

6.11.2. MỤC TIN CỤC BỘ VÀ TOÀN CỤC

Về mặt logic, trong CSDLPT có thể xem một mục như được cấu tạo bởi nhiều *mảnh* (frament) và các mảnh được phân bố vào trong các nút. Chẳng hạn:

1. Một mục có thể có nhiều bản sao giống nhau, mỗi bản là một "mảnh". Nhằm bảo đảm tính thích ứng, các mảnh được lưu ở những vị trí khác nhau.

2. Một mục có thể được phân chia cục bộ thành nhiều mảnh riêng biệt. Chẳng hạn nếu coi quan hệ ngân hàng là một mục thì mục đó:

ACCOUNTS (NUMBER, BALANCE) được tách ngang theo địa lý, nghĩa là những bộ chứa thông tin về *tài khoản* (account) ở mỗi chi nhánh được lưu tại chi nhánh đó.

3. Có thể có một tổ hợp của (1) và (2). Chẳng hạn, các bản sao dự phòng của các tài khoản tại tất cả các chi nhánh trong một vùng được lưu tại văn phòng của vùng này, và một bản sao của toàn bộ quan hệ ACCOUNTS được lưu tại trung tâm.

Vì vậy cần nhắc lại trong CSDL phân tán ta phải phân biệt một mục theo nghĩa *toàn cục* (global) hay *cục bộ* (local), đó là những bản sao cụ thể của một mục tồn tại trong CSDL. Trong thực tế, hành động khoá trên một mục toàn cục A là khoá trên các bản sao cục bộ của A, và những hành động khoá cục bộ có thể tách ra, cùng với nhiều hành động cục bộ đan xen. Vì vậy chúng ta phải xem xét cẩn thận về cách dịch một hành động lấy khoá chốt toàn cục thành hành động lấy khoá chốt cục bộ sao cho hành động trao khoá chốt cục bộ xuất hiện như một hành động nguyên tử.

6.11.3. GIAO DỊCH TOÀN CỤC, GIAO DỊCH CỤC BỘ VÀ TÍNH KHẢ TUẦN TỰ

Giao dịch toàn cục có thể được cấu tạo từ nhiều *giao dịch con cục bộ*, mỗi giao dịch thực hiện tại những vị trí khác nhau. Nhiệm vụ của hệ thống CSDL là phải bảo đảm cho các giao dịch toàn cục hoạt

động theo phương thức khả tuần tự. Khái niệm về "tính khả tuần tự" trong CSDL phân tán là sự khái quát hoá tự nhiên từ định nghĩa đã đưa ra trong phần trước. Một lịch biểu cho các giao dịch trong CSDL phân tán là khả tuần tự nếu tác dụng của nó trên các mục toàn cục giống như tác dụng khi các giao dịch được thực hiện tuần tự, nghĩa là đến phiên của mình, mỗi giao dịch sẽ thực hiện tất cả các giao dịch con ở tất cả các vị trí trước khi giao dịch kế tiếp bắt đầu.

Thí dụ 6.26:

Giả sử giao dịch T chuyển 10 đô la từ tài khoản A sang tài khoản B. Tài khoản A có lưu tại N_2, N_3, còn B lưu tại N_1, N_5. Cũng giả sử rằng T được bắt đầu tại nút N_1, và N_5. Thế thì T phải khởi động hai giao dịch con để làm giảm 10 trong các bản sao cục bộ của A tại các nút N_2 và N_3 đồng thời phải khởi động hai giao dịch con khác cộng thêm 10 vào các bản sao vật lý của B tại nút N_1 và N_5. Vì vậy giao dịch toàn cục T cấu tạo bởi các giao dịch con cục bộ tại các nút N_1,..., N_5, và tác dụng của các giao dịch này phải được điều phối và tuần tự hoá. Chẳng hạn, thay đổi cho một bản sao của một mục không được ghi vào CSDL cố định nếu không đảm bảo rằng một thay đổi giống như thế cùng xảy ra trong các bản sao khác. Yêu cầu đó luôn phải duy trì ngay cả nếu N_3 gặp sự cố sau khi A được cập nhật tại N_2 chẳng hạn. Trong trường hợp đó, chúng ta phải bảo đảm rằng A sẽ được cập nhật tại N_3 sau khi nút này hoạt động trở lại.

6.12. KHOÁ CHỐT PHÂN TÁN

Khi chúng ta mở rộng khái niệm điều khiển đồng thời từ trường hợp vị trí đơn sang trường hợp phân tán, nhiệm vụ đầu tiên là phải phân tích xem làm thế nào để xây dựng các khoá trên mục tin toàn cục từ các khoá trên các mục tin cục bộ là xin một khoá trên một bản sao A_i của mục A từ bộ quản lý khoá cục bộ tại vị trí của A. Chẳng hạn nếu chúng ta dùng các khoá đọc và khoá ghi thì chúng ta không thể có hai giao dịch cùng lúc giữ khoá ghi trên cùng một mục, hoặc một khoá đọc và một khoá ghi trên cùng một mục. Tuy nhiên, vẫn có thể có nhiều giao dịch cùng giữ khoá đọc trên cùng một mục đồng thời.

Nếu chỉ có một bản sao của một mục, thì mục toàn cục đồng nhất với bản sao cục bộ này. Vì vậy chúng ta có thể duy trì các khoá trên mục toàn cục này nếu và chỉ nếu chúng ta duy trì các khoá trên các bản sao một cách đúng đắn. Những giao dịch muốn khoá một mục A chỉ có một bản sao sẽ gửi thông báo yêu cầu khoá đến vị trí lưu giữ bản sao này. Bộ quản lý khoá tại vị trí đó sẽ gửi lại thông báo trao khoá hoặc từ chối khoá.

Tuy nhiên nếu có nhiều bản sao thì công việc dịch từ khoá cục bộ sang khoá toàn cục có thể thực hiện theo nhiều cách, mỗi cách đều có những ưu điểm riêng. Chúng ta sẽ đề cập đến một số cách tiếp cận và so sánh số lượng các thông báo cần dùng cho mỗi cách.

6.12.1. KHOÁ - GHI - TẤT - CẢ - KHOÁ- ĐỌC - MỘT

Một cách đơn giản để duy trì các khoá cục bộ là duy trì các khoá trên các bản sao của các mục và yêu cầu các giao dịch tuân theo một nghi thức với những quy tắc xác định các khoá trên các mục cục bộ sau đây:

1. Để nhận được một khoá đọc trên một mục toàn cục A, một giao dịch có thể nhận một khoá đọc trên một bản sao của A (đọc một).

2. Để nhận được một khoá ghi trên mục toàn cục A, một giao dịch phải nhận được các khoá ghi trên tất cả các bản sao của A (ghi tất cả).

Chiến lược này được gọi là *Khoá - ghi - tất - cả-khoá- đọc-một.*

Tại mỗi vị trí, các quy tắc trao hay từ chối khoá trên các bản sao hoàn toàn giống như trước: chúng ta có thể trao một khoá đọc nếu không có giao dịch nào đang có khoá ghi trên bản sao đó, và có thể trao một khoá ghi nếu không có giao dịch nào hiện đang giữ một khoá ghi hoặc một khoá đọc trên bản sao đó.

Tác dụng của những quy tắc này là không thể có hai giao dịch cùng lúc giữ một khóa đọc và một khoá ghi trên cùng một mục toàn cục A. Bởi thế để giữ được khoá ghi trên một mục toàn cục A, giao dịch này phải giữ các khoá ghi trên tất cả các bản sao của A. Tuy nhiên để giữ một khoá đọc trên A, một giao dịch chỉ cần giữ một khoá đọc ít nhất trên một bản sao A_1 nào đó của A. Nhưng các quy tắc khóa trên bản sao cục bộ A_1 không cho một giao dịch giữ một khoá đọc cùng lúc với một giao dịch khác đang giữ khoá ghi trên A_1. Tương tự không thể có hai giao dịch cùng giữ khoá ghi trên A, bởi vì sẽ có các khoá ghi đang tranh chấp trên tất cả bản sao của A.

6.12.2.PHÂN TÍCH KHOÁ GHI TẤT CẢ

Giả sử rằng có n vị trí có bản sao của A. Để thực hiện lệnh WLOCK A, giao dịch phải gửi thông báo yêu cầu khoá đến tất cả n vị trí. Sau đó, n vị trí này sẽ trả lời là giao dịch có nhận được khoá hay không, nếu nó nhận được khoá thì phải gửi đến n vị trí giá trị mới của A. Cuối cùng, một thông báo UNLOCK A sẽ được gửi đi.

Các thông báo chứa giá trị của các mục tin có thể dài hơn những thông báo khóa chốt, vì vậy chúng ta chỉ nên gửi những mục lớn chứ không phải là giá trị mới của mục đầy đủ. Dưới đây, chúng ta sẽ phân biệt vài dạng thông báo:

1. *Các thông báo điều khiển* (control message), là các thông báo liên quan đến các khoá, uỷ thác hoặc huỷ bỏ giao dịch, và những vấn đề điều khiển đồng thời khác.

2. *Các thông báo dữ liệu* (data message), là các thông báo mang giá trị của các mục.

Đôi khi chúng ta cũng có thể gắn thông báo điều khiển vào thông báo dữ liệu, khi đó chỉ cần thông báo dữ liệu.

Như vậy khi một giao dịch khoá ghi mục toàn cục A, chúng ta thấy rằng nó cần gửi 2n thông báo điều khiển và n thông báo dữ liệu. Nếu một trong những bản sao của A nằm tại vị trí đang thực hiện giao dịch này, chúng ta tiết kiệm được hai thông báo điều khiển và một thông báo dữ liệu, mặc dù chúng ta vẫn phải yêu cầu và giữ một khoá tại vị trí này. Nếu một hoặc nhiều vị trí từ chối trao khoá thì khoá trên A không được trao.

Để có được khoá đọc, chúng ta chỉ cần khóa trên một bản sao, vì thế nếu ta biết được vị trí có giữ một bản sao, ta có thể gửi thông báo RLOCK A đến vị trí đó và đợi kết quả đồng ý trao khoá hay từ chối yêu cầu khoá. Nếu khoá được trao, giá trị của A sẽ được gửi cùng với thông báo này. Vì thế trong trường hợp đơn giản nhất, là trường hợp chúng ta biết được vị trí có thể tìm được A và yêu cầu khóa đọc được trao, chỉ có hai thông báo được trao, một thông báo điều khiển (yêu cầu), và một thông báo dữ liệu (trả lời, bao gồm cả giá trị cần đọc). Nếu yêu cầu bị từ chối, có lẽ không nên yêu cầu một khoá ở vị trí khác ngay sau đó bởi vì rất có thể có một giao dịch đang khoá ghi A, vì vậy đã khoá tất cả các bản sao của A.

6.12.3.CHIẾN LƯỢC KHOÁ CHỐT QUÁ BÁN

Một nghi thức khác để định nghĩa các khoá trên các mục toàn cục là *chiến lược khoá chốt quá bán*:

1. Để nhận được một khoá đọc trên một mục toàn cục A, một giao dịch phải nhận được khoá đọc của hơn một nửa các bản sao của A.

2. Để nhận được một khoá ghi trên một mục cục bộ A, một giao dịch phải nhận được khoá ghi của hơn một nửa các bản sao của A.

Chúng ta để ý rằng, nếu hai giao dịch đều giữ các khoá trên A trong chiến lược quá bán (khoá đọc hay ghi đều không quan trọng) thì chúng sẽ giữ các khoá của hơn phần nửa các bản sao, suy ra rằng phải có ít nhất một bản sao bị khoá bởi cả hai giao dịch. Nhưng nếu một trong hai khoá đó là khoá ghi, thì sẽ có tranh chấp khóa ở bản sao đó, điều này không được bộ quản lý khoá tại điểm đó cho phép. Chúng ta kết luận rằng hai giao dịch không thể cùng giữ khoá ghi trên một mục

cục bộ A đồng thời, và cũng không thể có một giao dịch giữ khoá ghi còn giao dịch kia giữ khoá đọc. Dĩ nhiên chúng có thể cùng giữ khoá đọc trên một mục.

6.12.4.SO SÁNH CÁC PHƯƠNG PHÁP

Trước khi bàn tiếp các phương pháp khác, chúng ta hãy so sánh *phương pháp khoá ghi tất cả* và *phương pháp quá bán*. Mỗi phương pháp đều sử dụng n thông báo dữ liệu cho thao tác ghi và một thông báo dữ liệu cho thao tác đọc. Phương pháp *khoá ghi tất cả sử dụng 2n* thông báo điều khiển cho thao tác ghi và một thông báo điều khiển cho thao tác đọc, còn *phương pháp quá bán* sử dụng (n + 1) thông báo cho thao tác ghi n thông báo cho thao tác đọc. Vì thế nếu các giao dịch điển hình yêu cầu số lượng các khoá ghi và đọc bằng nhau, không có phương pháp nào tốt hơn. Ngược lại, nếu phần lớn là khoá đọc, phương pháp *khoá - ghi - tất cả* được ưa chuộng hơn, và nếu khoá ghi chiếm ưu thế, chúng ta sẽ chọn lựa phương pháp quá bán.

Hai phương pháp này khác nhau ở tác dụng đối với khả năng xuất hiện khoá gài. Khi sử dụng cách tiếp cận *khoá ghi tất cả*, hai giao dịch nào đó có thể bắt đầu cùng một lúc và cùng muốn ghi vào mục A, và rất có thể mỗi giao dịch đã có được một khoá trên ít nhất một bản sao của A. Kết quả là có một khoá gài. Nó phải được hệ thống giải quyết bằng một trong những phương pháp khá tốn kém. Nếu so sánh với cách tiếp cận quá bán, một trong hai giao dịch đang tranh chấp được khoá trên mục đó, và giao dịch kia phải đợi hoặc bị huỷ bỏ.

6.12.5. CHIẾN LƯỢC TỔNG QUÁT

Hai phương pháp trên thực ra là hai thái cực của một *phổ các chiến lược* (spectrum of strategies). Đó là các chiến lược "*k trong n*", với n/2<k≤ n, được định nghĩa như sau:

1. Để nhận được khoá ghi trên một mục cục bộ A, giao dịch phải nhận được các khoá trên k bản sao của A.

2. Để nhận được khoá đọc trên một mục cục bộ A, giao dịch phải nhận được các khoá trên n - k + 1 bản sao của A.

Để chứng tỏ rằng phương pháp này định nghĩa chính xác các khoá, chúng ta cần nhận xét là nếu một giao dịch đã giữ một khoá đọc trên mục toàn cục A, nó sẽ giữ khoá đọc trên n - k + 1 bản sao của A, trong khi đó nếu một giao dịch khác đồng thời giữ một khoá ghi trên A, nó sẽ giữ khoá ghi trên k bản sao của A. Do đó, không thể xảy ra trường hợp một bản sao nào đó vừa bị khoá đọc vừa bị khoá ghi bởi các giao dịch khác nhau cùng lúc. Tương tự, nếu hai giao dịch đồng thời giữ khoá ghi trên một mục toàn cục A, thì mỗi giao dịch phải giữ các khoá trên k bản sao của A. Vì k > n/2, nên một bản sao nào đó phải bị khoá ghi bởi cả hai giao dịch cùng lúc, điều này cũng không thể xảy ra.

Với chiến lược được gọi là "khoá ghi tất cả" chính là chiến lược "n trong n", còn chiến lược quá bán là chiến lược "(n + 1)/2 trong n". Khi k tăng lên, chiến lược sẽ thực hiện tốt hơn trong những trường hợp thực hiện thao tác đọc nhiều hơn. Xác xuất hai giao dịch tranh chấp khoá ghi trên cùng một mục gây ra *khoá gài* cũng tăng lên khi k tăng, bởi vì mỗi giao dịch nhận được một số khoá đủ để phong tỏa giao dịch kia.

6.12.6.NGHI THỨC BẢN CHÍNH

Một quan điểm khác có liên quan đến việc quản lý khoá là trao trách nhiệm khoá một mục toàn cục A cho một vị trí đặc biệt, bất kể số lượng bản sao là bao nhiêu. Nghĩa là, một nút trong mạng sẽ được trao nhiệm vụ quản lý các khoá của tất cả các mục; cách tiếp cận này gọi là *phương pháp nút trung tâm (central node method).* Tuy nhiên, dưới dạng tổng quát nhất, trách nhiệm khóa một mục A có thể được trao cho một nút bất kỳ, và các nút khác nhau có thể được sử dụng cho các mục khác nhau.

Chẳng hạn, một chiến lược khá hợp lý là xác định một *vị trí chính (primary site)* cho mỗi mục. Giả dụ xét CSDL của một ngân hàng, và các nút là các chi nhánh, một điều sẽ hết sức tự nhiên khi chọn vị trí chính cho một mục biểu thị tài khoản là chi nhánh giữ tài khoản đó. Trong trường hợp này, vì phần lớn các giao dịch truy xuất trong ngân hàng là tài khoản nên sẽ khởi hoạt tại vị trí chính của nó, ta sẽ có các khoá thường dùng mà không cần phải gửi đi một thông báo nào.

Nếu một giao dịch không ở tại vị trí chính của A muốn khoá A, nó gửi một thông báo đến vị trí chính của A và vị trí này sẽ trả lời, hoặc trao khoá hoặc từ chối khoá. Vì vậy khoá một mục A cũng giống như khoá một bản sao của A tại vị trí chính. Thật ra cũng không cần có một bản sao của A tại vị trí chính, chỉ cần có một bộ quản lý khoá xử lý các khoá trên A.

6.12.7. THẺ BẢN CHÍNH

Có một chiến lược tổng quát hơn so với phương pháp thiết lập một vị trí chính cho một mục. Chúng ta giả sử tồn tại các thẻ đọc (read – token- đó là những quyền ưu tiên mà các nút trong mạng có thể được nhận), làm đại diện cho các giao dịch để truy xuất các mục. Đối với một mục A, chỉ có thể tồn tại một thẻ ghi duy nhất cho A. Nếu không có thẻ ghi thì có thể có một số thẻ đọc cho A. Nếu một vị trí có thẻ ghi của A thì nó có thể trao một khoá đọc hoặc một khoá ghi trên A cho một giao dịch đang thực hiện tại vị trí đó. Một vị trí chỉ có thẻ đọc của A có thể trao một khoá đọc trên A cho giao dịch tại vị trí đó nhưng không thể trao khoá ghi. Cách tiếp cận này được gọi là *phương pháp thẻ bản chính (Primary copy token method).*

Nếu một giao dịch tại một vị trí N muốn khoá ghi A, nó phải thu xếp để thẻ ghi của A chuyển sang vị trí của nó. Nếu thẻ ghi của A đã ở sẵn tại vị trí đó, nó không cần phải thực hiện nữa. Nếu không, chuỗi thông báo sau sẽ được gửi đi:

1. N gửi thông báo đến tất cả vị trí để yêu cầu thẻ ghi.

2. Mỗi vị trí M nhận được thông báo sẽ trả lời, cụ thể:

a) M không có thẻ đọc lẫn thẻ ghi của A, hoặc nó có nhưng đang muốn trao thẻ, do đó N có thể nhận được thẻ ghi.

b) M có thẻ đọc hoặc thẻ ghi của A nhưng không trao (bởi vì có một giao dịch đang sử dụng thẻ này, hoặc M đã dành thẻ đó cho một vị trí khác).

Trong trường hợp (a), M phải ghi nhớ rằng N đã yêu cầu nó trao thẻ, nhưng không biết N có thể có được thẻ này hay chưa (vì một vị trí

khác có thể trả lời (b)). M "dự trữ" thẻ này cho N; nó làm như thế để tránh trường hợp một vị trí P khác có thể nhận được câu trả lời của M rằng M không phản đối việc P nhận thẻ ghi.

3. Nếu tất cả vị trí đều trả lời (a), N biết rằng nó có thể nhận được thẻ ghi. Nó gửi một thông báo đến mỗi vị trí đã trả lời (a), báo cho nó biết rằng N đã nhận được thẻ ghi, và chúng phải huỷ các thẻ cho A mà chúng có. Nếu một vị trí nào đó trả lời (b) thì N không thể nhận được thẻ ghi, và nó phải gửi thông báo đến các vị trí trả lời (a) để báo cho chúng biết rằng chúng có thể ngưng dự trữ thẻ ghi của A cho N và có thể cho phép một vị trí khác nhận thẻ đó.

Để đọc A, một quá trình tương tự cũng xảy ra, ngoại trừ nếu vị trí cục bộ có một thẻ đọc A thì không cần phải gửi thông báo nữa. Trong (2), vị trí M sẽ không phản đối (gửi thông báo (b)) nếu nó có thẻ đọc của A, hoặc chỉ nếu nó có một thẻ ghi. Trong (3), nếu N được phép nhận thẻ đọc của A thì chỉ những thẻ ghi, không phải là các thẻ đọc, phải bị huỷ tại những vị trí khác.

6.12.8. SO SÁNH TIẾP CÁC PHƯƠNG PHÁP

Rõ ràng, phương pháp thẻ bản chính dùng nhiều thông báo hơn so với những phương pháp khác; cả hai thao tác đọc và ghi đều có thể dùng đến 3m thông báo điều khiển, trong đó m là số nút trong mạng. Những phương pháp khác sử dụng số lượng thông báo tỷ lệ với số bản sao của một mục trong trường hợp xấu nhất. Ngược lại, phương pháp thẻ bản chính trung bình dùng ít hơn 3m thông báo điều khiển cho mỗi thao tác khoá khi vị trí thực hiện phần lớn các giao dịch có tham chiếu đến một mục đặc biệt. Khi đó thẻ ghi cho mục đó có xu hướng nằm tại vị trí

đó, vì thế nhiều thông báo điều khiển trở nên không cần thiết cho phần lớn các giao dịch. Như vậy, không thể so sánh trực tiếp với những phương pháp "k trong n". Phương pháp nào được ưa chuộng hơn tuỳ thuộc vào sự phân bố vị trí của các giao dịch cần khóa một mục nào đó.

Tương tự, chúng ta không thể so sánh trực tiếp phương pháp vị trí chính với phương pháp khoá ghi tất cả; mặc dù tính trung bình, phương pháp vị trí chính sử dụng số thông báo ít hơn, còn phương pháp sau lại tiện lợi khi các khoá phần lớn là khóa đọc trên những bản sao ít hơn, còn phương pháp sau lại tiện lợi khi các khoá phần lớn là khoá đọc trên những bản sao không nằm tại vị trí chính của mục đó. Dường như là cách tiếp cận vị trí chính có hiệu quả hơn so với phương pháp "k trong n" khi $k > 1$. Tuy nhiên những kết luận trên thực sự chưa chính xác lắm. Chẳng hạn, phương pháp vị trí chính dễ bị sai lệch do có sự cố tại một vị trí chính của một mục, vì các vị trí khác phải phát hiện sự cố và gửi các thông báo nhằm thoả thuận một vị trí mới.

Chúng ta cũng có thể so sánh phương pháp thẻ bản chính với phương pháp vị trí chính. Ở phương pháp sau, một thao tác ghi cần hai thông báo điều khiển, một để yêu cầu và một để nhận khoá từ vị trí chính, và sau đó như thường lệ là n thông báo dữ liệu để ghi một giá trị mới. Thao tác đọc cần một thông báo điều khiển yêu cầu khoá và một thông báo dữ liệu để trả lời, vừa trao yêu cầu vừa gửi giá trị. Nếu tất cả các giao dịch tham chiếu đến A đều thực hiện tại vị trí chính của A thì hai cách tiếp cận hoàn toàn như nhau; không cần gửi thông báo nào, trừ khi thao tác ghi bị bắt buộc phải cập nhật các bản sao khác của A nếu có. Khi những vị trí khác tham chiếu đến A, phương pháp vị trí chính tiết kiệm một số lượng thông báo đáng kể.

Tuy nhiên, phương pháp thẻ áp dụng dễ dàng hơn cho những thay đổi tạm thời trong quá trình hoạt động. Chẳng hạn trong một CSDL ngân hàng tưởng tượng, giả sử một khách hàng đi nghỉ phép và bắt đầu sử dụng một chi nhánh khác với chi nhánh họ vẫn thường dùng theo phương pháp vị trí chính, mỗi giao dịch tại chi nhánh mới sẽ đòi hỏi phải trao đổi những thông báo khoá. Trong khi đó theo cách tiếp cận thẻ, sau khi giao dịch đầu tiên thực hiện tại chi nhánh mới, thẻ ghi cho tài khoản này sẽ lưu tại chi nhánh đó, miễn là khách hàng này vẫn còn đang nghỉ phép.

6.12.9. PHƯƠNG PHÁP NÚT TRUNG TÂM

Chúng ta xét cách tiếp cận cuối cùng đối với vấn đề khoá chốt, trong đó một nút đặc biệt của mạng sẽ chịu trách nhiệm về tất cả các khoá. Phương pháp này rất giống với phương pháp vị trí chính: khác biệt duy nhất là vị trí chính cho một mục, chính là một nút trung tâm, có thể không là vị trí chứa một bản sao của mục đó. Vì vậy, khoá đọc phải được lấy qua những bước sau:

1. Yêu cầu một khoá đọc từ nút trung tâm

2. Nếu không trao khoá, nút trung tâm gửi một thông báo từ chối đến vị trí đang yêu cầu. Nếu trao khoá, nút trung tâm sẽ gửi một thông báo đến vị trí đang giữ bản sao của mục đó.

3. Vị trí có bản sao gửi một thông báo kèm với giá trị đến vị trí đang yêu cầu.

Vì vậy, phương pháp nút trung tâm thường cần thêm một thông báo điều khiển gửi đến một vị trí có giá trị đang được yêu cầu. Tương tự, khi cần ghi, vị trí đang thực hiện giao dịch thường phải gửi thêm một

thông báo đến nút trung tâm yêu cầu nút này giải phóng khoá. Trong phương pháp vị trí chính, thông báo này sẽ được gộp vào các thông báo uỷ thác giao dịch.

Phương pháp nút trung tâm có một khuyết điểm lớn là lưu lượng thông báo được đổ về hoặc đi khỏi một nút rất lớn và có thể làm quá tải (tắc nghẽn) đường truyền. Ngoài ra, phương pháp này dễ bị ảnh hưởng khi có sự cố tại nút trung tâm.

6.13. KHOÁ CHỐT HAI PHA PHÂN TÁN

Trong phần trước chúng ta đã thấy rằng có thể định nghĩa các khoá trên các mục toàn cục theo nhiều cách khác nhau. Bây giờ chúng ta phải đề cập đến cách sử dụng khoá chốt để bảo đảm được tính khả tuần trự cho các giao dịch có chứa nhiều giao dịch con, mỗi giao dịch con thực hiện ở một vị trí khác nhau. Cần để ý rằng một lịch biểu trong môi trường phân tán là một chuỗi sự kiện xảy ra tại những vị trí khác nhau. Mặc dù các vị trí có thể có nhiều hoạt động đồng thời, chúng ta sẽ phân nhỏ chúng ra và giả sử rằng có một thứ tự tuyến tính cho các sự kiện này theo một đồng hồ hệ thống. Một lịch biểu *là tuần tự* nếu nó thực hiện theo một thứ tự nào đó của các giao dịch (bao gồm tất cả mọi hành động của một giao dịch, theo sau là tất cả các hành động của một giao dịch khác, v.v.). Một lịch biểu được gọi *là khả tuần tự* nếu nó tương đương với một lịch biểu tuần tự khi xét về tác dụng của nó trên CSDL.

Với mối liên hệ mật thiết giữa tính khả tuần tự và khoá chốt hai pha trong phần trước, chúng ta thử xét các phương thức tổng quát khoá hai pha cho môi trường phân tán. Chúng ta có thể cho rằng tại một nút,

các giao dịch con phải tuân theo nghi thức hai pha. Tuy nhiên điều đó chưa đủ.

Thí dụ 6.27:

Giả sử giao dịch toàn cục T_1 có hai giao dịch con:

1. $T_{1.1}$ thực hiện tại vị trí S_1 và ghi một giá trị mới cho bản sao A_1 của mục toàn cục A.

2. $T_{1.2}$ thực hiện tại vị trí S_2 và cũng ghi giá trị đó cho bản sao A_2 của A.

Đồng thời, giao dịch T_2 có hai giao dịch con $T_{2.1}$ thực hiện tại S_1 và ghi một giá trị mới của A_1, $T_{2.2}$ thực hiện tại S_2 và ghi cùng giá trị vào A_2. Giả sử rằng nghi thức *khoá ghi tất cả* được sử dụng ở đây để định nghĩa khoá trên các mục toàn cục. Trong hình 6.26 chúng ta thấy một lịch biểu cho các hành động tại hai vị trí. Các cặp sự kiện trên mỗi dòng có thể xảy ra đồng thời, hoặc chúng ta có thể giả sử chúng xảy ra theo một thứ tự nào đó. Rõ ràng, tình huống tại S_1 cho thấy $T_{1.1}$ phải đi trước $T_{2.1}$ theo thứ tự tuần tự. Ở S_2, $T_{2.2}$ phải đi trước $T_{1.2}$. Rất tiếc là, một thứ tự tuần tự phải được tạo thành không chỉ từ những giao dịch con mà còn phải từ những giao dịch toàn cục. Vì thế chúng ta cho T_1 đi trước T_2 thì $T_{1.2}$ đi trước $T_{2.2}$, vi phạm thứ tự cục bộ tại S_2. Tương tự, nếu thứ tự là T_2 đi trước T_1 thì thứ tự cục bộ tại S_1 bị vi phạm. Sự thật, với thứ tự các sự kiện trong hình 6.26 hai bản sao của A nhận được các giá trị cuối cùng khác nhau, điều đó lập tức khẳng định rằng không tồn tại một lịch biểu tuần tự tương đương.

$T_{1.1}$	$T_{2.1}$	$T_{1.2}$	$T_{2.2}$
WLOCK A_1			WLOCK A_2
UNLOCKA_1			UNLOCKA_2
	WLOCK A_1	WLOCK A_2	
	UNLOCK A_1	UNLOCK A_2	
Tại S_1		Tại S_2	

Hình 6.26 Các giao dịch với khoá chốt hai pha tại mỗi nút

Vấn đề minh hoạ trong Thí dụ 6.27 chính là, để cho các giao dịch trong môi trường phân tán hoạt động được theo phương pháp khoá hai pha, chúng ta không những phải xét đến các lịch biểu cục bộ mà còn phải xét đến lịch biểu toàn cục của các hành động, và lịch biểu đó phải được khoá hai pha. Kết quả là một giao dịch con của T không thể giải phóng khoá nếu sau đó có một giao dịch con của T tại một vị trí khác yêu cầu khoá. Chẳng hạn $T_{1.1}$ của hình 6.26 vi phạm nguyên tắc này khi mở khoá A_1 trước khi $T_{1.2}$ nhận được khoá trên A_2.

Vì vậy, mỗi giao dịch con của một giao dịch đã cho phải thông tin cho những giao dịch con khác biết rằng nó đã yêu cầu tất cả khoá cần thiết. Chỉ sau khi tất cả các giao dịch con đã đạt tới điểm khoá (lock point) của riêng nó, toàn bộ giao dịch T mới đạt tới điểm khoá, sau đó các giao dịch con mới có thể giải phóng các khoá. Khi tất cả các giao dịch con đạt tới điểm khoá là một thí dụ về *vấn đề thoả thuận phân tán (distributed agreement problem)*. Rõ ràng là, thoả thuận phân tán, đặc biệt trong trường hợp có sự cố mạng, rất phức tạp và tốn kém. Do đó, gửi thông báo điều khiển để khẳng định rằng các giao dịch con đã đạt tới điểm khoá thường không phải là một hành động tối ưu.

Đúng ra, có nhiều lý do để buộc những giao dịch trong môi trường phân tán phải là nghiêm ngặt, nghĩa là chúng chỉ mở khoá sau khi đạt tới điểm uỷ thác. Nếu các giao dịch tuân theo nghi thức nghiêm ngặt thì chúng ta có thể sử dụng điểm uỷ thác làm điểm khoá. Các giao dịch con đồng ý uỷ thác bởi một quá trình được mô tả trong phần sau, và chỉ sau khi uỷ thác, các khoá mới được giải phóng.

Trong tình huống như ở hình 6.26 $T_{1.1}$ và $T_{2.2}$ sẽ không giải phóng các khoá ở dòng thứ hai nếu tuân theo nghi thức nghiêm ngặt. Trường hợp này sẽ có một *khoá gài* (deadlock) giữa T_1 và T_2 bởi vì mỗi giao dịch này đều có một giao dịch con đang đợi một khoá được giữ bởi một giao dịch kia. Chúng ta sẽ thảo luận về phương pháp phát hiện khoá gài phân tán trong phần sau.

6.14. UỶ THÁC PHÂN TÁN

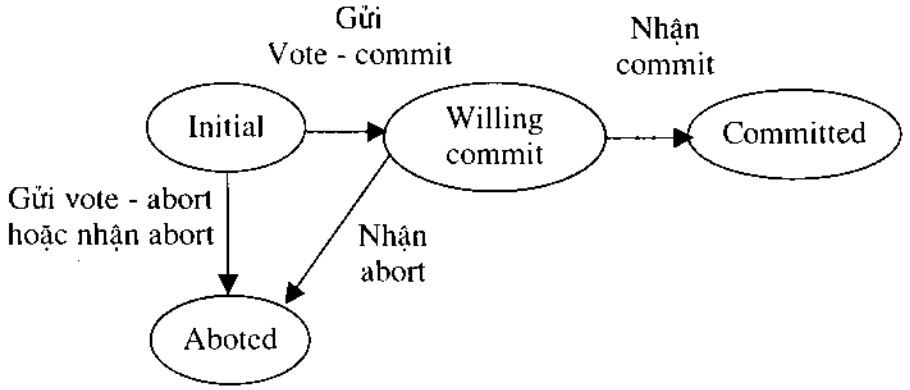
Giả sử chúng ta có một giao dịch T khởi hoạt tại một vị trí vá sinh ra những giao dịch con tại nhiều vị trí khác nhau. Coi thành phần của T thực hiện tại vị trí gốc của nó là một giao dịch con của giao dịch toàn cục T; vì thế T đơn thuần chỉ chứa các giao dịch con, mỗi giao dịch thực hiện tại một vị trí khác nhau. Chúng ta sẽ gọi giao dịch con thực hiện ở vị trí gốc là *điều phối viên* (coordinator), còn các giao dịch con khác là *thành viên* (participant). Sự phân biệt này rất cần thiết khi mô tả quá trình uỷ thác phân tán.

Khi không có sự cố, uỷ thác phân tán rất đơn giản. Mỗi giao dịch con T_i của giao dịch toàn cục T quyết định sẽ uỷ thác (commit) hay huỷ bỏ (abort). Cần nhớ lại rằng, T_i có thể bị huỷ bỏ bởi rất nhiều lý do, như bị kẹt trong một khoá gài hoặc truy xuất CSDL bất hợp lệ. Khi T_i quyết

định gửi thông báo vote - commit (biểu quyết uỷ thác) hoặc thông báo vote - abort (biểu quyết hủy bỏ) đến điều phối viên. Nếu thông báo vote - abort được gửi đi, T_i biết chắc rằng giao dịch T phải huỷ bỏ, vì thế nó có thể chấm dứt. Tuy nhiên, nếu T_i gửi thông báo vote - commit đi nó không biết được cuối cùng T sẽ uỷ thác hay huỷ bỏ vì có thể một giao dịch con khác đã quyết định huỷ bỏ.

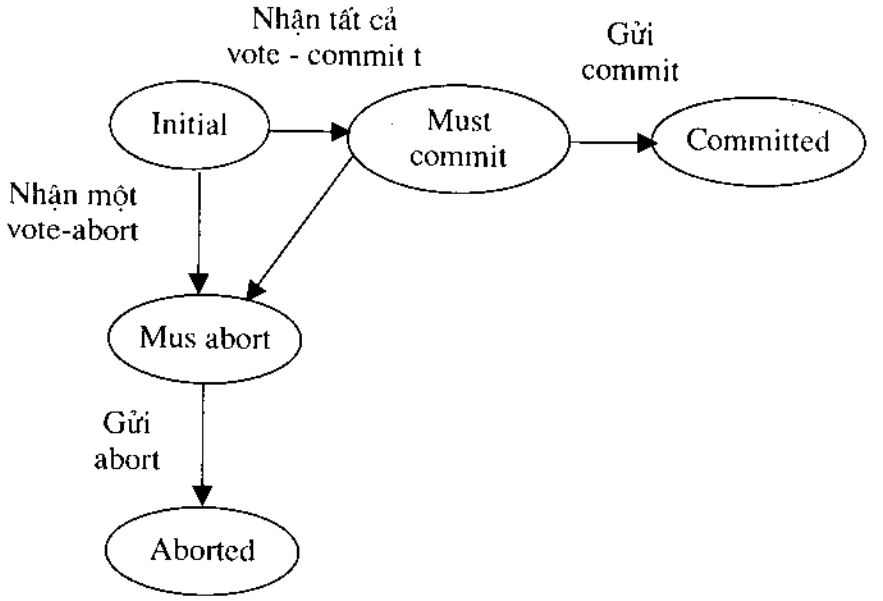
Như vậy, sau khi gửi biểu quyết uỷ thác, T_i phải đợi một thông báo từ điều phối viên. Nếu điều phối viên nhận được một thông báo vote - abort từ một giao dịch con khác, nó sẽ gửi thông báo vote - abort đến tất cả các giao dịch con, và tất cả chúng đều huỷ bỏ, vì thế giao dịch T cũng phải huỷ bỏ. Nếu điều phối viên nhận được thông báo vote - commit từ tất cả các giao dịch con (bao gồm cả chính nó) thì nó biết rằng T có thể uỷ thác. Khi đó điều phối viên sẽ gửi thông báo commit đến tất cả các giao dịch con. Bây giờ, tất cả giao dịch con đều biết rằng T có thể uỷ thác, vì thế chúng sẽ thực hiện các bước cần thiết để uỷ thác tại vị trí của chúng, chẳng hạn ghi vào nhật ký và giải phóng khoá.

Cũng có thể hình dung rằng các giao dịch con thay đổi trạng thái để đáp lại những thông tin chúng nhận được về giao dịch toàn cục.



Hình 6.27 (a) : Thành viên

Bạn đọc cần lưu ý rằng, hình 6.27 (a), (b) chỉ diễn đạt các hoạt động trạng thái của các giao dịch con. Tất nhiên các giao dịch con có ảnh hưởng đến các giao dịch toàn cục và cũng có thể hình dung rằng các giao dịch con thay đổi trạng thái để đáp lại những thông tin chúng nhận được.



Hình 6.27 (b) Điều phối viên

Hình 6.27 Các trạng thái giao dịch cho uỷ thác phân tán

1. Chúng ta cần phân biệt các thông báo biểu quyết (voting message) do các giao dịch thành viên gửi đến điều phối viên và thông báo quyết định (decision message) do điều phối viên gửi đến các thành viên.

2. Điều phối viên cũng là một thành viên, về nguyên tắc có thể gửi thông báo cho chính nó, mặc dù chúng ta không phải trả chi phí cho những thông báo này trong mạng. Chẳng hạn, điều phối viên có thể quyết định huỷ bỏ bởi vì nó thực hiện một phép chia cho zero, như vậy trong hình 6.27(b), điều phối viên sẽ nhận được thông báo vote - abort từ chính nó.

3. Chỉ đến được trạng thái Committed (đã được uỷ thác) và trạng thái Aborted (đã huỷ bỏ) sau khi các giao dịch con đã thực hiện xong các bước cần thiết, chẳng hạn giải phóng khoá và ghi vào nhật ký.

4. Khi một thành viên đang ở trạng thái khởi đầu (Initial), và quyết định gửi thông báo vote - abort hoặc vote - commit, rồi chuyển qua trạng hái tương ứng là Aborted hoặc Willing - to - commit (mong muốn uỷ thác). Quyết định này dựa vào tình hình hiện tại của chính thành viên đó; chẳng hạn, nó "quyết định" huỷ bỏ nếu hệ thống thông báo rằng nó đang bị kẹt trong một khoá gài và được yêu cầu phải huỷ bỏ.

5. Cũng có thể một thành viên chuyển qua trạng thái Aborted do yêu cầu của điều phối viên. Điều này có thể xảy ra nếu có một thành viên khác đã quyết định huỷ bỏ và thông báo cho điều phối viên, và điều phối viên sẽ chuyển thông báo này đến tất cả thành viên.

6. Sử dụng một điều phối viên không phải là vấn đề quan trọng. Tất cả mọi thành viên đều có thể gửi biểu quyết của nó đến các thành viên khác. Tuy nhiên, số lượng thông báo sẽ tỷ lệ với bình phương số thành viên chứ không phải tỷ lệ tuyến tính theo số lượng này. Các thuật toán như thế sẽ được thảo luận trong phần bài tập.

6.14.1. PHONG TOẢ CÁC GIAO DỊCH

Khi mạng có sự cố, nghi thức uỷ thác phân tán đơn giản như hình 6.27 có thể dẫn đến tình trạng *phong tỏa* (blocking), là tình trạng một giao dịch con nằm tại một nút tuy không có sự cố nhưng không thể uỷ thác hoặc huỷ bỏ cho đến lúc điều chỉnh được sự cố. Bởi vì một vị trí có thể bị ngưng trệ vô hạn định và bởi vì giao dịch con bị phong toả có thể đang giữ các khoá nhưng lại không thể giải phóng chúng nên chúng ta sẽ gặp phải một tình huống khó khăn. Có nhiều bối cảnh gây ra tình trạng này, thí dụ:

Thí dụ 6.28:

Giả sử một giao dịch con T_i giữ một khoá trên một bản sao của mục A, và T_i đạt tới điểm uỷ thác. Nghĩa là T_i gửi thông báo vote - commit đến điều phối viên và chuyển qua trạng thái sẵn sàng uỷ thác (Willing - to – commit) như trong hình 6.27 (a). Sau một thời gian dài, T_i không nhận được thông báo commit hay abort của điều phối viên. Như thế T_i phải ở lại trong trạng thái này và giữ khoá trên bản sao cục bộ của A; nghĩa là T_i bị phong toả. Bất kỳ một hành động nào khác đều có thể gây lỗi.

1. Nếu T_i quyết định uỷ thác mà không có hướng dẫn của điều phối viên, rất có thể một giao dịch con khác đã quyết định huỷ bỏ, nhưng vì điều phối viên gặp sự cố nên không thể thông báo cho T_i biết được. Nếu T_i uỷ thác, một giao dịch khác có thể đọc bản sao này của A, là bản sao không được phép thay đổi; khi đó nó chính là dữ liệu rác.

2. Nếu T_i quyết định huỷ bỏ không có hướng dẫn của điều phối viên, rất có thể điều phối viên đã nhận được thông báo vote - commit từ tất cả các thành viên khác nhưng sau đó mạng bị treo, làm mất liên lạc giữa T_i và điều phối viên. Tuy nhiên, một số thành viên khác không bị mất liên lạc với điều phối viên nên đã nhận được thông báo commit và ghi giá trị mới vào các bản sao của chúng. Vì vậy những bản sao của A không còn giữ các giá trị giống nhau nữa.

6.14.2. UỶ THÁC HAI PHA

Một biến thể của thuật toán ở hình 6.27 và là cách tiếp cận thông dụng nhất đối với vấn đề uỷ thác phân tán. Nghi thức này được gọi là *uỷ thác hai pha* (two-phase commit) bởi vì có hai pha, *pha biểu quyết* (voting phase) và *pha quyết định* (decision phase). Uỷ thác hai pha không tránh được tất cả các phong toả, nhưng nó làm giảm đáng kể những tình huống phong toả. Nghi thức hai pha đưa ra hai cải tiến cho nghi thức đơn giản. Trước tiên, các giao dịch con ghi nhận thời gian đã trôi qua tính từ lúc đáng lẽ phải nhận được thông báo trả lời, và nếu thông báo này quá chậm, rất có thể mạng gặp sự cố, và giao dịch con sẽ *đánh dấu quá hạn* (time out), và chuyển qua một trạng khác để khôi phục. Vấn đề nghiêm trọng nhất như chúng ta đã thấy trong thí dụ 6.30 xảy ra khi một thành viên đang ở trạng thái Willing - to - commit, và quá hạn (timeout) xảy ra, nghĩa là thời gian tính từ khi nó gửi thông báo vote - commit đã vượt quá giới hạn cho phép. Nhằm tránh phong toả, giao dịch con này sẽ gửi thông báo help - me (hãy giúp tôi) đến tất cả các thành viên.

Khi nhận được thông báo help - me:

1. Một thành viên trong trạng thái Committed sẽ trả lời commit. Nó có thể thực hiện điều đó một cách an toàn bởi vì nó đã nhận được thông báo commit của điều phối viên, vì thế biết rằng tất cả các thành viên khác đều đã biểu quyết uỷ thác.

2. Một thành viên đang trong trạng thái Aborted có thể gửi thông báo abort bởi vì nó biết rằng giao dịch T phải huỷ bỏ.

3. Một thành viên chưa biểu quyết (nghĩa là đang trong trạng thái Initial) có thể giải quyết vấn đề này bằng cách quyết định huỷ bỏ, vì thế nó cũng trả lời abort và gửi thông báo vote - abort đến điều phối viên.

4. Một thành viên ở trạng thái Willing - to - commit không có khả năng giải quyết vấn đề này, vì thế nó không trả lời.

Một giao dịch đang bị phong toả, sau khi nhận được thông báo trả lời abort hoặc commit nếu thực hiện đúng theo hướng dẫn này sẽ chuyển một trạng thái phù hợp. Điều đó được khẳng định trong định lý sau.

Định lý 6.7:

Một thành viên T_i gửi thông báo help - me

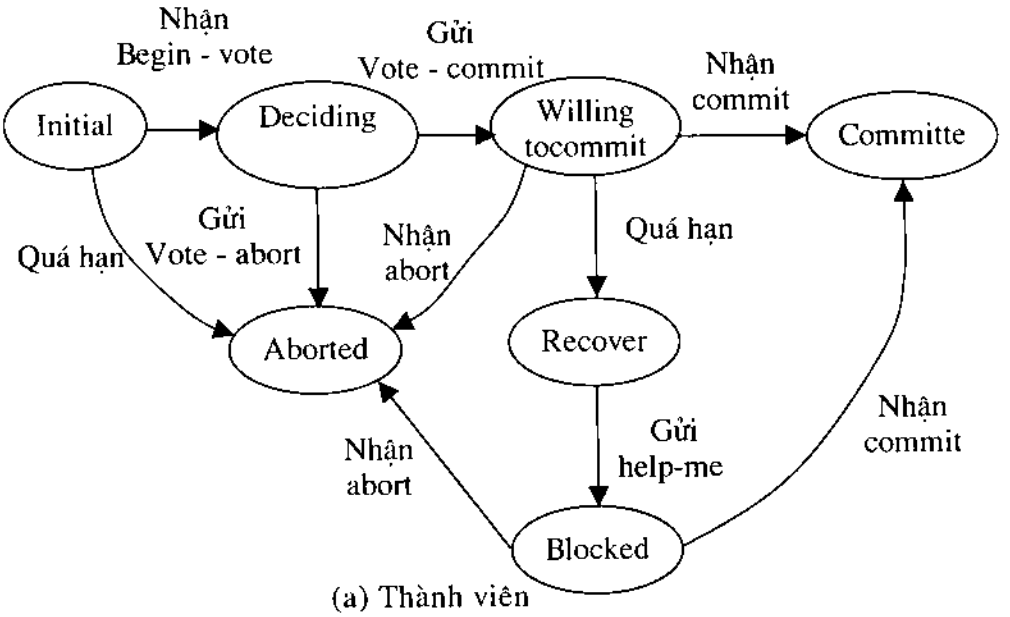
a) Không thể nhận được cả hai thông báo trả lời commit và abort từ các thành viên khác nhau.

b) Không thể nhận được trả lời commit từ một thành viên nếu cuối cùng điều phối viên có thể gửi thông báo abort.

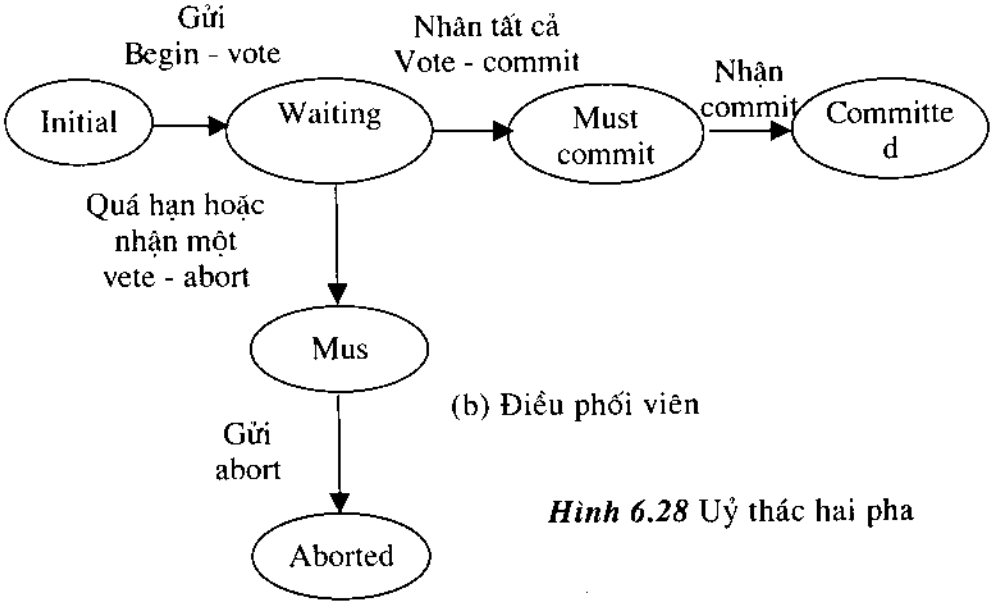
c) Không thể nhận được trả lời abort từ một thành viên nếu cuối cùng điều phối viên có thể gửi thông báo commit.

Chứng minh: T_i nhận được commit chỉ nếu một thành viên khác đã nhận được commit, điều này có nghĩa là điều phối viên đã gửi commit. Vì vậy (b) đúng. T_i có thể nhận được abort trong trường hợp (2) và (3). Trong trường hợp (2), thành viên gửi thông báo abort.

a) Hoặc đã nhận abort từ điều phối viên. b) Hoặc đã quyết định huỷ bỏ.



(a) Thành viên



(b) Điều phối viên

Hình 6.28 Uỷ thác hai pha

Trường hợp (a), điều phối viên đã gửi đi thông báo abort; trường hợp (b) điều phối viên sẽ nhận hoặc đã nhận thông báo vote - abort, hoặc nó sẽ gặp sự cố, vì thế không bao giờ gửi đi thông báo commit. Trong trường hợp (3), thành viên gửi câu trả lời chưa biểu quyết, vì thế sẽ không gửi cả commit lẫn abort. Tất nhiên bây giờ thành viên đó gửi vote - abort nên điều phối viên không bao giờ gửi commit. Vì vậy (c) đúng.

Đối với (a), chúng ta đã nhận xét rằng T_i nhận commit từ một thành viên chỉ khi nếu điều phối viên đã gửi commit. Vì thế các trường hợp (2) và (3) không thể có. Vậy T_i không thể nhận được abort từ một thành viên khác.

Hình 6.28 trình bày quá trình dịch chuyển trạng thái của nghi thức uỷ thác hai pha, cho cả các thành viên và điều phối viên. Chú ý rằng khi một thành viên đánh dấu quá hạn trong trạng thái Willing - to - commit, nó chuyển sang trạng thái Recover (khôi phục), từ đó nó đưa thông báo help - me. Sau đó chuyển sang trạng thái Blocked (bị phong toả), và nó chờ đợi thông báo commit hay abort từ một thành viên khác. Nếu không nhận được thông báo nào do tất cả những thành viên khác bị mất liên lạc với nó, hoặc gặp sự cố, hoặc cùng đang trong trạng thái Willing - to – commit thì thành viên này vẫn phải ở tình trạng bị phong toả.

Trong hình 6.28(b), điều phối viên sẽ đánh dấu quá hạn nếu sau khi gửi begin - vote, nó không thấy một hay nhiều thành viên gửi biểu quyết sau khoảng thời gian đã ấn định. Khi đó, điều phối viên quyết định huỷ bỏ và gửi thông báo abort đến tất cả các thành viên còn lien lạc được. Tất nhiên, các thành viên bị mất liên lạc với điều phối viên vào lúc này sẽ không nhận được thông báo; chúng vẫn bị phong toả nếu

chúng đã nhận được thông báo begin - vote và không thể khôi phục thành công khi chúng quá hạn.

Vị trí cuối cùng mà một quá hạn có thể xảy ra nằm trong hình 6.28 (a). Ở đó một thành viên hoàn tất công việc đã lâu, và trong thời gian đó, nó không được yêu cầu biểu quyết. Rất có thể điều phối viên đã gặp sự cố hoặc bị mất liên lạc với thành viên này. Chính thành viên này sẽ quyết định huỷ bỏ, vì thế nó có thể giải phóng các khoá. Trong hình 6.28(a) không trình bày vấn đề là nếu sau đó thành viên này nhận được thông báo begin - vote từ điều phối viên, thì nó chỉ cần biểu quyết abort. Một số điểm bổ sung về các giao dịch của hình 6.28 như sau:

1. Một giao dịch có thể đã chuyển đến trạng thái Abort hoặc Committed và vẫn được yêu cầu giúp đỡ qua thông báo help - me. Ở đây không có gì sai vì với giả định rằng một giao dịch bất hoạt sẽ trả lời những thông báo này. Trong thực tế, hệ thống sẽ dựa vào nhật ký để trả lời; tất cả mọi thông báo và thay đổi trạng thái đều do hệ thống quản lý chứ không phải được cài vào trong các giao dịch.

2. Ở trạng thái bị phong toả, giao dịch cần lập lại thông báo help - me sau mỗi khoảng thời gian nào đó, với hy vọng rằng có thể một nút đã bị sự cố hoặc mất liên lạc nhận ra thông báo giúp đỡ này. Trong nhiều hệ thống, một nút sau khi khôi phục lại từ một sự cố sẽ thông báo sự hiện diện của nó bằng một cách nào đó, bởi vì nó cần biết điều gì đã xảy ra đối với những giao dịch có liên quan đến nó và những mục chúng đã thay đổi. Vì vậy một giao dịch con bị phong toả có thể gửi lại thông báo help-me khi một nút có một giao dịch con thành viên bắt lại được liên lạc.

6.14.3 KHÔI PHỤC

Ngoài việc ghi chép vào nhật ký tất cả những thông tin đã thảo luận trong phần trước, hệ thống phân tán phải ghi vào nhật ký ở mỗi vị trí những thông báo nó đã gửi và nhận. Sau khi một nút được khôi phục, hoặc nối trở lại vào các thành phần của mạng, nó có trách nhiệm tìm ra xem điều gì đã xảy ra cho những giao dịch đang thực hiện tại nút đó khi xảy ra sự cố. Nhật ký tại một nút sẽ cho nó biết những giao dịch con nào đã bắt đầu nhưng chưa uỷ thác tại nút đó. Nếu thông báo begin - vote đã được nhận, và được ghi vào nhật ký cùng với những thành viên đã biểu quyết. Vì thế nút khôi phục sẽ biết rằng phải hỏi những thành viên nào để có được những thông tin cần thiết.

Chẳng hạn, thành viên T_i có thể đã nhận được thông báo begin - vote đã biểu quyết uỷ thác rồi gặp sự cố. Điều phối viên có thể đã gửi thông báo commit, nhưng T_i không bao giờ nhận được. Tuy nhiên, một thành viên khác, có thể là chính điều phối viên, biết chắc rằng quyết định cuối cùng là uỷ thác, vì thế nó có thể cho nút của T_i biết rằng nó cũng phải uỷ thác. Một thí dụ khác, nếu thông báo begin - vote được ghi vào nhật ký, nhưng không có biểu quyết nào được ghi nhận thì chắc chắn rằng điều phối viên đã đánh dấu quá hạn trong thời gian chờ đợi trả lời, vì thế quyết định huỷ bỏ đã được thực hiện, và T_i có thể huỷ bỏ.

6.15. NGHI THỨC ỦY THÁC KHÔNG PHONG TỎA

Về trực quan, nghi thức uỷ thác hai pha cho phép một thành viên uỷ thác ngay khi nó biết rằng tất cả các thành viên khác đã biểu quyết uỷ thác. Trong *nghi thức uỷ thác ba pha*, một thành viên sẽ không uỷ

thác cho đến khi nó biết được rằng tất cả mọi thành viên đều đã biểu quyết uỷ thác và đồng thời cũng phải biết rằng tất cả mọi thành viên khác cũng đã biết điều đó.

Thí dụ 6.29:

Trong nghi thức uỷ thác hai pha, một thành viên T_i có thể gửi thông báo vote - commit rồi bị mất liên lạc với điều phối viên. Điều phối viên có thể thu thập các biểu quyết gửi commit đến một thành viên T_j khác. Rồi cả điều phối viên và T_j đều gặp sự cố hoặc bị mất liên lạc. Bây giờ T_i và mọi thành viên mà nó có thể liên lạc được, đang trong trạng thái Willing - to - commit nhưng không biết được tất cả các thành viên khác có muốn uỷ thác hay không vì vậy chúng bị phong toả. Tất nhiên Tj đã biết rằng tất cả đều muốn uỷ thác nên nó đã uỷ thác. Việc T_j uỷ thác trước khi nó biết rằng T_i biết mọi thành viên đều muốn uỷ thác, đã làm cho T_i bị phong toả.

Trong nghi thức uỷ thác ba pha, có một đợt thông báo thứ ba. Thông báo thứ hai từ điều phối viên (bây giờ chúng ta gọi là prepare - commit chứ không phải commit) báo cho mọi thành viên biết rằng tất cả đều muốn uỷ thác . Tuy nhiên các thành viên sẽ không uỷ thác khi nhận được thông báo này. Thay vào đó nó xác nhận rằng, nó đã nhận được thông báo prepare - commit bằng cách gửi thông báo ready - commit trở lại cho điều phối viên. Trong pha thứ ba, điều phối viên thu thập tất cả các thông báo, và khi đã nhận được tất cả nó gửi thông báo commit đến tất cả các thành viên, báo cho chúng rằng, tất cả đều muốn ủy thác; lúc đó, các thành viên sẽ thực hiện uỷ thác.

6.15.1. MÔ HÌNH CÁC SỰ CỐ

Chúng ta sẽ trình bày một biến thể uỷ thác ba pha và chứng minh rằng có thể tránh được phong toả miễn là các sự cố "có hình thái giới hạn và hợp lý". Đặc biệt, chúng ta sẽ giả sử rằng:

1. Chỉ có nút bị sự cố, và mạng không bao giờ bị tách thành hai hoặc nhiều nhóm nút vẫn còn hoạt động (nút sống) nhưng không liên lạc được với các nhóm khác. Chẳng hạn, nghi thức Ethernet cho phép liên lạc giữa hai nút sống, bất kể có bao nhiêu nút bị sự cố, chỉ cần chính mạng không bị treo.

2. Khi một nút bị sự cố, nó sẽ không thể liên lạc được. Nó không thể gửi các thông báo sai lạc (chẳng hạn gửi abort trong khi đáng lẽ phải gửi commit), và không gửi một số thông báo trong khi lại bỏ những thông báo khác.

3. Khi một nút bị sự cố, đối với một giao dịch, nó trở thành một người "ngoài cuộc" trong khoảng thời gian thực hiện uỷ thác. Nghĩa là, một nút bị sự cố sẽ biết rằng nó đã gặp sự cố khi được khôi phục, vì vậy nó không tái tham dự vào các quá trình uỷ thác nếu trước tiên nó không thông báo chính nó cho các nút khác biết và biết được điều gì đã xảy ra.

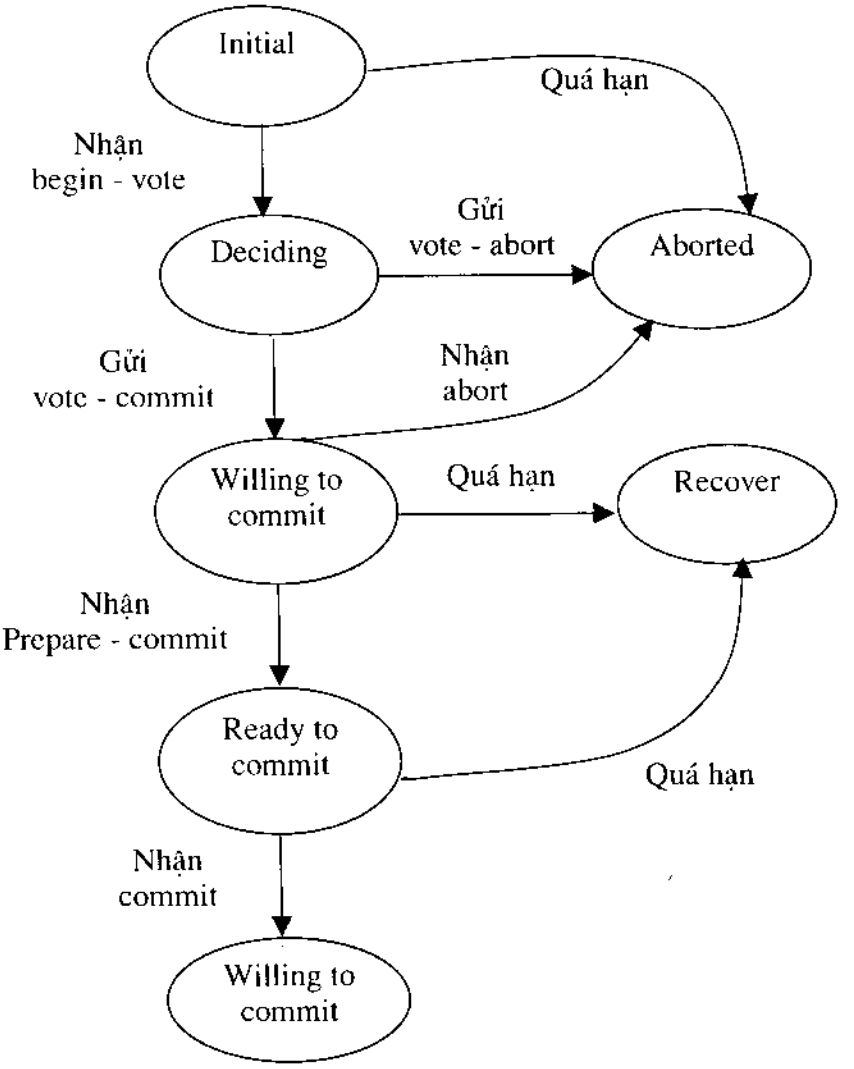
4. Một nút không bị sự cố sẽ phải trả lời các yêu cầu trong một khoảng thời gian quá hạn được dùng để phát hiện nút bị sự cố.

5. Mạng sẽ không làm mất các thông báo, và nó phân phối các thông báo từ nút A đến nút B theo thứ tự chúng được nhận.

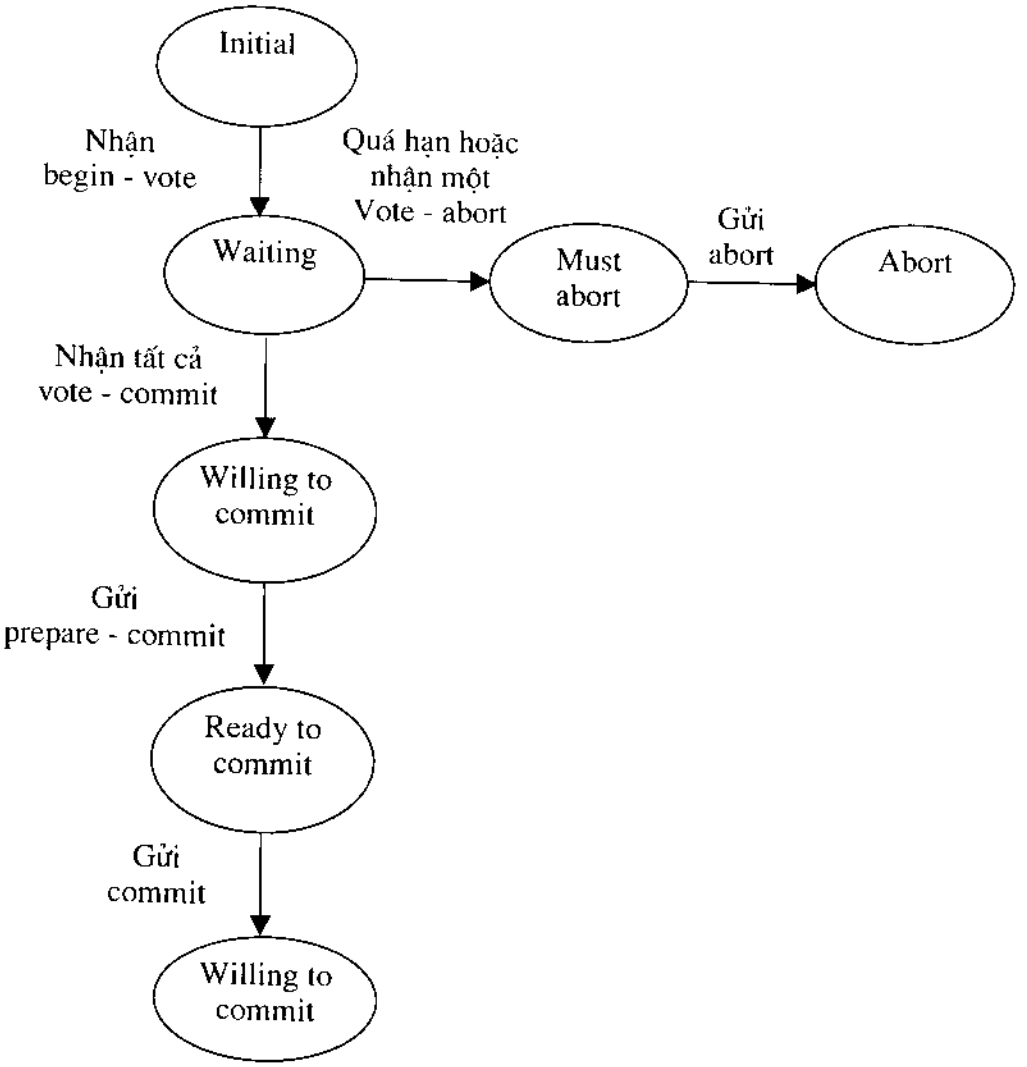
Sau đây chúng ta sẽ mô tả một nghi thức bảo đảm được không có giao dịch nào bị phong toả, chỉ cần các sự cố thoả mãn những giả thiết ở trên.

6.15.2. UỶ THÁC BA PHA

Chúng ta sẽ đưa ra một phiên bản đơn giản của nghi thức uỷ thác ba pha mà nếu tuân theo mô hình sự cố vừa được mô tả, nó sẽ bảo đảm rằng nếu một hoặc nhiều bộ xử lý còn sống trong quá trình uỷ thác thì sẽ không có bộ xử lý nào bị phong toả.



Hình 6.29 (a) Thành viên trong uỷ thác ba pha



Hình 6.29(b). Điều phối viên và uỷ thác ba pha

Phiên bản này bỏ qua thông báo xác nhận ready - commit trong pha thứ hai do không cần thiết. Tuy nhiên, cài đặt thuật toán cụ thể vẫn có thể sử dụng thông báo xác nhận đó bởi vì nó ngăn được một số sự cố khác, chẳng hạn như làm thất lạc các thông báo.

Hình 6.29 hình thức hoá các thảo luận về nghi thức uỷ thác, có thêm một pha thứ ba để xác định rằng tất cả các thành viên đều biết về ý định uỷ thác của mọi thành viên khác.

Trong hình 6.29 (a) là những bước chuyển trạng thái của các thành viên. Chúng ta lược bỏ các tình huống xảy ra khi chuyển đến trạng thái Recover; chúng sẽ được mô tả riêng. Hình 6.29 (b) trình bày các bước chuyển trạng thái của điều phối viên.

Trong pha 1, điều phối viên sẽ gửi thông báo begin - vote đến tất cả các thành viên, và mỗi thành viên sẽ biểu quyết giống y như uỷ thác hai pha. Pha 2 và 3 chỉ xảy ra nếu điều phối viên nhận được vote - commit từ tất cả các thành viên, và trong pha 3, điều phối viên gửi commit, và các thành viên khi nhận được commit sẽ uỷ thác.

Vị trí thứ hai có thể xảy ra một quá hạn khi một điều phối viên đang đợi các biểu quyết. Nếu có một biểu quyết nào đó không đến trong khoảng thời gian hạn định thì giống như uỷ thác hai pha, điều phối viên sẽ quyết định huỷ bỏ giao dịch.

Thứ ba, một thành viên đang muốn uỷ thác có thể đánh dấu quá hạn do đợi thông báo prepare - commit hoặc abort. Nếu, nó chuyển qua trạng thái Rocover sẽ được mô tả riêng.

Vị trí cuối cùng có thể xảy quá hạn là khi một thành viên đang ở trong trạng thái Ready - to - commit không nhận được thông báo commit của điều phối viên. Trong tình huống này, nó cũng chuyển sang trạng thái Recover, ở đó các giao dịch còn sống sẽ giải quyết vấn đề này.

Điểm mấu chốt trong uỷ thác ba pha đó là điều phối viên phải gửi tất cả thông báo prepare - commit trước khi gửi bất kỳ thông báo commit nào. Bởi vì prepare - commit thông tin cho các thành viên biết rằng tất cả đều muốn uỷ thác. Nếu thành viên T_i nhận được commit, nó biết rằng điều phối viên đã gửi đến tất cả các thành viên thông báo prepare - commit, vì vậy mỗi thành viên còn sống đều đã nhận được thông báo prepare - commit hoặc chuẩn bị nhận nó, do thông báo có thể đến trễ nhưng không thể thất lạc trong mạng. Nghĩa là việc T_i nhận được commit đã cho T_i biết rằng tất cả mọi thành viên đều đã biết rằng mọi thành viên đều muốn uỷ thác.

Về lý thuyết, T_i chỉ biết rằng mỗi thành viên T hoặc đã biết rằng tất cả các thành viên đều muốn uỷ thác, hoặc T sẽ biết ngay sau đó, hoặc T sẽ gặp sự cố trước khi nó nhận được prepare - commit. Tuy nhiên, bởi vì nghi thức của hình 6.29 chỉ chứa những thông báo giữa điều phối viên và các thành viên, và bởi vì giả thiết (5) bảo đảm rằng không có thông báo nào bị thất lạc, nên có thể giả sử rằng các thông báo đều được nhận ngay, nghĩa là khi T_i uỷ thác, mọi thành viên hoặc đã nhận được thông báo prepare - commit hoặc sự cố. Lý do là nếu một giao dịch T_j gặp sự cố sau lúc T_i nhận commit nhưng trước T_j nhận prepare - commit thì không có thay đổi nào được nhận ra trong hoạt động của mạng nếu chúng ta giả sử rằng T_j đã gặp sự cố trước T_i nhận commit. Điều chúng ta đã chứng minh là không thể có hai giao dịch đồng thời ở trong hai trạng thái Willing - to - commit và Committed. Vậy ta có bổ đề:

Bổ đề 6.1:

Trước khi các giao dịch chuyển sang trạng thái hồi phục (Recover), các trạng thái sau không tương thích với nhau:

a) Một thành viên (còn sống hoặc bị sự cố) không thể chuyển sang trạng thái Committed trong khi có một thành viên còn sống khác vẫn còn trong trạng thái Willing - to - commit.

b) Một thành viên (còn sống hoặc bị sự cố) không thể chuyển sang trạng thái Aborted trong khi có một thành viên khác (còn sống hoặc bị sự cố) đã chuyển sang trạng thái Committed, hoặc một thành viên còn sống đã chuyển sang trạng thái Ready - to - ommit.

Chứng minh:

Với (a), chúng ta để ý rằng để cho một thành viên chuyển sang trạng thái Committed trước khi quá trình khôi phục xảy ra, nó phải nhận được thông báo commit. Qua lập luận trên, chúng ta biết rằng mỗi thành viên còn sống (theo giả thiết về các thông báo tức thời) đều đã nhận được prepare - commit, vì vậy chúng đều đã rời khỏi trạng thái Willing - to - commit,

Phần (b) được lập luận tương tự, chúng ta chỉ cần phải kiểm tra hình 6.29 và lập luận rằng một thông báo prepare - commit không thể được gửi đi nếu một hoặc nhiều thành viên đã huỷ bỏ.

6.15.3.KHÔI PHỤC TRONG ỦY THÁC BA PHA

Kết quả của bổ đề 6.1 đó là không thể có một thành viên bị sự cố huỷ bỏ nếu một giao dịch còn sống đã đạt tới trạng thái Ready - to - commit, và chúng ta không thể có một thành viên bị sự cố nhưng đã uỷ thác nếu một giao dịch còn sống vẫn đang trong trạng thái Willing - to -,

commit. Vì thế khi một hoặc nhiều thành viên nhận thấy cần phải khôi phục (chuyển sang trạng thái Recover) do có một quá hạn, chúng ta chỉ cần sắp xếp mỗi thành viên còn sống báo cho các thành viên khác biết trạng thái của nó, hoặc chính xác hơn, trạng thái của nó ngay trước khi nó chuyển sang trạng thái Recover. Nếu tất cả đều ở trong trạng thái Willing - to - commit hoặc Aborted thì chúng ta biết rằng không có thành viên bị sự cố nào đã uỷ thác, và như thế tất cả các thành viên đều có thể huỷ bỏ một cách an toàn. Nếu có một thành viên đạt tới trạng thái Ready - to - commit hoặc Committed không có giao dịch bị sự cố nào đã huỷ bỏ, thì tất cả mọi thành viên có thể uỷ thác một cách an toàn.

Trong trường hợp sau, quá trình uỷ thác phân tán phải được thực hiện từng bước. Nghĩa là mọi thành viên còn ở trạng thái Willing - to - commit trước tiên đều phải chuyển sang trạng thái Ready - to - commit, rồi trong trạng thái đó tất cả chúng phải thực hiện uỷ thác. Lý do buộc chúng ta phải thực hiện theo từng giai đoạn đó là tại bất kỳ thời điểm nào đều có thể có nhiều thành viên bị sự cố, và chúng ta tránh tạo ra một tình huống trong đó một thành viên đang trong trạng thái Willing - to - commit trong khi một thành viên khác đã uỷ thác rồi.

6.15.4.CHỌN ĐIỀU PHỐI VIÊN MỚI

Chúng ta có thể giả sử rằng các thành viên được đánh số T_1,...., T_k, và thành viên còn sống với chỉ số thấp nhất sẽ là điều phối viên mới. Vì T_1 cũng có thể bị sự cố hoặc không, nên chúng ta không thể giả sử T_1 là điều phối viên mới. Đúng ra, mỗi thành viên cần phải thông tin cho các thành viên khác biết là nó hiện giờ còn sống.

Một cách khá hiệu quả để thực hiện quyết định là cho mỗi T_i gửi một thông báo mang chỉ số i của nó đến các thành viên T_{i+1}, T_{i+2},...T_k theo thứ tự đó. Tuy nhiên, nếu T_i nhận được một thông báo từ một thành viên có chỉ số thấp hơn, T_i biết rằng nó không phải là điều phối viên mới, vì thế sẽ ngừng gửi thông báo. Phần lớn các thành viên sẽ nhanh chóng ngừng gửi thông báo, nhưng nếu một số thông báo bị chậm trễ, thì khoảng k^2 thông báo vẫn có thể được gửi.

Sau bước này, nếu không có sự cố nào xảy ra trong quá trình bầu chọn thì mỗi thành viên còn sống sẽ biết được ai là điều phối viên mới. Tuy nhiên nếu thành viên có chỉ số thấp nhất lại gặp sự cố trong quá trình bầu chọn thì có thể có bất đồng liên quan đến điều phối viên mới.

Thí dụ 6.30:

Giả sử các thành viên là T_1, T_2, T_3, T_4 và trong quá trình bầu chọn, chuỗi các sự kiện sau xảy ra:

1. T_1 gửi thông báo đến T_2 trước khi T_2 có thể gửi thông báo của nó đến T_3, vì thế T_2 không gửi thông báo nào nữa.

2. T_1 gặp sự cố.

3. T_3 gửi một thông báo đến T_4. Do đó T_4 không gửi thông báo nào nữa.

Tác dụng thực sự của các sự kiện này là T_2 nghĩ rằng T_1 là điều phối viên trong khi T_3 và T_4 cho rằng T_3 là điều phối viên. Sau một thời gian quá hạn thích hợp để qua đó có thể xác định được rằng không còn thông báo nào được gửi nữa, T_3 khởi sự vai trò làm điều phối viên của nó bằng cách hỏi về trạng thái của tất cả các thành viên.

Dễ dàng chứng minh được rằng không thể có quá một thành viên còn sống nghĩ rằng nó là điều phối viên. Giả sử T_i và T_j còn sống và đều

nghĩ rằng chúng là điều phối viên, với i < j. Bởi vì T_i tự xem nó là điều phối viên, nó không bao giờ nhận được một thông báo nào từ những thành viên có chỉ số thấp hơn i. Vì thế nó tiếp tục gửi các thông báo đến những thành viên có chỉ số lớn hơn i, đặc biệt là T_j, vì thế T_j sẽ không còn nghĩ nó là điều phối viên nữa.

Cũng có thể không có thành viên nào tự xem nó là điều phối viên, khi đó những thành viên còn sống sẽ đánh dấu quá hạn khi chờ đợi quá trình khôi phục bắt đầu. Rồi chúng sẽ bầu chọn một điều phối viên mới.

6.15.5. THUẬT TOÁN KHÔI PHỤC

Các bước thực hiện khôi phục được tóm tắt như sau:

1. Các thành viên còn sống bầu chọn một điều phối viên mới.

2. Điều phối viên mới gửi thông báo đến tất cả các thành viên yêu cầu cho biết về trạng thái của chúng ngay trước lúc khôi phục, đó phải là các trạng thái Aborted, Willing - to - commit, Ready - to - commit hoặc committed. Dĩ nhiên, các thành viên bị sự cố sẽ không trả lời, vì thế điều phối viên sẽ chờ đợi trong một khoảng thời gian quá hạn rồi giả sử rằng nó đã có được tất cả các biểu quyết có thể có.

3. Nếu thấy có một trạng thái Ready - to - commit hay Committed, điều phối viên sẽ quyết định uỷ thác giao dịch. Theo bổ đề 6.1, không có thành viên nào huỷ bỏ trong trường hợp này. Nếu chỉ nhận được trả lời Aborted và Willing - to - commit, điều phối viên sẽ quyết định huỷ bỏ giao dịch này.

4. Nếu quyết định huỷ bỏ giao dịch, điều phối viên sẽ gửi thông báo abort đến mỗi thành viên. Nếu quyết định uỷ thác thì điều phối viên sẽ gửi.

a) Thông báo prepare - commit đến các thành viên đang ở trạng thái Willing - to - commit, sau đó là

b) Thông báo commit đến mỗi thành viên chưa chuyển sang trạng thái Committed.

Chỉ cần điều phối viên không gặp sự cố thì các bước trên đây sẽ được hoàn tất. Việc các thành viên khác có thể gặp sự cố trong thời gian đó không ảnh hưởng đến thuật toán. Nếu điều phối viên gặp sự cố tại một thời điểm trung gian nào đó, khi đó có một thành viên sẽ phát hiện ra điều này nhờ cơ chế quá hạn, và toàn bộ quá trình sẽ khởi sự lại từ bước (1). Nếu thuật toán khôi phục dừng tại một điểm trung gian nào đó, hậu quả là có một số thay đổi trạng thái đã xảy ra, chẳng hạn từ Willing - to - commit sang Aborted hoặc Ready - to - commit. Điều quan trọng của thuật toán này là vẫn không thể tồn tại các cặp trạng thái đã được bổ đề 6.1 khẳng định.

Bổ đề 6.2:

Điều kiện (a) và (b) của Bổ đề 6.1 vẫn đúng tại mọi thời điểm trong thuật toán khôi phục.

Chứng minh:

Có một số chi tiết đơn giản chúng ta bỏ qua. Chẳng hạn, trong bước (4), một giao dịch T_i có thể chuyển từ trạng thái Willing - to - commit sang trạng tái Aborted. Không có vi phạm nào đối với bổ đề 6.1 (b), bởi vì điều phối viên chỉ quyết định huỷ bỏ nếu không có thành viên còn sống nào đang ở những trạng thái ngoài hai trạng thái Aborted

và Willing - to - commit. Thế thì bổ đề 6.1 (a) khẳng định rằng không thể có một thành viên còn sống hoặc bị sự cố đang ở trạng thái Committed, suy ra rằng (b) vẫn đúng.

Định lý 6.8:

Thuật toán uỷ thác ba pha được mô tả ở trên không gây phong toả, chỉ cần có ít một thành viên không bị sự cố xảy ra làm cho nó quá hạn và phải khởi động lại. Cuối cùng, thuật toán sẽ thành công hoặc thành viên cuối cùng sẽ gặp sự cố. Vì vậy giao dịch không bị phong toả.

Về tính đúng đắn của thuật toán, bổ đề 6.2 khẳng định rằng những điều kiện của bổ đề 6.1 được duy trì, và dĩ nhiên, bổ đề 6.1 khẳng định rằng chúng đúng trước khi khôi phục. Nếu kết quả cuối cùng là giao dịch uỷ thác thì bổ đề 6.1 (b) khẳng định rằng không có thành viên bị sự cố nào ở trạng thái Aborted. Nếu giao dịch phải huỷ bỏ, bổ đề 6.1(b) khẳng định rằng không có thành viên bị sự cố nào ở trạng thái Committed. Các thành viên đó khi khôi phục và nhận ra chúng đang ở các trạng thái không phải Committed hay Aborted sẽ hỏi những nút còn sống tại thời điểm đó để xác định xem (nếu có thể) quyết định cuối cùng là gì.

6.16. ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN THEO NHÃN THỜI GIAN

Cách tiếp cận nhãn thời gian, được phân tích trong phần 6.10 có thể áp dụng vào cho CSDL phân tán. Về bản chất, những giao dịch thực hiện tại một vị trí, và chúng đọc và ghi vào một bản sao khi cần, để lại

nhãn thời gian của chúng tại vị trí của bản sao làm thời điểm đọc hoặc thời điểm ghi của bản sao. Dĩ nhiên nếu chúng ghi giá trị mới vào bản sao của một mục thì cũng phải ghi giá trị này vào tất cả bản sao của mục đó, và một thuật toán uỷ thác phân tán, được phân tích trong các phần 6.14 và 6.15, phải được sử dụng trước khi bất kỳ giá trị mới nào được ghi vào trong CSDL.

Giống như trong phần 6.10, chúng ta cần một cách nào đó để kiểm tra rằng giao dịch không thực hiện một hành động không thể xảy ra, như đọc một giá trị trước khi nó được ghi nếu các giao dịch được thực hiện theo thứ tự tuần tự theo nhãn thời gian. Trong phần 6.10, chúng ta đã dùng nhãn thời gian và thời điểm đọc, thời điểm ghi cho các mục nhằm duy trì một hoạt động mô phỏng một thứ tự tuần tự. Chúng ta cần nhớ rằng thứ tự tuần tự giả định là thứ tự trong đó một giao dịch được giả sử thực hiện ngay tại thời điểm được ghi nhận qua nhãn thời gian của chúng. Cách tiếp cận này vẫn có giá trị trong môi trường phân tán. Tuy nhiên, khái niệm về nhãn thời gian phải được tổng quát hơn để áp dụng cho CSDL phân tán. Đối với hệ thống phi phân tán, nhãn thời gian nói chung được giả sử là do hệ thống máy tính đưa ra. Nếu chỉ có một máy tính, giả thiết này chắc chắn được thoả mãn. Nhưng điều gì sẽ xảy ra nếu các máy tính tại nhiều điểm đều gắn được nhãn thời gian? Làm thế nào để biết rằng chúng thực hiện nhất quán?

6.16.1. NHÃN THỜI GIAN PHÂN TÁN

Chúng ta có thể để cho các máy tính tại mỗi nút của mạng duy trì các đồng hồ của riêng chúng, mặc dù các đồng hồ này có thể không đồng bộ. Để tránh việc đặt cùng một nhãn thời gian cho hai giao dịch,

chúng ta dùng k bit cuối cùng của "thời gian" để xác định duy nhất các nút. Chẳng hạn, nếu không có quá 256 nút, chúng ta dùng k = 8 và cho mỗi nút chuỗi 8 bit riêng biệt, gắn nó vào đồng hồ cục bộ như những bit thấp để tạo ra nhãn thời gian.

Cần lưu ý rằng, mặc dù những khác biệt nhỏ trong các đồng hồ ở hai nút không gây ra hậu quả nghiêm trọng nhưng với khác biệt lớn nó có thể dẫn đến những tại hoạ lớn. Chẳng hạn, giả sử tại nút N, đồng hồ chậm hơn năm giờ so với các đồng hồ khác của hệ thống,với giả thiết phần lớn các mục được đọc và ghi trong khoảng thời gian năm giờ, một giao dịch khởi hoạt tại N sẽ nhận được nhãn thời gian nhỏ hơn thời điểm đọc và ghi của phần lớn các mục nó cần truy xuất. Vì thế hầu như kết quả là các giao dịch không thể thực hiện được tại N.

Tuy nhiên, có một cơ chế đơn giản để ngăn sự chênh lệch quá lớn của các đồng hồ. Chúng ta vẫn để mỗi thông báo được gửi đi mang nhãn thời gian của riêng nó, là thời điểm tại đó thông báo rời khỏi nút gửi, theo đồng hồ của nút gửi. Nếu một nút nhận được một thông báo "từ tương lai", nghĩa là thông báo mang nhãn thời gian lớn hơn đồng hồ hiện hành của nó, nó chỉ việc tăng giá trị đồng hồ của nó lớn hơn nhãn thời gian của thông báo nhận được. Giả sử nếu một nút bị bất hoạt đến nỗi không khám phá ra rằng đồng hồ của nó đã chậm đến năm giờ thì lần đầu tiên nó thực hiện một giao dịch, nó sẽ nhận được một thông báo yêu cầu huỷ bỏ giao dịch nó đang thực hiện. Thông báo đó sẽ bao gồm cả "thời gian đúng". Và nút đó sẽ cập nhật tại đồng hồ rồi thực hiện lại giao dịch với nhãn thời gian mới. Vì vậy từ giờ trở đi chúng ta sẽ giả sử là nhãn thời gian luôn hợp lý, có giá trị toàn cục nằm trong khả năng của hệ quản trị CSDL phân tán.

6.16.2. THUẬT TOÁN DỰA VÀO NHÃN THỜI GIAN

Như trong phần 6.11, chúng ta sẽ xem những bước cơ bản là một hoạt động trên một bản sao của một mục, không phải trên chính mục đó. Tuy nhiên, khi giải quyết với nhãn thời gian, các bước cơ bản không là khoá hoặc mở khoá nhưng là kiểm tra và gán *thời điểm đọc* và *thời điểm ghi* trên các bản sao.

Chúng ta sẽ thảo luận ở đây một phương pháp tương ứng với khoá ghi tất cả. Khi đọc một mục A, chúng ta sẽ lấy một bản sao của A và kiểm tra thời điểm ghi của nó không được vượt quá nhãn thời gian của giao dịch đang đọc. Nếu thời điểm ghi lớn hơn nhãn thời gian, chúng ta phải huỷ bỏ giao dịch này.

Khi ghi A, phải ghi tất cả bản sao của A, và phải bảo đảm rằng đối với mỗi bản sao, thời điểm đọc phải nhỏ hơn nhãn thời gian của giao dịch. Nếu thời điểm đọc của một bản sao lớn hơn nhãn thời gian, giao dịch phải huỷ bỏ. Nếu thời điểm đọc nhỏ hơn nhãn thời gian, nhưng thời điểm ghi lại lớn hơn nhãn thời gian thì chúng ta không huỷ bỏ giao dịch, nhưng cũng không ghi mục này vì lý do đã được thảo luận trong phần 6.10.

Dễ dàng kiểm tra rằng nếu theo những quy tắc này một giao dịch sẽ không bao giờ đọc một giá trị được tạo ra "trong tương lai" hoặc không thể ghi một giá trị nếu một giá trị được ghi trước đó "sẽ được đọc trong tương lai". Vì vậy phương pháp này bảo đảm sẽ tạo ra một tác dụng tương đương với tác dụng của thứ tự mà trong đó, các giao dịch xảy ra theo thứ tự của nhãn thời gian.

6.16.3.KHOÁ CHỐT VÀ NHÃN THỜI GIAN

Cách tiếp cận nhãn thời gian tiết kiệm được một số thông báo, thao tác đọc chỉ cần một thông báo điều khiển và một thông báo dữ liệu để yêu cầu và nhận dữ liệu, còn một thao tác ghi cần n thông báo dữ liệu với n là số vị trí có bản sao.

Khuyết điểm của nó, giống như trong trường hợp phi phân tán, nhãn thời gian sẽ làm cho nhiều giao dịch phải huỷ bỏ và khởi động lại nếu có nhiều tình huống trong đó hai giao dịch đang cố truy xuất cùng một mục tại cùng một thời điểm.

6.17. KHÔI PHỤC CÁC NÚT

Như chúng ta đã đề cập trong các phần 6.14 và 6.15 khi một nút gặp sự có phải khôi phục, nó không chỉ đơn giản tái hoạt động trở lại, mà nó phải kiểm tra nhật ký và xác định giao dịch nào đang được uỷ thác khi nó gặp sự cố. Thông thường qua nhật ký, nó không thể biết được một giao dịch đã uỷ thác hay đã huỷ bỏ; vì vậy nó phải gửi thông báo đến ít nhất một thành viên khác để xác định xem điều gì đã xảy ra.

Trong thực tế, có thể có một vị trí N gặp sự cố và không khôi phục lại được trong một thời gian dài. Việc N gặp sự cố sẽ được một vị trí M khác phát hiện ngay khi một giao dịch khởi sự tại M cố truy xuất một bản sao nằm tại N của mục A. Nếu N chứa bản sao duy nhất của A, giao dịch này buộc phải huỷ bỏ, và chúng ta không cần phải làm thêm điều gì nữa. Tuy nhiên nếu A có nhiều bản sao, chúng ta vẫn có thể tiến hành và xem như không có bản sao tại nút N. Khi N khôi phục, nó không chỉ có trách nhiệm tìm ra những giao dịch đang uỷ thác hoặc huỷ

bỏ khi nó gặp sự cố mà còn phải tìm xem những mục nào không còn phù hợp nữa, theo nghĩa là có những giao dịch đã thực hiện tại những vị trí khác và đã sửa đổi các bản sao của một mục có một bản sao tại N và ở những vị trí khác.

6.17.1,CẬP NHẬT GIÁ TRỊ

Khi một vị trí hoạt động trở lại, nó phải ghi những giá trị mới nhất của tất cả các mục của nó. Ta sẽ nêu hai chiến lược tổng quát để thực hiện điều này.

1. Nếu vị trí M phát hiện rằng N đã gặp sự cố, M ghi sự kiện này vào nhật ký. Khi N khôi phục, nó gửi thông báo đến mỗi vị trí. Nếu M nhận được thông báo này, nó xem lại nhật ký và tìm lại điểm N gặp sự cố rồi gửi giá trị mới nhất của các mục chung của N và M cho N. Những giá trị của các mục này phải bị khoá trong khi đang tiến hành việc khôi phục lại N, và chúng ta phải chú ý ghi nhận giá trị mới nhất bởi vì tất cả các giao dịch đã uỷ thác một giá trị cho mục A phải thực hiện việc uỷ thác theo cùng một thứ tự tại tất cả các vị trí của A, với điều kiện là chúng ta có một phương pháp khoá thích hợp. Nếu chúng ta đang sử dụng nhãn thời gian, thời điểm ghi các giá trị sẽ xác định thứ tự của chúng.

2. Tất cả bản sao của tất cả các mục có thể được gán một thời điểm ghi dù hiện có sử dụng phương pháp điều khiển đồng thời dựa trên nhãn thời gian hay không. Khi một vị trí N khôi phục lại, nó gửi đi tất cả thời điểm ghi của tất cả các mục của nó, như đã được ghi nhận tại các vị trí khác, những mục này tạm thời bị khóa tại những vị trí khác, và

giá trị hiện hành của những mục có thời điểm ghi gần hơn thời điểm ghi tại N sẽ được gửi lại cho N.

Mô tả này chỉ đề cập lướt qua vấn đề quản lý sự cố. Chẳng hạn chúng ta phải xét xem điều gì sẽ xảy ra khi một vị trí có chứa các giá trị dùng để cập nhật cho một vị trí thứ hai nhưng chính nó lại bị sự cố, hoặc nếu một vị trí gặp sự cố trong khi một vị trí khác đang khôi phục. . . Trong phần 6.1 chúng ta đã có những phương pháp hữu hiệu và đơn giản để ngăn cản các khoá gài trong những hệ thống đơn bộ xử lý. Chúng ta có thể tổng quát hoá kỹ thuật này cho CSDL phân tán. Nếu chúng ta dùng phương pháp khoá như các lược đồ "k trong n", là phương pháp khoá các bản sao riêng rẽ, chúng ta vẫn có thể tránh được khoá gài nếu chúng ta yêu cầu tất cả các giao dịch khoá các bản sao theo một thứ tự đặc biệt.

1. Nếu A < B theo tứ tự chữ cái thì giao dịch T phải khoá tất cả các bản sao cần dùng của A trước khi khoá các bản sao cần dùng của B.

2. Các bản sao của mỗi mục A được xếp thứ tự, và một giao dịch khoá tất cả các bản sao cần dùng của A theo thứ tự đó.

Trong phần còn lại chúng ta xem sơ qua một số phương pháp tổng quát để phát hiện và tránh được khoá gài nhưng không ràng buộc thứ tự truy xuất các mục. Đó là các phương pháp: *sử dụng các quá hạn thời gian (timeout), xây dựng một đồ thị chờ (Waits - for graph)*, hoặc cách *tiếp cận nhãn thời gian để tránh khoá.*

6.17.2. SỬ DỤNG CÁC QUÁ HẠN THỜI GIAN

Một cách đơn giản để phát hiện khoá gài là buộc một giao dịch phải đánh dấu quá hạn và huỷ bỏ nếu nó đã đợi một khoá trong một

khoảng thời gian đủ để nghĩ rằng nó bị vướng vào một khoá gài. Thời gian quá hạn phải đủ ngắn để những giao dịch bị khoá gài không giữ các khoá quá lâu, nhưng cũng phải đủ dài để không phải thường xuyên buộc huỷ bỏ các giao dịch thực sự không bị khoá gài.

Phương pháp này có một số ưu điểm, nó không cần phải đưa thêm thông báo mới. Tuy nhiên, nó có xu hướng huỷ bỏ tất cả hoặc nhiều giao dịch bị vướng trong khoá gài chứ không phải chỉ một giao dịch mà đủ để tháo gỡ khóa gài.

6.17.3. ĐỒ THỊ CHỜ

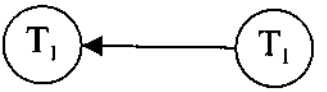
Chúng ta đã đề cập trong phần 6.1 rằng một phép kiểm tra khoá gài cần và đủ trong một hệ thống đơn bộ xử lý là xây dựng một đồ thị chờ (waits - for - graph) với các nút là các giao dịch. Đồ thị có một cung từ T_1 đến T_2 nếu T_1 đang chờ một khoá trên một mục được giữ bởi T_2. khi đó một khoá gài xảy ra nếu và chỉ nếu có một chu trình trong đồ thị này. Về nguyên tắc, kỹ thuật này cũng hoạt động được trong môi trường phân tán. Vấn đề là tại mỗi vị trí, chúng ta dễ dàng duy trì một đồ thị chờ cục bộ (lacal wait- for - graph), trong khi các chu trình có thể chỉ xuất hiện trong đồ thị chờ toàn cục (global waits - for - graph), là hợp của các đồ thị cục bộ.

Thí dụ 6.31:

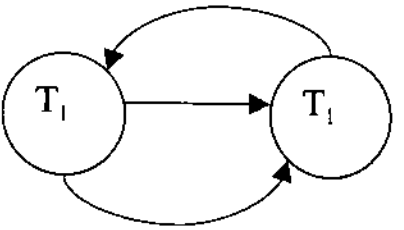
Giả sử chúng ta có các giao dịch T_1 và T_2 muốn khoá các mục A và B tại các nút N_A và N_B tương ứng. A và B có thể là các bản sao của cùng một mục hoặc có thể là các mục khác nhau. Cũng giả sử rằng tại N_A, T_2 (hoặc một giao dịch con của T_2) đã nhận một khoá ghi trên A, và T_1

(hoặc một giao dịch con của T_1) đang đợi khoá đó. Đồng thời tại N_B, T_1 có một khoá trên B trong khi T_2 đang đợi khoá đó.

(a) Đồ thị chờ cục bộ tại N_A



(b) Đồ thị chờ cục bộ tại N_B.



(c) Đồ thị chờ toàn cục

Hình 6.30 Phát hiện khoá gài toàn cục

Đồ thị chờ tai N_A và N_B được trình bày trong hình (a), (b), tất nhiên chúng không có chu trình. Hợp của những đồ thị này là đồ thị (c) có chu trình.

Thí dụ 6.31 giải thích tại sao để phát hiện các chu trình lại cần phải gửi các thông báo nhằm xây dựng một đồ thị chờ toàn cục. Có nhiều cách để thực hiện tác vụ này thí dụ:

1. Sử dụng một nút trung tâm để nhận các cập nhật cho các đồ thị chờ cục bộ từ tất cả các vị trí theo định kỳ. Kỹ thuật này có những ưu và khuyết điểm của những phương pháp khoá tập trung: nó dễ bị ngừng trệ khi nút trung tâm gặp sự cố và khi lưu lượng thông báo tập trung quá nhiều về nút này mặc dù tổng lưu lượng thấp. Hoặc

2. Chuyển các đồ thị chờ đợi cục bộ hiện hành qua tất cả các vị trí, bằng cách gắn nó vào một thông báo hướng đến một vị trí khác nếu có thể, nhưng theo định kỳ vẫn gửi đồ thị cục bộ này đến từng vị trí khác. Lưu lượng thông báo được sinh ra có thể lớn hơn so với phương pháp nút trung tâm. Tuy nhiên, nếu chi phí các thông báo không đổi so với chiều dài của chúng, và các thông tin của đồ thị chờ thường có thể "bám" vào những thông báo khác thì chi phí thực sự để chuyển thông tin sẽ nhỏ.

6.17.4. NGĂN CHẶN KHOÁ GÀI DỰA VÀO NHÃN THỜI GIAN

Điều quan trọng cần chú ý là nhãn thời gian chỉ dùng để tránh khoá gài; điều khiển truy vấn sử dụng phương pháp khoá chốt.

Theo một lược đồ, nếu T_1 (một giao dịch con của T_1) đang đợi T_2 (một giao dịch con của T_2) thì nhãn thời gian của T_1 phải nhỏ hơn nhãn thời gian của T_2, trong lược đồ thứ hai, điều ngược lại sẽ đúng. Trong mỗi lược đồ này, một chu trình trong đồ thị chờ sẽ chứa những giao dịch có nhãn thời gian tăng đơn điệu hoặc giảm đơn điệu khi chúng ta đi vòng quanh chu trình. Không có trường hợp nào có thể xảy ra bởi vì khi chúng ta đi vòng quanh chu trình, chúng ta sẽ trở lại vị trí nhãn thời gian lúc đầu.

Bây giờ chúng ta định nghĩa hai lược đồ tránh khoá gài. Giả sử chúng ta có các giao dịch T_1 và T_2 với nhãn thời gian tương ứng là t_1 và t_2 . Một giao dịch con của T_i đang muốn truy xuất một mục A bị khoá bởi một giao dịch con của T_2 .

1. Trong lược đồ chờ - huỷ diệt (wait - die): nếu $t_1 < t_2$ thì T_1 sẽ chờ để khoá mục A, ngược lại nếu $t_1 > t_2$ thì T_1 phải huỷ bỏ.

2. Trong lược đồ gây thương tổn - chờ (wound - wait): nếu $t_1 > t_2$ T_1 sẽ chờ để khoá mục A, ngược lại, nếu $t_1 < t_2$ thì T_2 phải huỷ bỏ và giải phóng khoá trên A cho T_1 .

Trong mỗi lược đồ, giao dịch bị huỷ bỏ phải khởi động lại với cùng nhãn thời gian cũ, không phải với một nhãn thời gian mới. Việc sử dụng lại nhãn thời gian ban đầu bảo đảm rằng giao dịch "già nhất" trong mỗi lược đồ không thể bị huỷ diệt hoặc bị thương tổn. Vì vậy cuối cùng mỗi giao dịch sẽ được phép hoàn tất, định lý sau đây sẽ khẳng định điều này.

Định lý 6.9:

Không có khoá gài cũng như không có khoá sống trong các lược đồ chờ huỷ diệt (wait- die) và thương tổn - chờ (wound-wait).

Chứng minh:

Xét lược đồ chờ - huỷ diệt. Giả sử có một chu trình trong đồ thị toàn cục, nghĩa là chuỗi các giao dịch $T_1, ..., T_k$ sao cho mỗi T_i đang đợi giải phóng một khoá từ T_{i+1} với $1 \le i < k$ và T_k đang đợi T_1 . Gọi t_i là nhãn thời gian của T_i . Thế thì $t_1 < t_2 < ... < t_k < t_1$ suy ra $t_1 < t_1$ là điều không thể tồn tại: tương tự, trong lược đồ gây thương tổn - chờ, một chu trình như thế cũng kéo theo $t_1 > t_2 > ... > t_k > t_1$ và cũng là điều không thể tồn tại.

Để chứng tỏ tại sao không có khoá sống, chúng ta lại xét lược đồ chờ - huỷ diệt. Nếu T là một giao dịch có nhãn thời gian nhỏ nhất, nghĩa là T là giao dịch già nhất chưa hoàn tất thì T không bao giờ chết. Nó có thể đợi những giao dịch"trẻ hơn" giải phóng khoá của chúng nhưng bởi vì không có khoá gài, cuối cùng những khoá này sẽ được giải phóng, và cuối cùng T sẽ hoàn tất. Lúc đầu khi T khởi sự, có một số hữu hạn các giao dịch "già hơn" còn sống. Như lập luận ở trên, mỗi giao dịch này cuối cùng sẽ hoàn tất, khi đó T trở thành "già nhất". Lúc đó, T chắc chắn sẽ hoàn tất vào lần khởi động sau. Dĩ nhiên trong hoạt động bình thường, các giao dịch không cần phải hoàn tất theo thứ tự tuổi của chúng và phần lớn sẽ tiến hành mà không cần phải huỷ bỏ.

Lập luận tương tự cho lược đồ gây thương tổn – chờ. Ở đây, giao dịch già nhất không phải chờ những giao dịch khác giải phóng khoá; nó lấy những khoá nó cần và "làm tổn thương" những giao dịch đang giữ chúng.

BÀI TẬP

6.1. Trong hình 6.31 dưới đây, chúng ta thấy một lịch biểu của bốn giao dịch. Giả sử rằng khoá ghi hàm chứa cả thao tác đọc như trong phần 6.4. Hãy vẽ một đồ thị tuần tự hoá và xác định xem lịch biểu có khả tuần tự không.

6.2. Lập lại bài tập 6.1 với giả thiết của phần 6.6, trong đó khoá hàm chứa thao tác đọc giá trị.

6.3. Trong hình 6.32 có hai giao dịch. Chúng ta có thể xếp lịch một cách hợp lệ cho chúng bằng bao nhiêu cách? Có bao nhiêu lịch biểu trong số này khả tuần tự?

6.4*. Hãy cho một thí dụ giải thích tại sao giả thiết của phần 6.2 là giả thiết buộc phải có một hàm duy nhất đi kèm theo một lần một giao dịch khoá một mục, là một yêu cầu quá "cao". Nghĩa là hãy đưa ra một lịch biểu cho các giao dịch mà thuật toán 6.1 khẳng định bất khả tuần tự nhưng thực sự có tác dụng như một lịch biểu tuần tự.

6.5. Giả sử chúng ta sử dụng phương pháp điều khiển đồng thời dựa vào nhãn thời gian. Hãy diễn giải lại các thao tác của Hình 6.31 là, RLOCK được xem là READ, WLOCK là WRITE và các bước UNLOCK không tồn tại. Giao dịch nào phải huỷ bỏ nếu giả thiết rằng nhãn thời gian của T_1 và T_4 tương ứng là:

a) 300, 310, 320 và 330

b) 250, 200, 210 và 275

Trong mỗi trường hợp, hãy cho biết thời điểm đọc và ghi cuối cùng của A, B và C?

(1)		RLOCK A		
(2)			RLOCK A	
(3)		WLOCK B		
(4)		UNLOCK A		
(5)			WLOCK A	
(6)		UNLOCK B		
(7)	RLOCK B			
(8)			UNLOCK A	
(9)				RLOCK B
(10)	RLOCK A			
(11)				UNLOCK B
(12)	WLOCK C			
(13)	UNLOCK A			
(14)				WLOCK A
(15)				UNLOCK A
(16)	UNLOCK B			
(17)	UNLOCK C			
	T_1	T_2	T_3	T_4

Hình 6.31 Một lịch biểu

LOCK A	LOCK B
LOCK B	UNLOCK B
UNLOCK A	LOCK A
UNLOCK B	UNLOCK A
T_1	T_2

Hình 6.32 Hai giao dịch

6.6. Tổng quát nghi *thức cảnh báo* của phần 6.7 cho phép các khoá đọc và khoá ghi, và các cảnh báo liên quan đến những khoá này. Vì thế về nguyên tắc có 16 thể thức khoá chốt mà một giao dịch có thể cho một mục, tương ứng với 16 tập con của hai loại khoá và hai loại cảnh báo. Tuy nhiên cũng có một số tổ hợp vô dụng. Chẳng hạn không cần thiết phải đặt một cảnh báo đọc và cảnh báo ghi trên cùng một mục bởi vì cảnh báo ghi cấm tất cả mọi hành động đọc cấm. Như vậy có bao nhiêu thể thức khoá chốt khác nhau mà một giao dịch muốn đặt cho mục. Hãy trình bày một ma trận tương thích cho những thể thức khoá này. Chẳng hạn chúng ta có hai giao dịch, mỗi giao dịch có thể đặt một cảnh báo đọc trên một mục nhưng không thể đặt một khoá đọc khi giao dịch kia đã đặt cảnh báo ghi.

6.7. Hai thể thức khoá là tương đương trong một ma trận tương thích nếu chúng có các hàng và các cột giống nhau. Chứng tỏ rằng chỉ có năm thể thức khoá không tương đương trong bảng của bài tập 6.6.

6.8. Giả sử tập các mục tạo ra một đồ thị DAG (directed acyclic graph, đồ thị có hướng không vòng). Chứng tỏ rằng nghi thức dưới đây bảo đảm được tính khả tuần tự:

i) Khoá đầu tiên có thể đặt trên một nút bất kỳ.

ii) Sau đó, một nút n chỉ có thể bị khoá nếu giao dịch này giữ khoá ít nhất tại một *nút tiền bối (predecessor) của n* và giao dịch này đã khoá mỗi nút tiền bối của n vào một thời điểm nào đó trong quá khứ.

6.9. Chứng minh rằng nghi thức sau cũng an toàn cho các đồ thị DAG.

i) Khoá đầu tiên có thể đặt trên một nút bất kỳ.

ii) Sau đó, một giao dịch có thể khoá một nút n chỉ nếu nó giữ

khoá trên hơn phân nửa (majorty) các nút tiền bối của n.

6.10. Chứng minh rằng khoá chốt hai pha là cần và đủ cho tính khả tuần tự trong mô hình của phần 6.4 (khoá đọc và khoá ghi đều hàm chứa thao tác đọc).

6.11. Thay vì lưu các khoá vào một bảng khoá, chúng ta có thể lưu khoá kèm với chính các mục. Cách tiếp cận này gây ra vấn đề gì?

Hướng dẫn: Hãy xét số lần truy xuất khối với giả thiết là bảng khoá sẽ đặt vừa vào trong bộ nhớ nhưng toàn bộ CSDL thì không. Đồng thời phải xét xem cần phải sử dụng cấu trúc nào để lưu các khoá kèm với các mục, giả sử rằng thể thức khoá như READ có thể được nhiều giao dịch giữ cùng một lúc.

6.12. Nhãn thời gian cũng có thể được lưu cùng với các mục thay vì lưu riêng trong một bảng. Ở phạm vi nào những khuyết điểm xuất hiện trong phương pháp khoá (như được đề cập trong bài tập 6.11) cũng xảy ra trong phương pháp nhãn thời gian?

6.13*. Chúng ta hãy xét một quan hệ R biểu diễn các tài khoản tại một ngân hàng. Giả sử rằng các mục là các bộ của R, nghĩa là mỗi bộ được khoá riêng rẽ. Nhiều giao dịch cần ghi giá trị mới của thuộc tính BALANCE của R và vì vậy không chắc có được một thời gian sao cho không có bộ nào bị khoá ghi. Đôi khi chúng ta muốn thực hiện một giao dịch dài dùng để tính tổng các số dư của R. Một số vấn đề có thể xảy ra là:

i) Thiếu tính khả tuần tự, nghĩa là tổng các số dư không phản ảnh tình huống đã xảy ra tại một thời điểm nào đó trong quá trình hoạt động của ngân hàng.

ii) Chờ đợi quá lâu; nghĩa là những giao dịch tính tổng số dư phải đợi vô hạn định.

iii) Chờ đợi quá lâu; nghĩa là những giao dịch ngắn phải đợi giao dịch tính tổng số dư hoàn tất.

iv) Cuộn ngược dây chuyền trong trường hợp có sự cố hệ thống.

v) Không thể hồi phục khi có sự có hệ thống (không phải sự cố vật liệu).

Hãy chỉ ra xem vấn đề nào có thể xảy ra nếu chúng ta sử dụng mỗi chiến lược điều khiển đồng thời dưới đây.

a) Khoá chốt hai pha nghiêm ngặt với tất cả các khoá được trao khi giao dịch cần sử dụng.

b) Khoá chốt hai pha nghiêm ngặt với tất cả các khoá được trao khi giao dịch bắt đầu.

c) Khoá chốt hai pha không nghiêm ngặt với các khoá được trao khi cần và giải phóng ngay sau điểm khoá bởi vì chúng không cần thiết nữa.

d) Khoá chốt không phải hai pha với các khoá được trao ngay khi đọc hoặc ghi và giải phóng ngay sau khi đọc hoặc ghi.

e) Điều khiển đồng thời lạc quan dựa vào nhãn thời gian với nhã thời gian được kiểm tra vào lúc kết thúc giao dịch.

f) Giống (e) nhưng với nhãn thời gian được kiểm tra khi các mục được đọc hoặc được ghi.

g) Một lược đồ đa phiên bản dựa vào nhãn thời gian với những phiên bản phù hợp được đọc và nhãn thời gian được kiểm tra vào lúc ghi.

6.14. Trong hình 6.33. có một danh sách các giao dịch và các mục bị

chúng khoá. Chúng ta giả sử rằng 5 giao dịch đều sẵn sàng để bắt đầu theo đúng thứ tự như trong hình. T_1 bắt đầu trước tiên rồi kết thúc sau khi T_3 đã sẵn sàng nhưng trước khi T_4 sẵn sàng không có giao dịch nào khác kết thúc cho đến khi tất cả 5 giao dịch đều sẵn sàng. Giả sử chúng ta sử dụng *nghi thức bảo toàn tránh khóa sống* và *khoá gài* của định lý 6.7. Hãy chỉ ra thứ tự các giao dịch thực sự bắt đầu và hàng đợt tại một bước.

Giao dịch	Các mục khoá
T_1	{A, B}
T_2	{A, C}
T_3	{B, C}
T_4	{B}
T_5	{C}

Hình 6.33 Cho bài tập 6.14

6.15. Giả sử trong mạng có ba nút 1, 2 và 3. Mục A có các bản sao tại ba nút, còn mục B chỉ có bản sao tại nút 1 và 3. Hai giao dịch T_1 và T_2 khởi hoạt cùng lúc tương ứng tại nút 1 và 2. Mỗi giao dịch thực hiện các bước sau:

RLOCK B; WLOCK A; UNLOCK A; UNLOCK B;

Giả sử rằng trong mỗi đơn vị thời gian, mỗi giao dịch có thể gửi một thông báo đến một vị trí, mỗi vị trí có thể đọc một thông báo. Khi cần phải chọn lựa vị trí gửi hoặc nhận thông báo, hệ thống luôn chọn vị trí được đánh số thấp nhất. Các thông báo bổ sung sẽ được đặt vào một hàng đợi để gửi hoặc nhận ở những đơn vị thời gian kế tiếp. Hãy mô phỏng hoạt động của mạng theo các quy tắc điều khiển đồng thời dưới đây:

a) Khoá ghi tất cả.

b) Khoá quá bán.

c) Theo vị trí chính với giả thiết rằng nút 1 cho A và nút 3 cho B.

d) Thẻ bản chính, khởi đầu với vị trí 2 và 3 giữ thẻ đọc cho A và 1 giữ thẻ ghi cho B.

e) Điều khiển đồng thời dựa vào nhãn thời gian, giả sử rằng nhãn thời gian của T_1 vượt quá nhãn thời gian của T_2 và cả hai đều lớn hơn các thời điểm ghi và đọc ban đầu của tất cả các bản sao.

6.16. Chứng tỏ rằng để sự cố của hai đường liên lạc không cắt đứt liên lạc trong mạng gồm n nút, mạng phải có ít nhất 3n/2 cạnh. Đồng thời cũng chứng tỏ rằng có những mạng với [3n/2] (phần nguyên bé hơn 3n/2) cạnh không bị cắt đứt liên lạc bởi sự cố của hai đường liên lạc.

6.17. Một mạng n nút cần có bao nhiêu cạnh để có thể thích ứng được khi có sự cố của k đường liên lạc bất kỳ.

6.18. Giả sử rằng một khoá đọc toàn cục (logic) yêu cầu phải khoá đọc j bản sao cục bộ (vật lý) và một khoá ghi logic cần phải khoá ghi trên k bản sao. Chứng tỏ rằng nếu $j + k \leq n$ hoặc $k \leq n/2$ thì các khoá logic không hoạt động như chúng ta mong đợi (vì vậy chiến lược k trong n là cách lựa chọn khả thi nhất).

6.19. Hãy xác định số lượng thông báo trung bình được sử dụng bởi phương pháp thẻ bản chính để định nghĩa khoá với giả thiết là khi cần khoá một mục A:

i) 50% thời gian một thẻ ghi cho A có sẵn ở vị trí cục bộ (và vì vậy không có thẻ đọc).

ii) 40% thời gian thẻ đọc cho A có sẵn ở vị trí cục bộ.

iii) 10% thời gian không có sẵn thẻ ghi lẫn thẻ đọc cho A tại vị trí cục bộ.

iv) Khi một thẻ được yêu cầu không có sẵn tại vị trí cục bộ, tất các vị trí đều giao nộp các thẻ chúng hiện có cho vị trí yêu cầu sau một quá trình trao đổi thông báo.

6.20. Điều gì xảy ra khi giao dịch của hình 6.26 thực hiện theo những phương pháp khoá khác với phương pháp khoá ghi tất cả?[128]

6.21. Chúng ta có thể thực hiện một quá trình uỷ thác hai pha phân tán mà không cần một điều phối viên nếu chúng ta yêu cầu mỗi thành viên (trong n thành viên) gửi biểu quyết đến tất cả n - 1 thành viên còn lại.

a) Hãy vẽ một sơ đồ trạng thái dịch chuyển cho mỗi thành viên, bao gồm cả các hoạt động phải thực hiện nếu cần phải khôi phục.

b) Cần phải có bao nhiêu thông báo?

c) Nếu sử dụng thao tác phát tán, có bao nhiêu thao tác cần thực hiện?

d) Các giao dịch có bị phong toả hay không? Nếu có hãy đưa ra một thí dụ.

6.22. Lập lại bài tập 6.22 cho quá trình uỷ thác ba pha không có điều phối viên.

6.23. Một cách tiếp cận để uỷ thác hai pha phân tán là sắp xếp các thành viên vào trong một vòng xoay (ring) và hy vọng rằng chúng truyền và lưu giữ các biểu quyết theo vòng này, bắt đầu từ điều phối viên rồi cuối cùng quay trở lại điều phối viên. Sau đó điều phối viên chuyển kết quả biểu quyết quanh vòng. Trong trường hợp có một hoặc nhiều nút gặp sự cố, các thành viên sẽ bỏ qua những vị trí có sự cố để tìm thành viên kế

tiếp còn sống. Hãy thực hiện lại các câu hỏi của bài tập 6.22 cho mô hình này.

6.24. Lập lại bài tập 6.21 uỷ thác ba pha

6.25. Chúng ta cũng đã khẳng định rằng trong thời gian khôi phục với uỷ thác hai pha, nếu một thành viên chưa gửi biểu quyết uỷ thác thì tình trạng không nhất quán có thể xảy ra. Hãy đưa ra một tình huống minh hoạ cho kết luận này.

6.26. Giả sử có bốn thành viên T_1 (điều phối viên), T_2, T_3 và T_4 đang uỷ thác hai pha. Hãy dự đoán điều gì sẽ xảy ra nếu các sự cố sau đây xảy ra. Trong mỗi trường hợp hãy chỉ ra xem điều gì xảy ra trong quá trình khôi phục (nếu pha khôi phục được đưa vào) và cho biết xem có giao dịch nào bị phong tỏa không?.

a. T_1 gặp sự cố sau khi gửi vote - commit đến T_2 và T_3 nhưng chưa gửi đến T_4.

b. T_2 gặp sự cố sau khi gửi vote - abort; những thành viên khác biểu quyết uỷ thác.

c. T_2 gặp sự cố trước khi biểu quyết; những thành viên khác biểu quyết uỷ thác.

d. Tất cả đều biểu quyết uỷ thác nhưng T_1 gặp sự cố trước khi gửi thông báo commit.

e. Tất cả đều biểu quyết uỷ thác và T_1 gặp sự cố sau khi gửi thông báo commit đến T_2 (chỉ mình T_2)

f. Tất cả đều biểu quyết uỷ thác và T_1 gửi commit đến tất cả các giao dịch khác nhưng T_2 gặp sự cố trước khi nhận được thông báo commít.

6.27. Chứng minh rằng trong uỷ thác ba pha, nếu điều phối viên gửi commit dù chỉ đến một thành viên trước khi gửi prepare - commit đến mọi thành viên thì hành động lỗi (hoặc phong toả) có thể xảy ra theo mô hình sự cố của phần 6.15.

6.28. Có thể tồn tại một hành động gây lỗi trong uỷ thác ba pha hay không nếu mô hình sự cố của phần 6.15 được sửa đổi lại, cho phép tình huống thất lạc các thông báo ngay cả khi không có nút hoặc đường liên lạc nào bị sự cố. Giả sử rằng không có thông báo xác nhận cho thông báo prepare - commit nhưng một thành viên đang đợi commit có thể đánh dấu quá hạn và chuyển sang trạng thái Recover. Điều gì sẽ xảy ra nếu các thông báo prepare - commit được xác nhận.

6.29*. Hoàn tất phần chứng minh bổ đề 6.1 (b):

6.30*. Hãy xét thuật toán bầu chọn lãnh đạo được mô tả trong phần 6.15, được áp dụng cho một tập gồm k thành viên.

a. Chứng minh rằng thuật toán có thể sử dụng đến O(k) thông báo.

b. Giả sử rằng tất cả thông báo đều cần một thời gian như nhau. Chứng minh rằng chỉ có O(k) thông báo được sử dụng, giả thiết là không có sự cố.

c. Điều gì sẽ xảy ra nếu tất cả thông báo đều cần thời gian như nhau nhưng xảy ra sự cố trong khi đang bầu chọn? Hãy cho biết số thông báo tối đa có thể được gửi đi dưới dạng một hàm số theo k.

6.31*. Hoàn tất phép chứng minh bổ đề 6.2.

6.32*. Hãy đưa ra một tình huống cho thuật toán khôi phục của uỷ thác ba pha trong đó cần thực hiện nhiều vòng khôi phục rồi quyết định cuối

cùng là phải huỷ bỏ dù rằng có thể một thành viên nào đó đã chuyển sang trạng thái Ready - to - commit.

6.33. Mô tả trường hợp tương tự với khoá chốt quá bán trong phương pháp nhãn thời gian.

6.34. Giả sử rằng có ba mục A_1, A_2 và A_3 tại các vị trí tương ứng S_1, S_2 và S_3. Đồng thời có ba giao dịch T_1, T_2 và T_3 với T_i được khởi đầu từ S_i, i = 1, 2, 3. Sáu sự kiện sau lần lượt xảy ra:

T_1 khoá A_1; T_2 khoá A_2; T_3 khoá A_3;

T_1 yêu cầu khoá trên A_2 ; T_2 yêu cầu khoá trên A_3 ;

T_3 yêu cầu khoá trên A_1 .

a. Giả sử chúng ta chuyển các đồ thị chờ đợi cục bộ quanh vòng, gắn chúng vào các thông báo yêu cầu khoá. Hãy trình bày đồ thị chờ đợi toàn cục thu được mỗi vị trí sau chuỗi hành động trên. Có phát hiện được khoá gài không?

b. Những thông báo bổ sung nào (có chứa đồ thị chờ đợi cục bộ), nếu có, cần phải gửi để một vị trí phát hiện được khoá gài.

c. Giả sử chúng ta sử dụng chiến lược chờ đợi - huỷ diệt để ngăn chặn khóa gài. Hãy cho biết điều gì sẽ xảy ra nếu thời gian t_i cho T_i có thứ tự $t_1 < t_2 < t_3$.

d. Lập lại câu (c) với giả thiết $t_1 > t_2 > t_3$

e. Lập lại câu (c) cho lược đồ gây thương tổn - chờ đợi.

f. Lập lại câu (e) với giả thiết $t_1 > t_2 > t_3$.

CHƯƠNG 7

CƠ SỞ DỮ LIỆU ĐỐI TƯỢNG PHÂN TÁN

Công nghệ CSDL đã nhanh chóng chuyển sang hỗ trợ các ứng dụng mới. CSDL quan hệ đã được chứng minh là rất thành công trong việc hỗ trợ cho các ứng dụng xử lý các dữ liệu kinh doanh. Tuy nhiên hiện nay đang có một lớp ứng dụng quan trọng, thường được nhắc đến với tên gọi là *"ứng dụng CSDL nâng cao"* (advanced database appliciation), đặt ra nhu cầu cấp bách đối với việc quản lý CSDl. Thí dụ như *thiết kế các hệ thống với sự trợ giúp của máy tính* (computer - aided design viết tắt là CAD), xây dung các *hệ thông tin văn phòng* (office information system, OIS), *hệ thông tin đa môi trường* (multimedia information system) và *trí tuệ nhân tao* (artificial intelligence, AI). Đối với những ứng dụng này, các *hệ quản trị CSDL đối tượng* (object DBMS) được xem là thích hợp hơn do những đặc tính sau:

1. Những ứng dụng nâng cao này đòi hỏi phải lưu trữ và thao tác các kiểu dữ liểu trừu tượng hơn (thí dụ hình ảnh, tư liệu thiết kế) và khả năng cho phép người dùng định nghĩa các kiểu cho từng ứng dụng. Vì thế cần phải có một hệ thống kiểu phong phú có hỗ trợ các kiểu trừu tượng do người dùng định nghĩa. Trong khi đó các hệ thống quan hệ giải quyết với một kiểu đối tượng duy nhất, đó là quan hệ, với các thuộc tính lấy từ các miền giá trị đơn giản và cố định (thí dụ số, ký tự, chuỗi, ngày tháng . . .). Không có hỗ trợ nào cho việc định nghĩa và thao tác các kiểu dựa vào nhu cầu ứng dụng.

2. Mô hình quan hệ tổ chức dữ liệu theo một phương thức tương đối đơn giản và "phẳng". Biểu diễn các đối tượng ứng dụng có cấu trúc bằng mô hình quan hệ phẳng làm mất đi cấu trúc tự nhiên vốn có mà có khi rất quan trọng đối với ứng dụng. Thí dụ trong các ứng dụng thiết kế công nghệ, người ta muốn biểu diễn đối tượng xe có chứa một đối tượng động cơ. Tương tự trong các hệ thông tin đa môi trường, cần chú ý là một đối tượng siêu tài liệu chứa một đối tượng video và một đối tượng văn bản có đề mục. Mối liên hệ "hàm chứa" giữa các đối tượng không dễ gì biểu diễn được trong mô hình quan hệ nhưng lại khá đơn giản trong các mô hình đối tượng bằng các *đối tượng hợp phần* (composite object) và *đối tượng phức* (complex object) mà chúng ta sẽ xét sơ qua sau đây.

3. Các hệ thống quan hệ cung cấp một ngôn ngữ khai báo và đơn giản để truy xuất dữ liệu, đó là SQL. Vì đây không phải là một ngôn ngữ đầy đủ về khả năng tính toán, các ứng dụng CSDL phức tạp đều phải được viết bằng một ngôn ngữ lập trình tổng quát có gắn kèm những câu lệnh vấn tin. Điều này gây ra một bài toán *bất khả kháng* (impedance mismatch), nảy sinh do những khác biệt về mức độ trừu tượng hoá giữa các ngôn ngữ quan hệ và các ngôn ngữ lập trình mà chúng tương tác. Các khái niệm và kiểu của vấn tin, điển hình là lối thao tác mỗi lần một tập không phù hợp với lối hoạt tác của ngôn ngữ lập trình theo kiểu mỗi lần một mẩu tin. Trong một hệ thống đối tượng, các ứng dụng phức tạp có thể được viết hoàn toàn bằng một ngôn ngữ lập trình CSDL đối tượng duy nhất. Đương nhiên điều này có thể thay đổi khi SQL phát triển hướng về một ngôn ngữ tính toán hoàn chỉnh.

Trong phần 7.1, chúng ta sẽ thảo luận các khái niệm cơ bản về đối tượng và các vấn đề liên quan đến các mô hình hướng đối tượng. Phần 7.2 chúng ta xem xét việc thiết kế phân tán các CDSL đối tượng. Phần 7.3 dành

để thảo luận các vấn đề kiến trúc DBMS đối tượng phân tán. Trong phần 7.4 chúng ta thảo luận những vấn đề mới, nảy sinh trong việc quản lý các đối tượng và trong phần 7.5 chúng ta tập trung xem xét việc lưu trữ đối tượng. Các phần 7.6 và 7.7 được dành cho các chức năng cơ bản của DBMS: xử lý vấn tin và quản lý giao dịch. những vấn đề này tạo ra những bước ngoặt đáng chú ý khi được xem xét bên trong ngữ cảnh của công nghệ mới này.

7.1. KHÁI NIỆM CƠ BẢN VỀ CƠ SỞ DỮ LIỆU ĐỐI TƯỢNG

Hệ quản trị CSDL đối tượng là một hệ thống sử dụng "đối tượng" làm đơn vị cơ bản để mô hình hoá và truy xuất. Ngoài khẳng định khá tầm thường này, cái gì cấu tạo nên DBMS đối tượng là một đề tài được nhiều người quan tâm và thảo luận. Một số đặc tả mô hình đối tượng chuẩn đã xuất hiện như một phần của các chuẩn ngôn ngữ (thí dụ như mô hình ODMG, Object Data Managemnt Group Cattell, 1997) hoặc SQL - 3 (Melton, 1998) nhưng không được thừa nhận rộng rãi. Trong những đoạn còn lại của phần này chúng ta sẽ xem lại một số vấn đề thiết kế và các chọn lựa cho việc định nghĩa một mô hình đối tượng.

7.1.1. ĐỐI TƯỢNG

Như đã chỉ ra ở trên, tất cả các DBMS đối tượng đều được xây dựng từ các *đối tượng* (object). *Đối tượng biểu diễn một thực thể có thực trong hệ thống đang được mô hình hoá.* Đơn giản nhất, nó được biểu diễn bằng một

cặp (*OID, trạng thái*), trong đó OID là một *định danh đối tượng* (object indentifier) và *trạng thái* tương ứng *là một biểu diễn nào đó* cho trạng thái hiện tại của đối tượng.

Định danh đối tượng là một tính chất bất biến của một đối tượng, phân biệt nó với những đối tượng khác, bất kể trạng thái của nó. Điều này cho phép dùng chung đối tượng, là cơ sở để hỗ trợ các cấu trúc hợp phần và cấu trúc phức. Trong một số mô hình, sự bằng nhau của OID là một so sánh nguyên thuỷ duy nhất; còn các loại so sánh khác, người định nghĩa kiểu sẽ đặc tả ngữ nghĩa của phép so sánh đó. Trong một số mô hình , hai đối tượng được xem là *đồng nhất* (indetical) *nếu chúng có cùng OID*, và là *bằng nhau* (equal) *nếu chúng có cùng trạng thái.*

Các mô hình đối tượng thường bắt đầu phân chia từ chỗ định nghĩa về *trạng thái* (state). Trạng thái là một gía trị *nguyên tử* hoặc một giá trị được xây dựng (thí dụ là *bộ* hoặc *tập*).

Gọi D là hợp các *miền do hệ thống định nghĩa* và *miền* của các kiểu dữ liệu trừu tượng ADT (abstract data type) *do người dùng định nghĩa* (thí dụ miền các công ty).

Gọi *I* là *miền các định danh* được dùng để đặt tên các đối tượng.

Gọi A là *miền các thuộc tính tên.*

Một *giá trị* (value) được định nghĩa như sau:

1. Một phần tử của *D* là một giá trị và được gọi là *trị nguyên tử (atomic value).*

2. $[a_1: v_1, ..., a_n: v_n]$ được gọi là *trị bộ* (tuple value), trong đó a_i là một phần tử của A và v_i là một giá tri hoặc là một phần tử của I. [ ] được gọi là *toán tử dựng bộ* (tuple constructor).

3. { v_1,....., v_n} được gọi là *trị tập hợp* hoặc *trị tập* (set value), với v_i là một giá trị hoặc là một phần tử của I. {} được gọi là *toán tử dựng tập* (set constructor).

Tập và bộ là các toán tử xây dựng dữ liệu mà chúng ta xem như có vai trò chủ yếu trong các ứng dụng CSDL. Những toán tử xây dựng khác, ví dụ như *danh sách* hoặc *mảng*, cũng được đưa vào để làm tăng khả năng của mô hình.

Thí dụ 7.1:

Xét các đối tượng sau

(i_1,123)

(i_2,S10)

(i_3,{i_6, i_{11}})

(i_4,{1,3,5})

(i_5, [LF: i_7, RF:i_8, LR: i_9,RR :i_{10}])

Các đối tượng $i_1$và i_2 là các đối tượng nguyên tử, còn i_3 và i_4 là các đối tượng được xây dựng. i_3, i_4 là OID của một đối tượng với trạng thái của nó là một tập hợp. Khác biệt giữa hai đối tượng này ở chỗ trạng thái của i_4 gồm có một tập các giá trị còn của i_3 gồm một tập các OID. Vì thế đối tượng i_3 tham chiếu đến những đối tượng khác. Khi *xem định danh đối tượng là gía trị trong mô hình đối tượng, chúng ta có thể xây dựng được các đối tượng phức tạp tuỳ ý*. Đối tượng i_5 có một trạng thái nhận trị bộ gồm bốn thuộc tính (hoặc các biến thể hiện), gía trị của mỗi thuộc tính này là một đối tượng khác.

Ngược lại với giá tri, các đối tượng hỗ trợ một thao tác cập nhật chuẩn, cho phép thay đổi trạng thái đối tượng mà không làm thay đổi định

danh đối tượng. Điều này tương tự như các cập nhật trong các ngôn ngữ lập trình theo lệnh, trong đó các định danh đối tượng được cài đặt bằng các con trỏ trong bộ nhớ chính. Tuy nhiên, định danh còn tổng quát hơn con trỏ, theo nghĩa là nó tồn tại cả khi chương trình đã chấm dứt. Một điều suy ra từ định danh đối tượng là chúng ta có thể dùng chung các đối tượng để bớt rắc rối về dư thừa dữ liệu.

Thí dụ 7.2:

Xét các đối tượng sau:

(i_1, Volvo)

(i_2,[name: Jon, mycar: i_1])

(i_3 ,[name: Mary, mycar:i_1])

John và Mary dùng chung đối tượng ký hiệu là *mycar* (cả hai đều sở hữu xe Volvo). Khi thay đổi giá trị của đối tượng i_1 từ "Volvo" sang "Chevrolet", cả hai đối tượng i_2 và i_3 đều tự đông thay đổi mycar của mình.

Qua thí dụ 7.2 cho thấy cấu trúc của một mô hình - trạng thái được biểu diễn như một tập các *biến thể hiện* (instance varible) hoặc *thuộc tính* (attribute) mà thực sự chúng là các giá trị.

Hành vi của mô hình được biểu hiện qua các *phương thức* (method). *Phương thức* định nghĩa các hành động được phép trên những đối tượng và được dùng để thao tác những đối tượng đó. Phương thức biểu diễn mặt hành vi của mô hình vì chúng định nghĩa các hành vi hợp lệ mà đối tượng có thể thừa nhận. Một thí dụ điển hình là hành vi của một thang máy. Nếu chỉ có hai phương thức được định nghĩa trên đối tượng "thang máy" là "đi lên" và "đi xuống", chúng ta đinh nghĩa *hành vi của đối tượng thang máy*: nó có thể đi lên và đi xuống nhưng không thể đi ngang được.

Các mô hình đối tượng đi theo lối tiếp cận hành vi không có biến thể hiện hoặc phương thức nào; chỉ có các hành vi được áp dụng cho các đối tượng để thu được một kết quả nào đó. Các hành vi có thể được cài hoặc bằng các *hàm tồn trữ* (stored function) hoặc bằng các *hàm đã được tính* nhưng phân biệt này không được định nghĩa bằng các kết quả trả về khi áp dụng các hành vi này.

Một số mô hình đối tượng loại bỏ sự phân biệt giữa giá trị và đối tượng. Trong các mô hình đơn giản như thế, mỗi vật là một đối tượng, kể cả các thực thể hệ thống và thực thể do người sử dụng định nghĩa. Trong những hệ thống đó, không có toán tử xây dựng (tập hợp hoặc bộ); đối tượng tạo ra bằng cách chuyên biệt hoá hoặc tổng quát hoá các đối tượng khác.

Một phân biệt quan trọng *giữa mô hình quan hệ và mô hình đối tượng* là quan hệ giải quyết với giá trị dữ liệu theo một lối thống nhất, dữ liệu được nhận diện bởi giá trị. Một quan hệ được xác định bởi một tên, và một bộ được nhận diện bởi tập giá trị. Ngược lại, trong các mô hình đối tượng, dữ liệu được nhận diện bởi OID của nó. Phân biệt này rất quan trọng; mô hình hoá các mối quan hệ giữa dữ liệu dẫn đến dư thừa dữ liệu hoặc đưa ra khái niệm khoá ngoại trong mô hình quan hệ. Quản lý tự động các khoá ngoại đòi hỏi phải có sự hỗ trợ của ràng buộc toàn vẹn (toàn vẹn tham chiếu).

Thí dụ 7.3:

Xét lại thí dụ 7.2 Trong mô hình quan hệ, để đạt được cùng mục đích, người ta phải đặt gía trị của thuộc tính mycar là " Volvo". Đòi hỏi cả hai bộ đều phải cập nhật khi nó đổi thành "Chevrolet". Để làm giảm đi dư

thừa , người ta vẫn biểu diễn i_1 là một bộ trong quan hệ kia và tham chiếu đến đó từ i_2 và i_3 bằng cách dùng khoá ngoại. Cần nhớ rằng đây là cơ sở của các dạng chuẩn 3NF và BCNF. Trong trường hợp này, việc loại bỏ dư thừa trong mô hình quan hệ đòi hỏi phải chuẩn hoá các quan hệ. Tuy nhiên i_1 có thể là một đối tượng có cấu trúc mà biểu diễn nó bằng một quan hệ chuẩn có thể bất tiện. Khi đó chúng ta không thể gán nó làm giá trị của thuộc tính *mycar* ngay cả nếu chúng ta chấp nhận dư thừa vì mô hình quan hệ đòi hỏi rằng các giá trị thuộc tính phải là nguyên tử.

7.1.2. KIỂU DỮ LIỆU TRỪU TƯỢNG

Kiểu dữ liệu trừu tượng ADT (abstract data type) từ lâu đã được dùng trong các ngôn ngữ lập trình và gần đây trong các CSDL quan hệ. Giờ đây kiểu dữ liệu trừu tượng ta hiểu như *kiểu dữ liệu đối tượng- mô tả các đối tượng*.Kiểu dữ liệu đối tượng gắn với các hệ quản trị CSDL quan hệ - đối tượng. Kiểu dữ liệu trừu tượng ADT (thường được gọi đơn giản là *kiểu*) là một khuôn mẫu cho tất cả các đối tượng thuộc kiểu đó. Trong trường hợp này chúng ta không phân biệt giữa các đối tượng hệ thống (nghiã là các giá trị), các đối tượng cấu trúc (bộ hoặc tập) và các đối tượng do người dùng định nghĩa. Một ADT mô tả kiểu của dữ liệu bằng cách cung cấp một miền dữ liệu với vùng cấu trúc cũng như các thao tác (cũng gọi là phương thức) áp dụng được cho các phần tử của miền đó. Nói chung các thao tác ADT chuẩn chẳng hạn như phép chuyển đổi cho nguyên liệu - thành phẩm là bắt buộc. Người sử dụng ADT chỉ thấy cấu trúc dữ liệu giao diện và tên các thao tác giao diện với các kiểu nguyên liệu và thành phẩm đi kèm.

Thí dụ 7.4:

Trong chương này chúng ta sẽ dùng một thí dụ để minh hoạ sức mạnh của các mô hình đối tượng. Chúng ta sẽ mô hình hoá một chiếc xe hơi với nhiều bộ phận (động cơ, giảm sóc, vỏ xe) và sẽ lưu những thông tin khác như hãng sản xuất, năm sản xuất, số xe ri, . . . với bất kỳ một cú pháp cụ thể nào:

```
type Car
                attributes
                engine: Engine
                bumpers: {Bumper}
                tires: [LP: Tire, RF:Tire, LR: Tire, RR:Tire]
                make: Manufacturer
                model: String
                year: Data
                serial no: String
                capacity: Integer
        methods
                age: Real
```

Định nghĩa kiểu đặc tả rằng Car có tám thuộc tính và một phương thức. Bốn thuộc tính (model, year, serial no, capacity) là những thuộc tính dựa trên giá trị còn những thuộc tính khác (engine, bumpers và make) dựa trên đối tượng (nghĩa là chúng dùng đối tượng khác làm giá trị của chúng). Thuộc tính bumpers nhận trị tập (nghĩa là dùng toán tử dựng tập) và thuộc tính tires nhận trị bộ trong đó vỏ trước trái LF (left front), trước phải RF (right front), sau trái LR (left rear) và sau phải (right rear) được xác định riêng rẽ. Chúng ta đã sử dụng ký pháp trong đó các thuộc tính được viết

thường và kiểu được viết hoa. Vì thế engine là một thuộc tính còn Engine là một kiểu của hệ thống.

Một điểm nữa cần được nói đến trong thí dụ này. Phương thức age nhận ngày hệ thống và giá trị thuộc tính year rồi tính ra ngày tháng. Tuy nhiên vì cả hai điều này đều có tính chất nội tại của đối tượng, chúng không được trình bày trong định nghĩa kiểu , là giao diện đối với người sử dụng.

Cấu trúc dữ liệu giao diện của ADT có thể phức tạp và lớn tuỳ ý. Thí dụ ADT car có một thao tác age lấy ngày hiện tại và sản xuất của xe, tính ra tuổi của xe. Một mô hình đối tượng với ADT cho phép mô hình hoá đồng thời cả dữ liệu lẫn các thao tác. Tuy nhiên điều này không nói rằng các thao tác được lưu chung với dữ liệu. Chúng có thể được lưu trong một thư viện, tương tự như thư mục dữ liệu.

7.1.3 HỢP PHẦN

Trong những thí dụ chúng ta đã thảo luận cho đến lúc này một số biến thể hiện dựa trên giá tri, chẳng hạn như model và year trong thí dụ 7.3, còn những biến khác dựa trên đối tượng, chẳng hạn thuộc tính make với miền của nó là tập các đối tượng có kiểu manufacturer. Trong trường hợp này, kiểu Car là một kiểu *hợp phần (composite type)* và các thể hiện của nó được gọi là các *đối tượng hợp phần (composite object)*. Hợp phần (composition, aggregation) là một trong những đặc trưng mạnh của các mô hình đối tượng. Nó cho phép chia sẻ các đối tượng, thường hay gọi là *chia sẻ tham chiếu (referential sharing)* vì các đối tượng "tham chiếu" đến các đối tượng khác bằng OID giống như giá trị của các thuộc tính dựa trên đối tượng.

Thí dụ 7.5:

Chúng ta hãy sửa lại thí dụ 7.3 như sau. Giả sử c_1 là một thể hiện của kiểu car được định nghĩa trong thí dụ 7.3. Nếu những điều sau đúng:

(i_2, [name;John, mycar:c_1])

(i_3, [name: Mary, mycar:c_1])

Thì điều này chỉ ra rằng John và Mary có chung một chiếc xe.

Mối liên hệ đối tượng hợp phần giữa các kiểu có thể được biểu diễn bằng *đồ thị hợp phần (compsition graph)* hoặc bằng *phân cấp hợp phần (aggregation hierarchy)*.

7.1.4 LỚP

Phần lớn các hệ DBMS đối tượng hiện tại không phân biệt giữa kiểu và lớp. Đây là hệ quả của việc thừa nhận hệ thống kiểu của các ngôn ngữ lập trình đối tượng như Small- talk và C_{++} do chúng chỉ hỗ trợ khái niệm lớp (class). Trong những hệ thống đó, một lớp vừa biểu diễn *khuôn* (template) của tất cả các đối tượng chung (nghĩa là được dùng như một kiểu) vừa biểu diễn nhóm các đối tượng chung (nghĩa là *dòng tộc*, extent). Trong trường hợp này, lược đồ CSDL gồm một tập các định nghĩa lớp với các mối liên hệ giữa chúng.

Có sự khác biệt về khái niệm giữa kiểu và lớp. *Kiểu là một khuôn cho tất cả các đối tượng thuộc kiểu đó*, còn *lớp là một nhóm các thể hiện đối tượng thuộc kiểu đã cho*. Theo một nghĩa nào đó, *kiểu tương ứng với một lược đồ quan hệ* trong CSDL quan hệ, còn *lớp tương ứng với một thể hiện*

quan hệ cụ thể. Nói cách khác, một lớp có một *dòng tộc* (extent), đó là tập hợp tất cả các đối tượng có cùng kiểu đi kèm với lớp đó.

7.1.5 TẬP THỂ

Tập thể (collection) là một *nhóm các đối tượng do người dùng định nghĩa*. Các hệ thống kinh điển nhất đều cung cấp *kết cấu lớp* (class construct) hoặc *kết cấu tập thể* (collection construct). Tập thể cung cấp một ngữ nghĩa bao đóng rõ ràng cho các mô hình vấn tin và tạo thuận lợi cho việc định nghĩa các khung nhìn của người dùng.

Trong các hệ thống có hỗ trợ cả lớp và tập thể, một tập thể tương tự như một lớp ở chỗ nó nhóm các đối tượng lại nhưng khác nhau ở những phương diện sau đây. Trước tiên là việc tạo ra đối tượng không thể xảy ra qua tập thể: tạo đối tượng chỉ xảy ra thông qua các lớp. Điều này muốn nói rằng tập thể chỉ tạo ra các nhóm do người dùng định nghĩa từ các đối tượng đã có sẵn. Thứ hai, một đối tượng có thể tồn tại trong nhiều tập thể, nhưng chỉ là một phần tử của một lớp duy nhất. Thứ ba, việc quản lý các lớp là ngầm định ở chỗ hệ thống tự động duy trì các lớp dựa trên *dàn kiểu* (type lattice), trong khi đó việc quản lý các tập thể phải *tường minh*, với nghĩa là người dùng chịu trách nhiệm về các *dòng tộc* (extent) của chúng. Cuối cùng, một lớp gộp toàn bộ các mở rộng của một kiểu duy nhất (dòng tộc gần, shallow extent) với các mở rộng của các kiểu con của nó (*dòng tộc xa*, deep extent) vì thế phần tử của một lớp là cùng chung về phương diện *sinh kiểu con* (subtyping). Một tập thể là đa chủng ở chỗ nó có thể chứa các đối tượng thuộc các kiểu không có liên quan với nhau qua hình thức sinh kiểu con.

7.1.6 KIỂU CON VÀ KẾ THỪA

Các hệ đối tượng cung cấp tính mở rộng bằng cách cho phép người dùng định nghĩa các kiểu và hệ thống sẽ quản lý. Điều này được thực hiện bằng hai cách *nhờ dùng toán tử dựng kiểu* hoặc bằng việc *định nghĩa dựa vào các kiểu nguyên thuỷ có sẵn* qua quá trình *sinh kiểu con* (subtyping). Sinh kiểu con dựa trên mối quan hệ *chuyên biệt hoá* (specialization relationship) giữa các kiểu. Một kiểu A là một *chuyên biệt* (specialization) của một kiểu B nếu giao diện của nó là một tập bao hàm trong giao diện của B. Vì thế một kiểu chuyên biệt được định nghĩa chi tiết hơn (hoặc cụ thể hơn) so với kiểu mà từ đó nó được chuyên biệt. Một kiểu có thể là chuyên biệt của một số kiểu; nó được chuyên biệt tường minh là *kiểu con* (subtype) của một tập con của chúng. Trong thí dụ , A is-a B (A là một B), tạo ra *tính khả thế* (substitutability) là một thể hiện của một kiểu con (A) có thể được thay cho một thể hiện của một trong các kiểu cha (B) trong một biểu thức bất kỳ.

Ngoài việc cho phép khả năng mở rộng, sinh kiểu con cũng đưa đến một hệ thống kiểu cấu tạo nên lược đồ CSDL.

Một dàn kiểu thiết lập một lược đồ CSDL trong các CSDL đối tượng. Nó cho phép chúng ta mô hình hóa các tính chất chung và kiểu khác biệt giữa các kiểu bằng một cách chính xác.

Thí dụ 7.6:

Xét kiểu Car đã được định nghĩa trước kia. Một xe hơi có thể được mô hình hoá là một kiểu đặc biệt của loại đối tượng chuyển động có động cơ (Vehicle). Như vậy chúng ta có thể định nghĩa Car là một kiểu con của Vehicle; các kiểu con khác có thể là xe máy (Motocycle), xe tải (Truck), và xe buýt (Bus). Trong trường hợp này, Vehicle sẽ định nghĩa những tính chất chung của tất cả các kiểu này:

```
type  Vehicle as Object
attributes
        engine: Engine
        make: Manufacturer
        model: String
        year: Data
        serial no: String
methods
        age: Real
```

Vehicle được định nghĩa là kiểu con của Object mà chúng ta giả thiết là gốc của dàn kiểu. Nó được định nghĩa với năm thuộc tính và một phương thức, nhận ngày sản xuất và ngày hiện tại (cả hai đều thuộc kiểu Date do hệ thống định nghĩa) và trả về một giá trị số thực. Hiển nhiên Vehicle là một tổng quát hoá của Car mà chúng ta định nghĩa trong thí dụ 7.3. Bây giờ có thể định nghĩa Car như sau:

```
type   Car as Vehicle
attributes
        bumpers: {Bumper}
        tires: [LP: Tire, RF:Tire, LR: Tire, RR:Tire]
```

capacity: Integer

Mặc dù Car được định nghĩa chỉ với hai thuộc tính, giao diện của nó giống như định nghĩa đã được cho trong thí du 7.3. Điều này là do Car is Vehicle, và vì vậy nó kế thừa các thuộc tính và phương thức của Vehicle.

Khai báo rằng một kiểu là kiểu con của một kiểu khác tạo ra *sự kế thừa* (inkerit- ance). Một kiểu con có thể kế thừa hành vi của kiểu cha hoặc kế thừa cài đặt hoặc kế thừa cả hai. Chúng ta nói đến đơn kế thừa và đa kế thừa dựa trên mối liên hệ kiểu con giữa các kiểu.

7.2. THIẾT KẾ PHÂN TÁN ĐỐI TƯỢNG

Nhớ lại trong Chương 4, hai khía cạnh quan trọng của việc thiết kế phân tán là *phân mảnh và cấp phát*. Thiết kế phân tán trong thế giới đối tượng làm nảy sinh những vấn đề phức tạp mới. Về mặt khái niệm, các đối tượng bao gồm các phương thức và các trạng thái. Trong thực tế, phương thức được cài đặt trên các kiểu và được chia sẻ bởi tất cả các đối tượng thể hiện của kiểu đó.

Trong phần này chúng ta sẽ xét điểm tương đồng của CSDL đối tượng với bài toán thiết kế phân tán đã giới thiệu trong Chương 4 bằng cách xem xét việc phân mảnh và cấp phát trong ngữ cảnh mô hình đối tượng. Thiết kế phân tán trong thế giới đối tượng dẫn đến những vấn đề phức tạp do việc bao gói các phương thức cùng với trạng thái đối tượng. Điều này dẫn đến nhiều rắc rối bởi vì các phương thức được cài đặt trên kiểu và được dùng chung bởi tất cả các đối tượng thể hiện của kiểu đó. Vì vậy người ta phải quyết định xem có nên phân mảnh trên các thuộc tính và nhân các phương

thức ra cho mỗi mảnh hay phân mảnh luôn cả các phương thức . Vị trí các đối tượng ứng với kiểu của chúng và kiểu các thuộc tính trở thành một vấn đề cần xem xét. Như đã thảo luận trong phần 7.1, miền các thuộc tính có thể là các lớp khác, nên phân mảnh các lớp ứng với một thuộc tính như thế có thể ảnh hưởng đến những lớp khác. Cuối cùng nếu phân mảnh cũng được thực hiện ứng với các phương thức, chúng ta cần phân biệt giữa các phương thức giản đơn và các phương thức phức. Các phương thức giản đơn là những phương thức không kích hoạt những phương thức khác, còn những phương thức phức có thể kích hoạt phương thức của những lớp khác.

Tương tự với trường hợp quan hệ, chúng ta có 3 kiểu phân mảnh cơ bản: phân mảnh ngang, dọc và kết hợp. Ngoài những trường hợp cơ bản này, người ta cũng định nghĩa *phân hoạch ngang dẫn xuất* (dereved horizontal partitioning), *phân hoạch ngang kết hợp* (associated horizontal partitioning), và *phân hoạch đường dẫn* (path partitioning). Phân hoạch ngang dẫn xuất có ngữ nghĩa tương tự như đối tác của chúng trong CSDL quan hệ. Phân hoạch ngang kết hợp tương tự như phân hoạch ngang dẫn xuất chỉ trừ không có "mệnh đề vị trí" ràng buộc các thể hiện đối tượng. Trong những phần còn lại, để cho đơn giản chúng ta sẽ thừa nhận mô hình đối tượng dựa trên lớp và nó không phân biệt giữa kiểu và lớp.

7.2.1 PHÂN HOẠCH NGANG LỚP

Có nhiều điểm giống nhau giữa phân mảnh ngang các CSDL đối tượng và đối tác quan hệ của chúng. Chúng ta có thể xác định phân mảnh ngang nguyên thuỷ trong trường hợp CSDL đối tượng giống như trường hợp quan hệ. Tuy nhiên phân mảnh dẫn xuất cho thấy có những khác biệt.

Trong các CSDL đối tượng, phân mảnh ngang dẫn suất có thể xẩy ra một số cách:

1. Phân hoạch một lớp nảy sinh từ phân mảnh các lớp con của nó. Điều này xảy ra khi một lớp chuyên biệt hơn được phân mảnh, vì thế kết quả của phân mảnh này phải được phản ánh trong trường hợp tổng quát hơn. Ở đây chúng ta cần phải lưu ý là phân mảnh theo một lớp con có thể gây xung đột với phân mảnh theo những lớp con khác. Vì vậy, người ta sẽ bắt đầu phân mảnh lớp chuyên biệt nhất và di chuyển dần lên trong giàn lớp.

2. Phân mảnh của một thuộc tính phức hợp có thể ảnh hưởng đến việc phân mảnh lớp chứa nó.

3. Phân mảnh của một lớp dựa trên một dãy kích hoạt phương thức từ một lớp đến một lớp khác có thể cần được phản ánh trong thiết kế. Điều này xảy ra trong trường hợp các phương thức được định nghĩa ở trên.

Chúng ta bắt đầu xét trường hợp đơn giản nhất. đó là phân mảnh một lớp có các thuộc tính và phương thức đơn giản. Trong trường hợp này, phân hoạch ngang nguyên thuỷ có thể được thực hiện theo một vị từ được định nghĩa trên các thuộc tính của lớp. Phân hoạch khi đó hoàn toàn dễ dàng: cho trước lớp C cần phân hoạch, chúng ta tạo ra các lớp C_1,....., C_n, mỗi lớp nhận các thể hiện của C thoả vị từ phân hoạch cụ thể. Nếu những vị từ này độc quyền tương hỗ, thì các lớp C_1,....., C_n là tách biệt. Trong trường hợp này, chúng ta có thể định nghĩa C_1,....., C_n như các lớp con của C và sửa lại định nghĩa của C thành một *lớp trừu tượng* - là lớp không có một *dòng tộc* (extent) tường minh (nghĩa là không có thể hiện nào của riêng nó).

Sự phức tạp nảy sinh nếu vị từ phân hoạch không độc quyền tương hỗ. Một số mô hình đối tượng cho phép mỗi đối tượng thuộc về nhiều lớp. Nếu không, chúng ta cần định nghĩa các "lớp gối chồng" để giữ các đối tượng thoả nhiều vị từ.

Thí dụ 7.7:

Xét định nghĩa của lớp Engine trong thí dụ 7.6:

```
Class Engine as Object
    attributes
        no-cylinder: Integer
        capacity: Real
        horsepower: Integer
```

Trong định nghĩa đơn giản của Engine, tất cả mọi thuộc tính đều thuộc loại giản đơn. Xét các vị từ phân hoạch

p_1: horsepower $\leq$ 150

P_2: horsepower > 150

Trong trường hợp này, Engine có thể được phân hoạch thành hai lớp Enginel và Engine2. Chúng kế thừa tất cả các thuộc tính của lớp Engine. Các đối tượng của lớp Engine được phân phối cho các lớp Engine1 và Engine2 dựa trên giá trị của thuộc tính horepower (mã lực) của chúng.

Phân mảnh ngang nguyên thuỷ các lớp được sử dụng cho tất cả các lớp cần phải phân mảnh trong hệ thống.Vào cuối quá trình này, chúng ta thu được các lược đồ phân mảnh trong cho mỗi lớp. Tuy nhiên những lược đồ này không phản ánh tác dụng của phân mảnh cho mỗi lớp. Vì vậy bước tiếp theo là tạo ra một tập các mảnh dẫn xuất cho mỗi lớp cha. Thành phẩm của bước này là tập các mảnh nguyên thuỷ được tạo ra trong bước hai và tập các mảnh dẫn xuất từ bước ba.

Bước cuối cùng là tổ hợp hai tập mảnh này bằng một cách nhất quán. Các mảnh ngang cuối cùng của một lớp cấu tạo bởi các đối tượng được truy xuất bởi cả những ứng dụng này chỉ chạy trên một lớp con của nó. Vì vậy chúng ta phải xác định mảnh nguyên thuỷ thích hợp nhất để trộn với mỗi

mảnh dẫn xuất của mỗi lớp. Chúng ta có thể dùng nhiều heuristic đơn giản, chẳng hạn chọn mảnh nguyên thuỷ lớn nhất hoặc nhỏ nhất, hoặc mảnh nguyên thuỷ gối chồng nhiều nhất với mảnh dẫn xuất, nhưng những heuristic này dẫu đơn giản và trực quan, chúng chỉ nắm bắt các thông tin định lượng về CSDL đối tượng phân tán. Vì vậy một phương pháp chính xác hơn đã được phát triển dựa trên *số đo tụ lực* (affinity measure) giữa các mảnh. Kết quả là các mảnh được nối lại với các mảnh có tụ lực với chúng là cao nhất.

Xét phân hoạch ngang của một lớp với các biến thể hiện dựa trên đối tượng (nghĩa là miền biến thiên của một số biến thể hiện là một lớp khác), nhưng các phương thức đều giản đơn. Trong trường hợp này, mối *liên hệ hợp phần* (composition relationship) giữa các lớp rất có tác dụng. Theo một nghĩa nào đó, mối liên hệ này thiết lập mối liên hệ chủ nhân - thành viên đã thảo luận trong Chương 4, Nếu lớp C_1 có một thuộc tính A_1 với miền biến thiên của nó là lớp C_2, thì C_1 là chủ nhân và C_2 là thành viên. Vì vậy phân rã của C_1 tuân theo nguyên tắc giống như phân mảnh ngang dẫn xuất đã thảo luận trong Chương 4.

7.2.2 PHÂN HOẠCH DỌC LỚP

Phân mảnh dọc phức tạp hơn nhiều. Cho trước một lớp C, *phân mảnh dọc* thành C_1,..., C_m sinh ra một số lớp, *mỗi lớp chứa một số thuộc tính và một số phương thức*. Vì vậy mỗi mảnh sẽ được định nghĩa ít hơn so với ban đầu. Các vấn đề cần giải quyết gồm có mối liên hệ sinh kiểu con giữa các lớp cha, các lớp con của lớp ban đầu và các lớp phân mảnh, mối liên hệ của

các lớp phân mảnh với nhau, và vị trí các phương thức. Nếu tất cả các phương thức đều đơn giản thì chúng có thể được phân hoạch dễ dàng.

Các phương pháp phân mảnh dọc quan hệ dựa trên ái lực đã được phát triển cho các CSDL đối tượng. Tuy nhiên tháo bỏ việc bao gói trong khi phân mảnh dọc trong các DBMS đối tượng. Độc giả nên xem thêm các tài liệu tham khảo để biết thêm thông tin về vấn đề này.

7.2.3 PHÂN HOẠCH ĐƯỜNG DẪN

Đồ thị hợp phần trình bày một biểu diễn cho các đối tượng hợp phần. nhiều ứng dụng cần truy xuất toàn bộ đối tượng hợp phần. *Phân hoạch đường dẫn* (path partitioning) là một khái niệm mô tả việc làm tụ tất cả các đối tượng tạo ra một đối tượng hợp phần vào một phân hoạch. Một phân hoạch đường dẫn bao gồm việc nhóm các đối tượng thuộc tất cả các lớp tương ứng với tất cả các biến thể hiện trong cây con có gốc tại đối tượng hợp phần đó.

Một phân hoạch đường dẫn có thể được trình bày như một cây phân cấp với các nút tạo thành chỉ một cấu trúc. Mỗi nút của chỉ mục chỉ đến các đối tượng thuộc lớp miền của đối tượng thành phần. Vì vậy chỉ mục chứa các tham chiếu đến tất cả các đối tượng thành phần của một đối tượng hợp phần, loại bỏ nhu cầu duyệt qua cây phân cấp hợp phần. Các thể hiện của chỉ mục cấu trúc là một tập các OID chỉ đến tất cả các đối tượng thành phần của một lớp hợp phần.

7.2.4 CÁC THUẬT TOÁN PHÂN HOẠCH LỚP

Vấn đề chính của phân hoạch lớp là cải thiện hiệu năng các vấn tin và ứng dụng bằng cách làm giảm đi các truy xuất dữ liệu không cần thiết. D vậy phân hoạch lớp là một kỹ thuật thiết kế CSDL logic, trong đó nó cấu trúc lại lược đồ CSDL đối tượng dựa trên ngữ nghĩa ứng dụng. Cần chú ý rằng phân hoạch lớp phức tạp hơn nhiều so với phân mảnh quan hệ, và nó cũng là bài toán NP complete. Các thuật toán phân hoạch lớp đặt nền tảng trên cách tiếp cận ái lực và chi phí.

PHƯƠNG PHÁP DỰA VÀO TỤ LỰC CỦA CÁC THUỘC TÍNH

Như đã đề cập trong phần 4.3, lực hút giữa các thuộc tính được dùng để phân mảnh dọc các quan hệ. Tương tự, tụ lực giữa các biến thể hiện và phương thức, tụ lực giữa các phương thức có thể được dùng để phân hoạch ngang và dọc lớp. Một biến thể hiện phức là biến thể hiện dựa trên đối tượng và là bộ phận của cây phân cấp hợp phần lớp, mở rộng *thể ước lượng phân hoạch* (partition evaluator) của các CSDL quan hệ cho các CSDL hướng đối tượng, và thuật toán đề xuất sử dụng lối liệt kê vét cạn để phát triển một phương pháp để thu lược đồ phân hoạch. Nó áp dụng ngữ nghĩa phương thức và sản sinh các mảnh làm cho các phương thức phù hợp với các yêu cầu dữ liệu.

PHƯƠNG PHÁP DỰA VÀO CHI PHÍ

Mặc dù tiếp cận dựa trên các tụ lực cung cấp các lược đồ phân hoạch hấp dẫn một cách trực quan, người ta đã chứng minh rằng các lược đồ phân hoạch không phải lúc nào cũng có thể làm giảm truy xuất đĩa nhiều nhất khi

xử lý một tập các ứng dụng. Vì thế người ta đã phát triển một mô hình chi phí theo số lượng truy xuất đĩa để xử lý các vấn tin lẫn phương thức trên một CSDL hướng đối tượng. Một heuristic" leo đồi" có sử dụng cả phương pháp ái lực (cho giải pháp khởi đầu) và phương pháp chi phí (để điều chỉnh) cũng đã được đề xuất (1996) và nghiên cứu này cũng phát triển các phân cấp chỉ mục nối cấu trúc cho các truy xuất đối tượng phức hợp, và nghiên cứu tính hiệu quả của nó đối với phương pháp duyệt trỏ và những phương pháp khác, chẳng hạn như phân cấp chỉ mục nối cấu trúc, quan hệ đa chỉ mục và hỗ trợ truy xuất. Mỗi phân cấp chỉ mục nối là một dạng vật chất hoá của phân mảnh đường dẫn, tạo dễ dàng cho việc truy xuất trực tiếp đến một đối tượng phức hợp và các đối tượng thành phần của nó.

7.2.5 CẤP PHÁT

Bài toán cấp phát dữ liệu trong các CSDL đối tượng gồm cả cấp phát phương thức và lớp. Bài toán cấp phát phương thức có liên quan chặt chẽ với bài toán cấp phát lớp do vấn đề bao gói. Vì vậy cấp phát các lớp kéo theo cấp phát phương thức cho các lớp chủ tương ứng của chúng. Việc cấp phát các phương thức cần truy xuất nhiều lớp nằm ở những vị chí khác nhau lại là một bài toán chưa được giải quyết. Bốn chọn lựa được đưa ra (Fang et al., 1994).

A) HÀNH VI CỤC BỘ - ĐỐI TƯỢNG CỤC BỘ

Đây là trường hợp tầm thường nhất và được đưa ra để tạo ra trường hợp cơ bản. Hành vi, đối tượng mà nó áp dụng, tất cả đều ở chung một chỗ. Vì thế không cần đến một cơ chế đặc biệt nào để xử lý trường hợp này.

B) HÀNH VI CỤC BỘ - ĐỐI TƯỢNG Ở XA

Đây là một trong những trường hợp hành vi và đối tượng mà nó áp dụng nằm tại những vị trí khác nhau. Có hai cách giải quyết trường hợp này. Một giải pháp là di chuyển đối tượng ở xa về vị trí có chứa hành vi. Giải pháp thứ hai là di chuyển cài đặt hành vi về vị trí của đối tượng. Điều này có thể làm được nếu vị trí nhận có thể cho chương trình chạy được.

C) HÀNH VI Ở XA- ĐỐI TƯỢNG CỤC BỘ

Đây là trường hợp đảo ngược của trường hợp B).

D) HÀNH VI Ở XA- ĐỐI TƯỢNG Ở XA

Đây là trường hợp ngược lại với (A).

Các thuật toán tụ lực để cấp phát tĩnh các mảnh có dùng đến một kỹ thuật phân hoạch đồ thị.Tuy nhiên, những thuật toán này không tập trungvào cấp phát phương thức và không xét đến sự phụ thuộc qua lại giữa các phương thức và lớp. Vấn đề này đã được xem xét bằng phương tiện của một giải pháp lặp cho việc cấp pháp phương thức của lớp.

7.2.6 NHÂN BẢN

Nhân bản (copy) gây thêm một khó khăn mới cho bài toán thiết kế. Đối tượng, lớp các đối tượng hoặc tập thể các đối tượng (hoặc tất cả) đều có thể là các đơn vị nhân bản. Quyết định này ít nhất cũng phụ thuộc một phần vào mô hình đối tượng. Đặc tả kiểu có được đặt tại một vị trí hay không đều có thể được xem là một bài toán nhân bản.

7.3. CÁC KIỂU KIẾN TRÚC

Như ta đã biết, một phương pháp phát triển một hệ phân tán là phương pháp client/server (khách/chủ hoặc khách/đại lý). Phần lớn, nếu không phải là tất cả các hệ quản trị CSDL đối tượng hiện nay đều là các hệ khách/chủ. Các vấn đề có liên quan đến những hệ thống này thường phức tạp hơn do những đặc trưng của mô hình đối tượng. Một số điều cần xem xét được liệt kê dưới đây:

(1) Vì dữ liệu và thủ tục được bao gói thành các đối tượng, đơn vị chuyển giao giữa khách và chủ là một vấn đề phải đặt ra, vì đơn vị này có thể là một trang, một đối tượng hoặc một nhóm đối tượng.

(2) Quan hệ mật thiết với vấn đề trên là quyết định thiết kế có liên quan đến các hàm được khách và chủ cung cấp. Điều này đặc biệt quan trọng vì đối tượng không phải là những dữ liệu thụ đông, và cần phải xét đến các vị trí nơi mà các phương thức được thực thi.

(3) Trong các hệ khách/đại lý quan hệ, khách chỉ đơn giản gửi các vấn tin đến đại lý, đại lý thực hiện chúng rồi trả các bản kết quả về cho khách. Điều này gọi là *chuyển gửi chức năng* (function shipping). Trong các DBMS khách/đại lý đối tượng, đây không phải là cách tốt nhất bởi vì việc duyệt qua các cấu trúc đối tượng phức hoặc hợp phần của chương trình ứng dụng có thể chỉ ra rằng dữ liệu phải được chuyển cho bên khách (được gọi là các *hệ chuyển dữ liệu*). Vì dữ liêu được nhiều khách dùng chung, quản lý *vùng trữ* (cache) bên khách để bảo đảm nhất quán dữ liệu trở thành một việc quan trọng. Quản lý vùng trữ bên khách có liên quan mật thiết đến việc điều khiển đồng thời vì dữ liệu đã được đệm cho có thể được nhiều khách dùng chung, và điều này phải được kiểm soát. Phần lớn các DBMS đối tượng

thương mại đều dùng phương pháp khoá chốt để điều khiển đồng thời, vì vậy một vấn đề kiến trúc căn bản là việc đặt các khoá chốt, và có nên trữ các khoá chốt cho khách hay không.

(4) Vì đối tượng có thể thuộc loại hợp phần hoặc phức, có khả năng phải gửi trước các đối tượng thành phần khi một đối tượng được yêu cầu. Các hệ khách/ đại lý quan hệ thường không phải gửi trước dữ liệu, nhưng đây có thể là một giải pháp có thể chọn trong trường hợp các DBMS đối tượng.

7.3.1 CÁC CLIENT/SERVER

Hai loại kiến trúc client/server đã được đề xuất: *server đối tượng* (object server) và *server trang* (page server). Phân biệt này một phần dựa trên *độ mịn* (granularity) của dữ liệu được chuyển đi giữa khách và đại lý, và một phần vào chức năng được cung cấp cho khách và đại lý.

Đại lý đối tượng là khách đưa yêu cầu về các "đối tượng" cho đại lý, và đại lý truy tìm chúng từ CSDL rồi trả chúng về cho khách có yêu cầu. Những hệ thống này được gọi là *đại lý đối tượng*. Bộ quản lý đối tượng cung cấp một số chức năng, thí dụ như thực thi các phương thức ở bên khách có thể kích hoạt thực thi các phương thức không được chuyển đến đại lý cùng với đối tượng, giải quyết việc cài đặt các định danh đối tượng (xoá thực sự hoặc dọn rác). Ở đại lý, nó cũng hỗ trợ cho việc làm tụ đối tượng ở bên khách và bên đại lý cài đặt một *vùng trữ đối tượng* (cache) (ngoài vùng trữ trang ở đại lý), . . .

Kiến trúc đại lý trang, trong đó đơn vị chuyển tải giữa đại lý và khách là một đơn vị dữ liệu vật lý, chẳng hạn một trang hoặc một segment

chứ không phải một đối tượng. Kiến trúc đại lý trang tách rời các dịch vụ xử lý đối tượng giữa khách và đại lý. Thực sự các đại lý không giải quyết với các đối tượng nữa mà thay vào đó nó sẽ hoạt động như các bộ quản lý lưu trữ "thêm trị" (value - added).

Các nghiên cứu ban đầu về tính kết quả thường dùng các đại lý kiến trúc trang hơn là kiến trúc đại lý đối tượng. Thực sự những kết quả này đã tác động đến toàn bộ những nghiên cứu hướng vào việc thiết kế tối ưu cho các DBMS đối tượng dựa trên đại lý trang. Tuy nhiên những kết quả này không mang tính chính xác quyết định vì chúng đã chỉ ra rằng kiến trúc đại lý trang sẽ tốt hơn khi có sự tương hợp giữa *kiểu làm tụ dữ kiệu* (clustering pattern) và *kiểu truy xuất* (access pattern) của người dùng. Những nghiên cứu đầu tiên này còn bị giới hạn nhiều hơn qua việc chúng chỉ xem xét ở các môi trường một khách/một đại lý và nhiều khách/ một đại lý. Vậy phải tiếp tục nghiên cứu.

Một khả năng giải quyết với bài toán xét duyệt là chuyển tải mã lệnh của ứng dụng đến đại lý và thực thi nó ở đó. Tuy nhiên, điều này cần phải cẩn thận vì mã lệnh không thể xem là an toàn và có thể làm tổn hại đến sự an toàn và độ khả tín của DBMS. Một số hệ thống dùng một ngôn ngữ an toàn để giải quyết bài toán này. Hơn nữa, vì việc thực thi được chia giữa khách và đại lý, dữ liệu sẽ nằm ở cả vùng trữ đại lý và vùng trữ khách, và sự nhất quán của nó trở thành một vấn đề cần xem xét. Tuy nhiên, cách tiếp cận "chuyển gửi chức năng" hàm chứa cả khách và đại lý trong việc thực thi một vấn tin/ ứng dụng phải được xét để giải quyết với các tải trọng được trộn chung này. Phân phối công việc thực thi giữa các máy khác nhau cũng phải được điều chỉnh khi hệ thống chuyển hướng sang các kiến trúc ngang hàng.

Đại lý trang làm đơn giản mã lệnh của DBMS vì cả đại lý và khách

đều duy trì vùng nhớ trang, và dạng biểu diễn một đối tượng đều như nhau trên suốt quá trình đi từ đĩa đến giao diện người dùng. Vì vậy các cập nhật trên các đối tượng chỉ xảy ra trong các vùng đệm khách và những cập nhật này được phản ánh vào đĩa khi trang đó được đẩy từ khách đến đại lý. Một ưu điểm khác của các đại lý trang là việc tận dụng đầy đủ khả năng máy trạm của khách khi thực thi các vấn tin và ứng dụng, nên ít có cơ hội làm đại lý bị ùn tắc. Đại lý thực hiện một số chức năng nhất định, nên có thể phục vụ cho một số lớn khách. Chúng ta có thể thiết kế những hệ thống này sao cho sự phân phối công việc giữa khách và đại lý có thể xác định bằng thể tối ưu hoá vấn tin. Đại lý trang cũng có thể tận dụng khả năng của hệ điều hành và cả phần cứng để giải quyết một số vấn đề, chẳng hạn như điều chế con trỏ vì đơn vị thao tác thống nhất một trang.

Rõ ràng cả hai kiến trúc này đều có những ưu điểm và hạn chế quan trọng.

7.3.2 QUẢN LÝ VÙNG TRỮ BÊN CLIENT

Bên khách có thể quản lý một vùng trữ trang, một vùng trữ đối tượng hoặc một vùng trữ kép (nghĩa là cả trang và đối tượng). Nếu bên khách có một vùng trữ trang thì toàn bộ các trang được đọc hoặc ghi từ đại lý mỗi khi có một *khuyết trang* (page fault) xảy ra hoặc một trang được dồn vào đĩa. Vùng trữ đối tượng có thể đọc/ghi từng đối tượng và cho phép các ứng dụng truy xuất từng đối tượng một.

Vùng trữ đối tượng quản lý truy xuất ở một độ mịn hơn, vì vậy có thể có được các mức độ hoạt động đồng thời cao hơn. Tuy nhiên chúng có thể

gặp phải sự phân mảnh vùng trữ vì nó có thể không có khả năng chứa trọn một số đối tượng, và do đó để lại một ít chỗ không dùng đến. Một vùng trữ trang không gặp phải vấn đề này nhưng nếu việc làm tụ dữ liệu trên đĩa không phù hợp với kiểu mẫu truy xuất dữ liệu của ứng dụng thì các trang sẽ chứa một số lớn đối tượng không được truy xuất và chiếm nhiều chỗ của vùng trữ khách. Trong những tình huống như vậy, tiện dụng của một vùng trữ trang sẽ thấp hơn tiện dụng của một vùng trữ đối tượng.

Để có được những tiện lợi của vùng trữ trang lẫn vùng trữ đối tượng, người ta đã đề xuất vùng trữ kép trang/đối tượng (Kemper and Kossman, 1994, Castro et al.,1997). Trong một hệ thống trữ kép, bên khách tải các trang vào vùng trữ trang. Tuy nhiên khi bên khách ghi dồn một trang, nó vẫn giữ lại các đối tượng có ích từ trang đó bằng cách chép chúng vào vùng trữ đối tượng. Do đó bộ quản lý vùng trữ khách cố gắng giữ lại các trang có tụ cao và các đối tượng biệt lập ra khỏi các trang kém tụ. Bộ quản lý vùng trữ khách giữ lại các trang và đối tượng giữa ranh giới các giao dịch (thường được gọi là *trữ liên giao dịch*, intern - transaction caching). Nếu bên khách dùng một cơ chế khôi phục dựa trên nhật ký , chúng cũng quản lý một *vùng trữ nhật ký nội nhớ* (in - memory log buffer) ngoài vùng trữ dữ liệu ra. Trong khi vùng trữ dữ liệu thường được quản lý bằng cách dùng biến thể của chiến lược LRU (the least recently used, gần đây ít được dùng nhất), vùng trữ nhật ký thường dùng chiến lược *vào trước/ra trước* (first - in/first - out). Giống như trong việc quản lý vùng trữ của các DBMS tập trung, điều quan trọng quyết định là xem tất cả các giao dịch khách tại một trạm có *chia vùng trữ* hay mỗi giao dịch đều *duy trì vùng trữ của riêng nó*. Xu hướng gần đây cho các hệ thống có cả hai vùng trữ dùng chung và dùng riêng.

7.3.3 QUẢN LÝ VÙNG TRỮ BÊN SERVER

Các vấn đề quản lý vùng trữ không có thay đổi gì trong các hệ khách/ đại lý đối tượng vì bên đại lý thường quản lý một vùng trữ trang. Tuy nhiên để cho đầy đủ chúng ta sẽ thảo luận vấn đề đó ở đây một cách ngắn gọn. Các trang từ vùng trữ trang, đến lượt chúng lại được gửi cho bên khách để đáp ứng yêu cầu dữ liệu của chúng. Đại lý đối tượng sẽ xây dựng các nhóm đối tượng của nó bằng cách chép các đối tượng cần thiết từ các trang đệm có liên quan trên đại lý và gửi nhóm đối tượng đó cho khách. Ngoài đệm ở mức trang, đại lý cũng duy trì một *vùng trữ đối tượng đã sửa* MOB (modified object buffer). Một MOB sẽ lưu các đối tượng đã được cập nhật và được trả về bởi khách. Những đối tượng đã được cập nhật phải được lấp vào các trang dữ liệu tương ứng của chúng và như thế chúng có thể cần các hành động *đọc lấp* đã mô tả ở trên. Cuối cùng trang đã sửa phải được ghi lại vào đĩa. MOB cho phép đại lý trả dần các chi phí xuất nhập đĩa bằng cách thực hiện các kiểu lồng các thao tác *đọc lấp* (installation read) và *ghi lấp* (installation write).

7.3.4 TÍNH NHẤT QUÁN CỦA VÙNG TRỮ

Bảo đảm nhất quán vùng trữ là một vấn đề quan trọng. Nghiên cứu về nhất quán vùng trữ có liên quan chặt chẽ với nghiên cứu điều khiển đồng thời vì dữ liệu đệm có thể đồng thời được nhiều khách truy xuất, và các khoá chốt cũng có thể được đệm cùng với dữ liệu ở bên khách. Các thuật toán

nhất quán vùng trữ có thể được phân thành loại *tránh né* và loại *phát hiện*. Các thuật toán *tránh né* (avoidance - based) ngăn các truy xuất đến dữ liệu đệm *quá cũ* (stale) bằng cách bảo đảm rằng khách không cập nhật một đối tượng nếu nó đang được đọc bởi những khách khác. Vì vậy chúng bảo đảm rằng dữ liệu quá cũ không bao giờ có mặt trong các vùng trữ khách. Các thuật toán *phát hiện* (detection - based) cho phép truy xuất dữ liệu đệm quá cũ vì khách có thể cập nhật các đối tượng đang được đọc bởi những khách khác. Tuy nhiên các thuật toán phát hiện phải thực hiện một bước kiểm tra vào lúc uỷ thác để đảm bảo tính nhất quán dữ liệu.

Các thuật toán tránh né và phát hiện có thể phân tiếp thành loại *đồng bộ* (synchronous) hoặc *trì hoãn* (deferred), tuỳ thuộc vào thời điểm chúng thông tin cho đại lý biết rằng một thao tác ghi đang được thực hiện. Trong các thuật toán đồng bộ, bên khách gửi một thông báo liên tiếp vào lúc nó muốn thực hiện một thao tác ghi và tự phong toả cho đến khi đại lý trả lời. Trong các thuật toán không đồng bộ, bên khách gửi một thông báo khoá liên tiếp vào lúc thực hiện thao tác ghi nhưng không phong toả để chờ trả lời của đại lý (nó tiếp tục lạc quan). Trong các thuật toán trì hoãn, bên khách lạc quan trì hoãn việc thông tin cho đại lý biết về thao tác ghi của nó cho đến lúc uỷ thác. Trong chế độ trì hoãn, bên khách nhóm tất cả các yêu cầu khoá và gửi chúng đến đại lý vào lúc uỷ thác. Vì vậy chi phí chuyển giao sẽ thấp hơn trong lược đồ nhất quán vùng trữ nếu so với các thuật toán đồng bộ và không đồng bộ.

Dưới đây chúng ta sẽ thảo luận mỗi khả năng trong phạm vi thiết kế và bình luận về các đặc trưng của chúng.

**TRÁNH NÉ ĐỒNG BỘ.* CBL (Callback - Read Locking) là thuật toán nhất quán vùng trữ tránh né đồng bộ thông dụng nhất. Trong thuật toán này, bên khách

giữ lại các khoá đọc giữa các giao dịch, nhưng nhường lại các khoá ghi vào cuối giao dịch. Bên khách gửi yêu cầu khoá cho đại lý và chúng tự phong toả cho đến khi bên đại lý trả lời. Nếu bên khách yêu cầu một khoá ghi trên một trang đã được đệm ở chỗ các khách khác, bên đại lý đưa ra các thông báo *gọi lại* (callback), yêu cầu khách ở xa nhường các khóa đọc của chúng trên trang đó. Callback - Read bảo đảm được một tỷ lệ huỷ bỏ thấp và nói chung thực hiện tốt các thuật toán tránh né trì hoãn, phát hiện đồng bộ, và phát hiện không đồng bộ.

**TRÁNH NÉ KHÔNG ĐỒNG BỘ.* Thuật toán AACC (asynchronous avoidance – based cache consistency) không mất chi phí phong toả có hiện diện trong các thuật toán đồng bộ. Khách gửi thông báo khoá đến đại lý và tiếp tục xử lý ứng dụng. Cách tiếp cận lạc quan như cách này thường gặp phải tỷ lệ huỷ bỏ cao nhưng nó bị khử mất trong các thuật toán tránh né bởi các hành động trực tiếp của đại lý đã vô hiệu hoá các đối tượng đệm quá cũ ở chỗ các khách ở xa ngay khi hệ thống biết về cập nhật đó. Bởi vậy các thuật toán không đồng bộ có tỷ lệ huỷ bỏ do khoá gài thấp hơn so với các thuật toán tránh né trì hoãn sẽ được thảo luận tiếp .

**TRÁNH NÉ TRÌ HOÃN.* Họ thuật toán hai pha lạc quan O2PL cho việc đảm bảo nhất quán vùng trữ là các thuật toán tránh né trì hoãn. Trong những thuật toán này bên khách bó gói các yêu cầu của chúng và gửi chúng cho đại lý vào lúc uỷ thác. Đại lý sẽ phong toả khách cập nhật nếu những khách khác đang đọc các đối tượng được cập nhật. Khi mức xung đột dữ liệu tăng lên, các thuật O2PL dễ bị ảnh hưởng bởi tỷ lệ huỷ bỏ do khoá gài cao hơn so với các thuật toán CBL.

**PHÁT HIỆN ĐỒNG BỘ.* Khoá hai pha có trữ C2PL (Caching Two - Phase Locking) là một thuật toán nhất quán vùng trữ phát hiện đồng bộ. Trong thuật toán này, bên khách tiếp xúc với đại lý mỗi khi chúng truy xuất một

trang trong vùng trữ của chúng để đảm bảo trang này không quá cũ hoặc đang được ghi bởi những khách khác. Hiệu năng của C2PL nói chung kém hơn so với CBL và O2PL vì nó không trữ các khoá đọc trong các giao dịch.

*PHÁT HIỆN KHÔNG ĐỒNG BỘ. Thuật toán No - Wait Locking (NWL) with Notification là một thuật toán phát hiện không đồng bộ. Trong thuật toán này khách gửi yêu cầu khoá đến đại lý nhưng lạc quan giả thiết rằng các yêu cầu của chúng sẽ thành công. Sau khi một giao dịch khách uỷ thác, đại lý sẽ làm lan truyền trang cập nhật đến tất cả những khách cũng đã đệm các trang đó. Người ta chứng minh rằng CBL thực hiện tốt hơn thuật toán NWL.

*PHÁT HIỆN TRÌ HOÃN. AOCC (Adaptive Optimistic Concurrency Control) là một thuật toán phát hiện trì hoãn. Đã chứng minh được rằng AOCC có thể thực hiện tốt hơn thuật toán CBL ngay cả khi có một tỷ lệ huỷ cao nếu trạng thái giao dịch khách (dữ liệu và nhật ký) hoàn toàn khớp vừa vào vùng trữ khách, và tất cả mọi xử lý ứng dụng đều được thực hiện nghiêm ngặt ở chỗ khách (Kiến trúc chuyển gửi dữ liệu đơn thuần). Vì AOCC dùng các thông báo trì hoãn, chi phí gửi thông báo của nó thấp hơn CBL. Hơn nữa trong môi trường khách/đại lý chuyển gửi dữ liệu đơn thuần, ảnh hưởng của một khách bị huỷ đối với hiệu năng của những khách khác là cực tiểu. Những yếu tố này góp phần làm cho hiệu năng của AO CC hơn hẳn.

7.4. QUẢN LÝ ĐỐI TƯỢNG

Bản chất thật sự của công việc quản lý đối tượng còn để ngỏ chưa bàn đến, công việc này bao gồm các tác vụ như *quản lý định danh* đối tượng, *điều chế con trỏ, di trú đối tượng, xoá đối tượng, thực thi phương*

thức, và tác 0vụ *quản lý chỗ lưu trữ tại đại lý*. Trong phần này chúng ta sẽ thảo luận một số tác vụ đó.

7.4.1 QUẢN LÝ ĐỊNH DANH ĐỐI TƯỢNG

Như đã chỉ ra trong phần 7.1, định danh đối tượng OID được hệ thống phát sinh và được dùng để xác định một cách duy nhất mỗi đối tượng (ngắn hay trường tồn, do hệ thống tạo hay người dùng tạo) trong hệ thống. Cài đặt định danh cho các đối tượng trường tồn khác với cài đặt các đối tượng ngắn hạn. Nói cụ thể, định danh cho đối tượng ngắn hạn có thể được cài đặt hiệu quả hơn.

Cài đặt OID của các đối tượng trường tồn có hai cách thông dụng, dựa trên định danh logic hoặc định danh vật lý. *Định danh vật lý POID* (physical OID) làm cho OID bằng với địa chỉ vật lý của đối tượng. Địa chỉ này có thể là địa chỉ trang và một offset tính từ địa chỉ cơ sở của trang.

Định danh logic LOID (logical OID) gồm có cấp phát một OID duy nhất cho mỗi đối tượng trên toàn bộ hệ thống (nghĩa là một đại diện). Vì OID là bất biến nên không phải trả chi phí khi di chuyển đối tượng. Điều này có được qua một bảng OID liên kết mỗi OID với địa chỉ đối tượng vật lý, bù lại phải tốn một chi phí tìm kiếm bảng cho mỗi truy xuất đối tượng. Để tránh phải trả chi phí của OID cho các đối tượng nhỏ không được dùng chung, cả hai phương pháp đều có thể xem giá trị đối tượng như định danh của chúng. Các hệ CSDL ban đầu là mạng và phân cấp đã dùng phương pháp định danh vật lý này. Các hệ CSDL hướng đối tượng có xu hướng dùng phương pháp định danh logic, vì nó hỗ trợ các môi trường động tốt hơn.

Cài đặt định danh cho đối tượng ngắn hạn gồm những kỹ thuật được dùng trong các ngôn ngữ lập trình. Giống với định danh đối tượng trường

tồn, chúng có thể thuộc loại vật lý hoặc logic. Định danh vật lý có thể là địa chỉ thực hay ảo của đối tượng, tuỳ thuộc vào việc bộ nhớ ảo có được cung cấp hay không. Phương pháp định danh vật lý có hiệu quả nhất nhưng không cho phép di chuyển các đối tượng. Cách dùng định danh logic, được đẩy mạnh sử dụng bởi lập trình hướng đối tượng, xử lý các đối tượng một cách thống nhất thông qua *bảng giám định* (indiretion table) có tính cục bộ đối với sự thực thi chương trình. Bảng này liên kết một định danh logic, được gọi là *con trỏ hướng đối tượng* OOP (object oriented pionter) trong Smalltalk, chỉ đến định danh vật lý của đối tượng.

Điều cần cân nhắc đối với bộ quản lý đối tượng là sự được- mất giữa tính tổng quát và tính hiệu quả. Hỗ trợ tổng quát của mô hình đối tượng sẽ phải mất một chi phí nào đấy. Thí dụ, các định danh có các đối tượng nhỏ có thể làm cho bảng OID rất lớn. Nếu hạn chế sự hỗ trợ của mô hình đối tượng bằng cách không cung cấp sự chia sẻ trực tiếp các đối tượng, và bằng cách đặt cơ sở trên các mức hệ thống cao hơn (thí dụ ở mức trình biên dịch hoặc ngôn ngữ CSDL) cho hỗ trợ đó, chúng ta có thể có được tính hiệu quả cao hơn. Quản lý định danh đối tượng có liên quan mật thiết đến các kỹ thuật lưu trữ.

Trong các hệ quản trị CSDL đối tượng phân tán, dùng LIOD có thể sẽ thích hợp hơn vì các thao tác như làm tái tụ dữ liệu, di chú, nhân bản và phân mảnh xảy ra thường xuyên. Sử dụng LIOD làm nảy sinh các vấn đề phân tán sau đây:

- Phát sinh LOID phải là duy nhất bên trong tầm vục của toàn bộ miền phân tán. Chúng ta dễ dàng bảo đảm tính duy nhất nếu LOID được sinh ra tại vị trí trung tâm. Trong các môi trường nhiều đại lý, mỗi vị trí đại lý phát sinh LOID cho các đối tượng được lưu tại vị trí đó. Tính duy nhất của LOID được bảo đảm bằng cách gắn định danh đại lý làm thành phần của

LOID. Vì thế LOID được cấu tạo bởi định danh đại lý và một con số tuần tự. Con số tuần tự là một biểu diễn logic cho vị trí trên đĩa của đối tượng. Số tuần tự là duy nhất bên trong một đại lý cụ thể, và thường không được dùng lại để tránh tham chiếu hiện có đến đối tượng đã bị xoá do chỉ đến đối tượng mới nhưng được nhận con số cũ. Trong thời gian truy xuất đối tượng, nếu phần định danh đại lý của LOID không được dùng trực tiếp để nhận diện vị trí đối tượng, định danh đối tượng chỉ đóng vai trò là một LOID đơn thuần. Tuy nhiên nếu phần định danh đại lý của LOID được dùng, LOID đóng vai trò như một LOID giả.

- Vị trí ánh xạ và cấu trúc dữ liệu cho LOID. Vị trí của thông tin ánh xạ LOID - POID là quan trọng, nếu các LOID đơn thuần được dùng và nếu một khách LOID_ POID là quan trọng. Nếu các LOID đơn thuần được dùng và nếu một khách được nối kết trực tiếp và đồng thời với nhiều đại lý thì thông tin ánh xạ LOID - POID phải hiện diện ở chỗ khách. Nếu LOID giả được dùng thì thông tin ánh xạ chỉ cần có ở đại lý. Thông tin ánh xạ được để ở chỗ khách là điều không mong muốn vì giải pháp này không nâng tầm được (nghĩa là thông tin ánh xạ phải được cập nhật cho tất cả các khách có truy xuất đối tượng).

7.4.2 ĐIỀU CHẾ CON TRỎ

Trong các hệ quản trị CSDL đối tượng, người ta có thể duyệt từ một đối tượng đến một đối tượng khác bằng cách dùng các biểu thức đường dẫn (path expression) có chứa các thuộc tính với giá trị của chúng dựa trên đối tượng (thí dụ nếu c thuộc kiểu Car thì c.engine.manufacturer.name là một biểu thức đường dẫn). Về cơ bản chúng đều là con trỏ thông thường trên đĩa, định danh đối tượng được dùng để biểu diễn những con trỏ này. Tuy

nhiên trong bộ nhớ, người ta muốn dùng các *con trỏ nội nhớ* (in-memorypoiter) để duyệt từ đối tượng này đến đối tượng khác. Qúa trình chuyển phiên bản con trỏ đĩa sang con trỏ nội nhớ được gọi là " *điều chế con trỏ*". Các lược đồ dựa trên phần cứng và dựa trên phần mềm là hai loại cơ chế điều chế con trỏ. Trong các lược đồ phần cứng, cơ chế *khuyết trang* (Page fault) của hệ điều hành được dùng, khi một trang được đưa vào bộ nhớ, tất cả các con trỏ trong đó đều được điều chế, và chúng chỉ đến các *khung nhớ ảo dành riêng* (reseved virtual memory frame). Các trang dữ liệu tương ứng với các khung ảo này chỉ được tải vào bộ nhớ khi có truy xuất đến chúng. Sau đó việc truy xuất trang sẽ sinh ra một khuyết trang của hệ điều hành mà nó được ghi nhận và xử lý. Trong các lược đồ phần mềm, bảng đối tượng được dùng để điều chế con trỏ. Nghĩa là một con trỏ được điều chế chỉ đến một vị trí trong bảng đối tượng. Có các thế hệ "hăng hái" và "trễ nải" của các lược đồ dựa vào phần mềm, tuỳ thuộc vào lúc con trỏ cần được điều chế. Vì thế mỗi truy xuất đối tượng đều có một mức độ *giản định* đi kèm với nó. Ưu điểm của lược đồ phần cứng đó là nó dẫn đến một hiệu quả tốt hơn khi phải duyệt lặp đi lặp lại một cây phân cấp đối tượng do không có mức độ giám định cho mỗi truy xuất đối tượng. Tuy nhiên, trong những tình huống tụ kém khi chỉ có một ít các đối tượng cần truy xuất mỗi trang, chi phí cao cho cơ chế xử lý khuyết trang làm cho lược đồ phần cứng không được hấp dẫn lắm. Lược đồ phần cứng cũng không ngăn được các ứng dụng khách tránh truy xuất các đối tượng đã bị xoá trên một trang. Hơn nữa, trong những tình huống tụ kém, lược đồ phần cứng có thể vét cạn toàn bộ không gian địa chỉ bộ nhớ ảo do các khung trang ngày càng được dành riêng càng nhiều, bất kể các đối tượng trong trang có thực sự truy xuất hay không. Cuối cùng, vì lược đồ phần cứng ngầm hướng trang nên rất khó cung cấp một khả

năng điều khiển đồng thời, quản lý vùng trữ, chuyển tải dữ liệu và khôi phục ở mức đối tượng.

7.4.3 DI TRÚ ĐỐI TƯỢNG

Một khía cạnh của các hệ phân tán là theo thời gian, các đối tượng cần di chuyển giữa các vị trí. Điều này làm nảy sinh một vấn đề. Trước tiên là *đơn vị di trú* (unit of migration). Trong các hệ thống có các trạng thái tách rời phương thức, chúng ta có thể di chuyển trạng thái của đối tượng mà không di chuyển phương thức. Tương ứng của tình huống này trong các hệ thống chỉ thuần tuý là phân mảnh một đối tượng theo các hành vi của nó. Trong mỗi trường hợp, việc áp dụng các phương thức cho một đối tượng đòi hỏi phải kích hoạt các thủ tục ở xa. Vấn đề này đã được thảo luận ở trên trong phần phân tán đối tượng. Ngay cả nếu từng đối tượng là đơn vị di trú, việc tái định vị trí của chúng có thể đưa chúng xa khỏi các đặc tả kiểu và ta phải quyết định xem các kiểu có được phân đôi tại mỗi vị trí có chứa các thể hiện hay phải truy xuất kiểu từ xa khi các hành vi hoặc phương thức được áp dụng cho các đối tượng. Ba khả năng có thể được xem xét khi di trú các lớp (kiểu):

(1) Mã nguồn được di chuyển và được biên dịch lại tại vị trí đích.

(2) Phiên bản đã được biên dịch của một lớp được di trú giống như một đối tượng khác.

(3) Mã nguồn của định nghĩa lớp được di chuyển nhưng không di chuyển các thao tác đã được biên dịch của nó mà sẽ dùng đến chiến dịch di chuyển muộn.

Một vấn đề khác, đó là việc di chuyển đối tượng phải được theo vết để có thể tìm ra những chỗ mới của chúng. Một cách thường dùng để theo

vết các đối tượng là để lại các *đại diện* (surrogate), hoặc các *đối tượng uỷ nhiệm* (poxy object). Đây là những đối tượng giữ chỗ được để lại tại vị trí cũ của đối tượng và chỉ đến vị trí mới. Truy xuất đến các đối tượng uỷ nhiệm được hệ thống chuyển hướng một cách vô hình đến đúng các đối tượng tại vị trí mới. Di trú các đối tượng có thể được thực hiện dựa trên trạng thái hiện hành của chúng. Đối tượng có thể ở một trong bốn trạng thái:

Sẵn sàng (ready): Các đối tượng sẵn sàng hiện không được kích hoạt, hoặc chưa nhận được thông báo nào nhưng sẵn sàng được kích hoạt khi nhận được một thông báo.

Đương hoạt (active): Các đối tượng đương hoạt hiện có mặt trong một hoạt động đáp lại một kích hoạt hoặc một thông baó.

Chờ (waitting): Đối tượng chờ đã kích hoạt (hoặc đã gửi môt thông báo cho) một đối tượng khác và đang chờ trả lời.

Treo (suspended): Các đối tượng treo (tạm ngưng) hiện tạm thời không sẵn sàng đối với kích hoạt .

Các đối tượng đang trong trạng thái *đương hoạt* hoặc *chờ* không được phép di trú vì hoạt động mà chúng đang tham gia sẽ bị phá vỡ. Việc di trú gồm có các bước:

(1) Chuyển tải đối tượng từ nguồn đến đích, và

(2) Tạo ra một uỷ nhiệm tại nguồn để thay đổi đối tượng ban đầu.

Có hai vấn đề cần quan tâm ở đây. Một có liên quan đến việc bảo trì thư mục hệ thống. Khi đối tượng di chuyển, thư mục hệ thống phải được cập nhật để phản ánh vị trí mới. Điều này có thể được thực hiện muộn khi một đối tượng đại diện hoặc một uỷ nhiệm định hướng lại một kích hoạt chứ không phải "hăng hái" vào lúc di chuyển. Vấn đề thứ hai là trong một môi trường năng động nơi mà các đối tượng di chuyển thường xuyên, xâu đại diện hoặc xâu uỷ nhiệm có thể rất dài. Hệ thống khi đó cần "cô đặc" những

xâu này lại theo thời gian. Tuy nhiên, kết quả cô đặc phải được phản ánh vào thư mục, và nó có thể không được thực hiện muộn.

Một vấn đề di trú quan trọng khác nảy sinh khi chuyển giao các đối tượng hợp phần. Chuyển tải một đối tượng hợp phần có thể phải chuyển tải tất cả những đối tượng được đối tượng hợp phần tham chiếu đến.. Một phương pháp giải quyết gọi là *tổng hợp đối tượng* (object assembly) sẽ được xem xét khi bàn đến vấn đề xử lý vấn tin trong phần 7.6.

7.5. LƯU TRỮ ĐỐI TƯỢNG PHÂN TÁN

Trong những vấn đề có liên quan đến việc lưu trữ đối tượng, có hai vấn đề liên quan đặc biệt với một hệ phân tán; làm *tụ đối tượng* (object clusterimg) và *dọn rác phân tán* (distributed garbage collection). Các đối tượng phức hợp và hợp phần cung cấp nhiều cơ hội cho việc làm tụ dữ liệu trên đĩa, chi phí xuất nhập đĩa cần cho việc truy xuất chúng được giảm. Dọn rác là một bài toán nảy sinh trong các CSDL đối tượng vì chúng cho phép chia sẻ dựa trên tham chiếu.

7.5.1 LÀM TỤ ĐỐI TƯỢNG

Làm tụ đối tượng (ojbect clustering) là việc nhóm các đối tượng trong các vật chứa vật lý (nghĩa là các *công đĩa*, disk extent) theo những tính chất chung, chẳng hạn theo giá trị giống nhau của một thuộc tính hoặc theo các đối tượng con của cùng một đối tượng được làm tụ.

Làm tụ đối tượng là một việc khó khăn vì hai lý do. Trước tiên nó không liên quan trực tiếp đến việc cài đặt định danh đối tượng (nghĩa là các OID logic và vật lý). OID logic mất nhiều chi phí (Vì *bảng gián định*, indirection table) nhưng cho phép phân hoạch dọc các lớp. Định danh đối tượng vật lý cho truy xuất hiệu quả hơn nhưng lại đòi hỏi mỗi đối tượng phải chứa tất cả các thuộc tính được kế thừa. Thứ hai, làm tụ các đối tượng phức theo mối liên hệ hợp phần phức tạp hơn do việc dùng chung đối tượng (đối tượng có nhiều cha).

Cho trước một *đồ thị lớp* (class graph), có ba mô hình lưu trữ cơ bản cho việc làm tụ đối tượng:

1. *Mô hình lưu trữ phân rã* DSM (decompositon storage model) phân hoạch mỗi lớp đối tượng vào các quan hệ hai ngôi (OID, attribute) và vì vậy nó dựa trên định danh logic. Ưu điểm của DSM là tính đơn giản.

2. *Mô hình lưu trữ chuẩn tắc* NMS (normanlized storage model) lưu trữ mỗi lớp như một quan hệ riêng biệt. Nó có thể được dùng với các OID logic và vật lý. Tuy nhiên, chỉ có định danh logic mới cho phép phân hoạch dọc các đối tượng theo mối liên hệ kế thừa [Kim et al., 1987].

3. *Mô hình lưu trữ trực tiếp* (direct storage model) cho phép làm tụ nhiều lớp đối tượng phức dựa trên mối liên hệ hợp phần. Mô hình này tổng quát hoá các kỹ thuật của các CDSL phân cấp và mạng, và hoạt động tốt nhất với các định danh vật lý. Nó có thể nắm bắt được tính cục bộ truy xuất đối tượng và vì vậy nó ưu việt hơn khi các kiểu mẫu truy xuất đã được biết rõ. Tuy nhiên khó khăn chính của nó là làm tụ lại một đối tượng cha đã bị xoá.

Trong một hệ phân tán, cả hai mô hình DSM và NMS đều đơn giản khi dùng phân hoạch ngang. Goblin (1993) đã cài đặt DMS như cơ sở cho một DBMS đối tượng phân tán có một bộ nhớ lớn. DSM cung cấp tính linh hoạt, và khiếm khuyết hiệu năng của nó được bù trừ lại bằng cách dùng một

bộ nhớ và vùng trữ thật lớn. Gruber và Amsaleg (1993) đã cài đặt mô hình lưu trữ trực tiếp trong một kiến trúc lưu trữ đơn cấp phân tán, trong đó mỗi đối tượng có một định danh vật lý có giá trị đối với toàn hệ thống.

7.5.2 DỌN RÁC PHÂN TÁN

Một ưu điểm của các hệ dựa trên đối tượng nằm ở chỗ các đối tượng có thể tham chiếu đến các đối tượng khác bằng định danh đối tượng. Khi các chương trình sửa đổi đối tượng và xoá bỏ tham chiếu, một đối tượng trường tồn có thể trở nên không đến được từ các gốc trường tồn của hệ thống khi không còn tham chiếu nào chỉ đến nó nữa. Một đối tượng như thế chính là "rác" và cần được phải thu hồi bởi *thể dọn rác* (garbage collector). Trong các DBMS quan hệ, dọn rác tự động là không cần thiết vì các tham chiếu đối tượng được hỗ trợ bởi các giá trị nối. Tuy nhiên những cập nhật dây chuyền theo như đặc tả của các ràng buộc tham chiếu là một hình thái đơn giản của dọn rác "thủ công". Trong ngữ cảnh các hệ điều hành hoặc ngữ cảnh ngôn ngữ lập trình tổng quát, dọn rác thủ công thường dễ gây lỗi. Vì vậy tính tổng quát của các hệ đối tượng phân tán đòi hỏi phải dọn rác phân tán tự động.

Các thuật toán dọn rác cơ bản được phân chia thành loại *đếm tham chiếu* (reference counting) và loại *dò vết* (tracing - based). Trong một hệ đếm tham chiếu, mỗi đối tượng có một con đếm (count) ghi số lượng các tham chiếu đến nó. Mỗi khi một chương trình tạo ra tham chiếu đến một đối tượng, con đếm của đối tượng được tăng lên. Khi một tham chiếu hiện có bị huỷ, con đếm tương ứng sẽ giảm đi. Bộ nhớ dành cho một đối tượng có thể được tái sử dụng khi con đếm đối tượng xuống đến zero (vào lúc đó, đối tượng chính là rác).

Dọn rác dò vết được phân thành loại *đánh dấu và quét dọn* (mark and sweep) và các loại *sao chép* (Copy- based). Thể thu dọn *đánh dấu và quét dọn* là những thuật toán hai pha. Pha đầu tiên được gọi *pha đánh dấu* (mark phase), bắt đầu từ gốc và đánh dấu mỗi đối tượng đến được (khả với) (thí dụ bằng cách đặt một bít đi kèm với mỗi đối tượng). Dấu này cũng được gọi là "màu" (color), và thể thu dọn được nói là *tô màu* (coloring) cho các đối tượng mà nó đến được. Bit đánh dấu này có thể được gắn vào trong các đối tượng hoặc trong các *bản đồ màu* (color map) để ghi nhận, cho mỗi trang bộ nhớ, màu của các đối tượng được lưu trong trang đó. Một khi tất cả các đối tượng còn sống đều được đánh dấu, bộ nhớ sẽ được kiểm tra và các đối tượng chưa được đánh dấu sẽ được tái sử dụng. Đây chính là *pha quét dọn* (sweep phase).

Thể thu dọn kiểu sao chép (copy- based) chia bộ nhớ thành hai phần tách biệt được gọi là *nơi-đi* (from- space) và *nơi- đến* (to-space). Chương trình thao các đối tượng ở nơi - đi còn nơi - đến được để trống. Thay vì đánh dấu và quét dọn, các thể thu dọn sẽ chép (thường là bằng cách theo hướng sâu trước), các đối tượng ở nơi- đi, có thể đến được từ gốc vào chỗ của nơi - đến. Khi tất cả các đối tượng còn sống đã được chép hết, quá trình thu dọn ngừng lại, nội dung của nơi - đi bị xoá bỏ, vai trò của nơi - đến và nơi - đi hoán đổi cho nhau. Quá trình sao chép lần lượt chép các đối tượng vào nơi - đến, và ép bộ nhớ, nhớ lại.

Các cài đặt cơ bản của các thuật toán *đánh dấu và quét dọn* hoặc *sao chép* đều làm "ngừng thế giới lại" nghĩa là chương trình người dùng bị tạm treo khi chương trình thu dọn xảy ra. Tuy nhiên đối với nhiều ứng dụng, các thuật toán "làm ngừng thế giới lại" lại không thể dùng được hành vi phá hoại của nó. Bảo toàn thời gian đáp ứng của ứng dụng đòi hỏi việc sử dụng các kỹ thuật "dần dần". Thể thu dọn "dần dần" phải giải quyết những vấn đề nảy

sinh do sự hoạt động đồng thời. Trong một số trường hợp, thể dọn rác có thể làm mất dấu một số đối tượng, và vì thế có thể sai sót khi tái sử dụng chúng.

Thiết kế một thuật toán thu dọn rác cho các DBMS đối tượng rất khó khăn. Đó là những vấn đề nảy sinh do phải bảo đảm sự thích ứng đối với sự cố hệ thống và ngữ nghĩa giao dịch. Nghiên cứu gần đây nhất đã đề cập đến các kỹ thuật thu dọn rác dung nạp được sai sót cho các hệ thống trường tồn với các kiến trúc tập trung (Koldnerand Weihl 1983, Toole 1993) và kiến trúc khách/đại lý (Yong 1994, Amsaleg,1995).

Tuy nhiên dọn rác phân tán khó khăn hơn dọn rác tập trung. Vì những lý do về tính hiệu quả và những tính nâng tầm được, thể dọn rác cho một hệ phân tán cần tổ hợp các thể dọn rác ở từng vị trí độc lập với thể dọn rác toàn cục liên vị. Điều phối thu dọn cục bộ và toàn cục gặp nhiều khó khăn vì nó đòi hỏi phải theo dõi cẩn thận các trao đổi tham chiếu giữa các vị trí. Ngoài ra, một đối tượng nằm tại một vị trí có thể được tham chiếu từ những đối tượng còn sống tại những vị trí ở xa nhưng không được tham chiếu bởi các đối tượng còn sống cục bộ. Một đối tượng như thế không được thể thu dọn cục bộ tái dụng bởi vì nó vẫn còn *khả với* từ gốc của một vị trí ở xa. Khâu chính là theo dõi các tham chiếu liên vị trong một môi trường phân tán, ở đó các thông báo có thể bị thất lạc, bị trùng lặp hoặc trễ, hoặc các vị trí có thể bị ngừng hoạt động.

Thể dọn tác phân tán có thể dựa trên thuật toán đếm tham chiếu phân tán hoặc dò vết phân tán. Đếm tham chiếu phân tán có nhiều rắc rối vì hai lý do. Trước tiên, đếm tham chiếu không có khả năng dọn các chu trình không đến được của các đối tượng rác (nghĩa là các đối tượng rác tham chiếu qua lại). Thứ hai, đếm tham chiếu bị thất bại bởi các sự cố thông báo thường gặp. Nghĩa là nếu thông báo không được chuyển đi an toàn theo thứ tự nhân quả của chúng thì việc duy trì bất biến cho đếm tham chiếu là điều rất khó.

Thuật toán dò vết phân tán thường tổ hợp các thể thu dọn từng vị trí độc lập với một thể thu dọn liên vị. Vấn đề chính của công việc dò vết phân tán là đồng bộ hoá pha phát hiện rác phân tán (toàn cục) với các pha tái dụng rác (cục bộ). Khi các thể dọn rác cục bộ và chương trình người dùng đều hoạt động song song, cưỡng chế một hình ảnh toàn cục và nhất quán của đồ thị đối tượng là điều không thể làm, đặc biệt là trong một môi trường nơi các thông báo không được nhận ngay lập tức, và nơi các sự cố truyền giao có thể xảy ra. Vì vậy thu dọn rác dò vết phân tán sẽ dựa trên thông tin không nhất quán để quyết định xem *một* đối tượng có phải là rác hay không. Thông tin không nhất quán này làm cho thể dọn rác dò vết phân tán trở nên rất phức tạp vì thế dọn rác phải cố gắng theo dõi tập cực tiểu các đối tượng đến được để cuối cùng ít nhất cũng tái dụng được một số đối tượng thực sự là rác.

7. 6. XỬ LÝ VẤN TIN ĐỐI TƯỢNG

Các DBMS đối tượng thế hệ đầu tiên không có ngôn ngữ vấn tin khai báo. Người ta thường nghĩ rằng các lĩnh vực ứng dụng của những hệ thống đối tượng không cần những khả năng vấn tin. Đến nay ý nghĩ này không còn đúng nữa, và khả năng vấn tin khai báo bây giờ đã được chấp nhận như một đặc trưng cơ bản của các DBMS đối tượng.

Có một mối liên hệ mật thiết giữa các kỹ thuật tối ưu hoá vấn tin, mô hình vấn tin và ngôn ngữ vấn tin. Mô hình vấn tin dựa trên mô hình dữ liệu (hoặc mô hình đối tượng).Với phần còn lại, chúng ta sẽ dùng ngôn ngữ vấn tin là OQL (object Query Language). OQL là một phiên bản hướng đối tượng của ngôn ngữ SQL. OQL đã được phát triển bởi hiệp hội chuyên khảo

của một số nhà sản xuất DBMS đối tượng có tên là Object Database group-ODBG, vì thế nó được dùng như một chuẩn công nghiệp.

Phần lớn các thể xử lý vấn tin đối tượng cho đến giờ đều dùng các kỹ thuật tối ưu hoá đã được dùng trong các hệ thống quan hệ. Tuy nhiên trong DBMS đối tượng có một số vấn đề xử lý và tối ưu hoá vấn tin phức tạp hơn nhiều trong các DBMS quan hệ. Các vấn đề quan trọng được trình bày dưới đây:

1. Ngôn ngữ vấn tin quan hệ hoạt tác trên những hệ thống kiểu rất đơn giản bao gồm một kiểu duy nhất: quan hệ, các ngôn ngữ quan hệ cho thấy rằng một toán tử quan hệ nhận một hoặc hai quan hệ hàm toán hạng và sinh ra một quan hệ làm kết quả. Ngược lại, các hệ đối tượng có các hệ thống kiểu phong phú hơn. Kết quả của các phép toán đại số đối tượng thường là các tập đối tượng (hoặc tập thể) mà chúng có thể có những kiểu khác nhau. Nếu ngôn ngữ đối tượng là đóng với những toán tử đại số thì các tập đối tượng đa chủng này có thể làm toán hạng cho những toán tử khác. Điều này đòi hỏi phải phát triển các lược đồ suy diễn kiểu chi tiết để xác định những phương thức nào có thể áp dụng cho tất cả các đối tượng trong một tập như thế. Hơn nữa, các đại số đối tượng thường hoạt tác trên các kiểu tập thể (thí dụ tập hợp, túi, danh sách) có ý nghĩa khác nhau, đặt thêm những yêu cầu cho các lược đồ suy diễn kiểu khi xác định kiểu các kết quả của các thao tác trên các tập thể của nhiều kiểu khác nhau.

2. Tối ưu hoá vấn tin quan hệ phụ thuộc vào hiểu biết về cách lưu trữ dữ liệu vật lý (đường truy xuất) mà chúng thường có sẵn cho thể tối ưu hoá vấn tin. Bao gói các phương thức với dữ liệu mà chúng hợp tác trong các DBMS đối tượng làm nảy sinh hai vấn đề quan trọng. Trước tiên xác định (hoặc đánh giá) chi phí thực thi các phương thức rõ ràng là khó khăn hơn

nhiều so với tính toán chi phí truy xuất một thuộc tính theo một đường truy xuất thực sự vì các phương thức có thể được viết bằng một ngôn ngữ lập trình tổng quát. Thứ hai, việc bao gói làm nảy sinh các vấn đề có liên quan đến tính khả truy các thông tin lưu trữ của thể xử lý vấn tin. Một số hệ thống giải quyết khó khăn này bằng cách xem thể tối ưu hoá vấn tin như một ứng dụng đặc biệt, có thể phá bỏ được sự bao gói và truy xuất trực tiếp các thông tin. Những hệ thống khác đề xuất một cơ chế mà qua đó các đối tượng "bộc lộ" chi phí của chúng như một phần trong giao diện của chúng.

3. Các đối tượng có thể (và thường như thế) có các cấu trúc phức tạp, qua đó trạng thái của một số đối tượng lại tham chiếu đến một đối tượng khác. Truy xuất các đối tượng phức như thế phải chứa cả các *biểu thức đường dẫn* (path expression). Tối ưu hoá biểu thức đường dẫn là một vấn đề chủ chốt và quan trọng trong các ngôn ngữ vấn tin đối tượng. Hơn nữa các đối tượng thuộc các kiểu có liên hệ với nhau qua các phân cấp kế thừa. Tối ưu hoá việc truy xuất các đối tượng qua các phân cấp kế thừa của chúng cũng là một bài toán phân biệt các xử lý vấn tin hướng đối tượng với sử lý vấn tin quan hệ.

4. Như đã nói trước kia trong các DBMS thiếu một định nghĩa mô hình đối tượng được thừa nhận rộng rãi. Mặc dù có một số điểm thống nhất về tập các đặc trưng cơ bản cần phải được hỗ trợ trong mọi mô hình đối tượng (chẳng hạn như định danh đối tượng bao gói trạng thái và hành vi, kế thừa kiểu, và các tập thể kiểu), cách thức hỗ trợ những đặc trưng này đều khác nhau trong các mô hình và hệ thống. Kết quả những dự án thử nghiệm với các thể tối ưu hoá đối tượng đi theo những xu hướng hoàn toàn khác nhau và ở một mức độ nào đó đều không tương thích, làm cho chúng ta hết sức khó khăn khi muốn sử dụng các kết quả thí nghiệm của những mô hình khác. Do sự đa dạng về các phương thức tiếp cận có lẽ sẽ thịnh hành trong

một thời gian nào đó, các tiếp cận mở rộng được về vấn đề tối ưu hoá vấn tin cho phép thử nghiệm những ý tưởng mới khi chúng tiến triển là điều rất quan trọng đối với việc xử lý vấn tin đối tượng.

7.6.1.THỂ XỬ LÝ VẤN TIN ĐỐI TƯỢNG

Như đã chỉ ra trong các chương trước, tối ưu hoá vấn tin có thể được mô hình hoá như một bài toán tối ưu hoá mà lời giải của nó là sự chọn lựa trạng thái "tối ưu" (tương ứng với một vấn tin đại số) dựa trên một *hàm chi phí* (cost function) trong một *không gian trạng thái* (cũng được gọi là không gian tìm kiếm) biểu diễn cho một họ các vấn tin đại số tương đương. Về mặt kiến trúc, bộ xử lý vấn tin khác nhau ở cách thức chúng mô hình hoá những thành phần này.

Nhiều thể tối ưu hoá DBMS đối tượng hiện có được cài đặt như thành phần của bộ quản lý đối tượng bên trên môt hệ thống lưu trữ, hoặc như các mô đun khách trong một kiến trúc khách/đại lý. Trong phần lớn các trường hợp, các thành phần được đề cập đến ở trên được "nối cứng" vào trong thể tối ưu hoá vấn tin. Cứ cho rằng tính mở rộng được là các mục tiêu chính của các DBMS đối tượng, người ta hy vọng sẽ phát triển một thể tối ưu hoá có thể mở rộng để đáp ứng được các chiến lược tìm kiếm, các đặc tả đại số. Thể tối ưu hoá vấn tin dựa trên quy tắc cung cấp một số khả năng mở rộng bằng cách định nghĩa những quy tắc biến đổi mới.

Dự án Open OODB tại Texas instruments tập trung vào định nghĩa một bộ khung kiến trúc mở cho các DBMS đối tượng và mô tả không gian thiết kế cho những hệ thống này. Mô đun vấn tin là một thí dụ về tính mở rộng được nội mô đun trong Op OODB. Thể tối ưu hoá vấn tin xây dựng bằng cách dùng bộ

sinh thể tối ưu hoá Volcano (Volcano optimizer generator) có thể mở rộng ứng với các toán tử đại số, các quy tắc biến đổi logic, các thuật toán thực thi, các quy tắc cài đặt, hàm đánh giá chi phí và các hàm cưỡng chế tính chất vật lý. Sự tách biệt giữa các cấu trúc phân tích cú pháp của ngôn ngữ vấn tin với đồ thị toán tử được thể tối ưu hoá hoạt tác cho phép thay thế chính ngôn ngữ hoặc chính thể tối ưu hoá. Sự tách biệt giữa các toán tử đại số và các thuật toán đại số, và các thuật toán thực thi cho phép khám phá những phương pháp cài đặt khác nhau cho các toán tử đại số. Việc sinh mã cũng là một thành phần con được định nghĩa rõ ràng của mô đun vấn tin và hoạt tác trên các DBMS khác. Thể xử lý vấn tin Open OODP gồm có một động cơ thực thi vấn tin chứa những cài đặt hiệu quả của các thao tác quét, quét chỉ mục, nối băm lại [Shapiro,1986] và tổng hợp đối tượng phức hợp mà nó sẽ được thảo luận sau.

Dự án EPO là một cách tiếp cận khác về vấn đề mở rộng khả năng tối ưu vấn tin, ở đó không gian tìm kiếm được chia thành các *vùng* (region). Mỗi vùng tương ứng với một họ các biểu thức vấn tin tương đương có thể đến được từ những họ khác. Các vùng không nhất thiết phải hoàn toàn độc lập nhau về các vấn tin mà chúng thao tác, các chiến lược điều khiển được dùng, các quy tắc biến đổi vấn tin và các mục tối ưu hoá cần đạt. Thí dụ một vùng có thể bao quát các quy tắc biến đổi sẽ giải quyết với các vấn tin chọn đơn giản, còn vùng khác có thể giải quyết với các biến đổi cho các vấn tin lồng. Tương tự một vùng có thể có mục tiêu cực tiểu hoá hàm chi phí, còn một vùng khác có thể cố gắng biến đổi các vấn tin thành một dạng mong muốn nào đó. Mỗi vùng có thể lồng đến một số mức, cho phép tìm kiếm phân cấp bên trong một vùng. Vì các vùng không biểu diễn cho các lớp tương đương, chiến lược điều khiển toàn cục cần phải có để xác định xem thể tối ưu hoá vấn tin cần chuyển như thế nào từ vùng này đến vùng khác.

Mô hình đối tượng TIGUKAT là một mô hình hành vi thống nhất có thể mở rộng, được đặc trưng bởi một ngữ nghĩa hành vi đơn thuần và một cách tiếp cận thống nhất đối với đối tượng. Mô hình này có đặc trưng hành vi ở chỗ cách duy nhất truy xuất được các đối tượng là áp dụng các hành vi cho các đối tượng (chúng thay cho cả biến thể hiện và phương thức có trong mô hình đối tượng khác). Hành vi được định nghĩa trên kiểu và cài đặt của chúng được mô hình hoá như các hàm. Kiểu và lớp là khác nhau, và đưa ra khái niệm về một tập thể được quản lý tường minh. Các vấn tin hoạt tác trên các tập thể và trả về các tập thể làm kết quả. Mỗi khái niệm, kể cả kiểu, lớp, tập thể, meta thông tin, v.v. là một *đối tượng hạng nhất* (first- class object). Tính thống nhất của mô hình đối tượng mở rộng cho mô hình vấn tin, xử lý các vấn tin như các đối tượng hạng nhất. Một kiểu Query được định nghĩa như kiểu con của kiểu Function. Vì vậy vấn tin là một loại hàm đặc dụng có thể được biên dịch và thực thi. Hơn nữa, kiểu Query có thể được đặc dụng dựa trên một lược đồ phân loại- như các vấn tin chuyên dụng và vấn tin thực dụng. Nguyên liệu và thành phẩm của các vấn tin là các tập thể (cũng là các đối tượng) cung cấp được một hệ thống.

Thể tối ưu hoá vấn tin TIGUKAT tuân theo ý tưởng biểu diễn các khái niệm hệ thống bằng các đối tượng. Không gian tìm kiếm, chiến lược tìm kiếm và hàm chi phí được mô hình bằng các đối tượng. Việc đưa các thành phần này vào trong hệ thống kiểu, cung cấp khả năng mở rộng thông qua nguyên tắc đối tượng cơ bản là sinh kiểu con và chuyên biệt hoá.

Mô hình hoá các đơn vị xây dựng của thể tối ưu hoá dựa trên chi phí như các đối tượng đã cung cấp cho thể tối ưu hoá khả năng mở rộng vốn có trong các mô hình đối tượng. Thể tối ưu hoá về cơ bản cài đặt một chiến lược tìm kiếm có liên kết một chiến lược tìm kiếm và hàm chi phí với mỗi vấn tin.

7.6.2 CÁC VẤN ĐỀ XỬ LÝ VẤN TIN

Phương pháp luận xử lý vấn tin trong các DBMS đối tượng tương tự như trong các hệ thống quan hệ, nhưng có nhiều chi tiết khác biệt do đặc trưng của mô hình đối tượng và mô hình vấn tin. Trong phần này, sẽ xem xét khi chúng được áp dụng cho việc tối ưu hoá đại số. Chúng ta cũng thảo luận một bài toán đặc biệt của mô hình vấn tin đối tượng, đó là vấn đề thực thi các biểu thức đường dẫn.

7.6.3 TỐI ƯU HOÁ ĐẠI SỐ QUAN HỆ

KHÔNG GIAN TÌM KIẾM VÀ CÁC QUY TẮC BIẾN ĐỔI

Các quy tắc biến đổi phụ thuộc vào từng đại số đối tượng cụ thể vì chúng được định nghĩa riêng biệt cho mỗi đại số đối tượng và cho các tổ hợp của chúng. Việc không có một định nghĩa chuẩn cho đại số đối tượng gây nhiều khó khăn bởi vì cộng đồng nghiên cứu không thể thu được những ích lợi từ việc tổng quát hoá nhiều nghiên cứu khác nhau. Các vấn đề tổng quát của việc định nghĩa các quy tắc biến đổi và sự thao tác các biểu thức đại số hoàn toàn tương tự như trong các hệ thống quan hệ nhưng có một khác biệt quan trọng. Các biểu thức vấn tin quan hệ được định nghĩa trên các quan hệ phẳng, còn các vấn tin đối tượng được định nghĩa trên các lớp (hoặc tập thể) hay tập hợp các đối tượng mà chúng có mối liên hệ kiểu con hoặc hợp phần với nhau. Do vậy chúng ta có thể dùng ngữ nghĩa của những mối liên hệ này trong các thể tối ưu hoá vấn tin đối tượng để có thể có thêm những biến đổi khác.

Chẳng hạn, xét các toán tử đại số đối tượng là hợp-union ($\cup$) , giao-intersection ($\cap$), và toán tử chọn select có tham số (Chọn trong P với các tham số Q_1, ...,Q_k, ký hiệu: $P\sigma <Q_1.....Q_k>$), trong đó hợp và giao có *ngữ nghĩa* như trong lý thuyết tập hợp thông thường, select chọn các đối tượng từ một tập *P* bằng cách dùng các tập đối tượng Q_1 ....Q_k làm tham số. Kết quả của những toán tử này cũng là các tập đối tượng. Dưới đây là một số quy tắc biến đổi có thể được áp dụng trong khi tối ưu hoá nhằm thu được các biểu thức vấn tin tương đương (để cho ngắn gọn, chúng ta dùng ký hiệu

$Q= \{ Q_1,...,Q_k \}.; R = \{ R_1,.....R_k\})$:

$$(P\sigma < Q >)\sigma < R > \quad \Leftrightarrow \quad (P\sigma < R >)\sigma < Q >$$

$$(P \cup Q)\sigma < R > \quad \Leftrightarrow \quad (P\sigma < R >) \cup (Q\sigma < R >)$$

$$(P\sigma < Q >)\sigma < R > \quad \Leftrightarrow \quad (P\sigma < Q >) \cap (P\sigma < R >)$$

Quy tắc đầu tiên biểu hiện tính chất giao hoán select còn quy tắc thứ hai diễn tả rằng select phân phối trên union. Quy tắc thứ ba là một đồng nhất thức biểu thị rằng select chỉ hạn chế nguyên liệu của nó và trả về tập con của đối thứ nhất.

Hai quy tắc đầu tiên hoàn toàn tổng quát ở chỗ chúng biểu thị cho các hệ thức tương đương kế thừa từ lý thuyết tập hợp. Quy tắc thứ 3 là một quy tắc biến đổi đặc biệt cho một toán tử đại số đối tượng cụ thể , được định nghĩa với một ngữ nghĩa cụ thể. Ta lại xét những quy tắc sau đây, trong đó C_i biểu thị cho tập đối tượng trong dòng tộc của lớp c_i và C^*_j biểu thị dòng tộc xa của lớp c_j (nghĩa là tập các đối tượng trong dòng tộc của c_j cũng như các dòng tộc của tất cả những lớp con của c_j):

$$C_1 \cap C_2 = \varnothing \text{ nếu } c_1 \neq c_2$$

$$C_1 \cup C_2^* = C_2^* \text{ nếu } c_1 \text{ là một lớp con của } c_2$$

$$(P\sigma < Q >) \cap R \Leftrightarrow^C (P\sigma < Q >) \cap (R\sigma < Q >)$$

$$\Leftrightarrow^C P \cap (R\sigma < Q >)$$

Những quy tắc biến đổi này có bản chất ngữ nghĩa và chúng đặc tả mô hình đối tượng và mô hình vấn tin. Thí dụ, quy tắc thứ nhất đúng vì mô hình đối tượng hạn chế mỗi đối tượng chỉ thuộc về một lớp duy nhất. Quy tắc thứ hai đúng vì mô hình vấn tin cho phép truy tìm các đối tượng trong dòng tộc xa của lớp đích. Cuối cùng quy tắc thứ ba đặt cơ sở trên quy tắc thứ nhất kiểu về khả năng áp dụng của nó cũng như một điều kiện (ký hiệu là $\Leftrightarrow^C$) rằng F đồng nhất với F, ngoại trừ mỗi xuất hiện của p được thay bằng r.

7.6.4 THUẬT TOÁN TÌM KIẾM

Như đã biết, các thuật toán tìm kiếm vét cạn sẽ liệt kê toàn bộ không gian tìm kiếm, áp dụng một hàm chi phí cho mỗi biểu thức tương đương để xác định biểu thức có chi phí thấp nhất. Một cách tốt hơn là dùng lối tiếp cận quy hoạch động, qua đó các biểu thức mới được xây dựng từ dưới lên bằng cách dùng các biểu thức con tối ưu hoá volcano sử dụng lối tiếp cận quy hoạch động từ trên xuống để tìm với *kỹ thuật tỉa rễ - nhánh - và - buộc* (brach- and- bound pruning) . Chúng được gọi chung là *các thuật toán liệt kê* (enumertive algorthm).

Bản chất của các thuật toán tìm kiếm liệt kê quan trọng hơn trong các DBMS quan hệ. Người ta cho rằng nếu số lượng nối trong một vấn tin vượt quá mười thì các chiến lược tìm kiếm liệt kê sẽ không khả thi. Các DBMS đối tượng

rất thích hợp với các ứng dụng như các hệ *hỗ trợ quyết định* (decision support system). Hơn nữa như chúng ta đã bàn luận, một phương pháp thực thi các biểu thức đường dẫn là biểu diễn chúng như các *nối hiển* (explicit jion) và dùng các thuật toán nối đã được biết để tối ưu hoá chúng.

HÀM CHI PHÍ. Như đã thấy từ trước kia, đối cho hàm chi phí dựa trên nhiều thông tin có liên quan đến việc lưu trữ dữ liệu. Điển hình, thể tối ưu hoá sẽ xem xét số lượng các mục dữ liệu (lực lượng), kích thước mỗi mục (thí dụ có chỉ mục trên nó hay không), tổ chức của nó,... Thông tin này có sẵn cho thể tối ưu hoá vấn tin trong các hệ thống quan hệ nhưng có thể không có trong các DBMS đối tượng. Như đã nói owr phần trước, có sự bất đồng trong cộng đồng nghiên cứu về vấn đề: liệu thể tối ưu hoá vấn tin có cần khả năng phá bỏ sự bao gói các đối tượng và nhìn thấy các cấu trúc dữ liệu được dùng để cài đặt chúng hay không? Nếu điều này được phép thì các hàm chi phí có thể được đặc tả tương tự như trong các hệ thống quan hệ, nếu không, cần phải xem xét đến một đặc tả khác.

Hàm chi phí có thể được định nghĩa một cách đệ quy dựa trên cây xử lý đại số. Nếu cấu trúc nội tại của các đối tượng không thể thấy được bởi thể tối ưu hoá vấn tin, chi phí của mỗi nút (biểu diễn một phép toán đại số) phải được định nghĩa. Một cách để định nghĩa nó là yêu cầu các đối tượng "bộc lộ" chi phí của chúng như thành phần của giao diện. Một tiếp cận tương tự được cung cấp trong dự án TIGUKAT. Vì các phép toán đại số là những hành vi được định nghĩa trên kiểu collection, các nút của cây xử lý đại số là các áp dụng hành vi. Có nhiều hàm khác nhau cài đặt mỗi hành vi "bộc lộ" chi phí của chúng như một hàm của (a) thuật toán thực thi và (b) tập thể mà chúng hoạt tác trên đó. Trong cả hai trường hợp một hàm toán chi phí trừu tượng hơn cho các hành vi được đặc tả vào lúc định nghĩa kiểu để từ đó thể tối ưu hoá vấn tin có thể tính được chi phí của toàn bộ cây xử lý. Định nghĩa các hàm chi phí, đặc biệt là trong

cách tiếp cận dựa trên việc các đối tượng bộc lộ chi phí của chúng cần được nghiên cứu thêm trước khi đạt được những kết luận cuối cùng.

THAM SỐ HOÁ. Tối ưu hoá vấn tin lúc biên dịch là một quá trình tĩnh ở chỗ thể tối ưu hoá sử dụng các số liệu thống kê CSDL vào lúc vấn tin được biên dịch và được tối ưu hoá bằng việc chọn phương án thực thi tối ưu. Quyết định này độc lập với số liệu thống kê lúc thực thi như tải trọng hệ thống chẳng hạn. Hơn nữa, nó không quan tâm đến những thay đổi trên các số liệu thống kê CSDL do kết quả các cập nhật có thể xảy ra giữa thời điểm vấn tin được tối ưu hoá và thời điểm nó được thực thi. Đặc biệt đây là một vấn đề trong các vấn tin thuộc loại sản xuất với khả năng được tối ưu hoá một lần và được thực thi nhiều lần. Những CSDL định nghĩa thường xuyên bị biến đổi, dẫn đến nhiều thay đổi có ý nghĩa đối với CSDL (điều này giải thích vì sao sự phát triển các lược đồ động rất quan trọng trong các DBMS đối tượng). Chiến lược tối ưu hoá vấn tin phải có khả năng thích ứng được với những thay đổi này.

Một chọn lựa đã được nghiên cứu và được cài đặt trong ObjectStore *là tối ưu hoá vấn tin với tham số* (parametric que optmization và cũng được gọi là *chọn lựa phượng án động* (dynamic plan selection). Trong trường hợp này, thể tối ưu hoá duy trì nhiều chiến lược thực thi vào lúc biên dịch và thực hiện phương án cuối cùng vào lúc chạy dựa vào các tham số hệ thống và số liệu thống kê hiện thời của CSDL. Nếu thể tối ưu hoá không có quyền truy xuất tất cả những dữ liệu này, việc tối ưu hoá đại số có thể bỏ qua tất cả các đặc trưng thực thi vật lý thay vì sinh ra một tập các biểu thức vấn tin tương đương mà chúng được chuyển giao cho bộ quản lý đối tượng. Bộ quản lý đối tượng sau đó có thể so sánh các chọn lựa (vào lúc chạy) dựa trên các đặc

trưng thực thi của chúng. Tuy nhiên cách tiếp cận này cũng có những vấn đề tiêu tốn chi phí cao vào lúc thực thi.

BIỂU THỨC ĐƯỜNG DẪN. Phần lớn các ngôn ngữ vấn tin đều cho phép dùng các vấn tin với các vị từ chứa những điều kiện về cách truy xuất đối tượng dọc theo các *xâu tham chiếu* (reference chain). Những xâu tham chiếu này được gọi là *biểu thức đường dẫn* (path expression) (đôi khi cũng được gọi là các vị từ phức tạp hoặc *nối ẩn* (implicit join). Biểu thức đường dẫn c.engine. manufacturer.name được dùng trong mục 7.4. truy xuất giá trị của thuộc tính name của đối tượng là giá trị của thuộc tính manufacturer của đối tượng là giá trị của thuộc tính engine của đối tượng c đã được định nghĩa là thuộc kiểu Car. Chúng ta có thể tạo ra các biểu thức đường dẫn chứa các thuộc tính cũng như phương thức. Tối ưu hoá việc tính toán biểu thức đường dẫn là một bài toán đã nhận được nhiều chú ý trong lĩnh vực xử lý vấn tin đối tượng.

Biểu thức đường dẫn đưa đến một ký pháp trừu tượng ngắn gọn để biểu diễn hành động duyệt qua các đồ thị hợp phần đối tượng, cho phép định hình các vị từ trên các giá trị được lồng sâu trong cấu trúc của một đối tượng. Chúng cung cấp một cơ chế thống nhất cho việc định hình các vấn tin có chứa hợp phần và các hàm thành viên kế thừa. Biểu thức đường dẫn có thể thuộc loại *trị đơn* (singer - valued) hoặc *trị tập* (set - valued), và có thể xuất hiện trong một vấn tin như thành phần của một vị từ, một đích của một vấn tin (khi là trị tập), hoặc thành phần của danh sách chiếu. Một biểu thức đường dẫn *trị đơn* nếu mỗi thành phần của một biểu thức đường dẫn đều là trị đơn; nếu ít nhất một thành phần là trị tập thì toàn bộ biểu thức là trị tập.

Bài toán tối ưu hoá biểu thức trải rộng trong toàn bộ quá trình tối ưu hoá vấn tin, nhưng trong hoặc sau khi phân tích cú pháp cho một vấn tin, trước khi tối ưu hoá đại số, trình biên dịch phải nhận dạng được biểu thức

đường dẫn có khả năng tối ưu hoá được hay không. Điều này thường đạt được qua các kỹ thuật viết lại (rewriting), biến đổi các biểu thức đường dẫn thành các biểu thức đại số logic tương đương. Một khi biểu thức đường dẫn được biểu diễn ở dạng đại số, thể tối ưu hoá vấn tin sẽ khám phá không gian của các phương án thực thi và *đại số tương đương*, tìm một phương án có chi phí nhỏ nhất. Cuối cùng phương án thực thi tối ưu có thể chứa những thuật toán tính toán các biểu thức đường dẫn một cách hiệu quả, gồm có nối băm, tổng hợp đối tượng phức, hoặc quét có chỉ mục qua các chỉ mục đường dẫn.

VIẾT LẠI VẤN TIN VÀ TỐI ƯU HOÁ ĐẠI SỐ. Xét lại biểu thức đường dẫn c.engine.manufact-turer.name. Giả sử rằng mỗi thể hiện của xe hơi có một tham chiếu đến một đối tượng Engine mỗi động cơ có một tham chiếu đến một đối tượng Manufacturer, và mỗi nhà sản xuất có một trường name. Cũng giả thiết rằng các kiểu Engine và Manufact có một dòng tộc kiểu tương ứng. Hai đường nối đầu tiên của đường dẫn ở trên có thể phải truy tìm các đối tượng động cơ và nhà sản xuất nằm trên đĩa. Đường dẫn thứ ba chỉ gồm một tìm kiếm của một trường bên trong một đối tượng nhà sản xuất. Vì thế chỉ hai đường nối đầu tiên là đưa ra các cơ hội tối ưu hoá vấn tin trong việc tính toán đường dẫn đó. Một trình biên dịch vấn tin đối tượng cần một cơ chế để phân biệt những đường nối này trong một đường dẫn biểu diễn các khả năng tối ưu hoá. Điều này thường có được qua một *pha viết lại* (rewriting phase).

Một toán tử khả chọn đã được đề xuất để tối ưu hoá các biểu thức đường dẫn là toán tử materialize (viết tắt là Mat, vật chất hoá) nó biểu diễn việc tính toán mối tham chiếu liên đối tượng (các đường nối trên đường dẫn) một cách tường minh. Điều này cho phép thể tối ưu hoá vấn tin diễn tả việc vật chất hoá nhiều thành phần như một nhóm bằng cách dùng một toán tử

Mat, hoặc sử dụng riêng lẻ từng toán tử Mat cho mỗi thành phần. Một cách khác là xem toán tử này như một "định nghĩa tầm vực" bởi vì nó mang các phần tử của một biểu thức đường dẫn vào tầm vực để các phần tử này có thể được dùng trong các phép toán sau đó hoặc trong việc ước lượng vị từ. Quy tắc định tầm sẽ phải làm sao cho một thành phần đối tượng đi vào trong tầm vực đang được quét (được ghi nhận bởi toán tử lôgic Get trong các nút lá của cây biểu thức) hoặc do đang được tham chiếu (ghi nhận bằng toán tử Mat). Các thành phần còn nằm trong tầm vực cho đến khi một phép chiếu loại bỏ chúng đi. Toán tử Mat cho phép xử lý vấn tin hợp phần tất cả các vật chất hoá thành phần bắt buộc phải có để tính một câu vấn tin, bất kể các thành phần đó cần cho việc ước lượng vị từ hoặc được sinh ra kết quả của vấn tin hay không. Mục đích của toán tử Mat là chỉ ra cho thể tối ưu hoá biết các biểu thức đường dẫn được dùng chỗ nào và chỗ nào được phép biến đổi đại số có thể áp dụng được một số quy tắc biến đổi có chứa Mat đã được định nghĩa.

CHỈ MỘT ĐƯỜNG DẪN. Nghiên cứu đáng chú ý về tối ưu hoá vấn tin đối tượng đã được giành cho việc thiết kế các cấu trúc chỉ mục nhằm làm tăng tốc độ tính toán của các biểu thức đường dẫn. Tính biểu thức đường dẫn qua các chỉ mục đường dẫn trình bày một lớp thuật toán thực thi vấn tin được dùng trong việc tối ưu hoá vấn tin đối tượng. Nói cách khác, tính toán hiệu quả các biểu thức đường dẫn qua chỉ mục đường dẫn trình bày một họ các chọn lựa cài đặt cho các toán tử đại số, chẳng hạn như Mat và nối, được dùng để biểu diễn các tham chiếu liên đối tượng.

Biểu thức đường dẫn đưa đến một ký pháp trừu tượng ngắn gọn để biểu diễn hành động duyệt qua các đồ thị hợp phần đối tượng, cho phép định hình các vị từ trên các giá trị được lồng sâu trong cấu trúc của một đối

tượng. Chúng cung cấp một cơ chế thống nhất cho việc định hình các vấn tin có chứa hợp phần và các hàm thành viên kế thừa. Biểu thức đường dẫn có thể thuộc loại trị đơn (single-valued) hoặc trị tập (set-valued) và có thể xuất hiện trong một vấn tin như thành phần của một vị từ, một đích của một vấn tin (khi là trị tập) hoặc thành phần của danh sách chiếu. Một biểu thức đường dẫn là trị đơn nếu mỗi thành phần của một biểu thức đường dẫn đều là trị đơn; nếu ít nhất một thành phần là trị tập thì toàn bộ biểu thức là trị tập.

7.6.5 THỰC THI VẤN TIN

Các DBMS quan hệ tận dụng được sự tương ứng gần gũi các phép toán đại số quan hệ và các nguyên thuỷ truy xuất của hệ thống lưu trữ. Vì vậy việc tạo ra các phương án thực thi cho một biểu thức vấn tin về cơ bản có liên quan đến việc chọn lựa và cài đặt các thuật toán hiệu quả nhất để thực thi từng phép toán đại số và các tổ hợp của chúng. Trong các DBMS đối tượng, vấn đề phức tạp hơn do sự khác biệt về các mức trừu tượng của các đối tượng được định nghĩa theo hành vi và việc lưu trữ chúng. Bao gói các đối tượng nhằm che dấu chi tiết cài đặt của chúng và việc lưu các phương thức đã đặt ra một bài toán thiết kế đầy thách thức có thể được khẳng định như sau: "Vào lúc nào trong quá trình xử lý vấn tin, thể tối ưu hoá vấn tin cần truy xuất các thông tin liên quan đến việc lưu trữ đối tượng"? Một chọn lựa là để công việc này cho bộ quản lý đối tượng. Hệ quả là phương án thực thi vấn tin sinh ra từ biểu thức vấn tin sẽ thu được vào cuối bước viết lại vấn tin bằng cách ánh xạ biểu thức vấn tin thành một tập các lời gọi giao diện của bộ quản lý đối tượng. Giao diện bộ quản lý đối tượng gồm một tập các thuật toán thực thi. Phần này sẽ xem lại một số thuật toán thực thi rất có thể

sẽ là thành phần của các động cơ thực thi vấn tin đối tượng có hiệu năng tốt trong tương lai.

Một động cơ thực thi vấn tin đòi hỏi phải có ba lớp thuật toán cơ bản trên các tập thể đối tượng: quét tập thể (collection scan), quét có chỉ mục (indexed scan) và đối sách tập hay so tập (set matching). Quét tập thể là một thuật toán đơn giản bằng cách truy xuất lần lượt tất cả các đối tượng trong một tập thể. Quét có chỉ mục cho phép truy xuất hiệu quả đến các đối tượng được chọn trong một tập thể thông qua một chỉ mục. Chúng ta có thể dùng một trường của đối tượng hay một giá trị được trả về từ một phương thức nào đó làm khoá cho chỉ mục. Cũng có thể định nghĩa các chỉ mục trên các giá trị được lồng sâu trong cấu trúc của một đối tượng (nghĩa là các chỉ mục đường dẫn). Trong phần này chúng ta nói đến một thí dụ đại diện cho các đề xuất về chỉ mục đường dẫn (path index). Các thuật toán đối sách tập nhận nguyên liệu là các tập thể đối tượng và sinh ra các đối tượng hợp phần có liên quan với nhau theo một tiêu chuẩn nào đó. Phép nối (join), giao (intersection) và tổng hợp (assembly) là những thí dụ về các thuật toán thuộc loại này.

7.6.6 CHỈ MỤC ĐƯỜNG DẪN

Như đã nói ở phần trên, hỗ trợ cho biểu thức đường dẫn là một đặc trưng phân biệt các vấn tin đối tượng với vấn tin quan hệ. Nhiều kỹ thuật tạo chỉ mục đã được thiết kế để làm tăng tốc độ tính toán các biểu thức đường dẫn dựa trên khái niệm của chỉ mục nối (join index).

Một trong số các kỹ thuật chỉ mục đường dẫn đã được phát triển cho hệ quản trị CSDL đối tượng GemStone là tạo ra một chỉ mục trên mỗi lớp

cần được duyệt qua bởi một đường dẫn. Kỹ thuật này cũng được đề xuất cho DBMS đối tượng Orion. Ngoài chỉ mục trên các biểu thức đường dẫn, chúng ta có thể định nghĩa các chỉ mục trên các đối tượng qua sự kế thừa kiểu của chúng.

Quan hệ hỗ trợ truy xuất (access support relation) là một kỹ thuật tổng quát khác để biểu diễn và tính các biểu thức đường dẫn. Quan hệ hỗ trợ truy xuất là một cấu trúc dữ liệu lưu trữ các biểu thức đường dẫn đã được chọn. Những biểu thức được chọn là những biểu thức thường được duyệt qua nhất. Các nghiên cứu đã cung cấp những bằng chứng ban đầu rằng hiệu năng vấn tin được thực hiện bằng các quan hệ hỗ trợ truy xuất cải thiện được vào khoảng hai lần so với các vấn tin không dùng nó. Một hệ thống có dùng các quan hệ hỗ trợ truy xuất cũng phải xét đến chi phí duy trì chúng khi có cập nhật đến các quan hệ cơ sở.

7.6.7 ĐỐI SÁNH TẬP

Như đã nói trên, biểu thức đường dẫn chỉ ra các đường duyệt dọc theo các mối liên hệ hợp phần của các đối tượng. Chúng ta đã thấy rằng một cách để thực thi một biểu thức đường dẫn là biến đổi nó thành một nối giữa các tập đối tượng nguồn và đích. Một số thuật toán nối đã được đề xuất, chẳng hạn nối băm lai (hybrid-hash join) hoặc nối băm theo trỏ (pointer-based hash join) [Shekita and Carey, 1990]. Thuật toán đầu dùng nguyên tắc chia để trị nhằm phân hoạch đệ quy hai tập thể toán hạng vào các lô bằng cách dùng một hàm băm trên thuộc tính nối. Mỗi lô này có thể được chứa vừa trong bộ nhớ. Mỗi cặp lô sau đó được nối lại trong bộ nhớ để sinh ra kết quả. Nối băm theo trỏ được dùng khi mỗi đối tượng trong một tập thể toán

hạng (gọi là R) có một con trỏ chỉ đến một đối tượng trong một tập thể làm toán hạng (gọi là S). Thuật toán thực hiện qua ba bước, bước thứ nhất là phân hoạch R giống như thuật toán băm lai, ngoại trừ nó được phân hoạch theo giá trị OID chứ không phải thuộc tính nối, tập đối tượng S không được phân hoạch. Trong bước thứ hai, mỗi phân hoạch R_i của R được nối với S bằng cách lấy R_i và xây dựng một bảng băm cho nó trong bộ nhớ. Bảng này được xây dựng bằng cách băm mỗi đối tượng $r \in R$ trên giá trị con trỏ của nó chỉ đến đối tượng tương ứng trong S. Kết quả là tất cả các đối tượng R có tham chiếu đến cùng một trang trong S đều được nhóm lại trong cùng một mục ghi của bảng băm. Bước thứ ba, sau khi xây dựng bảng băm cho R_i, mỗi mục của nó sẽ được quét. Với mỗi mục trong bảng băm, trang tương ứng trong S được đọc, và tất cả các đối tượng trong R có tham chiếu đến trang đó được nối với các đối tượng tương ứng trong S. Hai thuật toán này về cơ bản là thuật toán tập trung và không có bất kỳ một đối tác phân tán nào.

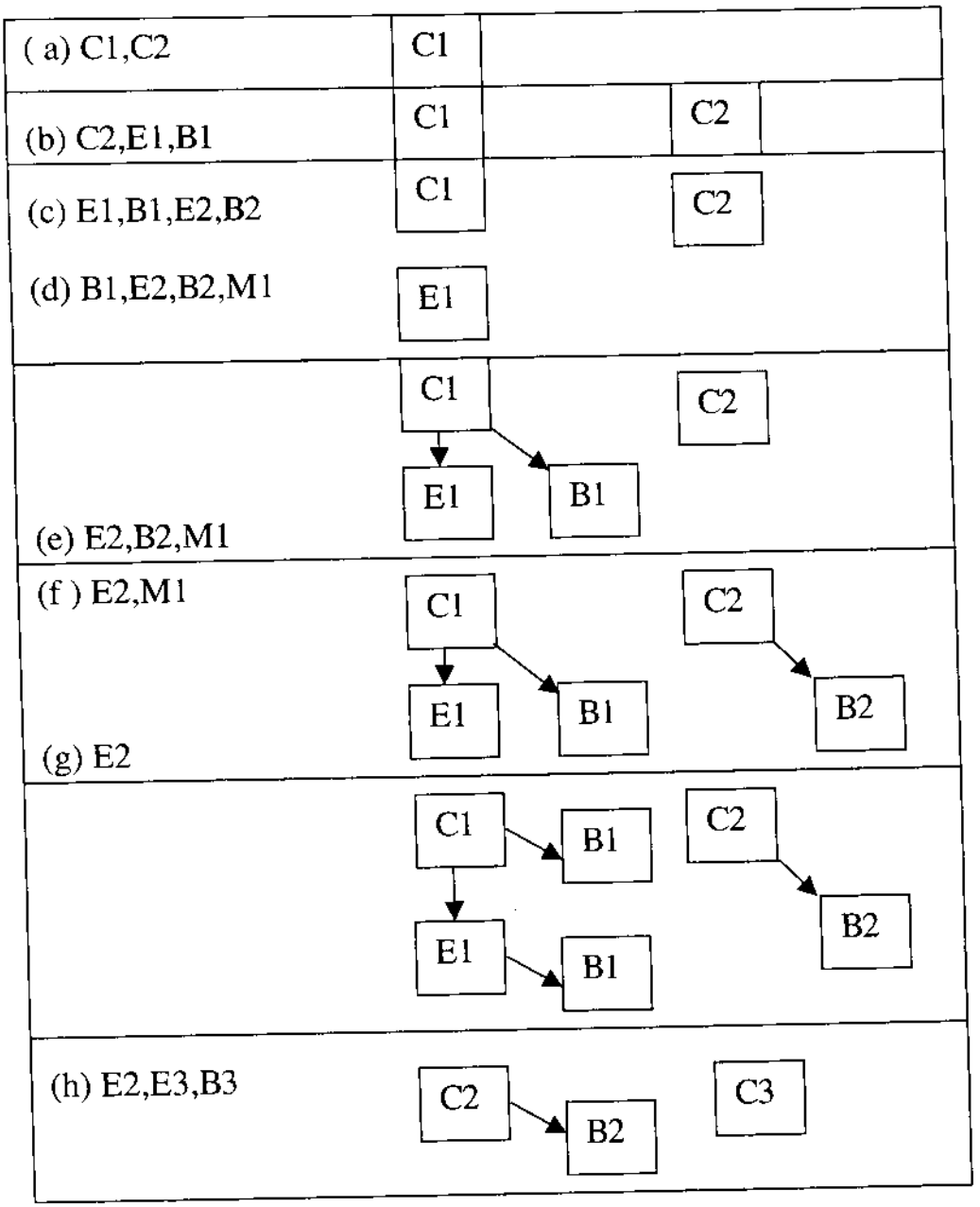
Một phương pháp khả thi được chọn cho thuật toán thực thi nối, được gọi là *tổng hợp (assembly)* là một tổng quát hoá thuật toán nối băm theo trỏ cho trường hợp chúng ta cần tính một *nối đa dòng (multi-way join).* Assembly đã được đề xuất như một toán tử đại số đối tượng. Phép toán này tổng hợp hiệu quả các mảnh của các trạng thái đối tượng cần thiết cho một bước xử lý cụ thể, và trả chúng về làm một đối tượng phức trong bộ nhớ. Nó dịch các biểu diễn trên đĩa của các đối tượng phức thành các biểu diễn có thể duyệt được trong bộ nhớ.

Tổng hợp một đối tượng phức có gốc tại các đối tượng thuộc kiểu R chứa các thành phần có kiểu S, U và T cũng tương tự như việc tính một nối bốn dòng của những tập này. Khác biệt giữa tổng hợp và nối trỏ n – dòng ở chỗ tổng hợp không cần phải quét toàn bộ tập thể các đối tượng gốc trước khi tạo ra một kết quả duy nhất.

Thay vì tổng hợp mỗi lần một đối tượng phức, toán tử asembly tổng hợp một cửa sổ (window) có kích thước W của các đối tượng phức một cách đồng thời. Ngay khi một trong những đối tượng phức này được tổng hợp và được truyền lên trên cây thực thi vấn tin, toán tử assembly truy xuất một cửa sổ để tiếp tục. Sử dụng một cửa sổ các đối tượng phức làm tăng kích thước dự trữ các tham chiếu chưa giải quyết và dẫn đến nhiều chọn lựa cho việc tối ưu hoá các truy xuất đĩa. Do tính ngẫu nhiên trong việc hoá giải các tham chiếu, toán tử assembly phân phối các đối tượng đã tổng hợp theo một thứ tự ngẫu nhiên hướng lên cây thực thi. Hành vi này là đúng đắn trong việc xử lý vấn tin hướng tập hợp nhưng có thể không đúng với các kiểu tập thể khác như danh sách chẳng hạn.

Thí dụ 7.9:

Xét thí dụ được biểu diễn việc tổng hợp một tập các đối tượng Car. Các hộp trong hình biểu diễn các thể hiện của các kiểu được chỉ ra ở bên trái và các cạnh biểu thị các mối liên hệ hợp phần (thí dụ Engine). Giả sử rằng assembly đang dùng một cửa sổ có kích thước 2. Toán tử assembly, bắt đầu bằng cách điền vào cửa sổ này hai tham chiếu đối tượng Car là C1 và C2 (vì W = 2) từ tập hợp đó (xem hình 7.1 (a)- dòng đầu). Toán tử asembly bắt đầu bằng cách chọn ra trong số các tham chiếu còn đang đơn độc, chẳng hạn như C1. Sau khi giải C1 (chuyển gửi) hai tham chiếu mới chưa giải được thêm vào danh sách (hình 7.1 (b)). Giải C2 khiến phải thêm hai tham chiếu nữa vào danh sách (hình 7.1 (c)), v.v. cho đến khi đối tượng phức thứ nhất được tổng hợp (hình 7.1 (g)). Đến lúc này, đối tượng đã tổng hợp được truyền lên trên cây thực thi vấn tin, giải phóng một cửa sổ nào đó. Một tham chiếu đối tượng Car mới là C3 được đưa vào danh sách rồi được giải, mang vào hai tham chiếu E3 và B3 (hình 7.1 (h)).



Hình 7.1 Minh hoạ thí dụ 7.9

Mục tiêu của thuật toán tổng hợp là tổng hợp đồng thời một cửa sổ của các đối tượng phức. Tại mỗi thời điểm trong thuật toán, tham chiếu đơn độc có tối ưu hoá các truy xuất đĩa sẽ được chọn. Có những thứ tự khác nhau (lịch biểu) giải quyết các tham chiếu, chẳng hạn như theo hướng sâu, theo hướng ngang, hoặc leo thang (elevator). Những kết quả hiệu năng chỉ ra rằng kiểu leo thang thực hiện tốt hơn so với theo hướng sâu và hướng ngang

Một số khả năng có thể chọn khi cài đặt một phiên bản phân tán của thao tác này. Một chiến lược là chuyển tất cả dữ liệu đến một vị trí trung tâm để xử lý. Chiến lược này dễ cài đặt nhưng nhìn chung có thể không hiệu quả. Một chiến lược khác là thực hiện các thao tác đơn giản (thí dụ chọn, tổng hợp cục bộ) tại những vị trí ở xa rồi chuyển tất cả dữ liệu về một vị trí trung tâm để tổng hợp cuối cùng. Chiến lược này cũng đòi hỏi một sự điều khiển đơn giản vì tất cả mọi truyền giao đều xảy ra qua vị trí trung tâm. Chiến lược thứ ba phức tạp hơn rất nhiều: thực hiện các phép toán phức tạp (thí dụ nối, tổng hợp đầy đủ các đối tượng ở xa) tại các vị trí ở xa rồi chuyển kết quả đến vị trí trung tâm để tổng hợp cuối cùng. Một DBMS đối tượng phân tán có thể gồm cả ba hoặc một số trong những chiến lược này.

7.7. QUẢN LÝ GIAO DỊCH ĐỐI TƯỢNG

Quản lý giao dịch (transaction management) trong các DBMS đối tượng phân tán chưa được nghiên cứu ngoại trừ ở phần có liên quan với bài toán vùng trữ đã thảo luận trước kia. Tuy nhiên các giao dịch trong các đối tượng làm nảy sinh một số vấn đề đáng chú ý, và việc thực thi của chúng trong một môi trường phân tán là một thách thức. Đây là một lĩnh vực cần được nghiên cứu nhiều hơn. Trong phần này chúng ta chỉ thảo luận các vấn

đề đặc biệt nảy sinh khi mở rộng khái niệm giao dịch cho các DBMS đối tượng.

Trong trường hợp hệ thống đối tượng, giao dịch không chỉ cấu tạo bởi các thao tác đọc/ ghi đơn giản mà lại cần đến các thuật toán đồng bộ hoá có khả năng giải quyết được với các thao tác phức tạp trên các đối tượng trừu tượng (có thể là các đối tượng phức).

Trong một số lĩnh vực ứng dụng, mô thức đồng bộ hoá giao dịch dựa trên sự tranh chấp giữa các giao dịch để hoàn tất một tác vụ chung nào đó, chẳng hạn như trong môi trường làm việc hiệp đồng.

Các yêu cầu quan trọng hơn về quản lý giao dịch trong các DBMS đối tượng có thể được liệt kê là:

1. Bộ quản lý giao dịch truyền thống chỉ đồng bộ hoá các thao tác Read và Write đơn giản. Tuy nhiên đối tác của chúng trong các DBMS đối tượng phải có khả năng giải quyết được với các thao tác trừu tượng (absstract operation). Nó cũng có thể cải thiện được các hoạt động đồng thời bằng cách dùng những hiểu biết ngữ nghĩa của chúng về các đối tượng và các thao tác trừu tượng.

2. Các giao dịch truyền thống truy xuất các đối tượng "phẳng" (thí dụ các trang, các bộ), còn các giao dịch trong các DBMS đối tượng đòi hỏi sự đồng bộ việc truy xuất đến các đối tượng hợp phần và phức hợp. Đồng bộ truy xuất đến những đối tượng này đòi hỏi phải đồng bộ hoá truy xuất đến các đối tượng thành phần.

3. Một số ứng dụng được DBMS hỗ trợ có những kiểu mẫu truy xuất CSDL khác với các ứng dụng truyền thống, ở đó các truy xuất là tương tranh (thí dụ hai người sử dụng cùng truy xuất đến một tài khoản). Vì vậy, việc dùng chung có tính hiệp đồng hơn như trong trường hợp nhiều người cùng truy xuất và làm việc chung trên cùng một tư liệu thiết kế. Trong trường hợp

này, truy xuất cần phải được đồng bộ hoá, nhưng người sử dụng mong muốn hiệp đồng chứ không phải tranh chấp truy xuất đến các đối tượng dùng chung.

4. Những ứng dụng này đòi hỏi phải hỗ trợ các hoạt động dài hơi trải qua nhiều giờ, nhiều ngày hoặc thậm chí nhiều tuần lễ (thí dụ khi làm việc trên một đối tượng thiết kế). Vì thế cơ chế giao dịch phải hỗ trợ việc chia sẻ các kết quả từng phần. Ngoài ra để tránh sự cố của tác vụ một phần làm tổn hại đến hoạt động lâu dài, chúng ta cần phân biệt giữa các hoạt động có tính quyết định đối với việc hoàn tất một giao dịch và những hoạt động không có điều đó, và cung cấp những hành động thay thế trong trường hợp hoạt động chính bị thất bại.

5. Người ta cho rằng nhiều ứng dụng sẽ có nhiều lợi ích từ các khả năng hoạt hoá để đáp ứng đúng lúc với các biến cố và thay đổi trong môi trường. Mô thức CSDL mới này đòi hỏi phải theo dõi các biến cố và việc thực thi của các hoạt động được hệ thống kích hoạt bên trong các giao dịch đang chạy.

Những đòi hỏi này chỉ ra một nhu cầu mở rộng các chức năng quản lý giao dịch truyền thống để nắm bắt được ngữ nghĩa ứng dụng và dữ liệu, và nhu cầu giảm bớt tính chất biệt lập của giao dịch. Điều này đòi hỏi phải xem xét lại các khía cạnh quản lý giao dịch đã được thảo luận trước đây.

7.7.1 TIÊU CHUẨN ĐÚNG ĐẮN

Trong chương trước chúng ta đã giới thiệu tính khả tuần tự như một tiêu chuẩn đúng đắn cơ bản cho việc thực thi đồng thời các giao dịch CSDL. Có một số cách định nghĩa tính khả tuần tự, những khác biệt dựa trên cách định nghĩa một tương tranh (conflict). Chúng ta sẽ tập trung vào ba khả

năng: *tính giao hoán (commutativity), sự vô hiệu hoá (invalidaiton)* và *tính khôi phục được (recoverability).*

TÍNH GIAO HOÁN

Tính giao hoán khẳng định rằng hai thao tác là tương tranh, nếu kết quả của các thực hiện tuần tự cho những giao dịch này không tương đương. Định nghĩa tương tranh truyền thống đã thảo luận trước đây chỉ là một trường hợp đặc biệt. Xét các thao tác đọc- ghi đơn giản R(x) và W(x). Nếu không biết gì về ngữ nghĩa trừu tượng của các thao tác Read và Write hoặc đối tượng x mà chúng hoạt tác, chúng ta phải chấp nhận rằng một thao tác R(x) đi sau một thao tác W(x) sẽ không truy xuất được giá trị giống như khi nó đi trước W(x). Vì thế thao tác Write luôn tương tranh với các thao tác Read và Write khác. Bảng tương tranh (hoặc ma trận tương thích) được cho trong hình 7.3 cho hai thao tác Read và Write thực sự được dẫn xuất từ mối liên hệ giao hoán giữa hai thao tác này. Bảng này được gọi là ma trận tương thích vì hai thao tác không có tương tranh được gọi là tương thích với nhau. Vì loại giao hoán này dựa trên thông tin cú pháp của các thao tác (có nghĩa chúng là Read và Write), chúng ta gọi đây là tính giao hoán cú pháp (syntactic commutativity).

Trong hình 7.3, các thao tác Read, Write và Write, Write không giao hoán. Vì thế chúng tương tranh, và tính khả tuần tự sẽ bảo đảm rằng hoặc tất cả các thao tác tương tranh của giao dịch T_i đều đi trước tất cả các thao tác tương tranh của T_k, hoặc ngược lại.

Tuy nhiên nếu có xem xét đến ngữ nghĩa của các thao tác, chúng ta có thể cung cấp một định nghĩa ít khắt khe hơn về sự tương tranh. Cụ thể, các thực thi đồng thời của Write-Write và Read-Write có thể được xem là không tương tranh. *Giao hoán ngữ nghĩa (semantic commutativity)* có dùng đến ngữ nghĩa các thao tác và tình huống kết thúc của chúng.

Thí dụ 7.10:

Xét một kiểu dữ liệu trừu tượng set và ba phép toán được định nghĩa trên nó: *insert* và *Delete*, tương ứng với một thao tác Write, và phép toán *Member*, kiểm tra tính chất thành viên và tương ứng với một thao tác Read. Do ngữ nghĩa của những phép toán này, hai phép toán insert trên một thể hiện của kiểu tập hợp sẽ giao hoán, cho phép chúng được thực hiện đồng thời. Tính giao hoán của insert với member và tính giao hoán của Delete với member tuỳ thuộc vào chúng có tham chiếu đến cùng một đối và các kết quả của chúng hay không ?

Cũng có thể định nghĩa tính giao hoán với tham chiếu đến trạng thái CSDL. Trong trường hợp này, chúng ta thường cho phép nhiều thao tác giao hoán hơn.

Thí dụ 7.11:

Trong thí dụ 7.10 chúng ta đã chỉ ra rằng Insert và member không giao hoán nếu chúng tham chiếu đến cùng một đối. Tuy nhiên nếu tập này đã chứa phần tử được tham chiếu, hai thao tác này sẽ giao hoán ngay cả nếu các đối của chúng giống nhau.

Vấn đề đặt ra là làm thế nào để hình thức hoá những hiểu biết trực quan về tính giao hoán của các thao tác trên các kiểu dữ liệu trừu tượng dựa trên ngữ nghĩa của chúng?

Quan hệ tương tranh (conflict relation) được định nghĩa là quan hệ hai ngôi giữa các phép toán và có xét đến cả phép toán và kết quả của nó. Một phép toán bây giờ được định nghĩa là một cặp kích hoạt và hồi đáp với kích hoạt đó; thí dụ x: [insert(3), ok] là một kích hoạt hợp lệ của một phép toán chèn trên tập x, hồi đáp rằng phép toán đã được thực hiện một cách

đúng đắn. Hai loại giao hoán và quan hệ giao hoán tương ứng của chúng có thể được định nghĩa: giao hoán tới (forward commutativity) và giao hoán lùi (backward commutativity). Giả sử có hai phép toán P và Q trạng thái s trong đó P và Q đều được định nghĩa như sau: với mỗi trạng thái s trong đó P và Q đều được định nghĩa (riêng rẽ), P(Q(x)) = Q(P(x)) và có nghĩa là nếu trước tiên chúng ta áp dụng phép toán Q cho trạng thái s rồi áp dụng phép toán P cho kết quả đó, chúng ta thu được kết quả giống như khi áp dụng P trước cho trạng thái s rồi áp dụng phép toán Q cho kết quả. Giao hoán lùi được định nghĩa (nghĩa là Q(s) được định nghĩa nhưng P(s) có thể không).

P(Q(s)) = Q(P(s)). Tất nhiên cả giao hoán tới và giao hoán lùi đều mở rộng cho trường hợp P và Q là dãy các phép toán chứ không phải chỉ là một phép toán đơn.

	[insert(i), ok[	[Delete(i), ok]	[Member(i), true]	[Member(i), falsse]
[insert(I), ok]	+	-	+	-
[Delete(I), ok]	-	+	+	+
[Member(I), ok]	+	-	+	+
[Member(i), false]	-	+	+	+

Hình 7.2 Bảng tương thích cho giao hoán tiến trong các tập hợp

	[insert(i), ok[	[Delete(i), ok]	[Member(i), true]	[Member(i), falsse]
[insert(i), ok]	+	-	-	-
[Delete(i), ok]	-	+	-	+
[Member(i), ok]	-	-	+	+
[Member(i), false]	-	+	+	+

Hình 7.3 Bảng tương thích cho giao hoán lùi trong các tập hợp.

Thí dụ 7.12:

Quan hệ tương thích tới và lùi cho kiểu ADT tập hợp được cho tương ứng trong hình 7.2 và 7.3. Trong những bảng này, phép toán Member được định nghĩa một lần với mã trả về cho việc thực thi thành công ("true"), và một với mã không thành công ("false").

Điều quan trọng là chú ý sự khác biệt về các trạng thái mà trên đó các phép toán được định nghĩa. Trong giao hoán tới, cả hai phép đều được định nghĩa trên cùng một trạng thái khởi đầu. Vì thế không có gì khác biệt bất kể phép toán nào được áp dụng trước, với điều kiện là kết quả cuối cùng đều như nhau.

Thí dụ 7.13:

Nếu trạng thái khởi đầu của đối tượng tập hợp là {1, 2. 3} phép toán đầu tiên trên đối tượng tập hợp đó là cặp kích hoạt hồi đáp [insert(3), ok] và phép toán thứ hai là [Member(3), true], cả hai phép toán đều được định nghĩa trên {1, 2, 3} và kết quả áp dụng chúng theo một trong hai thứ tự đều như nhau. Tuy nhiên nếu như tất cả điều mà chúng ta biết, đó là trạng thái sẽ là {1, 2, 3} sau khi áp dụng phép toán {insert(3), ok} chúng ta không thể nói được trạng thái khởi đầu là {1, 2} hay {1, 2, 3}. Vì thế với đối tượng tập hợp, các thao tác [insert(x), ok] và [Member(x), true] là giao hoán tới nhưng không giao hoán lùi.

Thí dụ 7.14:

Thí dụ sau minh hoạ rằng các quan hệ giao hoán tới và lùi không so sánh được bằng cách xét các quan hệ này đối với một kiểu dữ liệu trừu tượng về tài khoản ngân hàng. Các quan hệ được cho trong các hình 7.5 và 7.6. Các phép toán đều dễ hiểu, ngoại trừ Post(i), nó nhập tỷ lệ phần trăm i của lãi suất vào đối tượng tài khoản. Đối của các phép toán là số lượng tiền trong tài khoản.

	[withdraw(m),ok]	[withdraw(m),no]	[Deposit(n),ok]	[Balance,rr]	[Post(i), ok]
[withdraw(m),ok]	-	+	+	+	-
[withdraw(m),no]	+	+	-	-	+
[Deposit(n),ok]	+	-	+	+	-
[Balance,rr]	-	+	-	-	-
[Post(i), ok]	-	+	-	-	-

Hình 7.4 Bảng giao hoán tới cho một đối tượng tài khoản

	[withdraw(m),ok]	[withdraw(m),no]	[Deposit(n),ok]	[Balance,rr]	[Post(i), ok]
[withdraw(m),ok]	+	-	-	-	-
[withdraw(m),no]	-	+	-	+	+
[Deposit(n),ok]	-	-	+	-	-
[Balance,rr]	-	+	-	+	-
[Post(i), ok]	-	+	-	-	-

Hình 7.5 Bảng giao hoán lùi cho một đối tượng tài khoản

Tính không so sánh được của hai quan hệ này gây nhiều khó khăn trong việc cài đặt các bộ quản lý giao dịch có sử dụng chúng. Về cơ bản, một trong hai phải được chọn để cưỡng chế, mặc dù mỗi quan hệ đều cho

phép một số lịch biểu mà quan hệ kia loại bỏ. Nakajima (1994) đã mở rộng nghiên cứu này và định nghĩa một quan hệ giao hoán tổng quát (general commutativity relation), đó là một tập bao hàm của cả hai quan hệ giao hoán tới và giao hoán lùi. Nếu FC(o) và BC(o) tương ứng là quan hệ giao hoán tổng quát (general commutativity relation), đó là một tập bao hàm của cả hai quan hệ giao hoán tới và giao hoán lùi. Nếu FC(o) và BC(o) tương ứng là quan hệ giao hoán tới và giao hoán lùi cho đối tượng o thì quan hệ giao hoán tổng quát được định nghĩa là GC(o) = FC(o)∪ BC(o). Quan hệ giao hoán tổng quát cho thí dụ tài khoản được cho trong hình 7.6. Mặc dù việc dùng quan hệ giao hoán dường như được ưa chuộng hơn, cưỡng chế nó là điều không đơn giản.

	[withdraw(m),ok]	[withdraw(m),no]	[Deposit(n),ok]	[Balance,rr]	[Post(i), ok]
[withdraw(m),ok]	+	+	+	-	-
[withdraw(m),no]	+	+	-	+	+
[Deposit(n),ok]	+	-	+	-	-
[Balance,rr]	-	+	-	+	-
[Post(i), ok]	-	+	-	-	-

Hình 7.6 Quan hệ giao hoán tổng quát cho thí dụ tài khoản.

VÔ HIỆU HOÁ

Vô hiệu hoá hay làm mất hiệu lực (invalidation) là định nghĩa một tương tranh giữa hai phép toán không dựa trên cơ sở chúng có giao hoán hay không mà theo kết quả là liệu việc thực thi phép toán này có làm mất hiệu lực (vô hiệu hoá) phép toán kia hay không. Một phép toán P vô hiệu hoá một phép toán Q nếu có hai lịch biểu H1 và H2 sao cho H1 • P • H2 và H1 • H2

• Q đều hợp lệ nhưng H1 • P • H2 •Q thì không. Trong ngữ cảnh này, một lịch biểu hợp lệ (legal history) biểu diễn một lịch biểu đúng cho đối tượng tập hợp và được xác định theo ngữ nghĩa của nó. Theo đó một quan hệ bị – vô - hiệu – hoá - bởi (invalidated – by) được định nghĩa là cấu tạo bởi tất cả các cặp phép toán (P, Q) sao cho P vô hiệu hoá Q, Quan hệ bị – vô - hiệu – hoá - bởi thiết lập quan hệ tương tranh, tạo cơ sở để thiết lập tính khả tuần tự. Xét thí dụ Set, một phép toán insert không thể bị vô hiệu hoá bởi bất kỳ một phép toán khác, nhưng Member có thể bị vô hiệu hoá bởi Delete nếu đối của chúng giống nhau.

TÍNH KHÔI PHỤC ĐƯỢC

Tính khôi phục được (recoverability) là một quan hệ tương tranh khác được định nghĩa để xác định các lịch biểu khả tuần tự. Về trực quan, một phép toán P được nói là khôi phục được ứng với phép toán Q nếu giá trị được trả về bởi P độc lập với việc Q có được thực thi trước P hay không. Quan hệ tương tranh thiết lập trên cơ sở tính khôi phục được dường như đồng nhất với quan hệ được thiết lập bởi tính vô hiệu hoá,. Tuy nhiên nhận xét này chỉ dựa vào một số thí dụ và không hề có một chứng minh hình thức nào cho sự tương đương này. Thực sự, việc thiếu hụt một lý thuyết hình thức để suy diễn những quan hệ tương tranh này là một khiếm khuyết nghiêm trọng cần phải được tập trung giải quyết.

7.7.2 MÔ HÌNH GIAO DỊCH

Trong chương trước chúng ta đã xét một số mô hình giao dịch, từ các giao dịch phẳng đến các hệ thống lưu công. Trong thảo luận trước đây,

chúng ta không xét đến độ mịn của các đối tượng CSDL mà các giao dịch thao tác (chúng ta chỉ đơn giản gọi nó là một "đơn vị khoá"). Bây giờ chúng ta sẽ xét đến một số khả năng, và cùng với các khả năng chọn lựa mô hình giao dịch, chúng sẽ cung cấp cho chúng ta một không gian thiết kế hai chiều để cài đặt các hệ thống giao dịch.

Theo chiều cấu trúc đối tượng, chúng ta xác định các đối tượng đơn giản (thí dụ tập tin, trang, mẫu tin), các đối tượng là thể hiện của kiểu dữ liệu trừu tượng ADT, các đối tượng trưởng thành (ful – fledged) và các đối tượng năng động (active) theo độ phức tạp tăng dần.

Thảo luận trước đây của chúng ta chủ yếu đề cập đến những hệ thống hoạt tác trên các đối tượng đơn giản, phần lớn là các trang vật lý. Có những hệ thống cung cấp khả năng hoạt động đồng thời ở mức mẫu tin, nhưng chi phí thường cao và một mình các thao tác mẫu tin thì không có tính nguyên tử, đòi hỏi sự đồng bộ hoá tại mức trang. Các đặc trưng nhận biết của lớp này là các thao tác trên các đối tượng đơn giản không tận dụng ngữ nghĩa của các đối tượng. Thí dụ cập nhật một trang được xem như một thao tác Write trên trang đó mà không xem đối tượng logic nào được lưu trên trang đó.

Từ góc độ xử lý giao dịch, các ADT đặt ra một nhu cầu giải quyết với các thao tác trừu tượng như chúng ta đã thấy trong phần trước. Các thao tác trừu tượng bản thân chúng dẫn đến sự gắn kết ngữ nghĩa của chúng vào trong định nghĩa về tiêu chuẩn đúng đắn. Thực thi các giao dịch trên ADT có thể đòi hỏi một cơ chế nhiều mức. Trong những hệ thống như thế, từng giao dịch riêng lẻ biểu diễn mức trừu tượng cao nhất. Các thao tác trừu tượng được cấu tạo từ một mức trừu tượng thấp hơn và được phân ra tiếp thành các thao tác Read và Write tại mức thấp nhất. Dù tiêu chuẩn đúng đắn là gì, nó phải được áp dụng cho từng mức.

Phân biệt giữa các đối tượng là thể hiện các kiểu dữ liệu trừu tượng và các đối tượng trưởng thành để thấy rằng loại sau có một cấu trúc phức hợp (nghĩa là chứa các đối tượng khác) và kiểu của chúng (lớp) tham gia trong một đàn kiểu con (kế thừa). Chúng phải được xử lý một cách riêng rẽ vì một số điểm sau:

1. Cho chạy một giao dịch trên một đối tượng phức hợp có thể sinh ra các giao dịch trên các đối tượng thành phần. Điều này buộc phải có một *lồng ẩn (implicit nesting)* trên chính giao dịch. Quan trọng hơn, các thao tác trong các kiểu lồng này đều trừu tượng và cần phải xử lý như các giao dịch nhiều mức.

2. Việc sinh kiểu con/ kế thừa hàm chứa việc chia sẻ hành vi và/hoặc trạng thái giữa các đối tượng. Vì thế ngữ nghĩa của việc truy xuất một đối tượng tại một mức nào đó trong dàn phải tính đến điều này.

Chúng ta cũng phân biệt giữa các *đối tượng thụ động (passive) và năng động (active)*. Mặc dù phương pháp tiếp cận đối với việc quản lý các đối tượng năng động có khác nhau, tất cả các đề xuất đều giống nhau ở chỗ các đối tượng năng động có khả năng đáp ứng lại các biến cố bằng cách kích hoạt thực hiện các hành động khi một số điều kiện nào đó được thảo. Các biến cần được theo dõi, điều kiện phải được đáp ứng và hành động được thực thi đáp trả, điển hình được định nghĩa ở hình thái các quy tắc ECA (biến cố - điều kiện – hành động, event-condition-action). Vì biến cố có thể được phát hiện khi đang thực thi giao dịch trên đối tượng đó, thực hiện quy tắc tương ứng có thể phát sinh như một giao dịch lồng. Tùy theo các kết hợp các quy tắc với giao dịch ban đầu, các kiểu lồng khác nhau có thể xảy ra. Giao dịch được phát sinh có thể thực thi ngay, có thể để lại đến cuối giao dịch hoặc thực thi trong một giao dịch riêng rẽ. Vì các quy tắc bổ sung có thể làm hoạt hoá bên trong một thực thi quy tắc, các kiểu lồng với độ sâu tuỳ ý đều được. Chúng ta không bàn thêm nữa về các đối tượng năng động.

7.7.3 QUẢN LÝ GIAO DỊCH TRONG CÁC HỆ ĐỐI TƯỢNG

Như đã nói ở trên, quản lý giao dịch trong các hệ điieù hành đối tượng phải giải quyết với đồ thị hợp phần, trong đó nó trình bày cấu trúc đối tượng hợp phần và dàn kiểu (type lattice) biểu diễn mối liên hệ *isa* giữa các đối tượng.

Đồ thị hợp phần đòi hỏi các phương pháp xử lý việc đồng bộ hoá các truy xuất đến các đối tượng có chứa các đối tượng khác làm thành phần. Dàn kiểu đòi hỏi bộ quản lý giao dịch phải xem xét các vấn đề phát triển lược đồ.

Ngoài những cấu trúc này, DBMS đối tượng lưu trữ phương thức cùng với dữ liệu. Đồng bộ hoá truy xuất chung đến các đối tượng phải xét đến việc thực thi phương thức. Cụ thể là các giao dịch kích hoạt các phương thức để rồi đến lượt chúng lại có thể bị kích hoạt bởi các phương thức khác. Tất cả những yếu tố này đều gây ra những khó khăn trong việc thực thi giao dịch. Ngay cả việc định nghĩa các giao dịch tương tranh cũng trở nên rắc rối hơn. Định nghĩa tương tranh đã thảo luận trước kia có thể không còn áp dụng được trong các DBMS đối tượng. Định nghĩa tương tranh kinh điển dựa trên tính giao hoán của các thao tác truy xuất đến cùng một đối tượng. Trong các DBMS đối tượng, chúng ta có thể có các tương tranh giữa các thao tác truy xuất đến các đối tượng khác nhau. Điều này là do sự tồn tại của đồ thị hợp phần và dàn kiểu. Xét một thao tác O1 truy xuất đến đối tượng x có một đối tượng y khác làm một trong những thành phần của nó (có nghĩa x là một đối tượng hợp phần hoặc đối tượng phức). Có thể có một thao tác O2 khác (giả thiết O1 và O2 thuộc về các giao dịch khác nhau) truy xuất đến y. Theo định nghĩa điển hình về tương tranh, chúng ta không xem O1 và O2 là tương tranh vì chúng truy xuất đến các đối tượng khác nhau. Tuy nhiên O1

xem y là thành phần của x và có thể muốn truy xuất y khi đang truy xuất x, gây ra một tương tranh với O2.

Các lược đồ được phát triển cho các DBMS đối tượng phải xem xét đến những vấn đề này. Trong các đoạn còn lại của phần này chúng ta sẽ trình bày một số giải pháp đã được đề xuất.

ĐỒNG BỘ HOÁ TRUY XUẤT ĐẾN ĐỐI TƯỢNG

Lồng trong các kích hoạt phương thức có thể được dùng để phát triển các thuật toán dựa trên khoá hai pha lồng 2PL và thuật toán sắp thứ tự thời dấu. Trong quá trình này, thuộc tính của một đối tượng có thể được mô hình hoá như các phần tử dữ liệu trong CSDL, còn các phương thức được mô hình hoá như các giao dịch, cho phép nhiều kích hoạt các phương thức của một đối tượng hoạt động cùng một lúc. Điều này có thể mở rộng khả năng hoạt động đồng thời nhiều hơn.

Hệ quả là một thực thi phương thức (được mô hình hoá như một giao dịch) trên một đối tượng sẽ gồm có một số bước cục bộ (local step), tương ứng với việc thực thi các thao tác cục bộ cùng với các kết quả mà chúng trả về, và các bước phương thức (method step), đó là các kích hoạt phương thức cùng với các giá trị trả về. Một thao tác cục bộ là một thao tác nguyên tử (chẳng hạn đọc, ghi, tăng trị) có tác động lên các biến của đối tượng. Một thực thi phương thức định nghĩa một thứ tự bộ phận giữa các bước này theo cách thức thông thường.

Một trong những xu hướng cơ bản cho nghiên cứu này là cung cấp một sự tự do hoàn toàn cho các đối tượng trong cách thức chúng đạt được sự đồng bộ hoá nội đối tượng. Đòi hỏi duy nhất đó là chúng phải là các thực thi "đúng đắn" mà trong trường hợp này có nghĩa chúng phải tuần tự dựa trên

tính giao hoán. Do kết quả của việc uỷ quyền đồng bộ hoá nội đối tượng cho từng đối tượng, thuật toán điều khiển đồng thời tập trung vào việc đồng bộ hoá liên đối tượng.

Khoá chốt nhiều độ mịn định nghĩa một cây phân cấp cho các hạt CSDL có thể khoá được (vì thế có tên là "phân cấp độ mịn") như được mô tả trong hình 7.8. Trong các DBMS quan hệ, tập tin tương ứng với quan hệ và các mẫu tin tương ứng với các bộ. Trong các DBMS đối tượng, chúng lần lượt tương ứng với lớp và đối tượng thể hiện. Ưu điểm của phân cấp này là nó giải quyết những được/mất giữa khoá hạt thô và hạt mịn. Khoá hạt thô (tại mức tập tin hoặc cao hơn) có chi phí khoá thấp, vì chỉ một số nhỏ các khoá được đặt nhưng làm giảm đi nhiều khả năng đồng thời.

Hình 7.7 Nhiều độ mịn

Ý tưởng chính ẩn sau khoá chốt nhiều độ mịn đó là một giao dịch lấy khoá tại một mức thô ngầm khoá tất cả các đối tượng tương ứng ở mức mịn hơn. Thí dụ khoá tường minh tại mức tập tin gồm có khoá ngầm tất cả các mẫu tin trong tập tin đó. Để đạt được điều này, có thêm hai loại khoá được định nghĩa ngoài khoá dùng chung S (shared) và độc quyền X (exclusive) là: khoá dùng chung ẩn IS (implicit shared) và khoá độc quyền ẩn IX (implicit

exclusive). Một giao dịch muốn đặt một khoá S hoặc IS trên một đối tượng trước tiên phải đặt khoá IS trên các tổ tiên của nó (nghĩa là các đối tượng có liên quan và thô hơn). Tương tự, một giao dịch muốn đặt một khoá X hoặc IX trên một đối tượng phải đặt các khoá IX trên tất cả các tổ tiên của nó. Khoá dùng chung ẩn không được giải phóng trên một đối tượng nếu các hậu duệ của đối tượng đó hiện đang bị khoá.

Một điều phức tạp nảy sinh khi một giao dịch muốn đọc một đối tượng ở một độ mịn nào đó và sửa đổi một số các đối tượng của nó ở một độ mịn hơn. Trong trường hợp này, cả khoá S và IX phải được đặt trên đối tượng đó. Thí dụ, một giao dịch có thể đọc một tập tin và cập nhật một số mẫu tin nào đó trong tập tin (tương tự một giao dịch trong DBMS đối tượng có hể muốn đọc một định nghĩa lớp và cập nhật một số đối tượng thể hiện thuộc về lớp đó). Để giải quyết những tình huống này người ta đưa ra khoá độc quyền dùng chung ẩn SIX (shared intention exclusive), tương đương với việc giữ một khoá S và IX trên đối tượng đó. Ma trận tương thích cho khoá nhiều độ mịn được trình bày trong hình 7.8.

	S	X	OS	IX	SIS
S	+	-	+	-	-
X	-	-	-	-	-
IS	+	-	+	+	+
IX	-	-	+	+	-
SIX	-	-	+	-	-

Hình 7.8 Bảng tương thích cho khoá nhiều độ mịn

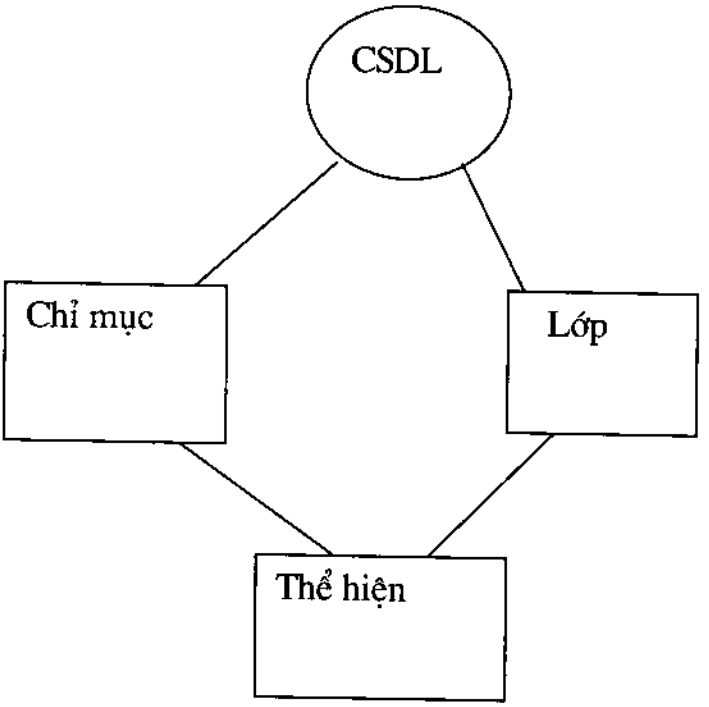
Phân cấp độ mịn trong Orion được trình bày trong hình 7.9. Thể thức khoá được hỗ trợ và sự tương thích của chúng được cho trong hình 7.8. Các

đối tượng thể hiện chỉ bị khoá với thể thức S và X, còn các đối tượng lớp có thể bị khoá với cả năm thể thức. Diễn giải các khoá này trên các đối tượng lớp như sau:

* Thể thức S. Định nghĩa bị khoá ở thể thức S và tất cả các thể hiện của nó ngầm bị khoá ở thể thức S. Điều này ngăn không cho một giao dịch khác cập nhật các thể hiện.

* Thể thức X. Định nghĩa lớp bị khoá ở thể thức X và tất cả các thể hiện của nó ngầm bị khoá ở thể thức X. Vì thế định nghĩa lớp và tất cả các thể hiện của lớp có thể được đọc hoặc được cập nhật.

* Thể thức IS. Định nghĩa lớp bị khoá ở thể thức IS và các thể hiện bị khoá ở thể thức S nếu cần.



Hình 7.9 Phân cấp độ mịn của Orion

* Thể thức IX. Định nghĩa lớp bị khoá có thể hức IS va các thể hiện bị khoá ở thể thức S hoặc X khi cần.

* Thể thức SIX. Định nghĩa lớp bị khoá ở thể thức S và tất cả các thể hiện ngầm bị khóa có thể thức S. Những thể hiện cần được cập nhật ngầm bị khoá ở thể thức X khi giao dịch cập nhật chúng.

QUẢN LÝ DÀN KIỂU

Một trong những yêu cầu quan trọng của các DBMS đối tượng là đánh giá lược đồ theo kiểu động. Hệ quả là hệ thống phải giải quyết với các giao dịch truy xuất đến các đối tượng lược đồ (nghĩa là kiểu, lớp, v.v.) cũng như các đối tượng thể hiện. Sự tồn tại của các thao tác thay đổi lược đồ hoà lẫn với các vấn tin và giao dịch thông thường cũng như mối liên hệ (đa) kế thừa được định nghĩa giữa các lớp đã làm phức tạp thêm cho hiện trạng này. Nhưng cũng có thể truy xuất các thể hiện của các lớp con của lớp đó (nghĩa là dòng tộc xa). Thứ hai trong một đối tượng hợp phần, miền của một thuộc tính cũng có hể là một lớp. Vì thế truy xuất đến một thuộc tính của một lớp có thể phải truy xuất các đối tượng trong dàn con có gốc tại lớp miền của thuộc tính đó.

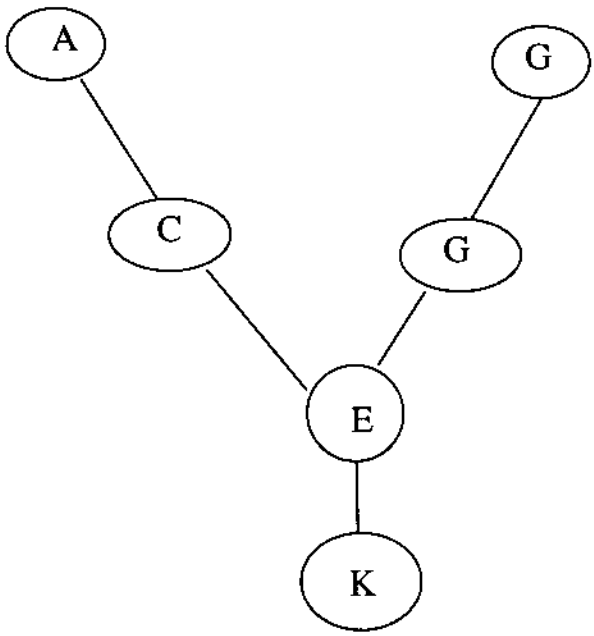
Một cách giải quyết với hai vấn đề này là lại sử dụng khoá nhiều độ mịn như trong Orion. Mở rộng đơn giản cho khoá nhiều độ mịn, ở đó lớp được truy xuất và những lớp con có nó sẽ bị khoá có thể thức thích hợp, đều không thích hợp, đều không hoạt động tốt lắm. Cách tiếp cận này là không hiệu quả khi các lớp nằm gần gốc được truy xuất vì nó hàm chứa quá nhiều khoá. Vấn đề này được giải quyết thoả đáng bằng cách đưa ra các thể thức khoá đọc trên dàn R

(read – lattice) và ghi trên dàn W (write-lattice). Chúng không chỉ khoá lớp đích ở thể thức S và X tương ứng nhưng còn ngầm khoá tất cả các lớp con của lớp đó tương ứng với các thể thức S hoặc X. Tuy nhiên giải pháp này không làm việc được với tình huống đa kế thừa, đây là vấn đề thứ ba.

Vấn đề đa kế thừa đó là một lớp có nhiều kiểu cha có thể ngầm bị khoá ở những thể thức không tương thích bởi hai giao dịch có đặt các khoá R và W trên các lớp cha khác nhau. Vì các thể thức trên các lớp chung đều thuộc loại ẩn, không có cách nào nhận ra rằng đã có một khoá trên lớp đó. Vì vậy chúng ta cần kiểm tra các lớp cha của một lớp đang được khoá. Orion giải quyết điều này bằng cách đặt các khoá hiển chứ không phải khoá ẩn trên các lớp con. Xét dàn kiểu của hình 7.11được lấy từ (arza and Kim, 1988) và đã được đơn giản hoá. Nếu giao dịch T1 đặt một khoá IR trên lớp A và một khoá R trên C, nó cũng sẽ đặt một khoá hiển R trên E. Khi một giao dịch T2 khác đặt một khoá IW trên F và một khoá W trên G, nó sẽ thử đặt một khoá hiển W trên E, tuy nhiên vì đã có một khoá R trên E, yêu cầu này bị từ chối.

Một chọn lựa khác là đặt các khoá ở một độ mịn hơn, sử dụng phương pháp dùng chung có thứ tự. Theo một nghĩa nào đó, thuật toán này là một mở rộng của lối tiếp cận dựa trên tính giao hoán của Weihl đối với các DBMS đối tượng bằng cách dùng một mô hình giao dịch lồng.

Các lớp được mô hình hoá như các đối tượng trong hệ thống tương tự như các hệ phản thân biểu diễn các đối tượng lược đồ như các đối tượng hạng nhất. Hệ quả là các phương thức có thể được định nghĩa đều hoạt tác trên các đối tượng lớp: add (m) đưa phương thức m vào lớp, del (m) xoá phương thức m ra khỏi lớp, rep (m) thay thế cài đặt của phương thức m bằng một cài đặt khác, và use (m) cho thực thi phương thức m. Tương tự, các thao tác nguyên tử cũng được định nghĩa để truy xuất các thuộc tính của một lớp.



Hình 7.10 Một dàn kiểu làm thí dụ

Chúng giống với các thao tác phương thức nhưng có sự thay đổi thích hợp về ngữ nghĩa nhằm phản ánh việc truy xuất thuộc tính. Điểm đáng chú ý ở đây là định nghĩa của thao tác use (a) cho thuộc tính a chỉ ra rằng việc truy xuất của một giao dịch đến thuộc tính a bên trong một thực thi phương thức sẽ được thông qua thao tác use. Điều này đòi hỏi rằng mỗi phương thức phải liệt kê rõ tất cả các thuộc tính mà nó truy xuất. Vì vậy dưới đây là dãy các bước một giao dịch cần phải tuân theo khi cho thực thi một phương thức m:

1. Giao dịch T đưa ra thao tác use (m).
2. Với mỗi thuộc tính A được phương thức m truy xuất, T đưa ra thao tác use (A).

Giao dịch T khởi hoạt phương thức m.

Các bảng giao hoán đã được định nghĩa cho các thao tác phương thức và thuộc tính. Dựa trên các bảng giao hoán, bảng khoá dùng chung có thứ tự cho mỗi thao tác nguyên tử được xác định. Đặc biệt, một khoá cho một thao tác nguyên tử p có một mối liên hệ dùng chung với tất cả các khoá được đi kèm với các thao tác mà p có một mối liên hệ không tương tranh, trong khi đó nó có một mối liên hệ dùng chung có thứ tự ứng với tất cả các khoá đi kèm với những thao tác mà p có một liên hệ tương tranh.

Dựa trên các bảng khoá này, một thuật toán khoá 2PL lồng được dùng với những điều cần xem xét sau đây:

1- Các giao dịch nhận biết quy tắc 2PL nghiêm ngặt và duy trì các khoá của chúng cho đến khi kết thúc.

2- Khi một giao dịch huỷ bỏ, nó giải phóng tất cả các khoá của nó.

3- Việc kết thúc của một giao dịch buộc phải chờ đợi sự kết thúc của các con của nó (ngữ nghĩa lồng đóng). Khi một giao dịch uỷ thác, các khoá của nó được kế thừa bởi cha của nó.

4- Quy tắc uỷ thác có thứ tự. Cho trước hai giao dịch Ti và Tj sao cho Ti đang đợi Tj, Ti không thể uỷ thác các thao tác của nó trên một đối tượng bất kỳ cho đến khi Tj kết thúc (uỷ thác hoặc huỷ bỏ), Ti được xem là đang đợi Tj nếu:

* Ti không phải là gốc của giao dịch lồng và Ti đã được trao một khoá với mối liên hệ dùng chung có thứ tự ứng với một khoá được giữ bởi Tj trên một đối tượng sao cho Tj là một hậu duệ của cha của Ti, hoặc

* Ti là gốc của một giao dịch lồng và Ti giữ một khoá (được kế thừa hoặc được trao) trên một đối tượng trong mối liên hệ dùng chung có thứ tự ứng với một khoá được Tj hoặc các hậu duệ của nó giữ.

QUẢN LÝ ĐỒ THỊ HỢP PHẦN

Các nghiên cứu giải quyết đồ thị hợp phần thì phổ biến hơn. Yêu cầu các DBMS đối tượng mô hình hoá các đối tượng hợp phần bằng một phương cách hiệu quả đã gây nhiều chú ý về bài toán này.

Chúng ta bắt đầu thảo luận bằng một sơ thảo về cách tiếp cận của Orion đối với việc quản lý cây phân cấp hợp phần mà một lần nữa nó lại dựa trên khoá nhiều độ mịn. Hai khả năng đều được xác định. Như đã gợi ý trong phần trước, một là khoá một đối tượng hợp phần và tất cả các lớp của các đối tượng thành phần của nó. Đây rõ ràng là điều không thể chấp nhận được vì nó hàm chứa rằng phải khoá toàn bộ cây đối tượng hợp phần, vì thế hạn chế hiệu năng rất nhiều. Chọn lựa thứ hai là khoá các thể hiện đối tượng thành phần bên trong một đối tượng hợp phần. Trong trường hợp này chúng ta cần giăng bắt tất cả các tham chiếu và khoá tất cả những đối tượng này. Điều này hoàn toàn ngớ ngẩn vì nó phải khoá quá nhiều đối tượng.

Vấn đề nằm ở chỗ là nghi thức khoá nhiều độ mịn không xem đối tượng hợp phần là một đơn vị khoá được. Để giải quyết được vấn đề này người ta đã đưa ra ba thể thức khoá mới: ISO, IXO và SIXO, tương ứng với IS, IX và SIX. Những thể thức khoá này được dùng để khoá các lớp thành phần của một đối tượng hợp phần. Tính tương thích của chúng được trình bày trong hình 7.11. Nghi thức cần dùng như sau: để khoá một đối tượng

hợp phần lớp gốc bị khoá ở thể thức X, IS, IX hoặc SIX, và mỗi lớp thành phần của cây phân cấp đối tượng hợp phần bị khoá lần lượt ở thể thức X, ISO, IXO và SIXO.

	S	X	IS	IX	SIX	ISO	IXO	SIXO
S	+	-	+	-	-	+	-	-
X	-	-	-	-	-	-	-	-
IS	+	-	+	+	+	+	-	-
IX	-	-	+	+	-	-	-	-
SIX	-	-	+	-	-	-	-	-
ISO	+	-	+	+	-	+	+	+
IXO	-	-	-	-	-	+	+	-
SIXO	N	N	N	N	N	Y	N	N

Hình 7.11 Ma trận tương thích cho các đối tượng hợp phần

Các thuật toán điều khiển đồng thời, dựa trên khoá nhiều độ mịn sẽ cưỡng chế tính khả tuần tự. Thuật toán khôi phục được dùng là ghi nhật ký bằng phương pháp no-fix/flush, chỉ đòi hỏi phải hồi lại nhưng không thực hiện lại.

Một mở rộng của khoá nhiều độ mịn để giải quyết với đồ thị hợp phần được thảo luận trong Herrmann 1990. Mở rộng phải thực hiện là thay thế một đồ thị khoá tĩnh duy nhất bằng một phân cấp các đồ thị đi kèm với mỗi kiểu và vấn tin. Có một "đồ thị khoá tổng quát" lo điều khiển toàn bộ quá trình này. Đơn vị nhỏ nhất có thể khoá được gọi là đơn vị khoá cơ bản BLU (basic lockable unit). Một số các BLU có thể tạo ra một đơn vị khoá đồng chủng HoLU (homogeneous lockable unit) cấu tạo bởi các dữ liệu có

cùng kiểu. Tương tự chúng có thể cấu tạo nên một đơn vị khoá đa chủng HeLU (heterogeneouss lockable unit) cấu tạo bởi các đối tượng thuộc về những kiểu khác nhau. HeLU có thể chứa các HeLU khác hoặc HoLU, chỉ ra rằng các đối tượng thành phần không phải đều là nguyên tử. Tương tự HoLU có thể chứa những HoLU hoặc HeLU khác, với điều kiện chúng phải cùng kiểu. Phân biệt giữa HoLU và HeLU là nhằm để tối ưu hoá các yêu cầu khoá. Thí dụ một tập các danh sách số nguyên, từ quan điểm bộ quản lý khoá, sẽ được xử lý như một HoLU do nhiều HoLU tạo ra, đến lượt nó lại được cấu tạo bởi các BLU. Kết quả là chúng ta có thể khoá toàn bộ tập, chỉ một trong các danh sách hoặc thậm chí một số nguyên.

Vào lúc định nghĩa kiểu, một đồ thị khoá cụ thể cho đối tượng sẽ được tạo ra và tuân theo đồ thị khoá tổng quát. Giống như một thành phần thứ ba, một đồ thị khoá cụ thể cho vấn tin được tạo ra trong khi phân tích vấn tin. Khi thực thi vấn tin (giao dịch), đồ thị khoá cụ thể cho vấn tin được dùng để yêu cầu khoá từ bộ quản lý khoá và nó sử dụng đồ thị khoá cụ thể cho đối tượng để đưa ra quyết định. Các thể thức khoá được dùng là những thể thức chuẩn (nghĩa là IS, IX, S, X).

Badrinath and Ramamritham, 1988 đã thảo luận về một khả năng khác để giải quyết với cây phân cấp đối tượng hợp phần dựa trên tính giao hoán. Một số các thao tác khác nhau được định nghĩa trên đồ thị hợp phần:

(1) Xem xét nội dung của một đỉnh (đó là một lớp).

(2) Xem xét một cạnh (mối liên hệ cấu tạo nên).

(3) Chèn một đỉnh và cạnh đi kèm.

(4) Xoá một đỉnh và cạnh đi kèm.

(5) Chèn một cạnh.

Chú ý rằng một số trong những thao tác này (1) và (2) tương ứng với

các toán tử đối tượng đã có, còn những thao tác khác (3 – 5) biểu diễn các thao tác lược đồ.

Dựa trên những thao tác này, tập bị ảnh hưởng (affected set) được định nghĩa cho các đồ thị độ mịn, làm cơ sở cho việc xác định những thao tác nào có thể được thực thi đồng thời. Tập bị ảnh hưởng của một đồ thị độ mịn là hợp của:

* Tập cạnh, là tập các cặp (e, a) trong đó e là một cạnh và a là một thao tác có ảnh hưởng đến e và có thể là một trong các thao tác insert, delete, examine.

* Tập đỉnh, là tập các cặp (v, a) trong đó v là một đỉnh và a là một thao tác có ảnh hưởng đến v và có thể là một trong các thao tác insert, delete, examine hoặc modify.

Sử dụng tập bị ảnh hưởng sinh bởi hai giao dịch Ti và Tj của một đồ thị hợp phần người ta có thể định nghĩa xem liệu Ti và Tj có thể được thực thi đồng thời hay không. Tính giao hoán được dùng như cơ sở của quan hệ tương tranh. Vì thế hai giao dịch Ti và Tj là giao hoán trên đối tượng K nếu:

$$\text{affected-set (Ti)} \cap K \text{ affected-set(Tj)} = \theta$$

Những nghi thức này thực hiện đồng bộ hoá trên cơ sở của các đối tượng không phải các thao tác trên chúng. Chúng ta cũng có thể cải thiện tính đồng thời bằng cách phát triển các kỹ thuật làm đồng bộ hoá các khởi hoạt thao tác thay vì phải khoá toàn bộ các đối tượng.

Một cách tiếp cận khác dựa trên ngữ nghĩa được mô tả trong Muth 1993. Đặc trưng của cách tiếp cận này là:

(1) Truy xuất đến các đối tượng thành phần có thể được phép mà không phải đi qua cây phân cấp các đối tượng (nghĩa là không khoá nhiều độ mịn).

(2) Ngữ nghĩa của các thao tác được xem xét bằng một đặc tả ưu tiên

về các giao hoán phương thức.

(3) Phương thức được khởi hoạt bởi một giao dịch có thể khởi hoạt những phương thức khác. Điều này dẫn đến một thực thi giao dịch lồng (động) mặc dù giao dịch về cú pháp là phẳng.

Mô hình giao dịch được dùng để hỗ trợ (3) thuộc loại lồng mở, đặc biệt là giao dịch nhiều mức được mô tả. Các hạn chế được đặt ra trên kiểu lồng động của các giao dịch là:

* Tất cả các cặp thao tác (p, g) có khả năng tương tranh trên cùng một đối tượng đều có cùng độ sâu trong các cây khởi hoạt cuả chúng.

* Với mỗi cặp tổ tiên (f, g') của f và g có độ sâu của các cây khởi hoạt đều như nhau, f' và g' hoạt tác trên cùng một đối tượng.

Với những hạn chế này, thuật toán hoàn toàn đơn giản. Một khoá ngữ nghĩa được liên kết với mỗi phương thức, và một bảng giao hoán định nghĩa xem các khoá ngữ nghĩa khác nhau có tương thích với nhau hay không. Các giao dịch thu nhận các khoá ngữ nghĩa này trước lúc khởi hoạt phương thức và chúng được giải phóng vào cuối giao dịch con (phương thức) đưa ra kết quả của chúng cho những giao dịch khác. Tuy nhiên cha của các giao dịch con đã uỷ thác có một khoá ngữ nghĩa mức cao hơn, chỉ cho nhìn thấy các kết quả của các giao dịch con đã uỷ thác đối với những giao dịch giao hoán với gốc của giao dịch con. Điều này đòi hỏi phải định nghĩa hành động kiểm tra tương tranh ngữ nghĩa, hoạt tác trên các cây phân cấp khởi hoạt bằng cách dùng các bảng tương thích.

Một vấn đề phức tạp nảy sinh ứng với hai điều kiện được phác thảo ở trên sẽ không hợp lý khi hạn chế khả năng áp dụng của nghi thức chỉ cho những tình huống những điều kiện này đúng. Để giải quyết khó khăn này, giải pháp được đề xuất là từ bỏ một số tính chất mở và biến đổi các khoá phải giải phóng vào cuối giao dịch con thành các khoá giữ lại (retained lock)

và trao cho cha của nó. Trong một số tình huống, khoá giữ lại có thể được huỷ đi để làm tăng khả năng hoạt động đồng thời.

Một cách tiếp cận rất gần gũi nhưng có nhiều hạn chế hơn được thảo luận trong Weikum and Hasse, 1993. Mô hình giao dịch nhiều mức được dùng nhưng bị hạn chế chỉ trong hai mức: mức đối tượng và mức trang. Vì thế kiểu lồng động xảy ra khi các giao dịch khởi hoạt các phương thức rồi chúng lại kích hoạt các phương thức sẽ không được xét đến. Sự tương tự với nghiên cứu ở trên nằm ở chỗ các khoá mức trang được giải phóng vào lúc cuối giao dịch con, còn các khoá mức đối tượng (về ngữ nghĩa phong phú hơn) được giữ lại cho đến khi giao dịch chấm dứt.

Trong cả hai cách tiếp cận vừa bàn (Muth 1993 và Weikum and Hasse, 1993), khôi phục có thể được thực hiện bởi các nghi thức hướng trạng thái ở mức trang. Vì các giao dịch con giải phóng khoá của chúng và cho thấy kết quả của chúng, các giao dịch bù phải có thực hiện các hành động "hồi lại" những kết quả của các giao dịch con đã uỷ thác.

7.7.4 GIAO DỊCH VÀ ĐỐI TƯỢNG

Một đặc trưng quan trọng của mô hình dữ liệu quan hệ là không có ngữ nghĩa rõ ràng về thao các cập nhật. Mô hình này, như chúng ta đã biết, cách truy tìm dữ liệu trong một CSDL thông qua các toán tử đại số quan hệ nhưng thực sự không mô tả một hành động cập nhật có nghĩa là gì. Kết quả là các định nghĩa và các kỹ thuật quản lý giao dịch đều chẳng có liên hệ gì với mô hình dữ liệu quan hệ. Chúng ta có thể và thường áp dụng các kỹ thuật tương tự cho các DBMS phi quan hệ, thậm chí cho cả các hệ thống lưu trữ không phải CSDL.

Đặc điểm của các DBMS đối tượng, chẳng hạn như các cấu trúc kiểu (lớp) các đối tượng hợp phần và các nhóm đối tượng (dòng tộc) được xem xét bằng các kỹ thuật về cơ bản là như nhau.

Người ta nhận thấy rằng trong các DBMS đối tượng, thực sự có ưu điểm quyết định là việc lồng các ngữ nghĩa cập nhật vào bên trong mô hình đối tượng. Các lập luận này là:

1- Trong các DBMS đối tượng, cái được lưu không chỉ là dữ liệu mà còn là các thao tác trên dữ liệu (được gọi là phương thức, hành vi hoặc thao tác trong nhiều mô hình đối tượng khác nhau). Các vấn tin truy xuất một CSDL đối tượng tham chiếu đến những thao tác như thành phần trong các vị từ của chúng. Nói cách khác, việc thực thi các vấn tin này khởi hoạt nhiều thao tác khác nhau được định nghĩa trên các lớp. Để đảm bảo tính an toàn của các biểu thức vấn tin, các phương pháp xử lý vấn tin hiện có đều hạn chế những thao tác này để không có các hiệu ứng phụ, với tác dụng là ngăn chúng cập nhật CSDL. Đây là một hạn chế khắt khe cần được giảm bớt bằng cách kết hợp ngữ nghĩa cập nhật vào các định nghĩa an toàn vấn tin.

2- Như chúng ta đã thảo luận trong mục trước, giao dịch trong các DBMS đối tượng có ảnh hưởng đến các dàn kiểu (lớp). Vì thế có một mối liên hệ trực tiếp giữa việc phát triển lược đồ động và việc quản lý giao dịch. Nhiều kỹ thuật đã thảo luận tận dụng khoá chốt trên dàn này thích ứng được với những thay đổi. Tuy nhiên các khoá chốt (ngay cả khoá nhiều độ mịn) lại hạn chế nặng nề đối với khả năng hoạt động đồng thời. Một định nghĩa xem cập nhật CSDL có nghĩa là gì, và một định nghĩa về tương tranh dựa trên định nghĩa này của ngữ nghĩa cập nhật sẽ cho phép nhiều khả năng đồng thời hơn.

Điều đáng chú ý là mối liên hệ giữa những thay đổi đối với dàn kiểu và việc xử lý vấn tin. Với sự vắng mặt của một định nghĩa rõ ràng về ngữ nghĩa cập nhật và việc kết nối nó vào phương pháp luận xử lý vấn tin, phần

lớn các bộ xử lý vấn tin hiện tại đều giả thiết rằng lược đồ CSDL (nghĩa là dàn kiểu) thuộc loại tĩnh trong khi thực thi một câu vấn tin.

3- Có một số mô hình đối tượng (thí dụ OODAPLEX Dayal, 1989 và TIGUKAT, 1995) xử lý tất cả các thực thể của hệ thống như các đối tượng. Theo lối tiếp cận này, cách tự nhiên nhất là mô hình hoá các giao dịch như đối tượng. Tuy nhiên vì cơ bản giao dịch là các kết cấu làm thay đổi trạng thái của CSDL, tác dụng của chúng trên CSDL phải được mô tả rõ ràng.

Bên trong ngữ cảnh này, chúng ta cũng cần chú ý rằng miền các ứng dụng cần đến các dịch vụ của DBMS đối tượng có xu hướng đặt ra các yêu cầu quản lý giao dịch có hơi khác, theo cả mô hình giao dịch và ràng buộc nhất quán. Mô hình hoá giao dịch như các đối tượng cho phép áp dụng các kỹ thuật đối tượng đã biết (chuyên biệt hoá và sinh kiểu con) để tạo ra các loại hệ thống quản lý giao dịch khác nhau. Điều này đưa ra được khả năng mở rộng hệ thống.

4- Một số yêu cầu cần đến sự hỗ trợ bằng quy tắc và khả năng của các CSDL năng động (active). Bản thân các quy tắc được thực thi như giao dịch mà chúng lại sinh ra các giao dịch mới. Người ta cho rằng các quy tắc cần được mô hình hoá như đối tượng. Như vậy một số giao dịch cũng cần được mô hình hoá như các đối tượng.

Theo kết quả của những điểm trên đây, phương pháp tiếp cận đối với một hệ thống quản lý giao dịch hoàn toàn khác với những vấn đề đã được thảo luận trong chương này. Đây là những tiền đề cần được nghiên cứu.

Thực tế trong CSDL nói chung và CSDL đối tượng nói riêng chúng ta còn phải đầu tư thời gian và tiếp tục nghiên cứu nhiều.

Tuy nhiên chúng ta có thể rút ra một vài kết luận sau khi đã nghiên cứu qua nội dung chương 7 này:

(1) Khi công nghệ và ứng dụng CNTT đã vượt ra ngoài những bài toán

quản lý thương mại truyền thống thì việc chuyển hướng tiếp cận theo đối tượng là cần thiết và bắt buộc.

(2) Mô hình đối tượng là mô hình bao tất cả các mô hình thực thể, quan hệ.v.v.

(3) Trong môi trường phân tán đối tượng cần phải có hệ quản trị đối tượng phân tán thíc hợp.

(4) Một xu hướng tiếp cận theo điểm (3) là hiện nay trên thị trường đã xuất hiện DBMS quan hệ - đối tượng. Hệ thống này tạo thuận lợi cho người dùng đã quen với mô hình quan hệ.

(5) Tất nhiên những mô hình theo (4) về thực chất chưa có nhiều khác biệt về chất so với mô hình quan hệ.

BÀI TẬP

7.1. Giải thích các cơ chế được dùng để hỗ trợ tính bao gói trong các DBMS đối tượng phân tán. Cụ thể là:

a) Mô tả xem làm thế nào việc bao gói được che dấu, không cho người sử dụng thấy khi cả đối tượng và phương thức được phân tán.

b) Một DBMS đối tượng phân tán trình bày một lược đồ toàn cục duy nhất cho người sử dụng như thế nào? Điều này khác với việc hỗ trợ tính vô hình phân mảnh trong các hệ CSDL quan hệ như thế nào?

7.2. Liệt kê các bài toán phân tán đối tượng mới nảy sinh trong các DBMS đối tượng là những vấn đề không hiện diện trong các DBMS quan hệ ứng với việc phân mảnh, di trú và nhân bản.

7.3.** Phân hoạch các CSDL đối tượng là tiền đề của việc làm giảm đi các truy xuất dữ liệu không liên quan cho các ứng dụng. Hãy phát triển một mô hình chi phí để thực thi các vấn tin trên các CSDL đối tượng không phân hoạch và các CSDL đối tượng phân hoạch ngang hoặc dọc. Sử dụng mô hình chi phí đó để minh hoạ các tình huống phân hoạch không làm giảm đi các truy xuất dữ liệu không liên quan.

7.4. Trình bày mối liên hệ giữa làm tụ và phân hoạch. Minh hoạ xem làm tụ có thể làm giảm hoặc cải thiện hiệu năng của các vấn tin trên các hệ CSDL đối tượng có phân hoạch hay không.

7.5. Vì sao các DBMS đối tượng khách/ đại lý chủ yếu lại dùng kiến trúc chuyển gửi dữ liệu còn các DBMS quan hệ lại dùng cách chuyển gửi chức năng?

7.6. Thảo luận về những điểm mạnh và điểm yếu của các đại lý trang và đại lý đối tượng ứng với các vấn đề di chuyển dữ liệu, quản lý vùng trữ, tính nhất quán vùng trữ, và cơ chế điều chế con trỏ.

7.7. Có khác biệt gì giữa thông tin trữ ở chỗ khách và nhân bản dữ liệu?

7.8. Một lớp ứng dụng mới mà DBMS đối tượng hỗ trợ là loại ứng dụng tương tác và giải quyết với các đối tượng lớn (thí dụ các hệ đa phương tiện tương tác). Thuật toán nhất quán vùng trữ nào được trình bày trong chương này thích hợp với lớp ứng dụng hoạt động trên các mạng diện rộng?

7.9.** Cơ chế điều chế con trỏ phần cứng và phần mềm có những điểm mạnh và điểm yếu bù trừ nào. Đề xuất một cơ chế điều chế con trỏ lai có chứa những điểm mạnh của cả hai.

7.10.** Giải thích xem làm sao dùng được phân mảnh ngang dẫn xuất để tạo dễ dàng cho các vấn tin đường dẫn hiệu quả trong các DBMS đối tượng phân tán và cho thí dụ.

7.11.** Cho biết một số heuristic mà một thể tối ưu hoá vấn tin trên các vấn tin OQL trong một DBMS đối tượng có thể dùng để xác định cách thức phân rã một vấn tin, nhờ đó một số phần có thể được chuyển gửi theo kiểu chức năng còn một số khác được thực thi tại chỗ khách đã đưa ra vấn tin bằng cách chuyển gửi dữ liệu.

7.12.** Ba cách khác nhau để thực hiện tổng hợp đối tượng phức phân tán đã được thảo luận trong chương này. Đưa ra một thuật toán cho chọn lựa, trong đó các thao tác phức như nối và tổng hợp đầy đủ các đối tượng ở xa được thực hiện ở các vị trí ở xa và kết quả từng phần được chuyển đến vị trí trung tâm để tổng hợp cuối cùng.

7.13*. Xét thí dụ đặt chỗ máy bay của Chương 6. Hãy định nghĩa một lớp Reservation (kiểu) và đưa ra các ma trận giao hoán tới và lùi cho nó.

TÀI LIỆU THAM KHẢO

1. M. Tamer Ôzsu, Patrick Valduriez. Nguyên lý các hệ cơ sở dữ liệu phân tán. NXB Thống kê, 2000.
2. J. Ulman. Nguyên lý các hệ cơ sở dữ liệu và cơ sở tri thức. Tập 1, 2 . NXB Thống kê, 1998.
3. Fred R. Mcfaden , Jfrey A. Hoffer. Modern Database Management. The Benjamin/Cummings. Publishing Company.INC 1993.
4. Vũ Đức Thi. Cơ sở dữ liệu – kiến thức và thực hành" .NXB Thống kê, 1997.
5. Đỗ Trung Tuấn. Cơ sở dữ liệu. NXB Giáo dục 1998.
6. Nguyễn Bá Tường. Lý thuyết về cơ sở dữ liệu. Giáo trình Học viện Kỹ thuật quân sự, 2000.
7. Nguyễn Bá Tường. Cơ sở dữ liệu – lý thuyết và thực hành. NXB Khoa học và Kỹ thuật, 2001.
8. Lê Tiến Vương. Nhập môn cơ sở dữ liệu quan hệ. NXB Khoa học và Kỹ thuật, 1997.
9. Nguyễn Văn Hoàng. Tự học Microsoft SQL SEVER 7.0". NXB Thống kê, 2000.
10. Phùng Kim Hoàng. Kiến thức thiết yếu về mạng máy tính. NXB Đà Nẵng, 2000.
11. J. Martin. Organizacja Baz Danych. Warszawa, 1983.
12. G. Coulouris, J. Dollimore, Tim Kindberg. Distributed Systems - Concepts and Design. Addison –Wesley. Publishing Company, 1994.
13. Hồ Thuần, Nguyễn Quang Vinh, Nguyễn Xuân Huy dịch. Nhập môn các hệ cơ sở dữ liệu. NXB Thống kê, 1986.
14. David W. Embley. Object Database Development - concepts and Principles. NXB Addison Wesley Longman, Inc., 1997.
15. Gary J. Nutt. Centralized and Distributed Operating Systems. Prentice-Hall International, Inc.
16. D. McGOVERAN, C.J. DATE. A gui to SYBASE and SQL Server. NXB Addison Wesley.

MỤC LỤC

CHƯƠNG 6. QUẢN LÝ GIAO DỊCH VÀ ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN

TS. NGUYỄN BÁ TƯỜNG

NHẬP MÔN
CƠ SỞ DỮ LIỆU PHÂN TÁN

Chịu trách nhiệm xuất bản : PGS.TS. TÔ ĐĂNG HẢI
Biên tập : ĐỖ TIẾN KHANG, KIM ANH
Vẽ bìa : HƯƠNG LAN

NHÀ XUẤT BẢN KHOA HỌC VÀ KỸ THUẬT
70 Trần Hưng Đạo, Hà Nội

6.6 T 7.3
KHKT-2004

In 700 cuốn, khổ 16 x 24cm, tại Nhà in Hà Nội thuộc Công ty Phát hành sách Hà Nội.
Giấy phép số 6-301 cấp ngày 5/1/2004. In xong và nộp lưu chiểu quý I/2005.

nhập môn cơ sở dữ liệu phân

Giá: 80.000đ