Co So Du Lieu Phan Tan - Cơ sở dữ liệu phân tán

Accounting Information System (Trường Đại học Kinh tế Thành phố Hồ Chí Minh)

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI GIẢNG
CƠ SỞ DỮ LIỆU PHÂN TÁN

(Dùng cho sinh viên hệ chính qui)

LƯU HÀNH NỘI BỘ

Biên soạn: TS. TÂN HẠNH

TP. HỒ CHÍ MINH 2010

MỤC LỤC

MỤC LỤC HÌNH

Chương 1 ĐẠI CƯƠNG VỀ CƠ SỞ DỮ LIỆU PHÂN TÁN

MỤC TIÊU

Chương này trình bày các khái niệm của cơ sở dữ liệu phân tán:định nghĩa cơ sở dữ liệu phân tán, lý do cần có cơ sở dữ liệu phân tán, các tính chất đặc trưng của nó. Từ đó so sánh cơ sở dữ liệu phân tán với cơ sở dữ liệu tập trung qua đó rút ra những lý do để phát triển một hệ thống dựa trên cơ sở dữ liệu phân tán. Giới thiệu về cơ sở dữ liệu phân tán

1.1 Giới thiệu

Trong những năm gần đây, cơ sở dữ liệu phân tán đã trở thành một lĩnh vực xử lý thông tin quan trọng và chúng ta dễ dàng nhận ra tầm quan trọng của nó ngày càng lớn mạnh. Chúng ta có lý do về tổ chức cũng như về kỹ thuật để phát triển theo xu hướng này: cơ sở dữ liệu phân tán khắc phục được một số hạn chế của cơ sở dữ liệu tập trung như quá tải server, nghẽn cổ chai khi truy xuất, tính sẵn sang/ độ tin cậy về khả năng chịu lỗi thấp. Hơn nữa cơ sở dữ liệu phân tán phù hợp hơn với các tổ chức dữ liệu phi tập trung cũng như với các ứng dụng phân tán.

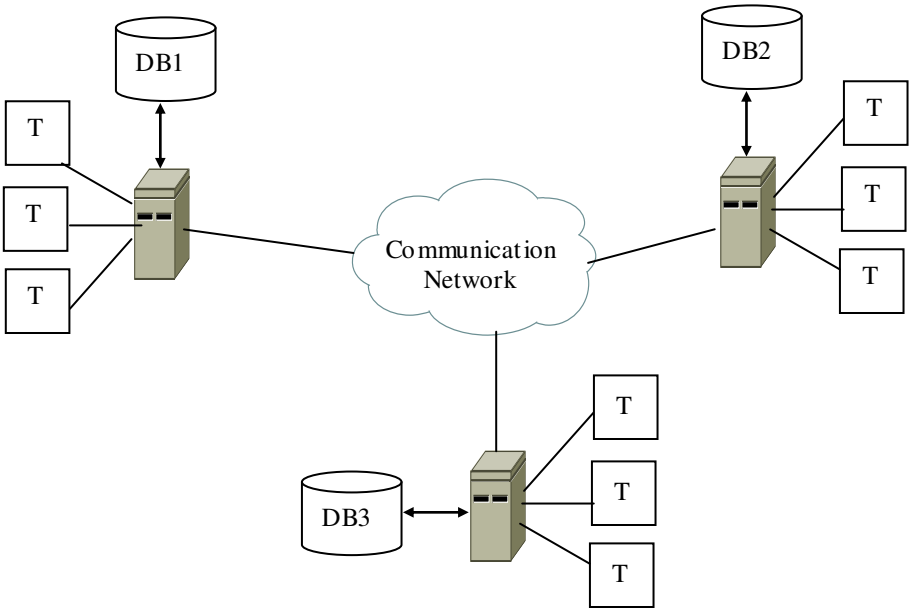
Chúng ta có thể xem cơ sở dữ liệu phân tán là một tập hợp dữ liệu (cơ sở dữ liệu) của một hệ thống thông tin nhưng được phân bố trên nhiều địa điểm (site) của một mạng máy tính (intranet). Khái niệm này nhấn mạnh đến hai khía cạnh quan trọng của cơ sở dữ liệu phân tán là :

1. Sự phân tán: dữ liệu không lưu trữ trên cùng một địa điểm vì thế chúng ta có thể phân biệt nó với cơ sở dữ liệu tập trung.

2. Mối tương quan luận lý (logical correlation): Các dữ liệu có một số thuộc tính ràng buộc với nhau từ các cơ sở dữ liệu cục bộ mà được lưu trữ tại các địa điểm khác nhau trên mạng.

Ví dụ : Xét một ngân hàng có ba chi nhánh nằm ở ba nơi khác nhau (hình 1.1). Tại mỗi nhánh, một hệ thống máy tính điều khiển các trạm thu hay rút tiền và quản lý cơ sở dữ liệu về tài khoản. Mỗi hệ thống này có cơ sở dữ liệu tài khoản cục bộ tạo

thành một site của cơ sở dữ liệu phân tán. Các hệ thống máy tính này được kết nối bởi một mạng truyền thông. Với những hoạt động thông thường, các yêu cầu từ các trạm chỉ cần truy xuất đến cơ sở dữ liệu tại chi nhánh của chúng. Vì thế ứng dụng này được gọi là ứng dụng cục bộ.



Hình 2.1 Cơ sở dữ liệu phân tán của ngân hàng có ba chi nhánh

Ví dụ trên làm nảy sinh hai câu hỏi sau:

1) Mỗi chi nhánh chỉ lưu trữ cơ sở dữ liệu cục bộ có đủ đáp ứng các ứng dụng chưa?

2) Cơ sở dữ liệu phân tán có phải là một tập hợp các cơ sở dữ liệu cục bộ?

Để trả lời các câu hỏi này chúng ta tìm hiểu xem việc xử lý trên cơ sở dữ liệu cục bộ khác gì trên cơ sở dữ liệu phân tán. Về mặt kỹ thuật, chúng ta thấy cần có các ứng dụng mà truy xuất dữ liệu đặt ở nhiều nhánh. Các ứng dụng này được gọi là ứng dụng toàn cục hay ứng dụng phân tán.

Một ứng dụng toàn cục thông thường trong ví dụ trên là việc chuyển tiền từ một tài khoản nầy đến tài khoản khác. Ứng dụng này yêu cầu cập nhật cơ sở dữ liệu ở cả hai nhánh.

Hơn nữa ứng dụng toàn cục giúp cho người sử dụng không phân biệt được dữ liệu đó cục bô hay từ xa. Đó là tính trong suốt dữ liệu trong cơ sở dữ liệu phân tán. Và đương nhiên khi ứng dụng toàn cục truy cập dữ liệu cục bộ sẽ nhanh hơn ứng dụng từ xa điều này nói lên sự nhân bản dữ liệu ở các nơi cũng làm tăng tốc độ xử lý chương trình.

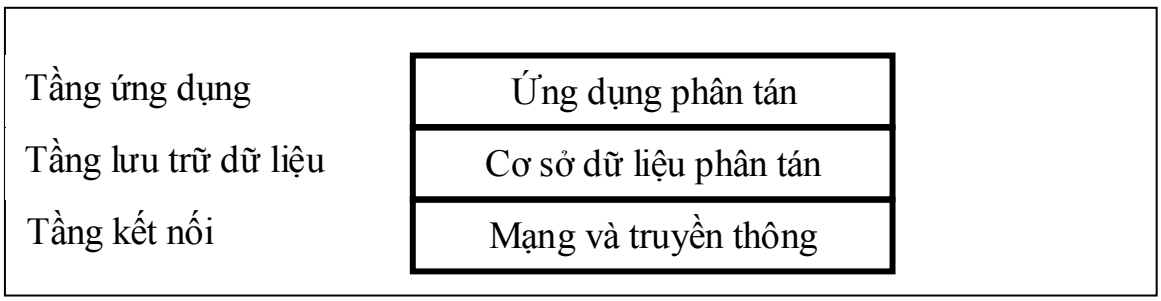
1.2 Định nghĩa về cơ sở dữ liệu phân tán

Một cơ sở dữ liệu phân tán là tập hợp dữ liệu quan hệ lẫn nhau một cách luận lý trên cùng một hệ thống nhưng được trãi rộng trên nhiều sites của một mạng máy tính.

Mỗi site có quyền tự quản cơ sở liệu cục bộ của mình và thực thi các ứng dụng cục bộ. Mỗi vị trí cũng phải tham gia vào việc thực thi ít nhất một ứng dụng toàn cục mà yêu cầu truy xuất dữ liệu tại nhiều vị trí qua mạng.

Hình ảnh của cơ sở dữ liệu phân tán (hình 1.2) minh họa mối quan hệ của cơ sở dữ liệu phân tán với môi trường kết nối mạng máy tính và các ứng dụng phân tán.

Để làm rõ các điểm đặc trưng của cơ sở dữ liệu phân tán hãy so sánh nó với cơ sở dữ liệu tập trung trong phần sau.



Hình 1.2 Mối liên hệ giữa mạng máy tính, cơ sở dũ liệu phân tán và ứng dụng phân tán

1.3 Các điểm đặc trưng của cơ sở dữ liệu phân tán so với cơ sở dữ liệu tập trung

Cơ sở dữ liệu phân tán không đơn giản là việc phân tán các cơ sở dữ liệu tập trung bởi vì nó cho phép thiết kế các hệ thống có các tính chất khác với hệ thống tập trung truyền thống. Vì thế nên xem lại các tính chất đặc trưng của cơ sở dữ liệu tập trung

truyền thống và so sánh nó với các tính chất của cơ sở dữ liệu phân tán. Các tính chất đặc trưng của cơ sở dữ liệu tập trung là điều khiển tập trung, độc lập dữ liệu, chuẩn hóa để loại bỏ sự dư thừa dữ liệu, các cấu trúc lưu trữ vật lý phức tạp đáp ứng cho việc truy xuất hiệu quả, toàn vẹn, phục hồi, điều khiển đồng thời và an toàn.

Dưới đây là bảng so sánh các tính chất đặc trưng của cơ sở dữ liệu tập trung và cơ sở dữ liệu phân tán:

Tính chất đặc trưng	**Cơ sở dữ liệu tập trung**	**Cơ sở dữ liệu phân tán**
Điều khiển tập trung	- Khả năng cung cấp sự điều khiển tập trung trên các tài nguyên thông tin. - Cần có người quản trị cơ sở dữ liệu.	- Cấu trúc điều khiển phân cấp: quản trị cơ sở dữ liệu toàn cục và quản trị cơ sở dữ liệu cục bộ phân tán.
Độc lập dữ liệu	- Tổ chức dữ liệu trong suốt với các lập trình viên. Các chương trình được viết có cái nhìn "quan niệm" về dữ liệu. - Lợi điểm: các chương trình không bị ảnh hưởng bởi sự thay đổi tổ chức vật lý của dữ liệu	- Ngoài tính chất độc dữ liệu như trong cơ sở dữ liệu tập trung, còn có tính chất trong suốt phân tán nghĩa là các chương trình được viết như cơ sở dữ liệu không hề được phân tán.
Sự dư thừa dữ liệu	Giảm thiểu sự dư thừa dữ liệu do: - Tính nhất quán dữ liệu. - Tiết kiệm dung lượng nhớ.	- Giảm thiểu sự dư thừa dữ liệu đảm bảo tính nhất quán. - Nhưng lại nhân bản dữ liệu đến các địa điểm mà các ứng dụng cần đến, giúp cho việc thực thi các ứng dụng không dừng nếu có một địa điểm bị hỏng. Từ đó vấn đề quản lý

		nhất quán dữ liệu sẽ phức tạp hơn.
Các cấu trúc vật lý phức tạp và truy xuất hiệu quả	Các cấu trúc vật lý phức tạp giúp cho việc truy xuất dữ liệu được hiệu quả.	Các cấu trúc vật lý phức tạp giúp liên lạc dữ liệu trong cơ sở dữ liệu phân tán .
Tính toàn vẹn, phục hồi, đồng thời	Dựa vào giao tác.	Dựa vào giao tác phân tán.

Từ bảng so sánh trên, chúng ta thấy việc chọn lựa cơ sở dữ liệu phân tán sẽ thích hợp hơn đối với các ứng dụng phát triển trong một hệ thống mạng diện rộng do giảm được chi phí truyền thông để truy xuất dữ liệu.

1.4 Tại sao cần có cơ sở dữ liệu phân tán?

1.4.1 Lý do tổ chức và kinh tế

Nhiều tổ chức có cơ cấu tổ chức phi tập trung nên giải pháp cơ sở dữ liệu phân tán thích hợp hơn. Những năm gần đây do sự phát triển mạnh mẽ của công nghệ máy tính cùng với sự phát triển rộng rãi của các tổ chức kinh tế trên thế giới nên việc lưu trữ thông tin trên cơ sở dữ liệu tập trung cần xem xét lại về mặt hiệu quả.

1.4.2 Lý do kết nối các cơ sở dữ liệu hiện có

Cơ sở dữ liệu phân tán là giải pháp tự nhiên khi tổ chức đã có sẵn các cơ sở dữ liệu và cần mở rộng nó cho các ứng dụng phổ quát hơn. Trong trường hợp này cơ sở dữ liệu phân tán được xây dựng theo phương pháp từ dưới lên, dựa trên các cơ sở dữ liệu cục bộ có sẵn. Quá trình này có thể yêu cầu cấu trúc lại cơ sở dữ liệu cục bộ tuy nhiên công việc này lại đơn giản hơn xây dựng một cơ sở dữ liệu tập trung hoàn toàn mới.

1.4.3 Lý do tăng trưởng tổ chức

Nếu một tổ chức phát triển bằng cách thêm vào những đơn vị tổ chức tự quản như chi nhánh, kho bãi thì cách tiếp cận theo cơ sở dữ liệu phân tán hỗ trợ cho việc tăng trưởng cơ sở dữ liệu với mức độ ảnh hưởng nhỏ nhất. Trong khi đó cách tiếp cận theo cơ sở dữ liệu tập trung thì ngay từ đầu phải quan tâm đến sự phát triển của nó

trong tương lai mà việc này thì khó dự đoán và tốn kém, nếu không dự liệu trước thì sẽ gây ra hậu quả nghiêm trọng không chỉ cho những ứng dụng mới mà còn cho cả hệ thống có sẵn.

1.1.4 Lý do tải truyền thông

Với một hệ cơ sở dữ liệu phân tán về mặt địa lý thì các ứng dụng truy cập sẽ giảm chi phí truyền thông so với cơ sở dữ liệu tập trung.

1.4.5 Đánh giá về hiệu suất

Sự tồn tại của các bộ xử lý tự quản nâng hiệu suất lên nhờ mức độ xử lý song song. Cơ sở dữ liệu phân tán có ưu thế là phân tán dữ liệu tại các địa điểm nên các ứng dụng có thể chạy riêng rẽ trên từng địa điểm và sự giao tiếp giữa các bộ xử lý là nhỏ nhất.

1.4.6 Độ tin cậy và tính hiệu quả

Mặc dầu việc phân tán dữ liệu làm tăng việc dư thừa dữ liệu trên toàn hệ thống nhưng lại cho chúng ta độ tin cậy và tính hiệu quả cao hơn trong cơ sở dữ liệu tập trung. Tuy nhiên để đạt được mục tiêu trên không phải dễ dàng mà đòi hỏi các kỹ thuật khá phức tạp. Sự hỏng hóc trong cơ sở dữ liệu phân tán có thể xảy ra thường hơn trong cơ sở dữ liệu tập trung vì số địa điểm tăng lên nhưng không bao giờ ảnh hưởng lên toàn hệ thống bởi thế nên nó có độ tin cậy và tính hiệu quả cao hơn cơ sở dữ liệu tập trung.

1.4.7 So sánh ưu và nhược điểm của việc phân tán dữ liệu

- Ưu điểm

- Chia sẽ dữ liệu và điều khiển phân tán: Người sử dụng tại một vị trí này có thể truy xuất dữ liệu (được phép) ở vị trí khác. Hơn nữa việc quản trị cơ sở dữ liệu có thể tđược phân tán và thực hiện tự quản tại mỗi vị trí.

- Độ tin cậy và tính sẵn sàng: Nếu một vị trí bị hỏng thì các vị trí còn lại trong hệ thống cơ sở dữ liệu phân tán vẫn tiếp tục hoạt động. Nếu dữ liệu được nhân bản ở một số vị trí thì một giao dịch cần truy xuất một mục dữ

liệu có thể tìm thấy ở bất kỳ vị trí nào trong số vị trí đó. Như thế sự cố tại một vị trí không ảnh hưởng đến hệ thống.

- Tăng tốc độ xử lý truy vấn: Nếu một truy vấn cần dữ liệu ở một số vị trí thì có thể chia câu truy vấn đó thành các câu truy vấn con rồi thực thi nó song song tại các vị trí.

- Nhược điểm

- Chi phí phát triển phần mềm: Việc phát triển một hệ thống cơ sở dữ liệu phân tán khá phức tạp vì thế cần chi phí lớn.

- Khó phát hiện lỗi: Việc phát hiện lỗi và đảm bảo tính đúng đắn của các thuật toán song song sẽ rất khó khăn.

- Chi phí xử lý tăng: Sự trao đổi các thông báo và xử lý phối hợp giữa các vị trí sẽ tăng chi phí xử lý hơn trong các hệ thống tập trung.

Chương 2 KIẾN TRÚC HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU PHÂN TÁN

MỤC TIÊU

Chương này trình bày khái quát về:

- Kiến trúc tham khảo của cơ sở dữ liệu phân tán nhằm cung cấp một tổng quan về việc phân tán và cấp phát dữ liệu quan hệ đến các sites cục bộ.

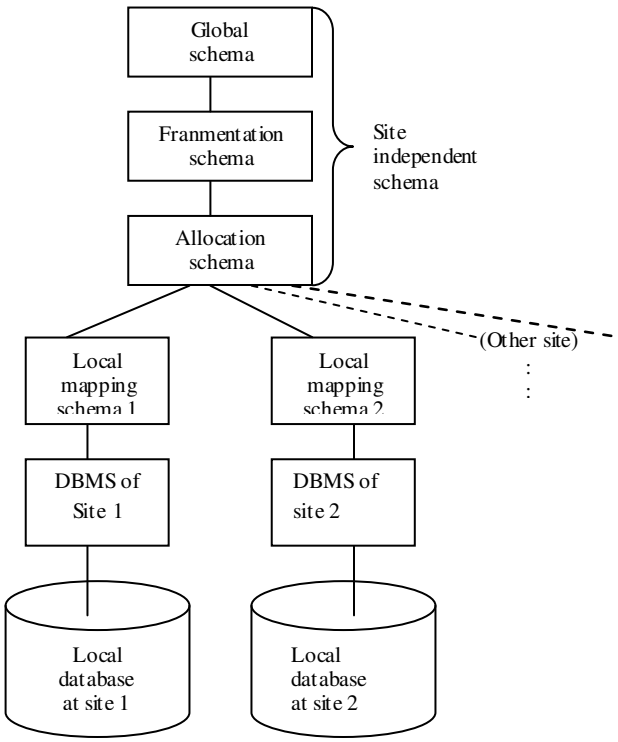
- Các thành phần của hệ quản trị cơ sở dữ liệu phân tán và mối quan hệ tương tác giữa chúng.

- Kiến trúc của hệ quản trị cơ sở dữ liệu phân tán.

2.1 Kiến trúc tham khảo cho cơ sở dữ liệu phân tán

Hình 2.1 mô tả kiến trúc tham khảo cho cơ sở dữ liệu phân tán. Kiến trúc tham khảo này không áp dụng cho mọi cơ sở dữ liệu phân tán. Tuy nhiên các mức của nó giúp cho ta hiểu tổ chức một cơ sở dữ liệu phân tán bất kỳ. Vì thế chúng ta sẽ phân tích và tìm hiểu tất cả các thành phần trong kiến trúc này.

Mỗi quan hệ toàn cục có thể được chia thành các thành phần không trùng nhau được gọi là các phân mảnh. Có nhiều cách để phân mảnh mà chúng ta sẽ bàn đến sau. Ánh xạ từ các quan hệ toàn cục đến các phân mảnh được định nghĩa trong lược đồ phân mảnh. Phép ánh xạ này là một-nhiều nghĩa là có một số phân mảnh tương ứng với một quan hệ toàn cục nhưng chỉ có một quan hệ toàn cục ứng với một phân mảnh. Các phân mảnh được chỉ định bởi tên quan hệ toàn cục với một chỉ mục (chỉ mục phân mảnh) ví dụ R_i chỉ phân mảnh thứ i của quan hệ toàn cục R.

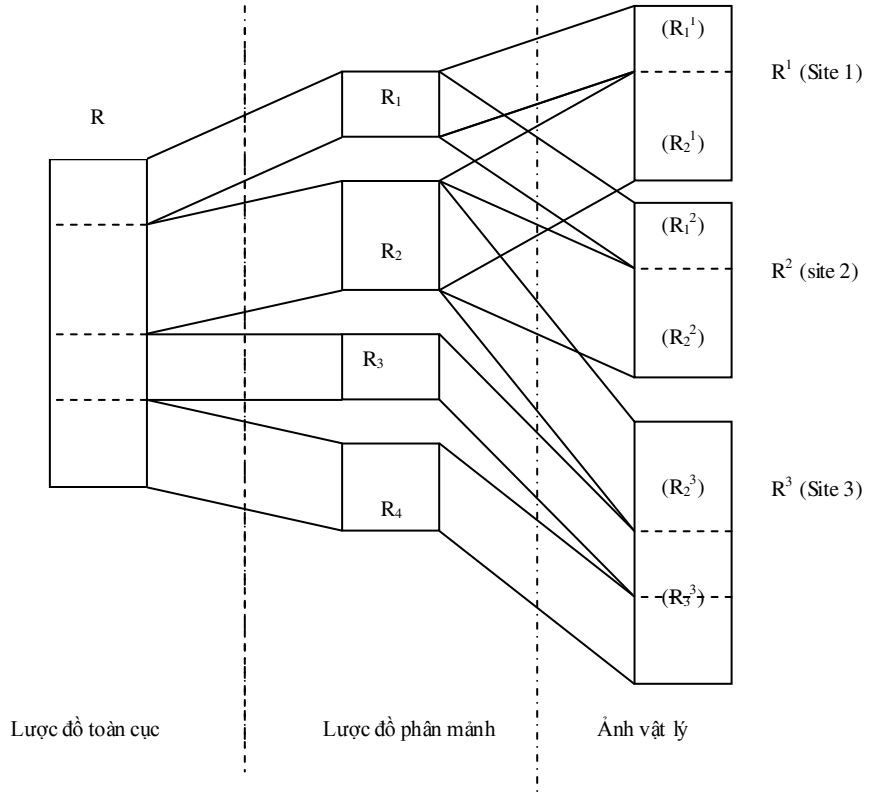


Hình 2.1 Kiến trúc tham khảo cho một cơ sở dữ liệu

Các phân mảnh là các thành phần của các quan hệ toàn cục mà được lưu trữ vật lý tại một hay một số địa điểm. Lược đồ cấp phát (allocation scheme) xác định vị trí của một phân mảnh. Kiểu ánh xạ định nghĩa trong lược đồ cục bộ xác định cơ sở dữ liệu phân tán có dư thừa hay không. Trong trường hợp ánh xạ là một-nhiều thì nó dư thừa, ngược lại nếu ánh xạ có kiểu một-một thì nó không dư thừa. Tất cả các phân mảnh tương ứng với cùng một quan hệ toàn cục R và được lưu trữ tại địa điểm j tạo thành ảnh vật lý của quan hệ R tại địa điểm j. Vì thế có một ánh xạ một-một giữa một ảnh vật lý và một cặp (quan hệ toàn cục, địa điểm); các ảnh vật lý có thể được chỉ ra bởi tên quan hệ toàn cục và chỉ số địa điểm. Để phân biệt các mảnh, chúng ta sẽ sử dụng một chỉ số mũ; ví dụ, R^j chỉ ảnh vật lý của quan hệ toàn cục R tại địa điểm j.

Một ví dụ của mối quan hệ giữa các kiểu đối tượng định nghĩa ở trên được minh họa ở hình 2.2. Một lược đồ quan hệ R được phân thành bốn mảnh R_1, R_2, R_3 và R_4. Bốn phân mảnh này được lưu trữ dư thừa tại ba địa điểm trên mạng máy tính, vì thế tạo ra ba ảnh vật lý R^1, R^2, R^3.

Để làm rõ kỹ thuật này, hãy xét một bản sao của một phân mảnh tại một địa điểm và chú thích nó bằng cách sử dụng tên của quan hệ toàn cục và hai chỉ số (chỉ số phân mảnh và chỉ số địa điểm). Ví dụ, trong hình 2.2, chú thích $R^3{}_2$ chỉ một bản sao của phân mảnh R_2 lưu trữ tại địa điểm 3.



Hình 2.2 Các phân mảnh và các ảnh vật lý đối với một quan hệ toàn cục

Cuối cùng sẽ thấy hai ảnh vật lý bất kỳ có thể được phân biệt. Trong trường hợp này ta sẽ nói một ảnh vật lý là một bản sao của một ảnh vật lý khác. Ví dụ trong hình 2.2, R^1 là một bản sao của R^2.

Kiến trúc tham khảo ở hình 2.1 đã mô tả mối quan hệ giữa các đối tượng tại ba mức trên cùng của kiến trúc này. Ba mức này độc lập vị trí, vì thế chúng không phụ thuộc vào mô hình dữ liệu của các hệ quản trị cơ sở dữ liệu cục bộ. Ở mức thấp hơn, cần ánh xạ các ảnh vật lý đến các đối tượng được thao tác bởi các hệ quản trị cơ sở dữ liệu cục bộ. Ánh xạ này được gọi là lược đồ ánh xạ cục bộ và nó phụ thuộc vào kiểu của hệ quản trị cơ sở dữ liệu cục bộ; vì thế trong hệ thống không đồng nhất, có nhiều kiểu ánh xạ cục bộ tại các vị trí khác nhau.

Kiến trúc này cung cấp một mô hình quan niệm tổng quát để hiểu được cơ sở dữ liệu phân tán. Ba đối tượng quan trọng nhất của kiến trúc này là sự tách biệt giữa sự phân mảnh dữ liệu và sự cục bộ hóa dữ liệu, điều khiển dư thừa dữ liệu và tính độc lập ở các hệ quản trị cơ sở dữ liệu cục bộ.

- *Sự tách biệt quan niệm phân mảnh dữ liệu và quan niệm định vị dữ liệu*: Sự tách biệt này giúp ta phân biệt hai mức khác nhau của sự trong suốt phân tán được gọi là sự trong suốt phân tán và sự trong suốt định vị. Sự trong suốt phân tán là mức độ cao nhất của sự trong suốt và bao gồm các yếu tố mà người sử dụng và các lập trình viên làm việc trên các quan hệ toàn cục. Sự trong suốt định vị là mức thấp hơn và yêu cầu người sử dụng và các lập trình viên làm việc trên các phân mảnh thay vì trên các quan hệ toàn cục. Tuy nhiên họ không cần biết các phân mảnh này lưu trữ ở đâu. Sự tách biệt hai quan niệm phân mảnh và định vị rất phù hợp trong thiết kế cơ sở dữ liệu phân tán vì sự xác định các thành phần thích hợp của dữ liệu được nhận biết từ bài toán định vị tối ưu.

- *Điều khiển tường minh sự dư thừa dữ liệu*: Kiến trúc tham khảo cung cấp một sự điều khiển tường minh cho sự dư thừa dữ liệu tại mức phân mảnh. Ví dụ trong hình 2.2 hai ảnh vật lý R^2_2 và R^3_2 trùng lắp nghĩa là chúng chứa chung dữ liệu. Định nghĩa các phân mảnh một cách tách biệt khi xây dựng các khối vật lý cho phép tham khảo tường minh đến từng phần trùng lắp này tức phân mảnh nhân bản R_2. Điều khiển sự dư thừa dữ liệu rất hữu dụng trong một số khía cạnh quản trị cơ sở dữ liệu phân tán.

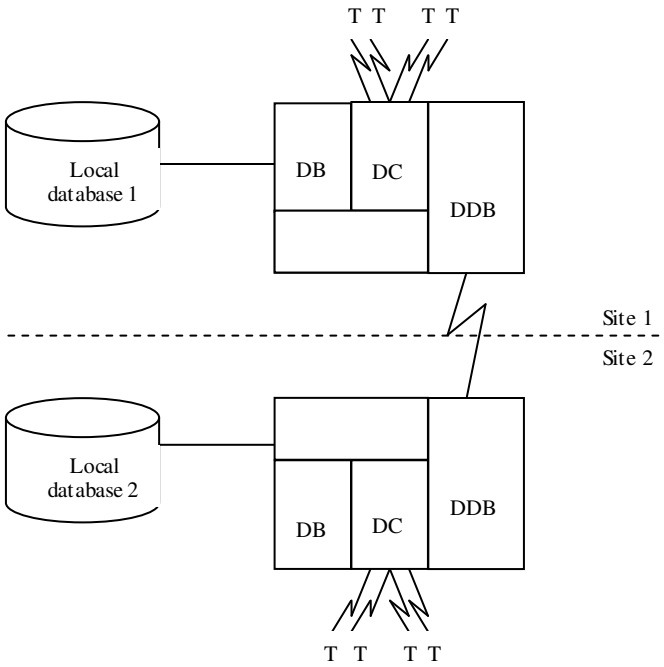
- *Tính độc lập tại các hệ quản trị cơ sở dữ liệu cục bộ*: Tính chất này gọi là sự trong suốt ánh xạ cục bộ, nó cho phép nghiên cứu một số vấn đề quản trị cơ sở dữ liệu mà không quan tâm đến mô hình dữ liệu cụ thể tại các hệ quản trị cơ sở dữ liệu cục bộ.

Một kiểu trong suốt khác liên quan chặt chẽ tới sự trong suốt định vị là sự trong suốt nhân bản. Sự trong suốt nhân bản có nghĩa là người sử dụng không nhận thấy được sự nhân bản của các phân mảnh.

2.2 Các thành phần của hệ quản trị cơ sở dữ liệu phân tán

Hệ quản trị cơ sở dữ liệu phân tán hỗ trợ việc tạo và duy trì cơ sở dữ liệu phân tán. Các hệ quản trị cơ sở dữ liệu phân tán hiện nay được phát triển bởi các nhà sản xuất các hệ quản trị cơ sở dữ liệu tập trung. Chúng chứa các thành phần bổ sung mở rộng các khả năng của các hệ quản trị cơ sở dữ liệu tập trung như hỗ trợ sự truyền thông và sự cộng tác giữa các hệ quản trị cơ sở dữ liệu trên các địa điểm khác nhau qua mạng máy tính. Các thành phần cơ bản cần thiết cho việc xây dựng một cơ sở dữ liệu phân tán là :

1. Thành phần quản trị cơ sở dữ liệu (DB Database Management)
2. Thành phần truyền dữ liệu (DC Data Communication)
3. Tự điển dữ liệu (DD (Data Dictionnary) mở rộng để biểu diễn thông tin về sự phân tán dữ liệu trên mạng.
4. Thành phần cơ sở dữ liệu phân tán (DDB Distributed Database)



Hình 2.3 Các thành phần của hệ quản trị cơ sở dữ liệu

Các thành phần này được minh họa ở hình 2.3 đối với hai địa điểm trên mạng.

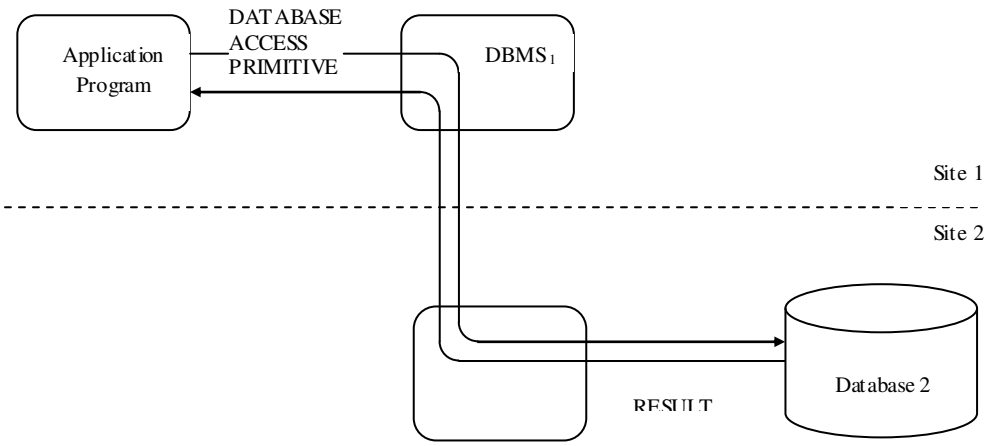
Các dịch vụ được hỗ trợ cho hệ thống trên thông thường là:

- Dịch vụ truy xuất cơ sở dữ liệu từ xa: tính chất này là một tính chất quan trọng nhất và được cung cấp bởi tất cả các hệ thống có thành phần cơ sở dữ liệu phân tán.

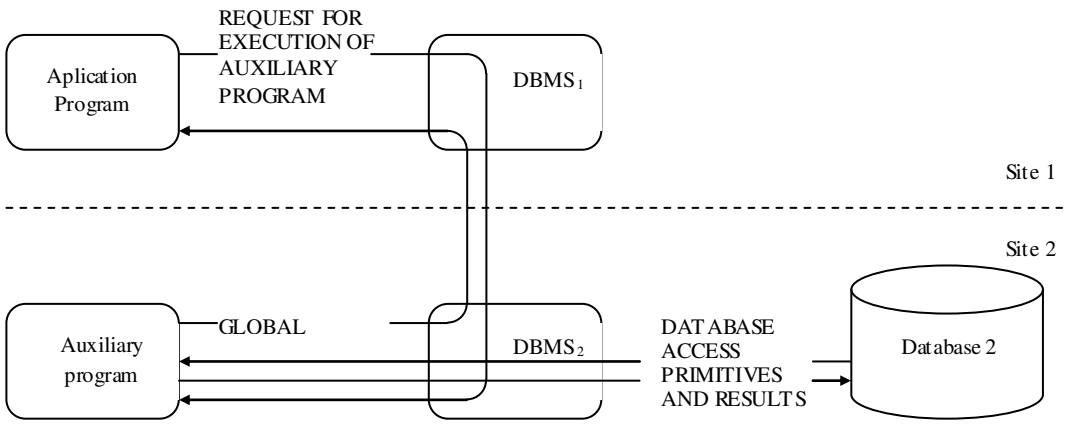
- Mức độ trong suốt của sự phân tán: tính chất này được hỗ trợ bởi các hệ thống khác nhau vì đó là sự cân bằng các yếu tố để đạt được sự kết hợp tốt nhất giữa sự trong suốt phân tán và hiệu suất.

- Hỗ trợ việc quản trị và điều khiển cơ sở dữ liệu: tính chất này bao gồm các công cụ để giám sát cơ sở dữ liệu, lấy thông tin về việc sử dụng cơ sở dữ liệu, cung cấp một cái nhìn toàn cục về các file dữ liệu lưu trữ trên các vị trí khác nhau.

- Hỗ trợ cho việc điều khiển đồng thời và phục hồi các giao tác phân tán.



(a) Truy xuất từ xa qua các nguyên thủy của hệ quản trị cơ sở dữ liệu



(b) truy xuất từ xa qua chương trình bổ trợ

Hình 2.4 Các kiểu truy xuất đến cơ sở dữ liệu phân tán

Việc truy xuất đến một cơ sở dữ liệu từ xa bởi một ứng dụng có thể được thực hiện bởi một hai cách cơ bản minh họa ở hình 2.4. Hình 2.4a minh họa một ứng dụng đưa ra một yêu cầu tham khảo dữ liệu từ xa. Yêu cầu này được định tuyến bởi hệ quản trị cơ sở dữ liệu phân tán đến vị trí mà dữ liệu đó được lưu trữ, sau đó yêu cầu được thực thi tại vị trí đó và trả kết quả về. Trong cách này, đơn vị cơ bản liên lạc giữa các hệ thống là các nghi thức truy xuất cơ sở dữ liệu và kết quả nhận về cũng từ nghi thức này. Nếu các tiếp cận này được sử dụng cho việc truy xuất từ xa, sự trong suốt phân tán có thể được thực hiện bằng cách cung cấp các tên file toàn cục; các nghi thức sẽ tự động định vị các vị trí từ xa thích hợp.

Hình 2.4b minh họa một tiếp cận khác, ứng dụng yêu cầu sự thực thi của một chương trình bổ trợ (auxiliary program) tại vị trí từ xa. Chương trình bổ trợ này truy xuất cơ sở dữ liệu từ xa và trả kết quả cho ứng dụng yêu cầu.

Lợi ích của cách tiếp cận thứ nhất là cung cấp sự trong suốt phân tán nhiều hơn trong khi cách tiếp cận thứ hai có thể linh động hơn nếu nhiều truy xuất cơ sở dữ liệu được yêu cầu vì ứng dụng bổ trợ có thể thực hiện tất cả các truy xuất yêu cầu và chỉ gởi kết quả về.

Một thuộc tính quan trọng của hệ quản trị cơ sở dữ liệu phân tán trong hệ thống là chúng cùng loại hay khác loại. Các hệ quản trị cơ sở dữ liệu phân tán khác loại phải thêm vấn đề thông dịch giữa các mô hình dữ liệu khác nhau, các cấu trúc dữ liêu khác nhau. Đây là một vấn đề rất khó giải quyết, nên nó được khắc phục bằng cách hỗ trợ sự truyền thông giữa các thành phần truyền thông dữ liêu (data communication component DC) khác nhau. Bài toán này cũng đã được công ty Microsoft giải quyết bằng các thành phần truy xuất dữ liệu (Microsoft Data Access Components (MDAC)). Cho nên một hệ thống bao gồm các hệ quản trị cơ sở dữ liệu cục bộ khác nhau sẽ thích hợp hơn cho việc phát triển hệ thống thông tin một cách linh động và tự trị.

2.3 Kiến trúc của hệ quản trị cơ sở dữ liệu phân tán

Phần này sẽ xem xét chi tiết các kiến trúc hệ thống là hệ khách/chủ (client/server), các hệ cơ sở dữ liệu phân tán và các phức hệ cơ sở dữ liệu.

2.3.1 Các hệ khách/chủ (Client/Server)

Trong hệ khách/chủ, ta phân biệt chức năng cần được cung cấp và chia những chức năng này thành hai lớp: chức năng chủ, chức năng khách. Nó cung cấp một kiến trúc hai mức, tạo dễ dàng cho việc quản lý mức độ phức tạp của các hệ quản trị cơ sở dữ liệu hiện đại và độ phức tạp của việc phân tán dữ liệu.

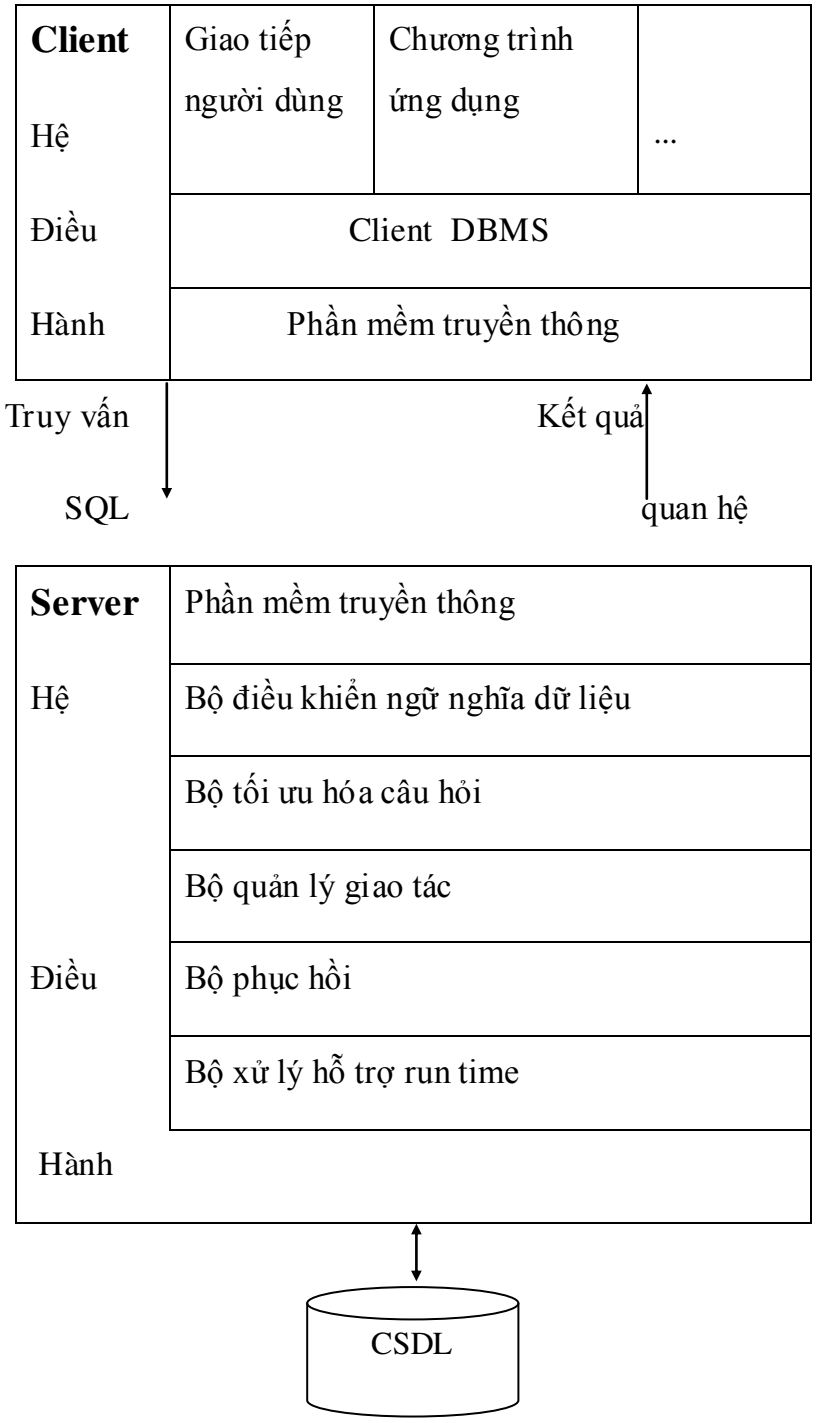
Vì thế, có thể nghiên cứu những khác biệt về chức năng khách và chức năng chủ. Điều đầu tiên phải chú ý là chủ thực hiện phần lớn các công việc quản lý dữ liệu. Điều này có nghĩa là mọi việc xử lý và tối ưu hóa vấn tin, quản lý giao tác và quản lý thiết bị lưu trữ đều được thực hiện tại máy chủ. Khách, ngoài giao diện và ứng dụng, sẽ có một module quản trị cơ sở dữ liệu, khách chịu trách nhiệm quản lý dữ liệu được gởi đến và đôi khi cả việc quản lý các khoá chốt giao tác.

Kiến trúc khách/chủ được biểu diễn trong hình 2.5. Kiến trúc này rất thông dụng trong các hệ thống quan hệ, ở đó việc giao tiếp giữa khách và chủ thông qua các câu lệnh SQL. Nói cách khác, khách sẽ chuyển các câu lệnh SQL cho máy chủ mà không tìm hiểu và tối ưu hoá chúng. Máy chủ thực hiện hầu hết các công việc và trả quan hệ kết quả về cho khách.

2.3.2 Hệ quản trị cơ sở dữ liệu phân tán

Chúnh ta bắt đầu mô tả kiến trúc này bằng cách xem xét hình ảnh tổ chức dữ liệu. Trước tiên ta chú ý rằng tổ chức dữ liệu vật lý trên mỗi máy có thể khác nhau. Vì thế cần có một định nghĩa nội tại riêng tại mỗi vị trí được gọi là lược đồ nội tại cục bộ LIS (local internal schema). Hình ảnh của mô hình dữ liệu toàn cục được mô tả bằng lược đồ quan niệm toàn cục GCS (global conceptual schema). Để xử lý hiện tượng nhân bản và phân mảnh, cần phải mô tả việc tổ chức logic của dữ liệu tại mỗi vị trí, vì thế cần có một tầng thứ ba được gọi là lược đồ quan niệm cục bộ LCS

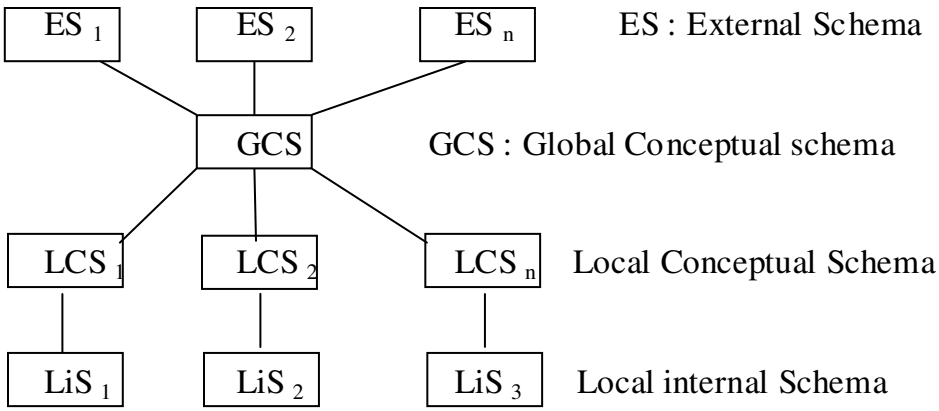
(local conceptual schema). Do vậy trong mô hình kiến trúc này, lược đồ quan niệm toàn cục là hợp của các quan niệm cục bộ. Cuối cùng các ứng dụng và truy xuất cơ sở dữ liệu được hỗ trợ qua lược đồ ngoài ES (external schema). Mô hình kiến trúc này được trình bày ở hình 2.6.



Hình 2.5 Kiến trúc máy khách/chủ

2.3.3 Kiến trúc của hệ quản trị cơ sở dữ liệu phân tán

Chúnh ta bắt đầu mô tả kiến trúc này bằng cách xem xét hình ảnh tổ chức dữ liệu. Trước tiên ta chú ý rằng tổ chức dữ liệu vật lý trên mỗi máy có thể khác nhau. Vì thế cần có một định nghĩa nội tại riêng tại mỗi vị trí được gọi là lược đồ nội tại cục bộ LIS (local internal schema). Hình ảnh của mô hình dữ liệu toàn cục được mô tả bằng lược đồ quan niệm toàn cục GCS (global conceptual schema). Để xử lý hiện tượng nhân bản và phân mảnh, cần phải mô tả việc tổ chức logic của dữ liệu tại mỗi vị trí, vì thế cần có một tầng thứ ba được gọi là lược đồ quan niệm cục bộ LCS (local conceptual schema). Do vậy trong mô hình kiến trúc này, lược đồ quan niệm toàn cục là hợp của các quan niệm cục bộ. Cuối cùng các ứng dụng và truy xuất cơ sở dữ liệu được hỗ trợ qua lược đồ ngoài ES (external schema). Mô hình kiến trúc này được trình bày ở hình 2.6.



Hình 2.6 Kiến trúc tham khảo cơ sở dữ liệu phân tán

Các thành phần cụ thể của một hệ quản trị cơ sở dữ liệu phân tán gồm hai thành phần chính (minh họa ở hình 2.7): bộ phận giao tiếp người sử dụng (user processor) và bộ phận xử lý dữ liệu (data processor).

Các thành phần của bộ phận giao tiếp người sử dụng gồm:

- Bộ phận giao tiếp (user interface handler): chịu trách nhiệm dịch các câu lệnh người sử dụng và định dạng dữ liệu kết quả để chuyển cho người sử dụng.

- Bộ phận kiểm soát dữ liệu ngữ nghĩa (semantic data controller): sử dụng các ràng buộc toàn vẹn và thông tin quyền hạn, được định nghĩa như thành phần của lược đồ quan niệm toàn cục để kiểm tra xem các câu truy vấn có thể xử lý được hay không.

- Bộ phận phân rã và tối ưu hoá vấn tin toàn cục (global query optimizer and decomposer): xác định như một chiến lược hoạt động nhằm giảm thiểu chi phí, phiên dịch các câu vấn tin toàn cục thành các câu vấn tin cục bộ bằng cách sử dụng các lược đồ quan niệm toàn cục, lược đồ quan niệm cục bộ và thư mục toàn cục. Bộ phận tối ưu vấn tin toàn cục còn chịu trách nhiệm tạo ra một chiến lược thực thi tốt nhất cho phép nối phân tán.

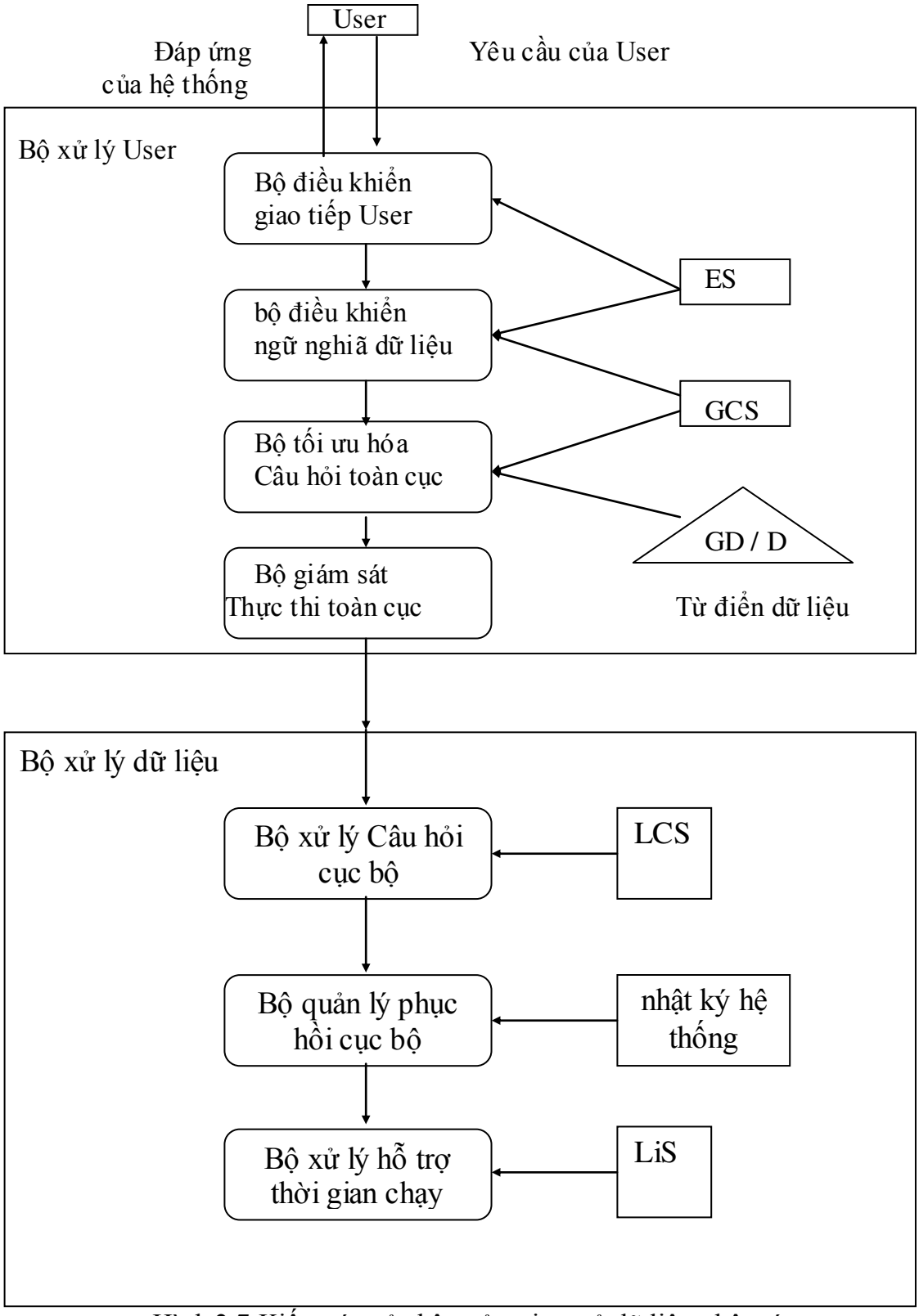
- Bộ phận giám sát hoạt động phân tán (distributed execution monitor): điều phối việc thực hiện phân tán các yêu cầu người sử dụng và cũng được gọi là bộ quản lý giao tác phân tán (distributed transaction manager).

Thành phần chủ yếu thứ hai của hệ quản trị cơ sở dữ liệu phân tán là bộ xử lý dữ liệu (data processor), bao gồm các thành phần:

- Bộ phận tối ưu hoá vấn tin cục bộ (local query optimizer): thường hoạt động như bộ chọn đường truy xuất, chịu trách nhiệm chọn ra một đường truy xuất thích hợp nhất để truy xuất các mục dữ liệu.

- Bộ phận khôi phục cục bộ (local recovery manager) bảo đảm cho các cơ sở dữ liệu cục bộ vẫn duy trì được tính nhất quán ngay cả khi có sự cố xảy ra.

- Bộ phận hỗ trợ lúc thực thi (run-time support processor): truy xuất cơ sở dữ liệu tùy thuộc vào các lệnh trong lịch biểu do bộ phận tối ưu vấn tin tạo ra. Nó chính là bộ giao tiếp với hệ điều hành và chứa bộ quản lý vùng đệm cơ sở dữ liệu, chịu trách nhiệm quản lý vùng đệm và quản lý việc truy xuất dữ liệu.



Hình 2.7 Kiến trúc của hệ quản trị cơ sở dữ liệu phân tán

CHƯƠNG 3 THIẾT KẾ CƠ SỞ DỮ LIỆU PHÂN TÁN

MỤC TIÊU

Chương này đề cập đến hai vấn đề chính sau:

1. Một số tiêu chuẩn thiết kế về cách thức phân tán dữ liệu một cách hợp lý.

2. Nền tảng toán học hỗ trợ cho nhà thiết kế xác định sự phân tán dữ liệu.

Chương này chia làm ba phần:

Phần thứ nhất giới thiệu mô hình thiết kế cơ sở dữ liệu phân tán với hai tiếp cận từ trên xuống và từ dưới lên.

Phần thứ hai trình bày sự thiết kế phân mảnh ngang, phân mảnh dọc và phân mảnh hỗn hợp.

Phần thứ ba trình bày sự cấp phát các phân mảnh. Vấn đề này nhắm đến sự ánh xạ các phân mảnh đến các ảnh vật lý.

Trong cơ sở dữ liệu phân tán, chúng ta đã biết các quan hệ trong lược đồ cơ sở dữ liệu thường được phân ra thành các mảnh nhỏ hơn nhưng chưa đưa ra lý do hoặc chi tiết nào về quá trình này. Phần này đề cập đến các chi tiết đó.

Các câu hỏi sau đây sẽ bao quát toàn bộ vấn đề:

- Tại sao 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 phân mảnh hay không?
- Chúng ta sẽ cấp phát như thế nào?
- Những thông tin nào cần thiết cho việc cấp phát?

3.1 Các vấn đề về thiết kế cơ sở dữ liệu phân tán

3.1.1 Các lý do phân mảnh

Đối với phân mảnh, điều quan trọng là có được một đơn vị phân mảnh thích hợp. Một quan hệ không phải là một đơn vị đáp ứng được yêu cầu đó.

Trước tiên, khung nhìn của các ứng dụng thường chỉ là tập con của quan hệ. Vì thế đơn vị truy xuất không phải là toàn bộ quan hệ mà chỉ là các tập con của quan hệ. Kết quả là xem tập con của các quan hệ là đơn vị phân tán sẽ là điều thích hợp nhất.

Thứ hai là nếu các ứng dụng có các khung nhìn được định nghĩa trên một quan hệ cho trước lại nằm tại những vị trí khác nhau thì có hai cách chọn lựa với đơn vị phân tán là toàn bộ quan hệ. Hoặc quan hệ không được nhân bản mà được lưu ở một vị trí hoặc quan hệ được nhân bản cho tất cả hoặc một số vị trí có chạy ứng dụng. Chọn lựa đầu gây ra một số lượng lớn các truy xuất không cần thiết đến dữ liệu ở xa. Còn chọn lựa sau thực hiện nhân bản không cần thiết, gây ra nhiều vấn đề khi cập nhật và có thể gây ra lãng phí không gian lưu trữ. Vì thế, 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 tác đồng thời. Ngoài ra, việc phân mảnh các quan hệ sẽ cho phép thực hiện song song một câu vấn tin bằng cách chia nó thành một tập các câu vấn tin con hoạt tác trên từng mảnh. Do đó việc phân mảnh sẽ làm tăng mức độ hoạt động song hành và như thế làm tăng lưu lượng hoạt động của hệ thống.

Tuy nhiên cũng cần chỉ rõ những khiếm khuyết của việc phân mảnh:

- Nếu ứng dụng cần phải truy xuất dữ liệu từ hai mảnh rồi nối hoặc hợp chúng lại thì chi phí rất cao.

- Vấn đề thứ hai liên quan đến tính toàn vẹn dữ liệu: do kết quả của việc phân mảnh, các thuộc tính tham gia vào một phụ thuộc hàm có thể bị phân rã vào các mảnh khác

nhau và được cấp phát cho những vị trí khác nhau. Trong trường hợp này việc kiểm tra các phụ thuộc hàm cũng phải thực hiện truy tìm dữ liệu ở nhiều vị trí.

3.1.2 Các kiểu phân mảnh

Có hai kiểu phân mảnh khác nhau là phân mảnh theo chiều dọc, phân mảnh theo chiều ngang và phân mảnh hỗn hợp sẽ được trình bày chi tiết ở phần sau.

3.1.2.1 Phân mảnh ngang

Phân mảnh ngang chia một quan hệ theo các bộ. Vì vậy mỗi mảnh là một tập con của quan hệ. Có hai loại phân mảnh ngang: phân mảnh ngang nguyên thủy và phân mảnh ngang dẫn xuất.

Phân mảnh ngang nguyên thủy (primary horizontal fragmentation) là sự phân mảnh một quan hệ dựa trên một vị từ được định nghĩa trên một quan hệ.

Phân mảnh ngang dẫn xuất (derived horizontal fragmentation) là phân rã một quan hệ dựa vào các vị từ được định nghĩa trên một quan hệ khác.

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 quan hệ R, các mảnh ngang R_i là :

$$R_i = \sigma_{Fi}(R)$$

Trong đó F_i là công thức chọn để có được mảnh R_i. F_i có dạng chuẩn hội.

Ví dụ: xét lược đồ quan hệ toàn cục :

SUPPLIER(SNUM,NAME,CITY)

Chúng ta có thể có hai phân mảnh ngang sau:

$SUPPLIER_1 = \sigma_{CITY=\text{“SF”}}(SUPPLIER)$

$SUPPLIER_1 = \sigma_{CITY=\text{“LA”}}(SUPPLIER)$

- Sự phân mảnh trên thỏa điều kiện đầy đủ nếu “SF” và “LA” chỉ là các giá trị có thể có của thuộc tính CITY; ngược lại chúng ta sẽ không biết những mảnh nào với các giá trị CITY khác.

- Điều kiện tái thiết được kiểm tra dễ dàng vì chúng ta luôn luôn có thể tái thiết lại quan hệ toàn cục SUPPLIER bằng phép toán hội:

$SUPPLIER = SUPPLIER_1 \cup SUPPLIER_2$

- Điều kiện tách biệt cũng được kiểm tra một cách rõ ràng.

3.1.2.2 Phân mảnh ngang dẫn xuất

Trong một số trường hợp sự phân mảnh ngang được dẫn ra từ một phân mảnh ngang của một quan hệ khác.

Ví dụ: Một quan hệ toàn cục

SUPPLY(SNUM, PNUM, DEPTNUM, QUAN)

Với SNUM là mã số người cung cấp. Chúng ta muốn phân chia quan hệ này sao cho một mảnh chứa các bộ cho những người cung cấp ở một thành phố cho trước. Tuy nhiên thành phố không phải là thuộc tính của quan hệ SUPPLY mà là thuộc tính của quan hệ SUPPLIER. Vì thế chúng ta cần thực hiện phép nữa kết để xác định các bộ của SUPPLY tướng ứng với những người cung cấp trong một thành phố cho trước. Sự phân mảnh dẫn xuất của SUPPLY có thể được định nghĩa như sau:

$SUPPLY_1 = SUPPLY \ltimes SUPPLIER_1$

$SUPPLY_2 = SUPPLY \ltimes SUPPLIER_2$

Với ►< là phép toán nửa kết (Semi Join)

- Tính tái thiết quan hệ toàn cục SUPPLY có thể được thể hiện qua phép hội.

- Tính đầy đủ của sự phân mảnh ở trên yêu cầu không được có mã số người cung cấp có trong quan hệ SUPPLY nhưng lại không tồn tại trong quan hệ SUPPLIER. Đây chính là một ràng buộc toàn vẹn về khoá ngoại trong cơ sở dữ liệu.

- Điều kiện tách biệt cũng được thoả nếu một bộ trong quan hệ SUPPLY không tương ứng với hai bộ của quan hệ SUPPLIER mà thuộc về hai mảnh khác nhau. Trường hợp này cũng dễ kiểm tra vì mã số người cung cấp là khoá duy nhất của quan hệ SUPPLIER; tuy nhiên, trong trường hợp tổng quát thì khó có thể chứng minh điều kiện này hơn.

3.1.2.3 Phân mảnh dọc

Sự phân mảnh dọc của một quan hệ toàn cục là việc chia các thuộc tính vào hai (nhiều) nhóm; các mảnh nhận được từ phép chiếu quan hệ toàn cục trên mỗi nhóm. Sự phân mảnh này sẽ đúng đắn nếu mỗi thuộc tính được ánh xạ vào ít nhất vào một thuộc tính của các phân mảnh; hơn nữa, nó phải có khả năng tái thiết lại quan hệ nguyên thủy bằng cách kết nối các phân mảnh lại với nhau. Chú ý, mỗi phân mảnh dọc của một quan hệ toàn cục phải chứa khóa của quan hệ toàn cục. Có hai kiểu phân mảnh dọc : phân mảnh dọc dư thừa và phân mảnh dọc không dư thừa.

Phân mảnh dọc dư thừa (redundant fragmentation):

Phân mảnh dọc dư thừa là các phân mảnh dọc chứa một hoặc nhiều thuộc tính chung không khóa.

Ví dụ : Xét quan hệ toàn cục nhân viên (EMP) mô tả mã nhận viên (EMPNUM), tên nhân viên (NAME), lương (SAL), thuế thu nhập (TAX), nhà quản lý (MNRNUM) và phòng ban họ làm việc (DEPTNUM) như sau :

EMP(EMPNUM, NAME, SAL, TAX, MNRNUM, DEPTNUM)

Quan hệ toàn cục này được phân mảnh dọc dư thừa (thuộc tính NAME) như sau :

EMP1 = $\Pi_{\text{EMPNUM, NAME, MGRNUM, DETPNUM}}$ (EMP)

EMP2 = $\Pi_{\text{EMPNUM, NAME ,SAL, TAX}}$ (EMP)

Phân mảnh dọc không dư thừa (non-redundant fragmentation):

Phân mảnh dọc không dư thừa là các phân mảnh dọc không chứa thuộc tính chung không khóa nào cả.

Ví dụ : Quan hệ toàn cục :

EMP(EMPNUM, NAME, SAL, TAX, MNRNUM, DEPTNUM)

Quan hệ toàn cục này được phân mảnh dọc không dư thừa (thuộc tính) như sau :

EMP1 = $\Pi_{\text{EMPNUM, NAME, MGRNUM, DETPNUM}}$ (EMP)

EMP2 = $\Pi_{\text{EMPNUM, SAL, TAX}}$ (EMP)

Phân mảnh này phản ánh lương và thuế của các nhân viên được quan lý riêng. Việc tái thiết lại quan hệ EMP có thể nhận được từ :

$EMP = EMP_1$ JNN EMP_2

(với JNN là phép kết nối tự nhiên hai quan hệ)

Vì EMPNUM là khóa của quan hệ EMP. Nói chung, việc chứa khóa của quan hệ toàn cục vào mỗi mảnh là cách tốt nhất để bảo đảm cho tính tái thiết.

Từ đó chúng ta thấy sự phân mảnh cũng thỏa tính đầy đủ và tính tách biệt.

Ví dụ : Xét quan hệ toàn cục

EMP(EMPNUM, NAME, SAL, TAX, MNRNUM, DEPTNUM)

Một sự phân mảnh dọc của quan hệ này được định nghĩa:

$EMP_1 = \pi_{EMPNUM, NAME, MGRNUM, DETPNUM}$ (EMP)

$EMP_2 = \pi_{EMPNUM, SAL, TAX}$ (EMP)

Phân mảnh này phản ánh lương và thuế của các nhân viên được quan lý riêng. Việc tái thiết lại quan hệ EMP có thể nhận được từ :

$EMP = EMP_1$ JNN EMP_2

(với JNN là phép kết nối tự nhiên hai quan hệ)

Vì EMPNUM là khóa của quan hệ EMP. Nói chung, việc chứa khóa của quan hệ toàn cục vào mỗi mảnh là cách tốt nhất để bảo đảm cho tính tái thiết.

Từ đó chúng ta thấy sự phân mảnh cũng thỏa tính đầy đủ và tính tách biệt.

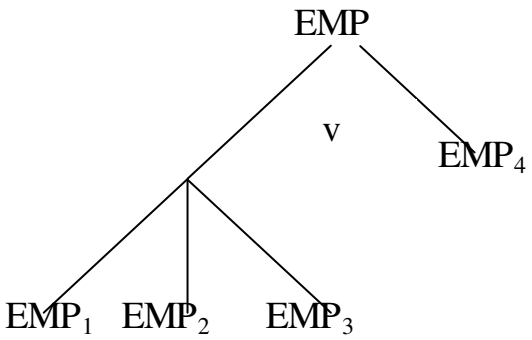
3.1.2.4 Sự phân mảnh hỗn hợp

Các phân mảnh nhận được bởi các phép phân mảnh trên là các quan hệ, vì thế chúng ta có thể áp dụng các phép toán phân mảnh một cách đệ qui. Việc tái thiết quan hệ thực hiện được bằng cách áp dụng các luật tái thiết theo thứ tự ngược. Các biểu thức mà định nghĩa các phân mảnh trong trường hợp này sẽ phức tạp hơn.

Ví dụ: Xét quan hệ toàn cục:

EMP(EMPNUM, NAME, SAL, TAX, MNRNUM, DEPTNUM)

Dưới đây là một sự phân mảnh hỗn hợp bằng cách áp dụng sự phân mảnh dọc rồi sau đó áp dụng phân mảnh ngang trên DEPTNUM:



Hình 3.1 Cây phân mảnh của quan hệ EMP

$EMP_1 = \sigma_{deptnum<=10} (\pi_{empnum, name, mgrnum, deptnum} (EMP))$

$EMP_2 = \sigma_{10<deptnum <=20}(\pi_{empnum, name, mgrnum, deptnum} (EMP))$

$EMP_3 = \sigma_{deptnum>20} (\pi_{empnum, name, mgrnum, deptnum} (EMP))$

$EMP_4 = \pi_{empnum, name, sal, tax} (EMP)$

Việc tái thiết lại quan hệ EMP được định nghĩa bởi biểu thức:

$EMP = U(EMP_1, EMP_2 , EMP_3)$ JNN $\pi_{empnum, sal, tax} (EMP_4)$

Sự phân mảnh hỗn hợp có thể được biểu diễn bởi cây phân mảnh. Trong cây phân mảnh, gốc tương ứng với quan hệ toàn cục, các lá tương ứng với các phân mảnh và các nút trung gian tương ứng với các kết quả trung gian của các biểu thức phân mảnh. Tập các nút con của một nút thể hiện sự phân rã của nút này bởi một phép toán phân mảnh ngang hoặc dọc. Hình 3.1 minh họa cây phân mảnh của quan hệ EMP.

3.2 Thiết kế phân mảnh

Trong chương này chúng ta chỉ tập trung vào những khía cạnh riêng biệt trong cơ sở dữ liệu phân tán mà không đề cập kỹ đến những vấn đề thiết kế cơ sở dữ liệu tập trung. Việc thiết kế cơ sở dữ liệu tập trung nhắm đến hai vấn đề sau:

Thiết kế lược đồ quan niệm: lược đồ cơ sở dữ liệu mô tả các thông tin và mối quan hệ của chúng cần lưu trữ;

Thiết kế "cơ sở dữ liệu vật lý" nghĩa là ánh xạ lược đồ quan niệm đến các vùng lưu trữ vật lý và xác định các phương pháp truy xuất thích hợp.

Trong cơ sở dữ liệu phân tán hai vấn đề này trở thành vấn đề thiết kế lược đồ phổ quát và thiết kế các cơ sở dữ liệu cục bộ tại mỗi site. Sự phân tán cơ sở dữ liệu cộng thêm vào các vấn đề trên hai vấn đề mới:

Thiết kế sự phân tán, nghĩa là xác định các quan hệ phổ quát được phân mảnh ngang, dọc hay hỗn hợp như thế nào?

Thiết kế sự cấp phát các phân mảnh, nghĩa là xác định các phân mảnh được ánh xạ đến các ảnh vật lý như thế nào, kể cả việc xác định sự nhân bản dữ liệu.

Hai vấn đề mới này đặc trưng đầy đủ cho sự thiết kế phân tán dữ liệu. Sự thiết kế phân mảnh là một tiêu chuẩn luận lý trong khi sự thiết kế cấp phát nhắm đến việc sắp đặt dữ liệu vật lý tại các sites.

Mặc dầu việc thiết kế các chương trình ứng dụng được xây dựng sau khi thiết kế lược đồ, sự hiểu biết về các yêu cầu của các chương trình ứng dụng cũng quyết định đến sự thiết kế lược đồ vì các lược đồ phải hỗ trợ các ứng dụng một cách hiệu quả. Các yêu cầu của ứng dụng như sau:

Site mà ứng dụng được đưa ra (còn được gọi là site gốc của ứng dụng)

Tần số hoạt động của ứng dụng (nghĩa là số lượng yêu cầu hoạt động trong một đơn vị thời gian); trong trường hợp tổng quát các ứng dụng có thể được đưa ra từ nhiều sites, chúng ta cần biết tần số hoạt động của mỗi ứng dụng tại mỗi site.

Số lượng, kiểu và sự thống kê phân tán của các truy xuất được tạo bởi các ứng dụng đến mỗi đối tượng dữ liệu được yêu cầu.

3.2.1 Các mục tiêu của việc thiết kế phân tán dữ liệu

Sự truy xuất cục bộ: mục tiêu của sự phân tán dữ liệu là để các ứng dụng truy xuất dữ liệu cục bộ càng nhiều càng tốt, giảm bớt các truy xuất dữ liệu từ xa.

Việc thiết kế sự phân tán dữ liệu để tối đa hoá truy xuất cục bộ có thể được thực hiện bằng cách thêm số lượng các tham khảo cục bộ và các tham khảo từ xa tương ứng cho mỗi phân mảnh dự tuyển và mỗi cấp phát phân mảnh, từ đó chọn ra giải pháp tốt nhất.

Tính sẵn sàng và khả tín của các dữ liệu phân tán: trong chương 1, chúng ta đã chỉ ra tính sẵn sàng và khả tín (độ tin cậy) là các điểm mạnh của cơ sở dữ liệu phân tán so với cơ sở dữ liệu tập trung. Mưc độ sẵn sàng cao đối với các ứng dụng chỉ đọc được thực hiện bằng cách lưu trữ nhiều bản sao của cùng một thông tin; hệ thống phải có khả năng chuyển đến bản sao được chọn thích hợp khi một bản sao không được truy xuất bình thường.

Độ khả tín cũng được thực hiện bằng cách lưu trữ nhiều bản sao, khi đó nó có khả năng phục hồi khi có sự phá huỷ một số bản sao.

Sự phân bố tải: sự phân tán bố trên các sites là một tính chất quan trọng của các hệ thống máy tính phân tán. Sự phân bố tải để tận dụng sức mạnh của việc sử dụng các máy tính, và cực đại hoá mức độ xử lý song song các lệnh thực thi của các ứng dụng. Vì sự phân bố tải có thể ảnh hưởng xấu đến sự truy xuất cục bộ nên cần xem xét để cân bằng hai mục tiêu này.

Chi phí lưu trữ: sự phân tán cơ sở dữ liệu phản ánh chi phí của sự lưu trữ tại các sites khác nhau. Tuy nhiên chi phí lưu trữ dữ liệu không đáng kể so với chi phí xuất nhập, chi phí truyền thông của các ứng dụng. Nhưng giới hạn của bộ lưu trữ phải được xem xét kỹ.

3.2.2 Các tiếp cận để thiết kế sự phân tán dữ liệu

Có hai cách tiếp cận cho sự thiết kế cơ sở dữ liệu: tiếp cận từ trên - xuống và tiếp cận từ dưới - lên.

Trong cách tiếp cận từ trên xuống, chúng ta bắt đầu thiết kế lược đồ phổ quát, thiết kế sự phân mảnh cơ sở dữ liệu và sau cùng cấp phát các mảnh đến các sites, tạo các ảnh vật lý của chúng.

Cách tiếp cận này thích hợp nhất đối với các hệ thống được phát triển từ đầu và nó cho phép thiết kế một cách hợp lý. Trong chương này chúng ta không đề cập đến cách thiết kế lược đồ phổ quát và lược đồ vật lý vì nó không riêng biệt gì đối với cơ sở dữ liệu phân tán mà tập trung vào sự thiết kế phân mảnh và cấp phát các phân mảnh.

Khi cơ sở dữ liệu phân tán được phát triển như là sự tổ hợp các cơ sở dữ liệu sẵn có thì nó lại không dễ dàng đối với phương pháp tiếp cận từ trên -xuống. Trong trường hợp này lược đồ phổ quát thường được tạo ra từ sự thoả hiệp giữa các mô tả dữ liệu sẵn có. Từ đó cách tiếp cận từ dưới-lên có thể được sử dụng để thiết kế sự phân tán dữ liệu. Cách thiết kế từ dưới lên yêu cầu:

Chọn một mô hình cơ sở dữ liệu chung để mô tả lược đồ phổ quát của cơ sở dữ liệu.

Chuyển dịch mỗi lược đồ cục bộ vào trong mô hình dữ liệu chung.

Tổ hợp lại lược đồ cục bộ vào trong lược đồ phổ quát chung.

Ba vấn đề này không riêng biệt gì đối với cơ sở dữ liệu phân tán mà nó hiện diện ngay trong các hệ thống tập trung. Bởi thế phương pháp thiết kế từ dưới lên không được đề cập ở đây. Tuy nhiên ba vấn đề này rất quan trọng trong các hệ thống cơ sở dữ liệu phân tán không đồng nhất.

3.2.3 Thiết kế sự phân mảnh dữ liệu

Thiết kế phân mảnh là vấn đề đầu tiên phải giải quyết trong phương pháp thiết kế phân tán dữ liệu từ trên xuống. Mục đích của việc thiết kế phân mảnh là xác định các phân mảnh không chồng chéo lên nhau. Đó là các đơn vị logic của sự cấp phát.

Thiết kế phân mảnh là nhóm các bộ (trong trường hợp phân mảnh ngang) hoặc nhóm các thuộc tính (theo phân mảnh dọc) mà có những tính chất giống nhau từ quan điểm cấp phát chúng. Mỗi một nhóm các bộ hay các thuộc tính có cũng tính chất sẽ tạo nên một phân mảnh.

Ví dụ 1: Xét sự phân mảnh ngang cho quan hệ phổ quát EMP. Giả sử rằng các ứng dụng quan trọng của cơ sở dữ liệu phân tán này yêu cầu thông tin từ quan hệ EMP về các nhân viên là thành viên của các dự án. Mỗi phòng ban là một site của cơ sở dữ liệu phân tán.

Các ứng dụng có thể được gọi từ bất kỳ phòng ban nào; tuy nhiên khi chúng được gọi từ một phòng ban thì nó sẽ ưu tiên tìm các bộ nhân viên trong phòng ban đó trước với xác suất cao hơn ở những nhân viên của phòng ban khác. Trong trường hợp này các nhân viên được phân mảnh ngang theo tính chất "làm việc cùng một phòng ban".

Một ví dụ đơn giản về sự phân mảnh dọc của quan hệ EMP như sau: giả sử các thuộc tính SAL và TAX chỉ được sử dụng bởi các ứng dụng quản trị thì các thuộc tính này sẽ nằm trong phân mảnh dọc thích hợp.

3.2.3.1 Sự phân mảnh nguyên thủy

Nhắc lại sự phân mảnh nguyên thủy được định nghĩa bằng cách sử dụng phép chọn lựa trên quan hệ toàn cục. Tính đúng đắn của sự phân mảnh nguyên thủy đòi hỏi mỗi bộ trong quan hệ toàn cục chỉ nằm trong một và chỉ một phân mảnh. Vì thế xác định một phân mảnh nguyên thủy của một quan hệ toàn cục yêu cầu xác định một tập các vị từ

chọn đầy đủ và rời nhau. Tính chất mà chúng ta yêu cầu cho mỗi phân mảnh phải được tham khảo đồng nhất bởi tất cả các ứng dụng.

Cho R là quan hệ toàn cục mà chúng ta phân mảnh ngang nguyên thủy. Chúng ta đưa ra một số định nghĩa sau:

1. Một vị từ đơn giản là vị từ có kiểu:

Thuộc tính = giá trị

2. Một vị từ sơ cấp y cho một tập các vị từ đơn giản P là chuẩn hội của tất cả các vị từ xuất hiện trong P :

$$y = \Lambda\ p^*_i$$

Với $p^*_i = p_i$ hoặc p^*_i= not p_i và $y \neq$ false.

3. Một phân mảnh là một tập các bộ tương ứng với một vị từ sơ cấp

4. Một vị từ đơn giản p_i là *thích hợp* đối với một tập các vị từ đơn giản P nếu tồn tại ít nhất hai vị từ sơ cấp mà biểu thức của nó chỉ khác nhau do vị từ p_i (xuất hiện ở dạng thông thường và dạng phủ định của nó) mà *các phân mảnh tương ứng được tham khảo đến bởi ít nhất một ứng dụng.*

Ví dụ 2: Xét sự phân mảnh ngang ở ví dụ 1. Giả sử có một số ứng dụng quan trọng yêu cầu các thông tin về các nhân viên tham gia vào các dự án; lại có một số ứng dụng quan trọng khác không chỉ yêu cầu thông tin trên mà còn cần thông tin về nghề nghiệp. Hai vị từ đơn giản cho ví dụ này là DEPT =1 và JOB =”P”. Các vị từ sơ cấp cho hai vị từ này là:

DEPT = 1 AND JOB = ”P”

DEPT = 1 AND JOB ≠ “p”

DEPT ≠ 1 AND JOB = "p"

DEPT ≠ 1 AND JOB ≠ "p"

Tất cả các vị từ đơn giản trên là thích hợp, trong khi, ví dụ, SAL > 50 không là một vị từ thích hợp; Nói chung, nếu chúng ta phân mảnh với bất kỳ vị từ sơ cấp nào ở trên cùng với vị từ đơn giản SAL>50 hoặc SAL <=50 thì mỗi phần trong hai phân mảnh đó sẽ được tham khảo như nhau trong các ứng dụng.

Các định nghĩa trên không dễ xây dựng. Thật không may, phép chọn lựa của các vị từ không được hổ trợ bởi các luật chính xác mà thường dựa trên trực quan của người thiết kế cơ sở dữ liệu. Tuy nhiên chúng ta cũng có thể định nghĩa hai tính chất đặc trưng cho một sự phân mảnh thích hợp.

Cho P = $\{p_1, p_2, ..., p_n\}$ là tập các vị từ đơn giản. Để P thể hiện sự phân mảnh một cách đúng đắn và hiệu qủa. P phải đầy đủ và cực tiểu.

1. Tập các vị từ đơn giản Pr được gọi là đầy đủ nếu và chỉ nếu xác xuất mỗi ứng dụng truy xuất đến một bộ bất kỳ thuộc về một mảnh hội sơ cấp nào đó được định nghĩa theo Pr đều bằng nhau.

2. Tập các vị từ đơn giản Pr được gọi là cực tiểu nếu tất cả các vị từ của nó thích hợp.

Ví dụ 3: Hai ví dụ 1, 2 có thể được sử dụng để làm rõ các định nghĩa này.

P_1 = { DEPT=1 } không đầy đủ, vì các ứng dụng tham khảo các bộ của các lập trình viên với xác suất lớn hơn những phân mảnh khác dẫn từ P_1.

P_2 ={ DEPT = 1, JOB = "P" } là đầy đủ và cực tiểu.

P_3 = {DEPT = 1, JOB = "P", SAL >5 }là đầy đủ nhưng không cực tiểu vì SAL > không thích hợp.

Sự phân mảnh có thể thực hiện như sau:

Nguyên tắc: Xét một vị từ p_1 phân chia các bộ của R vào hai phần mà chúng được tham khảo khác nhau bởi ít nhất một ứng dụng. Cho $P = p_1$.

Phương pháp: Xét một vị từ đơn giản mới p_i phân chia ít nhất một phân mảnh của P thành hai phần mà được tham khảo khác nhau bởi ít nhất bởi một ứng dụng. Đặt $P = P \cup p_i$. Xoá các vị từ không thích hợp khỏi P. Lặp lại bước này cho đến khi tập của các phân mảnh cơ sở là đầy đủ.

Ví dụ 4: Lấy lại các ví dụ để làm ví dụ minh họa cho phương pháp ở trên. Xét vị từ đầu tiên SAL > 50; giả sử lương trung bình của các lập trình viên lớn hơn 50, vị từ này xác định hai tập nhân viên mà được tham khảo một cách khác nhau bởi các ứng dụng. Ta có $P_1 = \{ SAL > 50\}$

Chúng ta xét DEPT =2; vị từ này là thích hợp và được thêm vào tập P_1, ta được $P_2 = \{SAL > 50, DEPT = 1\}$

Cuối cùng, xét JOB = "P". Vị từ này cũng thích hợp và thêm nó vào P_2, ta được $P_3 = \{$ SAL>50, DEPT = 1, JOB ="P"}. Chúng ta khám phá ra SAL>50 không thích hợp trong P_3. Vì thế chúng ta nhận được tập cuối cùng là $P_4 = \{$ DEPT =1 , JOB = "P"} đầy đủ và cực tiểu.

Xét một ví dụ tổng quát: Cơ sở dữ liệu gồm có các quan hệ EMP, DEPT, SUPPLIER, SUPPLY.

Giả sử cơ sở dữ liệu phân tán của công ty ở California có ba sites tại San Francisco (site 1), Fresno (site 2), và Los Angeles (site 3); Fresno nằm giữa San Francisco và Los Angeles. Có tất cả 30 phòng ban được nhóm lại như sau: 10 phòng ban đầu tiên ở gần San Francisco, các phòng ban từ 11 đến 20 ở gần Fresno và các phòng ban trên 20 thì ở gần Los Angeles. Tất cả các nhà cung cấp ở San Francisco hoặc ở Los Angeles. Ngoài

ra công ty cũng được chia theo khái niệm miền: San Francisco ở miền Bắc, Los Angeles ở miền nam còn Fresno nằm giữa hai miền đó nên một số phòng ban nằm gần Fresno sẽ rời vào miền bắc hoặc miền nam.

Chúng ta thiết kế sự phân mảnh của SUPPLIER và DEPT với sự phân mảnh ngang nguyên thủy.

Các nhà cung cấp trong quan hệ SUPPLIER(SNUM, NAME, CITTY) có giá trị của thuộc tính CITY là "SF" hoặc là "LA". Giả sử có một ứng dụng quan trọng yêu cầu cho biết tên nhà cung cấp (NAME) khi nhập mã số nhà cung cấp (SNUM). Câu lệnh SQL cho ứng dụng đó như sau:

```
Select NAME
from SUPPLIER
where SNUM = $X
```

Ứng dụng được gọi tại bất kỳ site nào; nếu nó được gọi tại site 1, nó sẽ tham khảo đến SUPPLIERS có CITY = "SF" với xác suất 80%; nếu được gọi từ site 2, nó sẽ tham khảo đến SUPPLIERS của "SF" và "LA" với xác suất bằng nhau; nếu nó được gọi từ site 3, nó sẽ tham khảo đến SUPPLIERS của "LA" với xác suất 80%. Điều này dẫn đến là các phòng ban sẽ liên hệ đến các nhà cung cấp ở gần đó.

Chúng ta đưa các vị từ sau:

p_1 : CITY = "SF"

P_2: CITY = "LA"

Tập $\{p_1, p_2\}$ là đầy đủ và cực tiểu.

Mặc dầu đơn giản, ví dụ này minh họa hai tính chất quan trọng sau:

- Các vị từ thích hợp mô tả cho phân mảnh này không thể được suy ra bằng cách phân tích mã lệnh của ứng dụng.

- Quan hệ mật thiết giữa các vị từ giảm đi số lượng phân mảnh. Trong trường hợp này chúng ta nên xem xét những vị từ tương ứng với các vị từ sơ cấp sau:

y_1: (CITY = "SF") AND (CITY = "LA")
y_2: (CITY = "SF") AND NOT(CITY = "LA")
y_3: NOT(CITY = "SF") AND (CITY = "LA")
y_4: NOT(CITY = "SF") AND NOT(CITY = "LA")

Nhưng chúng ta đã biết rằng:

(CITY = "LA") $\Rightarrow$ NOT (CITY = "SF")

và (CITY = "SF") $\Rightarrow$ NOT (CITY = "LA")

và vì thế chúng ta suy ra y_1 và y_4 mâu thuẫn lẫn nhau và y_2 và y_3 sẽ đơn giản thành hai vị từ p_1 và p_2.

Bây giờ chúng ta hãy xét quan hệ phổ quát sau:

DEPT(DEPTNUM, NAME, AREA, MGRNUM)

Chúng ta sẽ tập trung vào các ứng dụng quan trọng sau:

Các ứng dụng quản trị chỉ được gọi từ site 1 và site 3; các ứng dụng quản trị về các phòng ban ở miền bắc được gọi tại site 1 và các ứng dụng quản trị về các phòng ban ở miền nam được gọi tại site 3.

Các ứng dụng về công việc được quản lý tại mỗi phòng ban; chúng có thể được gọi từ bất kỳ phòng ban nào nhưng chúng phải tham khảo các bộ của phòng ban gần site của nó nhất với xác xuất cao hơn các bộ ở những lưu ở những nơi khác.

Chúng ta đưa ra các vị từ sau:

p_1: DEPTNUM <= 10

p_2: 10 < DEPTNUM <= 20

p_3: DEPTNUM > 20

p_4: AREA = "North"

p_5: AREA = "South"

Có một số quan hệ giữa các vị từ trên như AREA= "North" kéo theo DEPTNUM > 20 là sai 5 (Hình 3.2); vì thế sự phân mảnh giảm còn 4 phân mảnh:

y_1: DEPTNUM <=10

y_2: (10 < DEPTNUM <= 20) AND (AREA = "North")

y_3: (10 < DEPTNUM <= 20) AND (AREA = "South")

y_4: DEPTNUM > 20

	p_4: AREA = "North"	p_5: AREA = "South"
p_1: DEPTNUM <= 10	y_1	FALSE
p_2: 10 < DEPTNUM <= 20	y_2	y_3
p_3: DEPTNUM > 20	FALSE	y_4

Hình 3.2 Sự phân mảnh của quan hệ DEPT

Một nhận xét cuối cùng là sự cấp phát phân mảnh cũng dễ dàng thấy qua sự phân mảnh này. Các phân mảnh tương ứng với vị từ y_1 và y_4 được lưu trữ tại site 1 và site 3; các phân mảnh ứng với các vị từ y_2 hoặc y_3 thể hiện các ứng dụng quản trị thì được phân bố tại site 1 hoặc 3 và các phân mảnh ứng về công việc của phòng ban có thể được lưu trữ tại site 2.

Thuật toán COM_MIN được giới thiệu để tìm tập các vị từ đầy đủ và cực tiểu:

Thuật toán COM_MIN

Input : R: quan hệ; Pr: tậpcác vị từ đơn giản;

Output: Pr': tập các vị từ cực tiểu và đầy đủ;

Declare

F: tập các mảnh hội sơ cấp;

Begin

Pr'=ϕ; F = ϕ;

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 ∪ p; {f_i là mảnh hội sơ cấp theo p_i }

End; {Chúng ta đã chuyển các vị từ có phân mảnh R vào Pr'}

Repeat

For each p∈ Pr if p phân hoạch một mảnh f_k của Pr' theo quy tắc 1 then

Begin

Pr': = Pr'∪ p;

Pr: = Pr – p;

F: = F ∪ p;

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;

End. {COM_MIN}

Thuật toán bắt dầu bằng cách tìm một 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 tính cực tiểu của Pr'.

Sau khi có được tập các vị từ cực tiểu và đầy đủ, ta có thể tiến hành việc phân mảnh ngang nguyên thủy dựa vào thuộc toán PHORIZONTAL như sau:

Thuật toán 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 $m_i \in M$ **do**

Begin

IF m_i mâu thuẫn với I **then**

M:= M-m_i

End;

End. {PHORIZONTAL}

ng ta có tập Pr' là cực tiểu và đầy đủ.

3.2.3.2 Sự phân mảnh ngang dẫn xuất

Sự phân mảnh dẫn xuất ngang của một quan hệ toàn cục R không dựa trên các thuộc tính của nó mà được dẫn ra từ sự phân mảnh ngang của một quan hệ khác. Sự phân mảnh dẫn xuất ngang được sử dụng để thuận lợi cho việc kết nối các mảnh.

Một kết nối phân tán là một kết nối giữa các quan hệ phân mảnh ngang. Khi một ứng dụng yêu cầu một kết nối giữa hai quan hệ toàn cục R và S, tất cả các bộ của R và S cần được so sánh; vì thế, cần phải so sánh tất cả các phân mảnh R_i của R với các phân mảnh S_j của S. Tuy nhiên, đôi khi chúng ta có thể giảm một số kết nối cục bộ rỗng giữa các phân mảnh. Điều này xảy ra khi các giá trị của thuộc tính kết nối trong R_i và S_j rời nhau.

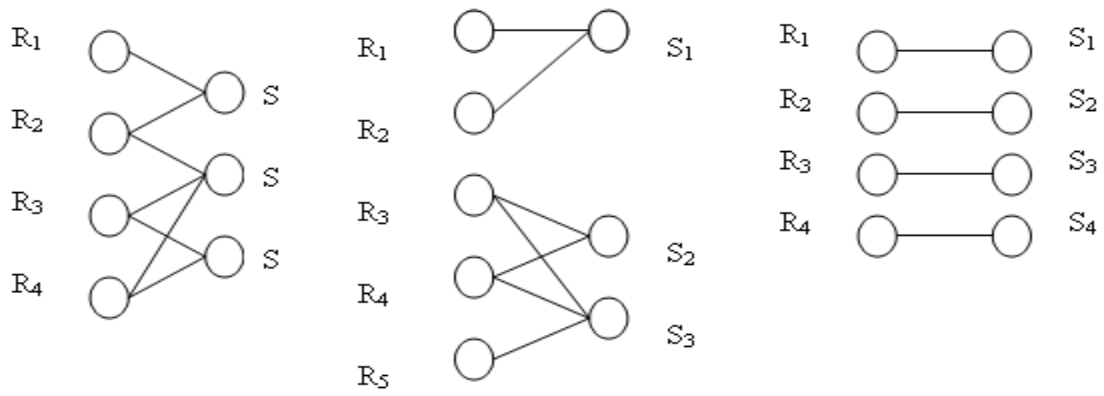
Kết nối phân tán được biểu diễn một cách hiệu quả bằng cách dùng đồ thị kết nối. Đồ thị kết nối G của kết nối phân tán R và S là một đồ thị (N,E) với các nút N thể hiện các phân mảnh của R và S và các cạnh vô hướng E biểu diễn các kết nối không rỗng giữa các phân mảnh. Để đơn giản, chúng ta không chứa trong các phân mảnh nào của R và S mà có kết nối rỗng. Đồ thị kết nối được minh họa ở hình 3.3.

Chúng ta nói một đồ thị kết nối là hoàn toàn khi nó chứa tất cả các cạnh có thể có giữa các phân mảnh của R và S. Nó được rút gọn khi mất một số cạnh. Có hai kiểu đồ thị rút gọn:

1. Đồ thị kết nối được phân hoạch nếu đồ thị gồm hai hay nhiều đồ thị con rời nhau.

2. Đồ thị kết nối đơn giản nếu nó được phân hoạch và mỗi đồ thị con có một cạnh.

Xác định một kết nối của một đồ thị kết nối đơn giản là rất quan trọng trong thiết kế cơ sở dữ liệu. Một cặp phân mảnh mà được kết nối bởi một cạnh trong một đồ thị đơn giản thì có một tập giá trị chung ứng của thuộc tính kết nối. Vì thế, nếu có thể xác định sự phân mảnh và sự định vị của hai quan hệ R và S sao cho đồ thị kết nối là đơn giản và các cặp phân mảnh tương ứng được lưu trữ tại một địa điểm thì kết nối này có thể được biểu diễn phân tán bằng cách kết nối cục bộ các cặp phân mảnh và sau đó suy ra các kết quả của những kết nối qua các địa điểm.



Hình 3.3 Các đồ thị kết nối

Tới đây có thể đưa ra một định nghĩa hình thức của sự phân mảnh dẫn xuất ngang. Cho một quan hệ toàn cục R, các phân mảnh R_i của nó được dẫn xuất từ sự phân mảnh R và S qua phép nửa kết nối ►<:

$$R_i = R \blacktriangleright < S_t$$

Kết nối $R \blacktriangleright < S_t$ là đơn giản nếu các điều kiện tách biệt và đầy đủ của sự phân mảnh được thỏa.

Ví dụ về sự phân mảnh hỗn hợp:

Xét quan hệ SUPPLY (SNUM, PNUM, DETPNUM, QUAN). Giả sử các ứng dụng mà sử dụng quan hệ này luôn luôn liên hệ đến quan hệ khác như quan hệ SUPPLIER, DEPT như sau:

- Một số ứng dụng yêu cầu thông tin về các giao dịch cung cấp từ các nhà cung cấp cho trước; vì thế phải kết nối SUPPLY với SUPPLIER trên thuộc tính SNUM.

SUPPLY(SNUM, PNUM, DEPTNUM, QUAN)

SUPPLY1 = SUPPLY ►< SUPPLIER1

SUPPLY2 = SUPPLY ►< SUPPLIER2

Với ►< là phép toán nửa kết (Semi Join)

Các ứng dụng khác yêu cầu thông tin về các giao dịch cung cấp từ các phòng ban cho trước. vì thế phải kết nối SUPPLY với DEPT trên thuộc tính DEPTNUM.

Giả sữ quan hệ phổ quát DEPT được phân mảnh ngang theo DEPTNUM và quan hệ SUPPLIER phân mảnh ngang theo DEPTNUM. Có hai phân mảnh ngang dẫn xuất cho quan hệ SUPPLY bằng phép nửa kết với các phân mảnh SUPPLIER và với các phân mảnh DEPT.

SUPPLY(SNUM, PNUM, DEPTNUM, QUAN)

SUPPLY1 = SUPPLY ►< DEPT1

SUPPLY2 = SUPPLY ►< DEPT2

Với ►< là phép toán nửa kết (Semi Join)

Kiểm tra tính đúng đắn

Bây giờ chúng ta cần phải kiểm tra tính đúng của phân mảnh ngang.

a. Tính đầy đủ

+ Phân mảnh ngang nguyên thuỷ: Với điều kiện các vị từ chọn là đầy đủ, phân mảnh thu cũng được đảm bảo là đầy đủ, bởi vì cơ sở của thuật toán phân mảnh là tập các vị từ cực tiểu và đầy đủ Pr', nên tính đầy đủ được bảo đảm với điều kiện không có sai sót xảy ra.

+ Phân mảnh ngang dẫn xuất: Có khác chút ít, khó khăn chính ở đây 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.

R là quan hệ thành viên của một đường nối mà chủ nhân là quan hệ S. Gọi A là thuộc tính nối giữa R và S, thế thì với mỗi bộ t của R, phải có một bộ t' của S sao cho

t.A=t'.A

Quy tắc này được gọi là **ràng buộc toàn vẹn** hay **toàn vẹn tham chiếu**, bảo đảm rằng mọi bộ trong các mảnh của quan hệ thành viên đều nằm trong quan hệ chủ nhân.

b. Tính tái thiết

Tái thiết một quan hệ toàn cục từ các mảnh được thực hiện bằng toán tử hợp trong cả phân mảnh ngang nguyên thủy lẫn dẫn xuất, Vì thế một quan hệ R với phân mảnh $F_r=\{R_1, R_2,\ldots,R_m\}$ chúng ta có

$$\mathbf{R = \cup R_i \ , \forall R_i \in F_R}$$

c. Tính tách biệt

Với phân mảnh nguyên thuỷ tính tách rời sẽ được bảo đảm miễn là các vị từ hội sơ cấp xác định phân mảnh có tính loại trừ tương hỗ (mutually exclusive). Với phân mảnh dẫn xuất tính tách rời có thể bảo đảm nếu đồ thị nối thuộc loại đơn giản.

3.2.3.3 Sự phân mảnh dọc

Xác định sự phân mảnh dọc của một quan hệ toàn cục đòi hỏi nhóm lại các thuộc tính mà được tham khảo cùng kiểu bởi các ứng dụng.

Điều kiện đúng đắn của sự phân mảnh dọc yêu cầu mỗi thuộc tính của R phụ thuộc vào ít nhất một tập thuộc tính và mỗi tập thuốc tính phải chứa thuộc tính khoá của R.

Mục đích của sự phân mảnh dọc là để xác định các mảnh R_i nào mà nhiều ứng dụng có thể thực thi trên chúng. Xét một quan hệ toàn cục R được phân hoạch dọc thành R_1 và R_2. Một ứng dụng sẽ có lợi qua việc phân hoạch này nếu nó có thể được thực thi bằng cách chỉ sử dụng R_1 hoặc R_2. Tuy nhiên nếu ứng dụng yêu cầu cả hai phân mảnh thì việc phân hoạch này không có lợi vì phải thực hiện phép kết nối để xây dựng lại R.

Việc xác định một phân mảnh dọc cho một quan hệ toàn cục không dễ dàng khi số lượng tổ hợp các thuộc tính lớn. Vì thế cách tiếp cận heuristic có ưu thế hơn. Chúng ta sẽ mô tả vắn tắt hai cách tiếp cận này:

1. ***Tiếp cận phân rã***: Một quan hệ toàn cục sẽ được lần lượt phân rã vào thành các phân mảnh.

2. ***Tiếp cận gom nhóm***: Các thuộc tính được gom nhóm lần lượt để tạo thành các phân mảnh.

Cả hai tiếp cận này giống nhau ở điểm: Chúng tiếp diễn bằng cách tạo ra một chọn lựa tốt nhất tại mỗi vòng lặp. Trong cả hai trường hợp, các công thức chọn lựa được dùng để xác định khả năng tốt nhất cho việc phân rã hay gom nhóm. Một số dạng quay lui (backtracking) có thể được tạo ra để chuyển một số thuộc tính từ tập này đến tập khác cho đến khi đạt dược phân mảnh cuối cùng.

Ví dụ: Xét quan hệ toàn cục:

EMP(EMPNUM, NAME, SAL, TAX, MGRNUM, DEPTNUM)

Giả sử các ứng dụng sử dụng quan hệ DEPTNUM như sau:

Các ứng dụng quản trị tập trung ở site 3 yêu cầu thông tin NAME, SAL, TAX của các nhân viên.

Các ứng dụng về cộng việc được quản lý tại các phòng ban yêu cầu thông tin về NAME, MGRNUM và DEPTNUM của các nhân viên; các ứng dụng này có thể được gọi tại tất cả các sites và tham khảo các bộ nhân viên trong cùng một nhóm của các phòng ban với xác suất 80%.

Vì thế sự phân mảnh dọc của EMP thành hai mảnh với các thuộc tính "quản trị" và các thuộc tính "mô tả công việc" là khá tự nhiên. Từ đó chún ta có được hai phân mảnh dọc như sau:

EMP_1(EMPNUM,NAME,TAX,SAL)
EMP_2(EMPNUM, NAME, MGRNUM,DEPT)

Trong hai phân mảnh này chúng ta quyết định để thuộc tính NAME ở cả hai phân mảnh nhằm tăng hiệu suất chương trình (khỏi phép thực hiện phép kết) và hơn nữa là tên của các nhân vien thì thường là không thay đổi.

Phương pháp phân mảnh dọc

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 khoá 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 chỉ cần chạy trên một mảnh. Một phân mảnh "tối ưu"là phân mảnh sinh ra một lược đồ phân mảnh cho phép giảm tối đa thời gian thực thi các ứng dụng chạy trên mảnh đó.

Phân mảnh dọc tất nhiên là phức tạp hơn so với phân mảnh ngang. Điều này là do tổng số chọn lựa có thể của một phân hoạch dọc rất lớn.

Vì vậy để có được các lời giải tối ưu cho bài toán phân hoạch dọc thực sự rất khó khăn. 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.

- 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 thỏa một tiêu chuẩn nào đó. Kỹ thuật này được được đề xuất lần đầu cho các CSDL tập trung và về sau được dùng cho các CSDL phân tán.

- 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.

Bởi vì phân hoạch dọc đặt vào một mảnh các thuộc tính thường được truy xuất chung với nhau, chúng ta cần có một giá trị đo nào đó để định nghĩa chính xác hơn về khái niệm "chung với nhau". Số đo này gọi là tụ lực hay lực hút (affinity) của thuộc tính, chỉ ra mức độ liên đới giữa các thuộc tính.

Yêu cầu dữ liệu chính có liên quan đến các ứng dụng là tần số truy xuất của chúng. 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 sẽ đưa ra một giá trị sử dụng thuộc tính, ký hiệu $use(q_i, A_j)$ được định nghĩa như sau:

$$use(q_i, A_j)= \begin{cases} 1: & \text{nếu thuộc tính } A_j \text{ được vấn tin } q_i \text{ tham chiếu} \\ 0: & \text{trong trường hợp ngược lại} \end{cases}$$

Các véctơ $use(q_i, \bullet)$ cho mỗi ứng dụng rất dễ định nghĩa nếu nhà thiết kế biết được các ứng dụng sẽ chạy trên CSDL.

Ví dụ:

Xét quan hệ dự án:

Project(JNO, JNAME, BUDJET, LOC)

Giả sử rằng các ứng dụng sau đây chạy trên các quan hệ đó. Trong mỗi trường hợp chúng ta cũng đặc tả bằng SQL.

q1: Tìm ngân sách của một dự án, cho biết mã của dự án

```
SELECT  Budjet
FROM    PROJECT
WHERE   JNO =giá trị
```

q2: Tìm tên và ngân sách của tất cả mọi dự án

```
SELECT  JNAME, Budjet
FROM    PROJECT
```

q3: Tìm tên của các dự án được thực hiện tại một thành phố đã cho

```
SELECT  JNAME
FROM    PROJECT
WHERE   LOC=giá trị
```

q4: Tìm tổng ngân sách dự án của mỗi thành phố

```
SELECT  SUM (Budjet)
FROM    PROJECT
WHERE   LOC=giá trị
```

Dựa theo bốn ứng dụng này, chúng ta có thể định nghĩa ra các giá trị sử dụng thuộc tính. Để cho tiện về mặt ký pháp, chúng ta gọi A_1=JNO, A_2=JNAME, A_3=Budjet, A_4=LOC. Giá trị sử dụng được định nghĩa dưới dạng ma trận sử dụng (hình 3.4), trong đó mục (i,j) biểu thị use(q_i, A_j).

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1

Hình 3.4 Ma trận sử dụng

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.

Xây dựng công thức để đo lực hút của hai thuộc tính A_i, A_j.

Gọi k là số các mảnh của R được phân mảnh. Tức là $R = R_1 \cup \ldots R_k$.

$Q= \{q_1, q_2,\ldots,q_m\}$ là tập các câu vấn tin (tức là tập các ứng dụng chạy 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:

$Q(A, B) = \{q \in Q: use(q, A) = use(q, B) = 1\}$

ví dụ dựa vào ma trận trên 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 giữa hai thuộc tính A_i, A_j được định nghĩa là:

$$aff(A_i, A_j)= \sum_{qk \in Q(Ai, Aj)} \sum_{l \in Rl} ref_l(q_k)acc_l(q_k)$$

Hoặc:

$$aff(A_i, A_j)= \sum_{Use(qk, Ai)=1 \wedge Use(qk, Aj)=1} \sum_{\forall Rl} ref_l(q_k)acc_l(q_k)$$

Trong đó $ref_l(q_k)$ là số truy xuất đến các thuộc tính (A_i, A_j) cho mỗi ứng dụng q_k tại vị trí R_l và $acc_l(q_k)$ là số đo tần số truy xuất ứng dụng q_k 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).

Ví dụ: Chúng ta hãy tiếp tục với ví dụ 11. Để cho đơn giản chúng ta hãy giả sử rằng $ref_l(q_k) = 1$ cho tất cả q_k và R_l. Nếu tần số ứng dụng là:

$Acc_1(q1) = 15$ $Acc_2(q1) = 20$ $Acc_3(q1) = 10$

$Acc_1(q2) = 5$ $Acc_2(q2) = 0$ $Acc_3(q2) = 0$

$Acc_1(q3) = 25$ $Acc_2(q3) = 25$ $Acc_3(q3) = 25$

$Acc_1(q4) = 3$ $Acc_2(q4) = 0$ $Acc_3(q1) = 0$

Số đo lực hút giữa hai thuộc tính A_1 và A_3 là:

$Aff(A_1, A_3) = \Sigma^{1}_{k=1}\Sigma^{3}_{t=1} acc_t(q_k) = acc_1(q_1)+acc_2(q_1)+acc_3(q_1) = 45$

Tương tự tính cho các cặp còn lại ta có ma trận ái lực (hình 3.5) sau:

	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 3.5 Ma trận ái lực

Thuật toán năng lượng nối BEA (Bond Energy Algorithm)

Đến đây ta có thể phân R làm các mảnh của các nhóm thuộc tính dựa vào sự liên đới (lực hút) giữa các thuộc tính, ví 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 dùng thuật toán năng lượng nối BEA của Hoffer and Severance, 1975 và Navathe., 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 BEA nhận nguyên liệu là một ma trận ái lực thuộc tính (AA), hoán vị các hàng và cột rồi sinh ra một ma trận ái lực tụ (CA) (Clustered affinity matrix). Hoán vị được thực hiện sao cho số đo ái lực chung AM (Global Affinity Measure) là lớn nhất. Trong đó AM là đại lượng:

$$AM=\Sigma^{n}_{i=1}\Sigma^{n}_{j=1}\ aff(A_i, A_j)[aff(A_i, A_{j-1})+aff(A_i, A_{j+1})+aff(A_{i-1}, A_j)+\ aff(A_{i+1}, A_j)]$$

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 $\forall$ 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 hoá 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 tích chất đối xứng nên hàm số vừa được xây dựng ở trên thu lại thành:

$$AM=\Sigma^{n}_{i=1}\Sigma^{n}_{j=1}\ aff(A_i, A_j)[aff(A_i, A_{j-1})+aff(A_i, A_{j+1})]$$

Quá trình sinh ra ma trận tụ lực (CA) được thực hiện qua ba bước:

Bước 1: Khởi gán:

Đặt và cố định một trong các cột của AA vào trong CA. ví dụ cột 1, 2 được chọn trong thuật toán này.

Bước 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 trong i+1 vị trí còn lại trong ma trận CA. Chọn nơi đặt sao cho cho ái lực chung AM lớn nhất. Tiếp tục lặp đến khi không còn cột nào để dặt.

Bước 3: Sắp thứ tự hàng

Một khi thứ tự cột đã được xác định, các hàng cũng được đặt lại để các vị trí tương đối của chúng phù hợp với các vị trí tương đối của cột.

Thuật toán BEA

Input: AA - ma trận ái lực thuộc tính;
Output: CA - ma trận ái lực tụ sau khi đã sắp xếp lại các hàng các cột;
Begin
{Khởi gán: cần nhớ rằng Â 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 by 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 cột.

end. {BEA}

Để hiểu rõ thuật toán chúng ta cần biết cont(*,*,*). Cần nhắc lại số đo ái lực chung AM đã được định nghĩa là:

$$AM=\Sigma^{n}_{i=1}\Sigma^{n}_{j=1} aff(A_i, A_j)[aff(A_i, A_{j-1})+aff(A_i, A_{j+1})]$$

Và có thể viết lại:

$$AM = \Sigma^{n}_{i=1}\Sigma^{n}_{j=1} [aff(A_i, A_j)\ aff(A_i, A_{j-1})+aff(A_i, A_j)\ aff(A_i, A_{j+1})]$$

$$= \Sigma^{n}_{j=1}[\Sigma^{n}_{i=1} aff(A_i, A_j)\ aff(A_i, A_{j-1})+ \Sigma^{n}_{i=1} aff(A_i, A_j)\ aff(A_i, A_{j+1})]$$

Ta định nghĩa cầu nối (Bond) giữa hai thuộc tính A_x, và A_y là:

$$Bond(A_x, A_y)=\Sigma^{n}_{z=1}aff(A_z, A_x)aff(A_z, A_y)$$

Thế thì có thể viết lại AM là:

$$AM = \Sigma^{n}_{j=1}[Bond(A_i, A_{j-1})+Bond(A_i, A_{j+1})]$$

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 $A_i A_j \quad A_{j+1} \ldots A_n$ thuộc nhóm AM"

Khi đó số đo lực hút chung cho những thuộc tính này có thể viết lại:

$$AM_{old} = AM' + AM"+ bond(A_{i-1}, A_i) + bond(A_i, A_j) + bond(A_j, A_i)+$$

$$bond(bond(A_{j+1}, A_j) = \Sigma^{n}_{l=1}[bond(A_l, A_{l-1})+bond(A_i, A_{l+1})] + \Sigma^{n}_{l=i+1}[bond(A_l, A_{l-1})+bond(A_i, A_{l+1})] + 2bond(A_i, A_l))$$

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:

$AM_{new} = AM' + AM'' + bond(A_i, A_k) + bond(A_k, A_i) + bond(A_k, A_j) + bond(A_j, A_k) = AM' + AM'' + 2bond(A_i, A_k) + 2bond(A_k, A_j)$

Vì thế đóng góp thực (net contribution) cho số đo ái lực chung khi đặt thuộc tính A_k giữa A_i và A_j là:

$Cont(A_i, A_k, A_j) = AM_{new} - AM_{old} = 2Bond(A_i, A_k) + 2Bond(A_k, A_j) - 2Bond(A_i, A_j)$

$Bond(A_0, A_k)=0$. Nếu thuộc tính A_k đặt bên phải thuộc tính tận bên phải vì chưa có thuộc tính nào được đặt ở cột k+1 của ma trận CA nên $bond(A_k, A_{k+1})=0$.

Ví dụ**:** Ta xét ma trận được cho trong ví dụ 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:

$Cont(A_1, A_4, A_2)= 2bond(A_1, A_4)+ 2bond(A_4, A_2)-2bond(A_1, A_2)$

Tính mỗi số hạng chúng ta được:

$Bond(A_1, A_4) = \Sigma^4_{z=1} aff(A_z, A_1) aff(A_z, A_4) = aff(A_1,A_1)\ aff(A_1,A_4) + aff(A_2,A_1)\ aff(A_2,A_4) + aff(A_1,A_3)\ aff(A_3,A_4) + aff(A_1,A_4)\ aff(A_4,A_4)$

$= 45*0 + 0*75 + 45*3 + 0*78 = 135$

$Bond(A_4, A_2)= 11865$

$Bond(A_1,A_2) = 225$

Vì thế $cont(A_1, A_4) = 2*135 + 2*11865 + 2*225 = 23550$

Ví dụ: Chúng ta hãy xét quá trình gom tụ các thuộc tính của quan hệ Dự án và dùng ma trận ái lực thuộc tính AA.

Bước khởi đầu chúng ta chép các cột 1 và 2 của ma trận AA vào ma trận CA và bắt đầu thực hiện từ cột thứ ba. Có 3 nơi có thể đặt được cột 3 là: (3-1-2), (1, 3, 2) và (1, 2, 3). Chúng ta hãy tính đóng góp số ái lực chung của mỗi khả năng này.

thứ tự (0-3-1):

$cont(A_0, A_3, A_1) = 2bond(A_0, A_3) + 2bond(A_3, A_1) - 2bond(A_0, A_1)$

$bond(A_0, A_3) = bond(A_0, A_1) = 0$

$bond(A_3, A_1) = 45*48+5*0+53*45+3*0=4410$

$cont(A_0, A_3, A_1) = 8820$

thứ tự (1-3-2)

$cont(A_1, A_3, A_2) = 10150$

thứ tự (2-3-4)

$cont(A_2, A_3, A_4) = 1780$

Bởi vì đóng góp của thứ tự (1-2-3) là lớn nhất, chúng ta đặt A_3 vào bên phải của A_1. Tính toán tương tự cho A_4 chỉ ra rằng cần phải đặt nó vào bên phải của A_2. Cuối cùng các hàng được tổ chức với cùng thứ tự như các cột và các hàng được trình bày trong hình (3.6) sau:

	A_1	A_2		
A_1	45	0		
A_2	0	80		
A_3	45	5		
A_4	0	75		

	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	

	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

	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

Hình 3.6 Gom nhóm các thuộc tính

Trong hình trên chúng ta thấy quá trình tạo ra hai tụ: một ở góc trên trái chứa các giá trị ái lực nhỏ, còn tụ kia ở dưới góc phải chứa các giá trị ái lực cao. Quá trình phân tụ này chỉ ra cách thức tách các thuộc tính của **Project**. Tuy nhiên, nói chung thì ranh rới các phần tách không hoàn toàn rõ ràng. Khi ma trận CA lớn, thường sẽ có nhiều tụ hơn được tạo ra và nhiều phân hoạch được chọn hơn. Do vậy cần phải tiếp cận bài toán một cách có hệ thống hơn.

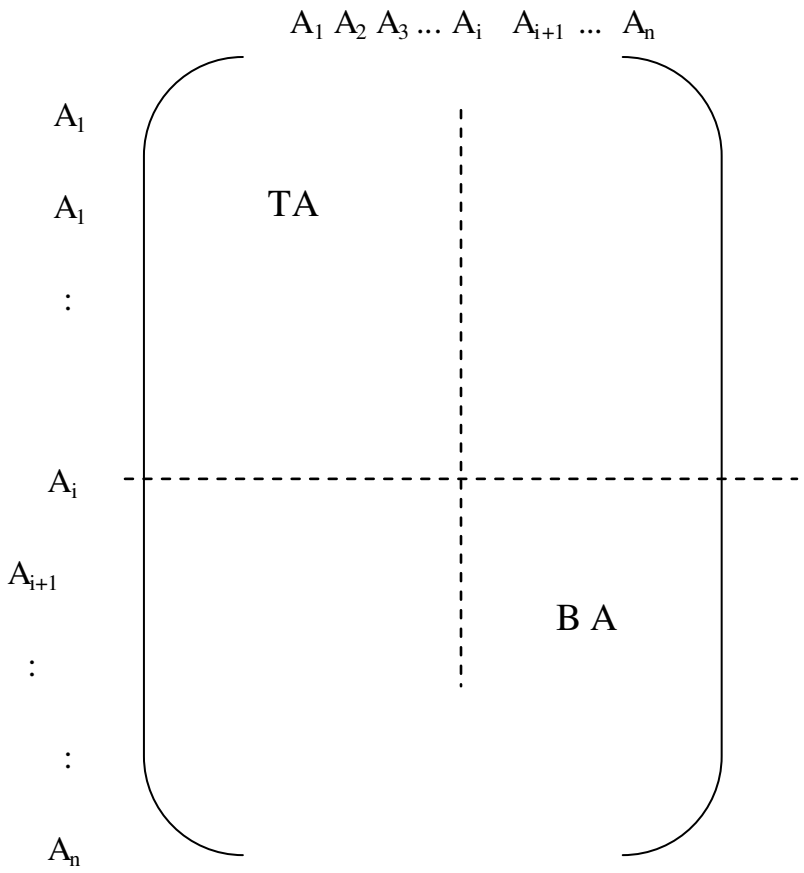
Thuật toán phân hoạch

Mục đích của hành động tách thuộc tính là tìm ra các tập thuộc tính được truy xuất cùng nhau hoặc hầu như là các tập ứng dụng riêng biệt. Xét ma trân thuộc tính tụ (ái lực tụ) (hình 3.7):

Nếu một điểm nằm trên đường chéo được cố định, hai tập thuộc tính này được xác định. Một tập $\{A_1, A_2,..., A_i\}$ nằm tại góc trên trái và tập thứ hai $\{A_{i+1}, A_{i+2},..., A_n\}$ nằm tại góc bên phải và bên dưới điểm này. Chúng ta gọi 2 tập lần lượt là TA, BA. Tập ứng dụng $Q=\{q_1, q_2,...,q_q\}$ và định nghĩa tập ứng dụng chỉ truy xuất TA, chỉ truy xuất BA hoặc cả hai, những tập này được định nghĩa như sau:

$AQ(q_i) = \{A_j \mid use(q_i, A_j)=1\}$

$TQ = \{q_i \mid AQ(q_i) \subseteq TA\}$



Hình 3.7 Ma trân thuộc tính tụ (ái lực tụ)

$AQ(q_i) = \{A_j \mid use(q_i, A_j)=1\}$

$TQ = \{q_i \mid AQ(q_i) \subseteq TA\}$

$BQ = \{q_i \mid AQ(q_i) \subseteq BA\}$

$OQ = Q - \{TQ \cup BQ\}$

Ở đây nảy sinh bài toán tối ưu hoá. Nếu có n thuộc tính trong quan hệ thì sẽ có n-1 vị trí khả hữu có thể là điểm phân chia trên đường chéo chính của ma trận thuộc

tính tụ cho quan hệ đó. Vị trí tốt nhất để phân chia là vị trí sinh ra tập TQ và BQ sao cho tổng các truy xuất chỉ một mảnh là lớn nhất còn tổng truy xuất cả hai mảnh là nhỏ nhất. Vì thế chúng ta định nghĩa các phương trình chi phí như sau:

$$CQ = \sum_{qi \in Q} \sum_{\forall Sj} ref_j(q_i) acc_j(q_i)$$

$$CTQ = \sum_{qi \in TQ} \sum_{\forall Sj} ref_j(q_i) acc_j(q_i)$$

$$CBQ = \sum_{qi \in BQ} \sum_{\forall Sj} ref_j(q_i) acc(q_i)$$

$$COQ = \sum_{qi \in OQ} \sum_{\forall Sj} ref_j(q_i) acc(q_i)$$

Mỗi phương trình ở trên đếm tổng số truy xuất đến các thuộc tính bởi các ứng dụng trong các lớp tương ứng của chúng. Dựa trên số liệu này, bài toán tối ưu hoá được định nghĩa là bài toán tìm điểm x ($1 \le x \le n$) sao cho biểu thức:

$$Z = CTQ + CBQ - COQ^2$$

lớn nhất. Đặc trưng quan trọng của biểu thức này là nó định nghĩa hai mảnh sao cho giá trị của CTQ và CBQ càng gần bằng nhau càng tốt. Điều này cho phép cân bằng tải trọng xử lý khi các mảnh được phân tán đến các vị trí khác nhau. Thuật toán phân hoạch có độ phức tạp tuyến tính theo số thuộc tính của quan hệ, nghĩa là O(n).

Thuật toán PARTITION

Input: CA: ma trận ái lực tụ; R: quan hệ; ref: ma trận sử dụng thuộc tính;
acc: ma trận tần số truy xuất;

Output: F: tập các mảnh;

Begin

{xác định giá trị z cho cột thứ nhất}

{các chỉ mục trong phương trình chi phí chỉ ra điểm tách}

tính CTQ_{n-1}

tính CBQ_{n-1}

tính COQ_{n-1}

best ← $CTQ_{n-1}*CBQ_{n-1} - (COQ_{n-1})^2$
do {xác định cách phân hoạch tốt nhất}
begin
for i from n-2 to 1 by -1 do
begin
tính CTQ_i
tính CBQ_i
tính COQ_i
z ← $CTQ_i*CBQ_i - (COQ_i)^2$
if z > best then
begin
best ← z
ghi nhận điểm tách bên vào trong hành động xê dịch
end-if
end-for
gọi SHIFT(CA)
end-begin
until không thể thực hiện SHIFT được nữa
Xây dựng lại ma trận theo vị trí xê dịch
$R_1 \leftarrow \prod_{TA}(R) \cup K$ {K là tập thuộc tính khoá chính của R}
$R_2 \leftarrow \prod_{BA}(R) \cup K$
F ← $\{R_1, R_2\}$
End. {partition}

Áp dụng cho ma trận CA từ quan hệ Project, kết quả là định nghĩa các mảnh

F_{Proj}={Project1, Project2}

Trong đó:

Project1={A_1, A_3} và Project2= {A_1, A_2, A_4}. Vì thế

Project1={JNO, Budjet}

Project2={JNO, JNAME, LOC}

JNO là thuộc tính khoá của **Project**

Kiểm tra tính đúng đắn:

Tính đầy đủ: được bảo đảm bằng thuật toán PARTITION vì mỗi thuộc tính của quan hệ toàn cục được đưa vào một trong các mảnh.

Tính tái thiết được: đối với quan hệ R có phân mảnh dọc $F_R=\{R_1, R_2,...., R_r\}$ và các thuộc tính khoá K

$$R= \bowtie_K R_i \ , \forall \ R_i \in F_R$$

Do vậy nếu điều kiện mỗi R_i là đầy đủ phép toán nối sẽ tái thiết lại đúng R. Một điểm quan trọng là mỗi mảnh R_i phải chứa các thuộc tính khoá của R.

3.2.4 Sự phân mảnh hỗn hợp

Cuối cùng chúng ta xét sự phân mảnh hỗn hợp. Cách thức dễ nhất để thực hiện sự phân mảnh hỗn hợp là:

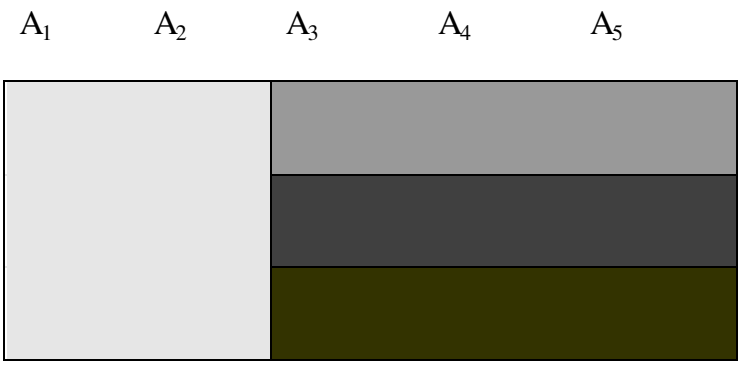
1. Áp dụng sự phân mảnh ngang đối với các phân mảnh dọc.
2. Áp dụng sự phân mảnh dọc đối với các phân mảnh ngang.

Mặc dầu các phép toán trên có thể lặp lại một cách đệ qui, nhưng trên thực tiễn sự phân mảnh không nên quá hai cấp.

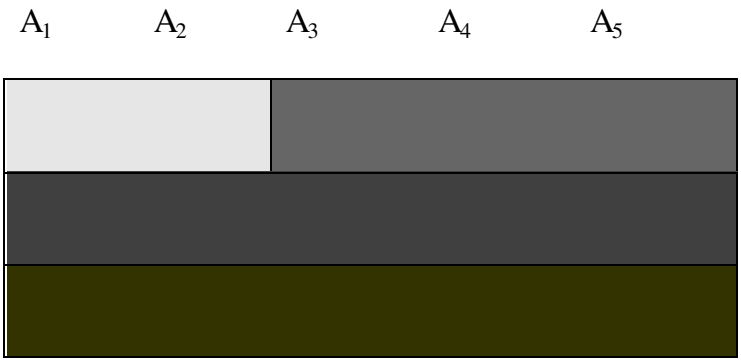
Hình 3.8 thể hiện thứ tự của sự phân mảnh như sau:

Sự phân mảnh ngang được áp dụng ngay trên một phân mảnh dọc.

Sự phân mảnh dọc được áp dụng ngay trên một phân mảnh ngang.



Sự phân mảnh dọc rồi sau đó phân mảnh ngang



Sự phân mảnh ngang sau đó phân mảnh dọc

Hình 3.8 Sự phân mảnh hỗn hợp của quan hệ R(A_1, A_2, A_3, A_4, A_5)

Ví dụ tổng quát

Xét lại quan hệ phổ quát EMP được phân mảnh dọc thành EMP_1 và EMP_2. Giả sử các ứng dụng về công việc được điều hành tại các phòng ban mà sử dụng phân mảnh EMP_2 tham khảo đến xác suất 80% các bộ của các phòng ban lân cận với site mà các ứng dụng đó được gọi. Vì thế EMP2 có thể được phân mảnh ngang tiếp tục theo nhóm các phòng ban.

3.3 Sự cấp phát các phân mảnh

3.3.1 Bài toán cấp phát

Giả sử đã có một tập các mảnh $F=\{F_1, F_2, ...,F_n\}$ và một mạng bao gồm các vị trí $S=\{S_1, S_2, ...,S_m\}$ trên đó có một tập các ứng dụng $Q=\{q_1, q_2, ...,q_q\}$ đang chạy.

Bài toán cấp phát là tìm một phân phối "tối ưu" của F cho S.

Tính tối ưu có thể được định nghĩa ứng với hai số đo:

- Chi phí nhỏ nhất: Hàm chi phí có chi lưu mảnh F_i vào vị trí S_j, chi phí vấn tin mảnh F_i vào vị trí S_j, chi phí cập nhật F_i tại tất cả mọi vị trí có chứa nó và chi phí tryền dữ liệu. Vì thế bài toán cấp phát cố gắng tìm một lược đồ cấp phát với hàm chi phí tổ hợp nhỏ nhất.

- Hiệu năng: 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 thuật giải heuristec tốt để có lời giải gần tối ưu.

3.3.2 Yêu cầu về thông tin

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.

Thông tin về CSDL

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 ký hiệu là $sel_i(F_j)$

Kích thước của 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.

Thông tin về ứng dụng

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}). ví dụ chúng có thể đếm số truy xuất khối cần phải thực hiện theo yêu cầu vấn tin.

Chúng ta định nghĩa hai ma trận UM và RM với các phần tử tương ứng u_{ij} và r_{ij} được đặc tả tương ứng như sau:

$$u_{ij} = \begin{cases} 1 & \text{nếu vấn tin } q_i \text{ có cập nhật mảnh } F_j \\ 0 & \text{trong trường hợp ngược lại} \end{cases}$$

$$r_{ij} = \begin{cases} 1 & \text{nếu vấn tin } q_i \text{ có cập nhật mảnh } F_j \\ 0 & \text{trong trường hợp ngược lại} \end{cases}$$

Một véctơ O gồm các giá trị o(i) cũng được định nghĩa, với o(i) đặc tả vị trí đưa ra câu vấn tin q_i .

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 phương pháp đánh giá đơn giản.

+ Chi phí đơn vị tính để lưu dữ liệu tại vị trí S_k sẽ được ký hiệu là USC_k.

+ 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.

Thông tin về mạng

Chúng ta giả sử tồn tại một mạng đơn giản, g_{ij} biểu thị cho 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.

3.3.3. Mô hình cấp phát

Mô hình cấp phát có mục tiêu làm giảm thiểu tổng chi phí xử lý và lưu trữ dữ liệu trong khi vẫn cố gắng đáp ứng được các đòi hỏi về thời gian đáp ứng. 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.

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:

$$TOC = \sum_{\forall q_i \in Q} QPC_i + \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} STC_{jk}$$

với QPC_i là chi phí xử lý câu vấn tin ứng dụng q_i, và STC_{jk} là chi phí lưu mảnh F_j tại vị trí S_k.

Chúng ta hãy xét chi phí lưu trữ trước. Nó được cho 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 phần: Chi phí xử lý chỉ đọc và chi phí xử lý chỉ cập nhật. Ở đây chúng tỗi đã chọn một hướng tiếp cận khác trong mô hình cho bài toán DAP và xác định nó như là chi phí xử lý vấn tin bao gồm 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:

$$AC_i = \sum_{\forall S_k \in S} \sum_{\forall F_j \in F} (u_{ij} * UR_{ij} + r_{ij} * RR_{ij}) * x_{jk} * LPC_k$$

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.

Một vấn đề rất quan trọng cần đề cập ở đây. Hàm chi phí truy xuất giả sử rằng việc xử lý một câu vấn tin có bao gồm cả việc phân rã nó thành một tập các vấn tin con hoạt tác trên một mảnh được lưu tại vị trí đó, theo sau là truyền kết quả trở lại về vị trí đã đưa ra vấn tin.

Hệ số chi phí duy trì tính toàn vẹn có thể được mô tả rất giống thành phần xử lý ngoại trừ chi phí xử lý cục bộ một đơn vị cần thay đổi nhằm phản ánh chi phí thực sự để duy trì tính toàn vẹn.

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 và 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 tryền 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 o(i) của q_i đến tất cả bản sao cập nhật. Số hạng thứ hai dành cho thông báo xác nhận.

Thành phần chi phí chỉ đọc có thể đặc tả là:

$$TCR_i = \sum_{F_j \in F} \min_{S_k \in S} (u_{ij} * x_{jk} * g_{o(i),k} + r_{ij} * x_{jk*} (sel_i(F_j) * length(F_j)/fsize) * g_{k,o(i)})$$

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 những 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 tính cho vấn tin q_i có thể được tính là:

$$TC_i = TCU_i + TCR_i$$

Ràng buộc

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 $q_i \ \forall q_i \in Q$

Người ta thích đặc tả số đo chi phí của hàm theo thời gian bởi vì nó đơn giản hoá đặc tả về ràng buộc thời gian thực thi.

Ràng buộc lưu trữ là: $\sum_{\forall Fj \in F} STC_{jk} \leq$ khả năng lưu trữ tại vị trí S_k, $\forall S_k \in S$

Trong đó ràng buộc xử lý là:

$\sum$ tải trọng xử lý của q_i tại vị trí $S_k \leq$ khả năng xử lý của S_k, $\forall S_k \in S$.

Chương 4 SỰ TRONG SUỐT PHÂN TÁN

Mục tiêu

Chương này giới thiệu sự trong suốt phân tán theo ba mức : trong suốt phân tán, trong suốt vị trí và trong suốt ánh xạ cục bộ đối với hai ứng dung:

1. *Ứng dụng chỉ đọc (tìm kiếm) và*
2. *Ứng dụng cập nhật dữ liệu.*

4.1 Sự trong suốt phân tán của ứng dụng chỉ đọc

Để hiểu được sự trong suốt phân tán của ứng dụng chỉ đọc, chúng ta sẽ xem xét các ví dụ được minh họa bằng ngôn ngữ tựa Pascal có nhúng ngôn ngữ SQL. Trong các ví dụ này chúng ta có một số chú ý sau:

- Các biến có kiểu chuỗi ký tự.
- Nhập xuất được thực hiện qua các thủ tục chuẩn: Read(filename, variable), write(filename, variable). Nếu nhập xuất được thực hiện tại một trạm thì filename sẽ là "terminal".
- Trong các pháp biểu SQL, các biến của ngôn ngữ Pascal sẽ bắt đầu bằng ký tự '$'.

Ví dụ: Xét câu truy vấn : Tìm tên của người cung cấp khi biết mã số người cung cấp.

```
Select NAME into $NAME
From SUPPLIER
Where SNUM = $SNUM
```

Trong câu truy vấn này $NAME là biến xuất còn $SNUM là biến nhập.

- Các biến của Pascal sử dụng để liên hệ với hệ quản trị cơ sở dữ liệu phân tán bắt đầu bằng ký tự '#'.

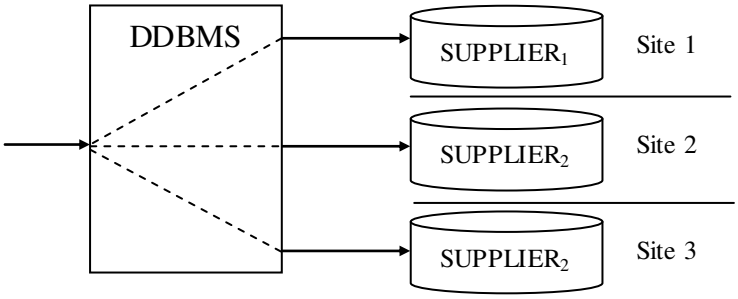
Ví dụ: Sau khi truy vấn một câu SQL, biến luận lý #FOUND = TRUE nếu kết quả trả về khác rỗng. Biến #OK = TRUE nếu phép toán thực hiện đúng bởi hệ quản trị cơ sở dữ liệu.

Các mức độ trong suốt sẽ được xét từ cao đến thấp qua hai trường hợp sau.

a. Xét trên một lược đồ quan hệ toàn cục

Mức 1: Sự trong suốt phân tán

```
Read (terminal, $SNUM)
        Select NAME into $NAME
        From SUPPLIER
        Where SNUM = $SNUM
Write(terminal, $NAME)
```



Hình 4.1a Sự trong suốt phân tán

Nhận xét: Theo hình 4.1a và đoạn lệnh ở trên, chúng ta thấy câu truy vấn trên tương tự như câu truy vấn cục bộ, không cần chỉ ra các phân mảnh cũng như các vị trí cấp phát cho các phân mảnh đó. Khi đó người sử dụng không hề có cảm giác là đang thao tác trên một câu truy vấn phân tán.

Mức 2: Sự trong suốt vị trí (location)

Read (terminal, $SNUM)

Select NAME into $NAME

From $SUPPLIER_1$

Where SNUM = $SNUM

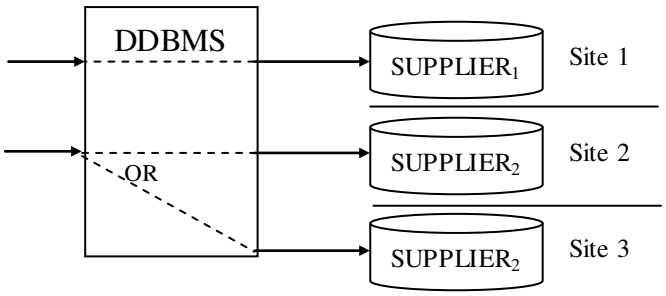
If not #Found then

Select NAME into $NAME

From $SUPPLIER_2$

Where SNUM = $SNUM

Write(terminal, $NAME)



Hình 4.1b Sự trong suốt vị trí

Nhận xét: Người sử dụng phải cung cấp các phân mảnh cụ thể cho câu truy vấn nhưng không cần chỉ ra vị trí của các phân mảnh.

Sự trong suốt vị trí ở hình 4.1b có thể được viết như sau:

Read(Terminal, $SNUM)

Read(Terminal, $CITY)

Case $CITY Of

"SF": Select NAME into $NAME

From $SUPPLIER_1$

Where SNUM = $SNUM;

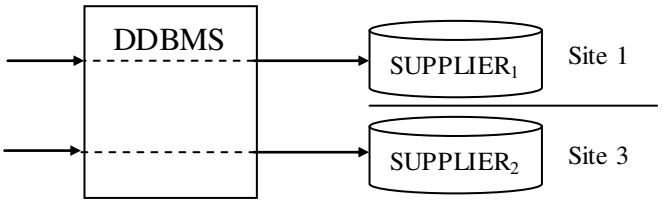
"LA": Select NAME into $NAME

From $SUPPLIER_2$

```
            Where SNUM = $SNUM
        End;
        Write(Terminal, $NAME)
```

Mức 3: Sự trong suốt ánh xạ cục bộ (hình 4.1c)

```
Read (terminal, $SNUM)
        Select NAME into $NAME
        From SUPPLIER1 AT SITE 1
        Where SNUM = $SNUM
If not #Found then
        Select NAME into $NAME
        From SUPPLIER2 AT SITE 2
        Where SNUM = $SNUM
Write(terminal, $NAME)
```

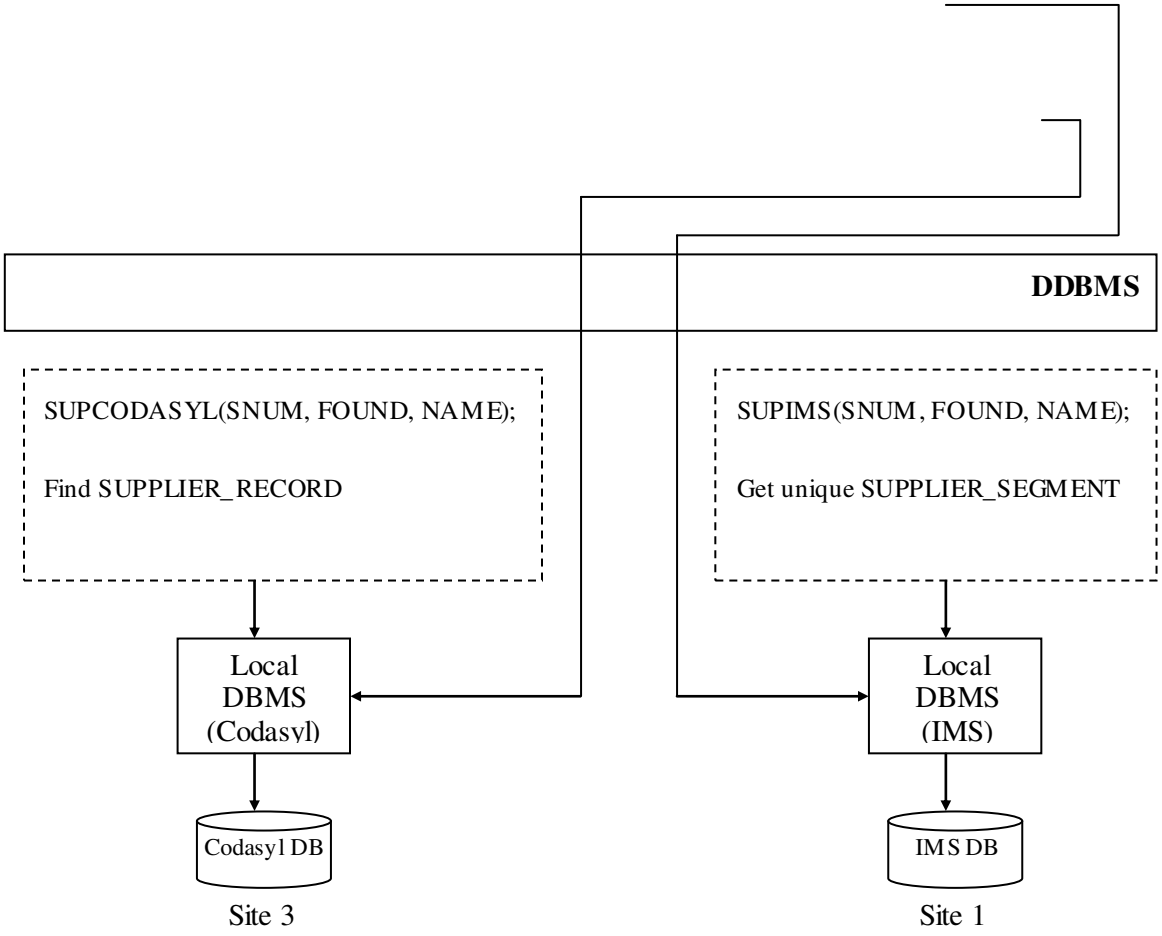


Hình 4.1c Sự trong suốt ánh xạ cục bộ

Nhận xét: tại mức trong suốt này người sử dụng phải cung cấp các phân mảnh và vị trí cấp phát của chúng.

Mức 4: Không trong suốt (hình 4.2)

```
Read(Terminal,SSUPNUM)
Execute SSUPIMS(SSUPNUM, $FOUND, $NAME) at site 1;
if not $FOUND then
Execute SSUPCODASYL(SSUPNUM, $FOUND, $NAME) at site 3;
Write(Terminal,$NAME)
```



Hình 4.2 Một ứng dụng trên cơ sở dữ liệu phân tán không đồng nhất và không trong suốt

Nhận xét: Tại mức thấp nhất này chúng ta cần phải viết lệnh theo hệ quản trị cơ sở dữ liệu tương ứng.

b. Xét trên hai lược đồ quan hệ toàn cục

Chúng ta hãy xét một ví dụ phức tạp hơn. Giả sử cần tìm tên những nhà cung cấp mặt hàng có mã số cho trước.

Mức 1: Trong suốt phân mảnh

```
Read(Terminal, $PNUM)
Select NAME into $NAME
from SUPPLIER, SUPPLY
where SUPPLIER.SNUM = SUPPLY.SNUM and
      SUPPLY.PNUM = $PNUM
write(Terminal, $NAME)
```

Mức 2: Trong suốt vị trí

```
Read(Terminal, $PNUM)
Select NAME into $NAME
from SUPPLIER1, SUPPLY1
where SUPPLIER1.SNUM = SUPPLY1.SNUM and
      SUPPLY1.PNUM = $PNUM
if not #FOUND then
 Select NAME into $NAME
from SUPPLIER2, SUPPLY2
where SUPPLIER2.SNUM = SUPPLY2.SNUM and
      SUPPLY2.PNUM = $PNUM
write(Terminal, $NAME)
```

Mức 3: Trong suốt ánh xạ cục bộ

Giả sử các sơ đồ cấp phát các phân mảnh của quan hệ SUPPLY và SUPPLIER như sau: $SUPPLIER_1$: Tại site 1

$SUPPLIER_2$: Tại site 2

$SUPPLY_1$: Tại site 3

$SUPPLY_2$: Tại site 4

Read(Terminal, $PNUM)
Select SNUM into $SNUM
from $SUPPLY_1$ at site 3
where $SUPPLY_1$.PNUM = $PNUM
if not #FOUND then
begin
send $SNUM from site 3 to site 1
Select NAME into $NAME
from $SUPPLIER_1$ at site 1
where $SUPPLIER_1$.SNUM = $SNUM
end
else begin
Select SNUM into $SNUM
from $SUPPLY_2$ at site 4
where $SUPPLY_2$.PNUM = $PNUM
send $SNUM from site 4 to site 2
Select NAME into $NAME
from $SUPPLIER_2$ at site 2
where $SUPPLIER_2$.SNUM = $SNUM
end;
write(Terminal, $NAME)

4.2 Sự trong suốt phân tán đối với các ứng dụng cập nhật

Trong phần trước, chúng ta chỉ xét các ứng dụng tìm kiếm dữ liệu trong cơ sở dữ liệu. Phần này chúng ta sẽ xem xét các ứng dụng cập nhật dữ liệu trong cơ sở dữ liệu phân tán. Bài toán cập nhật ở đây chỉ xét dưới khía cạnh trong suốt phân tán đối với các lập trình viên, trong khi bài toán bảo đảm tính nguyên tử của các giao tác cập nhật sẽ được xem xét ở chương sau.

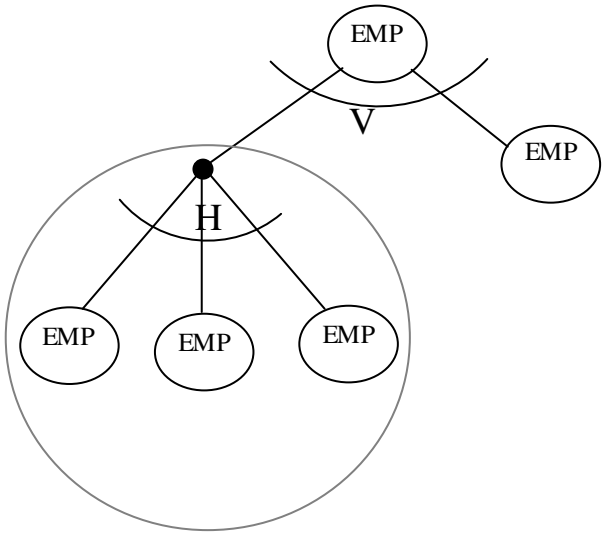
Các mức trong suốt phân tán cũng được cũng được phân tích như trong các ứng dụng chỉ đọc. Tuy nhiên một phép cập nhật phải được thực hiện trên tất cả các bản sao của một mục dữ liệu trong khi phép tìm kiếm chỉ cần thực hiện trên một bản sao. Điều này có nghĩa là nếu hệ quản trị cơ sở dữ liệu không hỗ trợ sự trong suốt vị trí và sự trong suốt nhân bản thì lập trình viên chịu trách nhiệm thực hiện mọi cập nhật được yêu cầu.

rong việc hỗ trợ sự trong suốt phân tán cho các ứng dụng cập nhật có một vấn đề khac phức tạp hơn việc cập nhật tất cả các bản sao của một mục dữ liệu. Đó là vấn đề di chuyển dữ liệu sau khi cập nhật.

Xét ví dụ sau: Điều gì xảy ra khia cập nhật lại giá trị của thuộc tính CITY trong quan hệ SUPPLIER. Rõ ràng, các bộ Supplier phải được chuyển từ một phân mảnh này đến phân mảnh khác. Hơn nữa, các bộ của quan hệ SUPPLY mà tham khảo đến cùng Supplier cũng phải thay đổi phân mảnh vì quan hệ SUPPLY có một phân mảnh dẫn xuất. Một cách trực quan chúng ta dễ dàng thấy việc thay đổi giá trị của một thuộc tính mà chúng ta định nghĩa trong lược đồ phân mảnh có dẫn đến các hệ quả phức tạp. Mức độ mà các hệ quản trị cơ sở dữ liệu phân tán quản lý các hệ quả đó chính là đặc trưng cho các mức trong suốt phân tán cho sự cập nhật.

Để minh họa cho các phép toán di chuyển dữ liệu trong việc cập nhật dữ liệu phân tán người ta sử dụng cây con cập nhật.

Xét một thuộc tính A được sử dụng trong vị từ phân mảnh ngang. Cây con cập nhật của A là cây con mà có nút gốc là nút đại diện cho sự phân mảnh ngang ở trên.



V: Phép toán phân mảnh dọc

H: Sự phân mảnh ngang

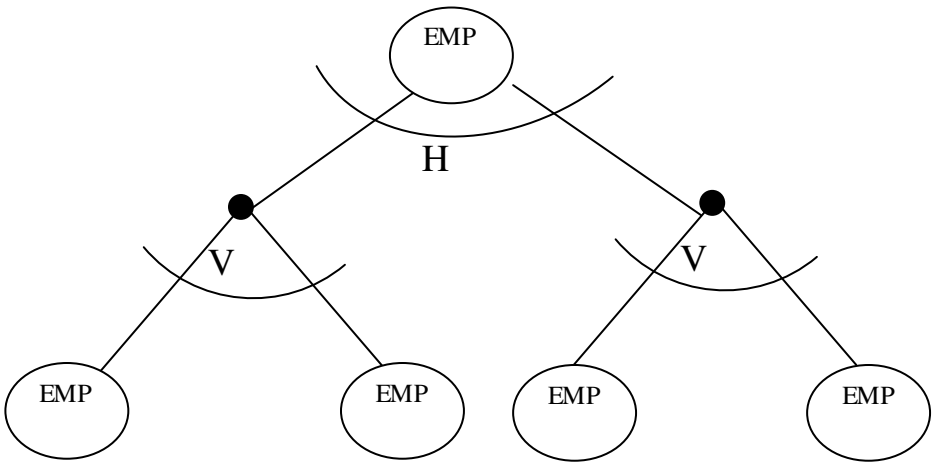
Hình 4.3 Cây con cập nhật của thuộc tính DEPTNUM trong cây phân mảnh của quan hệ EMP

Ví dụ: hình 4.3 minh họa cây con cập nhật cho thuộc tính DEPTNUM.

Các ảnh hưởng của sự thay đổi giá trị của một thuộc tính chỉ giới hạn trong các phân mảnh ở nút lá của cây con cập nhật.

Ví dụ: Một sự thay đổi giá trị của thuộc tính DEPTNUM chỉ ảnh hưởng đến EMP_1, EMP_2 và EMP_3. Một bộ có thể di chuyển giữa hai trong ba phân mảnh trên.

Xét một ví dụ khác phức tạp hơn. Giả sử quan hệ EMP có cây phân mảnh như hình 4.4a.



$EMP_1 = PJ_{EMPNUM, NAME, SAL, TAX} SL_{DEPTNUM <10}(EMP)$

$EMP_2 = PJ_{EMPNUM, MGRNUM, DEPTNUM} SL_{DEPTNUM <10}(EMP)$

$EMP_3 = PJ_{EMPNUM, NAME, DEPTNUM} SL_{DEPTNUM >10}(EMP)$

$EMP_4 = PJ_{EMPNUM, SAL, TAX, MGRNUM} SL_{DEPTNUM >10}(EMP)$

Hình 4.4a Cây phân mảnh khác của quan hệ EMP

EMP_1

EMPNUM	NAME	SAL	TAX
100	SMITH	10000	1000

EMP_2

EMPNUM	MGRNUM	DEPTNUM
100	20	3

Trước khi cập nhật

Sau khi cập nhật

EMPNUM	NAME	DEPTNUM
100	SMITH	15

EMPNUM	SAL	TAX	MGRNUM
100	10000	1000	20

Hình 4.4b Hệ quả của việc cập nhật DEPTNUM của EMPNUM=100

Trong trường hợp này cây con cập nhật của thuộc tính DEPTNUM tương như cây phân mảnh. Sự ảnh hưởng của việc thay đổi giá trị của DEPTNUM của bộ có EMPNUM=100 từ 3 thành 15 được minh họa ở hình 2.6b. Sự cập nhật của bộ này chỉ

thuộc về cây con trái, sau khi cập nhật nó trở thành một phần cây con phải. Chúng ta nhận thấy không chỉ dữ liệu được chuyển giữa các mảnh mà bộ này cũng được tổ hợp lại theo một cách khác.

Bây giờ chúng ta sẽ xem xét cac mức trong suốt phân tán cho một ứng dụng cập nhật đơn giản.

Mức 1: Sự trong suốt phân tán

Mức này minh hoạ chương trình ứng dụng cập nhật dữ liệu như trong một cơ sở dữ liệu không phân tán. Bởi thế các lập trình viên không cần biết thuộc tính nào được dùng để phân mảnh. Để thay đổi giá trị DEPTNUM của employee có EMPNUM=100, đoạn chương trình được viết như sau:

```
Update EMP
set DETPNUM=15
where EMPNUM =100
```

Mức 2: Sự trong suốt vị trí

Tại mức này, lập trình viên phải làm việc với các phân mảnh một cách tường minh.

Đoạn chương trình được viết như sau:

```
Select NAME, SAL, TAX into $NAME, $SAL, $TAX
from EMP1
where EMPNUM=100;
Select MGRNUM into $MGRNUM
from EMP2
where EMPNUM=100;
Insert into EMP3 (EMPNUM, NAME, DEPTNUM)
Values (100, $NAME, 15);
```

Insert into EMP_4 (EMPNUM, SAL, TAX, MGRNUM)

Values (100, $SAL, $TAX, $MGRNUM);

Delete EMP_1 where EMPNUM = 100;

Delete EMP_2 where EMPNUM = 100;

Mức 3: Sự trong suốt ánh xạ cục bộ

Tại mức này ứng dụng phải giải quyết vị trí của các phân mảnh một cách tường minh. Giả sử các phân mảnh của quan hệ EMP được cấp phát như sau:

EMP_1 : site 1 và 5

EMP_2 : site 2 và 6

EMP_3 : site 3 và 7

EMP_4 : site 4 và 8

Đoạn chương trình được viết như sau:

Select NAME, SAL, TAX into $NAME, $SAL, $TAX

from EMP_1 at site 1

where EMPNUM=100;

Select MGRNUM into $MGRNUM

from EMP_2 at site 2

where EMPNUM=100;

Insert into EMP_3 (EMPNUM, NAME, DEPTNUM) at site 3

Values (100, $NAME, 15);

Insert into EMP_3 (EMPNUM, NAME, DEPTNUM) at site 7

Values (100, $NAME, 15);

Insert into EMP_4 (EMPNUM, SAL, TAX, MGRNUM) at site 4

Values (100, $SAL, $TAX, $MGRNUM);

Insert into EMP_4 (EMPNUM, SAL, TAX, MGRNUM) at site 8

Values (100, $SAL, $TAX, $MGRNUM);

Delete EMP_1 at site 1 where EMPNUM = 100;

Delete EMP_1 at site 5 where EMPNUM = 100;

Delete EMP_2 at site 2 where EMPNUM = 100;

Delete EMP_2 at site 6 where EMPNUM = 100;

4.3 Các nguyên tắc truy xuất cơ sở dữ liệu phân tán

Trong các ví dụ ở phần trước chúng ta chỉ xét các truy vấn dữ liệu chỉ trả về một giá trị. Tuy nhiên trong trường hợp tổng quát, một câu truy vấn có thể trả về một quan hệ. Khi đó chúng ta sẽ qui ước dùng tham số có tiếp vị ngữ "REL", được xem như một file, để nhận kết quả trả về.

Ví dụ: Cho câu SQL sau:

Select EMPNUM, NAME INTO **$EMP_REL($EMPNUM, $NAME)**
from EMP

Bây giờ chúng ta sẽ xem xét các cách try xuất cơ sở dữ liệu và đánh giá hiệu quả của các cách đó theo yêu cầu sau:

cho biết danh sách các sản phẩm mà các nhà cung cấp (được nhập) đã cung cấp.

Cách 1: Cơ sở dữ liệu được truy xuất với mỗi giá trị $SNUM

```
Repeat
 read(terminal, $SNUM);
 select PNUM into $PNUM_REL($PNUM)
 from SUPPLY
 where SNUM = $SNUM;
 repeat
  read($PNUM_REL, $PNUM)
```

```
 write(terminal, $PNUM)
 until END-OF-$PNUM_REL
until END-OF-TERMINAL-INPUT
```

Cách 2: Cơ sở dữ liệu được truy xuất sau khi tất cả giá trị của $SNUM được nhập

```
Repeat
 read(terminal, $SNUM);
 write($SNUM_REL($SNUM), $SNUM)
Until END-OF-TERMINAL-INPUT
 select PNUM into $PNUM_REL($PNUM)
 from SUPPLY, $SNUM_REL
 where SUPPLY.SNUM = $SNUM_REL.$SNUM;
 repeat
  read($PNUM_REL, $PNUM)
  write(terminal, $PNUM)
 until END-OF-$PNUM_REL
```

Cách 3: Cơ sở dữ liệu được truy xuất trước khi nhập giá trị $SNUM

```
Select PNUM, SNUM into $TEMP_REL($TEMP_PNUM, $TEMP_NUM)
from SUPPLY;
Repeat
 read(terminal, $SNUM);
 select $TEMP_PNUM into $TEMP2_REL($TEMP2_PNUM)
 from $TEMP_REL
 where $TEMP_PNUM = $SNUM;
 repeat
  read($TEMP2_REL, $TEMP2_PNUM);
  write(terminal, $TEMP2_PNUM);
```

```
    until END-OF-$TEMP2_PNUM
  until END-OF-TERMINAL-INPUT
```

Nhận xét về hiêu quả của ba cách trên?

Một vấn đề nữa liên quan đến cách truy xuất cơ sở dữ liệu phân tán là hiện tượng có chung các biểu thức con.

Xét ví dụ tìm kiếm tên của các công nhân làm việc trong các phòng ban của cùng một vùng (được nhập) và tên các sản phẩm được lưu trong các phòng ban của cùng một vùng đó.

Ta có hai cách trả lời sau:

Cách 1: Các truy vấn độc lập

```
        Read(terminal, $AREA)
...
select NAME into $NAME_REL($NAME)
from EMP, DEPT
where EMP.DEPTNUM = DEPT.DEPTNUM and
      DEPT.AREA = $AREA
...
Select PNUM into $PNUM_REL($PNUM)
from SUPPLY, DEPT
where SUPPLY.DEPTNUM = DEPT.DEPTNUM
      and DEPT.AREA = $AREA
```

Cách 2:

```
    Read(terminal, $AREA);
...
```

```
select DEPTNUM into $NUM_REL(DEPTNUM)
from DEPT
where AREA=$AREA;
...
select NAME into $NAME_REL($NAME)
from EMP, $NUM_REL
where EMP.DEPTNUM = $NUM_REL.$DEPTNUM;
...
Select PNUM into $PNUM_REL($PNUM)
from SUPPLY, $NUM_REL
where SUPPLY.DEPTNUM = $NUM_REL.$DEPTNUM;
...
```

Chương 5 TỐI ƯU HÓA TRUY VẤN PHÂN TÁN

Mục tiêu

Chương này đề cập đến vấn đề tối ưu hoá trong cơ sở dữ liệu phân tán nghĩa là giảm chi phí bộ nhớ trung gian, giảm thời gian truy vấn cũng như giảm thời gian truyền dữ liệu trong các truy vấn phân tán.

Các vấn đề được đề cập trong chương này như sau:

Biểu thức chuẩn tắc của truy vấn:

Tối ưu hóa truy vấn trong cơ sở dữ liệu tập trung:

Tối ưu hóa trong cơ sở dữ liệu phân tán:

Mở đầu

Chương này trình bày về các bước thực hiện trong việc tối ưu hóa truy vấn trong cơ sở dữ liệu tập trung và trong cơ sở dữ liệu phân tán, các tiêu chuan tối ưu hóa nhằm để làm giảm thời gian thực hiện truy vấn, giảm vùng nhớ trung gian và chi phí truy vấn thông trong qúa trình thực hiện truy vấn, bộ suy diễn dùng trong đơn giản hóa biểu thức đại số quan hệ của truy vấn.

Chương này sử dụng một cơ sở dữ liệu sau nay để minh họa cho các nội dung được trình bày trong chương này :

Sinhvien (masv, hoten, tuoi, malop)

Lop (malop, tenlop, matl, tenkhoa)

Hoc (masv, mamh. Diem)

Monhoc(mamh, tenmh)

Trong đó :

Sinhvien: chứa thông tin về sinh viên gồm: mã sinh viên (masv), họ tên (hoten), Tuổi (tuoi), thuộc lớp (malop). Khóa là masv.

Lop: chứa thông tin về lớp học gồm: mã lớp (malop), tên lớp (tenlop), mã lớp Trưởng (malt), thuộc khoa (tenkhoa). Khóa là malop.

Monhoc: chứa thông tin về moan học gồm: mã môn học (mamh), tên môn học (tenmh).

Hoc: chứa thông tin về sinh viên (masv) học môn học (mamh) có điểm thi cuối Kỳ (diem). Khóa là masv và mamh.

5.1. Biểu thức chuẩn tắc của truy vấn

5.1.1. Truy vấn

Truy vấn (query) là một biểu thức được biểu diễn bằng một ngôn ngữ thích hợp và dùng để xác định một phần dữ liệu được chứa trong cơ sở dữ liệu.

Một truy vấn có thể được dùng để xác định ngữ nghĩa của một ứng dụng, hoặc nó có thể được dùng để xác định công việc can được thực hiện bởi một ứng dụng nhằm để truy xuất cơ sở dữ liệu.

Ví dụ: Xét truy vấn cho biết tên lớp của lớp có mã lớp là 'MT' . Truy vấn này có thể được biểu diễn bởi một biểu thức đại số quan hệ như sau :

$$\Pi_{Tenlop}(\sigma_{malop='MT'}(lop))$$

Một truy vấn có thể được biểu diễn bởi một cây toán tử. Một *cây toán tử operator tree* của một truy vấn, còn được gọi là *cây truy vấn (query tree)* hoặc *cây đại số quan hệ (relationalalgebra tree)*, là một cây mà một nút lá là một quan hệ trong cơ sở dữ liệu, và

một nút khác lá (nút trung gian hoặc nút gốc) là một quan hệ trung gian được tạo ra bởi một phép toán đại số quan hệ. Chuỗi các phép toán đại số quan hệ được thực hiện từ các nút lá đến nút gốc để tạo ra kết quả truy vấn.

Ví dụ : Truy vấn trên có thể được biểu diễn bằng một cây toán tử như sau:

Π_{Tenlop}

|

$\sigma_{malop='MT'}$

|

lop

5.1.2. Biểu thức chuẩn tắc của truy vấn

Biểu thức chuẩn tắc (canonical expression) của một biểu thức đại số quan hệ trên lược đồ toàn cục là một biểu thức có được bằng cách thay thế mỗi tên quan hệ toàn cục xuất hiện trong biểu thức bởi biểu thức tái lập của quan hệ toàn cục này.

Tương tự, chúng ta có thể biến đổi một cây toán tử trên lược đồ toàn cục thành một cây toán tử trên lược đồ phân mảnh bằng cách thay thế các nút lá của cây đầu tiên bằng các biểu thức chuẩn tắc của chúng. Một điều quan trong là bây giờ các nút lá của cây toán tử của biểu thức chuẩn tắc là các mảnh thay vì là các quan hệ toàn cục.

Ví dụ : Giả sử chúng ta có hai khoa tên là 'CNTT' và 'DIEN'. Quan hệ lop được phân mảnh ngang dựa vào tenkhoa thành hai mảnh *lop1* và *lop2*

$Lop1 = \sigma_{tenkhoa='CNTT'}(lop)$

$Lop2 = \sigma_{tenkhoa='DIEN'}(lop)$

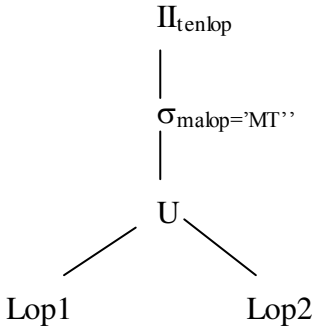
Biểu thức tái lập của quan hệ toàn cục lop là :

$Lop = lop1 \cup lop2$

Biểu thức chuẩn tắc của biểu thúc truy vấn là :

$\Pi_{tenlop}(\sigma_{malop='MT'}(lop1 \cup lop2))$

Thay thế quan hệ toàn cục lop trong cây toán tử bởi biểu thức tái lập ở trên, chúng ta được cây toán tử như sau :



5.2. Tối ưu hóa truy vấn trong cơ sở dữ liệu tập trung

Khi một hệ quản trị dữ liệu (DBMS) nhận một truy vấn viết bằng ngôn ngữ cao cấp, chẳng hạnSQL, DBMS thực hiện các bước sau đây:

5.2.1. Bước 1- Kiểm tra ngữ pháp (syntax Checking)

Trong bước này, DBMS sẽ kiểm tra ngữ pháp của truy vấn an đầu (SQL query). Nếu truy vấn bị sai ngữ pháp thì DBMS sẽ thông báo truy vấn bị sai ngữ pháp và truy vấn này sẽ không được thực hiện. Nếu truy vấn đúng ngữ pháp (syntactically correct SQL query) thì DBMS sẽ tiếp tục thực hiện bước 2.

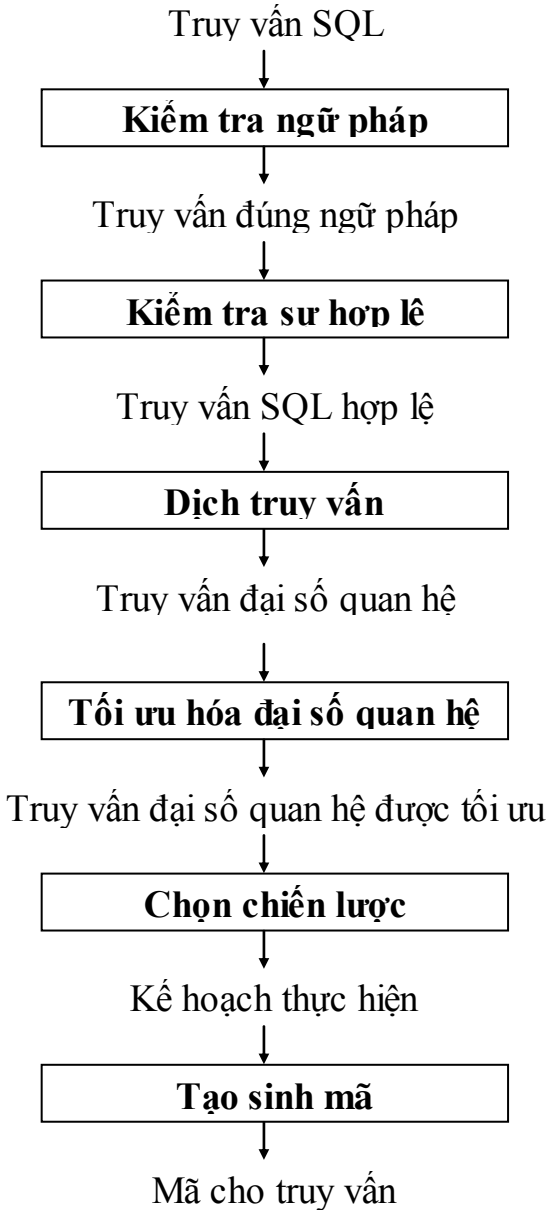
Ví dụ: Xét truy vấn Q1

Q1: SELECT masv,hoten

FORM sinhvien;

Truy vấn này bị sai ngữ pháp (viết sai từ khóa FROM)

Sơ đồ tối ưu hóa truy vấn trong cơ sở dữ liệu tập trung (hình 5.1) bao gồm các bước sau:



Hình 5.1 Sơ đồ tối ưu hóa truy vấn trong cơ sở dữ liệu tập trung

5.2.2. Bước 2- Kiểm tra sự hợp lệ (Validation)

Trong bước này, DBMS sẽ thực hiện các công việc:

- Kiểm tra sự tồn tại của các đối tượng dữ liệu (các cột, các biến, các bảng, …)

của truy vấn trong cơ sở dữ liệu.

- Kiểm tra sự hợp lệ về kiểu dữ liệu của các đối tượng dữ liệu (các cột, các biến, vv…) trong truy vấn.

Ví dụ : Xét truy vấn Q2

```
Q2:  SELECT masv, hoten
     FROM sinh_vien ;
```

Truy vấn này có bảng sinh_vien không tồn tại trong cơ sở dữ liệu.

Ví dụ: Xét truy vấn Q3

```
Q3:  SELECT masv, hoten
     FROM sinhvien
     WHERE masv='123';
```

Truy vấn này không hợp lệ vì có cột masv (thuộc kiểu dữ liệu number) so sánh với một hằng chuỗi '123'trong mệnh đề WHERE.

Nếu truy vấn chứa các đối tượng dữ liệu không tồn tại hoặc truy vấn cứa các đối tượng dữ liệu không phù hợp kiểu dữ liệu với nhau thì DBMS sẽ thông báo các đối tượng dữ liệu nào không tồn tại hoặc các đối tượng dữ liệu nào không phù hợp kiểu dữ liệu và truy vấn này sẽ không được thực hiện. Nếu các đối tượng dữ liệu này đều tồn tại trong cơ sở dữ liệu (truy vấn hợpp lệ – valid SQL query) thì DBMS sẽ tiếp tục thực hiện bước 3.

5.2.3. Bước 3 – Dịch truy vấn (Translation)

Trong bước này, DBMS sẽ biến đổi truy vấn hợp lệ này thành một dạng biểu diễn bên trong hệ thống ở mức thấp hơn mà DBMS có thể sử dụng được. Một trong các dạng biểu diễn bên trong này là việc sử dụng đại số quan hệ bởi vì các phép toán đại số quan hệ

được biến đổi dễ dàng thành các tác vụ của hệ thống : truy vấn ban đầu được biến đổi thành một biểu thức đại số quan hệ hay còn gọi là truy vấn đại số quan hệ (relational algebra query)

Ví dụ : Xét truy vấn Q4 sau đây cho biết các mã môn học mà các sinh viên thuộc lớp có mã 'MT' học.

Q4 : SELECT DISTINCT mamh

FROM sinhvien,hoc

WHERE sinhvien.masv=hoc.masv AND malop='MT'

Truy vấn này sẽ được biến đổi thành biểu thức đại số quan hệ như sau :

$$\Pi_{mamh}(\sigma_{malop='MT'}(sinhvien \blacktriangleright\blacktriangleleft_{masv=masv} hoc))$$

5.2.4. Bước 4- Tối ưu hóa biểu thức đại số quan hệ (relational Algebra Optimization)

Trong bước này DBMS sử dụng các phép biến đổi tương đương của đại số quan hệ để biến đổi biểu thức đại số quan hệ có được ở bước 3 thành một biểu thức đại số quan hệ tương đương (theo nghĩa chúng có cùng một kết quả) nhưng biểu thức sau sẽ hiệu quả hơn: loại bỏ các phép toán không cần thiết và giảm vùng nhớ trung gian. Cuối bước này, DBMS tạo ra một truy vấn đại số quan hệ đã được tối ưu hoá (optimized relational algebra query).

Ví dụ: Biểu thức quan hệ của truy vấn Q4 ở cuối bước 3 có thê' được biến đổi thành biểu thức đại số quan hệ tương đương tốt hơn như sau:

$$\Pi_{mamh}(\Pi_{masv}(\sigma_{malop='MT'}(sinhvien)) \blacktriangleright\blacktriangleleft_{masv=masv} \Pi_{masv,mamh}(hoc))$$

5.2.5. Bước 5- Chọn lựa chiến lược truy xuất (strategy selection)

Trong bước này, DBMS sử dụng các thông số về kích thước của các bảng, các chỉ mục vv… để xác định cách xử lý truy vấn. DBMS sẽ đánh giá chi phí của các kế hoạch thực hiện khác nhau có thể có để từ đó chọn ra một kế hoạch thực hiện (execution plan) cụ thể sao cho tốn ít chi phí nhất (thời gian xử lý và vùng nhớ trung gian). Các thông số dùng để đánh giá chi phí của kế hoạch thực hiện gồm: số lần và loại truy xuất đĩa, kích thức của vùng nhớ chính và vùng nhớ ngoài, và thời gian thực hiện của các tác vụ để tạo ra kết quả của truy vấn. Cối bước này, DBMS tạo ra một kế hoạch thực hiện cho truy vấn.

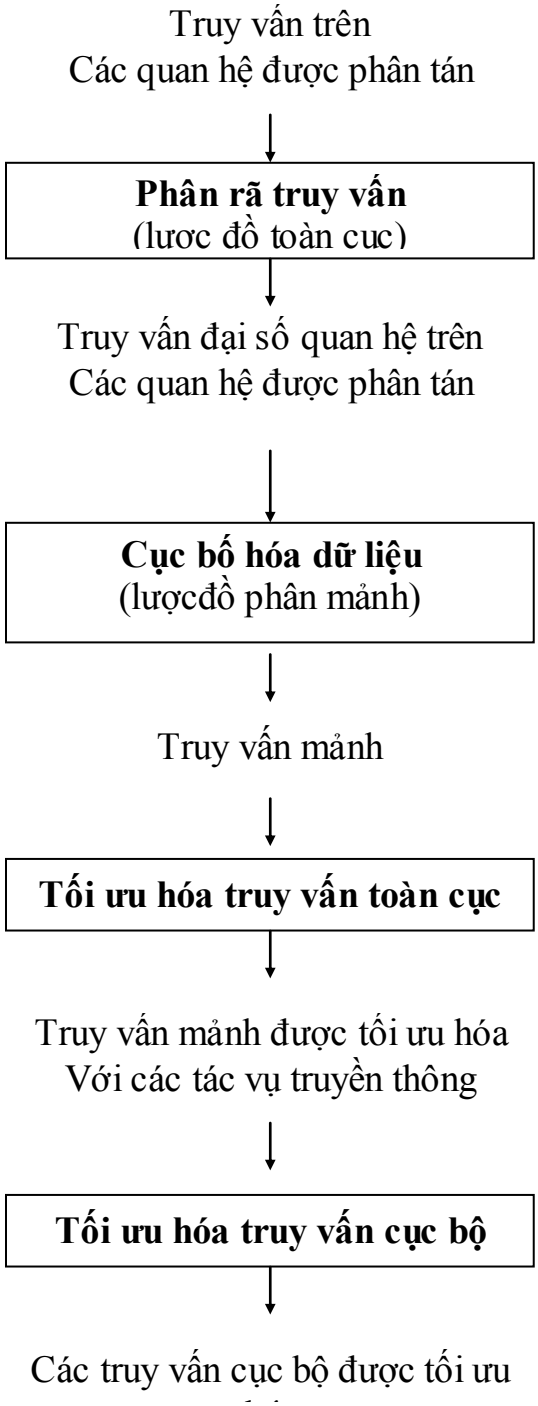
5.2.6. Bước 6- Tạo sinh mã (code Generation)

Trong bước này, kế hoạch thực hiện của truy vấn có được ở cối bước 5 sẽ được mãhoá và được thực hiện.

5.3. Tối ưu hóa truy vấn trong cơ sở dữ liệu phân tán

Tối ưu hoá truy vấn trong cơ sở dữ liệu phân tán bao gồm một số bước đầu của tối ưuhóa truy vấn trong cơ sở dữ liệu tập trung và một số bước tối ưu hóa có liên quan đ61n sự phân tán dữ liệu.

Sơ đồ tối ưu hóa truy vấn trong cơ sở dữ liệu phân tán (hình 5.3) bao gồm các bước sau:



Hình 5.3 Sơ đồ tối ưu hóa truy vấn trong cơ sở dữ liệu phân tán

5.3.1. Bước 1- Phân rã truy vấn (Query Decomposition)

Bước này còn được gọi là bước **Tối ưu hóa truy vấn trên lược đồ toàn cục.** Bước này giống với các bướoc 1, 2, 3 và 4 của tối ưu hóa truy vấn trong cơ sở dữ liệu tập trung, nhằm để biến đổi một truy vấn viết bằng ngôn ngữ cấp cao, chẳng hạn SQL, thành một biểu thức đại số quan hệ tương đương (theo nghĩa chúng cho ra cùng một kết quả) và hiệu quả (theo nghĩa loại bỏ các phép toán đại số quan hệ không cần thiết, giảm vùng nhớ trung gian). Bước này chưa đề cập đến sự phân tán dữ liệu.

Tối ưu hóa truy vấn trên lược đồ toàn cục bao gồm 4 bước sau:

5.3.1.1. Bước 1.1- Phân tích truy vấn

Trong bước này, DBMS kiểm tra ngữ pháp của truy vấn, kiểm tra sự tồn tại của các đối tượng dữ liệu (tên cột, tên bảng, vv…) của truy vấn trong cơ sở dữ liệu, phát hiện các phép toán trong truy vấn bị sai về kiểu dữ liệu, điều kiện của mệnh đề WHERE có hể bị sai về ngữ nghĩa.

Phân tích điều kiện của mệnh đề WHERE để phát hiện truy vấn bị sai. Có hai loại sai:

- Sai về kiểu dữ liệu (type incorrect)
- Sai về ngữ nghĩa (semantically incorrect)

Truy vấn bị sai về kiểu dữ liệu

Một truy vấn bị sai về kiểu dữ liệu nếu các thuộc tính của nó hoặc các tên quan hệ không được định nghĩa trong lược đồ qoàn cục, hoặc nếu các phép toán được áp dụng cho các thộc tính bị sai về kiểu dữ liệu.

Để giải quyết cho vấn đề này, trong lược đồ tòan cục chúng ta phải mô tả kiểu dữ liệu của các thuộc tính của các quan hệ.

Ví dụ: Xét truy vấn Q5

Q5: SELECT mssv, hoten

FROM sinhvien

WHERE masv='123';

Truy vấn này có hai lỗi sai:

(1) mssv không tồn tại trong quan hệ sinhvien, và

(2) masv thuộc kiểu number không thể so sánh với hằng chuỗi '123'.

Truy vấn bị sai về ngữ nghĩa

Một truy vấn bị sai về ngữ nghĩa nếu nó có chứa các thành phần không tham gia vào quá trình tạo ra kết quả của truy vấn.

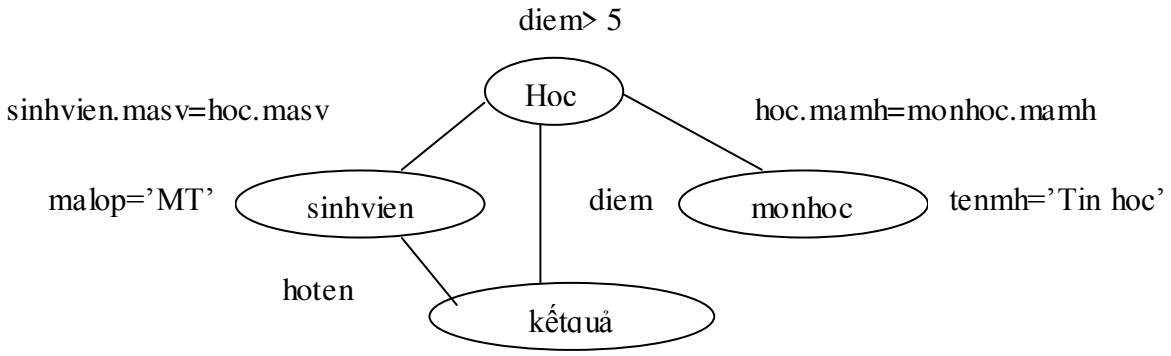
Để phát hiện một truy vấn bị sai về ngữ nghĩa, chúng ta dùng một 9ồ thị truy vấn (query graph) hoặc đồ thị kết nối quan hệ (relation connection graph) cho các truy vấn có chứa các phép chọn, phép chiếu và phép kết. Trong một đồ thị truy vấn, một nút biểu diễn cho một quan hệ kết quả (result relation) và các nút khác biểu diễn cho các quan hệ toán hạng (operand relation). Một cạnh giữa hai nút quan hệ toán hạng biểu diễn cho một phép kết, một cạnh giữa một nút quan hệ toán hạng với một nút quan hệ kết quả biểu diễn ho một phép chiếu. Một nút quan hệ toán hạng có thể chứa một điều kiện chọn. Một đồ thị con quan trọng của đồ thị này là đồ thị kết quả (join graph) được dùng trong bước tối ưu hóa truy vấn.

Ví dụ: Xét truy vấn Q6 liệt kê họ tên sinh viên và điểm của môn học 'Tin học' của lớp mã 'MT' với điề kiện đạt điểm trên 5.

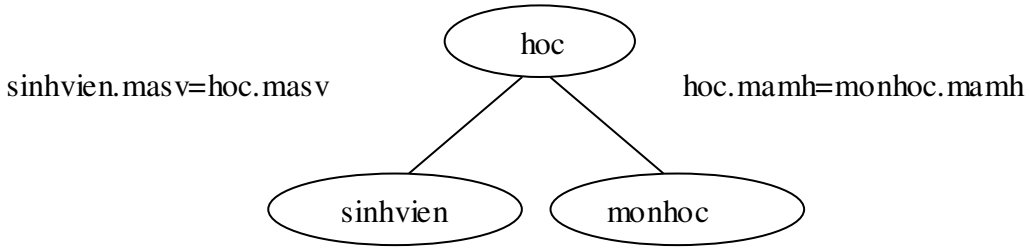
Q6: SELECT hoten, diem

```
FROM sinhvien, hoc, monhoc
WHERE sinhvien.masv=hoc.masv
    AND hoc.mamh=monhoc.mamh
    AND malop='MT'
    AND diem > 5
    AND tenmh = 'Tin hoc';
```

Đồ thị truy vấn của truy vấn này như sau:



Và đồ thị kết nối tương ứng là:



Một truy vấn bị sai về ngữ nghĩa nếu đồ thị truy vấn của nó là không liên thông. Đồ thị không liên thông là một đồ thị bao gồm nhiều thành phần liên thông, mỗi thành phần liên thông là một đồ thị con riêng biệt, hai thành phần liên thông không dược nối với nhau thông qua các cạnh. Trong trường hợp này, một truy vấn được xem là đúng đắn bằng cách chỉ giữ lại thành phần có liên quan đến quan hệ kết quả và loại bỏ các thành phần còn lại.

Ví dụ: Xét truy vấn Q7

Q7: SELECT hoten, diem

FROM sinhvien, hoc, monhoc

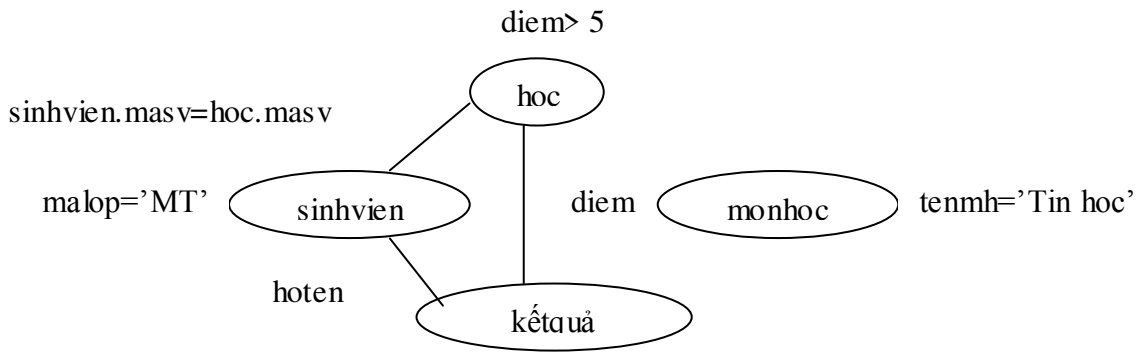
WHERE sinhvien.masv=hoc.masv

AND malop='MT'

AND diem > 5

AND tenmh = 'Tin hoc';

Đồ thị truy vấn của truy vấn này như sau:



Đồ thị truy vấn của truy vấn này là không liên thông, nên truy vấn bị sai về ngữ nghĩa. Có ba giải pháp cho vấn đề này là:

(1) Hủy bỏ truy vấn này.

(2) Hủy bỏ các bảng không cần thiết trong mệnh đề From và cácđiều kiện có liên quan đến các bảng này trong mệnh đề WHERE.

Giả sử truy xuất đến monhoc là không cần thiết, ta hủy bỏ bảng monhoc trong mệnh đề From và điều kiện tenmh ='Tin hoc' trong mệnh đề WHERE. Ta có truy vấn Q8 như sau:

Q8: SELECT hoten, diem

FROM sinhvien, hoc

WHERE sinhvien.masv =hoc.masv

AND malop =’MT’

AND diem > 5;

(3) Bổ sung điều kiện kết sao cho đồ thị truy vấn được liên thông. Một đồ thị truy vấn có thể không bị sai ngữ nghĩa nếu đồ thị này là một dồ thị đơn (có nhiều nhất một cạnh nối giữa hai đỉnh), liên thông và số cạnh bằng số đỉnh trừ 1.

Bổ sung điều kiện kết hoc.mamh = monhoc.mamh vào trong mệnh đề WHERE.

Ta có truy vấn Q9:

Q9: SELECT hoten, diem

FROM sinhvien, hoc

WHERE sinhvien.masv = hoc.masv

AND hoc.mamh =monhoc.mamh

AND malop =’MT’

AND diem > 5

AND tenmh =’Tin hoc’;

5.3.1.2. Bước 1.2- Chuẩn hóa điều kiện của mệnh đề WHERE

Điều kiện ghi trong mệnh đề WHERE là một biểu thức luận lý có thể bao gồm các phép toán luận lý (not, and, or) được viết dưới một dạng bất ký. Ký hiệu các phép tóa luận lý: not (-), and (^), or (v). Bước này nhằm mục đích chuẩn hóa điều kiện của mệnh đề Where về một trong hai dạng chuẩn:

- Dạng chuẩn giao (conjunctive normal form)

$$(P_{11} \vee P_{12} \vee \ldots \vee P_{1n}) \wedge \ldots \wedge (P_{m1} \vee P_{m2} \vee \ldots \vee P_{mn})$$

- Dạng chuẩn hợp (disjunctive normal form)

$$(P_{11} \wedge P_{12} \wedge \ldots \wedge P_{1n}) \vee \ldots \vee (P_{m1} \wedge P_{m2} \wedge \ldots \wedge P_{mn})$$

trong đó P_{ij} là một biến luận lý (có giá trị là true hoặc false) hoặc là một vị từ đơn giản (simple predicate) có dạng: a **R** b

với a,b là các biểu thức số học và **R** là một trong những phép toán so sánh:

=	bằng
< > hoặc !=	không bằng
<	nhỏ hơn
<=	nhỏ hơn hoặc bằng
>	lớn hơn
>=	lớn hơn hoặc bằng

Để biến đổi điều kiện của mệnh đề WHERE về một tronf hai dạng chuẩn trên, chúng ta sử dụng các phép biến đổi tương đương của các phép toán luận lý. Ký hiệu ≡ là sự tương đương.

Các phép biến đổi tương đương:

(1) $P_1 \wedge P_2 \equiv P_2 \wedge P_1$

(2) $P_1 \vee P_2 \equiv P_2 \vee P_1$

(3) $P_1 \wedge (P_2 \wedge P_3) \equiv (P_1 \wedge P_2) \wedge P_3$

(4) $P_1 \vee (P_2 \vee P_3) \equiv (P_1 \vee P_2) \vee P_3$

(5) $P_1 \wedge (P_2 \vee P_3) \equiv (P_1 \wedge P_2) \vee (P_1 \wedge P_3)$

(6) $P_1 \vee (P_2 \wedge P_3) \equiv (P_1 \vee P_2) \wedge (P_1 \vee P_3)$

(7) $\neg(P_1 \wedge P_2) \equiv \neg P_1 \vee \neg P_2$

(8) $\neg(P_1 \vee P_2) \equiv \neg P_1 \wedge \neg P_2$

(9) $\neg(\neg P) \equiv P$

Ví dụ: Xét truy vấn Q10

```
Q10: SELECT malop
     FROM sinhvien
     WHERE (NOT (malop='MT1')
          AND (malop='MT1' OR malop='MT2')
          AND NOT (malop='MT2'))
          OR hoten='Nam';
```

Điều kiện q của mệnh đề WHERE là:

```
(NOT (malop='MT1') AND (malop='MT1' OR malop='MT2')
AND NOT (malop='MT2')) OR hoten='Nam'
```

Ký hiệu:

P_1	là malop='MT1'
P_2	là malop='MT2'
P_3	là hoten='Nam'

Điều kiện q sẽ là:

$(\neg P_1 \wedge (P_1 \vee P_2) \wedge \neg P_2) \vee P_3$

Bằng cách áp dụng các phép biến đổi (3), (5) để đưa điều kiện q về dạng chuẩn hợp:

$(\neg P_1 \wedge P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2 \wedge \neg P_2) \vee P_3$

5.3.1.3. Bước 1.3- Đơn giản hoá điều kiện của mệnh đề WHERE

Bước này sử dụng các phép biến đổi tương đương của các phép toán luận lý (not, and, or) để rút gọn điều kiện của mệnh đề WHERE.

Các phép biến đổi tương đương gồm có :

(10) $P \wedge P \equiv P$

(11) $P \vee P \equiv P$

(12) $P \wedge true \equiv P$

(13) $P \vee false \equiv P$

(14) $P \wedge false \equiv false$

(15) $P \vee true \equiv true$

(16) $P \wedge \neg P \equiv false$

(17) $P \vee \neg P \equiv true$

(18) $P_1 \wedge (P_1 \vee P_2) \equiv P_1$

(19) $P_1 \vee (P_1 \wedge P_2) \equiv P_1$

Vídụ: Xét truy vấn Q10 ở trên, điều kiện q ở dạng chuẩn hợp là:

$(\neg P_1 \wedge P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2 \wedge \neg P_2) \vee P_3$

Bằng cách áp dụng phép biến đổi (16), chúng ta được:

$(false \vee \neg P_2) \vee (\neg P_1 \wedge false) \vee P_3$

Ap dụng phép biến đổi (14), cúng ta được:

$False \vee false \vee P_3$

Ap dụng phép biến đổi (15), chúng ta được diều kiện q cuối cùng là P_3, tức là hoten='Nam'. Vậy truy vấn Q10 trở thành truy vấn Q11 như sau:

```
Q11:  SELECT malop
      FROM sinhvien
      WHERE hoten='Nam';
```

5.3.1.4. Bước 1.4- Biến đổi truy vấn thành một biểu thức đại số quan hệ hiệu quả

Bước này sử dụng các phép biến đổi tương đương của các phép toán đại số quan hệ nhằm để loại bỏ các phép toán đại số quan hệ không cần thiết và giảm vùng nhớ trung gian được sử dụng trong quá trình thực hiện các phép toán đại số quan hệ cần thiết cho truy vấn.

Bước này bao gồm hai bước sau đây:

Bước 1.4.1 – Biến đổi truy vấn thành một biểu thức đại số quan hệ, biểu diễn biểu thức đại số quan hệ này bằng một cây toán tử.

Bước 1.4.2 – Đơn giản hóa cây toán tử để có được một biểu thức đại số quan hệ hiệu quả.

Bước 1.4.1. Biểu diễn truy vấn bằng cây toán tử

Quá trình biến đổi một truy vấn được viết bằng lệnh SELECT thành một cây toán tử bao gồm các bước sau:

(1) Các nút lá được tạo lập từ các quan hệ ghi trong mệnh đề From

(2) Nút gốc được tạo lập bằng phép chiếu trên các thuộc tính ghi trong mệnh đề SELECT.

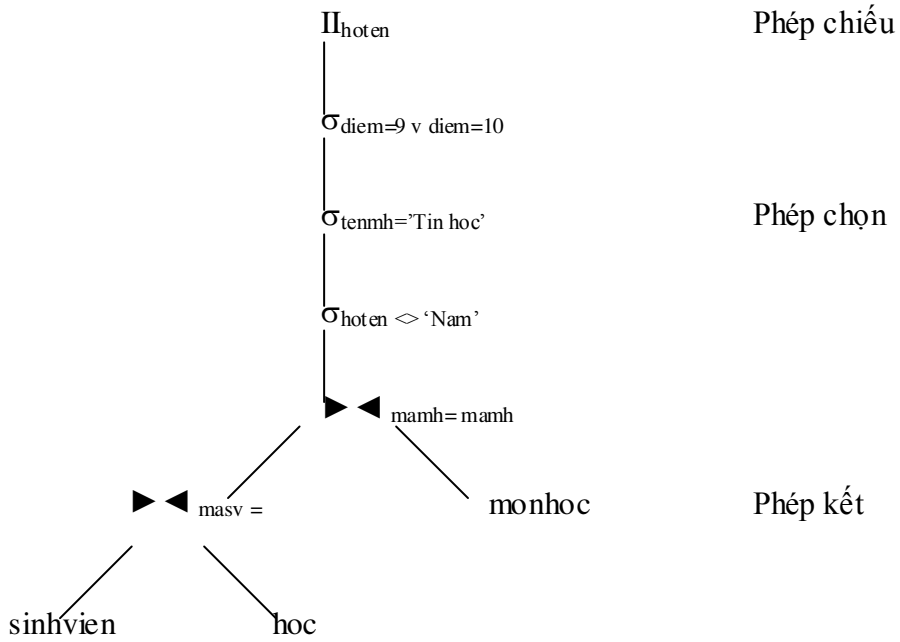
(3) Điều kiện ghi trong mệnh đề WHERE được biến đổi thành một chuỗi thích hợp các phép toán đại số quan hệ (phép chọn, phép kết, phép hợp…) đi từ các nút lá đến nút gốc. Chuỗi các phép toán này có thể được cho trực tiếp bởi thứ tự của các vị từ đơn giản và các phép toán luận lý.

Một cây toán tử tương ứng với một biểu thức đại số quan hệ.

Ví dụ: Xét truy vấn Q12 cho biết họ tên của các sinh viên không phải là ‘Nam’ học môn học ‘Tin hoc’ đạt điểm 9 hoặc 10.

```
Q12:  SELECT hoten
      FROM sinhvien, hoc, monhoc
      WHERE sinhvien.masv= hoc.masv
            AND hoc.mamh= monhoc.mamh
            AND hoten< > ‘Nam’
            AND tenmh= ‘Tin hoc’
            AND (diem= 9 OR diem = 10);
```

Truy vấn này có thể được biểu diễn thành một cây toán tử, các vị từ đơn giản được biến đổi theo thứ tự xuất hiện tương ứng với các phép kết rồi đến các phép chọn.



Biểu thức đại sô quan hệ tương ứng là:

Π_{hoten} (σ (diem=9 v diem=10) ^ tenmh= 'Tin hoc' ^ hoten <> 'Nam'

((sinhvien ►◄ masv=masv hoc) ►◄ mamh=mamh monhoc))

Bước 1.4.2. Đơn giản hóa cây toán tử

Đơn giản hoá cây toán tử nhằm mục đích để đạt hiệu quả (loại bỏ các phép toán dư thừa trên các quan hệ, giảm vùng nhớ trung gian, giảm thời gian xử lý truy vấn) bằng cách sử dụng các phép biến đổi tương đương của các phép toán đại số quan hệ.

Trong bước đơn giản hoá cây toán tử, một điều quan trọng trong việc áp dụng các phép biến đổi tương đương cho một biểu thức truy vấn là việc phát hiện các biểu thức con chung (common subexpression) có trong biểu thức truy vấn, nhgĩa là các biểu thức con xuất hiện nhiều lần trong biểu thức truy vấn. Điều này có ý nghĩa là tiết kiện thời gian thực hiện truy vấn vì các biểu thức con này chỉ được định trị duy nhất một lần. Một phương pháp để nhận biết chúng là ở chỗ việc biến đổi cây toán tử tương ứng thành một

đồ thị toán tử bằng cách trước tiên gộp các nút lá giống nhau của cây (nghĩa là các quan hệ giống nhau), và sau đó gộp các nút trung gian khác của cây tương ứng với cùng các phép toán và có cùng các toán hạng.

Khi các biểu thức con đã được xác định, chúng ta có sử dụng các phép biến đổi tương đương sau đây để đơn giản hóa một cây toán tử:

(1) $R \blacktriangleright\blacktriangleleft R \equiv R$

(2) $R \cup R \equiv R$

(3) $R - R \equiv \varnothing$

(4) $R \blacktriangleright\blacktriangleleft \sigma_F (R) \equiv \sigma_F R$

(5) $R \cup \sigma_F (R) \equiv R$

(6) $R - \sigma_F (R) \equiv \sigma_{\neg F} (R)$

(7) $\sigma_{F1} (R) \blacktriangleright\blacktriangleleft \sigma_{F2} (R) \equiv \sigma_{F1 \wedge F2} (R)$

(8) $\sigma_{F1} (R) \cup \sigma_{F2} (R) \equiv \sigma_{F1 \vee F2} (R)$

(9) $\sigma_{F1} (R) - \sigma_{F2} (R) \equiv \sigma_{F1 \wedge \neg F2} (R)$

(10) $R \cap R \equiv R$

(11) $R \cap \sigma_F (R) \equiv \sigma_F R$

(12) $\sigma_{F1} (R) \cap \sigma_{F2} (R) \equiv \sigma_{F1 \wedge F2} (R)$

(13) $\sigma_F (R) - R \equiv \varnothing$

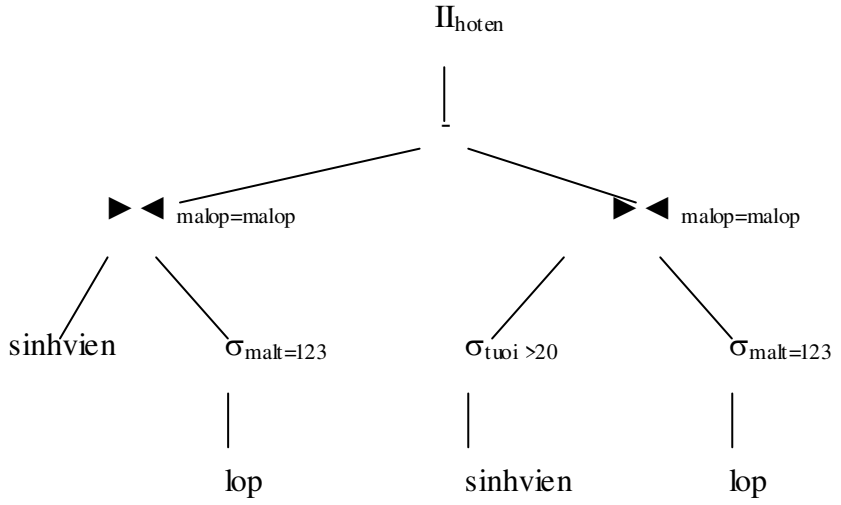
Ý nghĩa của các phép biến đổi này là loại bỏ các phép toán dư thừa.

Ví dụ: Xét truy vấn Q13 cho biết các họ tên của các sinh viên thuộc lớp có mã lớp trưởng là 123 và các sinh viên này có tuổi không lớn hơn 20 tuổi. Một biểu thức cho truy vấn này là:

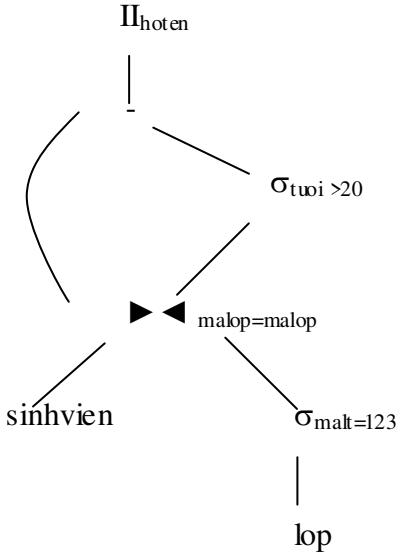
$$\Pi_{hoten}((sinhvien \blacktriangleright\blacktriangleleft_{malop=malop} \sigma_{malt=123}(lop)) -$$

$$(\sigma_{tuoi>20}(sinhvien) \blacktriangleright\blacktriangleleft_{malop=malop} \sigma_{malt=123}(lop))$$

Cây toán tử tương ứng :



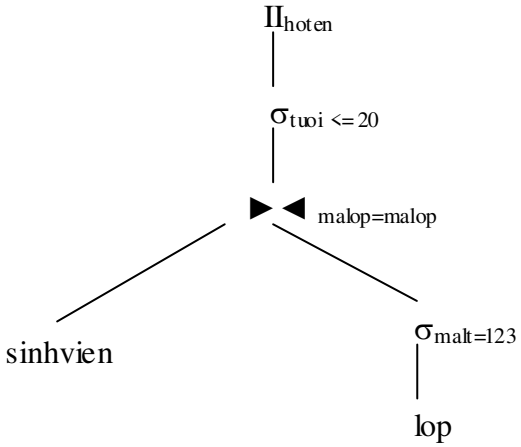
Để phát hiện ra biểu thức con vhung, chúng ta bắt đầu bằng cách gộp các nút lá tương ứng với các quan hệ sinhvien và lop. Sau đó chúng ta đặt thừa số là phép chọ trên tuoi đối với phép kết (trong cách làm này, chúng ta di chuyển phép chọn lên phía trên). Bây giờ chúng ta có thể trộn các nút tương ứng với phép chọn trên malt và cuối cùng các nút tương ứng với phép kết, chúng ta được cây toán tử sau:



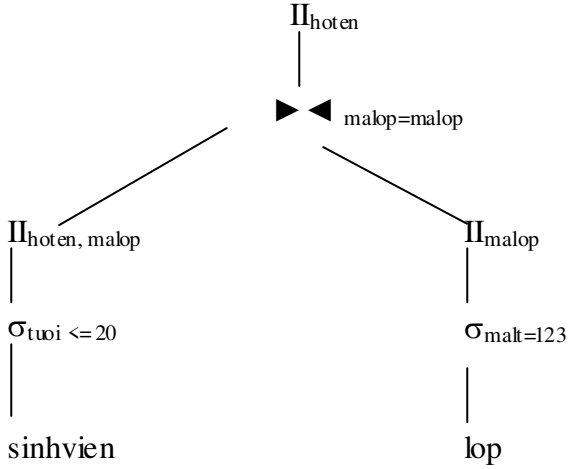
Áp dụng phép biến đổi tương đương (6) với R là biểu thức :

$$sinhvien \blacktriangleright\blacktriangleleft_{malop=malop} \sigma_{malt=123} lop$$

chúng ta được cây toán tử sau:



Sau đó áp dụng tính phân phối của phép chiếu và phép chọn đối với phép kết, ta được cây toán tử :



Và biểu thức đại số quan hệ sau khi đã đơn giản hoá là :

$\Pi_{hoten}(\Pi_{hoten,malop}(\sigma_{tuoi <= 20}(sinhvien)) \blacktriangleright\blacktriangleleft_{malop=malop} \Pi_{malop}(\sigma_{malt=123}(lop)))$

Đơn giản hoá một biểu thức đại số quan hệ được thực hiện dựa trên các tiêu chuẩn sau đây :

Tiêu chuẩn 1. Dùng tính idempotence của phép chọn và phép chiếu để tạo ra các phép chọn và phép chiếu thích hợp cho mỗi quan hệ toán hạng.

Tiêu chuẩn 2. Thực hiện các phép chọn và các phép chiếu càng sớm càng tốt, tức là đẩy các phép chọn và các phép chiếu xuống phía dưới cây càng xa càng tốt.

Tiêu chuẩn 3. Khi các phép chọn được thực hiện sau một phép tích thì kết hợp các phép toán này để tạo thành một phép kết.

Tiêu chuẩn 4. Kết hợp chuỗi các phép toán một ngôi liên tiếp nhau áp dụng cho một quan hệ toán hạng. Một chuỗi các phép chọn liên tiếp nhau (hoặc một chuỗi các

phép liên kết liên tiếp nhau) có thể được kết hợp thành một phép chọn (hoặc một phép kết).

Tiêu chuẩn 5. Khi phát hiện các biểu thức con chung trong biểu thức truy vấn, áp dụng các phép biến đổi tương đương để đơn giản hoá biểu thức truy vấn.

5.3.1.5. Một giải thuật tối ưu hóa một biểu thức đại số quan hệ trên lược đồ toàn cục

Vào: Một biểu thức đại số quan hệ trên lược đồ toàn cục

Ra: Một biểu thức đại số quan hệ đã được tối ưu hóa

Giải thuật tối ưu hoá một biểu thức đại số quan hệ trên lược đồ toàn cục bao gồm các bước sau đây:

Bước 1. Phát hiện các biểu thức con chung có trong cây toán tử, biến đổi cây toán tử dựa trên biểu thức con chung

Bước 2. Thực hiện phép chọn càng sớm càng tốt. Sử dụng tính idempotence của phép chọn, tính giao hoán của phép chọn với phép chiếu, và tính phân phối của phép chọn đối với phép hợp, phép giao, phép hiệu, phép kết và phép tích để di chuyển phép chọn càng xuống phía dưới cây càng tốt.

Sử dụng các phép biến đổi tương đương:

$\sigma_{F1}(\sigma_{F2}(R)) \equiv \sigma_{F2}(\sigma_{F1}(R))$

$\sigma_{F1}(\sigma_{F2}(R)) \equiv \sigma_{F1 \wedge F2}(R)$

$\Pi_X(\sigma_F(R)) \leftrightarrow \sigma_F(\Pi_X(R))$

$(\rightarrow$ nếu $Attr(F) \subseteq X)$

$\Pi_X(\sigma_F(R)) \equiv \Pi_X(\sigma_F(\Pi_{X \cup Attr(F)}(R)))$

$\sigma_F(R \cup S) \equiv \sigma_F(R) \cup \sigma_F(S)$

$\sigma_F(R \cap S) \equiv \sigma_F(R) \cap \sigma_F(S)$

$\sigma_{F1 \wedge F2}(R \cap S) \leftrightarrow \sigma_{F1}(R) \cap \sigma_{F2}(S)$

($\rightarrow$ nếu $Attr(F1) \subseteq Attr(R)$ và $Attr(F2) \subseteq Attr(S)$)

$\sigma_F(R - S) \equiv \sigma_F(R) - \sigma_F(S)$

$\sigma_F(R \blacktriangleright\blacktriangleleft_{F1} S) \leftrightarrow \sigma_F(R) \blacktriangleright\blacktriangleleft_{F1} S$

($\rightarrow$ nếu $Attr(F) \subseteq Attr(R)$)

$\sigma_{F1 \wedge F2}(R \blacktriangleright\blacktriangleleft_{F3} S) \leftrightarrow \sigma_{F1}(R) \blacktriangleright\blacktriangleleft_{F3} \sigma_{F2}(S)$

($\rightarrow$ nếu $Attr(F1) \subseteq Attr(R)$ và $Attr(F2) \subseteq Attr(S)$)

$\sigma_F(R \blacktriangleright\blacktriangleleft_{F3} S) \leftrightarrow \sigma_{F2}(\sigma_{F1}(R) \blacktriangleright\blacktriangleleft_{F3} S)$

($\rightarrow$ nếu $F=F1 \wedge F2$ và $Attr(F1) \subseteq Attr(R)$ và $Attr(F2) \subseteq Attr(R) \cup Attr(S)$)

$\sigma_F(R \times S) \leftrightarrow \sigma_F(R) \times S$

($\rightarrow$ nếu $Attr(F) \subseteq Attr(R)$)

$\sigma_{F1 \wedge F2}(R \times S) \leftrightarrow \sigma_{F1}(R) \times \sigma_{F2}(S)$

($\rightarrow$ nếu $Attr(F1) \subseteq Attr(R)$ và $Attr(F2) \subseteq Attr(S)$)

$\sigma_F(R \times S) \leftrightarrow \sigma_{F2}(\sigma_{F1}(R) \times S)$

($\rightarrow$ nếu F=F1 ^ F2 và Attr(F1) $\subseteq$ Attr(R) và Attr(F2) $\subseteq$ Attr(R) $\cup$ Attr(S))

Bước 3. Thực hiện phép chiếu càng sớm càng tốt. Sử dụng tính idempotence của phép chiếu, tính phân phối của phép chiếu đối với phép hợp, phép kết và phép tích để di chuyển phép chiếu càng xuống phía dưới cây càng tốt. Kiểm tra tất cả các phép chiếu là cần thiết, loại bỏ phép chiếu không cần thiết nếu phép này chiếu trên tất cả các thuộc tính của quan hệ toán hạng.

Sử dụng phép biến đổi:

$\Pi_{X1}(\Pi_{X2}(R)) \equiv \Pi_{X1}(R)$ với $X1 \subseteq X2$

$\Pi_X(R \cup S) \equiv \Pi_X(R) \cup \Pi_X(S)$

$\Pi_X(R \blacktriangleright\blacktriangleleft_F S) \leftrightarrow \Pi_X(R) \blacktriangleright\blacktriangleleft_F (S)$

($\rightarrow$ nếu $Attr(F_R) \subseteq X$ và $X \subseteq Attr(R)$)

$\Pi_{X1 \cup X2}(R \blacktriangleright\blacktriangleleft_F S) \leftrightarrow \Pi_{X1}(R) \blacktriangleright\blacktriangleleft_F \Pi_{X2}(S)$

($\rightarrow$ nếu Attr(F) $\subseteq$ X1 $\cup$ X2 và X1 $\subseteq$ Attr(R) và X2 $\subseteq$ Attr(S))

$\Pi_{X1 \cup X2}(R \times S) \leftrightarrow \Pi_{X1}(R) \times \Pi_{X2}(S)$

($\rightarrow$ nếu X1 $\subseteq$ Attr(R) và X2 $\subseteq$ Attr(S))

Bước 4. Nếu một phép chọn được thực hiện ngay sau mọt phép tích, mà phép chọn bao gồm các thuộc tính của các quan hệ trong phép tích, thì biến đổi phép tích thành phép kết. Nếu phép chọn chỉ bao gồm các thuộc tính của một quan hệ trong phép tích, thì thực hiện phép chọn cho quan hệ này trước khi thực hiện phép tích.

Sử dụng các phép biến đổi:

$\sigma_F (R \times S) \leftrightarrow \sigma_F (R) \times S$

$(\rightarrow$ nếu $Attr(F) \subseteq Attr(R))$

$\sigma_{F1 \wedge F2} (R \times S) \leftrightarrow \sigma_{F1} (R) \times \sigma_{F2} (S)$

$(\rightarrow$ nếu $Attr(F1) \subseteq Attr(R)$ và $Attr(F2) \subseteq Attr(S))$

$\sigma_F (R \times S) \leftrightarrow \sigma_{F2} (\sigma_{F1} (R) \times S)$

$(\rightarrow$ nếu $F=F1 \wedge F2$ và $Attr(F1) \subseteq Attr(R)$ và $Attr(F2) \subseteq Attr(R) \cup Attr(S))$

Bước 5. Nếu có một chuỗi các phép chọn và/ hoặc các pép chiếu, sử dụng tính giao hoán hoặc tính idempotence để kết hợp chúng thành một phép chọn, một phép chiếu hoặc một phép chọn đi trước một phép chiếu và áp dụng chúng cho mỗi bộ của quan hệ toán hạng. Nếu một phép kết hoặc phép tích đi trước một chuỗi các phép chọn hoặc các phép chiếu, thì áp dụng chúng cho mỗi bộ của phép kết hoặc phép chiếu ngay khi tạo ra kết quả.

Bước 6. Sử dụng tính kết hợp của phép giao, phép tích và phép kết để sắp xếp lại các quan hệ trong cây toán tử, sao cho phép toán nào mànó tạo ra kết quả ít nhất sẽ được thực hiện trưc tiên.

Sử dụng các phép biến đổi:

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

$(R \times S) \times T \equiv (R \times T) \times S$

$(R \blacktriangleright\blacktriangleleft_{F1} S) \blacktriangleright\blacktriangleleft_{F2} T \leftrightarrow (R \blacktriangleright\blacktriangleleft_{F2} T) \blacktriangleright\blacktriangleleft_{F1} S$

$(\rightarrow$ nếu $Attr(F2) \subseteq Attr(R) \cup Attr(T))$

($\leftarrow$ nếu Attr(F1) $\subseteq$ Attr(R) $\cup$ Attr(S))

5.3.2. Bước 2 Cục bộ hóa dữ liệu

Bước cục bố hóa dữ liệu (Data Localization) còn được gọi là bước **tối ưu hóa truy vấn trên lược đồ phân mảnh** . Bước này biến đổi truy vấn toàn cục (kết quả của Bước 1) thành các truy vấn mảnh hiệu quả*: loại bỏ các phép toán đại số quan hệ không cần thiết trên các mảnh và giảm vùng nhớ trung gian.*

Tối ưu hóa truy vấn trên lược đồ phân mảnh bao gồm 2 bước sau:

Bước 2.1. Biến đổi biểu thức đại số quan hệ trên lược đồ toàn cục (chứa các quan hệ toàn cục) thành biểu thức đại số quan hệ trên lược đồ phân mảnh (chứacác mảnh của quan hệ toàn cục) bằng cách thay thế các quan hệ toàn cục bởi biểu thức tái lập của chúng.

Bước 2.2. Đơn giản hoá biểu thức đại số qua hệ trên lược đồ phân mảnh để có được một biểu thức hiệu qủa (loại bỏ các phép toán không cần thiết giảm vùng nhớ trung gian) bằng cách sử dụng các phép biến đổi tương đương của đại số quan hệ và các đại số quan hệ được tuyển chọn.

5.3.2.1. Bước 2.1 – Biến đổi biểu thức đại số quan hệ trên lược đồ toàn cục

Bước này sẽ biến đổi biểu thức đại số quan hệ trên lược đồ toàn cục (chứa các quan hệ toàn cụ) hành biểu thức đại số quan hệ trên lược đồ phân mảnh (chứa các mảnh của quan hệ toàn cục) bằng cch thay thế mỗi quan hệ toàn cục trong cây toán tử bởi biểu thức tái lập của nó. Bểu thứctái lập của một quan hệ toàn cục là một biểu thức đại số quan hệ bao gồm các mảnh của quan hệ này mà biểu thức này cho phép tạo lại quan hệ toàn cục này. Biểu thức tái lập cũng được biểu diễn bằng một cây toán tử.

Xét lược đồ quan hệ sinhvien và lop sau đây:

Sinhvien (masv, hoten, tuoi, malop)

Lop (malop, tenlop, malt, tenkhoa)

Giả sử chúng ta có hai khoa tên là 'CNTT' và 'DIEN'. Quan hệ *lop* được phân mảnh ngang dựa vào *tenkhoa* thành hai mảnh *lop1* và *lop2* . Quan hệ *sinhvien* được phân mảnh ngang suy dẫn theo *lop* dựa vào *malop* thành hai mảnh *sinhvien1* và *sinhvien2*. Lược đồ phân mảnh như sau:

Lop1 (malop, tenlop, malt, tenkhoa)

Lop2 (malop, tenlop, malt, tenkhoa)

Sinhvien1 (masv, hoten, tuoi, malop)

Sinhvien2 (masv, hoten, tuoi, malop)

Các biểu thức tái lập của quan hệ *lop* và *sinhvien* là:

Lop = Lop1 U Lop2

Sinhvien= sinhvien1 U sinhvien2

Trong đó:

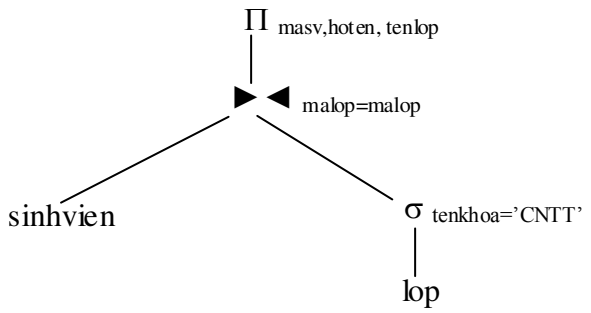
Lop1 = $\sigma_{tenkhoa = 'CNTT'}$ (lop)

Lop1 = $\sigma_{tenkhoa = 'DIEN'}$ (lop)

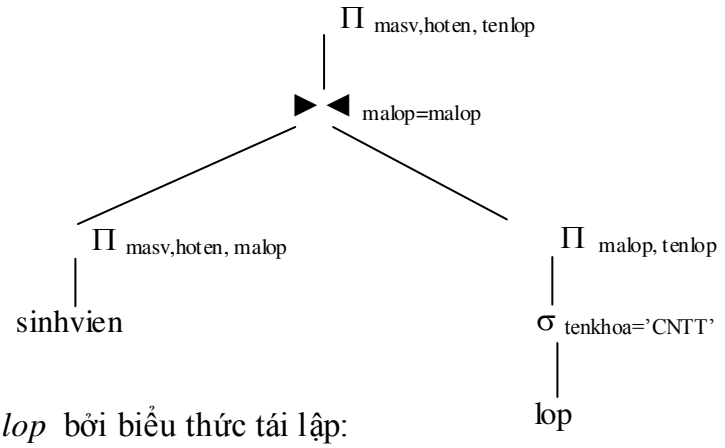
Sinhvien1 = sinhvien ►< (Lop1)

Sinhvien2 = sinhvien ►< (Lop2)

Ví dụ: xét cây toán tử



Áp dụng tính idempotence của phép chiếu, chúng ta được:

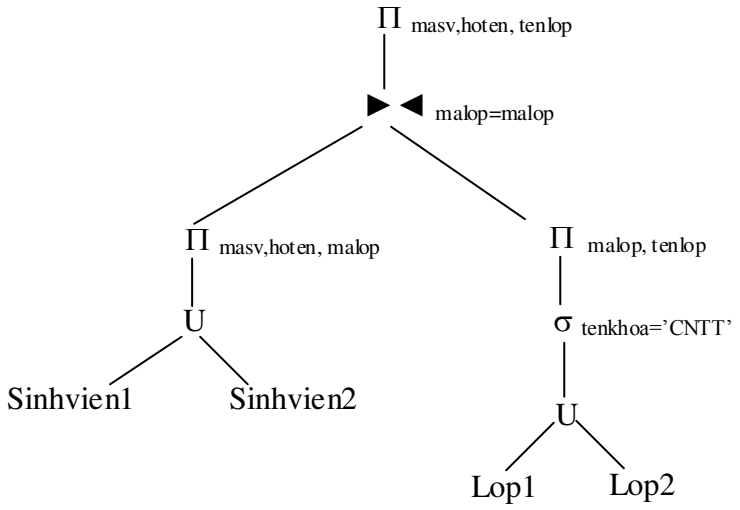


Thay thế *sinhvien* và *lop* bởi biểu thức tái lập:

Sinhvien = sinhvien1 U sinhvien2

Lop = lop1 U lop2

Ta được cây toán tử sau:



5.3.2.2 Bước 2.2– Đơn giản hoá biểu thức đại số quan hệ trên lược đồ phân mảnh

Đơn giản hoá biểu thức đại số quan hệ trên lược đồ phân mảnh để có được một biểu thức hiệu quả (*loại bỏ các phép toán không can thiết, giảm vùng nhớ trung gian)* bằng cách sử dụng các phép biến đổi tương đương của đại số quan hệ và của đại số quan hệ được tuyển chọn.

Các phép biến đổi tương đương (áp dụng cho các quan hệ và các quan hệ được tuyển chọn) gồm có:

(1) $\sigma_F(\varnothing) \equiv \varnothing$

(2) $\Pi_X(\varnothing) \equiv \varnothing$

(3) $R \times \varnothing \equiv \varnothing$

(4) $R \cup \varnothing \equiv \varnothing$

(5) $R \cap \varnothing \equiv \varnothing$

(6) $R - \varnothing \equiv \varnothing$

(7) $\varnothing - R \equiv \varnothing$

(8) $R \blacktriangleright \blacktriangleleft \varnothing \equiv \varnothing$

(9) $R \blacktriangleright < \varnothing \equiv \varnothing$

(10) $\varnothing \blacktriangleright < R \equiv \varnothing$

Đơn giản hoá một biểu thức đại số quan hệ trên lược đồ phân mảnh được thực hiện dựa trên các tiêu chuẩn sau:

Tiêu chuẩn 6: Di chuyển các phép chọn xuống các nút lá của cây, và sau đó áp dụng chúng bằng cách dùng đại số quan hệ được tuyển chọn; thay thế các kết quả chọn lựa bởi quan hệ rỗng nếu điều kiện chọn của kết quả bị mâu thuẫn.

Tiêu chuẩn 7: Để phân phối các phép kết xuất hiện trong một truy vấn toàn cục, các phép hợp (biểu diễn tập hợp của các phân mảnh) phải được di chuyển lên phía trên các phép kết mà chúng ta muốn phân phối để loại bỏ các phép kết không cần thiết.

Tiêu chuẩn 8: Dùng đại số quan hệ được tuyển chọn để định trị điều kiện chọn của các toán hạng của các phép kết; thay thế cây con, bao gồm phép kết và các toán hạng của nó, bằng quan hệ rỗng nếu điều kiện chọn của kết quả của phép két bị mâu thuẫn.

Ví dụ : Xét cây toán tử trên lược đồ phân mảnh trên

Đẩy phép chọn và phép chiếu xuống khỏi phép hợp ta được:

$\Pi_{malop,tenlop}(\sigma_{tenkhoa='CNTT'}(lop1 \cup lop2))$

$= \Pi_{malop,tenlop}(\sigma_{tenkhoa='CNTT'}(lop1)) \cup \Pi_{malop,tenlop}(\sigma_{tenkhoa='CNTT'}(lop2))$

Ta nhận thấy kết quả của phép chọn $\sigma_{tenkhoa='CNTT'}(lop2)$ là rỗng và phép chọn $\sigma_{tenkhoa='CNTT'}(lop1)$ là không cần thiết vì điều kiện chọn của lop1 là tenkhoa= 'CNTT'.

Do đó:

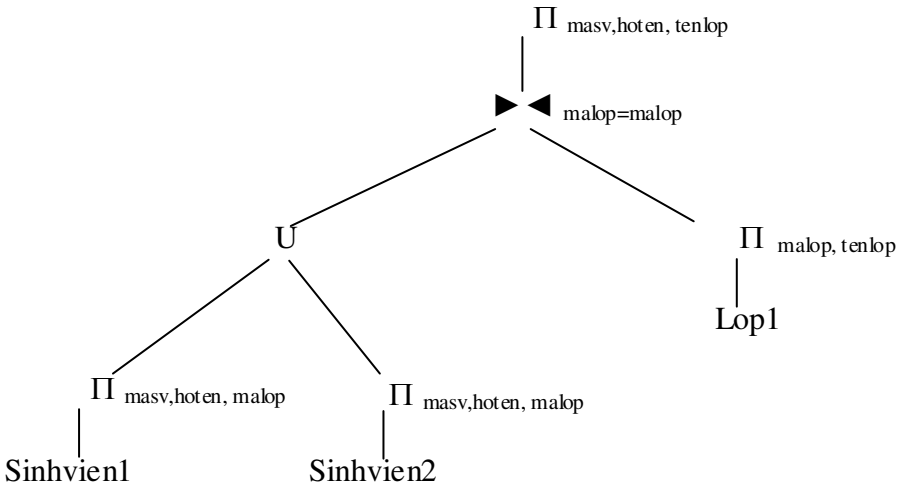
$$\Pi_{malop,tenlop}(\sigma_{tenkhoa='CNTT'}(lop1 \cup lop2)) = \Pi_{malop,tenlop}(lop1)$$

Đẩy phép chiếu xuống khỏi phép hợp trong biểu thức:

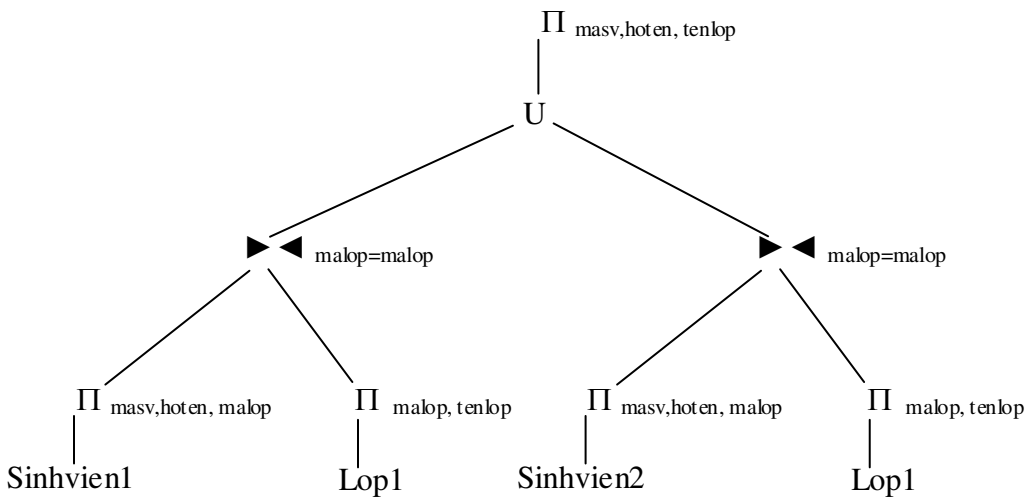
$$\Pi_{masv,hoten,\ malop}(sinhvien1 \cup sinhvien2) =$$

$$\Pi_{masv,hoten,\ malop}(sinhvien1) \cup \Pi_{masv,hoten,\ malop}(sinhvien2)$$

Ta có cây toán tử:

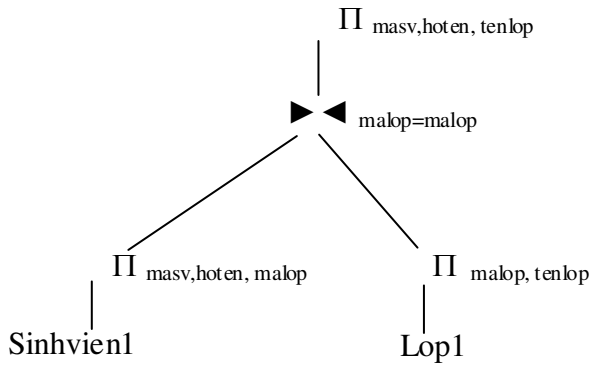


Sau đó phân phối phép kết với với phép hợp ta được:



Tuy nhiên phếp kết giữa sinhviên2 và lop1 là rỗng do điều kiện chọn của phân mảnh lop1 và sinhvien2 mâu thuẫn nhau.

Cuối cùng ta có cây toán tử trên lược đồ phân mảnh như sau:



Đơn giản hoá biểu thức đại số quan hệ trong lược đồ phân mảnh còn dựa vào một hệ suy diễn được gọi là Bộ chứng minh định lý (Theorem Prover).

Ví dụ: Giả sử chúng ta chỉ có hai khoa là 'CNTT' và có tối đa 20 lớp, các lớp có mã lớp từ 1 đến 10 thuộc khoa 'CNTT' và các lớp có mã từ 11 đến 20 thuộc khoa 'DIEN'. Từ đó, chúng ta có các luật suy diễn sau:

Malop > 10 → tenkhoa = 'DIEN'

Malop <= 10 → NOT (Malop >10)

Malop > 10 → NOT (Malop <=10)

tenkhoa = 'CNTT' → Malop <= 10

tenkhoa = 'DIEN' → Malop > 10

tenkhoa = 'CNTT' → not(tenkhoa = 'DIEN')

tenkhoa = 'DIEN' → not(tenkhoa = 'CNTT')

Xét truy vấn Q14 cho biết tên lớp của lớp có mã lớp bằng 1:

```
Q14: Select tenlop
     From lop
     Where malop = 1
```

Trước khi thực hiện truy vấn này, chúng ta có các suy diễn sau đây:

Malop = 1 → malop <=10

Malop <= 10 → tenkhoa= 'CNTT'

Do đó truy vấn này chỉ liên quan đến lop1 vì điều kiện chọn của lop1 là tenkhoa = 'CNTT'. Vì thế biểu thức đại số quan hệ của truy vấn này là:

$\Pi_{tenlop}(\sigma_{malop=1}(lop1))$

5.3.3 Bước 3 Tối ưu hoá truy vấn toàn cục

Bước tối ưu hoá truy vấn toàn cục nhằm để tìm ra một chiến lược thực hiện truy vấn sao cho chiến lược này gần tối ưu (theo nghĩa giảm thời gian thực hiện truy vấn trên dữ liệu được phân tán, giảm vùng nhớ trung gian).

Một chiến lược được đặc trưng bởi *thứ tự thực hiện các phép toán đại số quan hệ và các tác vụ truyền thông cơ bản (gởi/nhận)* dùng để truyền dữ liệu giữa các vị trí. Bằng các hoán đổi thứ tự của các phép toán trong biểu thức truy vấn phân mảnh, ta có thể có được nhiều truy vấn tương đương.

Tối ưu hóa truy vấn toàn cục là tìm ra một thứ tự thực hiện các phép toán trong biểu thưc truy vấn sao cho ít tốn thời gian nhất. Đặc biệt khâu tón kém thời gian trong cơ sử dữ liệu phân tán là khâu truyền dữ liệu do tốc độ và băng thông giới hạn.

Trong trường hợp nhân bản thì còn phải tính xem nhân bản nào được sử dụng nhằm giảm chi phí truyền thông.

Một khía cạnh quan trọng của tối ưu hoá truy vấn là *thứ tự thực hiện các phép kết phân tán*. Nhờ tính giao hoán của các phép kết, chúng ta có thể làm giảm chi phí thực hiện các phép kết này. Một kỹ thuật cơ bản để tối ưu hoá một chuỗi các phép kết phân tán là sử

dụng phép nửa kết nhằm làm giảm chi phí truyền thông giữa các vị trí và tăng tính xử lý cục bộ tại các vị trí.

$$R \blacktriangleright\blacktriangleleft_{A=B} S = S \blacktriangleright\blacktriangleleft_{A=B} (R \blacktriangleright<_{A=B} \Pi_B S)$$

Ví dụ: Giả sư có sự phân tán dữ liệu sau:

- mảnh *sinhvien1* đặt tại vị trí 1 và
- mảnh *lop1* đặt tại vị trí 2

Chúng ta cần thực hiện phép kết phân tán sau:

Sinhvien1 ▶◀lop1

Bằng cách áp dụng phép nửa kết biểu thức trên tương đương với:

Lop1 ▶◀ (sinhvien1 ▶< Π_{malop}(lop1))

Do đó ta có một chiến lược thực hiện cho phép kết phân tán này với các tác vụ truyền thông sau:

1) Thực hiện $T_1 = \Pi_{malop}$(lop1) cục bộ tại vị trí 2.
2) Truyền T_1 từ vị trí 2 qua vị trí 1.
3) Thực hiện T_2 = sinhvien1 ▶< T_1 cục bộ tại vị trí 1.
4) Truyền T_2 từ vị trí 1 qua vị trí 2.
5) Thực hiện T_3 = lop1 ▶< T_2 cục bộ tại vị trí 2.
6) Truyền T_3 từ vị trí 2 qua vị trí của ứng dụng cần thực hiện của phép kết này.

5.3.4 Bước 4 Tối ưu hoá truy vấn cục bộ

Tối ưu hoá truy vấn cục bộ nhằm để thực hiện các truy vấn con được phân tán tại mỗi vị trí, gọi là truy vấn cục bộ có chứa các mảnh, sau đó được tối ưu hoá trên lược đồ cục bộ tại mỗi vị trí. Tối ưu hoá truy vấn cục bộ sử dụng các thuật toán tối ưu hoá truy vấn của cơ sở dữ liệu tập trung.

Chương 6 GIAO TÁC PHÂN TÁN

MỤC TIÊU

Chương này giới thiệu khái niệm giao tác và định nghĩa hình thức của giao tác. Chương này chia làm ba phần:

1. Phần thứ nhất: Khái niệm giao tác, định nghĩa hình thức của giao tác.

2. Phần thứ hai: Các tính chất của giao tác

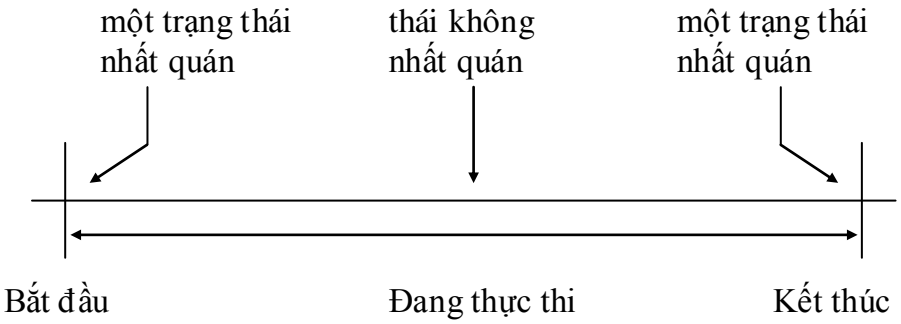
3. Phần thứ ba: Phân loại giao tác

Cho đến lúc này, đơn vị truy xuất cơ bản là câu truy vấn. Trong chương 5, chúng ta đã thảo luận về cách xử lý và tối ưu hoá các truy vấn. Tuy nhiên chúng ta chưa bao giờ xét đến các tình huống xảy ra, chẳng hạn khi hai câu truy vấn cùng cập nhật một mục dữ liệu, hoặc tình huống hệ thống bị sự cố phải ngừng hoạt động trong khi đang thực hiện câu truy vấn. Đối với những câu truy vấn chỉ truy xuất, không có tình huống nào ở trên gây rắc rối. Người ta có thể cho hai câu truy vấn đọc dữ liệu cùng một lúc. Tương tự, sau khi đã xử lý xong sự cố, các truy vấn chỉ đọc chỉ cần khởi động lại. Nhưng ngược lại có thể nhận ra rằng đối với những câu truy vấn cập nhật, những tình huống này có thể gây ra những tổn hại nghiêm trọng cho cơ sở dữ liệu. Như chúng ta không thể chỉ khởi động lại cho câu truy vấn cập nhật sau một sự cố hệ thống vì một số giá trị của các mục dữ liệu có thể đã được cập nhật trước khi có sự cố xảy ra và không cho phép cập nhật lại khi câu truy vấn được khởi động lại, nếu không thì cơ sở dữ liệu sẽ chứa những dữ liệu sai lệch.

Điểm mấu chốt ở đây là không có khái niệm "thực thi nhất quán" hoặc "tính toán đáng tin cậy" đi kèm với khái niệm truy vấn. Khái niệm *giao tác* (transaction) được sử dụng trong lãnh vực cơ sở dữ liệu như một đơn vị tính toán nhất quán và tin cậy được. Vì thế các câu truy vấn sẽ được thực thi như các giao tác một khi các chiến lược thực thi được xác định và được dịch thành các thao tác cơ sở dữ liệu nguyên thủy.

Trong thảo luận ở trên chúng ta đã dùng thuật ngữ *"nhất quán"* (consistent) và *"đáng tin cậy"* (reliable) một cách hoàn toàn không hình thức. Thế nhưng do tầm quan trọng của chúng mà chúng ta cần phải định nghĩa chúng một cách chuẩn xác. Trước tiên phải chỉ ra rằng cần phân biệt giữa *nhất quán* cơ sở dữ liệu (database consistency) và *nhất quán giao tác* (transaction consistency).

Một cơ sở dữ liệu ở trong một *trạng thái nhất quán* (consistent state) nếu nó tuân theo tất cả các ràng buộc toàn vẹn (nhất quán) được định nghĩa trên nó. Dĩ nhiên chúng ta cần bảo đảm rằng cơ sở dữ liệu không bao giờ chuyển sang một trạng thái không nhất quán. Cơ sở dữ liệu có thể tạm thời không nhất quán trong khi thực hiện giao tác. Điều quan trọng là cơ sở dữ liệu phải trở về trạng thái nhất quán khi quan hệ giao tác chấm dứt.



Hình 6.1 Mô hình giao tác

Ngược lại, tính nhất quán giao tác muốn nói đến hành động của các giao tác đồng thời. Chúng ta mong rằng cơ sở dữ liệu vẫn nhất quán ngay cả khi có một số yêu cầu của người sử dụng đồng thời truy xuất đến cơ sở dữ liệu (đọc hoặc cập nhật). Tính chất phức tạp nảy sinh khi xét đến các cơ sở dữ liệu có nhân bản. Một cơ sở dữ liệu được nhân bản ở trong một *trạng thái nhất quán lẫn nhau* (mutually consistent state) nếu tất cả các bản sao của mỗi mục dữ liệu ở trong đó đều có giá trị giống nhau. Điều này thường được gọi là *sự tương đương một bản* (one copy equivalence) vì tất cả các bản đều bị buộc phải nhận cùng một trạng thái vào cuối lúc thực thi giao tác. Một số khái niệm về tính nhất quán bản sao cho phép giá trị các bản sao có thể khác nhau. Những vấn đề này sẽ được thảo luận sau.

Độ *tin cậy* hay *khả tín* (reliability) muốn nói đến khả năng *tự thích ứng* (resiliency) của một hệ thống đối với các loại sự cố và khả năng khôi phục lại từ những sự cố này. Một hệ thống khả tín sẽ tự thích ứng với các sự cố hệ thống và có thể tiếp tục cung cấp các dịch vụ ngay cả khi xảy ra sự cố. Một hệ quản trị cơ sở dữ liệu khả hồi phục là hệ quản trị cơ sở dữ liệu có thể chuyển sang trạng thái nhất quán (bằng cách quay trở lại trạng thái nhất quán trước đó hoặc chuyển sang một trạng thái nhất quán mới) sau khi gặp một sự cố.

Quản lý giao tác (transaction management) là giải quyết các bài toán duy trì được cơ sở dữ liệu ở trong tình trạng nhất quán ngay cả khi có nhiều truy xuất đồng thời và khi có sự cố.

Mục đích của chương này là định nghĩa những thuật ngữ cơ bản và đưa ra một bộ khung trên cơ sở đó để thảo luận các vấn đề này. Đây cũng là phần giới thiệu ngắn gọn về bài toán cần giải quyết và các vấn đề có liên quan. Vì thế chúng ta sẽ trình bày các khái niệm ở một mức trừu tượng khá cao và không trình bày những kỹ thuật quản lý. Trong phần tiếp theo chúng ta sẽ định nghĩa một cách hình thức và một cách trực quan về khái niệm giao tác.

6.1 Định nghĩa giao tác

Giao tác được xem như một dãy các thao tác đọc và ghi trên cơ sở dữ liệu cùng với các bước tính toán cần thiết.

Ví dụ 6.1

Xét câu truy vấn SQL làm tăng ngân sách của dự án CAD/CAM lên 10%

```
UPDATE PROJ
SET    BUTGET = BUTGET * 1.1
WHERE PNAME = “CAD/CAM”
```

Câu truy vấn này có thể được đặc tả, qua ký pháp SQL gắn kết, như một giao tác bằng cách cho nó một tên (ví dụ BUDGET_UPDATE) và khai báo như sau:

```
Begin_transaction BUDGET_UPDATE
begin
  EXEC SQL UPDATE PROJ
                SET  BUTGET = BUTGET * 1.1
                WHERE  PNAME = “CAD/CAM”
end.
```

Các câu lệnh **Begin_transaction** và **end** ấn định ranh giới một giao tác. Chú ý rằng việc sử dụng các ký hiệu phân cách này hoàn toàn không bắt buộc trong mọi hệ

quản trị cơ sở dữ liệu. Chẳng hạn nếu các dấu phân cách không được đặc tả, DB2 sẽ xử lý toàn bộ chương trình thực hiện truy xuất cơ sở dữ liệu như một giao tác.

Ví dụ 6.2

Trong phần thảo luận về các khái niệm quản lý giao tác, chúng ta sẽ sử dụng ví dụ về một hệ thống đặt chỗ máy bay. Cài đặt thực tế của ứng dụng này luôn sử dụng đến khái niệm giao tác. Chúng ta giả sử rằng có một quan hệ **FLIGHT** ghi nhận dữ liệu về các *chuyến bay* (flight), quan hệ **CUST** cho các khách hàng có đặt chỗ trước và quan hệ **FC** cho biết khách hàng nào sẽ đi trên chuyến bay nào. Chúng ta cũng giả sử rằng các định nghĩa quan hệ sau (thuộc tính gạch dưới biểu thị khoá):

FLIGHT (FNO, DATE, SRC, DEST, STSOLD, CAP)

CUST (CNAME, ADDR, BAL)

FC (FNO, DATE, CNAME, SPECIAL)

Định nghĩa thuộc tính trong lược đồ này như sau: **FNO** là mã số chuyến bay, **DATE** biểu ngày tháng chuyến bay, **STSOLD** chỉ số lượng ghế (seat) đã được bán trên chuyến bay đó, **CAP** chỉ sức chuyên chở (số lượng hành khách có thể chở được, capacity) trên chuyến bay, **CNAME** chỉ tên khách hàng với địa chỉ được lưu trong **ADDR** và số dư trong **BAL**, còn **SPECIAL** tương ứng với các yêu cầu đặc biệt mà khách hàng đưa ra khi đặt chỗ.

Chúng ta xét một phiên bản đơn giản hoá của một ứng dụng đặt chỗ, trong đó một nhân viên bán vé nhập mã số chuyến bay, ngày tháng, tên khách hàng và thực hiện đặt chỗ trước. Giao tác thực hiện công việc này có thể được cài đặt như sau, trong đó các truy xuất cơ sở dữ liệu được đặc tả bằng SQL:

```
Begin_transaction Revervation
begin
  input (flight_no, date, customer_name);
  EXEC SQL UPDATE FLIGHT
           SET STSOLD = STSOLD +1
           WHERE FNO = flight_no
```

```
                AND DATE = date;
    EXEC SQL INSERT
                INTO FC (FNO, DATE, CNAME, SPECIAL)
                VALUES (flight_no, date, customer_name, null);
    output ("reservation completed")
end.
```

6.1.1 Tình huống kết thúc giao tác

Một giao tác luôn luôn phải kết thúc ngay cả khi có xảy ra sự cố. Nếu giao tác có thể hoàn tất thành công tác vụ của nó, chúng ta nói rằng giao tác có *uỷ thác* (commit). Ngược lại nếu một giao tác phải ngừng lại khi chưa hoàn tất công việc, chúng ta nói rằng nó *bị hủy bỏ* (abort). Một giao tác phải tự hủy bỏ vì có một điều kiện làm cho nó không hoàn tất được công việc. Ngoài ra hệ quản trị cơ sở dữ liệu có thể hủy bỏ một giao tác, chẳng hạn do bị *khoá chết* (deadlock). Khi một giao tác bị hủy bỏ, quá trình thực thi sẽ ngừng và tất cả mọi hành động đã được thực hiện đều phải được "undo", đưa cơ sở dữ liệu trở về trạng thái trước khi thực hiện giao tác. Quá trình này gọi là "rollback".

Vai trò quan trọng của ủy thác biểu hiện ở hai mặt. Thứ nhất lệnh ủy thác báo cho hệ quản trị cơ sở dữ liệu biết rằng tác dụng của giao tác đó bây giờ cần được phản ảnh vào cơ sở dữ liệu, qua đó làm cho các giao tác đang truy xuất các mục dữ liệu đó có thể thấy được chúng. Thứ hai, điểm mà giao tác ủy thác là một điểm "không đường về". Kết quả của một giao tác đã ủy thác bây giờ được lưu cố định vào cơ sở dữ liệu và không thể phục hồi lại được.

Ví dụ 6.3

Một điều chúng ta chưa xét đến là tình huống không còn chỗ trống trên chuyến bay. Để bao quát khả năng này, giao tác cần được viết lại như sau:

```
Begin_transaction Reservation
begin
  input(flight_no, date, customer_name);
  EXEC SQL SELECT STSOLD, CAP
              INTO temp1, temp2
```

```
                FROM FLIGHT
                WHERE FNO = flight_no
                AND DATE = date;
        if temp1 = temp2 then
        begin
            Output("no free seat");
            Abort
        end
        else begin
            EXEC SQL UPDATE FLIGHT
                        SET STSOLD = STSOLD +1
                        WHERE FNO = flight_no
                        AND DATE = date;
            EXEC SQL INSERT
                        INTO FC(FNO, DATE, CNAME, SPECIAL)
                        VALUES(flight_no, date, customer_name, null);
            Commit;
            Output("reservation completed")
            end
        end-if
    end.
```

Qua ví dụ này chúng ta thấy được nhiều điểm quan trọng. Rõ ràng nếu không còn chỗ trống, giao tác phải hủy bỏ. Thứ hai là việc sắp thứ tự các kết quả để trình bày ra cho người sử dụng tùy theo các lệnh **abort** và **commit**. Chú ý rằng nếu giao tác bị hủy bỏ, người sử dụng sẽ được thông báo trước khi hệ quản trị cơ sở dữ liệu được hướng dẫn để hủy bỏ nó. Thế nhưng với trường hợp ủy thác, thông báo cho người sử dụng phải xảy ra sau khi hệ quản trị cơ sở dữ liệu đã thực hiện xong lệnh ủy thác để bảo đảm độ khả tín.

6.1.2 Đặc trưng hoá các giao tác

Chúng ta nhận xét rằng các giao tác đều đọc và ghi một số dữ liệu. Điều này được dùng làm cơ sở nhận biết một giao tác. Các mục dữ liệu được giao tác đọc cấu tạo nên *tập đọc* RS (read set) của nó. Tương tự, các mục dữ liệu được một giao tác ghi được gọi là *tập ghi* WS(write set). Chú ý rằng tập đọc và tập ghi của một giao tác không nhất thiết

phải tách biệt. Cuối cùng hợp của tập đọc và tập ghi của một giao tác tạo ra *tập cơ sở* BS (base set), nghĩa là BS = RS $\cup$ WS.

Ví dụ 6.4

Chúng ta xét lại giao tác đặt chỗ của ví dụ 6.3 và thao tác chèn chứa một số thao tác ghi. Các tập nêu trên được định nghĩa như sau:

RS [Revervation] = {FLIGHT.STSOLD, FLIGHT.CAP}

WS[Revervation] = {FLIGHT.STSOLD, FC.FNO, FC.DATE,

FC.CNAME, FC.SPECIAL}

BS[Revervation] = {FLIGHT.STSOLD, FLIGHT.CAP, FC.FNO,

FC.DATE, FC.CNAME, FC.SPECIAL}

Chúng ta đã đặc trưng các giao tác chỉ trên cơ sở các thao tác đọc và ghi mà không xem xét các thao tác chèn, xoá. Như thế chúng ta đã thảo luận về khái niệm các giao tác dựa trên các cơ sở dữ liệu tĩnh, không nới rộng hoặc thu lại. Giản lược này được đưa ra để có được tính đơn giản. Các cơ sở dữ liệu động phải giải quyết bài toán *ảnh ảo* (phantom), được giải thích như ví dụ sau. Xét giao tác T_1, khi thực hiện cần tìm trong bảng **FC** tên những khách hàng đã yêu cầu dùng một bữa ăn đặc biệt. Nó nhận được một tập **CNAME** gồm những khách hàng thỏa thuận điều kiện tìm kiếm. Khi T_1 đang thực hiện, một giao tác T_2 chèn các bộ mới vào **FC** có yêu cầu bữa ăn đặc biệt rồi ủy thác. Nếu sau đó T_1 lại đưa ra câu truy vấn tìm kiếm như cũ, nó sẽ nhận được một tập **CNAME** khác với tập ban đầu mà nó đã nhận. Vì thế các bộ "ảnh ảo" đã xuất hiện trong cơ sở dữ liệu.

6.1.3. Hình thức hoá khái niệm giao tác

Cho đến lúc này, ý nghĩa trực quan của giao tác đã hoàn toàn rõ ràng. Để bàn luận về các giao tác và suy diễn tính đúng đắn của các thuật toán quản lý giao tác, chúng ta cần định nghĩa khái niệm này một cách hình thức. Chúng ta biểu thị *phép toán* Oj của giao tác T_i khi hoạt tác trên thực thể cơ sở dữ liệu x là $O_{ij}(x)$. Theo qui ước được thừa nhận ở phần trước, $O_{ij} \in$ {read, write}. Các phép toán được giả thiết là *nguyên tử* (nghĩa là mỗi phép toán được thực thi như một đơn vị không thể chia nhỏ được nữa). Chúng ta

hãy ký hiệu OS_i là tập tất cả các phép toán trong T_i (nghĩa là $OS_i = U_j O_{ij}$). Ni biểu thị cho tình huống của T_i, trong đó $N_i \in \{abort, commit\}$.[2]

Với các thuật ngữ này, chúng ta có thể định nghĩa T_i là một *thứ tự bộ phận* trên các phép toán và tình huống kết thúc của nó. Thứ tự bộ phận $P = \{\sum, \prec\}$ định nghĩa một trật tự giữa các phần tử của $\sum$ (được gọi là *miền*) qua một quan hệ hai ngôi bắc cầu và không phản xạ $\prec$ được định nghĩa trên $\sum$. Trong trường hợp đang xét, $\sum$ bao gồm các phép toán và tình huống kết thúc của một giao tác, trong đó $\prec$ chỉ thứ tự thực hiện của các phép toán này (được chúng ta đọc là "đứng trước theo thứ tự thực thi").

Một cách hình thức, một giao tác T_I là một thứ tự bộ phận $T_I = \{\sum_i, <_i\}$, trong đó

1. $\sum_i = OS_i \cup \{N_i\}$.
2. Với hai phép toán bất kỳ $O_{ij}, O_{ik} \in OS_i$, nếu $O_{ij} = \{R(x) \text{ or } W(x)\}$ và $O_{ik} = w(x)$ với một mục dữ liệu x nào đó, thế thì $O_{ij} <_i O_{ik}$ hoặc $O_{ik} <_i O_{ij}$.
3. $\forall O_{ij} \in OS_i, O_{ij} <_i N_i$.

Điều kiện thứ nhất về hình thức định nghĩa *miền* như một tập các thao tác đọc và ghi cấu tạo nên giao tác cộng với tình huống kết thúc, có thể là *commit* hoặc *abort*. Điều kiện thứ hai xác định quan hệ thứ tự giữa các thao tác đọc và ghi có tương tranh của giao tác, còn điều kiện cuối cùng chỉ ra rằng tình huống kết thúc luôn đi sau tất cả những thao tác khác.

Có hai điểm quan trọng cần lưu ý trong định nghĩa này. Trước tiên, quan hệ thứ tự $\prec$ được cho trước và định nghĩa này không hề xây dựng nó. Quan hệ thứ tự thực sự phụ thuộc vào ứng dụng. Kế đến, điều kiện thứ hai chỉ ra rằng thứ tự giữa các thao tác có tương tranh phải tồn tại bên trong $\prec$. Hai thao tác $O_i(x)$ và $O_j(x)$ được gọi là có tương tranh nếu O_i = Write hoặc O_j = Write (có nghĩa ít nhất một trong chúng là Write và chúng truy xuất cùng một mục dữ liệu).

Ví dụ 6.5: Xét một giao tác đơn giản T có các bước sau:

Read(x)

Read(y)

x← x+y

Write(x)

Commit

Đặc tả của giao tác này theo ký pháp hình thức đã được giới thiệu ở trên là:

Σ = {R(x), R(y), W(x), C}

< = {(R(x), W(x)), (R(y), W(x)), (W(x),C), (R(x), C), (R(y), C)}

trong đó (Oi, Oj), là một phần tử của quan hệ $\prec$, biểu thị rằng Oi < Oj.

Chú ý rằng quan hệ thứ tự tương đối của tất cả các thao tác ứng với tình huống kết thúc. Điều này là do điều kiện thứ ba của định nghĩa giao tác. Cũng cần chú ý rằng chúng ta không mô tả thứ tự giữa mỗi cặp thao tác. Điều đó giải thích vì sao đây là một thứ tự bộ phận.

Ví dụ 6.6

Giao tác đặt chỗ được xây dựng trong ví dụ 6.3 phức tạp hơn. Chú ý cho rằng có hai tình huống kết thúc, tùy vào tình trạng có còn chỗ trống hay không. Trước tiên, điều này dường như mâu thuẫn với định nghĩa của giao tác, đó là chỉ tồn tại một tình huống kết thúc. Tuy nhiên cần nhớ rằng giao tác là một thực thi của một chương trình. Rõ ràng là trong bất kỳ lần thực thi nào, chỉ một trong hai tình huống kết thúc xảy ra. Vì thế điều có thể xảy ra là một giao tác hủy bỏ và một giao tác khác ủy thác. Sử dụng ký pháp hình thức này, giao tác đầu có thể được đặc tả như sau:

Σ = {R(STSOLD); R(CAP),A}

< = {(O_1,A), (O_2,A)}

và giao tác sau được đặc tả như sau

Σ = {R(STSOLD), R(CAP), W(STSOLD), W(FNO),

W(DATE), W(CNAME), W(SPECIAL), C}

< = {(O_1,O_3), (O_2,O_3), (O_1,O_4), (O_1, O_5), (O_1, O_6),

(O_1, O_7), (O_2, O_4), (O_2, O_5), (O_2, O_6), (O_2, O_7),

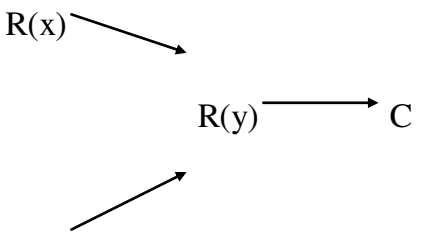
$(O_1, C), (O_2, C), (O_3, C) (O_4, C), (O_5, C), (O_6, C), (O_7, C)\}$

trong đó O_1 = R(STSOLD), O_2 = R(CAP), O_3 = W(STSOLD), O_4 = W(FNO), O_5= W(DATE), O_6 = W(CNAME), O_7 = W(SPECIAL).

Một ưu điểm của việc định nghĩa giao tác như một thứ tự bộ phận là sự tương ứng của nó với *đồ thị có hướng không vòng* DAG (directed acylic graph). Như thế một giao tác có thể được đặc tả như một DAG với đỉnh là các phép toán giao tác và cung chỉ ra mối liên hệ thứ tự giữa các cặp phép toán đã cho.

Ví dụ 6.7

Giao tác được thảo luận trong ví dụ 5 có thể được vẽ như một DAG của hình 6.2. Chú ý rằng chúng ta không vẽ các cung được suy ra nhờ tính chất bắc cầu dù rằng chúng ta xem chúng như những phần tử của $\prec$



Hình 6.2 Biểu diễn dạng DAG cho một giao tác

Trong phần lớn các trường hợp chúng ta không cần phải đề cập đến nhiều miền của thứ tự bộ phận một cách riêng rẽ với quan hệ thứ tự. Vì thế thông thường chúng ta bỏ $\sum$ ra khỏi định nghĩa giao tác và sử dụng tên của thứ tự bộ phận để chỉ đến cả miền lẫn tên của thứ tự bộ phận. Điều đó sẽ tiện lợi bởi vì nó cho phép chúng ta đặc tả thứ tự của các phép toán trong một giao tác nhờ một phương thức khá đơn giản bằng cách dùng thứ tự tương đối của định nghĩa giao tác. Chẳng hạn chúng ta có thể định nghĩa giao tác của ví dụ 5 như sau:

T = {R(x), R(y), W(x), C}

thay vì đặc tả dài dòng như trước.

6.2 Các tính chất của giao tác

Các khía cạnh nhất quán và khả tín của giao tác là do bốn tính chất: (1) *tính nguyên tử* (atomicity), (2) *tính nhất quán* (consistency), (3) *tính biệt lập* (isolation), (4) *tính bền vững* (durability); và chúng ta thường được gọi chung là tính chất ACID.

6.2.1 Tính nguyên tử

Tính nguyên tử liên quan đến sự kiện là một giao tác được xử lý như một đơn vị hoạt tác. Chính vì thế mà các hành động của giao tác, hoặc tất cả đều hoàn tất hoặc không một hành động nào hoàn tất. Điều này cũng được gọi là tính chất "được ăn cả ngã về không" (all-or-nothing). Tính nguyên tử đòi hỏi rằng nếu việc thực thi giao tác bị cắt ngang bởi một loại sự cố nào đó thì hệ quản trị cơ sở dữ liệu sẽ chịu trách nhiệm xác định công việc cần thực hiện đối với giao tác để khôi phục lại sau sự cố. Dĩ nhiên có hai chiều hướng hành động: hoặc nó sẽ được kết thúc bằng cách hoàn tất các hành động còn lại, hoặc có thể được kết thúc bằng cách hồi lại tất cả các hành động đã được thực hiện.

6.2.2 Tính nhất quán

Tính nhất quán (consistency) của một giao tác chỉ đơn giản là tính đúng đắn của nó. Nói cách khác, một giao tác là một chương trình đúng đắn, ánh xạ cơ sở dữ liệu từ trạng thái nhất quán này sang một trạng thái nhất quán khác.

Trong định nghĩa dưới đây, dữ liệu rác (dirty data) muốn nói đến những giá trị dữ liệu đã được cập nhật bởi một giao tác trước khi nó ủy thác. Do đó dựa trên khái niệm về dữ liệu rác, bốn mức độ được định nghĩa như sau:

Độ 3: Giao tác T thỏa **nhất quán độ 3** nếu:

1. T không đè lên dữ liệu rác của những giao tác khác.
2. T không ủy thác bất kỳ thao tác ghi nào cho đến khi nó hoàn tất mọi thao tác ghi [nghĩa là cho đến lúc cuối giao tác (end-of-transaction, EOT)].
3. T không đọc dữ liệu rác của những giao tác khác.
4. Những giao tác khác không làm cho dữ liệu mà T đã đọc trước khi T hoàn tất trở thành dữ liệu rác.

Độ 2: Giao tác T thỏa **nhất quán độ 2** nếu:

1. T không đè lên dữ liệu rác của những giao tác khác.
2. T không ủy thác bất kỳ thao tác ghi nào trước EOT.
3. T không đọc dữ liệu rác của những giao tác khác.

Độ 1: Giao tác T thỏa **nhất quán độ 1** nếu:

1. T không đè lên dữ liệu rác của những giao tác khác.
2. T không ủy thác bất kỳ thao tác ghi nào trước EOT.

Đương nhiên độ nhất quán cao bao trùm tất cả độ nhất quán mức thấp hơn. Ý tưởng trong việc định nghĩa nhiều mức nhất quán là cung cấp cho lập trình viên ứng dụng một khả năng linh hoạt khi định nghĩa các giao tác hoạt tác ở những mức khác nhau. Hệ quả là mặc dù một số giao tác hoạt tác ở mức nhất quán Độ 3, các giao tác khác có thể hoạt tác ở những mức thấp hơn, và rất có thể sẽ nhìn thấy các dữ liệu rác.

6.2.3 Tính biệt lập

Biệt lập là tính chất của các giao tác, đòi hỏi mỗi giao tác phải luôn nhìn thấy cơ sở dữ liệu nhất quán. Nói cách khác, một giao tác đang thực thi không thể làm lộ ra các kết quả của nó cho những giao tác khác đang cùng hoạt động trước khi nó ủy thác.

Có một số lý do cần phải nhấn mạnh đến tính biệt lập. Một là duy trì tính nhất quán qua lại giữa các giao tác. Nếu hai giao tác đồng thời truy xuất đến một mục dữ liệu đang được một trong chúng cập nhật thì không thể bảo đảm rằng giao tác thứ hai sẽ đọc được giá trị đúng.

Ví dụ 6.8

Xét hai giao tác đồng thời T1 và T2 cùng truy xuất đến mục dữ liệu x. Giả sử giá trị của x trước khi bắt đầu thực hiện là 50.

T1:	Read(x)	T2:	Read(x)
	x← x + 1		x← x + 1

Write(x) Write(x)

Commit Commit

Dưới đây là một dãy thực thi cho các hành động này.

T_1: Read(x)

T_1: x← x + 1

T_1: Write(x)

T_1: Commit

T_2: Read(x)

T_2: x← x + 1

T_2: Write(x)

T_2: Commit

Ở trường hợp này không có vấn đề gì; các giao tác T_1 và T_2 được thực hiện lần lượt và giao tác T_2 đọc được giá trị của x là 51. Chú ý rằng nếu T_2 thực thi trước T_1 thì T_2 đọc được giá trị 50. Vì thế nếu T1 và T2 được thực thi lần lượt giao tác này rồi đến giao tác kia, giao tác thứ hai sẽ đọc được giá trị của x là 51 và sau khi kết thúc hai giao tác x có giá trị 52. Tuy nhiên vì các giao tác đang thực thi đồng thời, dãy thực thi sau đây có thể sẽ xảy ra:

T_1: Read(x)

T_1: x← x + 1

T_2: Read(x)

T_1: Write(x)

T_2: x← x + 1

T_2: Write(x)

T_1: Commit

T_2: Commit

Trong trường hợp này, giao tác T_2 đọc được giá trị của x là 50. Giá trị này không đúng bởi vì T_2 đọc x trong khi giá trị của nó đang được thay đổi từ 50 thành 51. Hơn nữa giá trị của x sẽ là 51 vào lúc kết thúc các giao tác T_1 và T_2 bởi vì hành động ghi của T_2 sẽ đè lên kết quả ghi của T_1.

Bảo đảm tính biệt lập bằng cách không cho phép các giao tác khác nhìn thấy các kết quả chưa hoàn tất như trong Ví dụ trên sẽ giải quyết được vấn đề *cập nhật thất lạc* (lost update). Loại biệt lập này đã được gọi là *tính ổn định con chạy* (cursor stability). Trong Ví dụ ở trên, dãy thực thi thứ hai đã làm cho tác dụng của T_1 bị mất. Một lý do thứ hai của tính biệt lập là các *hủy bỏ dây chuyền* (cascading abort). Nếu một giao tác cho phép những giao tác khác nhìn thấy những kết quả chưa hoàn tất của nó trước khi ủy thác rồi nó quyết định hủy bỏ, mọi giao tác đã đọc những giá trị chưa hoàn tất đó cũng sẽ phải hủy bỏ. Xâu mắc xích này dễ làm tăng nhanh và gây ra những phí tổn đáng kể cho hệ quản trị cơ sở dữ liệu.

Cũng có thể xử trí các mức nhất quán đã thảo luận trong phần trước từ quan điểm của tính chất biệt lập (vì thế đã minh hoạ cho sự phụ thuộc giữa tính nhất quán và tính biệt lập). Khi di chuyển lên cây phân cấp các mức nhất quán, các giao tác ngày càng biệt lập hơn. Độ 0 cung cấp rất ít tính chất "biệt lập" ngoài việc ngăn cản các cập nhật thất lạc. Tuy nhiên vì các giao tác sẽ ủy thác trước khi chúng hoàn tất tất cả mọi thao tác ghi của chúng, nếu có một hủy bỏ xảy ra sau đó, nó sẽ đòi hỏi phải hồi lại tất cả các cập nhật trên các mục dữ liệu đã được ủy thác và hiện đang được truy xuất bởi những giao tác khác. Nhất quán độ 2 tránh được các hủy bỏ dây chuyền. Độ 3 cung cấp toàn bộ khả năng biệt lập, buộc một trong các giao tác tương tranh phải đợi cho đến khi giao tác kia kết thúc. Những dãy thực thi như thế được gọi là nghiêm ngặt (strict) và sẽ được thảo luận nhiều hơn trong chương tiếp theo. Rõ ràng là vấn đề biệt lập có liên quan trực tiếp đến tính nhất quán cơ sở dữ liệu và vì thế là đề tài của điều khiển đồng thời.

Ba hiện tượng được đặc tả cho những tình huống có thể xảy ra nếu sự biệt lập thích hợp không được duy trì là:

Đọc rác (Dirty Read): dữ liệu rác muốn nói đến các mục dữ liệu mà giá trị của chúng đã được sửa đổi bởi một giao tác chưa ủy thác. Xét trường hợp giao tác T1 sửa đổi một giá trị dữ liệu rồi nó lại được bọc bởi một giao tác T2 khác trước khi T1 thực hiện Commit hay Abort. Trong trường hợp Abort, T2 đã đọc một giá trị chưa được tồn tại trong cơ sở dữ liệu.

Một đặc tả chính xác trong hiện tượng này như sau (với các cước số chỉ ra tên các giao tác)

…, $W_1(x)$,…, $R_2(x)$,… C_1(hoặc A_1),…, C_2(hoặc A_2)

hoặc

…, $W_1(x)$,…, $R_2(x)$,… C_2(hoặc A_2),…, C_1(hoặc A_1)

Đọc bất khả lập (Non-repeatable Read): Giao tác T1 đọc một mục dữ liệu. Sau đó một giao tác T2 khác sửa hoặc xoá mục dữ liệu đó rồi ủy thác. Nếu sau đó T1 đọc lại mục dữ liệu đó, hoặc nó đọc được một giá trị khác hoặc nó không thể tìm thấy được mục đó; vì thế hai hành động đọc trong cùng một giao tác T1 trả về các kết quả khác nhau.

Một đặc tả chính xác của hiện tượng này như sau:

…, $R_1(x)$,…, $W_2(x)$,… C_1(hoặc A_1),…, C_2(hoặc A_2)

hoặc

…, $R_1(x)$,…, $W_2(x)$,… C_2(hoặc A_2),…, C_1(hoặc A_1)

Anh ảo (phantom): Điều kiện ảnh ảo trước kia đã được định nghĩa xảy ra khi T1 thực hiện tìm kiếm theo một vị từ và T2 chèn những bộ mới thỏa vị từ đó. Đặc tả chính xác của hiện tượng này là (P là vị từ tìm kiếm)

…, $R_1(P)$,…, W_2(y thuộc P),… C_1(hoặc A_1),…, C_2(hoặc A_2)

hoặc

…, $R_1(P)$,…, R_2(y thuộc P),… C2(hoặc A_2),…, C_1(hoặc A_1)

Dựa trên những hiện tượng này, các mức biệt lập đã được định nghĩa như sau. Mục tiêu của việc định nghĩa nhiều mức biệt lập cũng giống như việc định nghĩa các mức nhất quán.

6.2.4 Tính bền vững

Tính bền vững (durability) muốn nói đến tính chất của giao tác, bảo đảm rằng một khi giao tác ủy thác, kết quả của nó được duy trì cố định và không bị xóa ra khỏi cơ sở dữ liệu. Vì thế hệ quản trị cơ sở dữ liệu bảo đảm rằng kết quả của giao tác sẽ vẫn tồn tại dù có xảy ra sự cố hệ thống. Đây chính là lý do mà trong ví dụ 6.2 chúng ta đã nhấn mạnh rằng giao tác ủy thác trước khi nó thông báo cho người sử dụng biết rằng nó đã hoàn tất thành công. Tính bền vững đưa ra vấn đề *khôi phục* cơ sở dữ liệu (database recovery), nghĩa là cách khôi phục cơ sở dữ liệu về trạng thái nhất quán mà ở đó mọi hành động đã ủy thác đều được phản ánh.

6.3 Các loại giao tác

6.3.1 Giao tác phẳng

Giao tác phẳng (flat transaction) có một khởi điểm duy nhất (**Begin_transaction**) và một điểm kết thúc duy nhất (**End_transaction**). Tất cả các ví dụ của chúng ta đã xem xét đều nằm trong nhóm này. Phần lớn các nghiên cứu về quản lý giao tác trong cơ sở dữ liệu đều tập trung vào các giao tác phẳng.

6.3.2 Giao tác lồng

Đây là mô hình giao tác cho phép một giao tác gồm chứa giao tác khác với điểm bắt đầu và ủy thác của riêng chúng. Những giao tác như thế được gọi là *giao tác lồng* (nested transaction). Những giao tác được đặt vào trong giao tác khác thường được gọi là *giao tác con* (subtransaction)

Ví dụ 6.9

Chúng ta hãy mở rộng giao tác đặt chỗ của ví dụ 2. Phần lớn các hãng du lịch đều lo cả việc đặt chỗ khách sạn và mướn ô tô ngoài dịch vụ đặt vé máy bay. Nếu người ta muốn mô tả tất cả những công việc này bằng một giao tác, thì giao tác đặt chỗ sẽ có cấu trúc như sau:

Begin_transaction Reservation

```
begin
  Begin_transaction Airline
  …
  end. {Airline}
  Begin_transaction Hotel
  …
  end. { Hotel}
  Begin_transaction Car
  …
  end. {Car }
end.
```

Các giao tác lồng đã được chú ý như một khái niệm giao tác tổng quát hơn. Mức độ lồng nói chung là để ngỏ, cho phép các giao tác con cũng có thể có các giao tác lồng. Tính tổng quát này có ích trong các lãnh vực ứng dụng mà ở đó các giao tác phức tạp hơn so với việc xử lý dữ liệu truyền thống.

6.4 Điều khiển đồng thời phân tán

6.4.1 Lý thuyết khả tuần tự

Trong phần đầu của chương, chúng ta thảo luận vấn đề làm biệt lập các giao tác với nhau theo tác dụng của chúng trên CSDL. Chúng ta cũng đã chỉ ra rằng nếu việc thực thi đồng thời các giao tác làm cho CSDL ở một trạng thái có thể có được giống như khi cho chúng thực hiện tuần tự theo một số thứ tự nào đó, các vấn đề như cập nhật bị thất lạc sẽ được giải quyết. Đây là điểm mấu chốt của những lý luận về tính khả tuần tự. Phần còn lại sẽ tập trung vào các vấn đề khả tuần tự một cách hình thức hơn.

Một *lịch* S (schedule) được định nghĩa trên tập giao tác $T = \{T_1, T_2, \ldots, T_n\}$ và xác định thứ tự thực thi đan xen lẫn nhau của các thao tác trong giao dịch. Dựa trên định nghĩa giao tác đã được giới thiệu trong phần 6.1, lịch có thể mô tả như một thứ tự bộ phận trên T. Dầu vậy chúng ta cũng cần một khái niệm cơ bản trước khi đưa ra định nghĩa hình thức.

Nhắc lại định nghĩa về thao tác tương tranh đã đưa ra trong chương 5. Hai thao tác $O_{ij}(x)$ và $O_{kl}(x)$ (i và k không nhất thiết phải phân biệt) cùng truy cập đến một thực thể

CSDL x được gọi là *có tương tranh* (conflict) nếu ít nhất một trong chúng là *thao tác ghi* (write). Hai điều mà chúng ta cần chú ý trong định nghĩa này.

- Trước tiên các thao tác đọc không tương tranh với nhau. Vì thế chúng ta có thể nói về hai loại tương tranh: *đọc-ghi* (read-write) và *ghi-ghi* (write-write).

- Thứ hai, hai thao tác này có thể thuộc về cùng một giao tác hoặc thuộc về hai giao tác khác nhau .Trong trường hợp sau, hai giao tác được gọi là có tương tranh. Về trực quan, sự tồn tại của một tương tranh giữa hai thao tác cho thấy rằng thứ tự thực hiện của chúng ta là quan trọng. Việc sắp thứ tự cho hai thao tác đọc là không cần thiết.

Trước tiên chúng ta định nghĩa *một lịch đầy đủ* (complete schedule): là lịch định nghĩa thứ tự thực hiện của tất cả các thao tác trong miền biến thiên của nó. Sau đó chúng ta định nghĩa rằng một lịch được xem là một *tiền tố* (prefix) của một lịch đầy đủ. Về hình thức, một lịch đầy đủ S^c_T được định nghĩa trên một tập giao tác $T = \{T_1, T_2,..., T_n\}$ là một thứ tự bộ phận $= \{_T, <_T\}$, trong đó

1. $\sum_T = \cup^n_{i=1} \sum_i$
2. $<_T = \cup^n_{i=1} <_i$
3. Đối với hai thao tác trong tương tranh bất kỳ $O_{ij}, O_{kl} \in \sum_T$, chúng ta có

   $O_{ij} <_T O_{kl}$ hoặc $O_{kl} <_T O_{ij}$.

Ví dụ 6.10:

T_1:	Read(x)	T_2:	Read(x)
	X ← x + 1		X ← x + 1
	Write(x)		Write(x)
	Commit		Commit

Một lịch đầy đủ S^c_T khả hữu trên $T = \{T_1, T_2\}$ có thể được viết như thứ tự bộ phận sau đây (các chỉ số dưới biểu thị giao dịch):

$S^c_T = \{_T, <_T\}$

trong đó

$$\Sigma_1 = \{R_1(x), W_1(x), C_1\}$$

$$\Sigma_2 = \{R_2(x), W_2(x), C_2\}$$

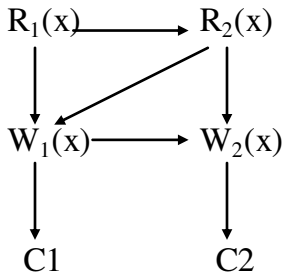
Vì vậy

$$\Sigma_T = \Sigma_1 \cup \Sigma_2 = \{R_1(x), W_1(x), C_1, R_2(x), W_2(x), C_2\}$$

và

$$<_T = \{(R_1, R_2), (R_1, W_1), (R_1, C_1), (R_1, W_2), (R_1, C_2), (R_2, W_1), (R_2, C_1), (R_2, W_2), (R_2, C_2), (W_1, C_1), (W_1, W_2), (W_1, C_2), (C_1, W_2), (C_1, C_2), (W_2, C_2)\}$$

có thể đặc tả như một DAG trong hình 6.3.



Hình 6.3 Biểu diễn DAG của một lịch đầy đủ

$$S^c_T = \{R_1(x), R_2(x), W_1(x), C_1, W_2(x), C_2\}$$

Một lịch được định nghĩa là một *tiền tố* (prefix) của một lịch đầy đủ. Một tiền tố của một thứ tự bộ phận có thể được định nghĩa như sau. Cho trước một thứ tự bộ phận P = {, <}, P' = {', <'} là một tiền tố của P nếu

1. $\Sigma' \subseteq \Sigma$;
2. $\forall e_i \in \Sigma'$, $e_1 <' e_2$ nếu và chỉ nếu $e_1 < e_2$; và
3. $\forall e_i \in \Sigma'$, nếu $\exists e_j \in \Sigma$ và $e_j < e_i$ thì $e_j \in \Sigma'$.

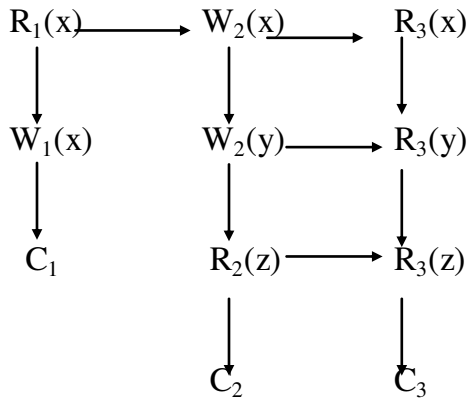
Hai điều kiện đầu tiên định nghĩa P' như một hạn chế của P trên miền ', trong đó các quan hệ thứ tự trong P được duy trì trong P'. Điều kiện cuối cùng chỉ ra rằng với mọi phần tử của ', tất cả các phần tử đứng trước nó trong cũng phải thuộc'.

Định nghĩa một lịch thức như một tiền tố của một thứ tự bộ phận để làm gì? Câu trả lời đơn giản là chúng ta bây giờ có thể xử lý các lịch không đầy đủ. Điều này là có ích vì một số lý do. Từ quan điểm lý thuyết khả tuần tự, chúng ta chỉ phải giải quyết một số thao tác của các giao tác có tương tranh chứ không phải với tất cả mọi thao tác. Hơn nữa, và có lẽ quan trọng hơn là khi xuất hiện sự cố, chúng ta cần phải có khả năng giải quyết với những giao tác không đầy đủ, mà đó là điều một tiền tố cho phép chúng ta làm được.

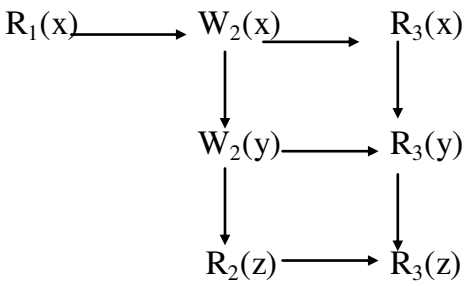
Ví dụ 6.11: Xét ba giao tác sau đây

T_1:	Read(x)	T_2:	Write(x)	T_3:	Read(x)
	Write(x)		Write(y)		Read(y)
	Commit		Read(z)		Read(z)
			Commit		Commit

Một lịch đầy đủ S^c cho những giao tác này được trình bày trong hình 6.4, và một lịch S (một tiền tố của S^c) được mô tả trong hình 6. 5.



Hình 6.4 Một lịch đầy đủ



Hình 6.5 Tiền tố của lịch đầy đủ của hình 6.4

Nếu trong lịch S, các thao tác của các giao tác khác nhau không được thực hiện xen kẽ (nghĩa là các thao tác của mỗi giao tác xảy ra liên tiếp), lịch được gọi là *tuần tự* (serial).

Ví dụ 6.12: Xét ba giao tác của ví dụ 6.11. Lịch sau đây:

$$S = \{W_2(x), W_2(y), R_2(z), C_2, R_1(x), W_1(x), C_1, R_3(x), R_3(y), R_3(z), C_3\}$$

là tuần tự bởi vì tất cả các thao tác của T_2 được thực hiện trước tất cả các thao tác của T_1 và tất cả thao tác của T_1 được thực hiện trước tất cả các thao tác của T_3. Một cách thường được dùng để biểu thị mối liên hệ thứ bậc giữa các thực thi giao tác là $T_2 <_s T_1 <_s T_3$ hoặc $T_2 \rightarrow T_1 \rightarrow T_3$.

Về trực quan, hai lịch S_1 và S_2 được định nghĩa trên cùng một tập giao tác T được gọi là *tương đương* nếu với mỗi cặp thao tác tương tranh O_{ij} và O_{kl} ($i \neq k$), mỗi khi $O_{ij} <_1 O_{ki}$ thì $O_{ij} <_2 O_{kl}$. Đây được gọi là *tương đương tương tranh* (conflict equivalence) bởi vì nó định nghĩa sự tương đương của hai lịch theo thực thi tương đối của các thao tác tương tranh trong các lịch biểu.

Ví dụ 6.13: Chúng ta xét lại ba giao tác của ví dụ 6.11. Lịch S dưới đây được định nghĩa trên chúng là tương đương tương tranh với S của Ví dụ 3:

$$S' = \{W_2(x), R_1(x), W_1(x), C_1, R_3(x), W_2(y), R_3(y), R_2(z), C_2, R_3(z), C_3\}$$

Một lịch S được gọi là *khả tuần tự* (serializable) nếu và chỉ nếu có tương đương tương tranh với một lịch tuần tự.

Chú ý rằng tính khả tuần tự chỉ tương đương với tính nhất quán độ 3 đã được định nghĩa trong phần 6.2.2. Tính khả tuần tự được định nghĩa như thế cũng được gọi là *khả tuần tự theo tương tranh* bởi vì nó được định nghĩa theo sự tương đương tương tranh.

Ví dụ 6.14: Lịch S' trong ví dụ 6.13 là khả tuần tự bởi vì nó tương đương với lịch tuần tự S của ví dụ 3. Cũng chú ý rằng vấn đề khi thực hiện một cách không kiểm soát các giao tác T_1 và T_2 đó là chúng có thể sinh ra một lịch bất khả tuần tự.

Bây giờ khi đã định nghĩa một cách hình thức tính khả tuần tự, chúng ta có thể chỉ ra rằng chức năng cơ bản của bộ phận điều khiển đồng thời là tạo ra một lịch khả tuần tự để thực hiện các giao tác đang chờ đợi.

Lý thuyết khả tuần tự có thể mở rộng cho các CSDL phân tán không nhân bản (hoặc phân hoạch). Lịch thực thi giao tác tại mỗi vị trí được gọi là *lịch cục bộ* (local schedule). Nếu CSDL không được nhân bản và mỗi lịch cục bộ đều khả tuần tự thì hợp của chúng (được gọi là lịch toàn cục) cũng khả tuần tự, với điều kiện là các thứ tự tuần tự hoá cục bộ đều giống nhau. Tuy nhiên trong các CSDL phân tán có nhân bản, mở rộng lý thuyết khả tuần tự đòi hỏi phải cẩn trọng. Có thể là các lịch cục bộ khả tuần tự nhưng tính nhất quán tương hỗ của CSDL vẫn bị tổn hại.

Ví dụ 6.15: Chúng ta sẽ đưa ra một ví dụ rất đơn giản nhằm minh hoạ cho điều này. Xét hai vị trí và một mục dữ liệu (x) hiện diện cả hai tại cả hai nơi. Xét giao tác sau:

T_1:	Read(x)	T_2:	Read(x)
	x← x + 5		x← x + 5
	Write(x)		Write(x)
	Commit		Commit

Rõ ràng cả hai giao tác đều phải thực hiện ở cả hai nơi. Xét các lịch có thể được tạo ra tại hai vị trí đó:

$S_1 = \{R_1(x), W_1(x), C_1, R_2(x), W_2(x), C_2\}$

$S_2 = \{R_2(x), W_2(x), C_2, R_1(x), W_1(x), C_1\}$

Tính nhất quán tương hỗ đòi hỏi rằng tất cả các giá trị của mọi mục dữ liệu nhân bản đều phải như nhau. Các lịch có thể duy trì được tính nhất quán tương hỗ được gọi là *khả tuần tự một bản* (one-copy serializable).

Về trực quan, lịch toàn cục khả tuần tự một bản phải thỏa mãn những điều kiện sau:

Mỗi lịch cục bộ đều phải khả tuần tự.

Hai thao tác tương tranh phải có cùng thứ tự tương đối trong tất cả các lịch cục bộ nơi mà chúng cùng xuất hiện.

Điều kiện thứ hai chỉ nhằm bảo đảm rằng thứ tự tuần tự hóa đều như nhau tại tất cả mọi vị trí có các giao tác tương tranh cùng thực hiện. Cần nhớ rằng các thuật toán điều khiển đồng thời bảo đảm được tính khả tuần tự bằng cách đồng bộ hóa các truy xuất tương tranh đến CSDL. Trong các CSDL nhân bản, nhiệm vụ bảo đảm tính khả tuần tự một bản thường là trách nhiệm của *nghi thức điều khiển bản sao* (replica control protocol).

Chúng ta hãy giả sử là tồn tại một mục dữ liệu x với các bản sao $x_1, x_2, \ldots, x_n$. Chúng ta sẽ xem như x là một mục dữ liệu logic và mỗi bản sao là một mục dữ liệu vật lý. Nếu tính vô hình nhân bản được cung cấp, các giao tác của người sử dụng sẽ đưa ra các thao tác đọc và ghi trên mục dữ liệu x. Nghi thức điều khiển bản sao chịu trách nhiệm ánh xạ mỗi thao tác đọc trên mục dữ liệu logic x [Read(x)] thành thao tác đọc trên một trong những bản sao vật lý x_j của x [Read(x_j)]. Ngược lại, mỗi thao tác ghi trên mục logic x được ánh xạ thành một tập thao tác ghi trên một tập con (có thể là tập con thực sự) của các bản sao vật lý x. Bất kể ánh xạ chuyển đến toàn bộ tập của các bản sao hay chỉ đến một tập con thì nó vẫn là cơ sở để phân loại các thuật toán điều khiển bản sao. Trong chương này và phần lớn cuốn sách, chúng ta sẽ xét các nghi thức điều khiển bản sao ánh xạ một thao tác đọc trên một mục logic đến một bản sao của nó nhưng lại ánh xạ thao tác ghi thành tập các thao tác ghi trên tất cả các bản sao vật lý. Nghi thức này thường được gọi là *nghi thức đọc một/ghi tất cả* (read-one/write-all protocol, ROWA).

Một nhược điểm của nghi thức ROWA là nó làm giảm độ khả dụng của CSDL, khi có sự cố bởi vì giao tác có thể không hoàn tất được trừ khi nó đã phản ánh tác dụng của tất cả các thao tác ghi trên các bản sao.

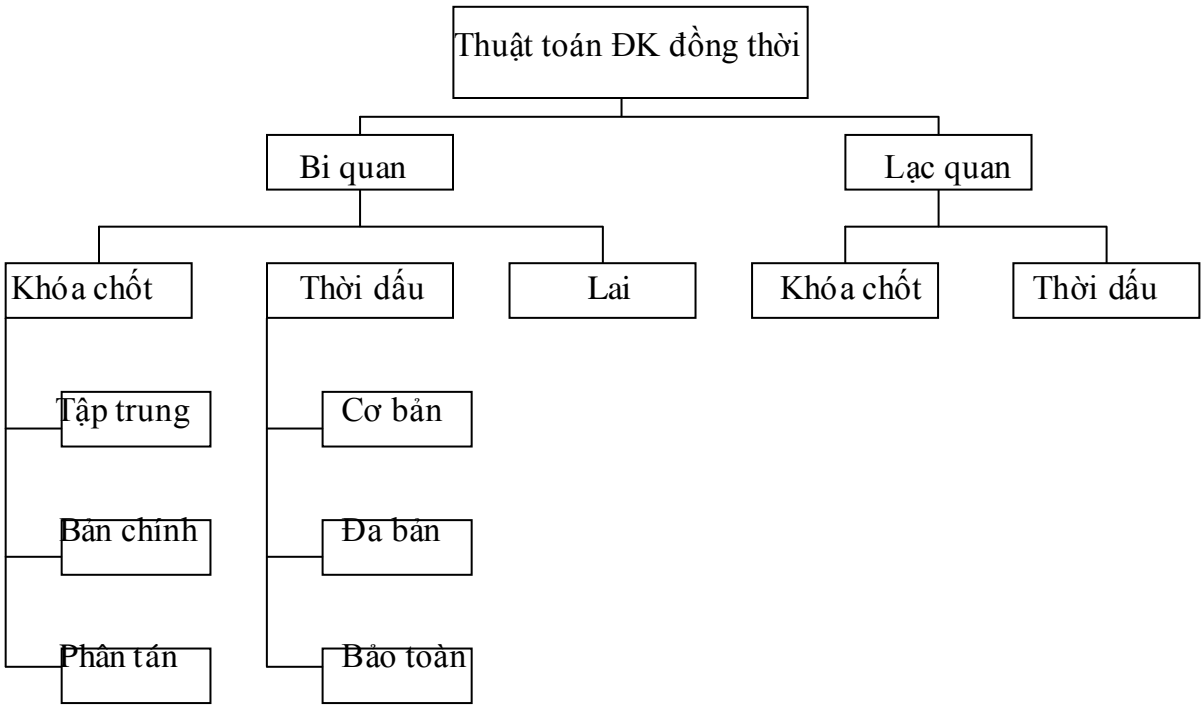
Vì thế có một số thuật toán cố gắng duy trì tính nhất quán tương hỗ mà không sử dụng đến nghi thức ROWA. Chúng đều dựa trên một tiền đề là chúng ta vẫn có thể tiếp tục tiến hành một thao tác, miễn là thao tác này có thể được xếp lịch tại một tập con các vị trí chiếm hơn phân nửa các vị trí có lưu các bản sao hoặc là tất cả các vị trí có thể đến được (nghĩa là còn dùng được). Vẫn có những nghi thức khác thực hiện các cập nhật trên một bản chính được chọn ra của mục dữ liệu nhân bản rồi lan truyền các cập nhật này cho những bản sao khác vào thời điểm thích hợp.

6.4.2 Phân loại các cơ chế điều khiển đồng thời

Có một số cách phân loại các phương pháp điều khiển đồng thời. Một tiêu chuẩn hiển nhiên là chế độ phân tán CSDL. Một số thuật toán đã được đề xuất đòi hỏi có một CSDL nhân bản hoàn toàn, còn một số khác có thể hoạt tác trên các CSDL phân hoạch hoặc nhân bản một phần. Các thuật toán điều khiển đồng thời cũng có thể được phân loại theo topo mạng, chẳng hạn như một mạng con phải có khả năng phát tán hoặc các thuật toán hoạt động trên các mạng hình sao hoặc các mạng kết vòng.

Tuy nhiên tiêu chuẩn phân loại thông dụng nhất là theo *nguyên thuỷ đồng bộ hóa* (synchronization primitive). Sự phân chia tương ứng đưa các thuật toán điều khiển đồng thời vào hai lớp: những thuật toán dựa trên các truy xuất độc quyền đến dữ liệu dùng chung (khóa chốt) và những thuật toán cố gắng sắp thứ tự hiện giao tác theo một tập qui tắc (nghi thức). Tuy nhiên các nguyên thủy này đều có thể dùng được dùng trong các thuật toán với hai quan điểm khác nhau: *quan điểm bi quan* (pessimistic view) cho rằng có nhiều giao tác sẽ tương tranh với nhau, còn *quan điểm lạc quan* (optimistic view) cho rằng không có quá nhiều giao tác tương tranh với nhau.

Vì vậy chúng ta sẽ xếp các cơ thể điều khiển đồng thời thành hai nhóm lớn: các phương pháp điều khiển đồng thời lạc quan và các phương pháp điều khiển đồng thời bi quan. Các thuật toán bi quan đồng bộ hóa việc thực hiện đồng thời của các giao tác trước khi thực hiện chúng, trong khi đó các thuật toán lạc quan để việc đồng bộ hóa các giao tác cho đến khi chúng kết thúc. Nhóm lạc quan gồm có các *thuật toán dựa theo khóa chốt* (locking-based algorithm), các thuật toán dựa theo thứ tự giao tác và các *thuật toán lai* (hybrid algorithm). Tương tự, nhóm lạc quan cũng có thể được phân loại thành các thuật toán dựa theo khóa và các thuật toán theo thứ tự thời gian. Phân loại này được trình bày trong hình 6.6.



Hình 6.6 Phân loại các thuật toán điều khiển đồng thời

Trong cách tiếp cận dùng khoá chốt, việc đồng bộ hóa giao tác có được bằng cách sử dụng các khoá chốt vật lý hoặc logic trên một phần CSDL. Kích thước của các phần này (thường được gọi là độ *mịn khóa*, locking granularity) là một vấn đề quan trọng. Tuy nhiên trong lúc này chúng ta sẽ bỏ qua nó và xem kích thước được chọn là *một đơn vị khóa* (lock unit). Lớp cơ chế này được chia nhỏ hơn nữa tùy theo vị trí thực hiện các hoạt động quản lý khóa:

Trong lối khóa tập quyền, một trong các vị trí của mạng được chỉ định làm vị trí chính, ở đó lưu trữ các bảng khóa cho toàn bộ CSDL và chịu trách nhiệm trao khóa cho các giao dịch.

Theo lối khóa bản chính thì ngược lại, một trong các bản sao (nếu có nhiều bản) của mỗi đơn vị khóa được chỉ định làm *bản chính* (primary copy), và đó chính là bản sẽ bị khóa khi giao tác truy xuất đến đơn vị đó. Ví dụ nếu đơn vị khóa x được nhân bản tại các vị trí 1,2, và 3, một trong những vị trí này (chẳng hạn 1) được chọn làm vị trí chính cho x. Tất cả mọi giao tác muốn truy xuất x sẽ nhận được một khóa của chúng tại vị trí 1 trước khi chúng có thể truy xuất được một bản sao của x. Nếu CSDL không được nhân bản

(nghĩa là mỗi đơn vị khóa chỉ có một bản duy nhất), các cơ chế khóa bản chính sẽ phân phối trách nhiệm quản lý khóa cho một số vị trí.

Theo lối khóa phi tập trung, nhiệm vụ quản lý khóa là của tất cả các vị trí trong mạng. Trong trường hợp này, thực hiện một giao tác có sự tham gia và điều phối của các bộ xếp lịch tại nhiều vị trí. Mỗi bộ xếp lịch cục bộ chịu trách nhiệm về các đơn vị khóa nằm cục bộ tại vị trí đó. Trong ví dụ trên, các thực thể muốn truy xuất x phải nhận được khóa tại tất cả ba vị trí.

Lớp cơ chế theo *thứ tự thời dấu* (timestamp ordering, viết tắt là TO) phải tổ chức thứ tự thực hiện của các giao tác nhằm duy trì được tính nhất quán lẫn tương hỗ giữa các vị trí (liên nhất quán). Việc xếp thứ tự này được duy trì bằng cách gán thời dấu cho cả giao tác lẫn mục dữ liệu được lưu trong CSDL. Những thuật toán này có thể thuộc loại *cơ bản* (basic TO), *đa phiên bản* (multiversion TO), hoặc *bảo toàn* (conservative TO).

Chúng ta cần chỉ ra rằng một số thuật toán dựa theo khóa cũng có thể dùng thời dấu, chủ yếu nhằm cải thiện hiệu quả và mức độ hoạt động đồng thời. Chúng ta gọi lớp thuật toán này là thuật toán lai. Chúng ta sẽ không thảo luận về chúng trong chương này bởi vì chúng chưa hề được cài đặt trong các hệ quản trị CSDL phân tán thương mại và các hệ thử nghiệm.

6.4.3 Các thuật toán điều khiển đồng thời bằng khóa chốt

Ý tưởng chính của việc điều khiển đồng thời bằng khóa chốt là bảo đảm dữ liệu dùng chung cho các thao tác tương tranh chỉ được truy xuất mỗi lần một giao dịch. Điều này được thực hiện bằng cách liên kết *một khóa chốt* (lock) với mỗi đơn vị khóa. Khóa này được giao tác đặt ra trước khi nó truy xuất và được điều chỉnh lại vào lúc nó hết sử dụng. Hiển nhiên là một đơn vị khóa không thể truy xuất được nếu đã bị khóa bởi một giao tác khác. Vì vậy yêu cầu khóa của một giao tác chỉ được trao nếu khóa đi kèm hiện không bị một giao tác khác giữ.

Bởi vì chúng ta quan tâm đến việc đồng hóa các thao tác tương tranh của các giao tác tương tranh nên có hai loại khóa chốt (thường được gọi là *thể thức khóa*, lock mode) được kèm với mỗi đơn vị khóa: *khóa đọc* (real lock, rl) và *khóa ghi* (write lock, wl). Một giao tác T_i đang muốn đọc một mục dữ liệu được chứa trong đơn vị khóa x sẽ nhận được

một khóa đọc trên x [ký hiệu là $rl_I(x)$] và cũng tương tự với các thao tác ghi. Thường thì chúng ta hay nói về *tính tương thích* (compatibility) của các thể thức khóa chốt. Hai thể thức khóa là *tương thích* nếu hai giao tác truy xuất đến cùng một mục dữ liệu có thể nhận được khóa trên mục dữ liệu đó cùng một lúc. Như Hình 6.7 cho thấy, các khóa đọc là tương thích với nhau, còn các khóa đọc-ghi hoặc ghi-ghi thì không. Vì vậy hai giao tác vẫn có thể đồng thời đọc cùng một mục.

	$rl_i(x)$	$wl_i(x)$
$rl_j(x)$	tương thích	không tương thích
$wl_j(x)$	không tương thích	không tương thích

Hình 6.7 Ma trận tương thích của các thể thức khóa

Các DBMS phân tán không chỉ phải quản lý các khóa mà còn phải có trách nhiệm xử lý khóa dùm cho giao dịch. Nói cách khác, người sử dụng không cần phải xác định khi nào phải khóa dữ liệu; DBMS phân tán sẽ lo liệu điều đó mỗi khi các giao tác đưa ra yêu cầu đọc hoặc ghi.

Trong các hệ thống dùng khóa chốt, *bộ xếp lịch* (scheduler) (xem hình 6.4) chính là *bộ quản lý khóa* (lock manager, LM). Bộ quản lý giao tác sẽ chuyển cho bộ quản lý khóa các thao tác CSDL (đọc hoặc ghi) và các thông tin kèm theo (như mục dữ liệu cần truy xuất, định danh của giao tác đưa ra yêu cầu). Sau đó bộ quản lý khóa sẽ kiểm tra xem đơn vị khóa có chứa mục dữ liệu đó đã bị khóa hay chưa. Nếu đã khóa, và nếu thể thức khóa đó không tương thích với thể thức của giao tác đang yêu cầu, thao tác sẽ bị hoãn lại. Ngược lại, khóa sẽ được đặt với thể thức mong muốn và thao tác này được chuyển cho bộ xử lý dữ liệu để truy xuất CSDL thực sự. Sau đó bộ quản lý giao tác được thông tin về các kết quả thực hiện. Việc kết thúc giao tác sẽ giải phóng các khóa của nó và làm khởi hoạt một giao tác khác đang đợi truy xuất mục dữ liệu này.

Thuật toán khóa chốt cơ bản nằm trong thuật toán 6.1. Hình 6.8 đưa ra các khai báo kiểu và các định nghĩa thủ tục được dùng trong các thuật toán của chương này. Chú ý trong thuật toán 6.1, chúng ta không quan tâm đến cách thực thi các thao tác ủy thác và hủy bỏ giao dịch.

Declare-type

Operation: một trong số Begin-Transaction, Read, Write, Abort, hoặc Commit

DataItem: một mục dữ liệu trong CSDL phân tán

TransactionId: một giá trị duy nhất được gán cho mỗi giao dịch.

DataVal: một giá trị có kiểu dữ liệu cơ bản (nghĩa là số nguyên, số thực, vân vân)

SiteId: một định danh duy nhất cho vị trí

Dbop: một bộ ba gồm {một phép toán trên CSDL của ứng dụng}

opn: Operation

data: DataItem

tid: TransactionId

Dpmsg: một bộ ba gồm

opn: Operation

tid: TransactionId

result: DataVal

Scmsg: một bộ ba gồm

opn: Operation

tid: TransactionId

result: DataVal

Transaction ←một bộ hai gồm

tid: TransactionId

body: thân giao tác như đã định nghĩa trong Chương 10

Message ← một chuỗi ký tự cần được truyền đi

OpSet: một tập các Dbop

SiteSet: một tập các SiteId

WAIT(msg: Message)

begin

{đợi cho đến khi có một thông báo đến}

end

Hình 6.8 Các định nghĩa chuẩn bị cho các thuật toán sắp tới

Thuật toán 6 .1. Bộ quản lý khóa cơ bản (Basic LM)

Declare-var

msg: Message

dop: Dbop

Op: Operation

x: DataItem

T: TransactionId

pm: Dpmsg

res: DataVal

SOP: OpSet

Begin

repeat

WAIT(msg)

case of msg

Dbop: // phép toán

begin

Op ←dop.opn

x ←dop.data

T ← dop.tid

case of Op

Begin_Transaction:

begin

gửi dop đến bộ xử lý dữ liệu

end

Read **or** Write:

begin

tìm đơn vị khóa lu sao cho x Í lu if lu chưa bị khóa or thể thức.khóa của lu tương thích với Op **then**

begin

đặt khóa trên lu ở thể thức thích hợp gửi dop đến bộ xử lý dữ liệu

```
            end
             else đưa dop vào một hàng đợi của lu
            end-if
          end
          Abort or Commit:
          begin
            gửi dop đến bộ xử lý dữ liệu
          end
        end-case
    Dpmsg: // Thông báo từ bộ xử lý dữ liệu          {trả lời của bộ xử lý dữ liệu}
    begin                                            {yêu cầu mở khoá}
                    Op ←pm.opn
                    res ←pm.result
                    T ←pm.tid
                    tìm đơn vị khóa lu sao cho x ⊆ lu, giải phóng khóa trên lu do T giữ
                    if không còn khóa nào trên lu and
          có những thao tác đang đợi khóa lu trong hàng đợi then
        begin
            SOP ← thao tác đầu tiên trong hàng đợi
            SOP ←SOP  {O½O là một thao tác trên hàng đợi có thể khóa lu ở thể thức
khóa tương thích với các thao tác hiện hành trong SOP}
          đặt các khóa trên lu cho các thao tác trong SOP
          for tất cả các phép toán trong SOP do
            gửi mỗi thao tác đến bộ xử lý dữ liệu
          end-for
                    end-if
                    end
            end-case
    until forever
end. (Basic LM)
```

Không may là, thuật toán khóa được cho trong Thuật toán 6.1 không đồng bộ hóa chính xác các thực thi giao dịch. Điều này là do khi tạo ra các lịch khả tuần tự, các thao

tác khóa và giải phóng khóa cũng cần phải được điều phối. Chúng ta minh họa nó bằng ví dụ sau.

Ví dụ 6.16

Xét hai giao tác sau đây: x = 50, y = 20

$T_1, T_2 \rightarrow x = 102, y = 38$
$T_2, T_1 \rightarrow x = 101, y = 39$
$S \rightarrow x = 102, y = 39$

T_1:	Read(x)	T_2:	Read(x)
	$x \leftarrow x + 1$		$x \leftarrow x * 2$
	Read(y)		Read(y)
	$Y \leftarrow y - 1$		$Y \leftarrow y * 2$
	Write(y)		Write(y)
	Commit		Commit

Dưới đây là một lịch hợp lệ được bộ quản lý khóa tạo ra khi sử dụng Thuật toán 6.1:

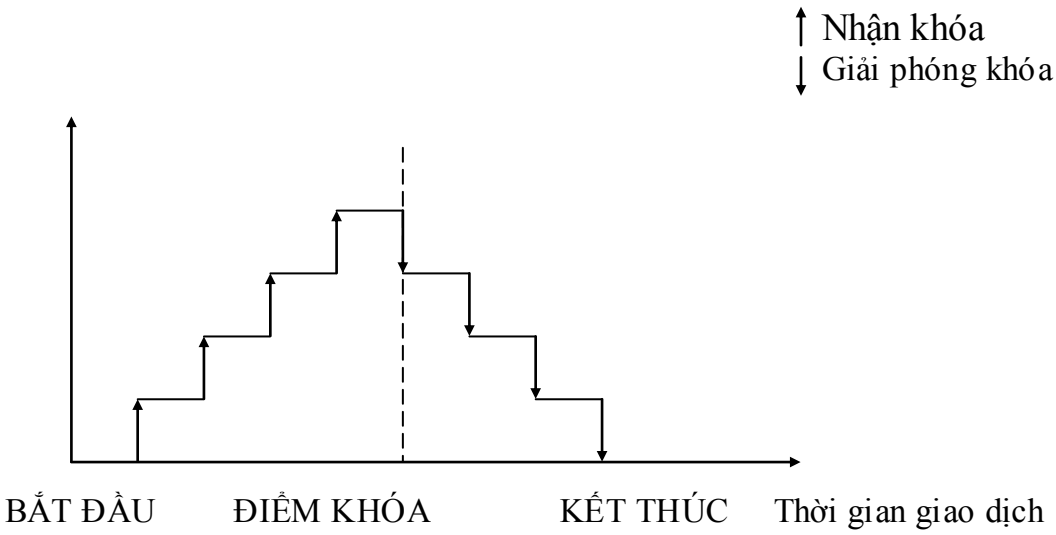
$S = \{wl_1(x), R_1(x), W_1(x), lr_1(x), wl_2(x), R_2(x), W_2(x), lr_2(x), wl_2(y),$
$R_2(y), W_2(y), lr_2(y), C_2, wl_1(y), R_1(y), W_1(y), lr_1(y), C_1\}$

Ở đây, $lr_i(z)$ biểu thị thao tác giải phóng khóa trên z đang được T_i giữ.

Chú ý rằng S không khả tuần tự. Chẳng hạn nếu trước lúc thực hiện các giao tác này, giá trị của x và y lần lượt là 50 và 20, chúng ta hy vọng rằng giá trị sau khi thực hiện tương ứng là 102 và 38 nếu T_1 thực hiện trước T_2, hoặc là 101 và 39 nếu T_2 thực hiện trước T_1. Tuy nhiên kết quả thực hiện S cho ra giá trị của x và y lần lượt là 102 và 39. Rõ ràng S không khả tuần tự.

Vấn đề của lịch S trong ví dụ 7 là, thuật toán khóa chốt đã giải phóng các khóa được một giao tác giữ (chẳng hạn T_1) ngay khi lệnh đi kèm (đọc hoặc ghi) được thực hiện, và đơn vị khóa (chẳng hạn x) không cần truy xuất nữa. Tuy nhiên bản thân giao tác đó đang khóa những mục khác (chẳng hạn Y) sau khi nó giải phóng khóa trên x. Dù rằng điều này dường như có lợi vì làm tăng khả năng hoạt động đồng thời, nó cho phép các giao tác đan xen với nhau, làm mất đi tính biệt lập và tính nguyên tử tổng thể. Đây chính là lập luận của *phương pháp khóa chốt hai pha* (two-phase locking, 2PL)

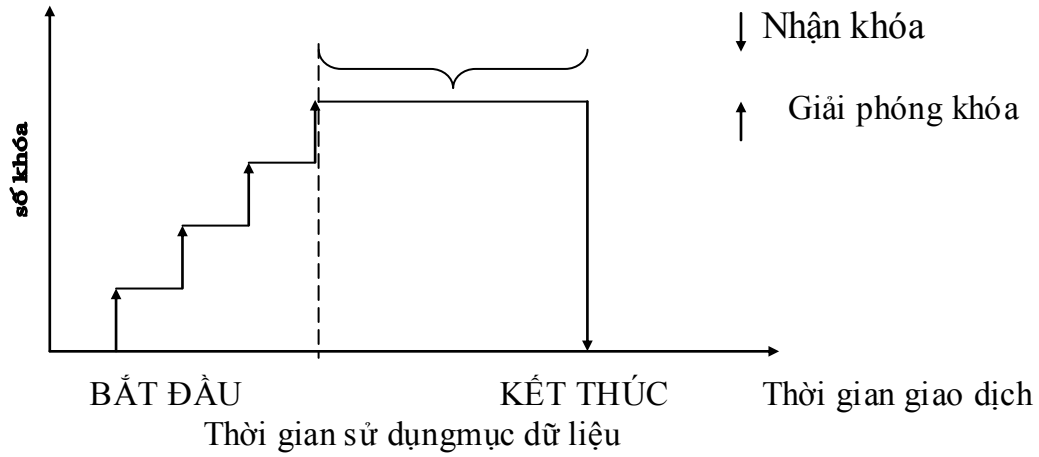
Qui tắc khóa hai pha chỉ đơn giản khẳng định rằng không có giao tác nào yêu cầu khóa sau khi nó đã giải phóng một trong các khóa của nó. Điều đó nói rằng một giao tác không được giải phóng khóa cho đến khi nó bảo đảm rằng không yêu cầu thêm khóa nữa. Các thuật toán 2PL thực hiện các giao tác qua hai pha. Mỗi giao tác có một *pha tăng trưởng* (growing phase), trong pha này nó nhận các khóa truy xuất các mục dữ liệu, và có một *pha thu hồi* (shrinking phase), là giai đoạn nó giải phóng các khóa (Hình 6.9). *Điểm khóa* (lockpoint) là thời điểm giao tác đã nhận được tất cả các khóa nhưng chưa bắt đầu giải phóng bất kỳ khóa nào. Vì thế điểm khóa xác định cuối pha tăng trưởng và khởi điểm pha thu hồi của một giao dịch. Một định lý nổi tiếng [Eswaran et al., 1976] khẳng định rằng mọi lịch tạo bởi một thuật toán điều khiển đồng thời tuân theo qui tắc 2PL đều khả tuần tự.



Hình 6.9 Biểu đồ khóa 2PL

Hình 6.9 chỉ ra rằng bộ quản lý khóa giải phóng các khóa ngay sau khi hoàn tất việc truy xuất. Điều này cho phép các giao tác đang đợi khóa tiếp tục tiến hành và nhận khóa, do vậy làm tăng hoạt động đồng thời. Tuy nhiên việc cài đặt gặp nhiều khó khăn bởi vì bộ quản lý khóa phải biết rằng giao tác đã nhận đủ tất cả mọi khóa và sẽ không cần khóa một mục nào nữa. Bộ quản lý khóa cũng phải biết rằng giao tác không còn cần truy xuất nữa mục dữ liệu đó nữa, vì thế khóa có thể được giải phóng. Cuối cùng nếu giao tác bị hủy bỏ sau khi giải phóng một khóa, nó có thể làm hủy bỏ luôn cả giao tác đã truy xuất các mục đã mở khóa. Hiện tượng này được gọi là *hủy bỏ dây chuyền* (cascading abort). Vì những khó khăn đó, phần lớn các bộ xếp lịch 2PL đều cài đặt một dạng khắt khe hơn có tên là *khóa chốt hai pha nghiêm ngặt* (strict two-phase locking) trong đó nó giải phóng

toàn bộ các khóa vào lúc giao tác kết thúc (ủy thác hoặc hủy bỏ). Biểu đồ khóa loại này được trình bày trong Hình 6.10



Hình 6.10 Biểu đồ khóa hai pha nghiêm ngặt

Bộ quản lý khóa 2PL nghiêm ngặt chỉ sửa lại một ít trong thuật toán 6.1. Thực sự chỉ cần sửa đổi phần xử lý các hồi đáp từ bộ xử lý dữ liệu nhằm bảo đảm rằng các khóa chỉ được giải phóng nếu thao tác là ủy thác hoặc hủy bỏ. Để cho đầy đủ, chúng tôi trình bày toàn bộ thuật toán 2PL nghiêm ngặt trong thuật toán 6.2. Thuật toán quản lý giao tác để xếp lịch theo 2PL được cho trong Thuật toán 6.3.

Thuật toán 6.2 S2PL-LM

```
declare-var
msg: Message
dop: Dbop
Op: Operation
x: DataItem
T: TransactionId
pm: Dpmsg
res: DataVal
SOP: OpSet
begin
        repeat
                WAIT(msg)
```

Case of msg

Dbop:

Begin

Op ←dop.opn

x ←dop. Data

T←dop.tid

case of Op

Begin_transaction:

begin

gửi dop cho bộ xử lý dữ liệu

end

Read **or** Write

begin

tìm đơn vị khóa lu sao cho x ⊆ lu

if lu chưa khóa **or** thể thức khóa của lu tương thích với Op **then**

begin

đặt khóa trên lu ở thể thức thích hợp

gửi dop đến bộ xử lý dữ liệu

end

else

đặt dop vào một hàng đợi cho lu

end-if

end

Abort **or** Commit:

begin

gửi dop đến bộ xử lý dữ liệu

end

end-case

Dpmsg:

begin

Op ← pm.opn

res ← pm.result

T←pm.tid

```
                    if Op = Abort or Op = Commit then
                  begin
        for mỗi đơn vị khóa ly bị khóa bởi T do
    begin
        giải phóng khóa trên lu do T giữ
        if không còn khóa nào trên lu and
        có các thao tác đang đợi trong hàng đợi cho lu then
                    begin
        SOP ← thao tác đầu tiên trên hàng đợi
        SOP ← SOP ∪ {O | O là một thao tác trên hàng đợi có thể khóa lu
            ở một thể thức tương thích với thao tác hiện tại trong SOP}
đặt các khóa trên lu cho các thao tác trong SOP
        for tất cả các thao tác trong SOP do
                                    gửi mỗi thao tác đến bộ xử lý dữ liệu
                    end-for
                end-if
            end-for
        end-if
end
end-case
until forever
end. {S2PL-LM}
```

Thuật toán 6.3 Bộ quản lý giao tác 2PL (2PL-TM)

```
Declare-var
msg: Messager
Op: Operation
x: DataItem
T: TransactionId
O: Dbop
sm: Scmsg
res: DataVal
SOP: OpSet
begin
```

```
        repeat
                WAIT(msg)
                case of msg
                  Dbop:
                begin
                    gửi O đến LM
  end
        Scmsg:
  begin
        Op ←sm.opn
        res ←sm.result
        T ← sm.tid
case of Op
        Read:
        begin
                trả res về cho ứng dụng của người sử dụng (nghĩa là
giao dịch)
        end
        Write:
        begin
                thông tin cho ứng dụng về việc hoàn tất hành động ghi
            trả res về cho ứng dụng
                end
        Commit:
        begin
                hủy vùng làm việc của T
                thông tin cho ứng dụng biết về việc hoàn tất thành công các
giao tác T
        end
        Abort:
        begin
                thông tin cho ứng dụng biết về việc hoàn tất hủy bỏ
giao tác T
                end
```

end-case

end

end-case

until forever

end. {2PL-TM}

Cần chú ý rằng mặc dù thuật toán 2PL cưỡng chế tính khả tuần tự tương tranh, nó không cho phép tất cả mọi lịch có tính khả tuần tự tương tranh. Xét thử lịch sau đây:

$$S = w_1(x)r_2(x)r_3(y)w_1(y)$$

S không được thuật toán 2PL cho phép dùng vì T_1 cần thu một khóa ghi trên y sau khi nó giải phóng khóa ghi của nó trên x. Tuy nhiên đây là lịch khả tuần tự theo thứ tự

$T_3 \leftarrow T_1 \leftarrow T_2$. Thứ tự khóa có thể được tận dụng để thiết kế các thuật toán khóa cho phép những lịch thuộc loại này.

Ý tưởng chính nằm ở chỗ trước tiên cần nhận xét rằng trong lý thuyết khả tuần tự, thứ tự tuần tự hóa các thao tác tương tranh cũng quan trọng như việc phát hiện tương tranh và điều này có thể được tận dụng khi định nghĩa các thể thức khóa. Hệ quả là ngoài các khóa đọc (dùng chung, shared) và khóa ghi (độc quyền, exclusive), chúng ta có thể định nghĩa một thể thức khóa thứ ba: *dùng chung có thứ tự* (ordered shared). Khóa dùng chung có thứ tự của một đối tượng x bởi các giao tác T_i và T_j mang ý nghĩa như sau: cho một lịch S có các khóa dùng chung có thứ tự giữa các thao tác $o \in {}_i T_i$ và $p \in Tj$, nếu T_i thu được khóa o trước khi T_j thu được khóa p thì o được thực thi trước p. Xét bảng tương thích giữa các khóa đọc và ghi đã cho trong hình 6.7. Nếu có thêm khóa dùng chung có thứ tự thì có tám biến thể của bảng này. Hình 6.7 chỉ là một trong số đó và hình 6.9 trình bày thêm hai biến thể nữa. ví dụ trong hình 6.11(a) có một mối liên hệ dùng chung có thứ tự giữa $rl_j(x)$ và $wl_i(x)$ [ký hiệu là $rl_j(x)$ Þ $wl_i(x)$] chỉ ra rằng T_i có thể thu được một khóa ghi trên x trong khi T_j giữ một khóa đọc trên x với điều kiện có một mối liên hệ dùng chung có thứ tự từ $rl_j(x)$ đến $wl_i(x)$. Tám bảng tương thích có thể được so sánh với nhau ứng với tính chất được phép của chúng (nghĩa là ứng với các lịch sinh ra nhờ chúng), tạo

ra một dàn các bảng sao cho bảng trong hình 6.7 là hạn chế nhất và bảng trong Hình 6.11(b) là tự do nhất.

	$rl_i(x)$	$wl_i(x)$
$rl_j(x)$	tương thích	không tương thích
$wl_j(x)$	dùng chung có thứ tự	không tương thích

(a)

	$rl_i(x)$	$wl_i(x)$
$rl_j(x)$	tương thích	dùng chung có thứ tự
$wl_j(x)$	dùng chung có thứ tự	dùng chung có thứ tự

(b)

Hình 6.11 Bảng tương thích có thể thức khóa dùng chung có thứ tự

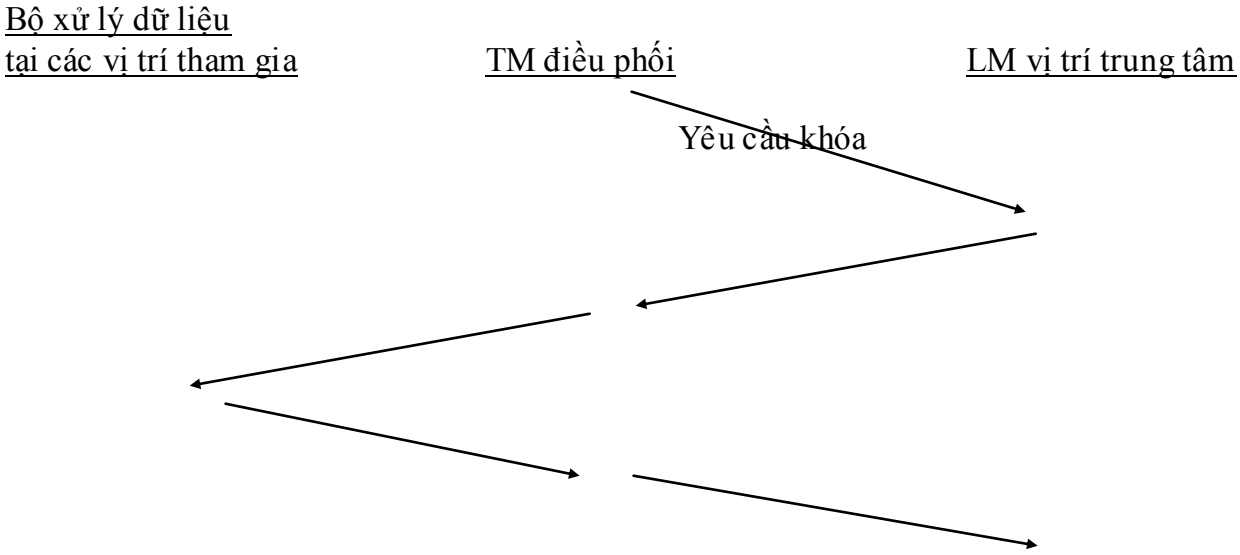
Trong ví dụ ở trên, khóa ghi dành cho T_i được gọi là đang được giữ (on hold) vì nó thu được sau khi T_j đã thu được khóa đọc trên x. Nghi thức khóa cưỡng chế một ma trận tương thích có chứa các thể thức khóa dùng chung có thứ tự hoàn toàn giống với 2PL, ngoại trừ là giao tác không giải phóng bất kỳ khóa nào khi một trong các khóa của chúng còn đang được giữ. Bằng không sẽ xảy ra các thứ tự tuần tự hóa lần quẩn.

6.4.4 Nghi thức 2PL tập quyền

Thuật toán 2PL được thảo luận trong phần trước có thể dễ dàng được mở rộng cho môi trường phân tán (nhân bản hoặc phân hoạch). Một cách để làm điều này là trao trách nhiệm quản lý khóa cho một vị trí duy nhất, nghĩa là chỉ có một vị trí là có bộ quản lý khóa. Các bộ quản lý giao tác ở các vị trí khác phải giao tiếp với nó chứ không phải với bộ quản lý khóa riêng của chúng. Cách tiếp cận này được gọi là thuật toán 2PL vị trí chính (primary site).

Truyền giao giữa các vị trí hiệp tác khi thực hiện một giao tác tuân theo thuật toán 2PL tập quyền (centralized 2PL hay C2PL) được trình bày trong hình 6.12. Truyền giao này xảy ra giữa bộ quản lý giao tác tại vị trí khởi đầu giao tác (được gọi là TM điều phối), bộ quản lý khóa tại vị trí trung tâm, và các bộ xử lý dữ liệu tại các vị trí có tham gia. Các vị trí tham gia là những vị trí có những thao tác xảy ra. Thứ tự các thông báo cũng được trình bày trong hình đó.

Một khác biệt quan trọng giữa thuật toán TM tập quyền và thuật toán TM của Thuật toán 6.3 đó là TM phân tán phải cài đặt nghi thức điều khiển bản sao nếu CSDL được nhân bản. Thuật toán C2PL-LM cũng khác với bộ quản lý khóa 2PL nghiêm ngặt ở một điểm. Bộ quản lý khóa trung tâm không gửi các thao tác đến các bộ xử lý dữ liệu tương ứng; việc này do TM điều phối thực hiện.



Hình 6.12. Cấu trúc truyền giao của 2PL tập quyền

Thuật toán quản lý giao tác 2PL tập quyền (C2PL-TM) được trình bày trong Thuật toán 6.4 và có đưa thêm những thay đổi này vào, còn thuật toán quản lý khóa 2PL tập quyền (C2PL-LM) được trình bày trong Thuật toán 6.5.

Thuật toán 6.4 Bộ quản lý giao tác 2PL tập quyền (C2PL-TM)

Declare-var

T: TransactionId

Op: Operation

x: DataItem

msg: Message

O: Dbop

pm: Dpmsg

res: DataVal

S: SiteSet

begin

repeat

```
        WAIT(msg)
        case of msg
          Dbop:
        begin
            Op←O.opn
             x ← O.data
             T←O.tid
case of Op
            Begin_transaction:
         begin
            S← ∅
         end
            Read:
         begin
               S ←S ∪ {vị trí lưu x và chi phí truy xuất đến x là nhỏ nhất}
               gửi O cho bộ quản lý khóa trung tâm
         end
            Write:
         begin
                                        S←S ∪ {S_i | x được lưu tại S_i }
                                        gửi O cho bộ quản lý khóa trung tâm
         end
            Abort or Commit
          begin
               gửi O cho bộ quản lý khóa trung tâm
            end
                        end-case
                    end
      Scmsg:        {yêu cầu khóa được trao khi các khóa được giải phóng}
    begin
        if yêu cầu khóa được trao then
          gửi O cho các bộ xử lý dữ liệu trong S
```

```
    else
        thông tin cho người dùng biết về việc kết thúc giao dịch
    end-if
                end
    Dpmsg:
begin
    Op ← pm.Opn
  res ←pm.result
  T ← pm.tid
   case of Op
    Read:
  begin
        trả res về cho ứng dụng người sử dụng (nghĩa là giao dịch)
  end
    Write:
  begin
        thông tin cho ứng dụng biết về việc hoàn tất thao tác ghi.
    end
                        Commit:
    begin
                                if tất cả các thành viên đều nhận được thông
    báo ủy thác then
        begin
            thông tin cho ứng dụng biết về việc hoàn tất thành công giao tác
gửi pm cho bộ quản lý khóa trung tâm
        else {đợi cho đến khi tất cả đều nhận được thông báo ủy thác}
            ghi nhận sự kiện thông báo ủy thác đến các vị trí
        end-if
                        end
                        Abort:
    begin
        thông tin cho ứng dụng biết về việc hủy bỏ giao tác T
        gửi pm đến bộ quản lý khóa trung tâm
```

```
            end
                    end-case
                end
            end-case
    until forever
end. {C2PL-TM}
```

Thuật toán 6.5 Bộ quản lý khóa 2PL tập quyền (C2PL-LM)

```
Declare-var
    msg: Message
dop: SingleOp
Op: Operation
x: DataItem
T: TransactionId
SOP: OpSet
  begin
        repeat
     WAIT(msg)  {Thông báo chỉ có thể do TM điều phối gửi đến}
     Op ←dop.opn
    x ← dop.data
    T ←dop.tid
     case of Op
                    Read or Write:
   begin
      tìm đơn vị khóa lu cho x ⊆ lu
         If lu chưa khóa or thể thức khóa của ly tương thích với Op then
     begin
        đặt khóa trên lu ở thể thức thích hợp
        msg ←"Khóa được trao cho thao tác dop"
        gửi msg đến TM điều phối của T
     end
         else
        đặt Op vào một hàng đợi cho lu
     end-if
```

```
            end
        Abort or Commit:
    begin
        for mỗi đơn vị khóa lu bị khóa T do
      begin
        giải phóng khóa trên lu do T giữ
        if còn những thao tác đang đợi lu trong hàng đợi then
      begin
          SOP ← thao tác đầu tiên (gọi O) từ hàng đợi
        SOP ←SOP  {O | O là một thao tác trên hàng đợi có thể khóa lu ở
              thể thức tương thích với các thao tác trong SOP}
          đặt các khóa trên lu cho các thao tác trog SOP
          for tất cả các thao tác O trong SOP do
        begin
            msg← "Khóa được trao cho thao tác O"
            gửi msg đến tất cả các TM điều phối
          end-for
                              end-if
      end-for
      msg ← "Các khóa của T đã giải phóng"
      gửi msg đến TM điều phối của T
    end
        end-case
  until forever
end. {C2PL-LM}
```

Một yếu điểm hay gặp của các thuật toán C2PL là có thể tạo ra một điểm ùn tắc quanh vị trí trung tâm. Hơn nữa hệ thống sẽ kém thích ứng (độ khả tín thấp) bởi vì sự cố hoặc tình trạng bất khả truy đến vị trí trung tâm có thể dẫn đến các sự cố hệ thống. Đã có những nghiên cứu chỉ ra rằng điểm ùn tắc thực sự sẽ hình thành khi tốc độ giao tác tăng lên, nhưng không đáng kể nếu tốc độ giao tác thấp. Thế nhưng người ta cũng thấy sự suy giảm hiệu năng khi tải trọng tăng cao trong các thuật toán dựa trên khóa chốt.

6.4.5 Thuật toán 2PL bản chính

Khóa chốt hai pha bản chính là sự mở rộng tầm thường của khóa chốt hai pha tập quyền với nổ lực giải quyết các vấn đề về hiệu năng đã được thảo luận ở trên. Về cơ bản, nó cài đặt các bộ quản lý khóa tại một số vị trí, trao trách nhiệm quản lý khóa trên một tập đơn vị khóa cho mỗi bộ quản lý. Sau đó bộ quản lý giao tác sẽ gửi các yêu cầu khóa và mở khóa đến các bộ quản lý khóa chịu trách nhiệm về đơn vị khóa đó. Thuật toán sẽ xử lý một bản của mỗi mục dữ liệu như bản chính của nó.

Chúng tôi không trình bày chi tiết thuật toán 2PL bản chính vì những thay đổi so với thuật toán 2PL tập quyền rất ít. Về cơ bản, thay đổi duy nhất là các nơi đặt bản chính phải được xác định cho mỗi mục trước khi gửi yêu cầu khóa hoặc mở khóa đến bộ quản lý khóa tại vị trí đó.

Thuật toán 2PL bản chính đã được đề xuất cho phiên bản phân tán thử nghiệm của hệ INGRES. Dù rằng nó đòi hỏi phải có một thư mục phức tạp tại mỗi vị trí, nó đã giảm được tải trọng cho vị trí trung tâm mà không phải trao đổi quá nhiều giữa các bộ quản lý giao tác và các bộ quản lý khóa. Về một nghĩa nào đó, đây là một bước trung gian giữa thuật toán 2PL tập quyền đã được thảo luận trong phần trước và thuật toán 2PL phân quyền sẽ được thảo luận trong phần kế tiếp.

6.4.6 Thuật toán 2PL phân quyền

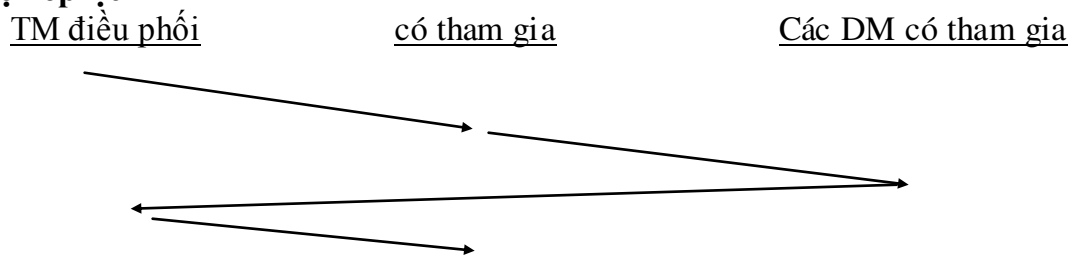
2PL *phân quyền* (distributed 2PL hay D2PL) mong muốn có sẵn các bộ quản lý khóa tại mỗi vị trí. Nếu CSDL không nhân bản, thuật toán 2PL phân quyền sẽ suy biến thành thuật toán 2PL bản chính. Nếu CSDL có nhân bản, giao tác sẽ cài đặt nghi thức điều khiển bản sao ROWA.

Truyền giao giữa các vị trí để thực hiện một giao tác theo nghi thức 2PL phân quyền được trình bày trong hình 6.13. Chú ý rằng hình 6.13 không trình bày việc áp dụng qui tắc ROWA.

Thuật toán quản lý giao tác 2PL phân quyền tương tự như 2PL-TM nhưng có hai sửa đổi chính. Các thông báo gửi đến bộ quản lý khóa của vị trí trung tâm trong C2PL-TM sẽ được gửi đến bộ quản lý khóa của tất cả các vị trí tham gia trong D2PL-TM. Khác biệt thứ hai là các thao tác không do TM điều phối chuyển đến các bộ xử lý dữ liệu

nhưng do các bộ quản lý khóa tham gia chuyển đi. Nghĩa là TM điều phối không chờ thông báo "yêu cầu khóa đã được trao". Một điểm khác về hình 6.13 là, các bộ xử lý dữ liệu sẽ gửi thông báo "kết thúc thao tác" đến TM điều phối. Chọn lựa khác là mỗi bộ xử lý dữ liệu sẽ gửi thông báo đó cho bộ quản lý khóa của riêng nó rồi bộ quản lý khóa sẽ giải phóng khóa và thông tin cho TM điều phối. Chúng ta đã giải quyết định mô tả theo cách thứ nhất vì nó dùng một thuật toán quản lý khóa giống với bộ quản lý khóa 2PL nghiêm ngặt đã được thảo luận và nó làm cho việc thảo luận các nghi thức ủy thác đơn giản hơn. Do những tương đồng này, chúng ta không đưa ra các thuật toán TM và LM phân quyền ở đây. Các thuật toán 2PL phân quyền được dùng trong System R*.

Các bộ xếp lịch



Hình 6.13 Cấu trúc truyền giao của 2PL phân quyền

CHƯƠNG 7 CÁC HỆ CƠ SỞ DỮ LIỆU

MỤC TIÊU

Chương này chỉ mang tính chất giơi thiệu các hệ cơ sở dữ liệu có quan hệ với cơ sở dữ liệu phân tán như hệ cơ sở dữ liệu song song và hệ cơ sở dữ liệu mobile.

7.1 Cơ sở dữ liệu song song

7.1.1 Giới thiệu

Vấn đề của hệ cơ sở dữ liệu quan hệ tập trung (đơn xử lý) :

- Khối lượng dữ liệu càng ngày càng lớn nên đòi hỏi không gian đĩa và bộ nhớ chính phải lớn ;
- Việc xuất /nhập dữ liệu bị thăt nút cổ chai (hoặc bộ nhớ truy cập bị thắt nút cổ chai) : do tốc độ (đĩa) << tốc độ (RAM) << tốc độ (bộ vi xử lý) :
- Dự báo :

Tốc độ của bộ vi xử lý tăng trưởng: 50% / năm

Dung lượng DRAM tăng trưởng: 4 lần mỗi ba năm

Dung lượng đĩa tăng trưởng : 2 lần trong mười năm qua.

Từ đó ta nhận thấy các hệ thống nhập/xuất dữ liệu của máy tính làm ảnh hưởng xấu đến tốc độ xử lý do bị nghẽn cổ chai. Nên giải pháp đặt ra nhằm tăng tốc độ xử lý là tìm cách tăng băng thong, phân tán dữ liệu và truy xuất dữ liệu xong xong. Điều này thì hệ quản trị cơ sở dữ liệu đơn xử lý không đủ khả năng đáp ứng do đó cần đến hệ cơ sở dữ liệu phân tán.

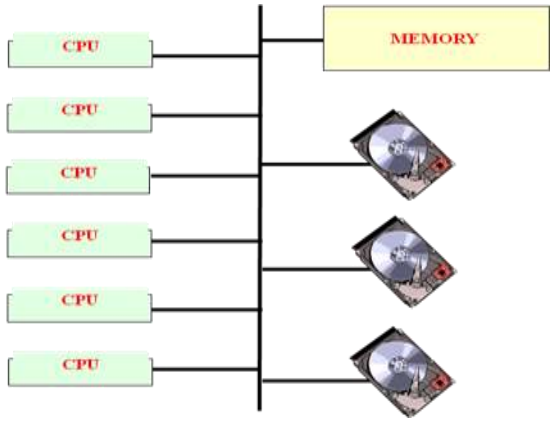
Hệ thống cơ sở dữ liệu song song tìm cách cải thiện hiệu suất thông qua việc song song hóa của các sự thực thi của các phép toán như tải dữ liệu, xây dựng các chỉ mục và đánh giá các câu truy vấn. Mặc dù dữ liệu có thể được lưu trữ bằng các phấn tán, sự phân tán dữ liệu chỉ được quy định bởi các tính chất hiệu suất truy xuất. Cơ sở dữ liệu song song cải thiện tốc độ xử lý và vào/ra bằng cách sử dụng nhiều CPU và đĩa song song. Các hệ quản trị cơ sở dữ liệu tập trung và client-server không đủ mạnh để xử lý các ứng dụng

như vậy. Trong xử lý song song, nhiều hoạt động được thực hiện đồng thời, trái với xử lý tuần tự, trong đó các bước tính toán được thực hiện tuần tự.

7.1.2 Kiến trúc hệ cơ sở dữ liệu song song

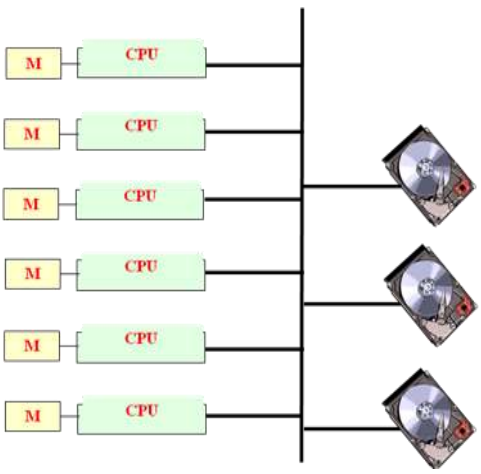
Kiến trúc hệ cơ sở dữ liệu phân tán có thể tạm chia thành ba loại:

* ***Kiến trúc chia sẻ bộ nhớ*** : nhiều bộ xử lý chia sẻ không gian bộ nhớ chính, cũng như bộ lưu trữ bền vừng (ví dụ các ổ đĩa cứng) (hình 7.1).



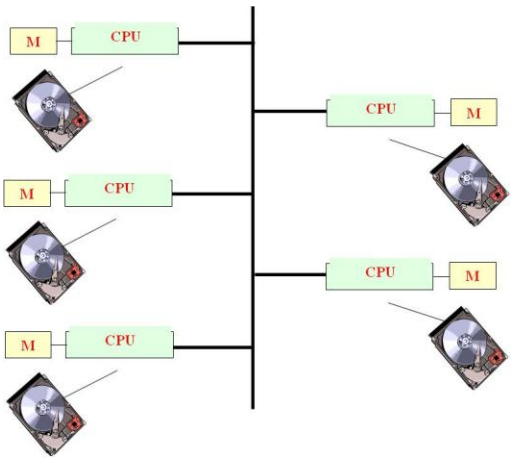
Hình 7.1 Kiến trúc chia sẻ bộ nhớ của hệ cơ sở dữ liệu song song

* ***Kiến trúc chia sẻ đĩa*** : nơi mỗi nút có bộ nhớ riêng của chính nó, nhưng tất cả các nút chia sẻ bộ lưu trữ bền vững, thường là một bộ lưu trữ mạng (hình 7.2). Trong thực tế, mỗi nút thường cũng có nhiều bộ xử lý.



Hình 7.2 Kiến trúc chia sẻ đĩa cứng của hệ cơ sở dữ liệu song song

* ***Kiến trúc không chia sẻ***: trong đó mỗi nút có lưu trữ riêng của mình cũng như bộ nhớ chính (hình 7.3).



Hình 7.3 Kiến trúc không chia sẻ tài nguyên của hệ cơ sở dữ liệu song song

7.1.3 Lợi ích của hệ cơ sở dữ liệu song song

- Cải thiện Thời gian đáp ứng: nó có thể xử lý một các truy vấn và giao dịch song song với nhau.

- Cải thiện hiệu suất Có thể xử lý các tác vụ trong một truy vấn hay giao dịch song song.

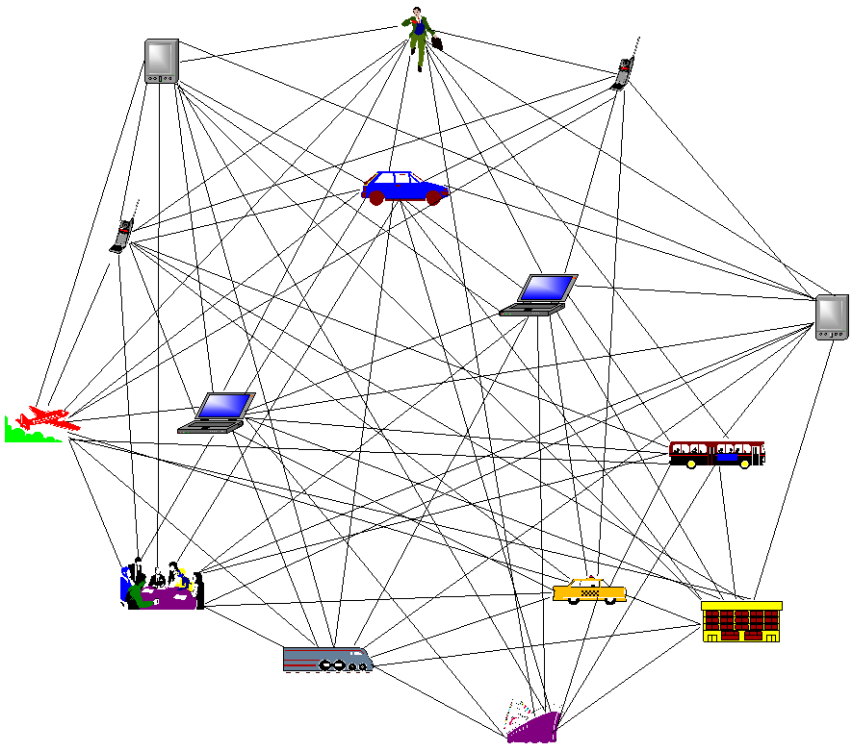
7.2 Hệ cơ sở dữ liệu mobile

7.2.1 Giới thiệu

Tìm kiếm thông tin như nhạc MP3 đối với người dùng với các thiết bị di động như điện thoại di động, PDA (Personal Digital Assistant) đã trở thành phổ biến trong đời sống hoạt động hàng ngày. Hệ thống chỉ dẫn đường trong xe ôtô hiện nay là một phụ kiện chuẩn như hệ thống âm nhạc. Những « món đồ chơi » khá hữu ích và thân thiện bởi vì họ có thể lấy thông tin mong muốn từ cơ sở dữ liệu từ bất cứ đâu thông qua các kênh không dây. Tuy nhiên, nó có một hạn chế lớn là các luồng thông tin trong các hệ thống này chỉ đi từ các máy chủ đến người dùng. Điều này hạn chế không cho phép người dùng truy vấn hoặc thao tác cơ sở dữ liệu có thể được nằm bất cứ nơi nào trên thế giới. Do đó, người dùng chỉ có sử dụng được những gì mà máy chủ gửi cho họ, có thể không phải luôn luôn được chính xác hoặc cập nhật. Trong thuật ngữ cơ sở dữ liệu hệ thống này không có khả năng quản lý hoạt động giao tác.

Các nhà nghiên cứu, và các tổ chức thương mại có chung một mong muốn xây dựng một hệ thống quản lý thông tin trên một nền tảng di động mà có khả năng cung cấp đầy đủ chức năng quản lý giao dịch và chức năng của hệ cơ sở dữ liệu từ bất cứ nơi nào và bất cứ lúc nào.

Hình 7.4 minh họa về một không gian kết nối thông tin đầy đủ với mọi người ngày hôm nay. Mỗi đối tượng của thế giới thực với một số chức năng được kết nối đối tượng khác thông qua liên kết không dây. Ví dụ, một ngân hàng hoặc một người được kết nối với hội nghị, xe buýt với các liên kết không dây hai chiều. Do đó bất cứ lúc nào một người hoặc một ngân hàng có thể có thông tin đầy đủ về tất cả các đối tượng khác. Liên kết không dây như vậy đã trở nên thiết yếu đối với một xã hội di động và năng động cao. Hãy xem xét trường hợp của một bậc cha mẹ đi làm việc còn các con cái của họ đi học trong các trường học khác nhau. Mỗi bậc cha mẹ trong gia đình muốn có thể nắm bắt được tình hình con em của họ tiếp cận với họ tại thời điểm cần thiết. Tương tự như vậy, chủ tịch của một công ty muốn có thông tin đầy đủ về tất cả các hoạt động của các công ty của mình để quản lý nó một cách hiệu quả.



Hình 7.4 Không gian thông tin được kết nối đầy đủ

7.2.2 Định nghĩa hệ cơ sở dữ liệu mobile

Một hệ cơ sở dữ liệu mobile là một hệ thống có cấu trúc và chức năng như sau :

- Hệ thống phân tán với kết nối điện thoại di động,
- đầy đủ chức năng hệ cơ sở dữ liệu,
- hoàn thành di động trong không gian,
- được xây dựng trên nền tảng PCS/ GSM ,
- có khả năng giao tiếp không dây và có dây.

Trong khái niệm vệ hệ cơ sở dữ liệu mobile ở trên chúng ta cần làm rõ một số khái niệm sau :

Kết nối di động là gì ?

Một chế độ trong đó một máy khách (thiết bị di động) hoặc một máy chủ có thể thiết lập liên lạc với nhau bất cứ khi nào cần thiết (không liên tục). Kết nối liên tục là một trường hợp đặc biệt của kết nối di động.

Kết nối không liên tục (intermittent) là gì ?

Một nút trong đó chỉ có những máy khách (thiết bị di động) có thể thiết lập các thông tin liên lạc bất cứ khi nào cần thiết với máy chủ nhưng máy chủ không thể làm như vậy.

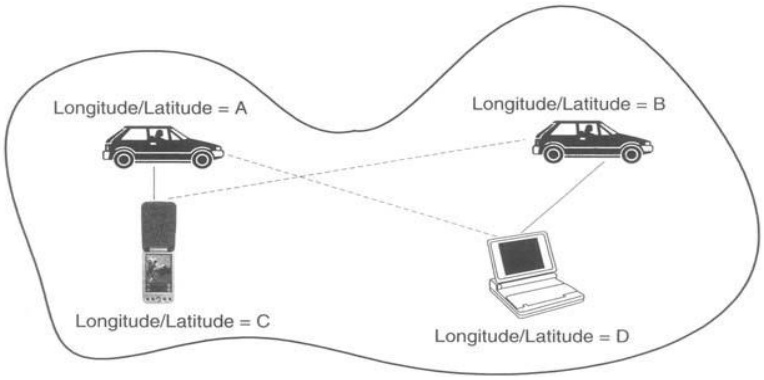
7.2.3 Các kiểu di động

Một framework di động bao gồm các thành phần có dây và không dây và người sử dụng. Thành phần không dây là các thiết bị di động đầu cuối và sự di động cá nhân để loại trừ một số hạn chế không gian và thời gian của việc xử lý dữ liệu.

Thiết bị đầu cuối di động: cho phép một đơn vị điện thoại di động (máy tính xách tay, điện thoại di động, PDA, v.v…) truy cập đến các dịch vụ mong muốn từ bất kỳ vị trí nào, lúc chuyển động hoặc tĩnh, không phân biệt người đang sử dụng thiết bị. Ví dụ một điện thoại di động có thể được sử dụng bởi chủ nhân của nó và nó cũng có thể được vay bởi ai khác để sử dụng. Một người ở vị trí C sử dụng các đơn vị điện thoại di động để giao tiếp với người lái xe ở vị trí A. Ông vẫn có thể thiết lập giao tiếp với người lái xe từ một vị trí mới mà không phân biệt của sự chuyển động của xe từ A đến B.

Tính di động cá nhân: Trong môi trường di động, thiết bị di động đầu cuối của thiết bị đầu cuối được hỗ trợ nghĩa là cùng một thiết bị đầu cuối có thể được sử dụng để kết nối với các bên khác từ bất cứ đâu bởi bất kỳ người sử dụng. Trong tính di động cá nhân khả năng này được cung cấp cho một con người.

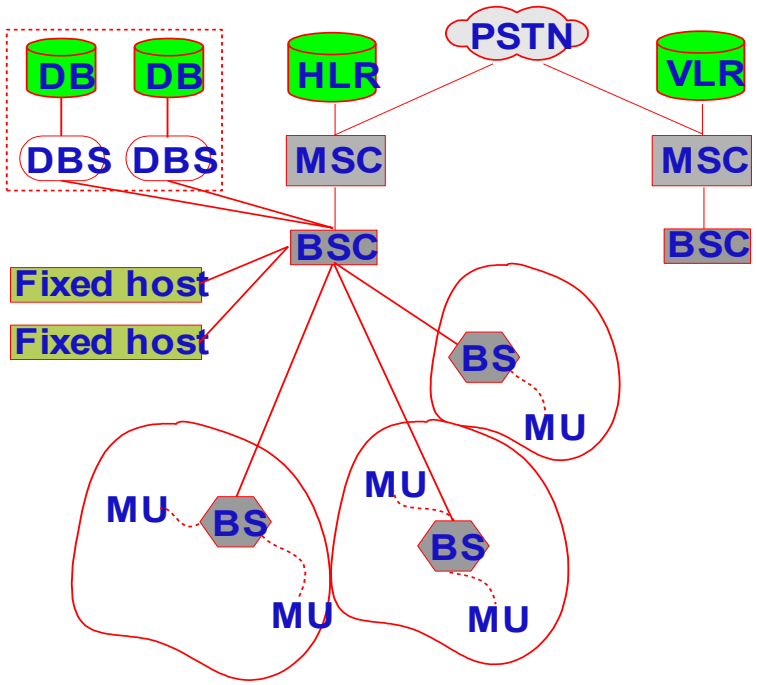
Như vậy, một người dùng có thể sử dụng bất kỳ thiết bị truyền thông cho việc thiết lập giao tiếp với bên kia. Việc này yêu cầu một chương trình nhận dạng để xác minh người có nhu cầu giao tiếp với mình. Hình 7.5 minh họa khái niệm về tính di động cá nhân. Một người ở vị trí C giao tiếp với xe tại địa điểm A bằng cách sử dụng PDA của mình, và từ vị trí D ông cũng có thể giao tiếp với chiếc xe tại địa điểm A bằng cách sử dụng máy tính xách tay của mình. Hiện nay, tính di động cá nhân sẵn sàng thông qua web. Một người dùng có thể đăng nhập vào các trang web từ các máy khác nhau đặt tại nhiều nơi khác nhau và truy cập thư điện tử của mình. Hệ thống điện thoại di động mở rộng cơ sở này để người dùng có thể sử dụng bất kỳ thiết bị di động để truy cập được internet. Trong tính di động cá nhân mỗi người phải được xác định duy nhất, và một cách để làm điều này là thông qua một mã số duy nhất.



Hình 7.5 Tính di động cá nhân

7.2.4 Kiến trúc hệ cơ sở dữ liệu mobile

Một hệ thống cơ sở dữ liệu mobile (MDS) cung cấp đầy đủ cơ sở dữ liệu và chức năng truyền thông di động. Nó cho phép một người sử dụng điện thoại di động để giao dịch từ bất cứ nơi nào và bất cứ lúc nào và đảm bảo tính nhất quán. Trong trường hợp của bất cứ loại sự cố lỗi nào (giao dịch, hệ thống, và các phương tiện truyền thông), MDS đảm bảo cơ sở dữ liệu được phục hồi. Các kiến trúc tham chiếu được hiển thị trong hình 7.6 cung cấp các tính chất cần thiết sau đây.



Hình 7.6 Kiến trúc của hệ cơ sở dữ liệu mobile

Tính di động địa lý: Khách hàng có thể di chuyển trong địa lý không gian mà không ảnh hưởng đến khả năng xử lý của họ và kết nối liên tục.

Kết nối và hủy kết nối: khách hàng có thể ngắt kết nối và kết nối lại với bất kỳ máy chủ bất kỳ lúc nào.

Khả năng xử lý dữ liệu: Khách hàng có một số quyền trong khi đó các máy chủ có đủ năng lực xử lý cơ sở dữ liệu.

Truyền thông không dây: Một khách hàng có thể giao tiếp với máy chủ và với bất kỳ khách hàng khác thông qua một mạng không dây.

Tính trong suốt: Việc xử lý dữ liệu được thực hiện thông qua kiến trúc di động, các chức năng xử lý dữ liệu của khách hàng không ảnh hưởng đến các trạm truyền thông. Khả năng mở rộng: Bất cứ lúc nào, khách hàng có thể được vào, hoặc một khách hàng hiện tại có thể được ra khỏi mạng.

MDS là một hệ thống client/server đa cơ sở dữ liệu phân tán dựa trên PCS hay GSM. Có một số khác biệt trong kiến trúc GSM và PCS, tuy nhiên, chúng không ảnh hưởng đến MDS. Chức năng cơ sở dữ liệu được cung cấp bởi một tập hợp các DBSs (máy chủ cơ sở dữ liệu) mà không ảnh hưởng đến bất kỳ khía cạnh của các mạng di động chung. Các thành phần của các hệ thống này trên thực tế là các máy tính chuyên dùng chịu trách

nhiệm kết nối người dùng với hệ thống. Các kiến trúc tham khảo của MDS được mô tả trong cho các máy tính có mục đích chung như máy tính cá nhân, máy trạm, PDA, di động điện thoại, v.v…

Trong MDS một tập hợp các máy tính có mục đích chung được kết nối với nhau thông qua một mạng tốc độ cao. Các máy tính được phân thành host cố định host (Fixed Host FH) và các trạm cơ sở (Base Station BS) hoặc các trạm hỗ trợ điện thoại di động (Mobile Support Station MSS).Các FH không được trang bị thu phát, do đó chúng không giao tiếp với các đơn vị di động. Một hoặc nhiều BS được kết nối với một trạm điều khiển (Base Station Control BSC) mà điều phối các hoạt động của các BS bằng cách sử dụng của chương trình phần mềm được lưu trữ trong khi được điều khiển bởi các MSC (Mobile Switching Center).

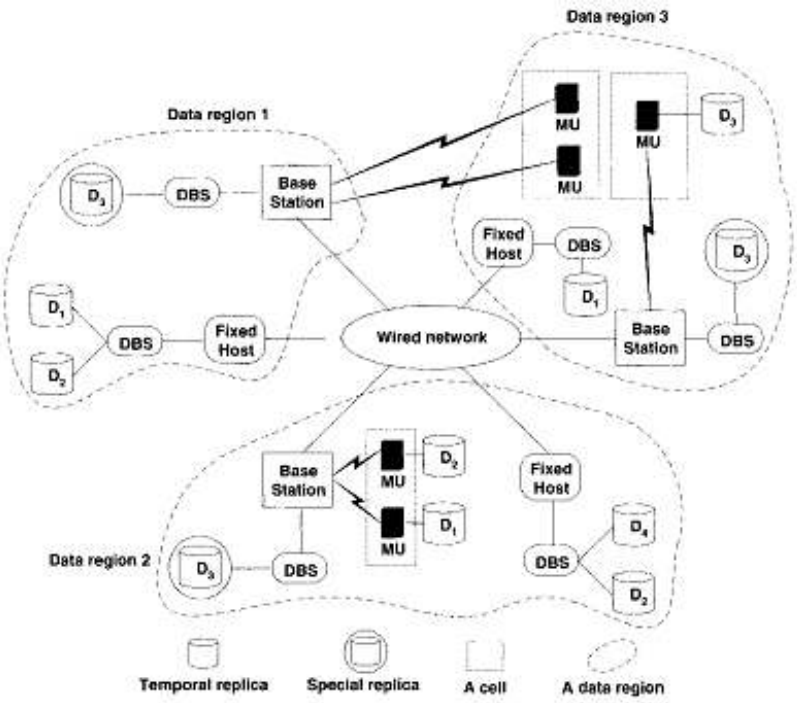
Sự phối hợp với các máy chủ cơ sở dữ liệu được kết hợp trong BS. Những điện thoại di động được gọi là host di động (Mobile Host MH) hoặc đơn vị di động (Mobile Unit MU). Các BS được trang bị thu phát và giao tiếp với các MU thông qua các kênh không dây. Mỗi BS phục vụ một tế bào có kích thước phụ thuộc vào năng lượng của BS của nó.

Để phối hợp đầy đủ các chức năng của hệ cơ sở dữ liệu, điều cần thiết là kết nối máy chủ (DBSs) chứa cơ sở dữ liệu với PCS hay GSM. Nó có thể được cài đặt (a) tại các trạm BS hoặc (b) tại các host HS. Tuy nhiên, một số vấn đề với cách cài đặt này. Một BS hoặc FH là thiết bị chuyển mạch, và chúng có nhiệm vụ cụ thể để thực hiện, mà không bao gồm các chức năng của cơ sở dữ liệu. Để thêm các chức năng cơ sở dữ liệu, cấu trúc toàn bộ của một BS (phần cứng và phần mềm) có thể phải được sửa đổi, đó là điều không thể chấp nhận từ quan điểm truyền thông di động. Hơn nữa, cách thiết lập này sẽ không module hoa được và cũng không có khả năng mở rộng. Hậu quả là, bất kỳ một thay đổi nào trong các thành phần cơ sở dữ liệu sẽ ảnh hưởng đến việc truyên thông dữ liệu và tiếng nói.

Bởi những lý do này, các DBSs được kết nối với hệ thống di động thông qua đường dây là các node riêng biệt, như minh họa trong hình 7.6. Mỗi DBS có thể truy cập được bởi bất kỳ BS hoặc FH, và DBSs mới có thể được kết nối và những DBSs cũ có thể được lấy ra khỏi mạng thông tin di động mà không ảnh hưởng gì.

Một DBS chỉ giao tiếp với một MU thông qua BS. Một người sử dụng điện thoại di động kết nối trung bình 2 đến 4 giờ trong một ngày, và ở tất cả các lần khác, nó phải tiết kiệm năng lượng pin. Để bảo tồn năng lượng, một đơn vị điện thoại di động có thể được chuyển sang trạng thái : (a) tắt chế độ hỗ trợ (không chủ động lắng nghe các BS) hoặc (b) chế độ nhàn rỗi (chế độ liều - không giao tiếp nhưng liên tục nghe các BS) hoặc (c) chế độ hoạt động (giao tiếp với bên kia, xử lý dữ liệu,v.v…).

Một MDS có thể có nhiều cơ sở dữ liệu, và các cơ sở dữ liệu này có thể được phân tán hoàn toàn hoặc một phần hoặc nhân bản hoàn toàn. An MDS có thể là một liên đoàn hoặc một hệ thống đa cơ sở dữ liệu.



Hình 7.7 Các kiểu nhân bản

Hình 7. 7 minh họa cách cơ sở dữ liệu được phân phối giữa các DBSs và MUs. Dữ liệu độc lập vị trí, ví dụ : D1, D2 và D4, có thể nhân bản ở tất cả các vùng và có cùng giá trị. Đây là một trong những lý do mà các bản sao tạm thời có thể được xử lý bằng cách sử dụng qui tắc đọc một - ghi tất cả các bản sao của cơ chế khóa hai pha điều khiển đồng thời phân tán. Dữ liệu phụ thuộc vị trí như tại D3 cũng có thể được nhân bản đến tất cả các vùng nhưng mỗi vùng phải có giá trị khác nhau.

Ba kiểu nhân bản cơ bản của cơ sở dữ liệu được minh họa trong hình 7.7: (a) "không sao chép," (b) "nhân phân tán truyền thống" (nhân bản tạm thời), và (c) " nhân bản phụ thuộc

vị trí" (nhân bản không gian). Đối tượng dữ liệu D_4 không nhân bản, và các đối tượng dữ liệu D_1 và D_2 là nhân bản truyền thống mà được sao chép đến vùng dữ liệu 1 và 2. Các phân tán bản sao là các bản sao của nhau, chúng có thể có giá trị khác nhau tạm thời, nhưng chỉ có một một trong những giá trị đúng.

Các dữ liệu phụ thuộc vị trí D_3 có nhiều bản sao và nhiều giá trị đúng. Các giá trị đúng được xác định theo vị trí. Nó được nhân bản ở cả ba khu vực. Mỗi vùng có một giá trị chính xác cho D_3, và nó cũng có thể có bản sao tạm thời trong khu vực đó. Giá trị của D_3 mà tồn tại ở các BS trong vùng dữ liệu 3 được nhân rộng ở MU, nhưng nó có giá trị như là ở DBS.

	Không nhân bản	**Nhân bản truyền thống**	**Nhân bản không gian**
Bản sao	1	Nhiều	Nhiều
Giá trị đúng	1	1/ đơn vị thời gian	1/đv thời gian và vị trí
Kiến trúc	Tập trung	Phân tán	Mobile
Tính di động	Không	Không	Có

Bảng 7.1 Sự khác biệt giữa ba kiểu nhân bản

Bảng 7.1 tóm tắt sự khác biệt giữa ba loại nhân bản dữ liệu. Chú ý rằng bản sao tại một bộ nhớ cache ở một MU là bản sao thực sự theo thời gian, tức là, có thể có bản sao tạm thời của các bản sao không gian (như Object O_3 trong hình 7.7). Trong một khu vực nhất định có thể có bản sao tạm thời của dữ liệu phụ thuộc vị trí.

TÀI LIỆU THAM KHẢO

1. M. Tamer Ozsu, Patrick Valduriez, *Nguyên lý các hệ cơ sở dữ liệu phân tán*, tập 1, 1999, Nhà xuất bản Thống kê.

2. Nguyễn Trung Trực, *Cơ sở dữ liệu phân bố*, 2003, Đại học Bách Khoa TP HCM.

3. Ceri, S. and Pelagatti, G., *Distributed Databases Principles and Systems*, 1984, McGraw-Hill, Inc.

4. Vijay Kumar, Mobile Database Systems, 2006, by John Wiley & Sons, Inc.

MỘT SỐ ĐỀ THI THAM KHẢO

ĐỀ SỐ 1

Ngân hàng ABC gồm có chi nhánh đặt tại ba thành phố lớn : TP Hồ Chí Minh, Cần Thơ và Hà Nội. Lược đồ ChiNhanh được mô tả như sau :

ChiNhanh(MaCN, tenCN, ThanhPho)

Mỗi chi nhánh có khách hàng của mình. Lược đồ KhachHang được mô tả như sau :

KhachHang(SoCMND, tênKH, điachi, nghenghiep, noicongtac, sodienthoai, MaCN)

Một tài khoản được mở ở một chi nhánh. Lược đồ TaiKhoan được mô tả như sau :

TaiKhoan(SoTK, ngaymoTK, ngayDongTK, MaCN)

Một tài khoản có thể thuộc về một hay nhiều khách hàng. Một khách hàng cũng có thể có nhiều tài khoản. Lược đồ TKKH được mô tả như sau :

TKKH(SoTK, SoCMND, tien)

Ngân hàng ABC muốn xây dựng một hệ thống thông tin quản lý TKKH có cơ sở dữ liệu phân tán tại ba thành phố TP Hồ Chí Minh, Cần Thơ và Hà Nội.

Câu hỏi

1. (4 điểm) Hãy thiết cơ sở dữ liệu phân tán cho hệ thống thông tin quản lý TKKH.
    a. Thiết kế các phân mảnh ngang nguyên thủy cho lược đồ ChiNhanh
    b. Thiết kế các phân mảnh ngang dẫn xuất của lược đồ KhachHang, TaiKhoan ứng với các phân mảnh theo chi nhánh tương ứng ở câu a.
    c. Thiết kế các phân mảnh ngang dẫn xuất ứng của lược đồ TKKH ứng với các phân mảnh theo TaiKhoan ở câu b.
    d. Gia sử ta có hai ứng dụng chuyên tìm thông tin như sau :

    Ứng dụng 1 : Tìm tên khách hàng và địa chỉ của khách hàng theo chi nhánh.

    Ứng dụng 2 : Tìm tên khách hàng, nghề nghiệp và số điện thoại theo chi nhánh.

    Hãy thiết kế phân mảnh dọc hỗn hợp tương ứng với các ứng dụng từ các phân mảnh Khách hàng ở câu b.

2. (1 điểm) Hãy mô tả quá trình cấp phất các phân mảnh tại ba sites TP Hồ Chí Minh, Cần Thơ và Hà Nội.
3. (1 điểm) Hãy vẽ cây phân mảnh của lược đồ KhachHang ở câu 1 (b và d)
4. (1 điểm) Dựa trên câu 1, viết câu truy vấn sau ở mức độ trong suốt ánh xạ cục bộ :

Biết khách hàng có số CMND là 120934875 của một chi nhánh tại Cần Thơ. Viết câu truy vấn đổi khách hàng số CMND là 120934875 về chi nhánh có tên là « ABC Đa Kao » tại TP Hồ Chí Minh. Nếu khách hàng này có tham gia tài khoản tại Cần Thơ thì xóa toàn bộ các bộ chứa khách hàng này.

5. (3 điểm) Dựa trên câu 1, trình bày quá trình chuyên câu truy vấn toàn cục thành câu truy vấn dựa trên các phân mảnh cho câu truy vấn sau :

Liệt kê các tên khách hàng, và địa chỉ và số điện thoại có số tài khoản là « 123456789087 5432 ». Biết rằng tài khoản thuộc một chi nhánh tại TP Hồ Chí Minh.

   a. Viết câu lệnh SQL dưới dạng lược đồ toàn cục.
   b. Chuyển câu SQL ở câu a thành cây truy vân dưới dạng đại số quan hệ.
   c. Chuyển cây truy vấn ở câu b dưới dạng cây truy vấn phân mảnh tối giản.

ĐỀ SỐ 2

Bài 1

Cho cơ sở dữ liệu về quản lý dự án như sau

Employee(ENO, ENAME, TITLE)
Assignment(ENO, JNO, RESP, DUR)
Project(JNO, JNAME, BUDJET, LOC)
Salary(TITLE, SAL)

Employee

Assignment

ENO	JNO	RESP	DUR
E1	J1	Manager	12
E2	J1	Analyst	24
E2	J2	Analyst	6
E3	J3	Consultant	10
E3	J4	Engineer	48
E4	J2	Programmer	18
E5	J2	Manager	24
E6	J4	Manager	48
E7	J3	Engineer	36
E8	J3	Manager	40

ENO	ENAME	TITLE
E1	J. DOE	Elect. Eng.
E2	M. SMITH	Sys. Anal.
E3	A. LEE	Mech. Eng.
E4	J. MILLER	Programmer
E4	B. CASEY	Sys. Anal.
E5	L. CHU	Elect. Eng.
E6	R. DAVIS	Mech. Eng.
E7	J. JONES	Sys. Anal.

Project

JNO	JNAME	BUDJET	LOC
J1	Instrumentation	150000	Montreal
J2	Database Develop.	135000	New York
J3	CAD/CAM	250000	Paris
J4	Maintenance	310000	Paris

Salary

TITLE	SAL
Elect. Eng.	29000
Sys. Anal	34000
Mech. Eng.	27000
Programmer	24000

Cho các vị từ đơn giản sau :
P1 : Loc = "Montreal"
P2: Loc = "New York"
P3: Loc = "Paris"
P4: Budget ≤ 30000
P5: Budget > 30000

Câu hỏi

1. Thiết kế các phân mảnh ngang nguyên thủy của lược đồ quan hệ toàn cục Project đáp ứng các vị từ đơn giản trên. Cho biết các quan hệ (bảng) phân mảnh kết quả .

2. Thiết kế các phân mảnh ngang dẫn xuất của lược đồ Employee đáp ứng hai vị từ sau:
P6: Sal > 20000, P7: Sal ≤ 20000.

Bài 2

Cho tập câu truy vấn Q = {q1, q2, q3, q4, q5}, tập các thuộc tính A = {A1, A2, A3, A4, A5} của lược đồ quan hệ R và tập các sites S = {S1, S2, S3}.

Cho ma trận U biểu diễn các thuộc tính được sử dụng ứng với các câu truy vấn và ma trận F thể hiện tần số sử dụng các câu truy vấn tại các sites.

Ma trận U

	A1	A2	A3	A4	A5
q1	0	1	1	0	1
q2	1	1	1	0	1
q3	1	0	0	1	1
q4	0	0	1	0	0
q5	1	1	1	0	0

Ma trận F

	S1	S2	S3
q1	10	20	0
q2	5	0	10
q3	0	35	5
q4	0	10	0
q5	0	15	0

Giả sử refi(qk) = 1 với mọi qk và Si và A1 là thuộc tính khóa của R.

Câu hỏi

Sử dụng thuật toán Bond Energy và Vertical Partitioning để tìm ra các phân mảnh dọc của R.

ĐỀ SỐ 3

Một công ty sản xuất có nhiều công nhân. Thông tin của mỗi công nhân được thể hiện qua lược đồ quan hệ :

Congnhan(MaCN, TenCN, Diachi, Phanxuong, To)

Mỗi công nhân làm việc ở một phân xưởng. Công ty có hai phân xưởng A và B. Phân xưởng A nằm ở SàiGòn, còn phân xưởng B thì ở Bình Dương.

Một công nhân thì thuộc về một tổ. Thông tin về tổ được thể hiện qua lược đồ quan hệ sau :

To(Mato, TenTo)

Công ty có ba tổ : tổ giao và nhận, tổ sản xuất, tổ đóng gói.

Công ty muốn xây dựng một hệ thống thông tin quản lý công nhân có cơ sở dữ liệu phân tán tại hai nơi Saigon và Bình Dương.

CÂU HỎI

1. Hãy thiết cơ sở dữ liệu phân tán cho hệ thống thông tin quản lý công nhân
    a. Thiết kế các phân mảnh ngang nguyên thủy cho lược đồ Congnhan :
    + Tìm các vị từ đơn giản
    + Tìm các vị từ sơ cấp đầy đủ và cực tiểu
    b. Thiết kế các phân mảnh ngang dẫn xuất cho lược đồ To.
2. Hãy mô tả quá trình cấp phát các mảnh trên các sites của công ty
3. Viết câu truy vấn sau ở mức độ trong suốt ánh xạ cục bộ :

a. Liệt kê các công nhân và tên tổ của công nhân ấy theo phân xưởng (được nhập).

b. Biết công nhân có mã số 1234 làm việc ở phân xưởng A. Viết câu truy vấn đổi công nhân 1234 về phân xưởng B.

ĐỀ SỐ 4

Phần 1: Lý thuyết (3 điểm)

Câu a (1 điểm) Cho lược đồ phổ quát :

R(a, b) : b có giá trị nguyên trong đoạn [1,5]

S(a, c) : c có giá trị nguyên trong đoạn [1,3]

$R_i = \sigma_{b=i}(R)$

$S_{ij} = \sigma_{c=j}(S) \blacktriangleright < R_i$

Cho biết tổng số phân mảnh?

Câu b (2 điểm) Phân mảnh đầy đủ được định nghĩa bởi

Cho R(a), S(a, b), T(b) và truy vấn $R \bowtie_a S \bowtie_b T$

Cho biết có bao nhiêu cây truy vấn tuyến tính phải trong không gian tìm kiếm ứng với cây truy vấn trên và xác đinh các cây đó?

Phần 2: Bài tập (7 điểm)

Một tổng công ty Điện lực gồm có nhiều công ty thành viên có trụ sở tại vùng Đồng bằng, Biển đảo và Cao nguyên. Lược đồ công ty thành viên được mô tả như sau :

Congty(<u>Macongty</u>, tencongty, vung)

Nhân viên thì thuộc về một công ty. Lược đồ nhân viên được mô tả như sau :

Nhânviên(<u>MaNV</u>, tênNV, điachi, bacluong, chuyenmon, macongty)

Một công ty đảm trách nhiều công trình điện. Lược đồ Congtrinh được mô tả như sau :

Congtrinh(<u>MaCongtrinh</u>, tenCongtrinh, diadiem, Macongty, Ngaybatdau, NgayKetthuc)

Một Công trình có nhiều nhân viên tham gia và một nhân viên có thể tham gia nhiều Công trình. Một nhân viên tham gia một Công trình với một vai trò xác định. Lược đồ tham gia Công trình của nhân viên được mô tả như sau :

Thamgia(Ma<u>Congtrinh, MaNV</u>, vaitro)

Tổng công ty xây dựng muốn xây dựng một hệ thống thông tin quản lý Công trình có cơ sở dữ liệu phân tán tại các công ty đặt tại các trung tâm dữ liệu Sài gòn (quản lý

Đồng bằng), Đà nẵng (quản lý Biển đảo) và Hà nội (quản lý Cao nguyên).
1. Hãy thiết cơ sở dữ liệu phân tán cho hệ thống thông tin quản lý Công trình

a. Thiết kế các phân mảnh ngang nguyên thủy cho lược đồ Công ty

b. Thiết kế các phân mảnh ngang dẫn xuất của lược đồ Nhân viên, Công trình ứng với các phân mảnh theo công ty tương ứng ở câu a và lược đồ Tham gia ứng ưng với các phân mảnh theo Công trình ở câu b.

c. Gia sử ta có hai ứng dụng chuyên tìm thông tin như sau :

   Ứng dụng 1 : Tìm tên nhân viên, địa chỉ và bậc lương của nhân viên theo công ty

   Ứng dụng 2 : Tìm tên nhân viên, chuyên môn theo công ty.

   Hãy thiết kế phân mảnh dọc hỗn hợp tương ứng với các ứng dụng từ các phân mảnh Nhân viên ở câu b.

2. Hãy mô tả quá trình cấp phất các phân mảnh tại ba sites Sàigòn, Hà nội, Đà nẵng.

3. Dựa trên câu 1, viết câu truy vấn sau ở mức độ trong suốt ánh xạ cục bộ :

Biết nhân viên có mã số 412011 làm việc ở một công ty tại Đồng bằng. Viết câu truy vấn đổi nhân viên 412011 về công ty có mã HN07 tại Cao nguyên. Nếu nhân viên này có tham gia các Công trình tại công ty ở Đồng bằng thì xóa toàn bộ các bộ chứa nhân viên này.

4. Dựa trên câu 1, trình bày quá trình chuyên câu truy vấn toàn cục thành câu truy vấn dựa trên các phân mảnh cho câu truy vấn sau :

Liệt kê các tên nhân viên, chuyên môn và vai trò của họ khi tham gia Công trình có mã là ‘P1218’. Biết rằng Công trình này do một công ty ở vùng Biển đảo phụ trách.

a. Viết câu lệnh SQL dưới dạng lược đồ toàn cục.

b. Chuyển cây truy vấn ở câu b dưới dạng cây truy vấn phân mảnh tối giản.

   Hãy thiết kế phân mảnh dọc hỗn hợp tương ứng với các ứng dụng từ các phân mảnh Khách hàng ở câu b.

ĐỀ TÀI
QUẢN LÝ ĐIỂM SINH VIÊN

Cho cơ sở dữ liệu QLDSV sau:

a.Khoa :

FieldName	Type	Constraint
MAKH	Char(8)	Primary Key
TENKH	Varchar(40)	Unique

b. Lop :

FieldName	Type	Constraint
MALOP	Char(8)	Primary Key
TENLOP	Varchar(40)	Unique
MAKH	Char(8)	FK

c. Table Sinhvien:

FieldName	Type	Constraint
MASV	Char(8)	Primary key
HO	Varchar(40)	
TEN	Varchar(10)	
MALOP	Char(8)	Foreign Key
PHAI	Bit	Default : 1 (1: Nam; 0: Nữ)
NGAYSINH	DateTime	
NOISINH	Varchar(40)	
DIACHI	Varchar(80)	
GHICHU	Text	
NGHIHOC	Bit	

d. Cấu trúc của Table Mônhoc:

FieldName	Type	Constraint
MAMH	Char(5)	Primary key
TENMH	Varchar(40)	Unique Key

e. Cấu trúc của Table DIEM :

FieldName	Type	Constraint
MASV	Char(8)	Foreign Key
MAMH	Char(8)	Foreign Key

HOCKY	SmallInt	Học kỳ>=1 và Học kỳ <=9
LAN	SmallInt	Lần thi >=1 và Lần thi <=2
DIEM	float	Điểm = -1 (Vắng) hay Điểm từ 0 đến 10
Primary key : **MASV + MAMH + LAN**		

Yêu cầu: Giả sử trường có 3 khoa chính : công nghệ thông tin (CNTT), quản trị kinh doanh (QTKD) và viễn thông (VT)

1. Phân tán cơ sở dữ liệu QLDSV ra làm 3 mảnh với điều kiện sau:

- QLDSV_CNTT được đặt trên server1: chứa thông tin của các sinh viên thuộc khoa công nghệ thông tin
- QLDSV_QTKD được đặt trên server2: chứa thông tin của các sinh viên thuộc khoa quản trị kinh doanh
- QLDSV_VT được đặt trên server3: chứa thông tin của các sinh viên thuộc khoa viễn thông.

Biết rằng 1 sinh viên chỉ có thể thuộc 1 khoa.

Lưu ý: tên CSDL của các phân mảnh nên đặt giống nhau

2. Hãy cho biết cách tạo Link Server như thế nào để đảm bảo tính "trong suốt" khi gọi 1 View hay Stored Procedure trong cơ sở dữ liệu phân tán.

- Mục đích: cho phép truy cập dữ liệu từ server 1 đến Server 2 qua tập lệnh DML
- Cú pháp : Giả sử ta đang đứng ở Server LUUTHU\TINTIN, ta muốn tạo 1 link server tên LINK2 đến Server LUUTHU\SUSU

```
EXEC  sp_addlinkedserver  @server='LINK2', @srvproduct=' ',
                @provider='SQLOLEDB', @datasrc='LUUTHU\SUSU'
```

Ví dụ áp dụng: Sử dụng link server vừa tạo để in ra danh sách lớp của khoa viễn thông đang đặt ở Server LUUTHU\SUSU

```
Select * from LINK2.QLDSV.DBO.LOP
```

- Quy tắc :

a. Tên các cơ sở dữ liệu ở các Server Subscriber phải giống nhau

b. Tạo Link theo quy tắc vòng tròn :

+ Link1 : link ngay đến chính nó

+ Link2: link đến Server gần nó nhất

+ Link3 : link đến Server gần kế nhất

3. Tạo các View và Stored Procedure sau trên cơ sở dữ liệu gốc. Sau đó, đẩy chúng qua các phân mảnh.

a. Liệt kê danh sách sinh viên (masv, hoten) của lớp có mã lớp @X

b. Tạo một View trả về tổng số sinh viên từng lớp của toàn trường

c. In ra các môn thi chưa đạt của sinh viên có mã số @masv . Kết xuất:

TenMH ĐiemLonNhat

d. Viết một Stored Procedure tên PhieuDiem để liệt kê các điểm thi lớn nhất các môn học của một sinh viên dựa vào mã sinh viên.