

ThS. ĐẶNG THỊ KIM ANH

Bài giảng
CƠ SỞ DỮ LIỆU PHÂN TÁN

TRƯỜNG ĐẠI HỌC LÂM NGHIỆP - 2015

LỜI NÓI ĐẦU

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

Bài giảng bao gồm 6 chương với nội dung như sau:

Chương 1: Tổng quan về cơ sở dữ liệu phân tán

Chương 2: Thiết kế cơ sở dữ liệu phân tán

Chương 3: Xử lý truy vấn trong cơ sở dữ liệu phân tán

Chương 4: Quản lý giao dịch và điều khiển đồng thời phân tán

Chương 5: Các hệ cơ sở dữ liệu song song

Chương 6: Hệ quản trị cơ sở dữ liệu đối tượng phân tán

Với nội dung như trên, sau mỗi chương đều có câu hỏi ôn tập và bài tập ôn luyện, nhằm giúp các bạn củng cố kiến thức lý thuyết của mình.

Mặc dù tác giả đã rất cố gắng nhưng không thể tránh khỏi những sai sót về cách diễn đạt, sự sắp xếp bố cục nội dung và các lỗi cú pháp, văn phong. Rất mong nhận được những ý kiến đóng góp của bạn đọc để Bài giảng hoàn thiện hơn.

Tác giả

Chương 1

TỔNG QUAN VỀ CSDL PHÂN TÁN

Tại sao phải phân tán? Phân tán cái gì?

- Nhu cầu dùng chung một dữ liệu hợp nhất;
- Các tổ chức kinh tế có nhiều trụ sở phân tán ở nhiều vị trí địa lý khác nhau;
- Làm thế nào để có thể quản lý các luồng dữ liệu và sử dụng chung dữ liệu, chương trình của tổ chức;
- Một hệ thống phân tán bao gồm hai phần: mạng máy tính và CSDL phân tán.

1.1. Sơ lược về mạng máy tính

- Một mạng máy tính là một tập các máy tính tự vận hành, được kết nối lại và có khả năng trao đổi thông tin giữa chúng.

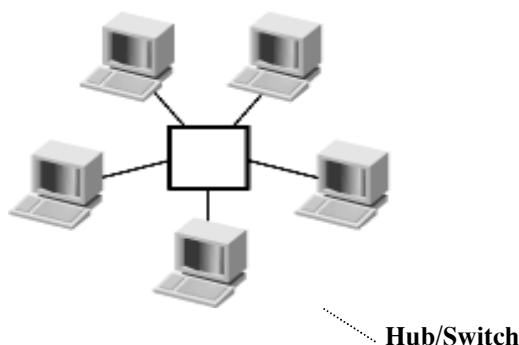
- Các máy tính trên một mạng thường được gọi là các nút hay các trạm, chúng tạo ra các phần cứng cơ bản của mạng và được kết nối lại với nhau bởi một đường truyền.

- Một mạng máy tính là một trường hợp đặc biệt của môi trường xử lý phân tán, trong đó các máy tính là các thiết bị được kết nối vào kênh truyền dữ liệu.

Những loại mạng máy tính cơ bản

a. Mạng star (hình sao)

Trong cấu trúc mạng star, các máy tính được nối cáp vào một thiết bị đầu nối trung tâm (Hub hoặc Switch). Tín hiệu được truyền từ máy tính gửi dữ liệu qua Hub (tín hiệu được khuếch đại) và truyền đến tất cả máy tính trên mạng.



Hình1.1. Cấu trúc mạng Star

*** Ưu điểm:**

- Cấu trúc Star cung cấp chế độ quản lý tập trung. Khi một đoạn cáp bị hỏng thì chỉ ảnh hưởng đến máy dùng đoạn cáp đó, mạng vẫn hoạt động bình thường.
- Mạng có thể mở rộng hoặc thu hẹp một cách dễ dàng.

*** Nhược điểm:**

- Do mỗi máy tính đều phải nối vào một trung tâm điểm nên cấu trúc này đòi hỏi phải nhiều cáp và phải tính toán đặt vị trí của thiết bị trung tâm.
- Khi thiết bị trung tâm bị hỏng thì toàn bộ hệ thống mạng ngừng hoạt động.

b. Mạng vòng (ring)

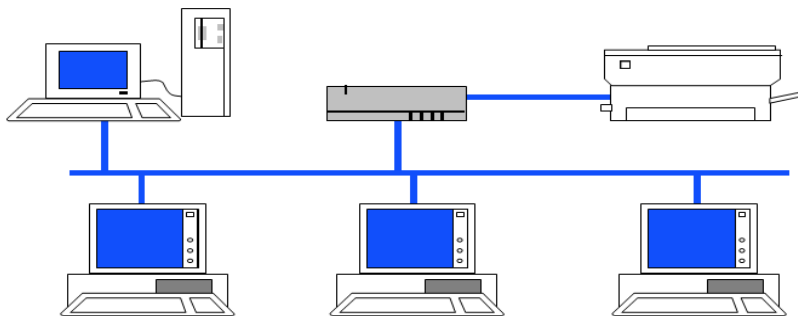
- Các máy tính được nối với nhau theo vòng khép kín;
- Mỗi trạm đóng vai trò là một bộ chuyển tiếp;
- Truyền dữ liệu quanh vòng thường theo một chiều;
- Việc điều khiển truyền tin trên mạng xoay vòng thường được thực hiện bằng thẻ điều khiển.

*** Nhược điểm:**

- Nếu đường nối chỉ bị cắt đứt thì có thể làm ngừng toàn bộ hoạt động của mạng.
- Lượng thông tin di chuyển trên đường truyền lớn.

c. Mạng bus:

- Có một kênh chung để truyền dữ liệu, các máy tính và các thiết bị đầu cuối sẽ được gắn vào đó.

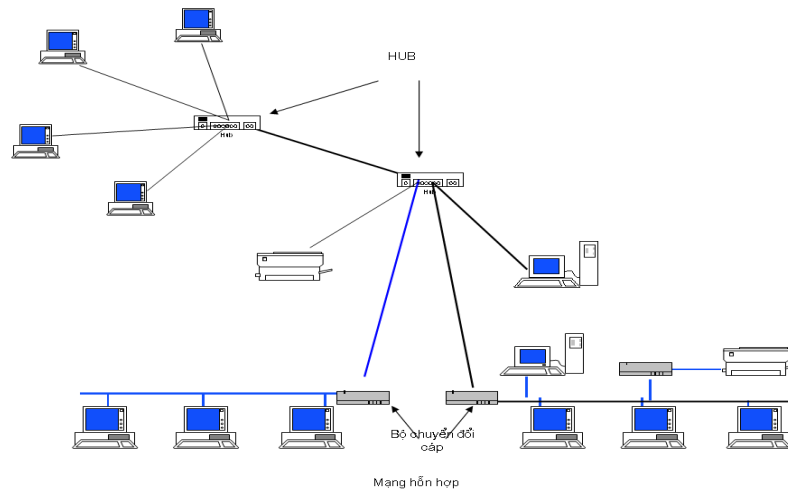


Hình 1.2. Mô hình mạng Bus

d. Mạng hỗn hợp (hybrid):

- Các mạng truyền thông thường có các đường nối vô định;
- Các đường nối không có tính hệ thống cũng không tuân theo một khuôn mẫu nào;
- Có thể gặp một nút chỉ nối với một nút khác và cả những nút nối với nhiều nút khác;

- Các kết nối giữa các máy tính trên mạng Internet thuộc loại này.



Hình 1.3. Mô hình mạng hỗn hợp

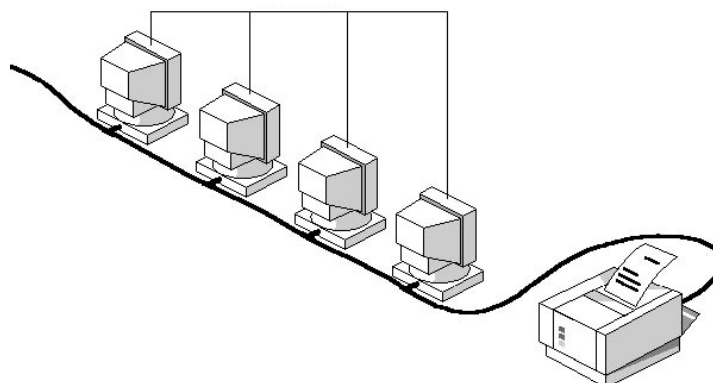
e. Mạng thảm (Mesh):

- Mỗi nút đều được kết nối với các nút còn lại;
- Có độ tin cậy cao hơn và khả năng hoạt động tốt hơn những cấu trúc đã nói ở trên;
- Cấu trúc này có chi phí cao.

1.2. Các hình thức tổ chức hệ thống phân tán

a. Peer-to-Peer (mô hình nhóm làm việc):

- Các máy tính cá nhân và máy trạm có thể được sử dụng như một hệ thống độc lập trợ giúp các ứng dụng địa phương;
- Mỗi thành viên trong mạng có vai trò ngang nhau, tự quản lý tài nguyên của chính mình và chia sẻ tài nguyên cho các máy tính khác trên mạng;
- Mỗi một máy tính trên mạng vừa đóng vai trò máy chủ (Server), vừa đóng vai trò là máy khách (Client);
- Đây là mô hình mạng đơn giản, phù hợp với những hệ thống mạng nhỏ không có yêu cầu cao về bảo mật.



Hình 1.4. Mô hình nhóm làm việc

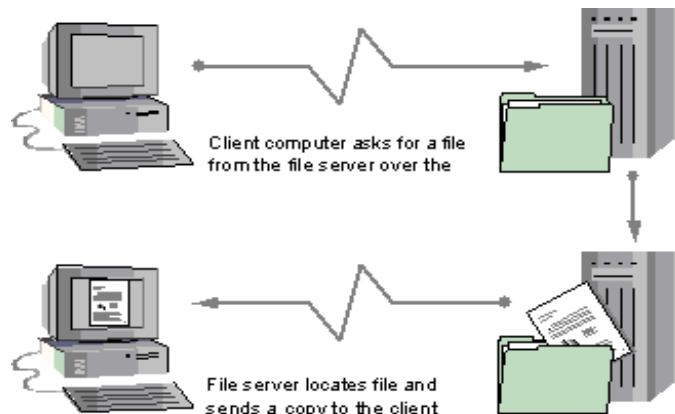
b. File server (máy dịch vụ file)

- **File server:** Một số máy dịch vụ file được gắn trực tiếp vào mạng LAN.
- **File server** là một thiết bị quản lý các hoạt động file và phục vụ các máy tính cá nhân được kết nối trong mạng LAN.
- Mỗi máy cá nhân được phân chia một dung lượng cố định trên ổ cứng của **File server**, chương trình ở các máy tính cá nhân có thể tham chiếu đến các file trên phần đĩa tương ứng của nó bằng một đặc tả đường dẫn.
- Những hạn chế của **File server**
 - Dữ liệu di chuyển trên mạng quá nhiều;
 - Việc kiểm soát dữ liệu là phi tập trung;
 - Các máy trạm phải đủ mạnh.

c. Client/server (mô hình khách/chủ)

Một số máy dịch vụ file được gắn trực tiếp vào mạng LAN.

- **Server** có chức năng điều khiển, lưu trữ CSDL, xử lý các truy vấn và quản lý việc khai thác tài nguyên trên mạng của các máy tính khác.
- Thuật ngữ client được sử dụng để chỉ người khai thác tài nguyên mạng.



Hình 1.5. Mô hình máy khách chủ

1.3. Các đặc trưng của các loại hình hệ thống phân tán

Việc chuyển các ứng dụng trên máy tính cá nhân cũng như các ứng dụng trên các hệ thống máy lớn trung tâm sang mô hình phân tán là một xu hướng phát triển mạnh.

Vấn đề đặt ra là, cần lựa chọn hình thức phân tán thích hợp nào cho mỗi

mô hình được chuyển đổi.

1.3.1. Đặc trưng của hệ thống File server và kiến trúc Client/Server

Đặc trưng	File Server	Client/Server
Xử lý	Chỉ ở máy khách	Cả máy khách và máy chủ
Truy nhập dữ liệu đồng thời	Thấp, mỗi máy khách thực hiện	Cao, Server đảm nhiệm
An toàn và toàn vẹn CSDL	Thấp, máy khách quản lý	Cao, Server đảm nhiệm
Sử dụng mạng	File lớn, chuyển cả file	Truyền dữ liệu nhiều mức
Bảo trì phần mềm	Thấp, chỉ ở máy Server	Hỗn hợp một số phần mềm có thể gửi đến máy khách
Phần cứng và hệ thống Sự mềm dẻo của phần mềm	Ghép nối máy khách và Server để có thể phối hợp	Ghép nối máy khách và Server để có thể phối hợp

1.3.2. Các chức năng của kiến trúc client/server

1.3.2.1. Trình diễn thông tin phân tán

Mục đích: Làm mới các ứng dụng trên các máy khách và để định dạng lại dữ liệu do server quản lý.

Chức năng	Client	Server
Quản lý dữ liệu		Quản lý mọi dữ liệu
Phân tích dữ liệu		Phân tích mọi dữ liệu
Trình diễn dữ liệu	Dữ liệu trình diễn trên server được định dạng để trình diễn cho các clients	Sử dụng công nghệ trình diễn của server để gửi dữ liệu gửi cho các clients

1.3.2.2. Trình diễn từ xa

Mục đích: Giúp USERS thay đổi các biểu mẫu, báo cáo hoặc nội dung mới thì chỉ cần bảo trì phần mềm trên máy khách.

Kiểu trình diễn này được cài đặt các chức năng trình diễn dữ liệu trên các clients, nên phần mềm trên các client có khả năng trình diễn những dữ liệu được

định dạng theo ý của client.

Chức năng	Client	Server
Quản lý dữ liệu		Quản lý mọi dữ liệu
Phân tích dữ liệu		Phân tích mọi dữ liệu
Trình diễn dữ liệu	Dữ liệu phân tích trên server được định dạng để trình diễn cho người dùng	

1.3.2.3. Quản lý dữ liệu từ xa

Mục đích: giúp client phân tích được dữ liệu thô lấy từ Server

Chức năng	Client	Server
Quản lý dữ liệu		Quản lý mọi dữ liệu
Phân tích dữ liệu	Dữ liệu thô được lấy từ server và được phân tích	
Trình diễn dữ liệu	Trình diễn tất cả dữ liệu	

1.3.2.4. Phân tán chức năng

- Các chức năng phân tán được đặt trên cả máy khách lẫn máy chủ.
- Toàn bộ chức năng trình diễn dữ liệu được đặt trên máy khách và toàn bộ chức năng quản lý dữ liệu được đặt trên máy chủ.

Chức năng	Client	Server
Quản lý dữ liệu		Quản lý mọi dữ liệu
Phân tích dữ liệu	Các dữ liệu được lấy và phân tích từ server	Các dữ liệu được lấy và phân tích từ server sau đó được truyền cho các clients
Trình diễn dữ liệu	Tất cả dữ liệu (được phân tích trên cả server và clients)	

1.4. Cơ sở dữ liệu phân tán (*Distributed Database*)

1.4.1. Định nghĩa

Một cơ sở dữ liệu (CSDL) phân tán là một tập hợp dữ liệu, mà về mặt logic tập hợp này thuộc cùng một hệ thống, nhưng về mặt vật lý dữ liệu đó được

phân tán trên các vị trí khác nhau của một mạng máy tính.

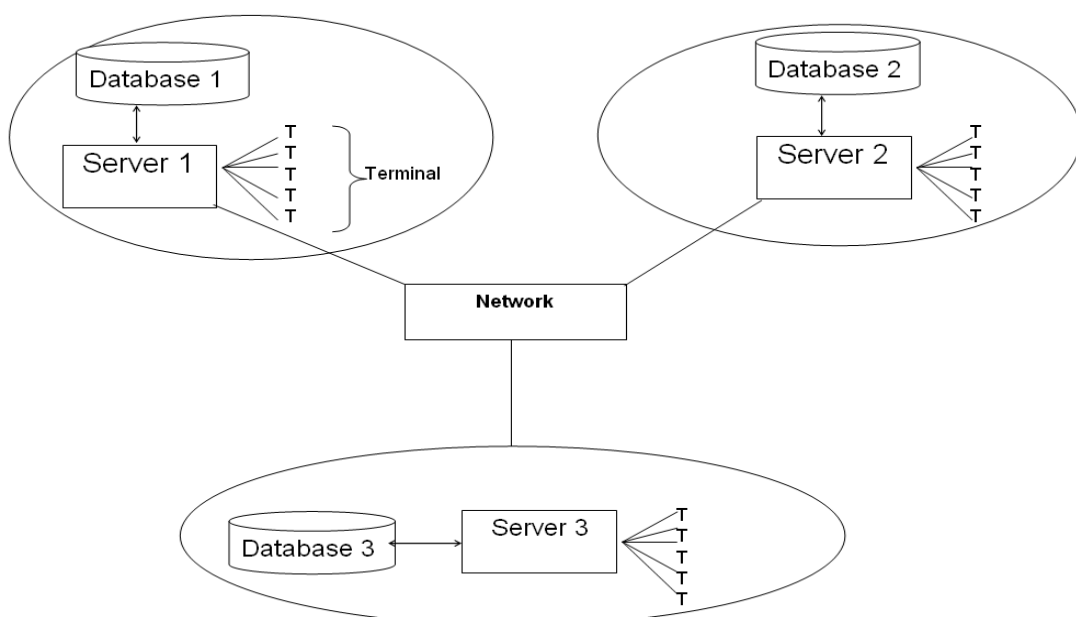
Có hai điểm quan trọng được nêu ra trong định nghĩa:

Phân tán: Dữ liệu không cư trú trên một vị trí mà được phân bố rộng khắp trên nhiều máy tính đặt tại nhiều vị trí khác nhau, đây là điểm phân biệt một cơ sở dữ liệu phân tán với một cơ sở dữ liệu tập trung.

Tương quan logic: Dữ liệu trong hệ phân tán có một số thuộc tính ràng buộc chúng với nhau. Điều này giúp chúng ta có thể phân biệt một cơ sở dữ liệu phân tán với một tập hợp cơ sở dữ liệu tập trung, các file dữ liệu được lưu trữ tại nhiều vị trí khác nhau, điều này thường thấy trong các ứng dụng mà hệ thống sẽ phân quyền truy nhập dữ liệu trong môi trường mạng.

Ví dụ 1.1:

- Một ngân hàng có ba chi nhánh đặt tại các vị trí khác nhau.
- Tại mỗi chi nhánh có một máy tính điều khiển một số máy kế toán cuối cùng (teller terminal) và cơ sở dữ liệu thống kê của chi nhánh đó.
- Mỗi máy tính với cơ sở dữ liệu thống kê địa phương của nó tại mỗi chi nhánh được đặt ở một vị trí của cơ sở dữ liệu phân tán. Các máy tính được nối với nhau bởi một mạng truyền thông. Các nút trong một mạng phân tán một mặt xử lý thông tin tại vị trí mà nó quản lý, mặt khác nó cũng tham gia vào việc xử lý các yêu cầu về thông tin cần truy cập qua nhiều địa điểm. Ví dụ như việc lên danh sách tất cả nhân viên của ngân hàng. Yêu cầu này đòi hỏi tất cả các máy tính ở các chi nhánh của công ty đều phải hoạt động để cung cấp thông tin.



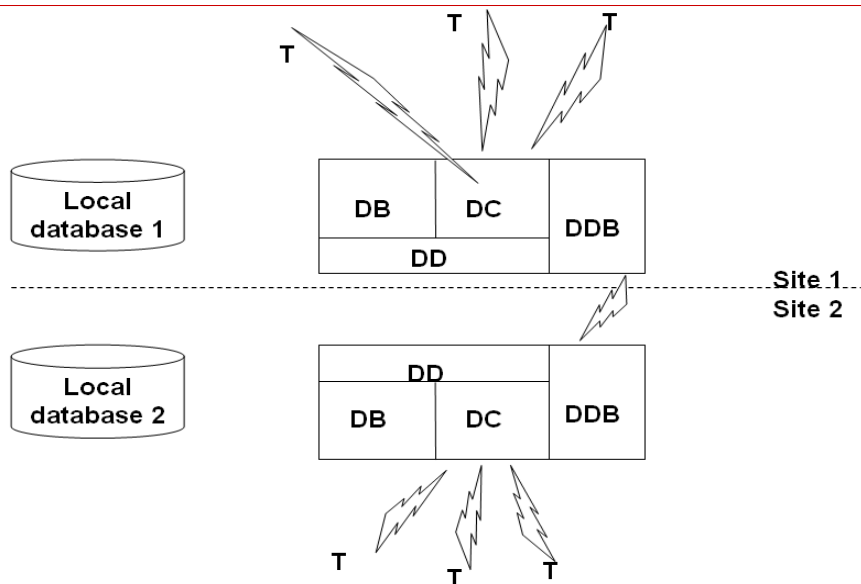
Hình 1.6. Ví dụ về một hệ thống phân tán

1.4.2. Hệ quản trị cơ sở dữ liệu phân tán (DDBMSs)

DDBMS có chức năng hỗ trợ việc tạo và bảo trì cơ sở dữ liệu phân tán, chúng có các thành phần tương tự như một hệ quản trị cơ sở dữ liệu tập trung và các thành phần hỗ trợ trong việc chuyển tải dữ liệu đến các trạm và ngược lại.

Các thành phần sau đây đòi hỏi một DDBMSs thương mại phải có:

- Quản trị dữ liệu (Database management): DB;
- Truyền thông dữ liệu (Data Communication): DC;
- Từ điển dữ liệu (Data Dictionary): DD dùng để mô tả thông tin về sự phân tán của dữ liệu trên mạng;
- Cơ sở dữ liệu phân tán (Distributed Database): DDB.



Các thành phần của một DDBMS thương mại

Hình 1.7. Mô hình hệ quản trị cơ sở dữ liệu phân tán thương mại

1.4.3. So sánh CSDL phân tán và CSDL tập trung

CSDL phân tán không đơn giản là những sự thực hiện phân tán của CSDL tập trung, bởi vì chúng cho phép thiết kế các đặc trưng khác với CSDL tập trung truyền thống.

Các đặc điểm tiêu biểu của CSDL truyền thống:

- Điều khiển tập trung;
- Độc lập dữ liệu;
- Giảm dư thừa;

- Biệt lập và bảo mật dữ liệu.

a. Điều khiển tập trung

Trong CSDL tập trung: Khả năng điều khiển tập trung trên toàn nguồn tài nguyên thông tin của tổ chức, được xem là động cơ mạnh nhất cho việc ra đời CSDL. Chúng được phát triển như là sự tiến hoá của hệ thống thông tin, mà trong đó mỗi ứng dụng có các tập tin riêng của nó.

Trong CSDL phân tán: Ý niệm về điều khiển tập trung ít được nhấn mạnh hơn. Điều này phụ thuộc vào kiến trúc của CSDL phân tán.

Một cách tổng quát, CSDL phân tán được điều khiển với cấu trúc phân lớp dựa vào một hệ quản trị CSDL toàn cục (có trách nhiệm trên toàn bộ CSDL phân tán) và hệ quản trị CSDL địa phương (có trách nhiệm với CSDL địa phương riêng).

b. Độc lập dữ liệu

Độc lập dữ liệu cũng là một trong những động lực cho việc mở đầu sự tiếp cận dữ liệu phân tán. Thuận lợi chính của độc lập dữ liệu là các chương trình không bị ảnh hưởng bởi sự thay đổi trong cấu trúc vật lý của dữ liệu. Trong CSDL phân tán, độc lập dữ liệu cũng quan trọng giống như trong CSDL truyền thống. Tuy nhiên, một khía cạnh mới được thêm vào trong ý niệm của độc lập dữ liệu là *trong suốt phân tán*. Với trong suốt phân tán chúng ta hiểu rằng các chương trình ứng dụng có thể sử dụng CSDL như là nó không được tổ chức phân tán. Vì thế sự chính xác của chương trình không bị ảnh hưởng bởi việc dịch chuyển dữ liệu từ trạm này đến trạm khác. Tuy nhiên, tốc độ thực hiện của chúng bị ảnh hưởng.

c. Giảm dư thừa dữ liệu

Trong CSDL truyền thống, dữ liệu dư thừa được giảm đến mức tối thiểu bởi hai lý do:

- Sự không tương thích giữa nhiều bản sao của cùng một tập dữ liệu;
- Tiết kiệm không gian lưu trữ bằng cách loại bỏ các dư thừa. Việc giảm dư thừa dữ liệu có thể đạt được bằng cách chia sẻ dữ liệu, cho phép nhiều ứng dụng truy cập cùng các bản tin và bản ghi.

Trong CSDL phân tán, việc giảm dư thừa phức tạp hơn vì ngoài hai lý do

trên, còn nhiều lý do để giảm dư thừa như:

- Hoạt động của các trình ứng dụng có thể bị tăng lên khi dữ liệu được sao lại tất cả các vị trí, nơi trình ứng dụng cần nó;

- Tính thường trực của hệ thống sẽ tăng lên, bởi vì khi có lỗi xảy ra ở một trạm nào đó sẽ không dừng việc thực hiện các ứng dụng của trạm khác nếu dữ liệu đã được sao chép lại.

d. Biệt lập và bảo mật

Trong CSDL truyền thống, hệ quản trị CSDL tập trung có thể bảo đảm chỉ truy cập đến dữ liệu đã được uỷ quyền.

Trong CSDL phân tán, hệ quản trị dữ liệu địa phương thực chất phải đương đầu với các vấn đề giống như hệ quản trị CSDL trong CSDL truyền thống.

Tuy nhiên, hai khía cạnh đặc biệt sau đây của CSDL phân tán cần phải được xem xét:

- Trong CSDL phân tán với một mức độ tự trị rất cao của các địa phương, người chủ dữ liệu địa phương cảm giác được bảo vệ tốt hơn vì họ có thể tự chủ thực hiện bảo vệ thay vì phụ thuộc vào người quản trị CSDL trung tâm;

- Vấn đề bảo mật là bản chất trong hệ phân tán nói chung, vì các mạng truyền thông diện rộng cho phép nhiều người cập nhật và khai thác dữ liệu nên cần được bảo vệ.

1.4.4. Ưu và nhược điểm của hệ phân tán

Ưu điểm:

- Đáp ứng nhanh hầu hết các ứng dụng sử dụng dữ liệu tại các trạm;
- Tăng cường các đơn thể ứng dụng và CSDL mà không làm cản trở người sử dụng hiện tại;
- Kiểm soát dữ liệu địa phương theo hướng hoàn thiện sự tích hợp và quản trị dữ liệu từ xa;
- Tăng cường khả năng của hệ thống liên quan đến sự dư thừa dữ liệu.

Nhược điểm:

- Phần mềm đắt và phức tạp;
- Phải xử lý các thay đổi thông báo trong mọi địa điểm;
- Khó kiểm soát tính toàn vẹn dữ liệu với nhiều bản sao dữ liệu được phân

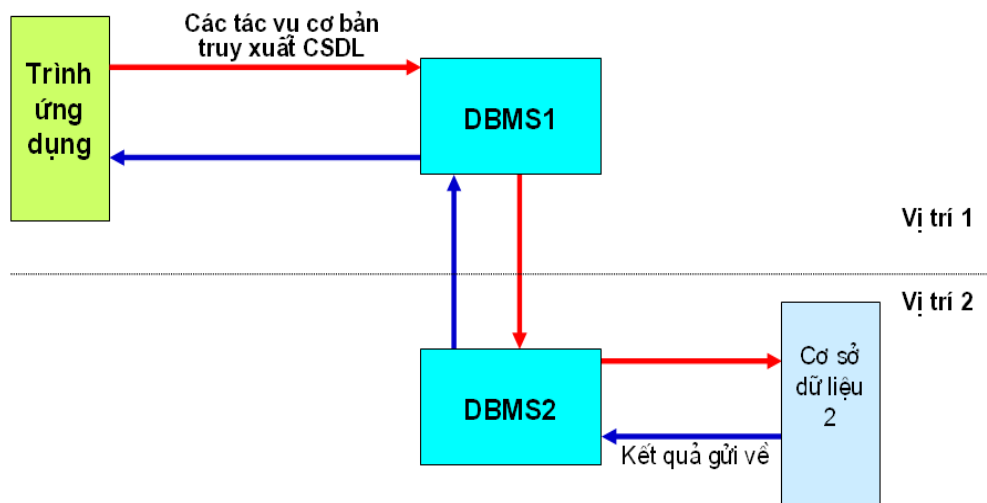
bổ khắp mọi nơi;

- Đáp ứng chậm nhu cầu của các trạm trong trường hợp các phần mềm ứng dụng không được phân bố phù hợp với việc sử dụng chung.

1.5. Các loại truy xuất CSDL phân tán

1.5.1. Truy xuất từ xa thông qua các tác vụ cơ bản:

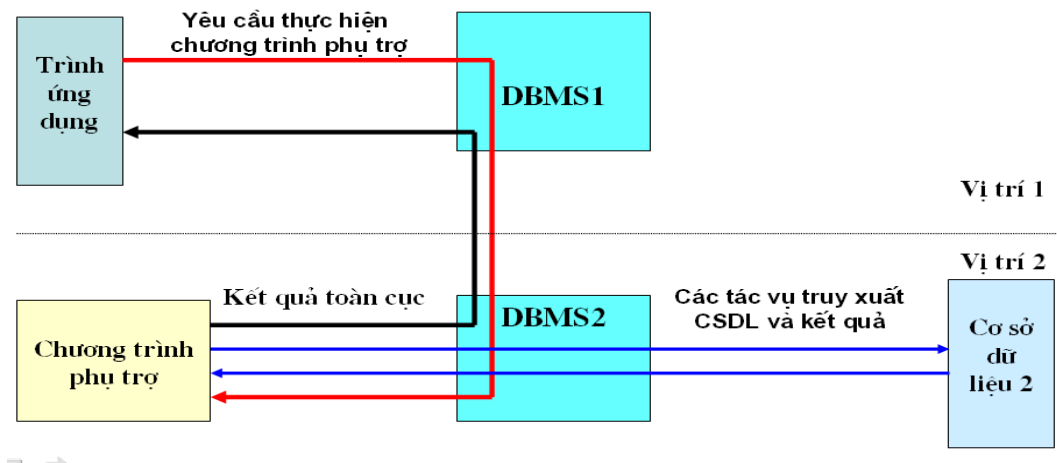
Ứng dụng phát ra một yêu cầu truy xuất CSDL ở một vị trí nào đó. Yêu cầu này sẽ được hệ quản trị CSDL phân tán gửi đến vị trí chứa dữ liệu đó. Thực hiện xong sẽ gửi kết quả về.



Hình 1.8. Mô hình truy xuất từ xa

1.5.2. Truy xuất từ xa thông qua chương trình phụ trợ

Một ứng dụng yêu cầu thực hiện một chương trình phụ trợ đặt tại vị trí từ xa. Chương trình phụ trợ này sẽ truy xuất CSDL từ xa và trả lại kết quả cho ứng dụng đang yêu cầu.

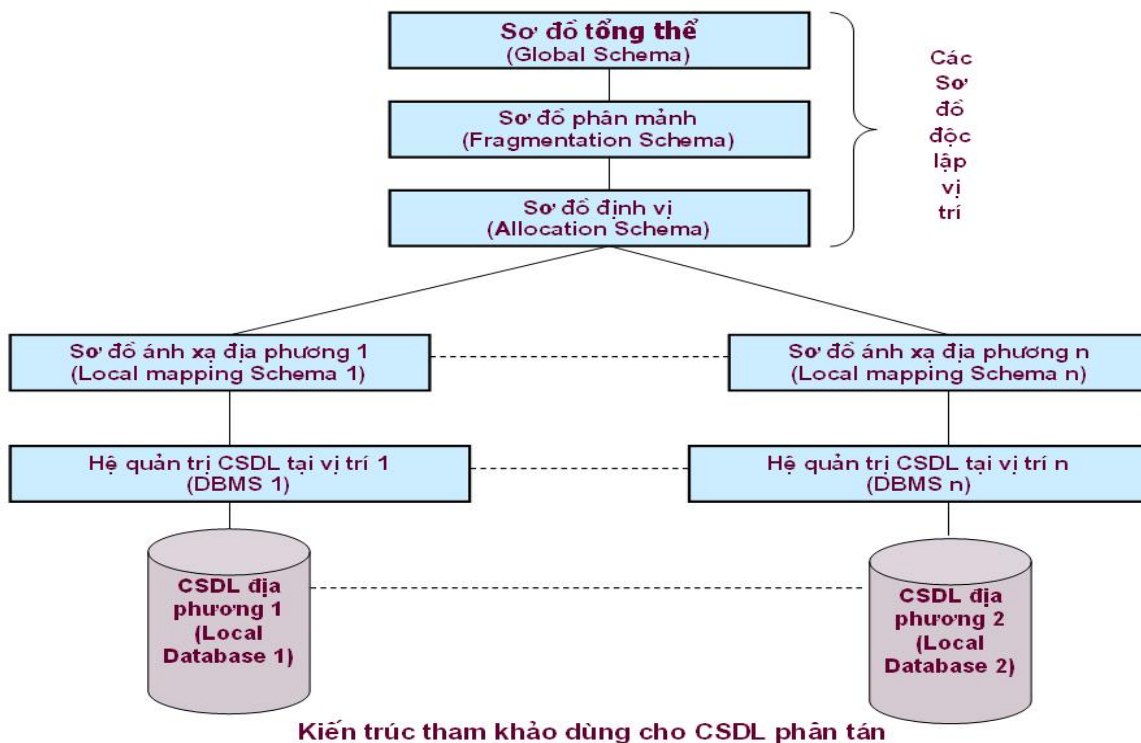


Hình 1.9. Mô hình truy xuất từ xa có phụ trợ

Kết luận:

- Cơ sở dữ liệu phân tán là quan trọng trong kinh tế, tổ chức và kỹ thuật với nhiều lý do khác nhau. Chúng có thể được cài đặt trên một mạng máy tính có phạm vi rộng lớn hoặc nhỏ bé;
- Hiện nay, các DDBMSs thương mại đều tích hợp các ứng dụng phân tán nên rất tiện cho người sử dụng.

1.6. Kiến trúc cơ bản của CSDL phân tán



Hình 1.10. Kiến trúc cơ bản của cơ sở dữ liệu phân tán

a. Sơ đồ tổng thể (Global Schema)

- Xác định tất cả các dữ liệu sẽ được lưu trữ trong cơ sở dữ liệu phân tán cũng như các dữ liệu không được phân tán ở các trạm trong hệ thống;
- Sơ đồ tổng thể được định nghĩa theo cách như trong CSDL tập trung;
- Trong mô hình quan hệ, sơ đồ tổng thể bao gồm định nghĩa của tập các quan hệ tổng thể (*Global relation*).

b. Sơ đồ phân đoạn (fragment schema)

- Mỗi quan hệ tổng thể có thể chia thành một vài phần không giao nhau gọi là phân đoạn (fragment);
- Có nhiều cách khác nhau để thực hiện việc phân chia này;
- Sơ đồ phân đoạn mô tả các ánh xạ giữa các quan hệ tổng thể và các

đoạn được định nghĩa trong sơ đồ phân đoạn (fragmentation Schema);

- Các đoạn được mô tả bằng tên của quan hệ tổng thể cùng với chỉ mục đoạn. Chẳng hạn, R_i được hiểu là đoạn thứ i của quan hệ R .

c. Sơ đồ định vị (allocation schema)

- Các đoạn là các phần logic của một quan hệ tổng thể được định vị vật lý trên một hay nhiều trạm;

- *Sơ đồ định vị* xác định đoạn dữ liệu nào được định vị tại trạm nào trên mạng;

- Tất cả các đoạn được liên kết với cùng một quan hệ tổng thể R và được định vị tại cùng một trạm j cấu thành ảnh vật lý quan hệ tổng thể R tại trạm j ;

- Ta có thể ánh xạ một-một giữa một ảnh vật lý và một cặp (quan hệ tổng thể, trạm);

- Các ảnh vật lý có thể chỉ ra bằng tên của một quan hệ tổng thể và một chỉ mục trạm.

Ký hiệu

- R_i để chỉ đoạn thứ i của quan hệ tổng thể R ;

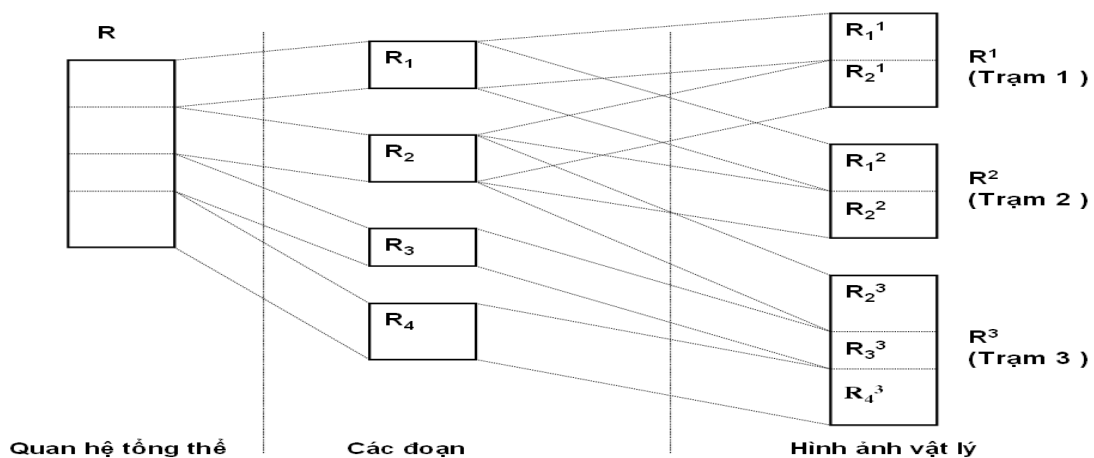
- R^j để chỉ ảnh vật lý của quan hệ tổng thể R tại trạm j ;

- Bản sao của đoạn i thuộc quan hệ R tại trạm j được ký hiệu là R_i^j

d. Sơ đồ ánh xạ địa phương (Local mapping schema):

- Thực hiện ánh xạ các ảnh vật lý lên các đối tượng được thực hiện bởi hệ quản trị cơ sở dữ liệu địa phương;

- Tất cả các đoạn của một quan hệ tổng thể trên cùng một trạm tạo ra một ảnh vật lý.



Các đoạn và hình ảnh vật lý của một quan hệ tổng thể

Hình 1.11. Sơ đồ ánh xạ địa phương

Ba yếu tố được suy ra từ kiểu kiến trúc này là:

- a. Tách rời khái niệm phân đoạn dữ liệu với khái niệm định vị dữ liệu;
- b. Biết được dữ liệu dư thừa;
- c. Độc lập với các DBMS địa phương.

Ba yếu tố này tương ứng với ba mức trong suốt tương ứng.

a. Tách rời khái niệm phân đoạn dữ liệu với khái niệm định vị dữ liệu

- **Phân đoạn dữ liệu**, bao gồm những công việc mà người lập trình ứng dụng làm việc với quan hệ tổng thể, phân chia quan hệ tổng thể thành các đoạn.

- Thông qua tính **trong suốt phân đoạn** (*fragmentation transparency*) người lập trình sẽ nhìn thấy được những đoạn dữ liệu bị phân chia như thế nào.

- **Định vị dữ liệu** lại liên quan đến các công việc của người sử dụng và người lập trình ứng dụng tại trên các đoạn dữ liệu được định vị tại các trạm.

- Thông qua tính **trong suốt vị trí** (*location transparency*) người lập trình sẽ biết được vị trí của các đoạn dữ liệu trên các trạm.

b. Biết được dữ liệu dư thừa

- Người lập trình ứng dụng có thể biết được dư thừa dữ liệu ở các trạm;

- Trên hình vẽ trên, chúng ta thấy rằng hai ảnh vật lý R2 và R3 có trùng lặp dữ liệu. Do đó các đoạn dữ liệu trùng nhau có thể tránh được khi xây dựng các khối ảnh vật lý.

c. Độc lập với các DBMS địa phương

Tính chất này còn được gọi là **trong suốt ánh xạ địa phương** (*local mapping transparency*), cho phép chúng ta khảo sát các vấn đề về quản lý CSDL phân tán mà không cần phải hiểu rõ mô hình dữ liệu của DBMS địa phương đang.

1.7. Các đặc điểm chính của hệ phân tán

1.7.1. Chia sẻ tài nguyên

- Được thực hiện qua mạng truyền thông;

- Mỗi tài nguyên cần phải được quản lý bởi một chương trình có giao diện truyền thông;

- Các tài nguyên có thể được truy nhập, cập nhật một cách tin cậy và nhất quán.

Quản lý tài nguyên bao gồm:

- Lập kế hoạch dự phòng;
- Đặt tên cho các lớp tài nguyên;
- Cho phép tài nguyên được truy nhập từ nơi này đến nơi khác;
- Ánh xạ tên tài nguyên vào địa chỉ truyền thông.

1.7.2. Tính mở

Tính mở của hệ thống phân tán là tính dễ dàng mở rộng phần cứng của nó. Một hệ thống được gọi là có tính mở thì phải có các điều kiện sau:

- Hệ thống có thể tạo nên bởi nhiều loại phần cứng và phần mềm của nhiều nhà cung cấp khác nhau;
- Có thể bổ sung vào các dịch vụ dùng chung tài nguyên mà không phá hỏng hay nhân đôi các dịch vụ đang tồn tại;
- Tính mở được hoàn thiện bằng cách xác định hay phân định rõ các giao diện chính của một hệ và làm cho nó tương thích với các nhà phát triển phần mềm;
- Tính mở của hệ phân tán dựa trên việc cung cấp cơ chế truyền thông giữa các tiến trình và công khai các giao diện dùng để truy nhập các tài nguyên chung.

1.7.3. Khả năng song song

- Hệ phân tán hoạt động trên một mạng truyền thông có nhiều máy tính, mỗi máy có thể có một hay nhiều CPU;
- Có thể thực hiện nhiều tiến trình trong cùng một thời điểm. Việc thực hiện tiến trình theo cơ chế phân chia thời gian (một CPU) hay (nhiều CPU).

Khả năng làm việc song song trong hệ phân tán được thể hiện qua hai tình huống sau:

- Nhiều người sử dụng đồng thời đưa ra các lệnh hay các tương tác với các chương trình ứng dụng;
- Nhiều tiến trình Server chạy đồng thời, mỗi tiến trình phải đáp ứng yêu cầu từ các Clients.

1.7.4. Khả năng mở rộng

- Khả năng mở rộng của một hệ phân tán được đặc trưng bởi tính không thay đổi phần mềm hệ thống và phần mềm ứng dụng khi hệ được mở rộng.

- Yêu cầu cho việc mở rộng không chỉ là mở rộng phần cứng, về mạng mà nó trải trên các khía cạnh khi thiết kế hệ phân tán.

Ví dụ: Tần suất sử dụng trên mạng tăng đột ngột. Để tránh tình trạng tắc nghẽn xảy ra khi chỉ có một Server và phải đáp ứng các yêu cầu truy nhập các file đó, người ta nhân bản các file trên một Server khác và hệ thống được thiết kế sao cho việc thêm Server được dễ dàng.

Một số giải pháp khác là sử dụng Cache và các bản sao dữ liệu.

1.7.5. Khả năng thứ lỗi

Việc thiết kế khả năng thứ lỗi các hệ thống máy tính dựa trên hai giải pháp sau:

- Dùng khả năng thay thế để đảm bảo sự hoạt động liên tục và hiệu quả;
- Dùng các chương trình hồi phục dữ liệu khi xảy ra sự cố.

1.7.6. Đảm bảo tin cậy và nhất quán

Hệ thống yêu cầu độ tin cậy như:

- Bí mật của dữ liệu;
- Các chức năng khôi phục hư hỏng phải đảm bảo;
- Ngoài ra các yêu cầu của hệ thống về tính nhất quán cũng thể hiện ở chỗ không có mâu thuẫn trong nội dung cơ sở dữ liệu.

1.8. Trong suốt phân tán

1.8.1. Tính trong suốt

Tính trong suốt của một hệ phân tán được hiểu như là việc che khuất đi các thành phần riêng biệt của hệ đối với người sử dụng và những người lập trình ứng dụng.

Các loại trong suốt trong hệ phân tán:

- *Trong suốt phân đoạn (fragmentation transparency);*
- *Trong suốt về vị trí (location transparency);*
- *Trong suốt ánh xạ địa phương (local mapping transparency);*
- *Không trong suốt (no transparency).*

1.8.2. Trong suốt phân đoạn (fragmentation transparency)

Khi dữ liệu đã được phân đoạn thì việc truy cập vào CSDL được thực hiện bình thường như là chưa bị phân tán và không ảnh hưởng tới người sử dụng.

Ví dụ: Xét quan hệ tổng thể NCC (Id, Tên, Tuổi) và các phân đoạn được tách ra từ nó:

NCC1 (Id, Tên, Tuổi);

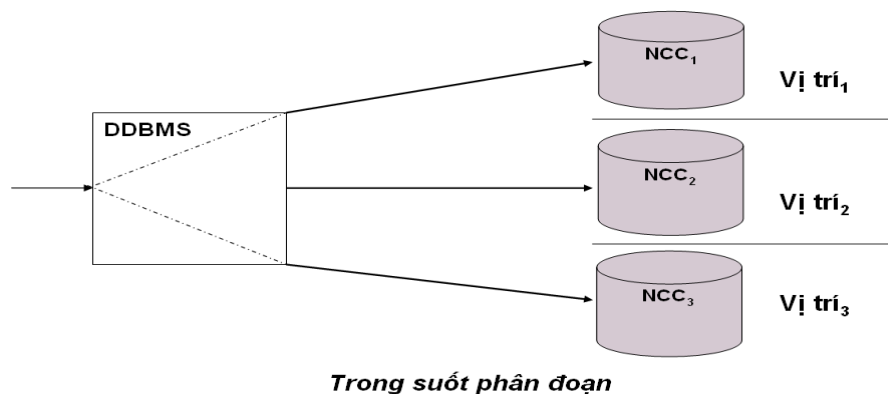
NCC2 (Id, Tên, Tuổi);

NCC3 (Id, Tên, Tuổi).

Giả sử DDBMS cung cấp tính trong suốt về phân đoạn, khi đó ta có thể thấy tính trong suốt này được thể hiện như sau:

Khi muốn tìm một người có **Id= “Id1”** thì chỉ cần tìm trên quan hệ tổng thể NCC mà không cần biết quan hệ NCC có phân tán hay không.

```
SELECT    *  
FROM      NCC  
WHERE     Id="Id1"
```



Hình 1.12. Hình ảnh trong suốt phân đoạn

1.8.3. Tính trong suốt về vị trí (location transparency)

- Người sử dụng không cần biết về vị trí vật lý của dữ liệu mà có quyền truy cập đến cơ sở dữ liệu tại bất cứ vị trí nào.

- Các thao tác để lấy hoặc cập nhật một dữ liệu từ xa được tự động thực hiện bởi hệ thống tại điểm đưa ra yêu cầu.

- Tính trong suốt về vị trí rất hữu ích, nó cho phép người sử dụng bỏ qua các bản sao dữ liệu đã tồn tại ở mỗi vị trí. Do đó có thể di chuyển một bản sao dữ liệu từ một vị trí này đến một vị trí khác và cho phép tạo các bản sao mới mà không ảnh hưởng đến các ứng dụng.

Ví dụ: Với quan hệ tổng thể R và các phân đoạn như đã nói ở trên nhưng giả sử rằng DDBMS cung cấp trong suốt về vị trí nhưng không cung cấp trong suốt về phân đoạn.

Xét câu truy vấn *tìm người có Id=“Id1”*.

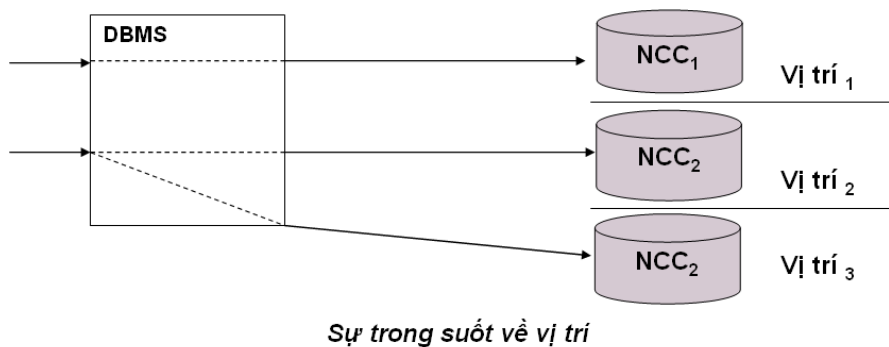
```

SELECT    *
FROM      NCC1
WHERE     Id="Id1"
IF NOT #FOUND THEN
SELECT    *
FROM      NCC2
WHERE     Id="Id1"

```

- Đầu tiên hệ thống sẽ thực hiện tìm kiếm ở phân đoạn NCC₁ và nếu DBMS trả về biến điều khiển #FOUND thì một câu lệnh truy vấn tương tự được thực hiện trên phân đoạn NCC₂ ,...

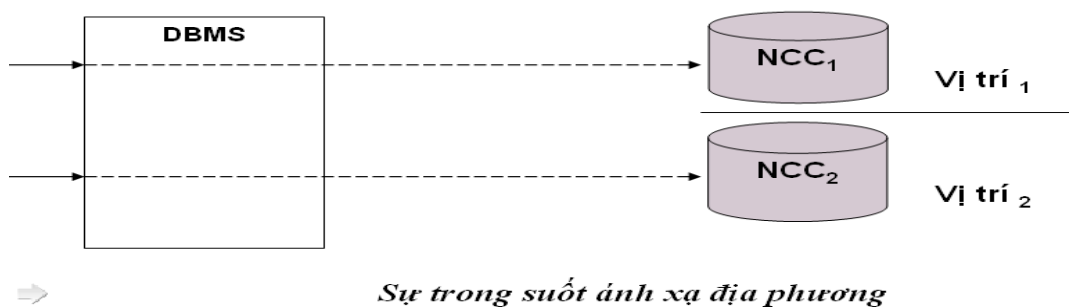
- Ở đây quan hệ NCC₂ được sao làm hai bản trên hai vị trí₂ và vị trí₃, ta chỉ cần tìm thông tin trên quan hệ NCC₂ mà không cần quan tâm nó ở vị trí nào.



Hình 1.13. Hình ảnh trong suốt về vị trí

1.8.4. Trong suốt ánh xạ địa phương (local mapping transparency)

- Là một đặc tính quan trọng trong một hệ thống DBMS không đồng nhất.
- Ứng dụng tham chiếu đến các đối tượng có các tên độc lập từ các hệ thống cục bộ địa phương.
- Ứng dụng được cài đặt trên một hệ thống không đồng nhất nhưng được sử dụng như một hệ thống đồng nhất.



Hình 1.14. Hình ảnh trong suốt ánh xạ địa phương

Danh mục tài liệu tham khảo

1. Phạm Thế Quế (2010). *Giáo trình Cơ sở dữ liệu phân tán*. Nhà xuất bản Thông tin và Truyền thông.
2. Nguyễn Bá Tường (2005). *Nhập môn cơ sở dữ liệu phân tán*. Nhà xuất bản Khoa học và Kỹ thuật.
3. Lê Tiến Vương (1996). *Nhập môn cơ sở dữ liệu quan hệ*. Nhà xuất bản Khoa học và Kỹ thuật.

Câu hỏi cuối chương

1. Ưu và nhược điểm của các loại mạng máy tính.
2. Trong mô hình client/Server, hãy cho biết mục đích của:
 - Trình diễn thông tin phân tán;
 - Trình diễn từ xa;
 - Quản lý dữ liệu từ xa;
 - Phân tán chức năng.
3. Định nghĩa cơ sở dữ liệu phân tán. Hệ quản trị CSDL phân tán. Các thành phần yêu cầu trong một DDBMSs thương mại. Cho ví dụ về một CSDL phân tán.
4. So sánh CSDL phân tán và CSDL tập trung.
5. Ưu và nhược điểm của hệ phân tán.
6. Các truy xuất từ xa trong hệ thống phân tán.
7. Nêu kiến trúc của cơ sở dữ liệu phân tán.
8. Thế nào là ảnh vật lý? Cho ví dụ.
9. Thế nào là trong suốt vị trí, trong suốt phân đoạn và trong suốt ánh xạ địa phương? Lợi ích của các kiểu trong suốt đó.

Chương 2

THIẾT KẾ CSDL PHÂN TÁN

2.1. Nội dung thiết kế một hệ thống phân tán

Yêu cầu:

- Phải qua bước phân tích trước khi thiết kế. Các bước này phải độc lập với mọi giải pháp cài đặt;

- Chọn những vị trí để cài đặt dữ liệu và các chương trình trên mạng máy tính;
- Đối với DBMS phân tán, việc phân tán các ứng dụng đòi hỏi hai điều:

1. Phân tán DBMS;
2. Phân tán các chương trình ứng dụng chạy trên DBMS đó.

Nhận xét:

- Có nhiều điểm tương đồng với việc thiết kế hệ thống tập trung;

- Điều khác nhau cơ bản là hệ thống được phân bố trên một số địa điểm khác nhau;

- Tính khả thi, chu kỳ sống, tính mở, tính sẵn sàng...
- Thiết kế phần cứng: máy trạm, máy chủ, mạng...

2.1.1. Các công việc cần phải làm để thiết kế HT phân tán

- Xác định kiến trúc mô hình phân tán tổng thể;

- Định vị các địa phương cần phân tán, loại hình phân tán sử dụng cho mỗi địa phương (toàn bộ, bản sao, lai...);

- Tiến hành cân đối các yếu tố được phân tán bao gồm các phần tử dữ liệu và các hoạt động xử lý trên mỗi trạm;

- Thiết kế cơ sở dữ liệu phân tán;
- Thiết kế các chương trình ứng dụng.

2.1.2. Các sản phẩm yêu cầu sau khi phân tích thiết kế

1. Mô tả các trạm

- Thông tin địa lý;
- Thiết bị vật lý;
- Thông tin hạ tầng;
- Đặc trưng về con người (trình độ, kỹ năng...).

2. Mô tả về sử dụng dữ liệu cho mỗi trạm

- Các phần tử dữ liệu sử dụng từ hệ thống;
- Các phần tử dữ liệu cần phải tạo ra;
- Các phần tử dữ liệu cập nhật;
- Các phần tử dữ liệu xóa.

3. Mô tả quá trình nghiệp vụ cho mỗi trạm

- Danh sách các xử lý (sơ đồ chức năng) ở các trạm;
- Mô tả các xử lý.

4. Các thỏa thuận về phương án kiến trúc hệ thống cho mỗi trạm: cho nhu cầu về dữ liệu và xử lý cho trạm đó

- Có cần hay không về các trợ giúp không phải kỹ thuật;
- Có cần hay không về hệ thống địa phương, về nối mạng;
- Có cần hay không về các cấu hình phân tán khác.

2.2. Các chiến lược phân tán dữ liệu

- Việc định vị và phân tán dữ liệu ở các nút trong một mạng máy tính sẽ quyết định tính hiệu quả và đúng đắn của hệ thống phân tán.

- Có 4 chiến lược phân tán dữ liệu cơ bản: tập trung dữ liệu, chia nhỏ dữ liệu, sao lập dữ liệu và phương thức lai.

2.2.1. Tập trung dữ liệu

Tất cả các dữ liệu được tập trung một chỗ. Cách này đơn giản nhưng có 3 nhược điểm:

- Dữ liệu không sẵn sàng cho người sử dụng truy nhập từ xa;
- Chi phí truyền thông lớn, thường làm cực đại việc truy nhập dữ liệu tới nơi tập trung;
- Toàn bộ hệ thống ngừng khi cơ sở dữ liệu bị sự cố.

2.2.2. Chia nhỏ dữ liệu

- Cơ sở dữ liệu được chia thành các phần nhỏ liên kết nhau (không trùng lặp);.
- Mỗi phần dữ liệu được đưa đến các trạm một cách thích hợp để sử dụng.

2.2.3. Sao lập dữ liệu

- CSDL được nhân thành nhiều bản *từng phần* hoặc *đầy đủ* và được đặt ở nhiều trạm trên mạng.

- Nếu bản sao của CSDL được lưu giữ tại mọi trạm của hệ thống ta có trường hợp sao lập đầy đủ.

- Hiện nay có nhiều kỹ thuật mới cho phép tạo bản sao không đầy đủ phù hợp với yêu cầu dữ liệu ở mỗi trạm và một bản đầy đủ được quản lý ở server.

- Sau một khoảng thời gian nhất định các bản sao được làm đồng bộ với bản chính bằng một ứng dụng nào đó.

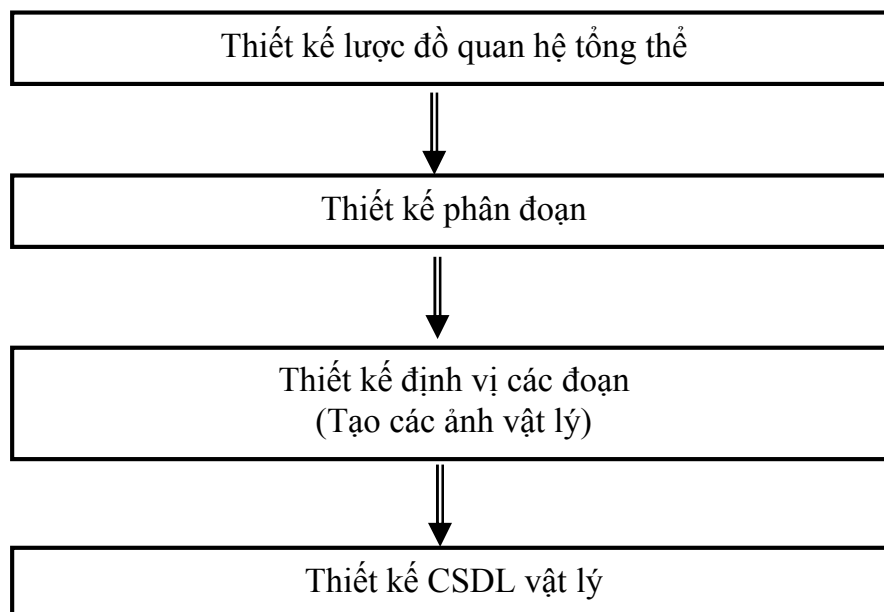
2.2.4. Phương thức lai

- Cơ sở dữ liệu được phân thành nhiều phần: quan trọng và không quan trọng;
- Phần ít quan trọng được lưu giữ một nơi;
- Phần quan trọng được lưu trữ ở nhiều nơi khác.

2.3. Phương pháp thiết kế CSDL phân tán

2.3.1. Sơ đồ thiết kế tổng thể cơ sở dữ liệu phân tán

Hiện nay chưa có một kỹ thuật cụ thể nào nói một cách chi tiết việc thiết kế một CSDL phân tán. Tuy nhiên, một cách tổng quát chúng ta có thể thiết kế CSDL phân tán theo các bước sau:



Hình 2.1. Sơ đồ thiết kế tổng thể

1. Thiết kế lược đồ quan hệ tổng thể

- Thiết kế các quan hệ tổng thể;
- Mô tả toàn bộ dữ liệu sẽ được dùng trong hệ thống.

2. Thiết kế phân đoạn: thực hiện chia nhỏ dữ liệu thành các phần.

3. Thiết kế định vị các đoạn

- Là quá trình thực hiện ánh xạ các đoạn vào các trạm khác nhau;

- Tạo các ảnh vật lý tại các trạm;
- Các đoạn dữ liệu được đưa vào các vị trí lưu trữ thích hợp với yêu cầu hoạt động thực tế của hệ thống.

4. Thiết kế cơ sở dữ liệu vật lý: thiết kế dữ liệu vật lý cho các quan hệ tại các trạm.

2.3.2. Các phương pháp thiết kế CSDL phân tán

Có 2 phương pháp thiết kế CSDL phân tán:

- Phương pháp tiếp cận từ trên xuống;
- Phương pháp tiếp cận từ dưới lên.

a. Phương pháp thiết kế từ trên xuống

- Thiết kế từ tổng thể đến riêng biệt;
- Phân rã một hệ thống lớn thành các hệ thống con;
- Phân tích các yêu cầu nhằm định nghĩa môi trường hệ thống;
- Thu thập các yêu cầu về dữ liệu và nhu cầu xử lý của các trạm có sử dụng CSDL.

• **Thiết kế view:** xây dựng khung nhìn dữ liệu cho người sử dụng ở các trạm.

• **Thiết kế mức quan niệm:** là một tiến trình kiểm tra và xác định rõ hai nhóm quan hệ: *phân tích thực thể* và *phân tích chức năng*.

+ *Phân tích thực thể:* xác định các tập thực thể, các thuộc tính và các mối quan hệ giữa chúng.

+ *Phân tích chức năng:* xác định các chức năng của hệ thống và đưa ra các chức năng cơ sở.

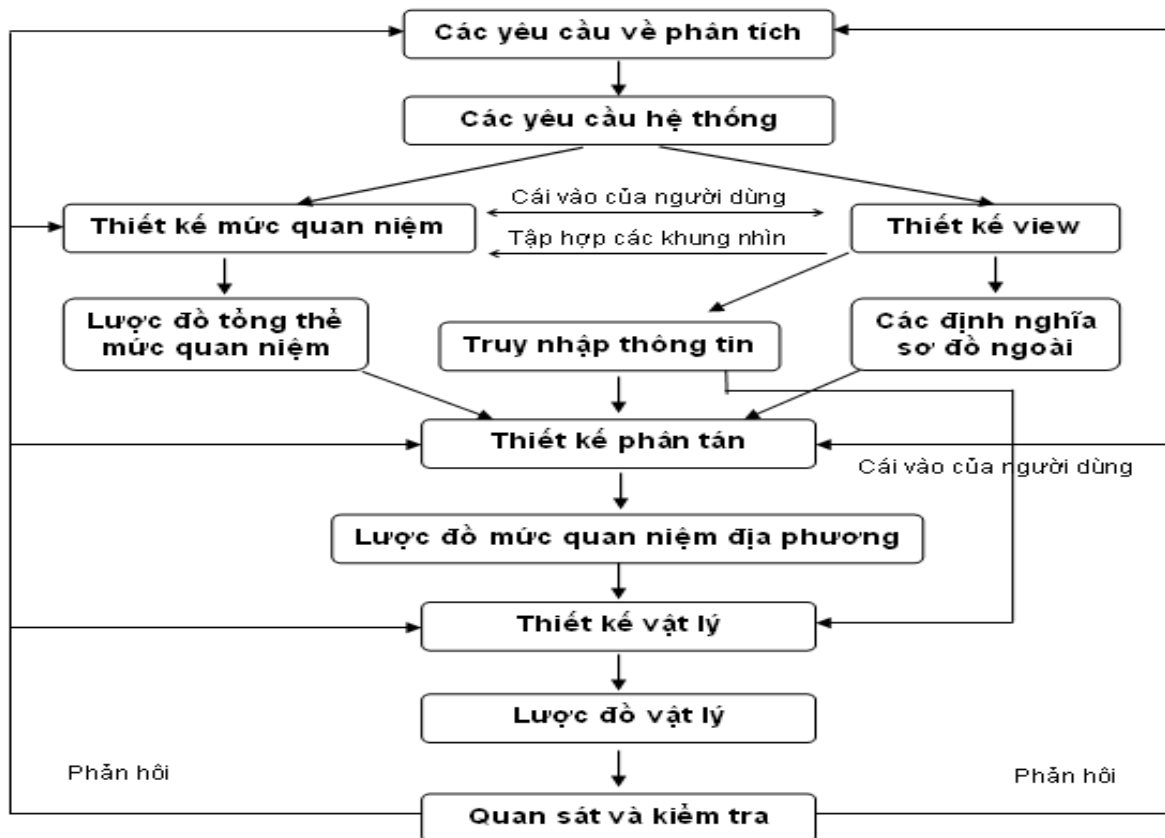
• **Thiết kế phân tán** bao gồm hai phần:

- + *Thiết kế phân đoạn;*
- + *Thiết kế định vị.*

• **Thiết kế lược đồ quan niệm địa phương:** tạo ra các lược đồ mức quan niệm tại các địa phương.

• **Thiết kế vật lý:** thực hiện ánh xạ lược đồ mức quan niệm tại các địa phương ra các đơn vị lưu trữ vật lý.

• **Quan sát và kiểm tra:** kiểm tra các giai đoạn của quá trình thiết kế cơ sở dữ liệu.



Sơ đồ thiết kế CSDL phân tán theo mô hình trên xuống

Hình 2.2. Sơ đồ thiết kế cơ sở dữ liệu phân tán từ trên xuống

b. Phương pháp thiết kế từ dưới lên

- Phương pháp thiết kế trên xuống thực sự có hiệu quả khi xây dựng một hệ thống mới.

- Trong thực tế, một số CSDL đã tồn tại trước, được tổ chức trong môi trường tập trung và CSDL phân tán được phát triển bằng cách liên kết chúng lại thành một CSDL mới thống nhất (các DBMS địa phương khác nhau đã được sử dụng).

Cách thiết kế

1. Chọn một mô hình dữ liệu chung để mô tả lược đồ tổng thể.
2. Chuyển mỗi lược đồ địa phương theo mô hình dữ liệu chung đã chọn.
3. Tích hợp các lược đồ địa phương vào lược đồ tổng thể.

2.4. Phân mảnh dữ liệu

Những câu hỏi đặt ra đối với phân mảnh dữ liệu:

1. Tại sao cần phải phân mảnh?
2. Làm thế nào để thực hiện phân mảnh?

3. Phân mảnh nên thực hiện đến mức độ nào?
4. Có cách gì kiểm tra tính đúng đắn của việc phân mảnh?
5. Việc cấp phát các mảnh dữ liệu như thế nào?
6. Những thông tin nào sẽ cần thiết cho việc phân mảnh và cấp phát?

Các phép toán về đại số quan hệ cần sử dụng:

- Phép hợp;
- Phép trừ;
- Phép tích Decartes;
- Phép chọn;
- Phép chiếu;
- Phép nối (q-nối, nối bằng, nối tự nhiên);
- Phép nửa nối.

SQL

Sơ lược về ngôn ngữ SQL (Structured query language)

- SQL trước kia được gọi là SEQUEL;
- IBM phát triển ở San Jose;
- Là một ngôn ngữ phi thủ tục;
- Mục đích để sử dụng trong CSDL thử nghiệm System R.

Câu lệnh SELECT

Cú pháp:

SELECT $R_1.A_1, R_2.A_2, \dots, R_i.A_i$

FROM R_1, R_2, \dots, R_k

WHERE Ψ

Ý nghĩa:

$\Pi_{R_1.A_1, R_2.A_2, \dots, R_i.A_i}(\sigma_{\Psi}(R_1 \times \dots \times R_k))$

Phân mảnh quan hệ là gì?

Việc chia một quan hệ thành nhiều quan hệ nhỏ hơn được gọi là phân mảnh quan hệ.

2.4.1. Các lý do phân mảnh

Khung nhìn hoặc đơn vị truy xuất của các ứng dụng không phải là toàn bộ quan hệ mà thường là một mảnh.

Việc phân rã một quan hệ thành nhiều mảnh, mỗi mảnh được xử lý như một đơn vị, sẽ cho phép thực hiện nhiều giao dịch đồng thời.

Việc phân mảnh các quan hệ sẽ cho phép thực hiện song song một câu vấn tin bằng cách chia nó ra thành một tập các câu vấn tin con hoạt tác trên các mảnh.

Nếu các ứng dụng có các khung nhìn được định nghĩa trên một quan hệ cho trước nằm tại những vị trí khác thì có hai cách chọn lựa đơn vị phân tán:

- Hoặc là toàn bộ quan hệ;
- Hoặc quan hệ được lưu ở một vị trí có chạy ứng dụng.

Nhận xét: Chọn lựa thứ nhất gây ra một số lượng lớn các truy xuất không cần thiết đến dữ liệu ở xa. Chọn lựa sau sẽ gây ra nhiều vấn đề khi cập nhật và lãng phí không gian lưu trữ.

Hạn chế của việc phân mảnh:

- Nếu ứng dụng có những yêu cầu ngăn cản việc phân rã thành các mảnh để được sử dụng độc quyền, thì những ứng dụng có các khung nhìn được định nghĩa trên nhiều mảnh sẽ bị giảm hiệu suất hoạt động.

- Nếu một khung nhìn đòi hỏi thông tin ở nhiều mảnh thì việc truy xuất dữ liệu để nối lại sẽ có chi phí cao.

- Kiểm soát dữ liệu ngữ nghĩa(semantic data control): Do kết quả của phân mảnh, các thuộc tính tham gia vào một phụ thuộc có thể bị phân rã vào các mảnh khác nhau và được cấp phát cho những vị trí khác nhau. Trong trường hợp này, một nhiệm vụ đơn giản như kiểm tra các phụ thuộc cũng phải thực hiện truy tìm dữ liệu ở nhiều vị trí.

2.4.2. Các kiểu phân mảnh

- Phân mảnh ngang (horizontal fragmentation);
- Phân mảnh dọc (vertical fragmentation);
- Phân mảnh hỗn hợp (hibrid fragmentation).

Chú ý: Quá trình phân mảnh phải được gắn liền với vấn đề cấp phát dữ liệu và bài toán cụ thể như thế nào.

2.4.2.1. Phân mảnh ngang

Phân mảnh ngang một quan hệ tổng thể n-bộ R là tách R thành các quan hệ con n-bộ R_1, R_2, \dots, R_k sao cho quan hệ R có thể được khôi phục lại từ các quan hệ con này bằng phép hợp: $R = R_1 \cup R_2 \cup \dots \cup R_k$.

Có hai loại phân mảnh ngang:

Phân mảnh ngang nguyên thủy (primary horizontal fragmentation): phân mảnh ngang nguyên thủy của một quan hệ được thực hiện dựa trên các vị từ được định nghĩa trên quan hệ đó.

Phân mảnh ngang dẫn xuất (derived horizontal fragmentation): phân mảnh ngang dẫn xuất của một quan hệ được thực hiện dựa trên các vị từ được định nghĩa trên quan hệ khác.

Như vậy, trong phân mảnh ngang tập các vị từ đóng một vai trò quan trọng.

2.4.2.2. Phân mảnh dọc

Phân mảnh dọc một quan hệ tổng thể n-bộ R là tách R thành các quan hệ con R_1, R_2, \dots, R_k sao cho quan hệ R có thể được khôi phục lại từ các quan hệ con này bằng phép nối:

$$R = R_1 \bowtie R_2 \bowtie \dots \bowtie R_k$$

2.4.2.3. Phân mảnh hỗn hợp

Là kết hợp cả phân mảnh ngang và phân mảnh dọc.

2.4.3. Các yêu cầu của việc phân mảnh

Việc phân mảnh một quan hệ tổng thể cũng phải tuân theo một số quy tắc nhất định để khi tái thiết lại quan hệ cũ vẫn bảo đảm ngữ nghĩa của nó.

Một phương pháp thiết kế các phân mảnh đúng đắn phải thỏa mãn ba tính chất sau:

- *Tính đầy đủ (completeness);*
- *Tính tái thiết được (reconstruction);*
- *Tính tách biệt (disjointness).*

a. Tính đầy đủ

Nếu một quan hệ R được phân rã thành các mảnh R_1, R_2, \dots, R_k thì mỗi mục dữ liệu có trong R phải có trong ít nhất một mảnh R_i nào đó.

b. Tính tái thiết được

Nếu một quan hệ R được phân rã thành các mảnh R_1, R_2, \dots, R_k thì phải tồn tại một toán tử θ sao cho $R = \theta(R_i), \forall i$.

- Toán tử θ thay đổi tùy theo từng loại phân mảnh.
- Trong thực tế khi các mảnh được phân mảnh ngang thì θ là phép hợp, phân mảnh dọc thì θ là phép nối và phân mảnh hỗn hợp thì θ là phép nửa nối.

c. Tính tách biệt

Nếu một quan hệ R được phân mảnh ngang thành các quan hệ R_1, R_2, \dots, R_k và mục dữ liệu t_i nằm trong mảnh R_i thì nó sẽ không nằm trong một mảnh $R_k, k \neq i$.

- Tiêu chuẩn này bảo đảm các mảnh ngang phải được tách rời nhau.

- Nếu quan hệ được phân mảnh dọc thì thuộc tính chung phải được lặp lại trong mỗi mảnh. Do đó, trong trường hợp phân mảnh dọc tính tách biệt chỉ được định nghĩa trên các trường không phải là thuộc tính chung của quan hệ.

Ví dụ: Xét cơ sở dữ liệu của một công ty máy tính được tổ chức như sau:

NHANVIEN (MANV, TENNV, CHUCVU): quan hệ này chứa dữ liệu về nhân viên của công ty.

TLUONG (CHUCVU, LUONG): quan hệ này chứa dữ liệu liên quan về lương và chức vụ của nhân viên.

DUAN (MADA, TENDA, NGANSACH): quan hệ này chứa dữ liệu về các dự án mà công ty đang phát triển.

HOSO (MANV, MADA, NHIEMVU, THOIGIAN): quan hệ này chứa dữ liệu về hồ sơ của nhân viên được phân công thực hiện dự án).

NHANVIEN (E)

MANV	TENNV	CHUCVU
A1	Nam	Phân tích HT
A2	Trung	Lập trình viên
A3	Đông	Phân tích HT
A4	Bắc	Phân tích HT
A5	Tây	Lập trình viên
A6	Hùng	Kỹ sư điện
A7	Dũng	Phân tích HT
A8	Chiên	Thiết kế DL

HOSO (G)

MANV	MADA	NHIEMVU	THOIGIAN
A1	D1	Quản lý	12
A2	D1	Phân tích	34
A2	D2	Phân tích	6
A3	D3	Kỹ thuật	12
A3	D4	Lập trình	10
A4	D2	Quản lý	6
A5	D2	Quản lý	20
A6	D4	Kỹ thuật	36
A7	D3	Quản lý	48
A8	D3	Lập trình	15

DUAN (J)

MADA	TENDA	NGANSACH
D1	CSDL	20000
D2	CÀI ĐẶT	12000
D3	BẢO TRÌ	28000
D4	PHÁT TRIỂN	25000

TLUONG (S)

CHUCVU	LUONG
Kỹ sư điện	1000
Phân tích HT	2500
Lập trình viên	3000
Thiết kế DL	4000

Ví dụ về phân mảnh ngang: xét các phép toán đại số quan hệ sau:

$$\text{DUAN 1} = \sigma_{\text{NGANSACH} \leq 20000} (\text{DUAN})$$

$$\text{DUAN 2} = \sigma_{\text{NGANSACH} > 20000} (\text{DUAN})$$

DUAN 1

MADA	TENDA	NGANSACH
D1	CSDL	20000
D2	CÀI ĐẶT	12000

DUAN 2

MADA	TENDA	NGANSACH
D3	BẢO TRÌ	28000
D4	PHÁT TRIỂN	25000

Dễ thấy, các mảnh thỏa mãn *tính tái thiết được* và *tính đầy đủ*

Ví dụ về phân mảnh dọc: xét các phép toán đại số quan hệ sau:

$$\text{DUAN 3} = \Pi_{\{1,3\}} \text{DUAN}; \quad \text{DUAN 4} = \Pi_{\{1,4\}} \text{DUAN}$$

DUAN 3

MADA	NGANSACH
D1	20000
D2	12000
D3	28000
D4	25000

DUAN 4

MADA	TENDA
D1	CSDL
D2	CÀI ĐẶT
D3	BẢO TRÌ
D4	PHÁT TRIỂN

Dễ thấy, các mảnh thỏa mãn *tính tái thiết được* và *tính đầy đủ*

$$\text{DUAN 3} \subseteq \text{DUAN}; \quad \text{DUAN 4} \subseteq \text{DUAN}; \quad \text{DUAN} = \text{DUAN 3} \bowtie \text{DUAN 4}$$

Ví dụ về phân mảnh hỗn hợp: xét các phép toán đại số quan hệ sau:

$$\text{DUAN 5} = \sigma_{\text{NGANSACH} \leq 20000} (\Pi_{\{1,3\}} \text{DUAN})$$

$$\text{DUAN 6} = \sigma_{\text{NGANSACH} > 20000} (\Pi_{\{1,3\}} \text{DUAN})$$

DUAN 5

TENDA	NGANSACH
CSDL	20000
CÀI ĐẶT	12000

DUAN 6

TENDA	NGANSACH
CSDL	28000
CÀI ĐẶT	25000

2.5. Cấp phát tài nguyên trong hệ phân tán

2.5.1. Bài toán cấp phát (allocation problem)

Giả sử có một tập các mảnh $F = \{F_1, F_2, \dots, F_k\}$ và một mạng máy tính bao gồm các vị trí $S = \{S_1, S_2, \dots, S_m\}$ trên đó có một tập các ứng dụng $Q = \{Q_1, Q_2, \dots, Q_q\}$ đang thực thi.

Hãy tìm một phân phối tối ưu các mảnh F cho các vị trí S .

Một phân phối được gọi là tối ưu nếu thỏa mãn hai yếu tố sau:

Chi phí nhỏ nhất: Hàm chi phí bao gồm chi phí lưu mỗi mảnh dữ liệu F_i tại vị trí S_j , chi phí vận tin F_i tại vị trí S_j , chi phí cập nhật F_i tại tất cả các vị trí có chứa nó, và chi phí truyền dữ liệu. Vì thế bài toán cấp phát sẽ tìm một lược đồ cấp phát với hàm chi phí là cực tiểu.

Hiệu quả: Chiến lược cấp phát được thiết kế nhằm cực tiểu hóa thời gian thực hiện và tăng tối đa lưu lượng hệ thống tại mỗi vị trí.

Bài toán cấp phát tổng quát, ký hiệu DAP (database allocation problem), là một bài toán NP-đầy đủ. Vì thế hầu hết các nghiên cứu đã được dành cho việc tìm ra được các thuật giải heuristic để có được lời giải tối ưu cho loại bài toán này.

Hiện nay chưa có một mô hình heuristic tổng quát nào nhận một tập các mảnh và sinh ra một chiến lược cấp phát gần tối ưu ứng với các ràng buộc cho trước mà chỉ mới đưa ra một số giả thiết đơn giản hóa và dễ áp dụng cho một số cách đặt vấn đề đơn giản.

2.5. Thông tin cấp phát

Ở giai đoạn cấp phát, chúng ta cần các thông tin định lượng về cơ sở dữ liệu, về các ứng dụng chạy trên đó, về cấu trúc mạng, về khả năng xử lý và giới hạn lưu trữ của mỗi vị trí trên mạng. Bao gồm các thông tin:

- Thông tin về cơ sở dữ liệu;
- Thông tin về ứng dụng;
- Thông tin về vị trí;
- Thông tin về mạng.

Danh mục tài liệu tham khảo

1. Phạm Thế Quế (2004). *Giáo trình Cơ sở dữ liệu: Lý thuyết và thực hành*. Nhà xuất bản Bưu điện, Hà Nội.
2. Phạm Thế Quế (2010). *Giáo trình Cơ sở dữ liệu phân tán*. Nhà xuất bản Thông tin và Truyền thông, Hà Nội.
3. Nguyễn Bá Tường (2005). *Nhập môn cơ sở dữ liệu phân tán*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
4. Lê Tiến Vương (1996). *Nhập môn cơ sở dữ liệu quan hệ*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
5. M.Tamer Ozsü And Patrice Valduriez (1999). *Principles of Distributed Database Systems*, Second Edition. Prentice Hall Upper Saddle River.

Câu hỏi và bài tập cuối chương

Cho Cơ sở dữ liệu của một công ty máy tính như sau:

NHANVIEN (E)

MANV	TENNV	CHUCVU
A1	Nam	Phân tích HT
A2	Trung	Lập trình viên
A3	Đông	Phân tích HT
A4	Bắc	Phân tích HT
A5	Tây	Lập trình viên
A6	Hùng	Kỹ sư điện
A7	Dũng	Phân tích HT
A8	Chiến	Thiết kế DL

HOSO (G)

MANV	MADA	NHIEMVU	THOIGIAN
A1	D1	Quản lý	12
A2	D1	Phân tích	34
A2	D2	Phân tích	6
A3	D3	Kỹ thuật	12
A3	D4	Lập trình	10
A4	D2	Quản lý	6
A5	D2	Quản lý	20
A6	D4	Kỹ thuật	36
A7	D3	Quản lý	48
A8	D3	Lập trình	15

DUAN (J)

MADA	TENDA	NGANSACH
D1	CSDL	20000
D2	CÀI ĐẶT	12000
D3	BẢO TRÌ	28000
D4	PHÁT TRIỂN	25000

TLUONG (S)

CHUCVU	LUONG
Kỹ sư điện	1000
Phân tích HT	2500
Lập trình viên	3000
Thiết kế DL	4000

Bài 1:

SƠ ĐỒ PHÂN MẢNH:

- Quan hệ DUAN được phân thành 3 mảnh:

Phân mảnh dọc thành các quan hệ DUAN1(MADA, TENDA, DIADIEM) và DUAN2 gồm các mảnh còn lại. DUAN2 lại được phân mảnh theo mức ngân sách ≥ 20.000 và ngân sách nhỏ hơn 20.000 (thành 2 quan hệ DUAN3, DUAN4)

- Quan hệ HOSO được phân thành 2 mảnh (Hoso1, Hoso2) theo thời gian ≥ 1 năm và nhỏ hơn 1 năm.

- Quan hệ NHANVIEN được phân mảnh theo các chức vụ (4 mảnh: NV1, NV2, NV3, NV4)

Có các trạm làm việc Site 1, Site 2, Site 3.

Giả sử cho sơ đồ định vị sau:

Tại Site 1: DUAN1, 3; HOSO1, NV1, NV2, TLUONG

Tại site 2: DUAN1, 4; HOSO2, NV3, NV4

Tại Site 3: DUAN1, 2, 3; HOSO2, NV1, NV4, TLUONG

Yêu cầu:

1. Vẽ đồ thị phân mảnh.
2. Vẽ ảnh vật lý của quan hệ tổng thể.
3. Viết truy vấn để tạo ra các đoạn và truy vấn để tái thiết lại các quan hệ tổng thể từ các mảnh đã bị phân mảnh bằng đại số quan hệ và Sql.
4. Giả sử hệ thống cung cấp tính trong suốt phân đoạn, hãy tìm những nhân viên làm việc tại dự án có ngân sách lớn hơn 30.000.
5. Giả sử hệ thống cung cấp trong suốt về vị trí nhưng không trong suốt phân đoạn, hãy viết truy vấn để tìm tên các nhân viên có chức vụ là “Phân tích HT”.

Bài 2:

1. Cho quan hệ NHANVIEN như trên,
gọi $p1: CHUCVU < \text{“Phân tích HT”}$ và $p2: CHUCVU > \text{“Phân tích HT”}$ là hai vị từ đơn giản. Giả sử chuỗi ký tự được sắp theo thứ tự chữ cái.
 - a) Thực hiện phân mảnh ngang cho quan hệ NHANVIEN ứng với $\{p1, p2\}$ thành NV1, NV2.
 - b) Giải thích tại sao phân mảnh kết quả NV1, NV2 không đáp ứng được quy tắc đúng đắn của phân mảnh.
 - c) Sửa lại $p1$ và $p2$ để chúng đáp ứng được quy tắc của phân mảnh.
2. Cho quan hệ TLUONG như trên,
gọi $p3: LUONG < 30000$ và $p4: LUONG \geq 30000$ là hai vị từ đơn giản.
 - a) Hãy thực hiện phân mảnh ngang cho TLUONG ứng với những vị từ này để có được hai mảnh TLUONG1 và TLUONG2.
 - b) Sử dụng phân mảnh của TLUONG để thực hiện phân mảnh ngang dẫn xuất cho NHANVIEN.
 - c) Chứng tỏ tính đầy đủ, tính tái thiết được và tính tách biệt của phân mảnh cho NHANVIEN.
3. Giả sử hệ thống cung cấp trong suốt phân đoạn, hãy tìm nhân viên làm việc ở dự án có ngân sách cao nhất.
4. Giả sử NHANVIEN không bị phân mảnh, TLUONG phân mảnh thành TLUONG1 và TLUONG2.

Giả sử hệ thống cung cấp trong suốt định vị, không cung cấp tính trong suốt phân đoạn.

Hãy tìm nhân viên có mức lương 3.000.

5. Làm lại câu 4 nếu NHANVIEN bị phân thành 2 mảnh NV1 và NV2; TLUONG phân mảnh thành TLUONG1 và TLUONG2.

Bài 3: Cơ sở dữ liệu gồm các quan hệ tổng thể sau:

- NHANVIEN (MaNV, HoTen, NgSinh, Luong, MaDV): Chứa thông tin về nhân viên: mã nhân viên, họ tên, ngày sinh, lương và mã đơn vị.

- DONVI (MaDV, TenDV, DienThoai, NguoiQuanLy): Chứa thông tin về đơn vị gồm: mã đơn vị, tên đơn vị, số điện thoại và người quản lý.

- NHACC (MaNCC, TenNCC, ThanhPho): Chứa thông tin về nhà cung cấp, gồm Mã nhà cung cấp, tên nhà cung cấp, thành phố.

- CUNGCAP (MaNCC, MaTB, MaDV, SoLuong): chứa thông tin về cung cấp thiết bị, gồm: Mã nhà cung cấp, Mã thiết bị, Mã đơn vị, số lượng

1. Giả sử quan hệ NHANVIEN được phân mảnh thành NV1, NV2 theo các vị từ p1: year(NgSinh)>1980 và p2: year(NgSinh)<=1980).

- Quan hệ NV2 lại được phân mảnh thành NV3, NV4 theo các vị từ p3: tên đơn vị="Hà Nội" và p4: tên đơn vị <>"Hà Nội".

- Hãy viết các truy vấn SQL và đại số qua hệ để tạo các mảnh trên. Chứng tỏ quá trình phân mảnh này đảm bảo tính tái thiết được.

2. Giả sử hệ thống cung cấp trong suốt phân đoạn, hãy tìm những nhân viên làm việc ở những đơn vị có nhà cung cấp cung cấp nhiều thiết bị nhất.

3. Giả sử quan hệ CUNGCAP được phân thành 2 mảnh thành CC1 và CC2 theo các vị từ p1: SoLuong>1000 và p2: SoLuong<=1000.

- Hãy phân mảnh ngang dẫn xuất quan hệ NHACC theo quan hệ CUNGCAP.

4. Các mảnh được phân như câu 1 và câu 3. Giả sử hệ thống cung cấp trong suốt định vị, không trong suốt phân đoạn, hãy tìm những nhân viên làm việc ở những đơn vị có cung cấp số lượng thiết bị > 500.

Chương 3

XỬ LÝ TRUY VẤN TRONG CSDL PHÂN TÁN

3.1. Giới thiệu về xử lý truy vấn

Mục đích của xử lý truy vấn:

- Giảm thiểu thời gian xử lý;
- Giảm vùng nhớ trung gian;
- Giảm chi phí truyền thông giữa các trạm.

Chức năng của xử lý truy vấn:

- Biến đổi một truy vấn ở mức cao thành một truy vấn tương đương ở mức thấp hơn;
- Phép biến đổi này phải đạt được cả về tính đúng đắn và hiệu quả;
- Mỗi cách biến đổi dẫn đến việc sử dụng tài nguyên máy tính khác nhau, nên vấn đề đặt ra là lựa chọn phương án nào dùng tài nguyên ít nhất.

Các phương pháp xử lý truy vấn cơ bản

•Phương pháp biến đổi đại số:

Đơn giản hóa câu truy vấn nhờ các phép biến đổi đại số tương đương nhằm giảm thiểu thời gian thực hiện các phép toán, phương pháp này không quan tâm đến kích thước và cấu trúc dữ liệu.

•Phương pháp ước lượng chi phí:

Xác định kích thước dữ liệu, thời gian thực hiện mỗi phép toán trong câu truy vấn. Phương pháp này phải xác định kích thước dữ liệu và chi phí thời gian thực hiện mỗi phép toán trong câu truy vấn.

3.2. Xử lý truy vấn trong môi trường tập trung

Tại sao phải nghiên cứu xử lý truy vấn tập trung?

Nghiên cứu xử lý truy vấn tập trung để hiểu được các kỹ thuật tối ưu phân tán vì ba lý do:

- Thứ nhất, câu truy vấn phân tán phải được dịch thành các câu truy vấn cục bộ, và được xử lý theo phương pháp tập trung;
- Thứ hai, các kỹ thuật tối ưu hoá phân tán thường là các mở rộng của kỹ thuật tập trung;
- Cuối cùng, tối ưu hoá tập trung thường đơn giản.

3.2.1. So sánh xử lý truy vấn tập trung và phân tán

Tập trung:

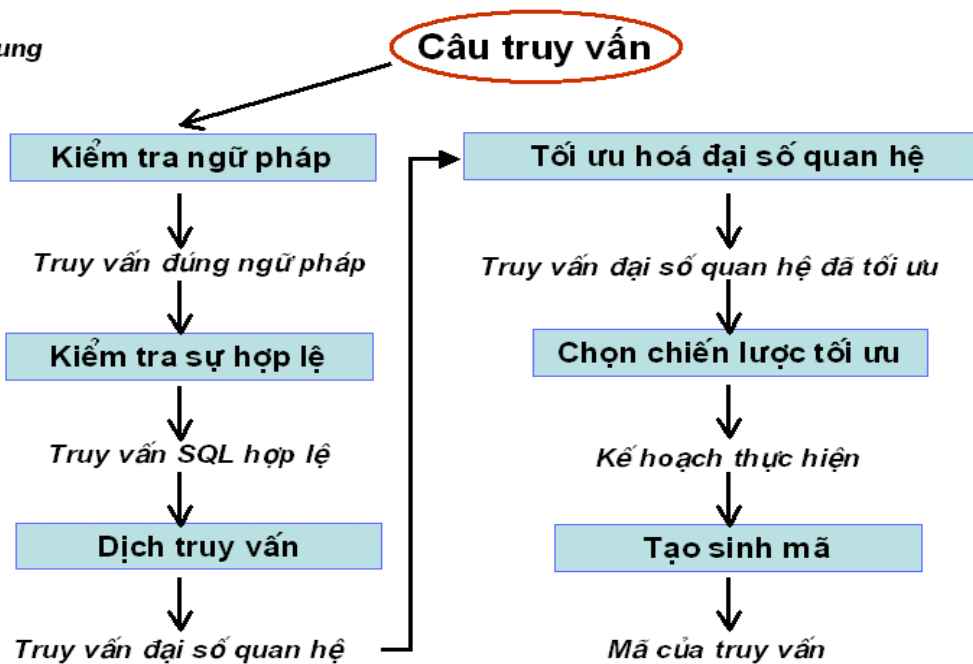
- Chọn một truy vấn đại số quan hệ tốt nhất trong số tất cả các truy vấn đại số tương đương;
- Các chiến lược xử lý truy vấn có thể biểu diễn trong sự mở rộng của đại số quan hệ.

Phân tán:

- Kế thừa chiến lược xử lý truy vấn như môi trường tập trung;
- Còn phải quan tâm thêm;
- Các phép toán truyền dữ liệu giữa các trạm;
- Chọn các trạm tốt nhất để xử lý dữ liệu;
- Cách thức và biến đổi dữ liệu.

Tối ưu hoá truy vấn trong môi trường tập trung

Sơ đồ chung



Hình 3.1. Mô hình tối ưu hóa truy vấn

3.2.2. Chiến lược tối ưu trong CSDL tập trung

Hai trong số những kỹ thuật tối ưu thông dụng nhất trong các hệ thống tập trung là các thuật toán **INGRES** và **SYSTEM R**.

Thuật toán INGRES

Ý tưởng thuật toán: Thuật toán tổ hợp hai giai đoạn phân rã và tối ưu hoá.

- Đầu tiên phân rã câu truy vấn dạng phép toán quan hệ thành các phần nhỏ hơn. Câu truy vấn được phân rã thành một chuỗi các truy vấn có một quan hệ chung duy nhất. Sau đó mỗi câu truy vấn đơn quan hệ được xử lý bởi một “*thể xử lý truy vấn một biến*” (one variable query processor-OVQP).

- OVQP tối ưu hoá việc truy xuất đến một quan hệ bằng cách dựa trên vị từ phương pháp truy xuất hữu hiệu nhất đến quan hệ đó. Trước tiên OVQP sẽ thực hiện các phép toán đơn ngôi và giảm thiểu kích thước của các kết quả trung gian bằng các *tách* (detachment) và *thay thế* (substitution).

Kí hiệu $q_{i-1} \rightarrow q_i$ để chỉ câu truy vấn q được phân rã thành hai câu truy vấn con q_{i-1} và q_i , trong đó q_{i-1} được thực hiện trước và kết quả sẽ được q_i sử dụng.

Phép tách: OVQP sử dụng để tách câu truy vấn q thành các truy vấn $q' \rightarrow \square q''$ dựa trên một quan hệ chung là kết quả của q' .

Nếu câu truy vấn q được biểu diễn bằng SQL có dạng:

q : **SELECT** **R₂.A₂, R₃.A₃, ..., R_n.A_n**
FROM **R₁, R₂, ..., R_n**
WHERE **P₁(R₁.A'₁) AND P₂(R₁.A₁, R₂.A₂, ..., R_n.A_n)**

Trong đó: A_1 và A'_1 là các thuộc tính của quan hệ R_1 .

P_1 là vị từ có chứa các thuộc tính của các quan hệ R_1, R_2, \dots, R_n . Một câu truy vấn như thế có thể phân rã thành hai câu truy vấn con, q' theo sau là q'' qua phép tách dựa trên quan hệ chung R_1 như sau:

q' : **SELECT** **R₁A₁ INTO R'₁**
FROM **R₁**
WHERE **P₁(R₁.A₁)**

Trong đó R'_1 là một quan hệ tạm thời chứa các thông tin cần thiết để thực hiện tiếp tục câu truy vấn:

q'' : **SELECT** **R₂A₂, ..., R_nA_n**
FROM **R'₁, R₂, ..., R_n**
WHERE **P₁(R₁.A₁, R₂.A₂, ..., R_n.A_n)**

Ví dụ minh họa: xét CSDL của một công ty máy tính.

Để minh họa kỹ thuật tách chúng ta sử dụng CSDL trên cho câu truy vấn sau: “**Cho biết tên của các nhân viên đang làm việc trong dự án có tên CSDL**”

Câu truy vấn này (q1) được diễn tả bằng SQL:

```
q1:  SELECT  NHANVIEN.TENNV
      FROM NHANVIEN, HOSO, DUAN
      WHERE   NHANVIEN.MANV = HOSO.MANV
      AND     HOSO.MADA = DUAN.MADA
      AND     TENDA = “CSDL”
```

Câu truy vấn q1 được tách thành $q11 \rightarrow q'$, trong đó TGIAN1 là quan hệ trung gian.

```
q11: SELECT  DUAN.MADA INTO TGIAN1
      FROM DUAN
      WHERE   TENDA = “CSDL”
q':  SELECT  NHANVIEN.TENNV
      FROM    NHANVIEN, HOSO, TGIAN1
      WHERE   NHANVIEN.MANV = HOSO.MANV
      AND     HOSO.MADA =TGIAN1.MADA
```

Các bước tách tiếp theo cho q' có thể tạo ra:

```
q12: SELECT  HOSO.MANV INTO TGIAN2
      FROM    HOSO, TGIAN1
      WHERE   HOSO.MADA =TGIAN1.MADA
q13: SELECT  NHANVIEN.TENNV
      FROM    NHANVIEN, TGIAN2
      WHERE   NHANVIEN.MANV = TGIAN2.MANV
```

Truy vấn q1 đã được rút gọn thành chuỗi truy vấn $q11 \rightarrow q12 \rightarrow q13$. Truy vấn q11 là loại đơn quan hệ và có thể cho thực hiện bởi OVQP. Tuy nhiên các truy vấn q12 và q13 không phải loại đơn quan hệ và cũng không thể rút gọn hơn nữa bằng phép tách.

Các câu truy vấn đa quan hệ không thể tách tiếp được nữa (chẳng hạn q12 và q13) được gọi là *bất khả giản* (irreducible).

Các truy vấn bất khả giản được biến đổi thành câu truy vấn đơn quan hệ nhờ *phép thế bộ* (tuple substitution).

Cho câu truy vấn n-quan hệ q , các bộ của một biến được thay bằng các giá trị của chúng, tạo ra được một tập các truy vấn $(n-1)$ biến.

Phép thế bộ được tiến hành như sau: Trước tiên chọn một quan hệ trong truy vấn q để thay thế. Gọi R_1 là quan hệ đó. Thế thì với mỗi bộ t_{1i} trong R_1 , các thuộc tính được tham chiếu trong q được thay bằng các giá trị thật sự trong t_{1i} , tạo ra một câu truy vấn q' có $(n-1)$ quan hệ. Vì vậy số câu truy vấn q' được sinh ra bởi phép thế bộ là $\text{card}(R_1)$ (bằng số bộ có trong quan hệ R_1).

Phép thế bộ có thể tóm tắt như sau:

$q(R_1, R_2, \dots, R_n)$ được thay bởi $\{q'(t_{1i}, R_2, R_3, \dots, R_n), t_{1i} \in R_1\}$

Vì thế đối với mỗi bộ thu được, câu truy vấn con được xử lý đệ quy bằng phép thế nếu nó chưa bất khả giản.

Ví dụ minh họa: Xét tiếp câu truy vấn q_{13}

```
q13: SELECT  NHANVIEN.TENNV
          FROM    NHANVIEN, TGIAN2
          WHERE   NHANVIEN.MANV = TGIAN2.MANV
```

Quan hệ được định nghĩa bởi biến $TGIAN2$ chạy trên thuộc tính duy nhất $MANV$. Giả sử rằng nó chỉ chứa hai bộ: $\langle A1 \rangle$ và $\langle A2 \rangle$. Phép thế cho $TGIAN2$ tạo ra hai câu truy vấn con đơn quan hệ:

```
q131: SELECT  NHANVIEN.TENNV
          FROM    NHANVIEN
          WHERE   NHANVIEN.MANV = "A1"
```

```
q132: SELECT  NHANVIEN.TENNV
          FROM    NHANVIEN
          WHERE   NHANVIEN.MANV = "A2"
```

Sau đó chúng có thể được OVQP quản lý và sử dụng.

Thuật toán INGRES- QOA

Input: MRQ: câu truy vấn đa quan hệ (có n quan hệ)

Output: Câu truy vấn tối ưu

Begin

Output $\leftarrow \phi$

If $n=1$ **then**

Output \leftarrow run(MRQ) {thực hiện câu truy vấn một quan hệ}

Else {Tách MRQ thành m truy vấn một quan hệ và một truy vấn đa quan hệ}

ORQ₁, ..., ORQ_m, MRQ' \leftarrow MRQ

For $i \leftarrow 1$ **to** m

Output' \leftarrow run(ORQ_i) {thực hiện ORQ_i}

Output \leftarrow Output \cup Output' {trộn tất cả các kết quả lại}

Endfor

R \leftarrow CHOOSE_VARIABLE(MRQ') {R được chọn cho phép thế bộ}

For mỗi bộ $t \in R$

MRQ'' \leftarrow thay giá trị cho t trong MRQ'

Output' \leftarrow INGRES-QOA(MRQ'') {gọi đệ qui}

Output \leftarrow Output \cup Output' {trộn tất cả các kết quả lại}

Endfor

Endif

End. {INGRES-QOA}

Nhân xét:

- Thuật toán tối ưu hoá INGRES (được gọi là INGRES - QOA) sẽ xử lý đệ qui cho đến khi không còn câu truy vấn đa quan hệ nào nữa.

- Thuật toán có thể được áp dụng cho các phép chọn và các phép chiếu ngay khi có thể sử dụng kỹ thuật tách.

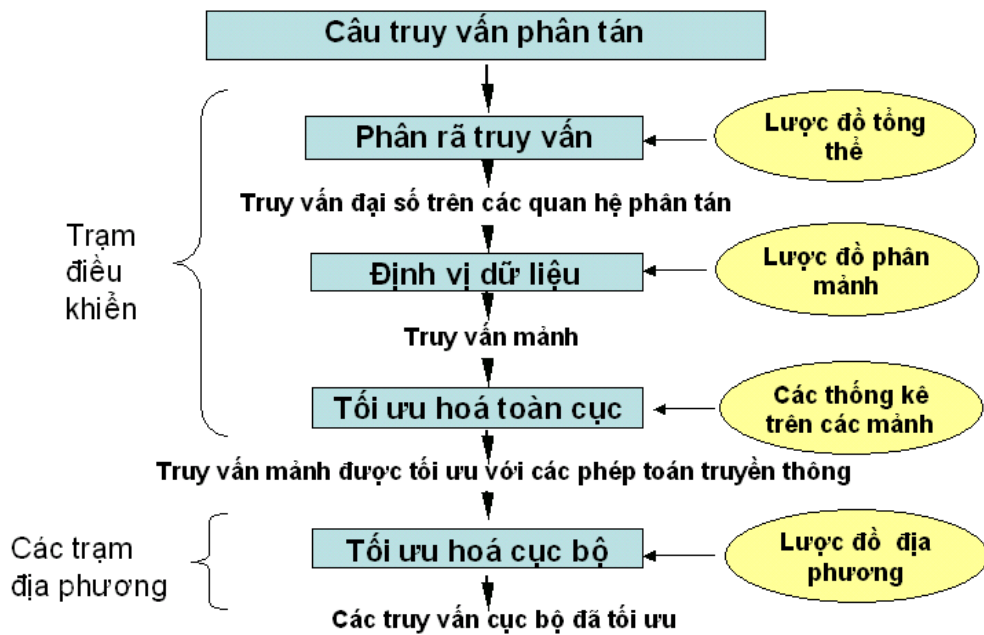
- Kết quả của câu truy vấn đơn quan hệ được lưu trong những cấu trúc dữ liệu có khả năng tối ưu hoá những câu truy vấn sau đó (như các nối) và sẽ được OVQP sử dụng.

- Các câu truy vấn bất khả giản còn lại sau phép tách sẽ được xử lý bằng phép thế bộ.

- Câu truy vấn bất khả giản, được kí hiệu là MRQ'. Quan hệ nhỏ nhất với lực lượng của nó đã được biết từ kết quả của câu truy vấn trước đó sẽ được chọn để thay thế.

3.3. Xử lý truy vấn trong môi trường phân tán

Tối ưu hoá truy vấn trong môi trường phân tán.



Hình 3.2. Sơ đồ phân lớp chung cho xử lý truy vấn phân tán

3.3.1. Phân rã truy vấn

- Biến đổi một phép tính quan hệ thành một truy vấn đại số trên quan hệ tổng thể.

- Cả hai truy vấn vào/ra đều được thực hiện trên quan hệ tổng thể và không quan tâm đến tính phân tán của dữ liệu. Vì vậy, phân rã truy vấn được thực hiện chung cho cả hệ tập trung và phân tán.

- Trong phần này chúng ta giả sử rằng các truy vấn vào luôn cú pháp đúng. Khi giai đoạn xử lý truy vấn thực hiện xong, thì truy vấn ra là đúng và tránh được các công việc dư thừa.

- Giai đoạn này chia làm bốn bước: chuẩn hoá, phân tích, loại bỏ dư thừa và viết lại.

Chúng ta trình bày ba bước đầu tiên trong phạm vi của phép tính quan hệ bộ. Chỉ có bước cuối cùng ghi truy vấn lại thành đại số quan hệ.

4.3.1.1. Chuẩn hoá

Mục đích: Chuyển đổi truy vấn thành một dạng chuẩn để thuận lợi cho các xử lý tiếp theo.

Với SQL, có hai dạng chuẩn cho các tân từ trong mệnh đề WHERE là:

Dạng chuẩn hội là hội (\wedge) của những phép toán tuyển (\vee):

$$(p_{11} \vee p_{12} \vee \dots \vee p_{1n}) \wedge \dots \wedge (p_{m1} \vee p_{m2} \vee \dots \vee p_{mn})$$

Dạng chuẩn tuyển là tuyển (\vee) của những phép toán hội (\wedge):

$(p_{11} \wedge p_{12} \wedge \dots \wedge p_{1n}) \vee \dots \vee (p_{m1} \wedge p_{m2} \wedge \dots \wedge p_{mn})$, trong đó p_{ij} là các biểu thức nguyên tố.

Các quy tắc biến đổi tương đương trên các phép toán logic:

1. $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$
2. $p_1 \vee p_2 \Leftrightarrow p_2 \vee p_1$
3. $\neg(\neg p) \Leftrightarrow p$
4. $(p_1 \wedge p_2) \Leftrightarrow \neg p_1 \vee \neg p_2$
5. $p_1 \vee (p_2 \wedge p_3) \Leftrightarrow (p_1 \vee p_2) \wedge (p_1 \vee p_3)$.
6. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$
7. $p_1 \vee (p_2 \vee p_3) \Leftrightarrow (p_1 \vee p_2) \vee p_3$
8. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$
9. $(p_1 \vee p_2) \Leftrightarrow \neg p_1 \wedge \neg p_2$

Ví dụ:

Từ các quan hệ NHANVIEN (MANV, TENNV, CHUCVU) và HOSO (MANV, MADA, NHIEMVU, THOIGIAN). Xét truy vấn:

“Tìm tên các nhân viên làm dự án J1 có thời gian 12 hoặc 24 tháng”.

Truy vấn trên được biểu diễn trong SQL:

```
SELECT   NHANVIEN.TENNV
FROM     NHANVIEN, HOSO
WHERE     NHANVIEN.MANV= HOSO.MANV
           AND HOSO.MADA=“J1”
           AND THOIGIAN=12 OR THOIGIAN=24
```

Điều kiện trong dạng chuẩn hội là:

$(\text{NHANVIEN.MANV}=\text{HOSO.MANV}) \wedge (\text{HOSO.MADA}=\text{“J1”}) \wedge (\text{THOIGIAN}=12 \vee \text{THOIGIAN}=24)$

Điều kiện trong dạng chuẩn tuyển là:

$(\text{NHANVIEN.MANV}=\text{HOSO.MANV} \wedge \text{HOSO.MADA}=\text{“J1”} \wedge \text{THOIGIAN}=12) \vee (\text{NHANVIEN.MANV}=\text{HOSO.MANV} \wedge \text{HOSO.MADA}=\text{“J1”} \wedge \text{THOIGIAN}=24)$

3.3.1.2. Phân tích

Mục đích: Phát hiện ra những thành phần không đúng (sai kiểu hoặc sai ngữ nghĩa) và loại bỏ chúng sớm nhất nếu có thể.

- *Truy vấn sai kiểu*: Nếu một thuộc tính bất kỳ hoặc tên quan hệ của nó không được định nghĩa trong lược đồ tổng thể, hoặc phép toán áp dụng cho các thuộc tính sai kiểu.

Ví dụ: Truy vấn dưới đây là sai kiểu

```
SELECT  E#  
FROM    E  
WHERE   E.TENNV > 200
```

vì hai lý do:

- Thuộc tính E# không khai báo trong lược đồ;
- Phép toán ">200" không thích hợp với kiểu chuỗi của thuộc tính E.TENNV.

- *Truy vấn sai ngữ nghĩa*: nếu các thành phần của nó không tham gia vào việc tạo ra kết quả.

Để xác định truy vấn có sai về ngữ nghĩa hay không, ta dựa trên việc biểu diễn truy vấn như một đồ thị gọi là *đồ thị truy vấn*. Đồ thị này được xác định bởi các truy vấn liên quan đến phép chọn, chiếu và nối. Nếu đồ thị truy vấn mà không liên thông thì truy vấn là sai ngữ nghĩa.

Đồ thị truy vấn:

- Một nút dùng để biểu diễn cho quan hệ kết quả;
- Các nút khác biểu diễn cho các toán hạng trong quan hệ;
- Cạnh nối giữa hai nút không phải là nút kết quả biểu diễn một phép nối;
- Cạnh có nút đích là kết quả thì biểu diễn một phép chiếu;
- Một nút không phải là kết quả có thể được gán nhãn bởi phép chọn hoặc phép tự nối (self-join: nối của quan hệ với chính nó).

Đồ thị kết nối:

Là một đồ thị con của đồ thị truy vấn (join graph), trong đó chỉ có phép nối.

Ví dụ: Từ các quan hệ

E=NHANVIEN (MANV, TENNV, CHUCVU)

G = HOSO (MANV, MADA, NHIEMVU, THOIGIAN) và

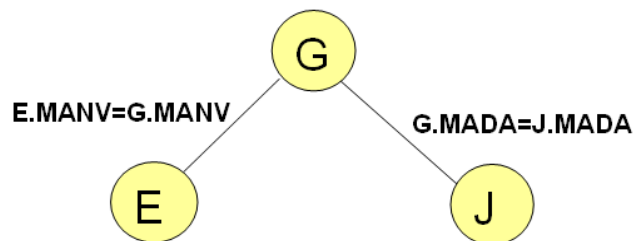
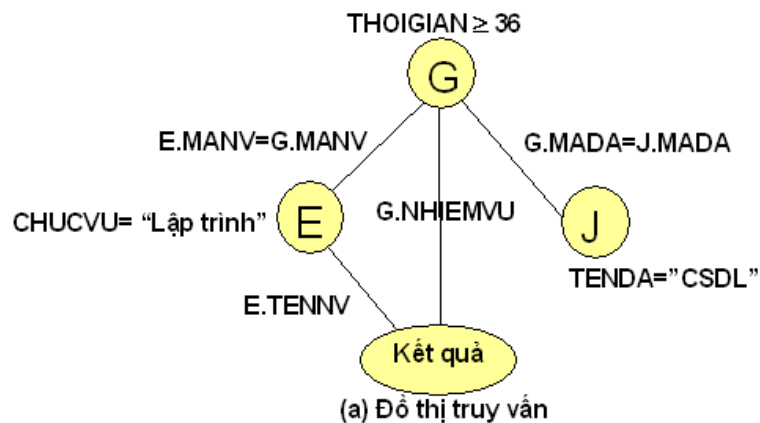
J=DUAN (MADA, TENDA, NGANSACH).

Hãy xác định "*Tên và nhiệm vụ các lập trình viên làm dự án CSDL có thời gian lớn hơn 3 năm.*"

Truy vấn SQL tương ứng là:

```
SELECT    E.TENNV, G.NHIEMVU
FROM      E, G, J
WHERE     E.MANV=G.MANV
          AND G.MADA.= J.MADA
          AND TENDA="CSDL"
          AND THOIGIAN≥36
          AND E.CHUCVU="LTRINH"
```

Đồ thị truy vấn và đồ thị kết nối tương ứng:



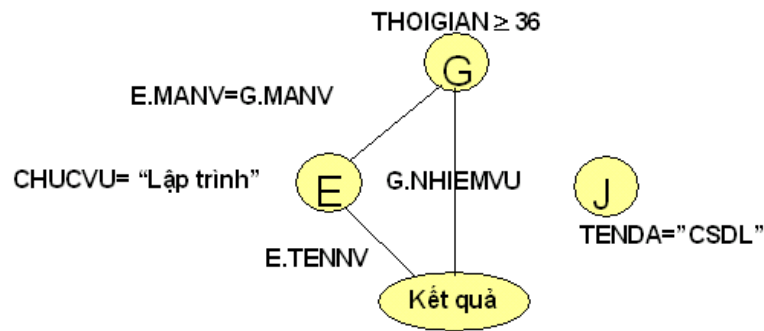
Hình 3.3. Đồ thị kết nối tương ứng

Nếu viết câu truy vấn SQL như sau:

```
SELECT    E.TENNV, G.NHIEMVU
FROM      E, G, J
WHERE     E.MANV=G.MANV
          AND J.TENDA="CSDL"
          AND G.THUOIGIAN = 36
          AND E.CHUCVU="Lập trình"
```

(thiếu điều kiện: G.MADA=J.MADA)

Truy vấn này là sai ngữ nghĩa vì đồ thị truy vấn của nó không liên thông.



Hình 3.4. Đồ thị truy vấn

3.3.1.3 . Loại bỏ dư thừa

- Điều kiện trong các truy vấn có thể có chứa các tân từ dư thừa.
- Một đánh giá sơ sài về một điều kiện dư thừa có thể dẫn đến lặp lại một số công việc.
- Sự dư thừa tân từ và dư thừa công việc có thể được loại bỏ bằng cách làm đơn giản hoá các điều kiện thông qua các luật lũy đẳng sau:

- | | |
|---|---|
| 1. $p \wedge p \Leftrightarrow p$ | 2. $p \vee \text{true} \Leftrightarrow \text{true}$ |
| 3. $p \vee p \Leftrightarrow p$ | 4. $p \wedge \neg p \Leftrightarrow \text{false}$ |
| 5. $p \wedge \text{true} \Leftrightarrow p$ | 6. $p \vee \neg p \Leftrightarrow \text{true}$ |
| 7. $p \vee \text{false} \Leftrightarrow p$ | 8. $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$ |
| 9. $p \wedge \text{false} \Leftrightarrow \text{false}$ | 10. $p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$ |

Ví dụ: xét câu truy vấn sau:

```

SELECT E.CHUCVU
FROM   E
WHERE  (NOT(E.CHUCVU="Lập trình")
        AND (E.CHUCVU="Lập trình" OR E.CHUCVU="Kỹ sư điện")
        AND NOT(E.CHUCVU="Kỹ sư điện")
        OR  E.TENNV="Dung")
    
```

Sử dụng các luật lũy đẳng nêu trên, truy vấn được biến đổi thành:

```

SELECT  E.CHUCVU
FROM    E
WHERE   E.TENNV="Dung"
    
```

Thực vậy, đặt

$p1: \langle E.CHUCVU = \text{"Lập trình"} \rangle$,

$p2: \langle E.CHUCVU = \text{"Kỹ sư điện"} \rangle$,

$p3: \langle E.TENNV = \text{"Dung"} \rangle$.

Khi đó, các tân từ sau mệnh đề WHERE được mô tả lại:

$p: (\neg p1 \wedge (p1 \vee p2) \wedge \neg p2) \vee p3$

$\Leftrightarrow (\neg p1 \wedge p1 \wedge \neg p2) \vee (\neg p1 \wedge p2 \wedge \neg p2) \vee p3$ (áp dụng luật 8)

$\Leftrightarrow (\text{false} \wedge \neg p2) \vee (\neg p1 \wedge \text{false}) \vee p3$ (áp dụng luật 5)

$\Leftrightarrow \text{false} \vee \text{false} \vee p3$ (áp dụng luật 7)

$\Leftrightarrow p3$

3.3.1.4. *Viết lại*

Bước này được chia làm hai bước con như sau:

- Biến đổi trực tiếp truy vấn phép tính sang đại số quan hệ;
- Cấu trúc lại truy vấn đại số quan hệ để cải thiện hiệu quả thực hiện.

Thông thường người ta biểu diễn các truy vấn đại số quan hệ bởi *cây đại số quan hệ*.

Cây đại số quan hệ là một cây mà nút lá biểu diễn một quan hệ trong CSDL, các nút không lá là các quan hệ trung gian được sinh ra bởi các phép toán đại số quan hệ.

Cách chuyển một truy vấn phép tính quan hệ thành một cây đại số quan hệ:

- Các nút lá khác nhau được tạo cho mỗi biến bộ khác nhau (tương ứng một quan hệ). Trong SQL các nút lá chính là các quan hệ trong mệnh đề FROM.

- Nút gốc được tạo ra xem bởi một phép chiếu lên các thuộc tính kết quả. Trong SQL nút gốc được xác định qua mệnh đề SELECT.

- Điều kiện (mệnh đề WHERE trong SQL) được biến đổi thành dãy các phép toán đại số thích hợp (phép chọn, nối, phép hợp, v.v...) đi từ lá đến gốc, có thể thực hiện theo thứ tự xuất hiện của các tân từ và các phép toán.

Ví dụ:

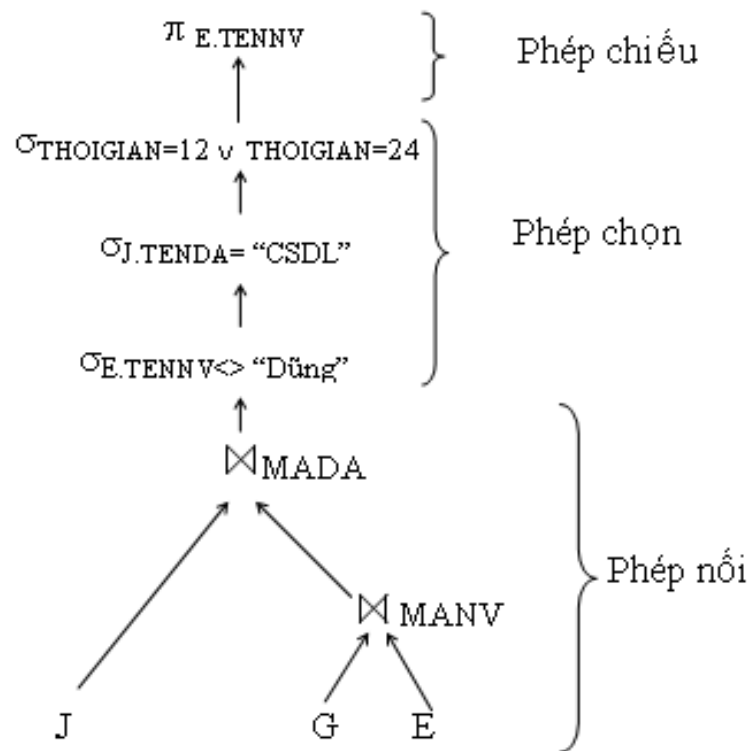
Truy vấn “*Tìm tên các nhân viên không phải là “Dũng”, làm việc cho dự án CSDL với thời gian một hoặc hai năm*”.

Biểu diễn truy vấn này trong SQL là:

```

SELECT  TENNV
FROM    J, G, E
WHERE   G.MANV=E.MANV
        AND G.MADA= J.MADA
        AND E.TENNV <> “Dũng”
        AND J.TENDA= “CSDL”
        AND (THOIGIAN=12 OR THOIGIAN=24)

```



Hình 3.5.Cây đại số quan hệ

06 luật biến đổi phép toán đại số quan hệ:

Mục đích: dùng để biến đổi cây đại số quan hệ thành các cây tương đương (trong đó có thể có cây tối ưu).

Giả sử R, S, T là các quan hệ, R được định nghĩa trên toàn bộ thuộc tính $A=\{A_1, \dots, A_n\}$, S được định nghĩa trên toàn bộ thuộc tính $B=\{B_1, \dots, B_n\}$.

1.Tính giao hoán của các phép toán hai ngôi:

Phép tích Decartes và phép nối hai quan hệ có tính giao hoán.

i. $R \times S \Leftrightarrow S \times R$

ii. $R \bowtie S \Leftrightarrow S \bowtie R$

2. Tính kết hợp của các phép toán hai ngôi:

Phép tích Decartes và phép nối hai quan hệ có tính kết hợp.

$$i. (R \times S) \times T \Leftrightarrow R \times (S \times T) \quad ii. (R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T)$$

3. Tính lũy đẳng của những phép toán một ngôi• Dãy các phép chiếu khác nhau trên cùng quan hệ được tổ hợp thành một phép chiếu và ngược lại:

$$\Pi_{A'}(\Pi_{A''}(R)) \Leftrightarrow \Pi_{A'}(R) \quad A', A'' \in R \text{ và } A' \subseteq A''$$

Dãy các phép chọn khác nhau $\sigma_{p_i(A_i)}$ trên cùng một quan hệ, với p_i là một tần từ được gán vào thuộc tính A_i , có thể được tổ hợp thành một phép chọn.

$$\sigma_{p_1(A_1)}(\sigma_{p_2(A_2)}(R)) = \sigma_{p_1(A_1) \wedge p_2(A_2)}(R)$$

4. Phép chọn giao hoán với phép chiếu

$$\Pi_{A_1, \dots, A_n}(\sigma_p(A_p)(R)) \Leftrightarrow \Pi_{A_1, \dots, A_n}(\sigma_p(A_p)(\Pi_{A_1, \dots, A_n, A_p}(R)))$$

Nếu A_p là thành viên của $\{A_1, \dots, A_n\}$, biểu thức trên thành

$$\Pi_{A_1, \dots, A_n}(\sigma_p(A_p)(R)) \Leftrightarrow \sigma_p(A_p)(\Pi_{A_1, \dots, A_n}(R))$$

5. Phép chọn giao hoán với những phép toán hai ngôi

Phép chọn với phép nhân:

$$\sigma_{p(A_i)}(R \times S) \Leftrightarrow \sigma_{p(A_i)}(R) \times S$$

Phép chọn với phép nối:

$$\sigma_{p(A_i)}(R \bowtie_{p(A_i, B_k)} S) \Leftrightarrow \sigma_{p(A_i)}(R) \bowtie_{(A_j, B_k)} S$$

Phép chọn với phép hợp: Nếu R và T cùng bộ thuộc tính.

$$\sigma_{p(A_i)}(R \cup T) \Leftrightarrow \sigma_{p(A_i)}(R) \cup \sigma_{p(A_i)}(T)$$

6. Phép chiếu giao hoán với những phép toán hai ngôi

Phép chiếu và tích Decartes: Nếu $C = A' \cup B'$ với $A' \subseteq A, B' \subseteq B$

và A, B là tập các thuộc tính trên quan hệ R, S ta có:

$$\Pi_C(R \times S) = \Pi_{A'}(R) \times \Pi_{B'}(S)$$

Phép chiếu và phép nối:

$$\Pi_C(R \bowtie_{p(A_i, B_j)} S) = \Pi_{A'}(R) \bowtie_{p(A_i, B_j)} \Pi_{B'}(S)$$

Phép chiếu và phép hợp:

$$\Pi_C(R \cup S) = \Pi_A'(R) \cup \Pi_B'(S)$$

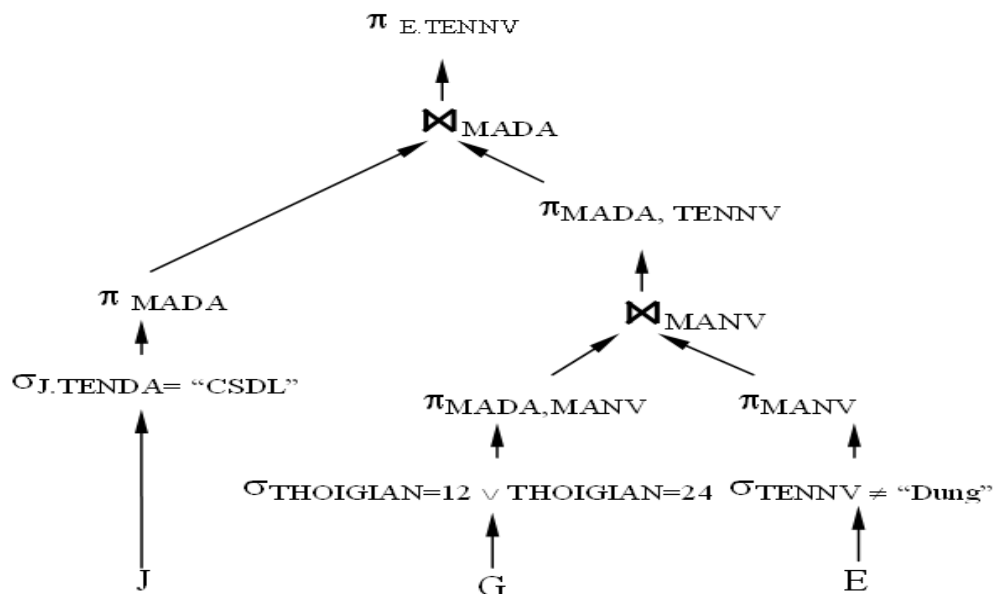
Chú ý: Việc sử dụng sáu luật trên có khả năng sinh ra nhiều cây đại số quan hệ tương đương nhau. Vấn đề là xác định cho được cây tối ưu

Chú ý: Trong giai đoạn tối ưu, sự so sánh các cây có thể thực hiện dựa trên chi phí dự đoán của chúng. Tuy nhiên, nếu số lượng các cây quá lớn thì cách tiếp cận này sẽ không hiệu quả. Có thể dùng các luật trên để cấu trúc lại cây, nhằm loại bỏ những cây đại số quan hệ “tồi”.

Các luật trên có thể sử dụng theo bốn cách như sau:

- Phân rã các phép toán một ngôi, đơn giản hóa biểu thức truy vấn;
- Nhóm các phép toán một ngôi trên cùng một quan hệ để giảm số lần thực hiện;
- Giao hoán các phép toán một ngôi với các phép toán hai ngôi để ưu tiên cho một số phép toán (chẳng hạn phép chọn);
- Sắp thứ tự các phép toán hai ngôi trong thực hiện truy vấn.

Ví dụ: Cấu trúc lại cây truy vấn ở ví dụ trên, cho ra cây kết quả tốt hơn cây ban đầu, tuy nhiên vẫn còn xa cây tối ưu.



Hình 3.6. Cây truy vấn

3.3.2. Định vị dữ liệu phân tán-Tối ưu hóa cục bộ

- Lớp định vị biến đổi một truy vấn đại số quan hệ tổng thể thành một truy vấn đại số được biểu thị trên các mảnh vật lý.

- Sử dụng thông tin được lưu trữ trên các lược đồ phân mảnh để định vị.

- Chương trình đại số quan hệ xây dựng lại quan hệ tổng thể từ các phân mảnh của nó gọi là *chương trình định vị*.

- Truy vấn có được từ chương trình định vị gọi là truy vấn ban đầu.

- Trong phần dưới đây, với *mỗi kiểu phân mảnh* chúng ta sẽ biểu diễn một *kỹ thuật rút gọn* để sinh ra truy vấn được tối ưu và đơn giản hoá.

3.3.2.1. Rút gọn theo phân mảnh ngang nguyên thủy

Xét quan hệ E(MANV,TENNV,CHUCVU). Tách quan hệ này thành ba mảnh ngang E₁, E₂ và E₃ như sau:

$$E_1 = \sigma_{MANV \leq "E3"}(E); E_2 = \sigma_{"E3" < MANV \leq "E6"}(E); E_3 = \sigma_{MANV > "E6"}(E)$$

Chương trình định vị cho một quan hệ E được phân mảnh ngang là hợp của các mảnh E₁, E₂, E₃. Nghĩa là, $E = E_1 \cup E_2 \cup E_3$. Vì vậy, dạng ban đầu của bất kỳ truy vấn nào được xác định trên E là có được bằng cách thay thế nó bởi $E = E_1 \cup E_2 \cup E_3$. Việc rút gọn các truy vấn trên các quan hệ đã được phân mảnh ngang bao gồm việc *xác định câu truy vấn*, sau khi đã cấu trúc lại cây con. Điều này sẽ sinh ra một số quan hệ rỗng, và sẽ *loại bỏ* chúng. Phân mảnh ngang có thể được khai thác để làm đơn giản cả phép chọn và phép nối.

a. **Rút gọn với phép chọn**: cho một quan hệ R được phân mảnh ngang thành R₁, R₂, ..., R_n với $R_j = \sigma_{p_j}(R)$

Luật 1: $\sigma_{p_j}(R_j) = \phi$ nếu $\forall x \in R : \neg(p_i(x) \wedge p_j(x))$.

Trong đó, p_i, p_j là tân từ chọn, x là bộ dữ liệu, p(x) là tân từ p chiếm giữ x.

Ví dụ: Xét truy vấn

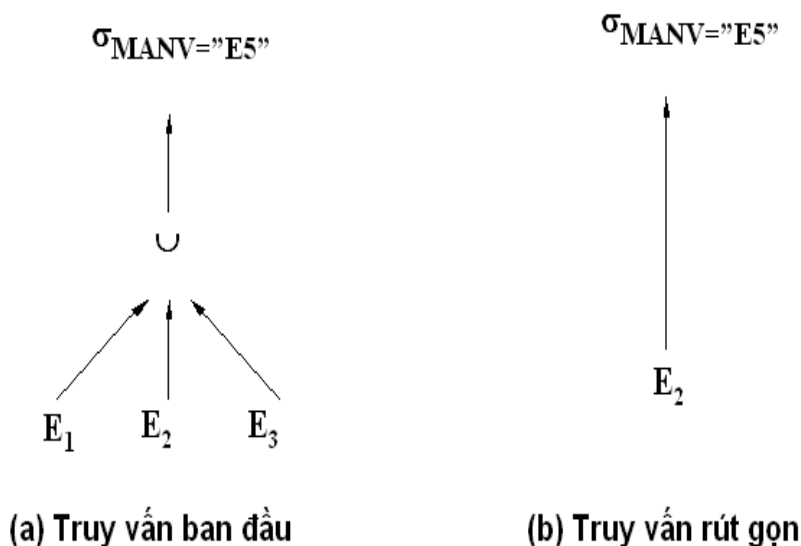
```
SELECT    *
FROM      E
WHERE     MANV="E5"
```

Áp dụng cách tiếp cận tự nhiên đến vùng E từ E₁, E₂ và E₃ cho truy vấn ban đầu, hình 4.7a. Bằng cách sử dụng tính chất giao hoán phép chọn với phép hợp, chúng ta thấy tân từ chọn đối lập với tân từ E₁ và E₃, sinh ra các quan hệ rỗng. Chúng ta thu được truy vấn rút gọn ở hình 4.7 b.

$$E_1 = \sigma_{MANV \leq "E3"}(E)$$

$$E_2 = \sigma_{"E3" < MANV \leq "E6"}(E)$$

$$E_3 = \sigma_{MANV > "E6"}(E)$$



Hình 3.7. Rút gọn cho phân mảnh ngang với phép hợp

b. Rút gọn với phép nối

- Các phép nối trên quan hệ đã được phân mảnh ngang có thể đơn giản khi chúng được phân mảnh theo thuộc tính nối.
- Việc rút gọn được thực hiện dựa trên tính phân phối giữa phép nối và phép hợp và loại bỏ các phép nối vô ích.

$$(R_1 \cup R_2) \bowtie R_3 = (R_1 \bowtie R_3) \cup (R_2 \bowtie R_3), R_i \text{ là các phân mảnh.}$$

Với tính chất này, có thể xác định được các phép nối vô ích của các mảnh khi các điều kiện nối mâu thuẫn nhau.

Khi đó, dùng luật 2 dưới đây để loại bỏ các phép nối vô ích.

Luật 2: $R_i \bowtie R_j = \emptyset$ nếu $\forall x \in R_i, \forall y \in R_j : \neg (p_i(x) \wedge p_j(y))$.

Trong đó R_i, R_j được xác định theo các tân từ p_i, p_j trên cùng thuộc tính.

Nhận xét:

- Việc xác định các phép nối vô ích được thực hiện bằng cách chỉ xem xét các tân từ mảnh.
- Truy vấn rút gọn không phải luôn tốt hơn hoặc đơn giản hơn truy vấn ban đầu.
- Một thuận lợi của truy vấn rút gọn là những phép nối có thể thực hiện song song.

Ví dụ: Giả sử quan hệ E được phân mảnh thành các mảnh

$$E_1 = \sigma_{MANV \leq 'E3'}(E); \quad E_2 = \sigma_{'E3' < MANV \leq 'E6'}(E); \quad E_3 = \sigma_{MANV > 'E6'}(E)$$

Quan hệ G được phân làm hai mảnh:

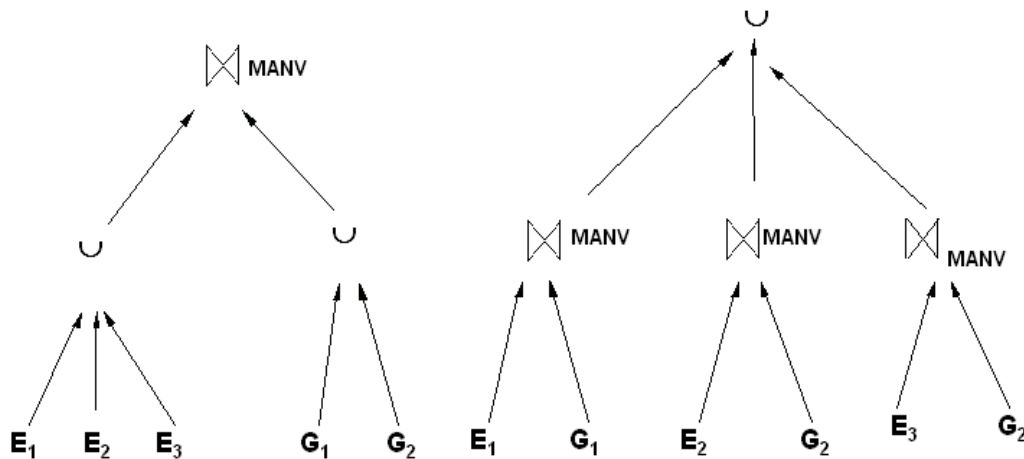
$$G_1 = \sigma_{MANV \leq 'E3'}(G) \text{ và } G_2 = \sigma_{MANV > 'E3'}(G).$$

Nhận xét:

- E_1 và G_1 được định nghĩa bởi cùng tân từ.
- Tân từ định nghĩa G_2 là hợp của các định nghĩa của những tân từ E_2 và E_3 .

Xét truy vấn

```
SELECT      *
FROM        E, G
WHERE       E.MANV=G.MANV
```



(a) Truy vấn ban đầu

(b) Truy vấn rút gọn

Hình 3.8. Sự rút gọn phân mảnh ngang với phép nối

3.3.2.2. Rút gọn theo phân mảnh dọc

• Chức năng của việc phân mảnh dọc là tách quan hệ dựa vào thuộc tính của các phép chiếu.

• Vì phép toán xây dựng lại đối với phân mảnh dọc là nối, nên chương trình định vị một quan hệ đã được phân mảnh dọc là nối của các mảnh trong vùng thuộc tính chung.

Ví dụ: Quan hệ E được phân mảnh dọc thành E_1 , E_2 , với thuộc tính khoá MANV được lặp lại như sau:

$$E_1 = \Pi_{MANV, TENNV}(E) \text{ và } E_2 = \Pi_{MANV, CHUCVU}(E)$$

Chương trình định vị là: $E = E_1 \text{ MANV } E_2$

- Các truy vấn trên phân mảnh dọc có thể rút gọn bằng cách xác định những quan hệ trung gian vô ích và loại bỏ các cây con chứa chúng.

- Các phép chiếu trên một phân mảnh dọc không có thuộc tính chung với các thuộc tính chiếu (ngoại trừ khóa của quan hệ) là vô ích, mặc dù các quan hệ là khác rỗng.

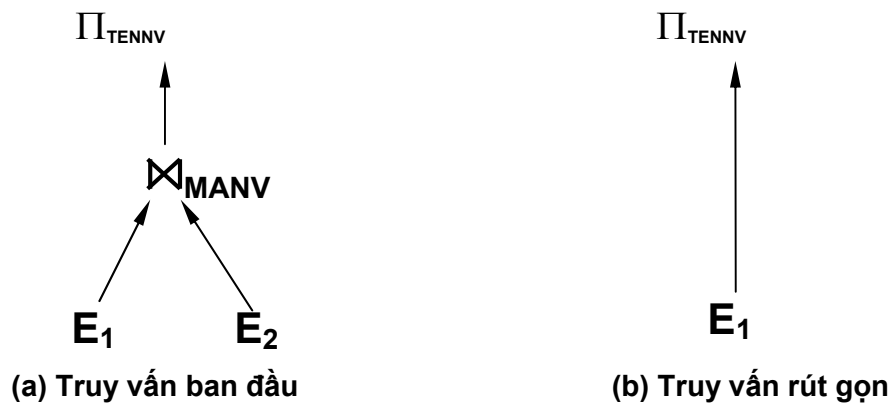
Luật 3: $\Pi_{D,K}(R_i)$ là vô ích nếu $D \cap A' = \emptyset$. Trong đó, quan hệ R xác định trên $A = \{A_1, \dots, A_n\}$; $R = \Pi_{A'}(R)$, $A' \subseteq A$, K là khoá của quan hệ, $K \subset A$, D là tập các thuộc tính chiếu, $D \subset A$.

Ví dụ: Với quan hệ E được phân mảnh dọc như sau:

$E_1 = \Pi_{MANV, TENNV}(E)$ và $E_2 = \Pi_{MANV, CHUCVU}(E)$

Xét truy vấn SQL:

```
SELECT  TENNV
FROM    E
```



Hình 3.9. Rút gọn đối với việc phân mảnh dọc

Nhận xét: Phép chiếu trên E_2 là vô ích vì $TENNV$ không có trong E_2 , nên phép chiếu chỉ cần gán vào E_1 .

3.3.2.3. Rút gọn theo phân mảnh gián tiếp

- Sự phân mảnh ngang gián tiếp là một cách tách hai quan hệ để việc xử lý nối của các phép chọn và phép nối.

- Nếu quan hệ R phụ thuộc vào sự phân mảnh ngang gián tiếp nhờ quan hệ S, thì các mảnh của R và S, mà có cùng giá trị thuộc tính nối, được định vị tại cùng trạm. Ngoài ra, S có thể được phân mảnh tùy thuộc vào tần từ chọn.

- Khi các bộ của R được đặt tùy theo những bộ của S, thì sự phân mảnh gián tiếp chỉ nên sử dụng mỗi quan hệ một nhiều từ $S \rightarrow R$ (với một bộ của S có thể phù hợp với n bộ của R, Nhưng với một bộ của R chỉ phù hợp với một bộ của S).

- Truy vấn trên các phân mảnh gián tiếp cũng có thể rút gọn được, nếu các tân từ phân mảnh mâu thuẫn nhau thì phép nối sẽ đưa ra quan hệ rỗng.
- Chương trình định vị một quan hệ đã được phân mảnh ngang gián tiếp là hợp của các mảnh.

Ví dụ: Cho mỗi quan hệ một nhiều từ E đến G, quan hệ G (MANV, MADA, NHIEMVU, THOIGIAN) có thể được phân mảnh gián tiếp theo những luật sau:

$$G_1 = G \bowtie_{\text{MANV}} E_1 \quad \text{và} \quad G_2 = G \bowtie_{\text{MANV}} E_2.$$

Trong đó E được phân mảnh ngang như sau:

$$E_1 = \sigma_{\text{CHUCVU}=\text{"Lập trình"}}(E) \quad \text{và} \quad E_2 = \sigma_{\text{CHUCVU} \neq \text{"Lập trình"}}(E)$$

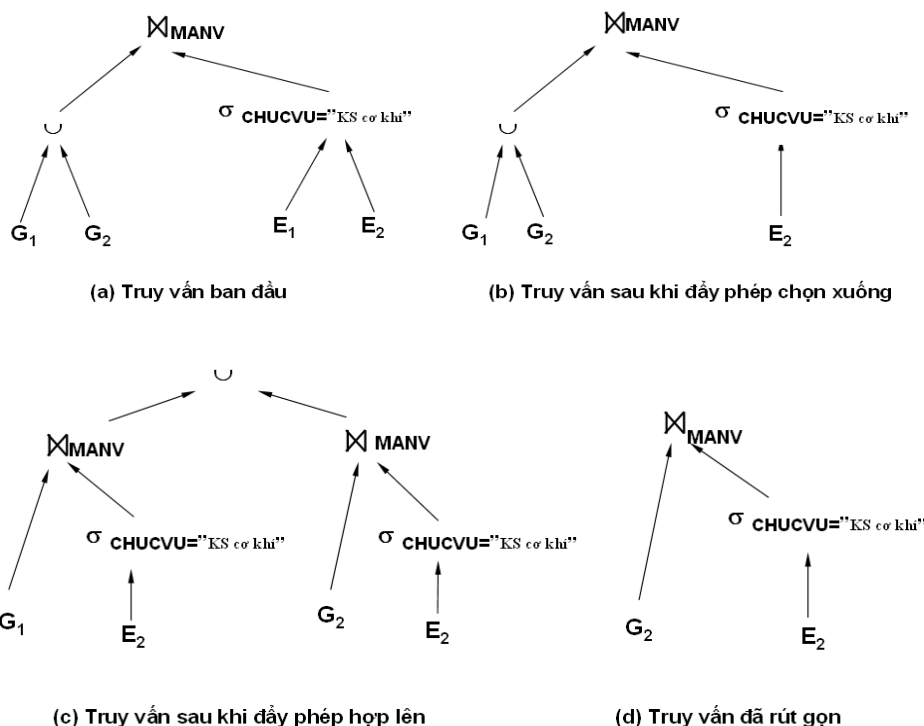
Chương trình định vị cho một quan hệ đã được phân mảnh gián tiếp là hợp của các mảnh $G = G_1 \cup G_2$.

Để rút gọn các truy vấn trên phân mảnh gián tiếp này, phép nối sẽ đưa ra quan hệ rỗng nếu các tân từ phân mảnh mâu thuẫn nhau.

Ví dụ tân từ G_1 và E_2 mâu thuẫn nhau, nên $G_1 \bowtie E_2 = \emptyset$.

Ví dụ: Xét truy vấn

```
SELECT      *
FROM        E, G
WHERE       G.MANV=E.MANV
AND CHUCVU= "KS cơ khí"
```



Hình 3.10. Các bước rút gọn truy vấn

Rút gọn của phân mảnh gián tiếp

Nhận xét:

- Truy vấn ban đầu trên các mảnh E_1 , E_2 , G_1 và G_2 tương ứng hình 3.10a.
- Bằng cách đẩy phép chọn xuống các mảnh E_1 và E_2 , được truy vấn rút gọn ở hình 3.10b.
- Phân phối các phép nối với phép hợp, chúng ta thu được cây hình 3.10c.
- Cây con bên trái đưa ra một quan hệ rỗng, nên cây rút gọn có được trong hình 3.10d.

3.3.2.4. Rút gọn theo phân mảnh hỗn hợp

- Sự phân mảnh hỗn hợp là sự kết hợp giữa phân dọc và phân mảnh ngang.
- Mục đích của phân mảnh hỗn hợp là hỗ trợ các truy vấn liên quan đến phép chiếu, phép chọn, phép nối.
- Chương trình định vị cho một quan hệ đã phân mảnh hỗn hợp sử dụng phép hợp và phép nối của các mảnh.

Ví dụ: Xét quan hệ E được phân mảnh hỗn hợp như sau:

$$E_1 = \sigma_{MANV \leq "E4"}(\Pi_{MANV, TENNV}(E))$$

$$E_2 = \sigma_{MANV > "E4"}(\Pi_{MANV, TENNV}(E))$$

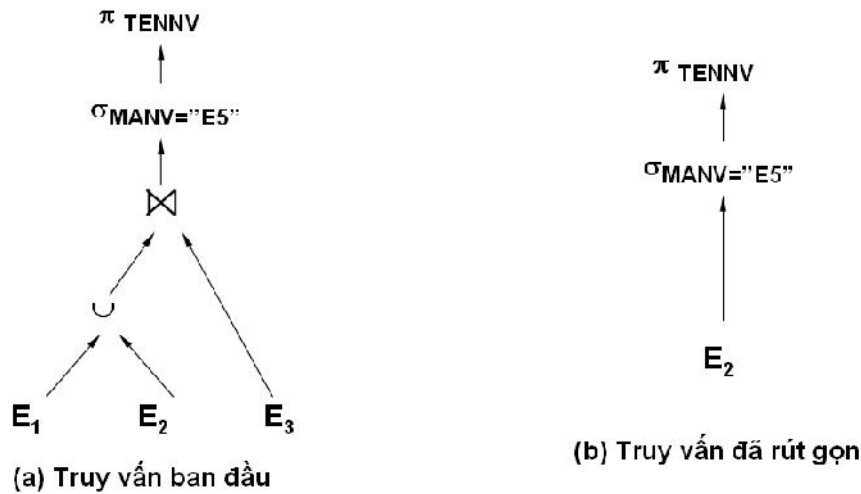
$$E_3 = \Pi_{MANV, CHUCVU}(E)$$

Chương trình định vị là: $E = (E_1 \cup E_2) \bowtie_{MANV} E_3$

Các truy vấn trên các mảnh hỗn hợp có thể được rút gọn bằng cách kết hợp các luật sử dụng trong phân mảnh ngang nguyên thủy, phân mảnh dọc, phân mảnh ngang gián tiếp, tương ứng như sau:

1. Loại bỏ các quan hệ rỗng sinh bởi sự mâu thuẫn giữa các phép chọn trên các phân mảnh ngang.
2. Loại bỏ các quan hệ vô ích sinh bởi các phép chiếu trên các phân mảnh dọc.
3. Phân phối các phép nối với các phép hợp để tách và loại bỏ các phép nối vô ích.

Ví dụ: Truy vấn trong SQL dưới đây minh họa việc ứng dụng các luật (1), (2) đến sự phân mảnh dọc_ngang của quan hệ E cho trên thành E_1 , E_2 và E_3 .



Hình 3.11. Rút gọn của phân mảnh hỗn hợp

3.4. Tối ưu hóa truy vấn trong CSDL phân tán

1. Truy vấn thu được từ giai đoạn phân rã và định vị dữ liệu có thể được thực hiện một cách đơn giản bằng việc thêm vào các thao tác truyền thông.

2. Việc hoán vị thứ tự các phép toán trong một câu truy vấn có thể cung cấp nhiều chiến lược tương đương khác nhau.

3. Bài toán xác định cây truy vấn tối ưu là NP-khó. Thông thường bộ tối ưu tìm tìm một chiến lược gần tối ưu và tránh các chiến lược “tồi”.

4. Đầu ra của bộ tối ưu là một lịch trình được tối ưu bao gồm truy vấn đại số được xác định trên các mảnh và các phép toán truyền thông hỗ trợ việc thực hiện truy vấn trên các trạm.

5. Để chọn lựa được một chiến lược tối ưu nói chung, bộ tối ưu phải xác định chi phí thực hiện câu truy vấn.

6. Chi phí thực hiện là tổ hợp có trọng số của chi phí truyền thông, chi phí I/O và chi phí CPU.

3.4.1. Mô hình chi phí của bộ tối ưu hóa truy vấn

Chi phí của một chiến lược thực hiện phân tán có thể được biểu diễn hoặc theo tổng chi phí hoặc theo thời gian trả lời.

- **Tổng chi phí** là tổng của tất cả các thành phần chi phí. bao gồm chi phí truyền thông, chi phí I/O và chi phí CPU.

Tuy nhiên, để đơn giản ta bỏ qua chi phí xử lý địa phương (I/O, CPU), coi chi phí truyền thông là trọng yếu.

• **Thời gian trả lời truy vấn** là thời gian được tính từ khi bắt đầu xử lý đến khi hoàn thành truy vấn.

Công thức chung cho sự xác định tổng chi phí:

Tổng chi phí: tổng của tất cả các chi phí C_{CPU} , $C_{I/O}$, C_{MSG}

$$\text{Total_cost} = C_{CPU} * \#instr + C_{I/O} * \#I/Os + C_{MSG} * \#msgs + C_{TR} * \#bytes$$

Trong đó: **Total_cost**: tổng chi phí

C_{CPU} : chi phí của một lệnh CPU

$C_{I/O}$: chi phí của một xuất/nhập đĩa

C_{MSG} : chi phí của việc khởi đầu và nhận một thông báo.

C_{TR} : chi phí truyền một đơn vị dữ liệu từ trạm này đến trạm khác, ta xem **C_{TR}** như là một hằng số.

$\#instr$: tổng tất cả các lệnh CPU ở các trạm

$\#I/Os$: số lần xuất/nhập đĩa

$\#msgs$: số thông báo

$\#bytes$: tổng kích thước của tất cả các thông báo.

Trong công thức:

$$\text{Total_cost} = C_{CPU} * \#instr + C_{I/O} * \#I/Os + C_{MSG} * \#msgs + C_{TR} * \#bytes$$

• Hai thành phần chi phí đầu (C_{CPU} , $C_{I/O}$) là *chi phí địa phương*.

• Hai thành phần chi phí sau (C_{MSG} , C_{TR}) là *chi phí truyền thông*.

• Chi phí truyền thông để chuyển $\#byte$ dữ liệu từ trạm này đến trạm khác được giả thiết là một hàm tuyến tính theo số $\#bytes$ được truyền đi, được xác định bởi công thức:

$$CC(\#byte) = C_{MSG} + C_{TR} * bytes$$

$$\text{Response_time} = C_{CPU} * seq_ \#instr + C_{I/O} * seq_ \#I/Os + C_{MSG} * seq_ \#msgs + C_{TR} * seq_ \#bytes$$

Trong đó:

$seq_ \#x$ (x có thể là số lệnh của CPU, I/O, số thông báo, số byte) là số lớn nhất của x khi thực hiện truy vấn một cách tuần tự.

Response_time: thời gian trả lời truy vấn

C_{CPU} : chi phí của một lệnh CPU

$C_{I/O}$: chi phí của một xuất/nhập đĩa

C_{MSG} : chi phí của việc khởi đầu và nhận một thông báo.

C_{TR} : chi phí truyền một đơn vị dữ liệu từ trạm này đến trạm khác

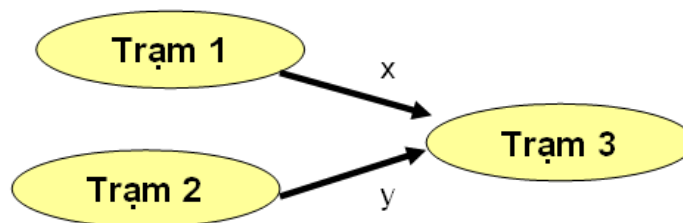
$\#instr$: tổng tất cả các lệnh CPU ở các trạm

$\#I/O$: số lần xuất/nhập đĩa

$\#msgs$: số thông báo

$\#bytes$: tổng kích thước của tất cả các thông báo.

Ví dụ: Minh họa sự khác nhau giữa **tổng chi phí** và **thời gian trả lời**, trong đó máy tính trả lời truy vấn tại trạm 3 với dữ liệu từ trạm 1 và 2, ở đây chỉ có chi phí truyền thông được xét.



Hình 3.12. Ví dụ của sự biến đổi 1 truy vấn

Giả sử, C_{MSG} và C_{TR} được biểu thị theo đơn vị thời gian. Tổng chi phí truyền x đơn vị từ trạm 1 đến trạm 3 và y đơn vị từ trạm 2 đến trạm 3 là:

$$\text{Total_cost} = C_{MSG} + C_{TR} * x + C_{MSG} + C_{TR} * y = 2C_{MSG} + C_{TR} * (x+y)$$

Vì việc truyền dữ liệu có thể được thực hiện song song nên thời gian trả lời của truy vấn là:

$$\text{Response_time} = \max\{C_{MSG} + C_{TR} * x, C_{MSG} + C_{TR} * y\}$$

3.4.2. Các thống kê dữ liệu

- Yếu tố chính ảnh hưởng đến hiệu suất của một chiến lược thực thi là kích thước của các quan hệ trung gian sinh ra trong quá trình thực hiện.

- Khi phép toán tiếp theo đặt tại một trạm khác, quan hệ trung gian phải được truyền trên mạng.

- Do đó để tối thiểu hoá khối lượng dữ liệu truyền đi, điều quan tâm đầu tiên là đánh giá kích thước kết quả trung gian của các phép toán đại số quan hệ.

- Đánh giá này dựa trên các thông tin thống kê về các quan hệ cơ sở và các công thức ước tính lực lượng của kết quả các phép toán quan hệ.

Mục đích của thống kê dữ liệu:

- Xác định kích thước của các quan hệ trung gian sinh ra trong quá trình thực hiện câu truy vấn;
- Xác định chi phí truyền thông cho các đại lượng trung gian.

Một số ký hiệu

Cho quan hệ R xác định trên tập thuộc tính $A=\{A_1, ..., A_n\}$. R được phân mảnh thành $R_1, R_2, ..., R_r$.

- **length(A_i)**: độ dài (byte) của thuộc tính $A_i, A_i \in R$;
- **card($\pi_{A_i}(R_j)$)**: lực lượng của phép chiếu của mảnh R_j lên thuộc tính A_i (số giá trị phân biệt trên thuộc tính A_i);
- **max(A_i)**: giá trị cực đại của thuộc tính A_i trong $\text{Dom}(A_i)$;
- **min(A_i)**: giá trị cực tiểu của thuộc tính A_i trong $\text{Dom}(A_i)$;
- **card(dom(A_i))**: lực lượng của thuộc tính A_i ;
- **card(R_i)**: số các bộ trong mảnh R_i .

Ngoài ra, dữ liệu thống kê cũng bao gồm *hệ số chọn của phép nối* (**SF_J**) đối với một số cặp đại số quan hệ, hệ số **SF_J** của quan hệ R và S là một số thực giữa 0 và 1, được xác định bởi:

$$\text{SF}_J = \frac{\text{card}(R \bowtie S)}{\text{card}(R) * \text{card}(S)}$$

- Hệ số **SF_J** nhỏ thì phép nối có tính chọn tốt, ngược lại có tính chọn tồi.
- Các thống kê này có lợi để đánh giá kích thước của quan hệ trung gian.
- Kích thước một quan hệ trung gian R được xác định bởi

$$\text{size}(R) = \text{card}(R) * \text{length}(R).$$

Trong đó,

+ **length(R)** là độ dài (số byte) của mỗi bộ trong R , được tính theo độ dài các thuộc tính của nó,

+ **card(R)** là số các bộ của R được tính theo công thức ở phần tiếp theo.

3.4.3. Lực lượng của các kết quả trung gian

Phần này sẽ đưa ra các công thức để ước tính lực lượng kết quả các phép toán cơ sở của đại số quan hệ (phép chọn, phép chiếu, phép tích Decartes, nối, nửa nối, phép hợp và phép trừ). Các toán hạng quan hệ được ký hiệu bởi R và S. Hệ số chọn của một phép toán SF_{OP} , (OP biểu thị phép toán) là tỷ lệ giữa các bộ của một toán hạng quan hệ tham gia vào kết quả của phép toán.

Ví dụ:

- SF_J : hệ số chọn của phép nối;
- SF_S : hệ số chọn của phép chọn.

1. Phép chọn

$$\text{card}(\sigma(R)) = SF_S(F) * \text{card}(R)$$

Trong đó $SF_S(F)$ phụ thuộc vào công thức chọn và có thể tính như sau, với $p(A_i)$, $p(A_j)$ là các tân từ tương ứng với các thuộc tính A_i , A_j .

$$SF_s(A=\text{value}) = \frac{1}{\text{Card}(\pi_A(R))}$$

$$SF_s(A > \text{value}) = \frac{\max(A) - \text{value}}{\max(A) - \min(A)}$$

$$SF_s(A < \text{value}) = \frac{\text{value} - \min(A)}{\max(A) - \min(A)}$$

$$SF_s(p(A_i) \wedge p(A_j)) = SF_s(p(A_i)) * SF_s(p(A_j))$$

$$SF_s(p(A_i) \vee p(A_j)) = SF_s(p(A_i)) + SF_s(p(A_j)) - SF_s(p(A_i)) * SF_s(p(A_j))$$

$$SF_s(A \in \{\text{value}\}) = SF_s(A=\text{value}) * \text{card}(\{\text{value}\})$$

2. Phép chiếu

Phép chiếu có thể có hoặc không loại bỏ các bản sao, ở đây chỉ xét phép chiếu loại bỏ các bản sao.

Lực lượng quan hệ kết quả của một phép chiếu tùy ý là khó đánh giá chính xác, vì tương quan giữa thuộc tính chiếu là thường không biết. Tuy nhiên, có hai trường hợp tầm thường nhưng đặc biệt có lợi:

- Nếu phép chiếu của R trên một thuộc tính đơn A thì lực lượng được tính đơn giản là số các bộ khi phép chiếu được thực hiện.

- Nếu một trong các thuộc tính chiều là khoá của R, thì:

$$\text{card}(\pi_A(R)) = \text{card}(R) \text{ và } \text{card}(R \times S) = \text{card}(R) * \text{card}(S)$$

3. Phép nối

- Không có một cách tổng quát để xác định lực lượng của một phép nối nếu không có các thông tin thêm.
- Cận trên của lực lượng của phép nối chính là lực lượng của tích Decartes.
- Tuy nhiên, có một số trường hợp xuất hiện thường xuyên và việc đánh giá là đơn giản:

- Nếu $R \bowtie_{AB} S$ với $A \in R, B \in S$, trong đó A là khoá của R, B là khoá ngoài của S, thì lực lượng của kết quả xấp xỉ là: $\text{card}(R \bowtie_{AB} S) = \text{card}(R)$

- Với các phép nối khác, lực lượng của kết quả là:

$$\text{card}(R \bowtie S) = SF_J * \text{card}(R) * \text{card}(S)$$

4. Phép nửa nối

Hệ số chọn của phép nửa nối (SF_{SJ}) xấp xỉ là:

$$SF_{SJ}(R \ltimes S) = \frac{\text{Card}(\pi_A(S))}{\text{Card}(\text{dom}[A])}$$

Công thức này chỉ phụ thuộc vào thuộc tính A của S, nên thường được gọi là hệ số chọn thuộc tính A của S, ký hiệu $SF_{SJ}(S.A)$ và là hệ số chọn của S.A trên bất cứ thuộc tính nối khác. Vì thế, lực lượng của phép nối được tính như sau:

$$\text{card}(R \ltimes AS) = SF_{SJ}(S.A) * \text{card}(R)$$

5. Phép hợp

• Rất khó đánh giá số lượng của $R \cup S$, vì các bộ giống nhau giữa R và S bị loại bỏ bởi phép hợp.

- Ở đây chúng ta chỉ đưa ra công thức tính:

- Cận trên của $\text{card}(R \cup S)$ bằng $\text{card}(R) + \text{card}(S)$,
- Cận dưới của $\text{card}(R \cup S)$ bằng $\max\{\text{card}(R), \text{card}(S)\}$

(giả sử R và S không chứa các bộ lặp).

6. Phép trừ

Cũng như phép hợp ở đây chỉ đưa ra cận trên và cận dưới, cận trên của $\text{card}(R-S)$ là $\text{card}(R)$, cận dưới là 0.

Ví dụ: Xét hai quan hệ trong cơ sở dữ liệu của công ty máy tính:

$E = \text{NHANVIEN}(\text{MANV}, \text{TENNV}, \text{CHUCVU})$ và

$G = \text{HOSO}(\text{MANV}, \text{MADA}, \text{NHIEMVU}, \text{THOIGIAN})$.

Với câu truy vấn “Cho biết tên các nhân viên hiện đang quản lý một dự án”. Ta có câu truy vấn SQL tương ứng là:

```
SELECT  TENNV
FROM    E, G
WHERE   E.MANV=G.MANV
        AND NHIEMVU="Quản lý"
```

Hai truy vấn đại số tương đương với truy vấn trên là:

$$\Pi_{\text{TENNV}}(\sigma_{\text{NHIEMVU}=\text{"Quản lý"} \wedge \text{E.MANV}=\text{G.MANV}}(\text{E} \times \text{G})) \quad (1)$$

$$\text{và} \quad \Pi_{\text{TENNV}}(\text{E} \bowtie_{\text{MANV}}(\sigma_{\text{NHIEMVU}=\text{"Quản lý"}} \text{G})) \quad (2)$$

Rõ ràng truy vấn (2) tránh được khối phải tích số của E và G, nên dùng ít phép tính tài nguyên hơn truy vấn (1).

Danh mục tài liệu tham khảo

1. Phạm Thế Quế (2004). *Giáo trình Cơ sở dữ liệu: Lý thuyết và thực hành*. Nhà xuất bản Bưu điện Hà Nội.
2. Phạm Thế Quế (2010). *Giáo trình Cơ sở dữ liệu phân tán*. Nhà xuất bản Thông tin và Truyền thông, Hà Nội.
3. Nguyễn Bá Tường (2005). *Nhập môn cơ sở dữ liệu phân tán*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
4. Lê Tiến Vương (1996). *Nhập môn cơ sở dữ liệu quan hệ*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
5. M.Tamer Ozsü And Patrice Valduriez (1999). *Principles of Distributed Database Systems*, Second Edition. Prentice Hall Upper Saddle River.

Câu hỏi và bài tập cuối chương

1. Mục đích của tối ưu hoá truy vấn trong CSDL phân tán.
2. Chức năng của tối ưu hoá truy vấn phân tán.
3. Các phương pháp xử lý truy vấn cơ bản.
4. Ý tưởng của thuật toán Ingres. Cho ví dụ.
5. Sơ đồ phân lớp chung cho xử lý truy vấn phân tán.
6. Định vị dữ liệu phân tán-Tối ưu hoá cục bộ.
7. Viết công thức tính tổng chi phí truyền thông và chi phí thời gian trả lời truy vấn. Giải thích các ký hiệu và cho ví dụ.

Bài tập 1: Xét CSDL của một công ty máy tính

NHANVIEN (E)

MANV	TENNV	CHUCVU
A1	Nam	Phân tích HT
A2	Trung	Lập trình viên
A3	Đông	Phân tích HT
A4	Bắc	Phân tích HT
A5	Tây	Lập trình viên
A6	Hùng	Kỹ sư điện
A7	Dũng	Phân tích HT
A8	Chiến	Thiết kế DL

HOSO (G)

MANV	MADA	NHIEMVU	THOIGIAN
A1	D1	Quản lý	12
A2	D1	Phân tích	34
A2	D2	Phân tích	6
A3	D3	Kỹ thuật	12
A3	D4	Lập trình	10
A4	D2	Quản lý	6
A5	D2	Quản lý	20
A6	D4	Kỹ thuật	36
A7	D3	Quản lý	48
A8	D3	Lập trình	15

DUAN (J)

MADA	TENDA	NGANSACH
D1	CSDL	20000
D2	CÀI ĐẶT	12000
D3	BẢO TRÌ	28000
D4	PHÁT TRIỂN	25000

TLUONG (S)

CHUCVU	LUONG
Kỹ sư điện	1000
Phân tích HT	2500
Lập trình viên	3000
Thiết kế DL	4000

1. Viết truy vấn SQL và đại số quan hệ để tìm “Tên nhân viên làm việc trong dự án Bảo trì và có lương lớn hơn 2000”. Với câu truy vấn trên làm các việc sau:

a. Sử dụng thuật toán INGRES để đưa truy vấn trên về dạng tối ưu (Mỗi truy vấn chỉ thực hiện trên 1 quan hệ).

b. Vẽ đồ thị truy vấn.

c. Vẽ cây đại số quan hệ.

d. Tối ưu hóa truy vấn trên bằng cây đại số quan hệ.

e. Viết lại truy vấn bằng SQL và đại số quan hệ.

2. Giả sử ta có sơ đồ phân mảnh sau:

Quan hệ E được phân mảnh thành các mảnh

$$E1 = \sigma_{MANV \leq "E3"}(E) \quad E2 = \sigma_{"E3" < MANV \leq "E6"}(E)$$

$$E3 = \sigma_{MANV > "E6"}(E)$$

Quan hệ G được phân làm hai mảnh:

$$G1 = \sigma_{MANV \leq "E3"}(G) \text{ và } G2 = \sigma_{MANV > "E3"}(G).$$

a. Viết truy vấn SQL để thực hiện phân mảnh các quan hệ trên.

b. Kiểm tra tính đầy đủ, tính tách biệt, tính tái thiết được.

c. Giả sử hệ thống cung cấp trong suốt định vị, không trong suốt phân đoạn, viết truy vấn để tìm tên nhân viên làm việc trong dự án Bảo trì.

3. Quan hệ E được phân mảnh hỗn hợp như sau:

a. Viết truy vấn SQL để thực hiện phân mảnh các quan hệ trên.

b. Kiểm tra tính đầy đủ, tính tách biệt, tính tái thiết được.

4. Quan hệ J được phân mảnh ngang thành 3 mảnh:

J1: Ngân sách > 40.000

J2: 20.000 ≤ Ngân sách ≤ 40.000

J3: Ngân sách < 20.000

- Quan hệ G được phân thành 2 mảnh

G1: Thời gian > 12 tháng

G2: Thời gian ≤ 12 tháng

- Quan hệ E không bị phân mảnh

- Hãy chuyển các truy vấn sau về truy vấn đoạn (bằng cách rút gọn truy vấn):

a. Tìm tên dự án có ngân sách 25.000.

b. Tìm tên dự án có thời gian thực hiện là 36 tháng.

c. Tìm tên nhân viên làm việc trong dự án có thời gian thực hiện là 2 năm.

5. Quan hệ S phân thành 3 mảnh:

S1: Lương > 3.000

S2: 2.000 ≤ Lương ≤ 3.000

S3: Lương < 2.000

- Quan hệ E được phân mảnh ngang dẫn xuất theo quan hệ S thành 3 mảnh:

E1 = E ⋈ S1

E2 = E ⋈ S2

E3 = E ⋈ S3

- Hãy chuyển các truy vấn sau về truy vấn đoạn (bằng cách rút gọn truy vấn):

a. Tìm tên nhân viên có lương 2.500

b. Tìm tên nhân viên có lương > 4000

6. Quan hệ G phân thành 2 mảnh:

G1: Gồm MANV, MaDA, Thời gian

G2: Gồm MaNV, MaDA, Nhiệm vụ

- Quan hệ E được phân mảnh ngang thành 2 mảnh:

E1 : Chức vụ="Phân tích hệ thống"

E2: Chức vụ \neq "Phân tích hệ thống"

- Hãy chuyển các truy vấn sau về truy vấn đoạn (bằng cách rút gọn truy vấn):

a. Tìm mã dự án có thời gian thực hiện > 1 năm.

b. Tìm tên nhân viên có chức vụ "Kỹ sư điện" và làm việc trong dự án có thời gian lớn hơn 1 năm.

7. Quan hệ J phân mảnh hỗn hợp thành 3 mảnh:

J1: Gồm MaDA, NganSach với $\text{NganSach} \leq 30.000$

J2: Gồm MaDA, NganSach với $\text{NganSach} > 30.000$

J3: Gồm MADA, TenDA

- Qua hệ G được phân thành 2 mảnh:

G1: Thời gian > 12 tháng

G2: Thời gian ≤ 12 tháng

- Hãy chuyển các truy vấn sau về truy vấn đoạn (bằng cách rút gọn truy vấn):

a. Tìm mã dự án có ngân sách 35.000.

b. Tìm tên dự án có ngân sách 25.000.

c. Tìm tên dự án có thời gian thực hiện 3 năm.

d. Tìm tên dự án có thời gian thực hiện 2 năm và ngân sách 20.000.

Bài tập 2: Cho một cơ sở dữ liệu phân tán gồm các quan hệ:

Khoa (MaKH, TenKH, KhuVuc, Tel, TrưởngKh);

CBGD (MaCB, TenCB, HocVi, Tel, MaKh);

SV (MaSV, TenSV, ĐịaChi, Lop, Khoa, MaKh);

HuongDan(MaCB, MaSV, DeTai).

Thiết kế phân đoạn xác định:

- Khoa được phân đoạn ngang theo KhuVuc với $\text{DOM}(\text{KhuVuc}) = \{'A', 'B', 'C'\}$.

- CBGD được phân đoạn dọc với 2 tập con thuộc tính (MaSV, TenSV, Khoa, MaKh) và (MaSV, TenSV, ĐịaChi, Lop) sau đó đoạn dọc thứ nhất được phân đoạn ngang suy diễn từ sự phân đoạn ngang đối với Khoa;

- Hướng dẫn được phân đoạn ngang suy diễn từ sự phân đoạn ngang đối

với CBGD.

Hãy đưa ra sơ đồ phân đoạn cho CSDL phân tán trên. Hãy đưa ra một cách sắp chỗ cho các đoạn của sơ đồ phân đoạn như đã mô tả trên 5 trạm biết rằng khu vực 'A' có trạm 1, 4; khu vực 'B' có trạm 2,5; khu vực 'C' có trạm 3.

Hãy xác định cây đoạn tối ưu (chuyên truy vấn về truy vấn đoạn – rút gọn truy vấn) cho yêu cầu sau: Đưa ra danh sách các CBGD với (TênCB) ở khoa 'CNTT' thuộc khu vực 'A' có tham gia hướng dẫn sinh viên lớp 'HTTT' khóa 49 về đề tài 'Cơ sở dữ liệu phân tán'.

- a) Hãy đưa ra sơ đồ phân đoạn cho CSDL phân tán trên.
- b) Hãy đưa ra một cách sắp chỗ cho các đoạn của sơ đồ phân đoạn trong câu
 - a) trên 5 trạm, biết rằng khu vực 'A' có trạm 1,4; khu vực 'B' có trạm 2,5; khu vực 'C' có trạm 3.
- c) Hãy xác định cây đoạn tối ưu cho yêu cầu sau:

Đưa ra danh sách các CBGD với (TênCB) ở khoa 'CNTT' thuộc khu vực 'A' có tham gia hướng dẫn sinh viên lớp 'HTTT' khóa 49 về đề tài 'Cơ sở dữ liệu phân tán'.

Chương 4

QUẢN LÝ GIAO DỊCH VÀ ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN

4.1. Tổng quan về giao dịch

Các khái niệm cơ bản về giao dịch

Trạng thái nhất quán (Consistent State) của một cơ sở dữ liệu: Một cơ sở dữ liệu ở trong một trạng thái nhất quán nếu nó tuân theo tất cả các ràng buộc toàn vẹn được định nghĩa trên nó.

Độ tin cậy (Reliability): Muốn nói đến khả năng tự thích ứng của một hệ thống đối với các loại sự cố và khả năng khôi phục từ những sự cố này. Một hệ thống tự thích ứng sẽ dung nạp được các sự cố hệ thống và có thể cung cấp dịch vụ ngay cả khi xảy ra sự cố.

Quản lý giao dịch (Transaction Management): Việc giải quyết các bài toán sao cho cố gắng duy trì cho được cơ sở dữ liệu ở trong tình trạng nhất quán ngay cả khi có nhiều truy cập đồng thời và khi giao dịch có sự cố.

Khái niệm giao dịch trong các hệ cơ sở dữ liệu phân tán hàm chứa một số đặc tính cơ bản như là tính nguyên tử và tính bền vững, nhằm chỉ ra những khác biệt giữa một giao dịch và một câu truy vấn.

Giao dịch là một đơn vị tính toán nhất quán và tin cậy trong các hệ cơ sở dữ liệu phân tán. Về mặt trực quan, giao dịch nhận một cơ sở dữ liệu và thực hiện trên nó một hành động, tạo ra một phiên bản cơ sở dữ liệu mới, tức là tạo ra một dịch chuyển trạng thái. Điều này tương tự như câu truy vấn thực hiện. Nếu cơ sở dữ liệu nhất quán trước khi thực thi giao dịch, có thể đảm bảo rằng cơ sở dữ liệu cũng sẽ nhất quán vào lúc kết thúc thực thi cho dù:

- Giao dịch có thể được thực hiện đồng thời với các giao dịch khác;
- Sự cố có thể xảy ra trong quá trình thực thi.

Như vậy, có thể hiểu một giao dịch là một dãy các thao tác, các câu hỏi truy vấn được biểu diễn trong các ngôn ngữ truy vấn hay là những đoạn chương trình được nhúng trong một ngôn ngữ chủ. Tính tương thích của giao dịch theo nghĩa được hiểu là tính đúng đắn của nó. Nói cách khác, giao dịch luôn luôn chuyển cơ sở dữ liệu từ một trạng thái tương thích đến một trạng thái tương thích khác. Tương thích của cơ sở dữ liệu phân tán khác với tương thích của một

giao dịch. Cơ sở dữ liệu có tính tương thích nếu nó thỏa mãn mọi ràng buộc tương thích được định nghĩa trên nó. Những thay đổi trạng thái của cơ sở dữ liệu xảy ra do các phép toán cập nhật (Update), chèn (Insert) và xóa (Delete). Khi thực hiện các giao dịch, cơ sở dữ liệu có thể tạm thời không tương thích, nhưng khi kết thúc giao dịch thì cơ sở dữ liệu phải ở trạng thái tương thích.

Một giao dịch phân tán gồm nhiều giao dịch con, thực hiện ở nhiều vị trí khác nhau. Môi trường phân tán có ảnh hưởng đến tất cả các khía cạnh quản lý giao dịch.

Một giao dịch là một dãy các thao tác đọc và ghi trên cơ sở dữ liệu cùng với các bước tính toán. Cụ thể hơn, có thể xem một giao dịch như là một chương trình truy cập vào cơ sở dữ liệu được gắn vào. Hoặc có thể định nghĩa một giao dịch là thực thi đơn của chương trình. Truy vấn đơn cũng có thể được coi là một chương trình và được đưa ra như là một giao dịch

Biểu diễn một giao dịch T: begin..., ..., ...end. Giữa begin và end là các thao tác cơ bản đọc, ghi và xử lý. Giao dịch đọc dữ liệu, ghi dữ liệu và thực hiện tính toán qua một vùng đệm làm việc (Private Workspace). Các tính toán của các giao dịch sẽ không có tác dụng trên cơ sở dữ liệu cho đến khi thực hiện ghi vào cơ sở dữ liệu.

Ví dụ 4.1: Xét câu truy vấn “tăng kinh phí các dự án CAD/CAM lên 10%”.

```
UPDATE      PROJ
SET          BUDGET = SUDGET *1.1
WHERE PNAME = “CAD/CAM”
```

Câu truy vấn này có thể đặc tả qua ký pháp SQL, như một giao dịch bằng cách gán cho nó tên BUDGET_UPDATE và khai báo như sau:

```
Begin_transaction BUDGET_UPDATE
Begin
EXEC SQL UPDATE PROJ
SET          BUDGET = BUDGET*1.1
WHERE PNAME = “CAD/CAM”
End.
```

Các lệnh Begin_transaction và end xác định ranh giới của một giao dịch. Biểu diễn giao dịch trên bằng dãy các thao tác.

Begin read PNAME, nếu thỏa mãn PNAME = “CAD/CAM”, read BUDGET, BUDGET = BUDGET*1.1, write BUDGET end.

Ví dụ 4.2: Xét bài toán đặt chỗ máy bay, giả sử có các quan hệ sau:

- FLIGHT (FNO, DATE, SCR, DEST, STSOLD, CAP) là quan hệ chuyến bay.
- CUST (CNAME, ADDR, BAL) là quan hệ về khách hàng đặt chỗ trước.
- FC (FNO, DATE, CNAME, SPECIAL) là quan hệ khách hàng- chuyến bay.

Trong đó FNO là mã số chuyến bay, DATE là ngày, tháng chuyến bay, SCR và DEST là nơi xuất phát và nơi đến của chuyến bay, STSOLD là số lượng ghế đã được bán trên chuyến bay đó. CAP là số lượng khách hàng mà chuyến bay có thể chở được, CNAME là tên khách hàng, ADDR là địa chỉ và số dư trong BAL, SPECIAL tương ứng với các yêu cầu đặc biệt mà khách hàng đưa ra khi đặt chỗ.

Giả sử có một nhân viên bán vé nhập mã số chuyến bay, ngày tháng, tên khách và thực hiện đặt chỗ trước. Giao dịch thực hiện như sau:

Begin_transaction Reservation

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)

Dòng (1): Nhập mã số chuyến bay, ngày tháng của chuyến bay và tên khách hàng, flight-no, date, customer_name.

Dòng (2): Tăng lên một ghế cho số ghế đã bán trên chuyến bay đang được yêu cầu.

Dòng (3): Chèn một bộ vào trong quan hệ FC. Giả sử rằng đây là khách hàng cũ nên không cần thiết phải chèn thêm dòng thông tin về khách hàng vào trong quan hệ CUST. Từ khóa Null chỉ ra rằng không có yêu cầu đặc biệt gì thêm.

Dòng (4): Ghi kết quả giao dịch ra màn hình của nhân viên bán vé.

Giao dịch trên ngầm được cho giả định sẽ được kết thúc, tức là luôn luôn còn chỗ trống trên chuyến bay đó, không kiểm tra còn vé hay không. Tuy nhiên đây là một giả thiết không thực tế và đặt ra một vấn đề là khả năng kết thúc giao dịch.

Chuỗi các thao tác các phép toán của giao dịch được diễn biến như sau: begin nhập flight_no, date và customer_name vào vùng đệm, read và so sánh flight_no, read và so sánh date, read STOLD, STOLD = STOLD +1, write STOLD, write ra một bộ (flight_no, date, customer_name, null), hiển thị ra màn hình end.

4.1.1. Điều kiện kết thúc giao dịch

Một giao dịch luôn phải được kết thúc ngay cả khi hệ thống có sự cố xảy ra. Có hai tình huống kết thúc giao dịch xảy ra:

a) *Nếu giao dịch thành công, nói rằng giao dịch là ủy thác (Commit).* Vai trò quan trọng của sự ủy thác biểu hiện ở hai mặt

- Lệnh commit thông tin cho hệ quản trị cơ sở dữ liệu phân tán – DBMS biết rằng kết quả thực hiện của giao dịch đến thời điểm này cần phải được phản ánh vào trong cơ sở dữ liệu. Qua đó làm cho các giao dịch đang truy xuất các mục dữ liệu đó có thể thấy được chúng.

- Điểm mà giao dịch Commit là một điểm “không quay trở lại” kết quả giao dịch đã commit bây giờ sẽ được lưu cố định trong cơ sở dữ liệu và không thể phục hồi lại trạng thái trước được.

b) *Ngược lại với Commit, nếu giao dịch dừng lại và chưa hoàn thành, nói rằng giao dịch đó bị hủy bỏ (Abort).* Một giao dịch bị hệ quản trị cơ sở dữ liệu hủy bỏ có thể vì bế tắc hay vì nhiều lý do khác. Ví dụ giao dịch sẽ bị hủy bỏ khi chuyến bay không còn vé. Khi một giao dịch bị hủy bỏ, tất cả quá trình thực thi sẽ bị ngừng lại và tất cả mọi hành động đã thực hiện phải được hồi phục lại (Undo), chuyển cơ sở dữ liệu về trạng thái trước khi thực hiện giao dịch, điều này được gọi là rollback.

Ví dụ 4.3:

Begin_transaction Reservation

Begin

Input (flight_no, date, customer_name)

EXEC temp1, temp2

FROM FLIGHT

```

WHERE FNO = flight_no
END DATE = date;
If temp1 = temp2 then
    begin
        Output("không còn ghế trống");
        Abort
    end
else begin
    EXEC SQL UPDATE FLIGHT
    SET STOLD = STOLD + 1
    WHERE FNO = flight_no
    END DATE = date;
    EXEC SQL INSERT
    INTO FC(FNO, DATE, CNAME, SPECIAL)
    VALUES(flight_no, date, customer_name, null);
    Commit;
    Output("giao dịch đã hoàn thành");
end.
end-if
end.

```

Giải thích hoạt động của giao dịch:

- Câu lệnh SQL đầu tiên sẽ chuyển STSOLD và CAP vào trong hai biến temp1 và temp2. Khác với giao dịch đã trình bày trong ví dụ 4.2 là cập nhật giá trị STSOLD vào trong cơ sở dữ liệu.

- So sánh temp1 và temp2 xem còn vé trên chuyến bay đó hay không. Giao dịch sẽ bị hủy bỏ nếu không còn vé. Ngược lại nếu còn vé thì giao dịch sẽ cập nhật lại giá trị STSOLD và chèn bộ mới vào trong quan hệ FC để ghi nhận rằng vé đã được bán.

Như vậy khi trên chuyến bay mà khách hàng đặt vé không còn chỗ trống thì giao dịch lập tức bị hủy bỏ.

4.1.2. Đặc tính của giao dịch

Giao dịch chỉ đọc và ghi dữ liệu, cơ sở cho việc nhận biết một giao dịch. Các đặc tính của giao dịch bao gồm:

- Các mục dữ liệu được giao dịch đọc tạo thành tập đọc RS (read set).
 - Các mục dữ liệu được giao dịch ghi được gọi là tập ghi WS (write set).
- Tập đọc hoặc tập ghi của giao dịch không nhất thiết phải tách biệt nhau.

- Hợp tập đọc và tập ghi của giao dịch tạo thành tập cơ sở $BS = RS \cup WS$.

Đặc trưng các giao dịch chỉ trên cơ sở các thao tác đọc và ghi mà không cần xem xét đến các thao tác chèn và xóa.

Ví dụ 4.4:

$RS = \{FLIGHT.STSODL, FLIGHT.CAP\}$

$WS = \{FLIGHT.STSODL, FC.DATE, FC.NO, FC.NAME, FC.SPECIAL\}$

$BS = \{FLIGHT.STSODL, FLIGHT.CAP, FC.NO, FC.NAME, FC.SPECIAL\}$.

4.1.3. Đặc trưng hóa khái niệm giao dịch

Định nghĩa khái niệm giao dịch một cách hình thức như sau:

Ký hiệu:

- Phép toán O_j của giao dịch T_i khi thực hiện trên thực thể x của cơ sở dữ liệu là $O_{ij}(x)$ theo qui ước $O_{ij} \in \{\text{read}, \text{write}\}$. Các phép toán giả sử là nguyên tố, nghĩa là mỗi phép toán được thực thi như là một đơn vị không thể chia nhỏ hơn được nữa.

- OS_i là tập tất cả các phép toán trong T_i , nghĩa là $OS_i = \bigcup_{j=1}^{N_i} O_{ij}$. N_i biểu thị cho tình huống kết thúc của T_i , trong đó $N_i \in \{\text{Abort}, \text{Commit}\}$.

Như vậy, có thể định nghĩa giao dịch T_i là một thứ tự bộ phận trên các phép toán và tình huống kết thúc của nó. Thứ tự bộ phận $P = \{\sum, \alpha\}$ định nghĩa một trật tự các phần tử của \sum (được gọi là miền qua một quan hệ hai ngôi bắc cầu và không phản xạ) được định nghĩa trên \sum bao gồm các phép toán và tình huống kết thúc của một giao dịch, trong đó α chỉ thứ tự thực hiện của các phép toán này. Một cách hình thức, một giao dịch T_i là một thứ tự bộ phận $T_i = \{\sum_{i,j} \alpha\}$, trong đó:

1. $\sum_{i,j} \alpha = OS_i \cup \{N_i\}$

2. Với hai phép toán bất kỳ $O_{ij}, O_{ik} \in OS_i$ nếu $O_{ij} = \{R(x) \text{ hoặc } W(x)\}$ và $O_{ik} = W(x)$ với mục dữ liệu x nào đó, khi đó hoặc $O_{ij} \alpha_i O_{ik}$ hoặc $O_{ik} \alpha_i O_{ij}$.

3. $\forall O_{ij} \in OS_i, O_{ij} \alpha_i N_i$.

Điều kiện thứ nhất định nghĩa trên miền như một tập hợp các thao tác đọc và ghi cấu tạo nên giao dịch với tình huống kết thúc, có thể là Commit hoặc Abort. Điều kiện thứ hai xác định qua quan hệ thứ tự giữa các thao tác đọc và ghi có tương tranh của giao dịch, điều kiện cuối cùng chỉ ra rằng tình huống kết thúc luôn đi sau tất cả các thao tác khác.

4.2. Các tính chất của giao dịch

4.2.1. Tính nguyên tử

Một giao dịch được xử lý như là một đơn vị thao tác. Hoặc tất cả các thao tác của giao dịch được hoàn thành, hoặc ngược lại, không có thao tác nào được hoàn thành. Tính nguyên tử đòi hỏi nếu thực thi của giao dịch bị ngắt bởi một 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 dịch để khôi phục lại sau sự cố. Có hai kiểu hoạt động: hoặc nó sẽ được kết thúc bằng cách hoàn thành các hoạt động còn lại, hoặc có thể được kết thúc bằng cách khôi phục lại tất cả các hoạt động đã được thực hiện.

Có hai loại sự cố, một giao dịch có thể có lỗi trong khi hoạt động do lỗi dữ liệu vào, bế tắc, hoặc các yếu tố khác. Trong các trường hợp này giao dịch tự hủy hoặc hệ quản trị cơ sở dữ liệu phải hủy giao dịch trong khi nó hủy bế tắc chẳng hạn. Duy trì được tính nguyên tử khi có hai sự hiện diện của hai loại sự cố thường được gọi là khôi phục giao dịch. Loại sự cố thứ hai là sự cố do hệ thống, như có sự cố thiết bị lưu trữ, sự cố của bộ xử lý, mất điện... Bảo đảm tính nguyên tử trong trường hợp này gọi là khắc phục sự cố.

Sự khác biệt giữa hai loại sự cố là trong một số loại tai nạn hệ thống, thông tin trong bộ nhớ chính có thể bị mất hoặc không truy cập được. Cả hai kiểu khôi phục đều là các vấn đề về độ tin cậy.

Mỗi nguyên tử là một thao tác cơ bản. Đảm bảo cho tính nguyên tử của các giao dịch là các phương pháp tuần tự hóa làm cho các thao tác của giao dịch thực hiện một cách tuần tự.

Một giao dịch không có tính nguyên tử nếu:

1. Trong hệ thống không phân chia theo thời gian, giao dịch T có thể kết thúc trong khi T đang tính toán và các hoạt động của giao dịch khác sẽ được thực hiện trước khi T hoàn tất. Trong trường hợp này hệ thống phải đảm bảo rằng, dù điều gì có xảy ra trong khi giao dịch thực hiện, thì cơ sở dữ liệu không bị ảnh hưởng bởi các tác động bất ngờ của giao dịch. Hoặc:

2. Một giao dịch chưa hoàn tất các công việc, phải chấm dứt hoạt động giữa chừng, có thể vì thực hiện các phép toán không hợp lệ (chia cho số 0), hoặc có thể do yêu cầu dữ liệu không được quyền truy xuất. Hệ thống buộc giao dịch phải ngừng hoạt động vì nhiều lý do. Trong trường hợp này hệ thống phải đảm bảo rằng, giao dịch hủy bỏ sẽ không ảnh hưởng gì lên cơ sở dữ liệu hoặc có không ảnh hưởng đến các giao dịch khác.

Mỗi giao dịch là một chuỗi các thao tác cơ bản như đọc, ghi các mục dữ liệu trên cơ sở dữ liệu và các phép toán xử lý đơn giản trong vùng làm việc, hoặc các bước khóa chốt và giải phóng khóa, hoặc ủy thác giao dịch.. là những bước nguyên tố, vì nó xảy ra trong vùng làm việc cục bộ, không bị ảnh hưởng bởi phép toán cho đến khi giao dịch tái khởi động trở lại.

4.2.2. Tính nhất quán

Tính nhất quán của giao dịch là tính đúng đắn của giao dịch. Nói rằng, một giao dịch có tính đúng đắn nếu ánh xạ cơ sở dữ liệu từ trạng thái nhất quán này sang trạng thái nhất quán khác. Việc xác nhận giao dịch nhất quán là vấn đề kiểm soát dữ liệu ngữ nghĩa. Đảm bảo tính nhất quán là mục tiêu của các cơ chế điều khiển đồng thời.

Tính nhất quán của các cơ sở dữ liệu chi thành bốn mức nhất quán:

Mức 3: Giao dịch T thỏa mãn nhất quán mức 3 nếu:

1. T không ghi đè lên dữ liệu rác của những giao dịch khác (Dirty Data là những giá trị dữ liệu được cập nhật bởi một giao dịch trước khi nó Commit).
2. T không Commit bất kỳ thao tác ghi nào cho đến khi nó hoàn tất mọi thao tác ghi (nghĩa là đến kết thúc giao dịch EOT – End Of Transaction).
3. T không đọc dữ liệu rác của các giao dịch khác.
4. Những giao dịch khác không làm cho dữ liệu mà T đã đọc trước khi T hoàn tất trở thành dữ liệu rác.

Mức 2: Giao dịch T thỏa mãn nhất quán mức 2 nếu:

1. T không ghi đè lên dữ liệu rác của những giao dịch khác;
2. T không commit bất kỳ thao tác ghi nào trước EOT;
3. T không đọc dữ liệu rác của những giao dịch khác.

Mức 1: Giao dịch T thỏa mãn nhất quán mức 1 nếu:

1. T không ghi đè lên dữ liệu rác của những giao dịch khác;
2. T không commit bất kỳ thao tác ghi nào trước EOT.

Mức 0: Giao dịch T thỏa mãn nhất quán mức 0 nếu:

1. T không ghi đè lên dữ liệu rác của những giao dịch khác.

Độ nhất quán mức cao bao trùm lên các mức độ nhất quán mức thấp hơn. Một số giao dịch hoạt tác ở mức nhất quán 3, các giao dịch khác có thể hoạt tác ở mức thấp hơn và rất có thể sẽ nhìn thấy các dữ liệu rác.

4.2.3. Tính cô lập

Tính cô lập đòi hỏi mỗi giao dịch phải luôn nhìn thấy cơ sở dữ liệu nhất quán. Nói cách khác, một giao dịch đang thực thi không thể làm lộ ra các kết quả của nó cho các giao dịch khác đang hoạt động trước khi nó commit. Tính nhất quán phải được duy trì qua lại giữa các giao dịch. Nếu hai giao dịch đồng thời truy cập đến cùng một mục dữ liệu đang được một trong số chúng cập nhật thì không thể bảo đảm rằng giao dịch thứ hai sẽ được giá trị đúng.

Bảo đảm tính cô lập bằng cách không cho phép các giao dịch khác nhìn thấy các kết quả chưa hoàn thành như trong ví dụ trên sẽ giải quyết được vấn đề cập nhật tổn thất (Lost Update). Nếu một giao dịch cho phép những giao dịch khác nhìn thấy những kết quả chưa hoàn thành của nó trước khi quyết định commit rồi quyết định hủy bỏ, thì mọi giao dịch đã đọc những giá trị chưa hoàn thành đó cũng sẽ phải hủy bỏ, gây tổn phí đáng cho hệ quản trị cơ sở dữ liệu.

Ví dụ 4.5: Xét hai giao dịch 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.

T ₁ : Read(x)	T ₂ : Read(x)
x ← x+1	x ← x+1
Write(x)	Write(x)
Commit	Commit

Dưới đây là một dãy các thực thi cho các hành động của những giao dịch này.

T₁: Read(x)
T₁: x ← x+1
T₁: Write(x)
T₁: Commit
T₂: Read(x)
T₂: x ← x+1
T₂: Write(x)
T₂: Commit

Giao dịch T₁ và T₂ thực hiện lần lượt. Giao dịch T₂ đọc được giá trị của x là 51. Nếu T₂ thực thi trước T₁ thì T₂ đọc được giá trị là 50. Như vậy nếu T₁ và T₂ được thực thi lần lượt, thì giao dịch thứ hai sẽ đọc được giá trị của x là 51 và sau

khi kết thúc hai giao dịch, x sẽ có giá trị 52. Nếu các giao dịch T_1 và T_2 thực thi đồng thời, giả sử dãy thực thi sau đây có thể xảy ra:

T_1 : Read(x)

T_1 : $x \leftarrow x+1$

T_2 : Read(x)

T_1 : Write(x)

T_2 : $x \leftarrow x+1$

T_2 : Write(x)

T_1 : Commit

T_2 : Commit

Khi đó giao dịch T_2 đọc được giá trị của x là 50, giá trị này không đúng, vì khi T_2 đọc x trong khi giá trị của x đang thay đổi từ 50 thành 51 và giá trị của x sẽ là 51 vào lúc kết thúc các giao dịch T_1 và T_2 . Hành động ghi của T_2 sẽ ghi đè lên kết quả ghi của T_1 . Như vậy các giao dịch T_1 và T_2 không cô lập lẫn nhau, nhìn thấy các kết quả chưa hoàn tất của nhau.

Loại cô lập cập nhật thất lạc (Lost Update), được gọi là tính ổn định con chạy (Cursor Stability). Trong ví dụ trên, dãy thực thi T_2 đã làm cho tác dụng của T_1 bị mất. Loại cô lập hủy bỏ dây chuyền (Cascading Abort), nếu một giao dịch cho phép các giao dịch khác nhìn thấy kết quả chưa hoàn tất của nó trước khi ủy thác rồi nó quyết định hủy bỏ, mọi giao dịch đã đọc những giá trị chưa hoàn tất đó cũng sẽ phải hủy bỏ. Xâu mắt xích này dễ dàng tăng nhanh và gây ra những phí tổn đáng kể cho hệ quản trị cơ sở dữ liệu.

Giữa tính chất cô lập và tính nhất quán của giao dịch cũng có những sự phụ thuộc lẫn nhau. Khi di chuyển lên cây phân cấp các mức nhất quán, các giao dịch càng cô lập hơn. Mức 0 cung cấp rất ít tính chất “cô lập” ngoài việc ngăn cản các cập nhật thất lạc. Tuy nhiên vì các giao dịch sẽ ủy thác trước khi hoàn tất tất cả các thao tác ghi, nếu có một hủy bỏ xảy ra sau đó, nó sẽ đòi hỏi phải bồi lại bởi những giao dịch khác. Nhất quán mức 2 tránh được các hủy bỏ dây chuyền. Mức 3 cung cấp toàn bộ khả năng cô lập, buộc một trong các giao dịch tương tranh phải đợi cho đến khi giao dịch kia kết thúc. Những dãy thực thi như thế được gọi là nghiêm ngặt (Strict). Như vậy tính cô lập của các giao dịch ảnh hưởng trực tiếp đến tính nhất quán cơ sở dữ liệu và đảm bảo tính nhất quán là mục tiêu của các cơ chế điều khiển đồng thời.

Các mức cô lập SQL được ANSI định nghĩa là những tình huống có thể xảy ra nếu sự cô lập thích hợp không được duy trì. Ba hiện tượng được đặc tả là:

1. *Đọc rác (Dirty read)*: Dữ liệu rác nghĩa là giá trị các mục dữ liệu đã bị sửa đổi bởi một giao dịch chưa ủy thác. Giao dịch T_1 sửa đổi một giá trị dữ liệu rồi nó lại được đọc bởi một giao dịch T_2 khác trước khi T_1 đã thực hiện Commit hay Abort. Trong trường hợp Abort, T_2 đã đọc một giá trị chưa hề tồn tại trong cơ sở dữ liệu. Một đặc tả chính xác của hiện tượng này như sau:

... $W_1(x), \dots, R_2(x), \dots, C_1(\text{hoặc } A_1), \dots, C_2(\text{hoặc } A_2)$

Hoặc

... $W_1(x), \dots, R_2(x), \dots, C_2(\text{hoặc } A_2), \dots, C_1(\text{hoặc } A_1)$

2. *Đọc không lặp lại (non – repeatable Read)*: Giao dịch T_1 đọc một dữ liệu. Sau đó một giao dịch T_2 khác sửa hoặc xóa mục dữ liệu đó rồi ủy thác. Nếu sau đó T_1 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 dịch T_1 trả về các kết quả khác nhau. Một đặc tả chính xác của hiện tượng này như sau:

... $R_1(x), \dots, W_2(x), \dots, C_1(\text{hoặc } A_2), \dots, C_2(\text{hoặc } A_2)$

Hoặc

... $R_1(x), \dots, W_2(x), \dots, C_2(\text{hoặc } A_2), \dots, C_1(\text{hoặc } A_1)$

3. *Ảnh ảo (phantom)*: Điều kiện ảnh ảo trước kia đã được định nghĩa xảy ra khi T_1 thực hiện tìm kiếm theo một vị từ và T_2 chèn những bộ giá trị mới thỏa vị từ đó. Đặc tả chính xác của hiện tượng này là (P là vị từ tìm kiếm)

..., $R_1(P), \dots, W_2(y \text{ thuộc } P), \dots, C_1(\text{hoặc } A_1), \dots, C_2(\text{hoặc } A_2)$

Hoặc

..., $R_1(P), \dots, W_2(y \text{ thuộc } P), \dots, C_2(\text{hoặc } A_2), \dots, C_1(\text{hoặc } A_1)$

Dựa trên những hiện tượng này, các mức cô lập đã được định nghĩa như sau:

- 1) *Đọc không ủy thác (Read Uncommitted)*: Với các giao dịch hoạt tác tại mức này, tất cả ba hiện tượng đều có thể.
- 2) *Đọc có ủy thác (Read Committed)*: Đọc bất khả lặp và ảnh ảo đều có thể nhưng đọc rác thì không.
- 3) *Đọc có lặp lại (Repeatable Read)*: Chỉ ảnh ảo là có thể.
- 4) *Khả năng tuần tự hóa bất thường (Anomaly Serializable)*: Không có hiện tượng nào có thể.

4.2.4. Tính bền vững

Tính bền vững (Durability) đảm bảo cho giao dịch đã Commit, kết quả của nó được duy trì cố định và không bị xóa bởi cơ sở dữ liệu. Vì vậy, hệ quản trị cơ sở dữ liệu bảo đảm kết quả của giao dịch vẫn tồn tại dù hệ thống có xảy ra sự cố gì. Tính bền vững theo nghĩa khôi phục cơ sở dữ liệu về trạng thái nhất quán khi đã Commit.

4.3. Các loại giao dịch

Có nhiều mô hình giao dịch tương ứng với các mô hình ứng dụng, được phân loại theo các quan điểm sau:

- Theo thời gian hoạt động của giao dịch;
- Theo tổ chức các hành động đọc và ghi;
- Theo cấu trúc bảng của chúng.

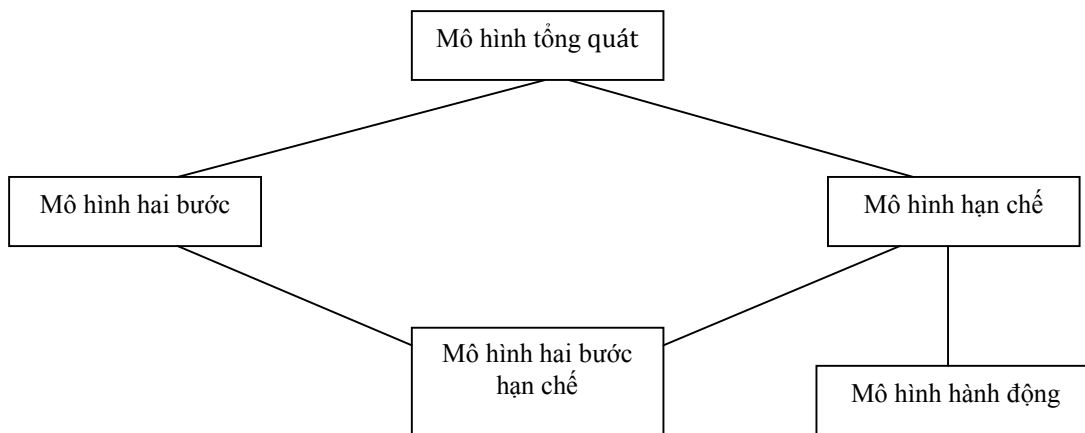
4.3.1. Các loại cấu trúc theo thời gian hoạt động

Các loại cấu trúc theo thời gian hoạt động bao gồm các loại theo kiểu trực tuyến (Online, hoặc Short Life, loại kiểu ba lô (Batch hoặc Long Life)). Các giao dịch trực tuyến đặc trưng bởi thời gian thực thi đáp ứng rất ngắn trong khoảng vài giây và truy cập một phần nhỏ cơ sở dữ liệu. Lớp giao dịch này bao quát phần lớn các ứng dụng hiện có. Ví dụ như các giao dịch ngân hàng và giao dịch đặt vé máy bay... Ngược lại, giao dịch kiểu lô sử dụng nhiều thời gian thực thi hơn (thời gian đáp ứng được tính bằng phút, giờ, ngày) và truy cập một phần khá lớn cơ sở dữ liệu. Các ứng dụng điển hình là các cơ sở dữ liệu CAD/CAM, các ứng dụng thống kê, tạo báo cáo, các truy vấn phức tạp và những xử lý hình ảnh. Có thể định nghĩa một giao dịch trực thoại thực hiện tương tác với người sử dụng đã đưa ra giao dịch.

4.3.2. Các loại giao dịch dựa trên việc tổ chức các hành động đọc và ghi

Có thể phân loại giao dịch dựa trên việc tổ chức các hành động đọc và ghi. Các giao dịch tổng quát (General) pha trộn các hành động đọc và ghi không theo bất kỳ một thứ tự cụ thể nào. Nếu một giao dịch bị hạn chế, buộc tất cả các hành động đọc ghi phải thực hiện trước hành động ghi, thì nó lại giao dịch hai bước (Two Step). Tương tự, nếu giao dịch bị buộc phải đo một mục dữ liệu trước khi cập nhật, thì giao dịch được gọi là hạn chế (Restricted) hay đọc trước ghi (Read before Write). Nếu một giao dịch vừa thuộc loại hai bước vừa thuộc

loại hạn chế, nó được gọi là hai bước hạn chế (Restricted Two Step). Cuối cùng là mô hình hành động (Action Mode) của các giao dịch, được cấu tạo bởi các loại hạn chế, yêu cầu từng cặp <đọc, ghi> được thực hiện theo kiểu nguyên tố.



Hình 4.1. Các mô hình giao dịch

4.3.3. Luồng công việc – WorkFlows

Trong quá trình quản lý các giao dịch, người ta nhận thấy rằng đối với các giao dịch ngắn và mối quan hệ đơn giản thì việc quản lý rất dễ dàng ngay cả việc thực thi cũng rất tốt. Tuy nhiên, khi dữ liệu trong hệ phân tán nhiều về số lượng và vị trí thì các giao dịch trở nên dài hơn. Mô hình giao dịch đơn giản không còn phù hợp cần thay thế bằng một mô hình giao dịch phức tạp hơn, kết hợp có tính mở các giao dịch với nhau.

Có thể hiểu “Workflow” – dòng công việc, là một tập hợp các tác vụ (Task) được tổ chức để hoàn thành một số tiến trình giao dịch (business Process).

Có ba loại luồng công việc như sau:

1. *Luồng công việc hướng con người* (Human Oriented) gồm con người trong việc thực hiện các tác vụ. Hỗ trợ của hệ thống được cung cấp nhằm tạo thuận lợi cho việc hợp tác và điều phối giữa con người với nhau nhưng chính con người phải chịu trách nhiệm bảo đảm tính nhất quán của các hành động.

2. *Luồng công việc hướng hệ thống* (System Oriented) chứa các tác vụ chuyên dùng và nặng về tính toán, có thể được máy tính thực hiện. Hỗ trợ của hệ thống trong trường hợp này rất đáng kể, bao gồm việc điều khiển đồng thời và khôi phục thực thi tác vụ tự động...

3. *Luồng công việc giao dịch* (transactional Workflow) nằm trong phạm vi

giữa luồng công việc hướng con người và luồng công việc hướng hệ thống và đặc trưng của chúng. Việc thực thi có điều phối của nhiều tác vụ:

a. Có thể có con người

b. Đòi hỏi truy cập đến các hệ thống HAD (hỗ trợ, tự trị và phân tán). Hỗ trợ việc sử dụng một cách chọn lọc các tính chất ACID của giao dịch cho từng tác vụ hoặc toàn bộ luồng công việc.

4. *Đặc điểm của luồng giao dịch hướng công việc này*: là dòng công việc đề cập đến thực thi kết hợp của nhiều tác vụ (task) liên quan đến:

a. Con người

b. Các yêu cầu truy cập đến hệ thống HAD (hệ thống HAD là hệ thống có 3 tính chất: hỗn tạp (Heterogeneous), tự trị (Autonomous), phân tán (Distributed).

c. Liên quan đến hỗ trợ lựa chọn sử dụng các đặc tính toàn tác cho các tác vụ riêng lẻ hoặc các dòng công việc toàn bộ.

Đặc điểm c là quan trọng nhất đối với các dòng công việc hoàn tác.

Có rất nhiều tổ chức về Transactional Workflow và theo nhiều cách khác nhau, nhưng đặc điểm chung mà về loại dòng công việc này đó là một tập hợp các tác vụ mà thứ tự quan hệ giữa chúng được định nghĩa tốt.

Ví dụ 4.6: Xét một ví dụ về dòng công việc, trong một hệ thống phục vụ khách du lịch có nhu cầu về đặt vé máy bay và thuê xe hơi. Mô hình các tác vụ trên các khách hàng (Database Customers), vé máy bay (Flight) và phát hành hóa đơn (Bills), được biểu diễn trong hình 4.2. Trong đó T_1 , T_2 , T_3 và T_4 là các tác vụ (Task). Có thể hiểu tác vụ là một thao tác thực hiện một vấn đề nào đó.

Customer Database là cơ sở dữ liệu chứa thông tin về khách hàng bao gồm các thông tin cá nhân và thông tin nhu cầu đặt vé máy bay và thuê xe.

- Cơ sở dữ liệu khách hàng được truy xuất để có thể được yêu cầu của khách hàng về đặt vé máy bay và thuê xe. Tác vụ T_1 là sự ghi nhận yêu cầu này.

- Tác vụ T_2 đặt vé máy bay bằng cách truy xuất vào cơ sở dữ liệu đặt vé máy bay và thực hiện đáp ứng yêu cầu khách hàng thuê xe và khách sạn.

- Tác vụ T_2 yêu cầu tác vụ T_3 trao đổi các thỏa thuận về xe hơi với các công ty cho thuê xe hơi.

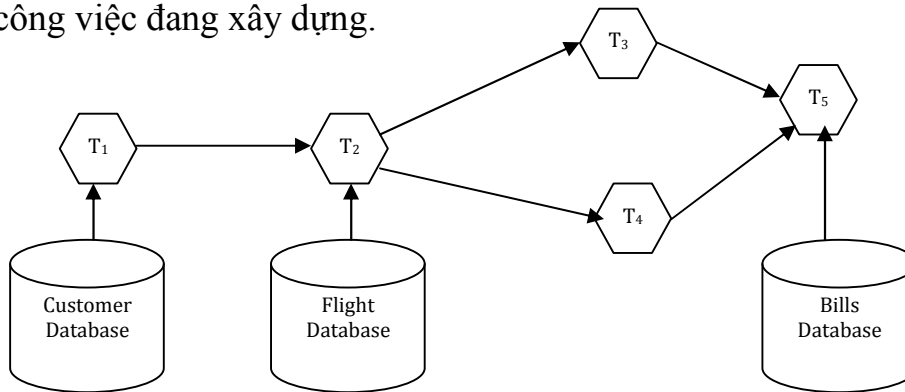
- Tác vụ T_2 yêu cầu tác vụ T_4 trao đổi các thỏa thuận về khách sạn.

- Tác vụ T_5 thao tác tính hóa đơn sau khi đã thỏa thuận mọi điều khoản đáp ứng các yêu cầu của khách lưu thông tin trong cơ sở dữ liệu hóa đơn.

Trong luồng công việc này tác vụ T_2 sẽ phụ thuộc vào T_1 , vì T_2 phải đáp ứng yêu cầu của T_1 và T_3 và T_4 phụ thuộc vào T_2 , vì T_3 và T_4 sinh ra sau khi T_2 thực hiện. T_5 phụ thuộc vào T_3 và T_4 , việc tính hóa đơn chỉ thực hiện khi các thỏa thuận đã hoàn tất.

Một luồng công việc được mô hình hóa như là một hoạt động (Activity), các ngữ nghĩa cho phép các kết quả chưa hoàn chỉnh được duyệt lại bởi các giới hạn bên ngoài của hoạt động trước đó, tức là các biên của các hoạt động. Vì vậy các tác vụ cấu thành một luồng công việc được cho phép cam kết một cách riêng biệt.

Các tác vụ có thể là các hoạt động khác cùng ngữ nghĩa giao dịch của luồng công việc đang xây dựng.



Hình 4.2. Một ví dụ về dòng công việc

4.4. Điều khiển các giao dịch đồng thời phân tán

4.4.1. Đặt vấn đề

Điều khiển các giao dịch một cách đồng thời là một vấn đề quan trọng trong môi trường cơ sở dữ liệu phân tán, càng có ý nghĩa khi nhu cầu thực hiện các giao dịch từ phía người sử dụng là rất lớn. Các giao dịch được tiến hành song song sẽ nâng cao hiệu năng hoạt động của hệ thống. Tuy nhiên việc thực hiện đồng thời các giao dịch sẽ nảy sinh nhiều vấn đề phức tạp. Một trong các vấn đề lớn nhất đó là sự tương tranh giữa các giao dịch cùng có nhu cầu sử dụng một mục dữ liệu. Làm thế nào để đảm bảo các giao dịch hoạt động có hiệu quả, đảm bảo được các tính chất biệt lập và nhất quán các giao dịch, tính độc lập và toàn vẹn dữ liệu. Đó chính là các mục tiêu của điều khiển đồng thời phân tán.

Điều khiển đồng thời phân tán giải quyết các vấn đề biệt lập (Isolation) và tính nhất quán (Consistency) của các giao dịch. Cơ chế điều khiển đồng thời phân tán của một hệ quản trị cơ sở dữ liệu là nhằm đảm bảo tính nhất quán của

cơ sở dữ liệu trong môi trường phân tán có nhiều người sử dụng. Nếu hệ quản trị cơ sở dữ liệu phân tán điều khiển hoạt động đơn giản các giao dịch phân tán, bằng cách thực hiện tuần tự các giao dịch, giao dịch sau bước vào hoạt động sau khi giao dịch trước đã hoàn tất. Phương pháp này hạn chế khả năng truy cập hệ thống đến tối thiểu, trong khi phần cứng của hệ thống có thể đáp ứng được cho nhiều truy cập cùng một lúc.

Mức độ đồng thời các giao dịch, nghĩa là số lượng các giao dịch cùng hoạt động một lúc, là một tham số quan trọng nhất trong các hệ phân tán. Vì vậy cơ chế điều khiển phân tán duy trì được mức độ đồng thời cao là tìm ra một phương án thích hợp vừa duy trì được tính nhất quán của cơ sở dữ liệu, vừa đảm bảo số lượng giao dịch.

Để giải quyết bài toán điều khiển đồng thời, giả sử hệ thống phân tán hoàn toàn tin cậy và không hề có bất cứ một sai sót nào về phần cứng cũng như phần mềm. Giả thiết này cho phép mô tả công việc quản lý đồng thời, không phải mô tả các hoạt động của hệ thống.

Lý thuyết tuần tự (Serializability Theory), là nền tảng cơ sở để xây dựng các thuật toán điều khiển đồng thời phân tán.

4.4.2. Tính khả tuần tự lịch biểu

Nếu việc thực thi đồng thời các giao dịch làm cho cơ sở dữ liệu ở vào trạng thái như khi thực hiện tuần tự theo một thứ tự nào đó thì các vấn đề như cập nhật bị thất lạc sẽ được giải quyết. Đây là điểm quan trọng của lý luận về tính khả tuần tự. Một lịch biểu S được định nghĩa trên tập giao dịch $T = \{T_1, T_2, \dots, T_n\}$ và xác định thứ tự thực thi đan xen lẫn nhau các thao tác trong giao dịch (lịch biểu tuần tự), khi có các sự cố tranh chấp chắc chắn không xảy ra và trong cơ sở dữ liệu sẽ được một kết quả nào đó. Lịch biểu có thể được mô tả như là một thứ tự bộ phận trên T .

Giả sử có tập k giao dịch $T = \{T_1, T_2, \dots, T_k\}$. Như vậy tương ứng với T có $k!$ (k giai thừa) các lịch biểu tuần tự khác nhau. Giả sử hoạt động của các giao dịch đồng thời là đúng đắn khi và chỉ khi tác động của nó cũng giống như tác động có được của một lịch biểu tuần tự. Một lịch biểu S (Schedule) cho một tập các giao dịch T là thứ tự (có thể xen kẽ) các bước cơ bản của các giao dịch được thực hiện.

Hai thao tác $O_{ij}(x)$ và $O_{kl}(x)$, i, k không nhất thiết phải phân biệt, cùng truy cập đến một thực thể cơ sở dữ liệu x được gọi là có tương tranh nếu ít nhất một trong chúng là thao tác ghi. Biết rằng, các thao tác đọc không tương tranh

với nhau, vì vậy, có hai loại tương tranh đọc – ghi và ghi – ghi. Hai thao tác này có thể cũng một giao dịch hoặc thuộc về hai giao dịch khác nhau và chúng tương tranh với nhau. Như vậy, sự tương tranh giữa các giao dịch sẽ xuất hiện giữa hai thao tác phụ thuộc vào thứ tự thực hiện của chúng.

a) Một lịch biểu đầy đủ (Complete Schedule)

Là lịch biểu định nghĩa thứ tự thực hiện cho tất cả các thao tác trong miền biến thiên của nó. Về hình thức, một lịch biểu đầy đủ S_T^c được định nghĩa trên một tập giao dịch $T = \{T_1, T_2, \dots, T_n\}$ là một thứ tự bộ phận $S_T^c = \{\Sigma_T, \alpha_T\}$ trong đó:

1. $\Sigma_T = \bigcup_{i=1}^n \Sigma_i$
2. $\alpha_T \supseteq \bigcup_{i=1}^n \alpha_i$
3. Hai thao tác tương tranh bất kỳ $O_{ij}, O_{kl} \in \Sigma_T$ có $O_{ij} \alpha O_{kl}$ hoặc $O_{kl} \alpha O_{ij}$

Điều kiện đầu tiên khẳng định miền biến thiên của lịch biểu là hợp của các miền biến thiên của từng giao dịch. Điều kiện thứ hai định nghĩa một quan hệ thứ tự là một tập bao hàm của các quan hệ thứ tự của các giao dịch. Điều này duy trì thứ tự thao tác bên trong mỗi giao dịch. Điều kiện cuối cùng định nghĩa thứ tự thực thi giữa các thao tác tương tranh.

Ví dụ 4.7: Xét hai giao dịch của ví dụ 4.7, được đặc tả:

T1: Read(x)	T2: Read(x)
$x \leftarrow x+1$	$x \leftarrow x+1$
Write(x)	Write(x)
Commit	commit

Một lịch biểu đầy đủ S_T^c trên $T = \{T_1, T_2\}$ có thể được biểu diễn như sau:

$$S_T^c = \{\Sigma_T, \alpha_T\}$$

Trong đó: $\Sigma_1 = \{R_1(x), W_1(x), C_1\}$

$$\Sigma_2 = \{R_2(x), W_2(x), C_2\}$$

Vì vậy:

$$\Sigma_T = \Sigma_1 \cup \Sigma_2 = \{R_1(x), W_1(x), C_1, R_2(x), W_2(x), C_2\}$$

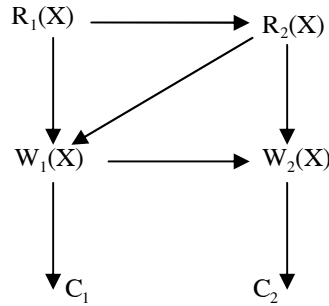
Và

$$\alpha_T = \{(R_1, R_2), (R_1, W_1), (R_1, C_1), (R_1, W_2), (R_1, C_2), \\ (R_2, W_1), (R_2, C_1), (R_2, W_2), (R_2, C_2), (W_1, C_1), (W_1, W_2), (W_1, C_2), \\ (C_1, C_2), (W_2, C_2)\}$$

Để đơn giản hóa ký pháp cho một lịch biểu thường được đặc tả như dãy các thao tác trong α_T , quy ước S_T^c có thể được đặc tả như sau:

$$S_T^c = \{R_1(x), R_2(x), W_1(x), C_1, W_2(x), C_2\}$$

Biểu diễn lịch biểu đầy đủ S_T^c dưới sơ đồ DAG như hình 4.3.



Hình 4.3. Biểu diễn DAG của một lịch biểu đầy đủ

b) Lịch biểu là một tiền tố (Prefix) của lịch biểu đầy đủ

Lịch biểu được định nghĩa như sau:

Cho một thứ tự bộ phận $P = \{\Sigma, \alpha\}$. Nói rằng $P' = \{\Sigma', \alpha'\}$ là một tiền tố của P nếu:

1. $\Sigma' \subseteq \Sigma$
2. $\forall e_i \in \Sigma'$, nếu $e_1 \alpha' e_2$ khi và chỉ khi $e_1 \alpha e_2$
3. $\forall e_i \in \Sigma'$, nếu $\exists e_j \in \alpha' \Sigma$ và $e_j \in e_j$ thì $e_i \in \Sigma'$

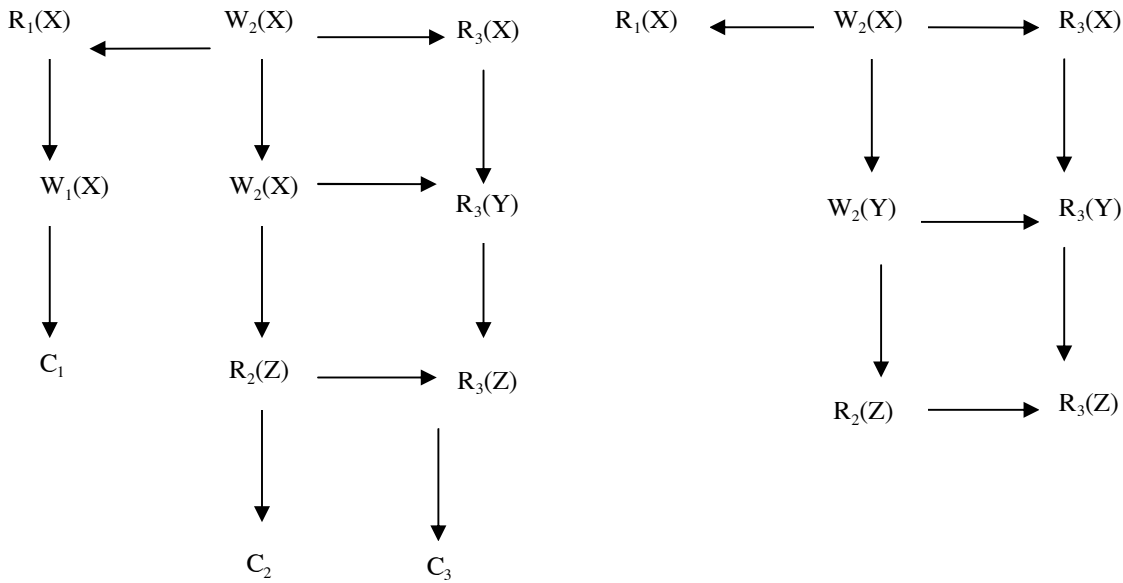
Điều kiện (1) và (2) xác định P' là một hạn chế của P trên miền Σ' và các quan hệ thứ tự trong P được duy trì trong P' . Điều kiện (3) chỉ ra rằng với mọi phần tử của Σ' , tất cả các phần tử đứng trước nó trong Σ cũng phải thuộc Σ' .

Dựa vào một lịch biểu tiền tố của thứ tự bộ phận có thể xử lý các lịch biểu không đầy đủ trong các trường hợp các giao dịch có tương tranh và khi xuất hiện sự cố cần phải có khả năng giải quyết với những giao dịch không đầy đủ.

Ví dụ 4.8: Minh họa một lịch biểu không đầy đủ. Xét ba giao dịch sau đây:

T1: Read(x)	T2: Write(x)	T3: Read(y)
Write(x)	Write(y)	Read(y)
Commit	Read(z)	Read(z)
	Commit	Commit

Một lịch biểu đầy đủ S^c cho những giao dịch này được trình bày trong hình 4.4 và một lịch biểu S là một tiền tố của S^c được mô tả trong hình 4.4b.



a) Một lịch biểu đầy đủ

b) Tiền tố của lịch biểu đầy đủ

Hình 4.4. Mô hình lịch biểu

c) *Lịch biểu tuần tự*

Nếu trong một lịch biểu S , các thao tác của các giao dịch khác nhau không được thực hiện xen kẽ, nghĩa là các thao tác của mỗi giao dịch xảy ra liên tiếp, khi đó lịch biểu này được gọi là tuần tự. Thực thi tuần tự là một tập các giao dịch duy trì được tính nhất quán của giao dịch, tức là duy trì được tính nhất quán của cơ sở dữ liệu. Mỗi giao dịch, khi được thực hiện độc lập trên cơ sở dữ liệu nhất quán, sẽ sinh ra một cơ sở dữ liệu nhất quán.

Ví dụ 4.9: Xét một lịch biểu của các giao dịch trong ví dụ 4.9 như sau:

$$S = \{W2(x), W2(y), R2(z), C2, R1(x), W1(x), C1, R3(x), R3(y), R3(z), C3\}$$

Lịch biểu S là tuần tự vì tất cả các thao tác của $T2$ được thực hiện trước các thao tác của $T1$ và các thao tác của $T1$ được thực hiện trước các thao tác của $T3$.

Có thể biểu diễn mối liên hệ thứ bậc giữa các thực thi giao dịch là:

$$T2 \alpha_S T1 \alpha_S T3 \text{ hoặc } T2 \rightarrow T1 \rightarrow T3$$

d) *Sự tương đương của các lịch biểu*

Nói rằng, hai lịch biểu S_1 và S_2 được định nghĩa trên cùng một tập hợp giao dịch T là tương đương nếu chúng có cùng tác dụng trên cơ sở dữ liệu. Nói

cách khác, hai lịch biểu S_1 và S_2 được định nghĩa trên cùng một tập giao dịch T được gọi là tương đương nếu với mỗi cặp thao tác tương tranh O_{ij} và O_{kl} thì $O_{ij} \alpha O_{kl}$. Gọi là tương đương tương tranh vì sự tương đương của hai lịch biểu theo thứ tự thực thi tương đối của các thao tác tương tranh trong các lịch biểu. Giả thiết T không chứa các giao dịch bị hủy bỏ.

Như vậy dựa vào mối liên hệ thứ bậc của các thứ tự bộ phận, có thể định nghĩa sự tương đương của các lịch biểu tương ứng tác dụng của chúng trên cơ sở dữ liệu.

Ví dụ 4.10: Cho hai lịch biểu sau được định nghĩa trên ba giao dịch của ví dụ 4.9:

$$S = \{W_2(x), W_2(y), R_2(z), C_2, R_1(x), W_1(x), C_1, R_2(x), R_3(x), R_3(y), R_3(z), C_3\}.$$

$$S' = \{W_2(x), R_1(x), W_1(x), C_1, R_3(x), W_2(y), R_3(y), R_2(z), C_2, R_3(z), C_3\}.$$

Hiển nhiên lịch biểu S và S' là tương đương tương tranh.

e) Tính khả tuần tự của lịch biểu

Một lịch biểu S được gọi là khả tuần tự hay khả tuần tự theo tương tranh khi và chỉ khi nó tương đương tương tranh với một lịch biểu tuần tự. Lưu ý rằng tính khả tuần tự chỉ tương đương với tính nhất quán mức 3.

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 biểu khả tuần tự để thực hiện các giao dịch đang chờ đợi. Như vậy, vấn đề là điều chỉnh lại các thuật toán để đảm bảo rằng chúng chỉ sinh ra các lịch biểu khả tuần tự.

Lý thuyết khả tuần tự có thể mở rộng cho các hệ cơ sở dữ liệu phân tán không nhân bản bằng phương pháp đơn giản. Lịch biểu thực thi giao dịch tại mỗi vị trí được gọi là lịch biểu cục bộ. Nếu cơ sở dữ liệu không được nhân bản và mỗi lịch biểu cục bộ khả tuần tự thì lịch biểu toàn cục cũng khả tuần tự, với điều kiện là các thứ tự tuần tự hóa cục bộ đều như nhau. Tuy nhiên, trong các cơ sở dữ liệu phân tán có nhân bản, có thể các lịch biểu cục bộ có khả năng tuần tự nhưng tính nhất quán tương hỗ của cơ sở dữ liệu vẫn bị tổn hại.

Ví dụ 4.11: Giả sử hai giao dịch T_1 và T_2 cùng truy cập vào một mục dữ liệu (x) có mặt tại hai vị trí. Giả sử có hai lịch biểu được tạo ra tại hai vị trí.

$$S1 = \{R_1(x), W_1(x), C_1, R_2(x), W_2(x), C_2\}$$

$$S2 = \{R_2(x), W_2(x), C_2, R_1(x), W_1(x), C_1\}$$

T1: Read(x)	T2: Read(x)
$x \leftarrow x + 1$	$x \leftarrow x + 1$
Write(x)	Write(x)
Commit	Commit

Hiển nhiên hai lịch biểu S_1 và S_2 là những lịch biểu khả tuần tự, mỗi lịch biểu biểu diễn theo một thứ tự thực hiện đúng đắn. Tuy nhiên, có thể nhận thấy các lịch biểu tuần tự hóa T_1 và T_2 theo thứ tự đảo ngược. Giả sử giá trị của x trước khi thực hiện các giao dịch là 1. Sau khi thực hiện lịch biểu S_1 tại vị trí 1, giá trị của x là 4. Giá trị x là 15 tại vị trí 2, sau khi kết thúc lịch biểu S_2 . Điều này vi phạm tính nhất quán tương hỗ của hai cơ sở dữ liệu cục bộ.

f) Lịch biểu khả tuần tự một bản (One – Copy Serializable)

Các lịch biểu có thể duy trì được tính nhất quán tương hỗ được gọi là khả tuần tự một bản. Tính nhất quán tương hỗ yêu cầu tất cả các giá trị của mọi mục dữ liệu nhân bản phải như nhau. Một lịch biểu toàn cục khả tuần tự một bản phải thỏa mãn những điều kiện sau:

1. Một lịch biểu cục bộ phải khả tuần tự;
2. Hai thao tác tương tranh phải có cùng thứ tự tương đối trong tất cả các lịch biểu cục bộ nơi mà chúng cùng xuất hiện.

Các thuật toán điều khiển đồng thời bảo đảm được tính khả tuần tự bằng cách đồng bộ hóa các truy cập tương tranh đến cơ sở dữ liệu. Trong các trường hợp dữ liệu nhân bản, nhiệm vụ bảo đảm tính khả tuần tự một bản thường là trách nhiệm của giao thức điều khiển bản sao.

Giả sử một mục dữ liệu x có các bản sao x_1, x_2, \dots, x_n . Coi x như là một mục dữ liệu logic và mỗi bản sao là một mục dữ liệu vật lý. Nếu tính trong suốt nhân bản được cung cấp, các giao dịch của người sử dụng sẽ đưa ra các thao tác đọc và ghi trên mục dữ liệu x . Giao thức điều khiển bản sao chịu trách nhiệm ánh xạ mỗi thao tác đọc trên mục dữ liệu logic x [$read(x_j)$] thành một thao tác đọc trong những bản sao vật lý. Ngược lại, mỗi thao tác ghi trên mục logic x được ánh xạ thành một tập các thao tác ghi trên một tập con của các bản sao vật lý x_j của x . Bất kể ánh xạ chuyển đến toàn bộ 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. Giao thức

đọc một/ghi tất cả (RO/WA – Read one/Write All protocol) 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. Nhược điểm của giao thức ROWA là giảm độ khả dụng của cơ sở dữ liệu khi có sự cố, giao dịch có thể không hoàn tất được trừ khi nó đã phản ánh tác dụng của tất cả các thao tác ghi trên các bản trừ khi nó đã phản ánh tác dụng của tất cả các thao tác ghi trên các bản sao. Vì vậy có một số thuật toán cố gắng duy trì tính nhất quán tương hỗ mà không sử dụng đến giao thức ROWA.

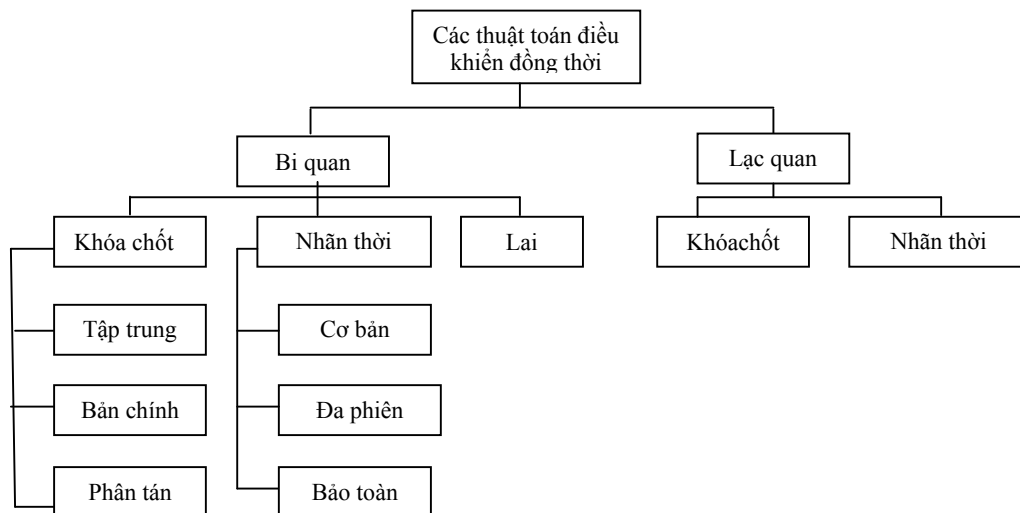
4.4.3. Phân loại các cơ chế điều khiển đồng thời

Có nhiều cách phân loại các cơ chế điều khiển đồng thời các giao dịch phân tán. Thông thường cách phân loại dựa trên sự nguyên thủy về sự đồng bộ hóa (Synchronization Primitive). Sự nguyên thủy có thể được sử dụng trong các thuật toán điều khiển đồng thời với hai quan điểm khác nhau như sau:

- Quan điểm bi quan: cho rằng có rất nhiều giao dịch tương tranh với nhau;
- Quan điểm lạc quan: cho rằng không có nhiều giao dịch tương tranh với nhau.

Vì vậy, có thể phân loại các cơ chế điều khiển đồng thời các giao dịch thành hai nhóm: phương pháp điều khiển đồng thời lạc quan và phương pháp điều khiển đồng thời bi quan. Các thuật toán bi quan đồng bộ hóa việc thực hiện các giao dịch trước khi thực hiện chúng, trong khi đó các thuật toán lạc quan đồng bộ hóa các giao dịch cho đến khi chúng kết thúc. Nhóm bi quan gồm các thuật toán dựa trên khóa chốt (Lockinh Based Algorithm), các thuật toán dựa theo thứ tự giao dịch 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 chốt và các thuật toán theo thứ tự thời gian. Được trình bày trong hình 4.5.

Từ hình vẽ, rõ ràng sự phân chia các thuật toán điều khiển đồng thời phân tán dựa trên hai phương pháp bi quan và lạc quan. Các thuật toán sẽ được chia thành ba loại: khóa chốt, theo nhãn thời gian và lai. Trong quan điểm bi quan mỗi loại thuật toán lại được chia nhỏ ra thành các dạng khác nhau. Đối với loại khóa chốt bao gồm có các dạng: tập trung, bản chính và phân tán. Đối với loại theo nhãn thời gian bao gồm các dạng: tập trung, bản chính và phân tán. Đối với loại theo nhãn thời gian bao gồm các dạng: cơ bản, đa phiên bản và bảo trì.



Hình 4.5. Phân loại các thuật toán điều khiển đồng thời

Trong cách tiếp cận dùng khóa chốt, việc đồng bộ hóa giao dịch có được bằng cách sử dụng các khóa vật lý hoặc logic trên một phần của cơ sở dữ liệu. Kích thước thường được gọi là độ mịn của khóa (Locking Granularity), giả sử kích thước được chọn là một đơn vị khóa (Lock Unit). Khóa chốt là một đặc quyền truy xuất trên một mục dữ liệu mà bộ quản lý khóa chốt có thể trao cho một giao dịch hay thu hồi lại. Thông thường tại mỗi thời điểm chỉ có một tập con các mục bị khóa chốt, vì vậy bộ quản lý khóa chốt có thể lưu các khóa chốt hiện tại trong các bảng khóa (Lock Table) với cấu trúc (I, L,T), trong đó T là giao dịch có một khóa chốt kiểu L trên mục I. Có thể có nhiều giao dịch với nhiều kiểu khóa chốt trên cùng một mục dữ liệu.

Lớp cơ chế theo thứ tự nhãn thời gian TO (Timestamp Ordering) phải tổ chức thứ tự thực hiện, các thứ tự thực hiện của các giao dịch nhằm duy trì được tính nhất quán và tương hỗ giữa các vị trí (liên nhất quán). Việc sắp xếp thứ tự này được duy trì bằng cách gán nhãn thời gian cho cả giao dịch lẫn mục dữ liệu được lưu trữ trong cơ sở dữ liệu. Những thuật toán này có thể thuộc loại cơ bản (Base TO), đa phiên bản hoặc bảo toàn. Phương pháp theo thứ tự thời gian đảm bảo không có hai giao dịch cùng nhãn thời gian và thứ tự của chúng là tương đối so với nhãn thời gian tương ứng với thứ tự trong đó giao dịch bắt đầu. Một cách khác là sử dụng trị số đồng hồ hệ thống tại thời điểm một tiến trình bắt đầu làm nhãn thời gian.

Một số thuật toán sử dụng khóa chốt cũng có thể dùng nhãn thời gian, chủ yếu nhằm cải thiện hiệu quả và mức độ hoạt động đồng thời. Lớp thuật toán này được gọi là lớp thuật toán lai. Chúng chưa hề được cài đặt trong các hệ quản trị

cơ sở dữ liệu phân tán thương mại và các hệ thử nghiệm.

4.5. Các thuật toán điều khiển đồng thời bằng khóa chốt

Một số định nghĩa:

Declare –Type

Operation: Các toán tử Begin_transaction, Read, Write, Abort, hoặc Commit.

DataItem: Một mục dữ liệu trong cơ sở dữ liệu phân tán.

TransactionId: Mã là duy nhất của mỗi giao dịch.

DataVal: Một giá trị có kiểu dữ liệu cơ bản (số nguyên, số thực...).

SiteId: Định danh duy nhất cho vị trí.

Dbop: Một bộ ba gồm:

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 dịch.

Message ← một thông báo cần được chuyển đi

OpSet: Một tập các Dbop

SiteSet: Một tập các Site Id

WAIT (msg:Message)

Begin

{Đợi một thông báo đến}

End

4.5.1. Thuật toán quản lý khóa cơ bản

1. Khi giao dịch có yêu cầu đọc hoặc ghi vào một mục cơ sở dữ liệu chứa khóa, bộ phận quản lý giao dịch sẽ chuyển cho bộ quản lý khóa các thao tác đọc, ghi và các thông tin như mục dữ liệu cần truy xuất, định danh của giao dịch đưa ra yêu cầu...

2. Bộ quản lý sẽ thực hiện kiểm tra đơn vị khóa có chứa mục điệu đã bị khóa hay chưa, tức là mục dữ liệu này đã được sử dụng bởi một giao dịch khác hay chưa.

3. Bộ quản lý giao dịch sẽ được trả về các kết quả của các hoạt động. Khi giao dịch kết thúc nó sẽ giải phóng các khóa mà nó đã sử dụng và khởi tạo và giao dịch khác cũng đang chờ để truy cập vào khoản mục dữ liệu khác.

Thuật toán quản lý khóa cơ bản (Basic lock manger)

Declere-var: msg: Massage, Dop: dbop, Op: Operation,

x: DataItem, TransactionId, pm: Dpmsg,

res: Data Val, SOP: OpSet

Begin

Repeat

WAIT(msg)

Case of msg

Dbop:

Begin

Op \leftarrow dop.opn

X \leftarrow dop.data

T \leftarrow Dop.tid

Case of Op

Begin_transaction:

Begin

Gửi Dop đến bộ xử lý dữ liệu

End

Read or Write (Yêu cầu khóa chốt)

Begin

Tìm một đơn vị khóa lu sao cho $x \sqsubseteq lu$ If lu chưa bị khóa hoặc khóa

Iu tương thích với Op then
 Begin
 Thiết lập khóa trên Iu với chế độ thích hợp
 Gửi dop đến bộ xử lý dữ liệu
 End
 Else đưa dop vào hàng đợi của Iu
 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
 Tìm khóa Iu sao cho $x \models Iu$ giải phóng khóa trên Iu do T giữ
 If không có khóa nào trên Iu and không có một hoạt động nào
 đang đợi trong hàng đợi của khóa Iu then
 Begin
 SOP \leftarrow hoạt động đầu tiên trong hàng đợi
 SOP \leftarrow SOP $\cup \{O \mid O \text{ là một hoạt động trong hàng đợi mà khóa Iu}$
 ở chế độ tương thích với các hoạt động trong SOP}
 Thiết lập các khóa trên Iu đại diện cho các hoạt động trong SOP
 For all các hoạt động trong SOP do
 Gửi mỗi hoạt động đến bộ xử lý dữ liệu
 End-for
 End-if
 End
 End-case

Until forever
End. {Basic LM}

Thuật toán khóa chốt cơ bản là một thuật toán đơn giản. Tuy nhiên thuật toán không đồng bộ hóa sự thực thi giao dịch một cách hoàn toàn đúng đắn, vì khi đưa ra các lịch biểu khả tuần tự, các thao tác khóa chốt và giải phóng khóa chốt cũng đòi hỏi phải được sắp xếp phối hợp. Vì vậy, có thể dẫn đến việc mất đi tính cô lập và tuyến tính từ tổng thể của giao dịch.

Ví dụ 4.12: Xét hai giao dịch sau đây:

T1: Read (x)	T2: Read(x)
$x \leftarrow x+1$	$x \leftarrow x*2$
Write (x)	write(x)
Read(y)	Read(y)
$Y \leftarrow y-1$	$Y \leftarrow y*2$
Write(y)	Write(y)
Commit	Commit

Thuật toán khóa bộ quản lý khóa cơ bản tạo ra một lịch biểu S như sau:

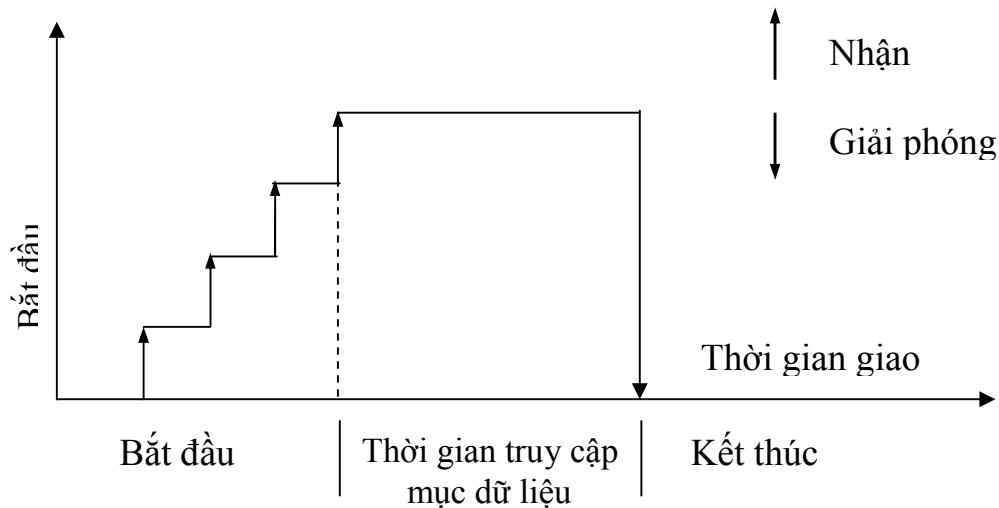
$S = \{w_1(x), R_1(x), W_1(x), lr_1(x), w_2(x), r_2(x), w_2(x), lr_2(x), w_2(x), r_2(y), lr_2(y), C_2, w_1(y), r_1(y), w_1(y), lr_1(y), c_1\}$.

4.5.2. Thuật toán khóa chốt hai pha 2PL

Thuật toán khóa chốt 2 pha (two phase Locking- 2PL) giải quyết triệt để những nhược điểm về điều khiển đồng thời các giao dịch trong thuật toán khóa chốt. Dễ dàng mở rộng cho các môi trường phân tán, bằng cách 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 dịch ở các vị trí khác nhau phải giao tiếp với bộ quản lý khóa. Cách tiếp cận này được gọi là thuật toán 2 PL- vị trí chính (primary site).

Thuật toán khóa chốt 2 pha (2PL) khẳng định rằng không có giao dịch nào yêu cầu khóa sau khi nó đã giải phóng một trong các khóa của nó. Nghĩa là một giao dịch không giải phóng khóa cho đến khi nó không yêu cầu thêm khóa. Các giao dịch trong các thuật toán 2 PL thực hiện hai pha, trong pha tăng trưởng nó nhận các khóa và truy cập các mục dữ liệu và pha thu hồi, nó giải phóng các khóa. Điểm khóa là điểm giao dịch đã nhận được tất cả các khóa nhưng chưa bắt đầu giải phóng bất kỳ khóa nào. Vì thế điểm khóa được xác định cuối pha tăng trưởng và khởi đầu pha thu hồi của một giao dịch. Tất cả các lịch biểu tạo bởi

thuật toán điều khiển đồng thời tuân theo quy tắc 2 PL là khá tuần tự.



Hình 4.6. Biểu đồ khóa chốt hai pha nghiêm ngặt

Thuật toán khóa chốt 2 pha làm tăng hoạt động đồng thời, bộ quản lý khóa cho phép các giao dịch đợi khóa tiếp tục truy cập mục dữ liệu và nhận khóa. Tuy nhiên bộ quản lý khóa không biết giao dịch đã nhận đủ khóa và còn truy cập mục dữ liệu hay không. Nếu giao dịch bị hủy bỏ sau khi giải phóng một khóa, nó có thể làm hủy bỏ luôn cả các giao dịch đã truy cập các mục đã mở khóa.

Thuật toán khóa chốt hai pha nghiêm ngặt (Strict two phase Locking), thực hiện giải phóng tất cả các khóa vào lúc giao dịch kết thúc. Bảo đảm các khóa chỉ được giải phóng nếu thao tác là Commit hay Abort.

Thuật toán khóa chốt hai pha nghiêm ngặt S2PL-LP

Declare-var: msg: Message, dop: Dbop, op: Operation,
x: DataItem, T: transactionId, pm: Dpmsg,
res: Data val, SOP: OpSet

Begin

Repeat

WAIT(msg)

Case of msg

Dbop:

Begin

Op ← dop.opn

x ← dop.data

$t \leftarrow dop.tid$

Case of Op

Begin_Transaction:

Begin

Gửi dop đến bộ xử lý dữ liệu

End

Read or Write: {yêu cầu khóa}

Begin

Tìm đơn vị khóa lu sao cho $x \subseteq lu$

If lu chưa bị khóa or chế độ khóa của lu tương thích với Op then

Begin

Đặt khóa trên lu ở chế độ 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:

begin

$op \leftarrow pm.opm$

$Res \leftarrow pm.result$

$T \leftarrow pm.tid$

if Op= Abort or Op= Commit then

begin

```

For mỗi đơn vị khóa Iu bị khóa bởi T do
begin
giải phóng khóa trên Iu do T giữ
If không còn khóa nào trên Iu and và có các thao tác đang đợi trong hàn
    đợi cho Iu then
begin
SOP ← thao tác đầu tiên trong hàng đợi
SOP ← SOP ∪ {O | O là một thao tác trên hàng đợi có thể khóa Iu ở chế
độ
        khóa tương thích với các thao tác hiện hành trong SOP}
đặt các khóa trên Iu 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-for
end
end-case
until forever
end. {S2PL-LM}

```

Thuật toán: Bộ quản lý khóa giao dịch 2PL (2PL-TM)

Declare-var: msg: Messenge, dop: Dbop, op: Operation,
x: DataItem, T: TransactionId, pm: Dpmsg,
res: Data val, SOP: OpSet

```

Begin
Repeat
WAIT(msg)
Case of msg
Dbop:
Begin

```

Op ← dop.opn

x ← dop.data

T ← dop.tid

Case of Op

Read:

Begin

 Trả res về cho ứng dụng của người dùng
 (nghĩa là giao dịch)

End

Write: {yêu cầu khóa}

Begin

 Thông tin cho ứng dụng về việc hoàn tất hành động
 ghi trả res về cho ứng dụng

end

Begin

Commit:

Begin

 hủy vùng làm việc của T

thông tin cho ứng dụng biết về việc hoàn thành giao dịch T

end

Abort:

Begin

 thông tin cho ứng dụng biết về việc đã hủy bỏ giao dịch T

end

end-case

until forever

end. {S2PL-TM}

4.5.3. Thuật toán quản lý giao dịch 2PL tập trung (C2PL TM)

Một vị trí duy nhất trên mạng máy tính (vị trí trung tâm) được cài đặt bộ quản lý khóa. Các bộ quản lý giao dịch ở các vị trí khác phải giao tiếp với vị trí chính. Phương pháp giao tiếp này được gọi là thuật toán quản lý giao dịch 2PL

tập trung cho các giao dịch đồng thời phân tán.

Khi thực hiện một giao dịch, bộ quản lý giao dịch của vị trí TM điều phối - nơi khởi đầu giao dịch yêu cầu vị trí LM trung tâm trao khóa cho nó. Khi đã được trao khóa, nó yêu cầu các vị trí chứa các bộ xử lý dữ liệu khác tham gia vào giao dịch. Kết thúc giao dịch, TM điều phối giải phóng khóa.

1. TM điều phối – khởi đầu giao dịch yêu cầu LM cung cấp khóa.
2. TM trung tâm trao khóa cho TM điều phối.
3. TM điều phối yêu cầu các bộ xử lý dữ liệu tại các vị trí tham gia giao dịch.
4. Sau khi hoàn tất các thao tác giao dịch, TM điều phối giải phóng khóa, trao quyền điều khiển cung cấp khóa cho LM trung tâm.

Thuật toán C2PL –TM phân tán khác với thuật toán TM tập trung là sự cài đặt giao thức điều khiển bản sao nếu cơ sở dữ liệu được nhân bản. Nó cũng khác với thuật toán bộ quản lý khóa 2PL nghiêm ngặt là bộ quản lý khóa trung tâm không gửi các thao tác đến các bộ xử lý dữ liệu tương ứng mà do LM điều phối thực hiện.

Thuật toán 4.4: Quản lý giao dịch 2PL tập trung (C2PL- TM)

Declare-var: msg: Message, dop: Dbop, op: Operation,
x: DataItem, T: transactionId, pm: Dpmsg,
res: Data val, 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
         $S \leftarrow \emptyset$ 
    End
Read:
    Begin
         $S \leftarrow S \cup \{\text{trạm lưu } x \text{ và có chi phí truy cập đến } x \text{ là nhỏ nhất}\}$ 
        gửi O cho bộ quản lý khóa trung tâm
    End
Write:
    Begin
         $S \leftarrow S \cup \{S_i | x \text{ được lưu tại } S_i\}$ 
        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
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 trong S
    Else
        Thông tin cho người dùng biết về việc kết thúc giao dịch
    End-if
End
Dpmsg:
    Begin
         $op \leftarrow pm.opm$ 

```



```

Res ← pm.result
T ← pm.tid:
Case of Op
Read:
begin
  Trả res về cho ứng dụng người dùng (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
  Commit then
  Begin
    thông tin ứng dụng biết về việc hoàn thành giao dịch
    gửi pm cho bộ quản lý trung tâm
  End
  Else {đợi cho đến khi tất cả đều nhận được các thông báo Commit}
  Ghi nhận sự kiện thông báo commit đến tất cả các trạm
  End-if
end
Abort:
Begin
  thông tin ứng dụng cho biết về việc hủy bỏ giao dịch T
  gửi pm đến bộ quản lý khóa trung tâm
  End
End-case
End
End-case

```

until forever

end. {C2PL-TM}

Thuật toán quản lý khóa 2PL tập trung (C2PL-LM)

Declare-var: msg: Message, dop: Dbop, op: Operation,
x: DataItem, T: transactionId, pm: Dpmsg,
res: Data val, SOP: OpSet

Begin

Repeat

WAIT(msg)

Op \leftarrow dop.opn

x \leftarrow dop.data

t \leftarrow dop.tid

Case of Op

Read or Write:

Begin

Tìm đơn vị khóa lu sao cho $x \subseteq lu$

If lu chưa bị khóa or chế độ khóa của lu tương thích với Op then

Begin

Đặt khóa trên lu ở chế độ thích hợp

Msg \leftarrow “khóa được trao cho thao tác dop”

Gửi msg đến TM điều phối của T

End

Else

đặt Op vào một hàng đợi của lu

end-if

end

Abort or commit:

begin

for mỗi đơn vị khoá lu bị khóa bởi 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 \leftarrow thao tác đầu tiên (gọi O) từ hàng đợi

SOP \leftarrow SOP $\cup \{O \mid O \text{ là một thao tác trên hàng đợi có thể khóa lu ở chế độ tương thích với các thao tác trong SOP}\}$
đặt các khóa trên lu cho các thao tác trên SOP

for tất cả các phép toán trong SOP do

Begin

Msg \leftarrow “khóa được trao cho thao tác dop”

Gửi msg đến TM điều phối

End-for

End-if

End-for

Msg \leftarrow “các khóa của T đã giải phóng”

Gửi msg đến TM điều phối của T

End

End- case

until forever

end. {C2PL-LM}

4.5.4. Thuật toán 2PL bản chính

Nhược điểm của các thuật toán C2PL là có thể tắc nghẽn tại vị trí LM trung tâm vì vậy độ tin cậy và hiệu năng thuật toán thấp. Thuật toán khóa chốt hai pha bản chính là sự cải tiến thuật toán khóa hai pha tập trung với mục đích cải thiện các vấn đề về hiệu năng thuật toán. Về cơ bản, nó cài đặt các bộ quản lý khóa trên một số vị trí và giao trách nhiệm quản lý một tập đơn vị khóa cho mỗi bộ quản lý. Bộ quản lý giao dịch sẽ gửi các yêu cầu khóa và mở khóa đến các bộ quản lý khóa. Thuật toán sẽ xử lý một bản của mỗi mục dữ liệu như bản chính của nó.

Phiên bản phân tán của thuật toán 2PL bản chính đã được đề xuất thử nghiệm trong hệ INGRES. Đòi hỏi phải có một thư mục phức tạp tại mỗi vị trí,

nhưng đã giảm tải cho vị trí trung tâm mà không phải trao đổi quá nhiều giữa các bộ quản lý giao dịch và các bộ quản lý khóa. Thuật toán 2PL bản chính là bước trung gian giữa thuật toán 2PL tập trung và thuật toán 2PL phân tán.

4.5.5. Thuật toán 2PL phân tán

Thuật toán 2PL phân tán cài đặt bộ quản lý khóa trên mỗi vị trí. Nếu cơ sở dữ liệu không nhân bản, thuật toán 2PL phân tán sẽ suy biến từ thuật toán 2PL bản chính. Nếu có cơ sở dữ liệu có nhân bản, giao dịch sẽ đặt giao thức điều khiển bản sao ROWA. Trao đổi thông tin giữa các vị trí để thực hiện một giao dịch xảy ra giữa bộ quản lý giao dịch tại vị trí khởi đầu giao dịch – TM điều phối với bộ quản lý khóa và các bộ xử lý dữ liệu tại các vị trí có tham gia.

1. TM điều phối- vị trí khởi đầu của giao dịch yêu cầu các vị trí DM có tham gia xử lý giao dịch cấp khóa và tham gia xử lý giao dịch cung cấp khóa và tham gia xử lý dữ liệu.

2. Sau khi các DM đã hoàn tất các thao tác giao dịch, TM điều phối yêu cầu giải phóng khóa.

Thuật toán quản lý giao dịch 2PL phân tán khác với thuật toán quản lý giao dịch tập trung 2PL- TM trong hai vấn đề:

1. Trong C2PL - TM: các thông báo gửi đến bộ quản lý khóa của vị trí trung tâm. Trong C2PL – TM: các thông báo được gửi đến các bộ quản lý khóa của tất cả các vị trí tham gia trong D2PL- TM.

2. TM điều phối không chuyển các thao tác đến các bộ xử lý dữ liệu, các thao tác này 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”.

3. Các bộ xử lý dữ liệu không thông báo “kết thúc thao tác” đến TM điều phối mà gửi đến các bộ quản lý khóa yêu cầu giải phóng khóa và thông báo cho TM điều phối.

4.6. Các thuật toán điều khiển đồng thời bằng nhãn thời gian

4.6.1. Đặt vấn đề

Khác với các thuật toán dựa vào khóa, các thuật toán điều khiển đồng thời bằng nhãn thời gian không duy trì tính khả tuần tự bằng phương pháp độc quyền truy cập, mà chọn một thứ tự và thực hiện các giao dịch theo thứ tự đó, bằng cách bộ quản lý giao dịch gán cho mỗi giao dịch T_i một nhãn thời gian (Timestemp) nhất $ts(T_i)$ vào lúc bắt đầu thực hiện giao dịch.

Mỗi giao dịch được gán cho duy nhất một mã thời gian nhằm phân biệt

giữa các giao dịch với nhau và sắp xếp các giao dịch theo thứ tự. Hai nhãn thời gian do một bộ quản lý giao dịch tạo ra phải tăng đơn điệu. Vì vậy, nhãn thời gian là những giá trị được lấy từ một miền có thứ tự toàn phần.

Có nhiều cách gán nhãn thời gian. Một phương pháp thường được sử dụng nhất là cho phép mỗi vị trí tự gán nhãn thời gian theo số đếm cục bộ. Để duy trì tính thống nhất, gán thêm định danh mỗi vị trí vào giá trị đếm. Vì vậy, nhãn thời gian là một bộ hai giá trị có dạng <giá trị đếm cục bộ, định danh vị trí>. Các hệ thống có thể truy cập đồng hồ hệ thống riêng của nó thì có thể sử dụng giá trị đồng hồ hệ thống thay cho giá trị đếm. Việc xếp thứ tự thực hiện các thao tác của các giao dịch theo nhãn thời gian rất đơn giản. Về hình thức, qui tắc xếp thứ tự thời gian (Timestamp Ordering- TO) có thể được mô tả như sau:

Quy tắc TO: Cho hai thao tác tương tranh O_{ij} và O_{kl} tương ứng các giao dịch T_i và T_k . O_{ij} được thực hiện trước O_{kl} hay gọi T_i là giao dịch già hơn và T_k được gọi là giao dịch trẻ hơn, khi và chỉ khi $ts(T_i) < ts(T_k)$.

Nếu thao tác mới thuộc về một giao dịch trẻ hơn so với các thao tác tương tranh của giao dịch đã được xếp lịch, thì thao tác này được chấp nhận. Ngược lại nó bị hủy bỏ, làm cho toàn bộ giao dịch phải khởi động lại với một nhãn thời gian mới.

Một bộ lập lịch theo nhãn thời gian hoạt động theo quy tắc TO sẽ bảo đảm tạo ra các lịch biểu khả tuần tự. Việc so sánh nhãn thời gian của các giao dịch chỉ được thực hiện nếu bộ lập lịch đã nhận được tất cả các thao tác cần lập lịch. Để có thể kiểm tra một thao tác đã đi ra khỏi dãy hay không, mỗi mục dữ liệu x được gán hai nhãn thời gian: nhãn thời gian đọc $[rts(x)]$, là nhãn thời gian lớn nhất trong số nhãn thời gian của các giao dịch đã đọc x và nhãn thời gian ghi $[wts(x)]$, là nhãn thời gian lớn nhất trong số nhãn thời gian của các giao dịch đã ghi x . Như vậy việc so sánh nhãn thời gian của một thao tác với nhãn thời gian đọc và nhãn thời gian ghi của mục dữ liệu nó đang cần truy cập có thể kiểm tra một giao dịch với nhãn thời gian lớn hơn đã truy cập đến mục dữ liệu hay chưa.

Bộ quản lý giao dịch có nhiệm vụ gán nhãn thời gian cho giao dịch mới và gán nhãn thời gian này cho thao tác cơ sở dữ liệu mà nó chuyển đến bộ lập lịch. Theo dõi các nhãn thời gian đọc và nhãn thời gian ghi và kiểm tra tính khả tuần tự.

4.6.2. Thuật toán bộ quản lý giao dịch TO cơ bản

Thuật toán bộ quản lý giao dịch TO cơ bản (Basic TO transaction

Manager – BTO – TM) điều khiển TM điều phối gán nhãn thời gian cho mỗi giao dịch, xác định vị trí lưu trữ dữ liệu và gửi các thao tác có liên quan đến những vị trí này.

Thuật toán TO cơ bản sẽ không gây bế tắc, vì các giao dịch không phải đợi trong khi chúng đang giữ các quyền truy cập trên các mục dữ liệu. Một giao dịch có một thao tác bị bộ lập lịch từ chối phải được khởi động lại bởi bộ phận quản lý giao dịch với một nhãn thời gian mới. Điều này đảm bảo giao dịch được thực hiện lần thử sau. Tuy nhiên cái giá phải trả cho việc không bị khóa gài là phải khởi động lại giao dịch nhiều lần. Khi một giao dịch đã được chấp nhận và được chuyển đến bộ xử lý dữ liệu, bộ lập lịch sẽ thực hiện việc trì hoãn bằng cách chỉ gửi thao tác được chấp nhận cho đến khi thao tác đầu được xử lý và được phản hồi. Điều này đảm bảo bộ xử lý dữ liệu thực hiện các thao tác theo đúng thứ tự mà bộ lập lịch gửi đến, nếu không, nhãn thời gian đọc và nhãn thời gian ghi cho mục dữ liệu được truy cập sẽ không chính xác.

Thuật toán BTO – TM

Declare-var: msg: Message, O: Dbop, Op: Operation,
x: DataItem, T: TransactionId, pm: Dpmsg,
res: Data val, S: SiteSet..

Begin

Repeat

WAIT(msg)

Case of msg

Dbop:

Begin

Op ← dop.opn

x ← dop.data

t ← dop.tid

Case of Op

Begin_Transaction:

Begin

S ← ∅

```

    End
  Read:
  Begin
     $S \leftarrow S \cup \{\text{trạm lưu } x \text{ và có chi phí truy cập đến } x \text{ là nhỏ nhất}\}$ 
    gửi O và ts (T) cho bộ lập lịch tại S
  End
  Write:
  Begin
     $S \leftarrow S \cup \{S_i | x \text{ được lưu tại } S_i\}$ 
    gửi O và ts (T) cho bộ lập lịch tại S
  End
  Abort or Commit:
  Begin
    gửi O đến các bộ lập lịch trong S
  end
end-case
Scmsg: {thao tác đã bị bộ lập lịch hủy bỏ}
begin
  Msg “hủy bỏ T”:
  Gửi msg đến các bộ lập lịch trong S
  Khởi động lại T
End
Dpmsg:
  Begin
     $op \leftarrow pm.opm$ 
     $Res \leftarrow pm.result$ 
     $T \leftarrow pm.tid$ :
  Case of Op
  Read:
  Begin

```

```

        Trả res về cho ứng dụng người dùng (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
        Commit then
        Begin
            thông tin ứng dụng biết về việc hoàn thành giao dịch
        End
    Abort:
    Begin
        thông tin ứng dụng cho biết về việc hủy bỏ giao dịch T
    End
    End-case
End
End-case
until forever
end. {BOT-TM}

```

Thuật toán bộ xếp lịch BTO-SC

```

Declare-var: msg: Messenge, dop: Dbop, Op: Operation,
            x: DataItem, T: TransactionId, SOP: OpSet.
Begin
    Repeat
        WAIT( msg)
    Case of msg
        Dbop:

```


Begin

$Op \leftarrow dop.opn$

$x \leftarrow dop.data$

$t \leftarrow dop.tid$

lưu các $rts(x)$ và $Wts(x)$ khởi đầu

Case of Op

Read:

Begin

If $ts(T) > rts(x)$ then

Begin

Gửi dop đến bộ xử lý dữ liệu

$Rts(x) \leftarrow ts(T)$

End

Else

Begin

$msg \leftarrow$ "bỏ T đi"

gửi msg cho TM điều phối

end

end-if

end {của trường hợp read}

Write:

Begin

If $ts(T) > rts(x)$ and $ts(T) > wts(x)$ then

Begin

Gửi dop đến bộ xử lý dữ liệu

$Rts(x) \leftarrow ts(T)$

$wts(x) \leftarrow ts(T)$

end

else

```

begin
    msg ← "bỏ T đi"
    gửi msg cho TM điều phối
end
end-if
end {của trường hợp write}
Commit:
Begin
    gửi dop đến các bộ xử lý dữ liệu
end {của trường hợp Commit}
Abort:
Begin
    For tất cả x đều được T truy cập do
    Đặt lại rst(x) và wrt(x) về các giá trị khởi đầu của chúng
    End-for
    Gửi dop đến bộ xử lý dữ liệu
End
End-case
End
End-case
until forever
end. {BTO-SC}

```

4.6.3. Thuật toán TO bảo toàn

Thuật toán TO cơ bản không yêu cầu các giao dịch phải đợi mà thay vào đó sẽ khởi động lại chúng. Đây là một ưu điểm của thuật toán, tránh được bế tắc nhưng phải khởi động lại giao dịch nhiều lần, làm ảnh hưởng đến hiệu năng của thuật toán. Thuật toán TO bảo toàn có gắng hạ thấp chi phí hệ thống bằng cách làm giảm số lần khởi động lại một giao dịch.

Biết rằng bộ lập lịch TO sẽ khởi động lại giao dịch nếu một giao dịch tương tranh trễ hơn đã được lập lịch hoặc đã được thực hiện. Giao dịch được khởi động lại sẽ tăng lên, nếu như một vị trí không hoạt động tương đối so với các vị trí khác và không đưa các giao dịch trong một thời hạn đã định. Số đếm

của giao dịch là một giá trị nhỏ hơn rất nhiều so với các số đếm tại những vị trí khác. Nếu TM tại vị trí nhận giao dịch, các thao tác được gửi đến các bộ lập lịch nằm tại vị trí khác chắc chắn bị từ chối, buộc giao dịch phải khởi động lại. Hơn nữa, chính giao dịch đó sẽ tái khởi động nhiều lần cho đến khi giá trị của số đếm tại vị trí ban đầu đạt được ngang mức với các số đếm của những vị trí khác.

Cần phải làm cho các số đếm tại các vị trí luôn đồng bộ. Việc đồng bộ hóa toàn bộ, đòi hỏi phải trao đổi các thông báo không cần thiết khi có một số đếm thay đổi. Vì vậy chi phí trao đổi cao. Khắc phục nhược điểm này bằng cách các bộ quản lý giao dịch từ xa có thể gửi các thao tác đến các bộ quản lý giao dịch trên những vị trí khác. So sánh giá trị số đếm của nó với giá trị của thao tác đến, tự động tăng thêm một. Điều này đảm bảo không có số đếm nào trong hệ thống chạy quá nhanh hoặc bị tụt lại phía sau quá nhiều. Sử dụng đồng bộ hệ thống thay cho số đếm thì việc đồng bộ hóa xấp xỉ này có thể được một cách tự động, với điều kiện các đồng hồ có tốc độ chạy gần như nhau.

Thuật toán TO cơ bản cố gắng cho thực hiện thao tác ngay sau khi nó được chấp nhận. Ngược lại thuật toán bảo toàn sẽ hoãn các thao tác lại cho đến khi nó bảo đảm rằng không có thao tác nào có nhãn thời gian nhỏ hơn đến bộ lập lịch đó. Nếu bảo đảm được điều kiện này, bộ lập lịch sẽ không bao giờ từ chối thao tác. Tuy nhiên, việc trì hoãn đó cũng gây nên khả năng bế tắc.

Kỹ thuật cơ bản trong thuật toán TO bảo toàn dựa trên ý tưởng sau đây: Các thao tác của mỗi giao dịch được đệm cho đến khi có được một thứ tự thực hiện để không xảy ra các loại trừ chúng được thực hiện theo thứ tự đó.

Giả sử mỗi bộ lập lịch duy trì một hàng đợi cho mỗi bộ quản lý giao dịch trong hệ thống. Bộ lập lịch tại vị trí i lưu trữ tất cả các thao tác mà nó nhận được từ bộ quản lý giao dịch tại vị trí j và hàng đợi Q_{ij} . Bộ lập lịch i có một hàng đợi như thế cho mỗi j . Khi nhận được một thao tác từ một bộ quản lý giao dịch, thao tác đó sẽ được đặt vào hàng đợi thích hợp của nó theo thứ tự nhãn thời gian tăng dần. Bộ lập lịch tại mỗi vị trí thực hiện các thao tác trong hàng đợi theo thứ tự nhãn thời gian tăng.

Cách tiếp cận này nhằm giảm đi số lần khởi động lại giao dịch, nhưng không đảm bảo loại bỏ hoàn toàn. Xét tình huống tại vị trí i , hàng đợi cho vị trí j (Q_{ij}) đang trống. Bộ lập lịch tại i sẽ chọn một thao tác, giả sử $R(x)$ có nhãn thời gian nhỏ nhất và chuyển nó cho bộ xử lý dữ liệu. Tuy nhiên vị trí j có thể đã gửi cho i một thao tác, giả sử là $W(x)$ có nhãn thời gian nhỏ hơn nhưng hiện còn

đang di chuyển trong mạng. Khi đến được i , nó bị từ chối vì vi phạm qui tắc TO: nó muốn truy cập một mục dữ liệu hiện đang được truy cập ở chế độ không tương thích bởi một thao tác khác có nhãn thời gian lớn hơn.

4.6.4. Thuật toán TO đa phiên bản

Thuật toán TO đa phiên bản nhằm loại bỏ hoàn toàn không khởi động giao dịch trong môi trường cơ sở dữ liệu tập trung. Trong thuật toán TO đa phiên bản, các phép cập nhật không sửa đổi cơ sở dữ liệu, mỗi thao tác ghi tạo ra một phiên bản mới của mục dữ liệu. Mỗi phiên bản được đánh dấu bằng nhãn thời gian của giao dịch tạo ra nó. Vì thế, thuật toán đa phiên bản phải mất nhiều không gian nhớ để lưu về thời gian. Khi xử lý giao dịch ở mỗi trạng thái của cơ sở dữ liệu được thực hiện tuần tự theo thứ tự nhãn thời gian.

Sự tồn tại các phiên bản hoàn toàn trong suốt đối với người sử dụng. Người sử dụng đưa ra yêu cầu bằng cách tham chiếu mục dữ liệu mà không chỉ ra phiên bản cụ thể nào. Bộ quản lý giao dịch sẽ gán một nhãn thời gian cho mỗi giao dịch, nó được dùng để theo dõi nhãn thời gian của mỗi phiên bản. Các thao tác được bộ lập lịch xử lý như sau:

Thao tác $R_i(x)$ được dịch thành một thao tác đọc trên một phiên bản của x , thực hiện bằng cách tìm một phiên bản của x , giả sử là x_v sao cho $ts(x_v)$ là nhãn thời gian lớn nhất trong số các nhãn thời gian nhỏ hơn $ts(T_i)$. $R_i(x_v)$ được gửi đến bộ xử lý dữ liệu.

1. Thao tác $W_i(x)$ được dịch thành $W_i(x_v)$ để $ts(x_v) = ts(T_i)$ và được gửi đến bộ xử lý dữ liệu khi và chỉ khi không có giao dịch nào khác có nhãn thời gian lớn hơn $ts(T_i)$ đã đọc giá trị của một phiên bản của x , giả sử x_r với $ts(x_r) > ts(x_s)$. Nói cách khác nếu bộ lập lịch đã xử lý một thao tác $R_j(x_r)$ sao cho $ts(T_i) < ts(x_r) < ts(T_j)$ thì thao tác $W_i(x)$ bị loại bỏ.

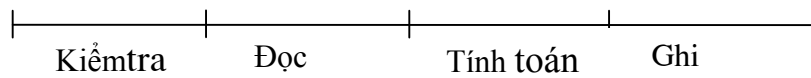
2. Các quy tắc trên sẽ bảo đảm cho một bộ lập lịch xử lý các yêu cầu đọc và ghi các giao dịch tạo ra bởi các lịch biểu khả tuần tự. Các phiên bản của cơ sở dữ liệu có thể được xóa bỏ để tiết kiệm không gian bộ nhớ, nếu hệ quản trị cơ sở dữ liệu không còn nhận một giao dịch nào có yêu cầu truy cập đến các phiên bản đã xóa.

4.7. Các thuật toán điều khiển đồng thời lạc quan

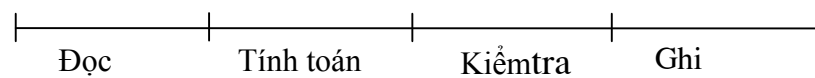
Các thuật toán điều khiển đồng thời bằng khóa chốt và bằng nhãn thời gian giả sử rằng tương tranh giữa các giao dịch thường xảy ra và không cho phép một giao dịch truy cập mục dữ liệu đó. Vì vậy, việc thực hiện một thao tác của giao dịch phải tuân thủ các quá trình theo thứ tự: kiểm tra, đọc, tính toán và

ghi. Thuật toán không xét đến các giao dịch cập nhật vì chúng gây ra các vấn đề về nhất quán. Các hành động chỉ đọc không có tính toán và ghi. Giả thiết ghi có cả hành động Commit.

Ngược lại, các giao dịch lạc quan (Optimistic) làm chậm quá trình kiểm tra cho đến trước quá trình ghi. Vì vậy, một thao tác được trao cho bộ lập lịch lạc quan không bao giờ trễ. Các thao tác đọc, tính toán và ghi được thực hiện tự do mà không có cập nhật vào cơ sở dữ liệu thực sự. Mỗi giao dịch lúc đầu đều cập nhật trên các bản sao cục bộ của mục dữ liệu và kiểm tra những cập nhật này có duy trì tính nhất quán của cơ sở dữ liệu hay không. Nếu nhất quán, thì các thay đổi được ghi vào cơ sở dữ liệu thực tế. Ngược lại giao dịch sẽ bị hủy bỏ và khởi động lại.



a) Kiểu bi quan



b) Kiểu lạc quan

Hình 4.7. Các pha thực hiện giao dịch

Thiết kế thuật toán điều khiển đồng thời lạc quan dựa nhãn thời gian, vì hai lý do:

Phần lớn các nghiên cứu về các phương pháp đồng thời lạc quan chỉ tập trung vào các hệ cơ sở dữ liệu tập trung.

Các thuật toán lạc quan chưa được cài đặt trên bất kỳ một hệ quản trị cơ sở dữ liệu thương mại hay hoặc thử nghiệm nào. Bản cài đặt duy nhất IMS-FASTPATH của hãng IBM cung cấp các hàm cơ sở truy cập đến cơ sở dữ liệu theo cách thức lạc quan tập trung.

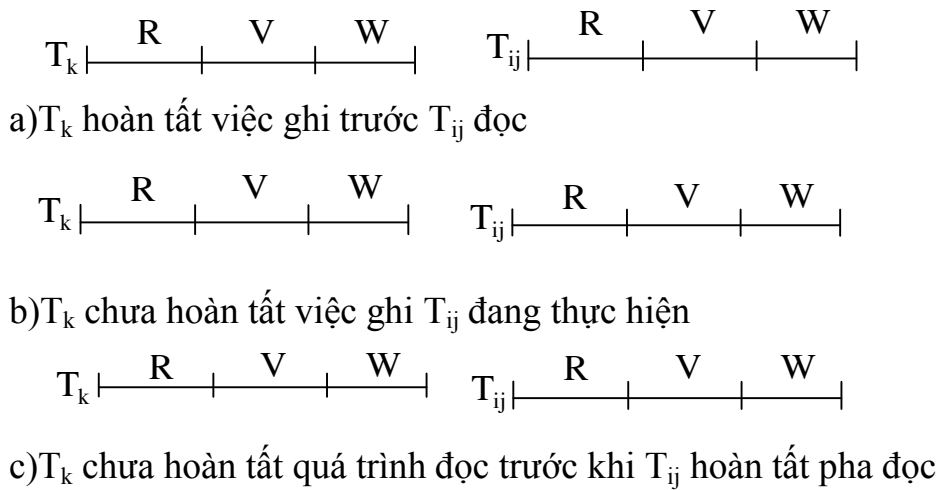
Thuật toán điều khiển đồng thời lạc quan được mở rộng cho các hệ quản trị cơ sở dữ liệu phân tán. Khác với thuật toán TO bi qua, không chỉ lạc quan mà còn gán nhãn thời gian. Nhãn thời gian chỉ được liên kết với các giao dịch, không có nhãn thời gian đọc và nhãn thời gian ghi. Hơn nữa, nhãn thời gian không được gán giá trị cho giao dịch vào lúc bắt đầu bước kiểm tra, vì nhãn thời gian chỉ cần cho bước kiểm tra và nếu gán nhãn sớm có thể gây ra loại bỏ giao dịch một cách không cần thiết.

Một giao dịch T_i được chia nhỏ bởi bộ quản lý giao dịch tại vị trí gốc thành một số giao dịch con, mỗi giao dịch con được thực hiện tại nhiều vị trí khác nhau. Ký hiệu T_{ij} là một giao dịch con của T_i thực hiện tại trạm j . Đến bước kiểm tra, thực hiện cục bộ là một dãy các thao tác được mô tả trong hình 4.10. Một nhãn thời gian được gán nhãn cho giao dịch và được sao lại cho tất cả các giao dịch con của nó. Kiểm tra cục bộ của T_{ij} được thực hiện theo các quy tắc sau đây:

Quy tắc 1: Với mọi giao dịch T_k cho $ts(T_k) < ts(T_{ij})$ đã hoàn tất bước ghi trước khi T_{ij} bắt đầu quá trình đọc (hình 4.8a), thì việc kiểm tra thành công các bước thực hiện giao dịch theo đúng thứ tự tuần tự.

Quy tắc 2: Nếu có một giao dịch T_k sao cho $ts(T_k) < ts(T_{ij})$ và T_k đã hoàn tất quá trình ghi trong khi T_{ij} đang trong bước đọc (hình 4.8b), thì kiểm tra thành công nếu $WS(T_k) \cap RS(T_{ij}) = \emptyset$.

Quy tắc 3: Nếu có một giao dịch T_k sao cho $ts(T_k) < ts(T_{ij})$ và T_k chưa hoàn tất quá trình đọc trước khi T_{ij} hoàn tất pha đọc (hình 4.8c), thì kiểm tra thành công nếu $WS(T_k) \cap RS(T_{ij}) = \emptyset$ và $WS(T_k) \cap WS(T_{ij}) = \emptyset$.



Hình 4.8. Các tình huống thực thi

Theo quy tắc 1, các giao dịch thực hiện tuần tự theo đúng dấu thời gian. Quy tắc 2 đảm bảo không có mục dữ liệu nào được cập nhật bởi T_k lại được T_{ij} đọc và T_k kết thúc việc ghi các cập nhật của nó vào cơ sở dữ liệu trước khi T_{ij} bắt đầu ghi. Vì thế các cập nhật của T_{ij} sẽ không bị ghi chồng lên bởi các cập nhật của T_k . Quy tắc 3 tương tự như quy tắc 2 nhưng không đòi hỏi T_k kết thúc việc trước khi T_{ij} bắt đầu chỉ ghi, chỉ yêu cầu các cập nhật T_k không ảnh hưởng

đến ghi hoặc đọc của T_{ij} .

Khi giao dịch đã được kiểm tra cục bộ để đảm bảo tính nhất quán cục bộ, thì yêu cầu cần phải được kiểm tra toàn cục để đảm bảo rằng tính nhất quán toàn cục. Đáng tiếc hiện nay chưa có phương pháp nào để kiểm tra việc này. Người ta sử dụng phương pháp nếu một giao dịch được kiểm tra toàn cục khi tất cả các giao dịch đi trước nó theo thứ tự tuần tự hóa cục bộ đều kết thúc. Đây là phương pháp bi quan vì nó thực hiện kiểm tra toàn cục sớm hơn và làm trễ một giao dịch. Tuy nhiên nó đảm bảo các giao dịch thực hiện theo cùng một thứ tự mỗi trạm.

Các thuật toán điều khiển đồng thời lạc quan cho phép mức độ hoạt động đồng thời cao. Trường hợp các tương tranh giao dịch ít xảy ra, cơ chế lạc quan thực hiện hiệu quả hơn là cơ chế khóa. Nhược điểm chủ yếu của các thuật toán lạc quan là chi phí lưu trữ cao. Để kiểm tra một giao dịch, thuật toán phải lưu các tập đọc và tập ghi của nhiều giao dịch đã kết thúc. Đặc biệt là các tập đọc tập ghi của các giao dịch đã kết thúc nhưng đã đang tiến hành, khi giao dịch T_{ij} đi đến vị trí j cũng phải được lưu trữ lại để kiểm tra T_{ij} .

4.8. Quản lý bế tắc

Phần lớn các thuật toán điều khiển đồng thời theo khóa và bằng nhãn thời gian đều có thể dẫn đến bế tắc (Deadlock) trong giao dịch. Các thuật toán điều khiển đồng thời theo khóa độc quyền truy cập đến các dữ liệu và các giao dịch có thể phải đợi để nhận được khóa. Một số thuật toán TO đòi hỏi giao dịch phải chờ đợi, như TO nghiêm khi ngặt. Vì vậy, hệ quản trị cơ sở dữ liệu phân tán cần phải có những cơ chế để quản lý sự bế tắc. Một bế tắc có thể xảy ra khi các giao dịch phải chờ đợi một giao dịch khác. Một cách không hình thức, một tình huống bế tắc là một tập các yêu cầu không bao giờ được đáp ứng từ cơ chế đồng thời.

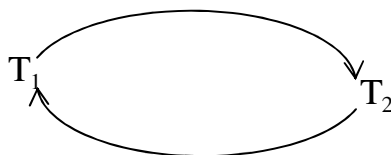
Ví dụ 4.13: Xét giao dịch T_i và T_j đang giữ các khóa ghi trên hai thực thể x và y nghĩa là $wli(x)$ và $wlj(x)$. Giả sử T_i cần $rli(y)$ hoặc $wli(y)$. Vì y đang bị khóa bởi giao dịch T_j , T_i sẽ phải đợi cho đến khi T_j giải phóng khóa trên y . Tuy nhiên, trong khi chờ đợi T_j lại yêu cầu một khóa (đọc hoặc ghi) trên x có thể xảy ra bế tắc. Vì T_i sẽ bị phong tỏa và chờ đợi T_j giải phóng ngay trên y , trong khi đó T_i cũng sẽ đợi T_j giải phóng khóa trên x . Như vậy T_i và T_j sẽ cùng phải chờ đợi vô hạn để giao dịch kia giải phóng khóa tương ứng.

Nếu bế tắc đã tồn tại trong hệ thống, thì nó sẽ biến mất cho đến khi có sự can thiệp của bên ngoài. Sự can thiệp có thể do người sử dụng, quản trị hệ thống, hoặc hệ điều hành, hoặc hệ quản trị cơ sở dữ liệu phân tán.

Sự bế tắc có thể phân tích bằng một đồ thị nhờ WFG (Wait For Graph) có

hướng được biểu diễn mối liên hệ chờ đợi giữa các giao dịch. Các nút là các giao dịch đang xảy ra đồng thời trong hệ thống. Một cung $T_i T_j$ tồn tại trong WFG nếu giao dịch T_i đang đợi T_j giải phóng khóa trên một thực thể nào đó.

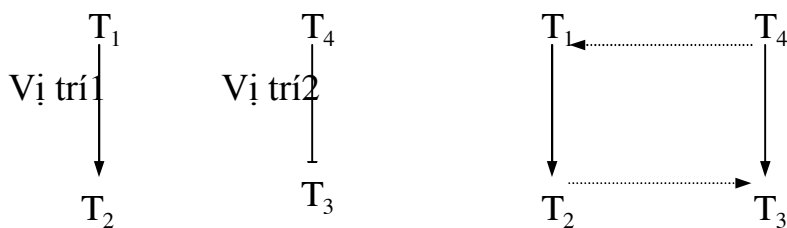
- Giao dịch T_i đang đợi giao dịch T_j giải phóng khóa trên một thực thể y .
- Giao dịch T_j đang đợi giao dịch T_i giải phóng khóa trên một thực thể x .



Hình 4.9. Một ví dụ về đồ thị chờ

Như vậy một bế tắc chỉ xảy ra khi trong WFG tồn tại một chu trình. Trong môi trường phân tán, đồ thị WFG sẽ phức tạp hơn bởi vì hai giao dịch có mặt trong bế tắc có thể đang chạy ở những vị trí khác nhau, gọi là bế tắc toàn cục. Do đó, trong hệ thống phân tán cần phải xây dựng một đồ thị chờ toàn cục GWFG (Global WFG), là hợp tất cả các đồ thị chờ cục bộ LWFG (Local WFG).

Ví dụ 4.14: Xét bốn giao dịch T_1, T_2, T_3 và T_4 với các mối liên hệ chờ được biểu thị như sau: $T_1 T_2 T_3 T_4 T_1$. Nếu T_1 và T_2 trên vị trí 1, T_3 và T_4 trên vị trí 2. Với các LWFG tại hai vị trí được trình bày trong hình 4.10a, không thể phát hiện được bế tắc. Bế tắc toàn cục có thể phát hiện được bằng đồ thị toàn cục LWFG được trình bày trong hình 4.10b. Các cung chờ đợi giữa các vị trí được vẽ bằng nét đứt.



a) Các đồ thị cục bộ LWFG; b) Đồ thị toàn cục GWF.

Hình 4.10. Một ví dụ về đồ thị LWFG và GWFG

Có ba phương pháp xử lý bế tắc: ngăn chặn, tránh, phát hiện và giải tỏa bế tắc

4.8.1. Ngăn chặn bế tắc

Ngăn chặn bế tắc là phương pháp bảo đảm không để xảy ra bế tắc. Bộ quản lý giao dịch phải kiểm tra giao dịch ngay khi nó bắt đầu và không cho phép nó thực hiện nếu có khả năng gây ra bế tắc. Vì vậy phải khai báo các mục sẽ

được truy cập của giao dịch và được lưu trữ tại bộ quản lý giao dịch. Nếu tất cả các mục dữ liệu cần truy cập sẵn có, bộ quản lý giao dịch cho phép giao dịch tiến hành. Ngược lại giao dịch sẽ không được phép tiến hành.

Nhằm đảm bảo an toàn việc truy cập các mục dữ liệu, hệ thống cần phải có một tập cực đại các mục, ngay cả khi kết thúc giao dịch mà vẫn không cần truy cập đến chúng. Điều này sẽ làm giảm các hoạt động đồng thời và cũng phải mất thêm các chi phí đánh giá một giao dịch tiến hành an toàn hay không.

4.8.2. Tránh bế tắc

Phương pháp tránh bế tắc (Deadlock Avoidance) sử dụng kỹ thuật điều khiển đồng thời không gây bế tắc hoặc yêu cầu bộ lập lịch phát hiện được các tình huống bế tắc và ngăn chặn không cho xảy ra được.

Phương pháp đơn giản nhất để tránh bế tắc là xếp thứ tự các tài nguyên, mỗi tiến trình truy cập đến các tài nguyên theo thứ tự đó. Giải pháp này đã được đề xuất cho các hệ điều hành. Các đơn vị khóa trong các cơ sở dữ liệu phân tán được sắp xếp theo thứ tự và các giao dịch yêu cầu khóa theo thứ tự đó. Việc xếp thứ tự các đơn vị khóa có thể cũng cần phải xếp thứ tự và yêu cầu các giao dịch truy cập tại nhiều vị trí nhận khóa bằng cách duyệt qua các vị trí theo một thứ tự định trước.

Một phương pháp khác sử dụng nhãn thời gian để đặt quyền ưu tiên cho các giao dịch và giải quyết bế tắc bằng cách hủy bỏ giao dịch có quyền ưu tiên cao hơn hoặc thấp hơn. Bộ quản lý khóa cần được sửa đổi theo hướng, nếu cần khóa của một giao dịch T_i bị từ chối, thì bộ quản lý khóa không tự động buộc T_i phải chờ đợi mà nó thực hiện phép kiểm tra dự phòng cho giao dịch đang yêu cầu và giao dịch hiện đang giữ khóa (giả sử T_j). Nếu việc kiểm tra tốt thì T_i được phép chờ T_j . Ngược lại một trong hai giao dịch phải bị hủy bỏ.

Các thuật toán WAIT-DIE và WOUND-DIE được thiết kế dựa trên việc gán nhãn thời gian cho giao dịch. WAIT-DIE là thuật toán không tước quyền (Nopreempty) nếu yêu cầu khóa của T_i từ chối do khóa đang được T_j giữ, thì nó không bao giờ tước quyền T_j . Quy tắc WAIT-DIE như sau: Nếu T_i yêu cầu khóa trên mục dữ liệu đã được khóa bởi T_j thì T_i được phép chờ khi và chỉ khi T_i già hơn T_j . Nếu T_i trẻ hơn T_j thì T_i bị hủy bỏ và được khởi động lại với nhãn thời gian cũ.

Thuật toán WOND-DIE là thuật toán tước quyền (Preempty) được khẳng định theo quy tắc sau: Nếu T_i yêu cầu khóa trên mục dữ liệu T_j khóa thì T_i được phép chờ khi và chỉ khi nó trẻ hơn T_j . Ngược lại T_j bị hủy và khóa được trao cho T_i .

Như vậy có các trường hợp: T_i chờ, T_i bị hủy bỏ và T_i làm cho T_j bị hủy

bỏ. Kết quả hủy bỏ và làm giao dịch khác bị hủy bỏ là như nhau, giao dịch bị hủy bỏ sẽ được khởi động lại. Hai quy tắc trên được mô tả như sau:

- If $ts(T_i) < ts(T_j)$ then T_i chờ else T_i bị hủy bỏ (WAIT-DIE)
- If $ts(T_i) < ts(T_j)$ then T_j bị hủy bỏ else T_i chờ (WOUND-DIE)

Thuật toán WAIT-DIE ưu tiên các giao dịch trẻ và không ưu tiên các giao dịch già. Một giao dịch già phải đợi lâu hơn khi nó trở nên già hơn. Ngược lại quy tắc WOUND-DIE ưu tiên các giao dịch già và nó không bao giờ đợi một giao dịch trẻ. Trong cả hai thuật toán, giao dịch trẻ hơn bị hủy bỏ và chúng có sự tước quyền các giao dịch đang hoạt động hay không.

Phương pháp tránh bế tắc thích hợp hơn các phương pháp ngăn chặn bế tắc trong các môi trường cơ sở dữ liệu. Nhược điểm cơ bản là chúng đòi hỏi phải hỗ trợ lúc chạy để quản lý bế tắc, làm tăng thêm chi phí lúc chạy để thực hiện giao dịch.

4.8.3. Phát hiện và giải tỏa bế tắc

Phát hiện và giải tỏa bế tắc (Deadlock Detection And Resolution) là một phương pháp quản lý bế tắc phổ biến nhất. Việc phát hiện bế tắc được thực hiện bằng cách phân tích GWFG xem có hình thành chu trình hay không. Giải tỏa bế tắc thông thường được thực hiện bằng cách chọn ra một hoặc nhiều giao dịch tước quyền rồi hủy bỏ chúng nhằm phá vỡ các chu trình trong đồ thị. Với giả thiết là chi phí tước quyền mỗi phần tử của một tập giao dịch bị bế tắc đã được biết. Bài toán cho tổng chi phí nhỏ nhất để phá vỡ chu trình bế tắc là NP đầy đủ.

Có ba phương pháp phát hiện bế tắc phân tán là phát hiện bế tắc tập trung, phân tán và phân cấp.

a) Phát hiện bế tắc tập trung

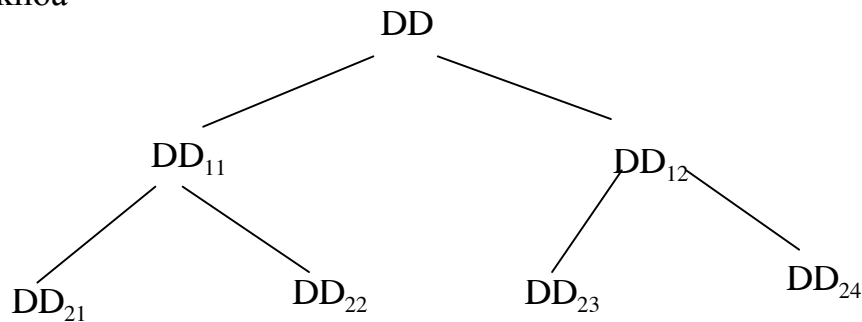
Một vị trí được chọn là bộ kiểm tra (Detector) cho toàn bộ hệ thống. Theo định kỳ, mỗi bộ quản lý khóa gửi LWFG cho bộ kiểm tra và nó sẽ tạo ra một đồ thị GWFG rồi tìm kiếm chu trình trong đó. Bộ quản lý khóa chỉ cần gửi những thay đổi trong đồ thị của nó, nghĩa là các cạnh mới thành lập hoặc mới xóa cho bộ kiểm tra. Thời gian càng nhỏ thì càng dễ phát hiện bế tắc nhưng chi phí truyền thông càng lớn. Phát hiện bế tắc tập trung đã được đề xuất cho hệ INGRER phân tán. Phương pháp này đơn giản và là một lựa chọn nếu thuật toán điều khiển đồng thời 2 PL tập trung.

b) Phát hiện bế tắc phân cấp

Xây dựng một cây phân cấp các bộ kiểm tra bế tắc. Qua đồ thị cục bộ

LWFG có thể phát hiện bế tắc tại vị trí cục bộ. Các vị trí gửi đồ thị của nó, nghĩa là các cạnh mới thành lập hoặc mới xóa cho bộ kiểm tra ở mức kế tiếp. Vì vậy bế tắc phân tán của hai hay nhiều vị trí có thể được phát hiện bởi bộ kiểm tra ở mức thấp hơn kế tiếp có quyền điều khiển trên vị trí đó. Ví dụ, bế tắc ở vị trí 1, sẽ được phát hiện bởi bộ kiểm tra bế tắc (DD) cục bộ tại vị trí 1, ký hiệu là DD_{21} (trong đó 2 là cấp 1 là vị trí). DD_{11} sẽ phát hiện bế tắc có liên quan đến vị trí 1 và vị trí 2. Nếu bế tắc có liên quan đến 4 vị trí, thì DD_{0x} phát hiện nó, trong đó x là một trong các vị trí 1, 2, 3 hoặc 4.

Phương pháp phát hiện bế tắc phân cấp không phụ thuộc vào vị trí trung tâm, vì vậy cũng làm giảm chi phí truyền thông. Nhược điểm của phương pháp này là khi cài đặt sẽ phức tạp hơn và phải sửa đổi các thuật toán quản lý giao dịch và khóa



Hình 4.11. Phát hiện bế tắc phân cấp

c) Phát hiện bế tắc phân tán:

Cài đặt cho các vị trí phát hiện bế tắc. Vì vậy, cũng như trong phương pháp phát hiện phân cấp, các bộ kiểm tra cục bộ tại các vị trí trao đổi các đồ thị cục bộ có các chu trình bế tắc với nhau. Phương pháp phát hiện bế tắc phân tán có thể khái quát như sau:

Đồ thị cục bộ LWFG được tạo ra và được sửa đổi theo nguyên tắc:

1. Vì các vị trí đều nhận được các chu trình tắc tiềm ẩn từ những vị trí khác nên các cạnh này được thêm vào các đồ thị cục bộ.
2. Các cạnh trong đồ thị cục bộ cho biết các giao dịch đang chờ đợi giao dịch tại những vị trí khác sẽ được nối với các cạnh trong các đồ thị cục bộ biểu thị các giao dịch ở xa đang đợi các giao dịch cục bộ.

Danh mục tài liệu tham khảo

1. Lê Tiến Vương (1996). *Nhập môn cơ sở dữ liệu quan hệ*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
2. Nguyễn Bá Tường (2005). *Nhập môn cơ sở dữ liệu phân tán*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
3. Phạm Thế Quế (2004). *Giáo trình Cơ sở dữ liệu: Lý thuyết và thực hành*. Nhà xuất bản Bưu điện Hà Nội.
4. Phạm Thế Quế (2010). *Giáo trình Cơ sở dữ liệu phân tán*. Nhà xuất bản Thông tin và Truyền thông, Hà Nội.
5. M.Tamer Ozsü And Patrice Valduriez (1999). *Principles of Distributed Database Systems*, Second Edition. Prentice Hall Upper Saddle River.

Câu hỏi và bài tập cuối chương

1. Trình bày trạng thái nhất quán của một cơ sở dữ liệu.
2. Trình bày độ tin cậy.
3. Trình bày quản lý giao dịch.
4. Trình bày khái niệm giao dịch.
5. Trình bày điều kiện kết thúc giao dịch.
6. Trình bày đặc tính của giao dịch.
7. Trình bày tính chất nguyên tố của giao dịch.
8. Trình bày tính nhất quán của giao dịch.
9. Trình bày tính cô lập của giao dịch.
10. Trình bày tính bền vững của giao dịch.
11. Trình bày các loại giao dịch theo thời gian hoạt động.
12. Trình bày các loại giao dịch dựa trên việc tổ chức các hành động đọc và ghi.
13. Trình bày luồng công việc – WorkFlows.
14. Trình bày tính khả tuần tự lịch biểu.
15. Phân loại các cơ chế điều khiển đồng thời.
16. Trình bày thuật toán lý khóa cơ bản (Basic Lock Manager).
17. Trình bày thuật toán khóa chốt 2 pha PL (Two Phase Locking).
18. Trình bày thuật toán quản lý giao dịch 2 PL tập trung (C2PL TM).
19. Trình bày thuật toán 2PL bản chính.
20. Trình bày thuật toán 2PL phân tán.
21. Trình bày thuật toán bộ quản lý giao dịch TO cơ bản – BTO TM.
22. Trình bày thuật toán TO bảo toàn.
23. Trình bày thuật toán TO đa phiên bản.
24. Trình bày ngăn chặn bế tắc.
25. Trình bày tránh bế tắc.
26. Trình bày phát hiện và giải tỏa bế tắc.

BÀI TẬP

1. Lịch biểu nào sau đây là tương đương tương tranh (bỏ qua các lệnh ủy thác C và hủy bỏ A).

$$S_1 = W_2(x), W_1(x), R_3(x), R_1(x), C_1, W_2(y), R_3(y), R_3(z), C_3, R_2(z), C_2.$$

$$S_2 = R_3(z), R_3(y), W_2(y), R_2(z), W_1(x), W_2(x), W_1(x), C_1, C_3.$$

$$S_3 = R_3(z), W_2(x), W_2(y), R_1(x), R_3(x), R_3(y), C_3, C_2, C_1.$$

$$S_4 = R_2(z), W_2(x), W_2(y), C_2, W_1(x), R_1(x), A_1, R_3(x), R_3(z), R_3(y), C_3.$$

2. Lịch biểu nào sau đây là không tuần tự?

$$S_1 = W_2(x), W_1(x), R_3(x), R_1(x), C_1, W_2(y), R_3(y), R_3(z), C_3, R_2(z), C_2.$$

$$S_2 = R_3(z), R_3(y), W_2(y), R_2(z), W_1(x), W_2(x), W_1(x), C_1, C_3.$$

$$S_3 = R_3(z), W_2(x), W_2(y), R_1(x), R_3(x), R_3(y), C_3, C_2, C_1.$$

$$S_4 = R_2(z), W_2(x), W_2(y), C_2, W_1(x), R_1(x), A_1, R_3(x), R_3(z), R_3(y), C_3.$$

3. Nói rằng lịch biểu S là khôi phục được, nếu mỗi giao dịch T_i đọc một mục x từ giao dịch T_j ($i \neq j$) trong S và C_i xảy ra trong S thì $C_j \alpha_S C_i$. T_i đọc x từ T_j trong S nếu:

a. $W_j(x) \alpha_S R_i(x)$

b. $A_j \text{ not } \alpha_S R_i(x)$

c. Nếu có một hành động $W_k(x)$ sao cho $W_j(x) \alpha_S W_k(x) \alpha_S R_j(x)$ khi đó

$$A_k \alpha_S R(x).$$

Lịch biểu nào sau đây là khôi phục được

$$S_1 = W_2(x), W_1(x), R_3(x), R_1(x), C_1, W_2(y), R_3(y), R_3(z), C_3, R_2(z), C_2.$$

$$S_2 = R_3(z), R_3(y), W_2(y), R_2(z), W_1(x), W_2(x), W_1(x), C_1, C_3.$$

$$S_3 = R_3(z), W_2(x), W_2(y), R_1(x), R_3(x), R_3(y), C_3, C_2, C_1.$$

$$S_4 = R_2(z), W_2(x), W_2(y), C_2, W_1(x), R_1(x), A_1, R_3(x), R_3(z), R_3(y), C_3.$$

4. Xây dựng thuật toán cho bộ quản lý giao dịch và bộ quản lý khóa chốt hai pha phân quyền.
5. Sửa lại thuật toán 2PL tập quyền để giải quyết ảnh ảo.

6. Giải thích mối quan hệ giữa yêu cầu lưu trữ bộ quản lý giao dịch và kích thước giao dịch (số thao tác của giao dịch) cho một bộ quản lý giao dịch bằng cách sử dụng một sắp xếp nhãn thời gian lạc quan để điều khiển đồng thời.

7. Xét các mục dữ liệu x và y được nhân bản trên các vị trí như sau:

Vị trí 1	Vị trí 2	Vị trí 3	Vị trí 4
x	x		x
	y	y	y

- Gán các biểu quyết cho mỗi vị trí và cho biết lượng biểu quyết theo quy định đọc và ghi.
- Xác định các tình huống có thể gây ra một phân hoạch mạng và với mỗi tình huống hãy mô tả xem trong nhóm vị trí nào một giao dịch cập nhật x (đọc và ghi) có thể kết thúc được và tình huống kết thúc là gì?

Chương 5

CÁC HỆ CƠ SỞ DỮ LIỆU SONG SONG

Chương này trình bày một số khái niệm cơ bản về hệ cơ sở dữ liệu song song, một cách khái quát về mô hình kiến trúc của hệ cơ sở dữ liệu song song, các kỹ thuật hệ quản trị cơ sở dữ liệu song song, các thuật toán được sử dụng khi thực hiện xử lý dữ liệu song song.

- Các chức năng hệ xử lý song song;
- Kiến trúc hệ cơ sở dữ liệu song song;
- Các kỹ thuật hệ quản trị cơ sở dữ liệu song song.

5.1. Mục tiêu của xử lý song song

Xử lý song song sử dụng các máy tính đa bộ xử lý để chạy các chương trình ứng dụng bằng cách dùng nhiều bộ xử lý để cải thiện hiệu năng. Các hệ cơ sở dữ liệu song song tổ hợp khả năng quản lý cơ sở dữ liệu và khả năng xử lý song song để làm tăng hiệu năng và độ khả dụng. Cơ chế song song đã giải quyết vấn đề tắc nghẽn tại các node khi thực hiện xuất nhập (I/O), vì thời gian truy xuất cao hơn nhiều so với thời gian truy xuất bộ nhớ chính. Ví dụ, một cơ sở dữ liệu có kích thước D được lưu trữ trên một thiết bị nhớ có lưu lượng là T . Như vậy lưu lượng của hệ thống sẽ bị chặn bởi T . Ngược lại nếu phân mảnh cơ sở dữ liệu và cấp phát trên n vị trí, mỗi vị trí chứa D/n và lưu lượng T , như vậy sẽ có lưu lượng $n \cdot T$.

5.2. Ưu điểm của các cơ sở dữ liệu song song

Các hệ thống dữ liệu song song kết hợp với việc quản trị dữ liệu và xử lý song song làm tăng hiệu năng và tính sẵn sàng của hệ thống. Hiệu năng là mục tiêu của các máy cơ sở dữ liệu (Database Machine) với hệ quản trị cơ sở dữ liệu truyền thống thường xảy ra ùn tắc vào/ra do thời gian truy cập bộ nhớ phụ cao hơn so với thời gian truy cập bộ nhớ chính. Việc phân vùng cơ sở dữ liệu trên nhiều đĩa sẽ đạt được khả năng song song của các liên truy vấn và nội truy vấn (inter and intran Query), cải thiện một cách đáng kể về thời gian đáp ứng và thông lượng các giao tác.

Một hệ cơ sở dữ liệu song song có thể định nghĩa đơn giản như một hệ quản trị cơ sở dữ liệu được cài đặt trên bộ đa xử lý kết chặt, bao gồm từ kết nối hệ quản trị cơ sở dữ liệu hiện có với yêu cầu ghi lại các thủ tục giao diện hệ điều

hành đến sự kết hợp phức tạp giữa xử lý song song và các chức năng hệ thống cơ sở dữ liệu thành kiến trúc phần cứng/mềm mới.

Hệ thống cơ sở dữ liệu song song hoạt động như một Database Server cho Application Server trong mô hình Client/Server của mạng tính. Hệ thống cơ sở dữ liệu song song hỗ trợ các chức năng cơ sở dữ liệu và giao diện Client/Server và có thể chức năng đa năng. Để hạn chế trao đổi thông tin giữa Client và Server. Cần thiết có một giao diện ở mức cao khuyến khích xử lý dữ liệu trên máy chủ.

Tỷ lệ giá thành/hiệu năng của các hệ cơ sở dữ liệu song song tốt hơn so với máy tính lớn tương ứng, cũng là những ưu điểm của các hệ cơ sở dữ liệu phân tán.

5.2.1. Hiệu năng cao

Hiệu năng cao (High Performance) có thể đạt được bằng các giải pháp hỗ trợ hệ điều hành nhận biết các yêu cầu của cơ sở dữ liệu, khả năng song song, tối ưu hóa và cân bằng tải. Hệ điều hành nhận biết các yêu cầu cụ thể của một cơ sở dữ liệu sẽ làm đơn giản hóa khả năng cài đặt các chức năng cơ sở dữ liệu ở mức thấp, vì vậy sẽ làm giảm chi phí. Cơ chế hoạt động song song có thể làm tăng lưu lượng bằng việc sử dụng khả năng song song liên truy vấn, giảm thời gian đáp ứng các giao tác bằng việc sử dụng khả năng song song của các nội truy vấn. Tuy nhiên, việc làm giảm thời gian đáp ứng các truy vấn phức tạp qua cơ chế song song quy mô lớn cũng có thể sẽ tăng tổng thời gian và làm ảnh hưởng đến lưu lượng. Do đó việc tối ưu và so sánh các truy vấn nhằm giảm tối thiểu các khả năng song song, ví dụ như việc ràng buộc tính song song của các câu truy vấn. Cân bằng tải là khả năng hệ thống chia khối lượng công việc bằng nhau giữa tất cả các bộ xử lý. Nó phụ thuộc vào kiến trúc bộ xử lý, nó có thể được lưu giữ bởi cơ sở dữ liệu tĩnh hoặc động.

5.2.2. Tính sẵn sàng

Hệ cơ sở dữ liệu song song có nhiều thành phần tương tự nhau. Khả năng nhân bản dữ liệu trên các vị trí của mạng làm tăng tính sẵn sàng của cơ sở dữ liệu khi truy cập dữ liệu. Trong hệ thống song song, các mảnh dữ liệu được cài đặt trên các thiết bị lưu trữ tại các node của mạng, xác suất đĩa hỏng ở bất cứ thời điểm nào cũng có thể xảy ra. Tuy nhiên các sự cố đĩa hỏng không làm mất cân bằng giải pháp phân vùng nhân bản trên các vị trí, có thể truy cập song song.

5.2.3. Khả năng mở rộng

Trong môi trường song song, khả năng mở rộng hệ thống dễ dàng hơn. Có thể dễ dàng tăng kích thước cơ sở dữ liệu hoặc có thể tăng thông lượng bằng thông. Tính mở rộng hệ thống là mở rộng khả năng xử lý và lưu trữ cho hệ thống. Hay nói cụ thể hơn khả năng mở rộng tuyến tính và tốc độ tuyến tính. Mở rộng tuyến tính là hiệu năng vẫn được duy trì khi tăng tuyến tính kích thước cơ sở dữ liệu và khả năng xử lý và lưu trữ. Tăng tốc độ tuyến tính nghĩa là nói đến sự tăng tuyến tính về hiệu năng trên một cơ sở dữ liệu không đổi về kích thước và tăng tuyến tính về khả năng xử lý và lưu trữ. Việc mở rộng hệ thống đòi hỏi phải tổ chức lại ít nhất trên cơ sở dữ liệu đã có.

5.3.Kiến trúc hệ cơ sở dữ liệu song song

5.3.1. Bộ quản lý phiên

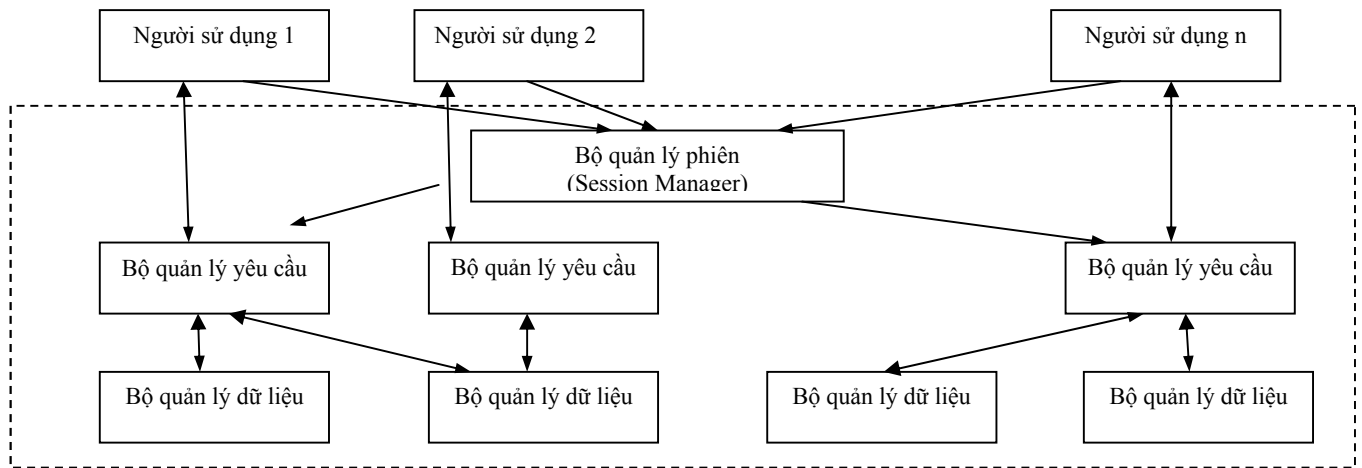
Bộ quản lý phiên giám sát giao dịch, hỗ trợ các giao dịch giữa Client với Server. Thực hiện kết nối và giải phóng kết nối các tiến trình giữa các Client và giữa hai hệ thống bộ quản lý yêu cầu và bộ quản lý dữ liệu. Vì thế nó khởi tạo và đóng các phiên giao dịch người sử dụng có nhiều giao thức.

5.3.2. Bộ quản lý yêu cầu

Bộ quản lý yêu cầu tiếp nhận các yêu cầu từ phía Client có liên quan đến việc biên dịch và thực thi truy vấn. Nó có thể truy xuất vào thư mục cơ sở dữ liệu chứa các thông tin về cấu trúc dữ liệu và chương trình. Phụ thuộc vào từng yêu cầu mà bộ xử lý này sẽ thực hiện nhiều pha biên dịch khác nhau, kích hoạt cho việc thực thi các câu truy vấn, trả kết quả cũng như mã lỗi về cho ứng dụng Client. Vì bộ quản lý yêu cầu giám sát việc thực thi và xác nhận các giao dịch nên nó có thể kích hoạt thủ tục khôi phục trong trường hợp sự cố giao dịch bị lỗi. Để tăng tốc độ biên dịch câu truy vấn nó có thể tối ưu hóa và song song hóa câu truy vấn vào lúc biên dịch.

5.3.3. Bộ quản lý dữ liệu

Bộ quản lý dữ liệu cung cấp tất cả các chức năng cần thiết để chạy song song các câu truy vấn đã biên dịch. Nghĩa là, thực hiện các toán tử cơ sở dữ liệu, hỗ trợ giao dịch song song, quản lý bộ nhớ Cache...Nếu bộ quản lý yêu cầu có khả năng biên dịch các lệnh điều khiển dòng dữ liệu thì việc đồng bộ hóa giao tiếp giữa các bộ quản lý dữ liệu có thể thực hiện. Nếu không,việc điều khiển giao dịch và đồng bộ hóa phải được thực hiện bằng module quản lý yêu cầu.



Hình 5.1. Kiến trúc tổng quát của một hệ cơ sở dữ liệu song song

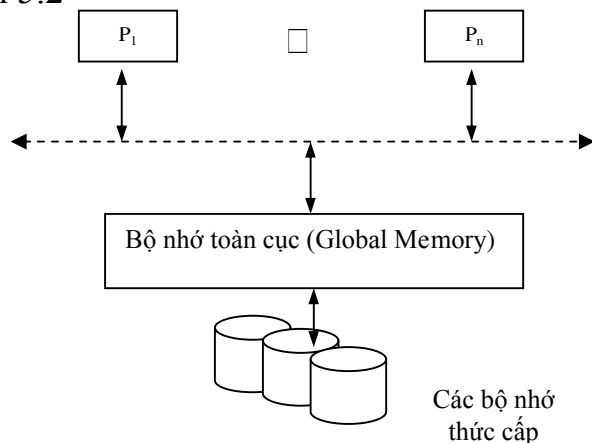
5.4. Các kiến trúc hệ thống song song

5.4.1. Tổng quan về kiến trúc song song và hệ thống song song

Một hệ thống song song là một hệ thống bao gồm nhiều bộ vi xử lý được kết nối với nhau và kết nối với các thành phần cứng khác. Như vậy, mỗi hệ thống song song là kết quả kết hợp các cách thiết kế khác nhau nhằm cung cấp những ưu điểm về hiệu năng, tính khả dụng và tính mở rộng với tỷ lệ chi phí hiệu năng tốt hơn so với một máy tính lớn tương đương. Các thành phần phần cứng liên kết với nhau bằng môi trường truyền vật lý. Có hai loại kiến trúc song song cơ bản là: kiến trúc chia sẻ bộ nhớ và kiến trúc không chia sẻ bộ nhớ, ví dụ như kiến trúc NUMA (Non Uniform Memory Access) truy cập bộ nhớ không đồng dạng.

5.4.2. Kiến trúc chia sẻ bộ nhớ

Kiến trúc chia sẻ bộ nhớ (Share Memory) là kiến trúc đơn giản của hệ thống song song. Trong kiến trúc này, mỗi bộ xử lý truy cập đến một modul nhớ hoặc một đơn vị đĩa bất kỳ qua một liên kết tốc độ cao, chẳng hạn như bus tốc độ nhanh hoặc chuyển mạch chữ nhật. Mô hình của kiến trúc chia sẻ bộ nhớ được thể hiện trong hình 5.2



Hình 5.2. Kiến trúc chia sẻ bộ nhớ

Một trong những mô hình thông dụng nhất của chia sẻ bộ nhớ là mô hình máy song song truy cập ngẫu nhiên. Mô hình này có thể được mở rộng tùy theo số lượng bộ vi xử lý, được đồng bộ hóa ở mức xử lý lệnh. Vì vậy, mỗi bộ vi xử lý sẽ thực hiện những tập lệnh của bản thân nó và sẽ được hệ thống điều hành ở mức chu kỳ lệnh.

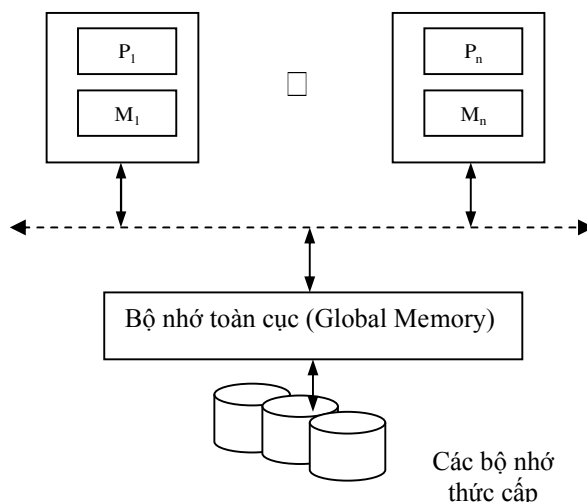
Một số hệ thống song song dựa trên kiến trúc chia sẻ bộ nhớ như: XPRS, DBS2 và Volcanol. Hệ quản trị cơ sở dữ liệu trên máy tính đa bộ vi xử lý chia sẻ bộ nhớ, sự thực thi của DB2 trên máy IBM 3090 với 6 bộ vi xử lý. Hầu hết các sản phẩm thương mại chia sẻ bộ nhớ đều lợi dụng song song hóa các liên truy vấn (Inter Query) để cung cấp khả năng thông qua các giao dịch lớn và song song qua các nội truy vấn, nhằm giảm thời gian đáp của truy vấn hỗ trợ quyết định.

Kiến trúc song song dựa trên việc chia sẻ bộ nhớ có những ưu điểm và nhược điểm khi triển khai cơ sở dữ liệu phân tán. Kiến trúc chia sẻ bộ nhớ có hai ưu điểm chính là đơn giản và cân bằng tải. Song song hóa các bộ vi xử lý có tính đơn giản. Các khái niệm liên quan đến bộ nhớ và các chính sách quản lý bộ nhớ không thay đổi. Thông tin về thư mục và các thông tin điều khiển (bằng khóa...) có thể thấy được chia sẻ bởi tất cả các bộ xử lý, vì vậy phần mềm ghi cơ sở dữ liệu không có sự khác biệt so với máy tính có một bộ xử lý. Cụ thể hơn, việc song song hóa các truy vấn giữa những bộ vi xử lý khác nhau có thể thực hiện một cách dễ dàng. Việc song song hóa truy vấn trong mỗi vi xử lý yêu cầu thực hiện một vài giao dịch song song nhưng đơn giản. Về cân bằng tải, kiến trúc chia sẻ bộ nhớ dễ dàng đạt được cân bằng tải khi thực thi, vì tất cả các đơn vị lưu trữ đều được sử dụng chung.

Tuy nhiên, kiến trúc chia sẻ bộ nhớ còn tồn tại một số vấn đề cần xem xét như giá cả, giới hạn khả năng mở rộng và tính khả dụng kém. Giá thành cao vì có các liên kết phức tạp và số lượng liên kết lớn khi mỗi bộ vi xử lý phải kết nối với mỗi modul bộ nhớ hay đĩa. Với các bộ vi xử lý tốc độ nhanh, dùng chung bộ nhớ trong quá trình truy cập bộ nhớ có thể dẫn đến số lượng xung đột lớn, vì vậy làm giảm hiệu năng hệ thống. Còn nữa, khi tăng thêm vi xử lý, ngoài việc phải tăng thêm nhiều kết nối, sự xung đột giữa các vi xử lý cũng vì đó có thể tăng nhanh, vì vậy khả năng mở rộng của hệ thống sử dụng kiến trúc này bị giới hạn trong phạm vi chục bộ vi xử lý. Nhưng nhược điểm cuối cùng là: vì không gian nhớ được chia sẻ bởi tất cả bộ vi xử lý, một bộ nhớ có lỗi, có thể ảnh hưởng đến tất cả các bộ vi xử lý, do đó làm giảm tính khả dụng của cơ sở dữ liệu.

5.4.3. Kiến trúc chia sẻ đĩa

Khác với kiến trúc chia sẻ bộ nhớ, trong kiến trúc chia sẻ đĩa (Shared-Disk), mỗi bộ vi xử lý truy cập đến mỗi đơn vị đĩa bất kỳ thông qua liên kết bên trong có bộ nhớ chính của riêng nó. Kiến trúc chia sẻ đĩa được biểu diễn trong hình 5.3.



Hình 5.3. Kiến trúc chia sẻ đĩa

Trong kiến trúc này, một bộ xử lý truy cập đến các đơn vị đĩa thông qua liên kết nối nhưng không được phép (không chia sẻ) truy cập đến bộ nhớ chính. Khi đó mỗi bộ xử lý có thể truy cập đến các trang dữ liệu (database page) trên ổ đĩa chia sẻ và sao chép chúng đến bộ nhớ cache của nó. Để tránh xung đột khi truy cập đến cùng một trang, cần phải có cơ chế khóa toàn cục (Global Locking) và các giao thức dùng để bảo trì sự gắn kết của cache.

Các ví dụ về các hệ thống CSDL song song chia sẻ ổ đĩa bao gồm sản phẩm chia sẻ dữ liệu IMS/VS của IBM và các sản phẩm VAX DBMS, Rdb của DEC. Sự thực thi của Oracle trên VAXcluster của DEC và các máy tính NCUBE cũng sử dụng kiến trúc chia sẻ ổ đĩa cứng khi nó yêu cầu mở rộng của hệ quản trị cơ sở dữ liệu quan hệ (RDBMS).

Chia sẻ đĩa có một số ưu điểm về giá thành, khả năng mở rộng, cân bằng tải trọng, tính sẵn sàng di chuyển từ các hệ thống có một bộ xử lý. Giá thành của kết nối giảm đáng kể so với phương pháp chia sẻ bộ nhớ từ khi công nghệ bus được dùng. Cho rằng mỗi bộ xử lý có đủ bộ nhớ cache, sự truy cập vào đĩa chia sẻ là nhỏ nhất, do đó sự mở rộng có thể tốt hơn. Khi đó bộ nhớ bị lỗi có thể bị cô lập với các bộ xử lý khác nhau, các node nhớ, tính sẵn sàng có thể cao hơn. Cuối cùng sự di chuyển từ hệ thống trung tâm tới đĩa chia sẻ dễ dàng hơn vì dữ liệu trên đĩa không cần tổ chức lại.

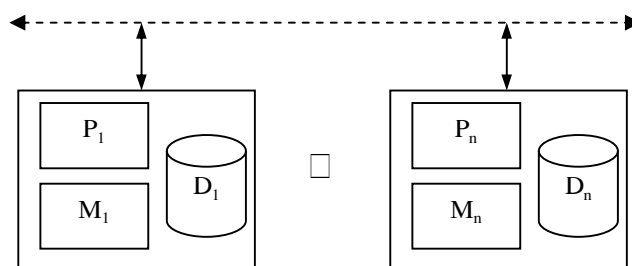
Chia sẻ đĩa có độ phức tạp cao hơn và hiệu năng cao hơn. Nó yêu cầu các giao thức của hệ phân tán dữ liệu như khóa phân tán và commit hai giai đoạn. Việc bảo trì độ kết dính của các bản sao có thể làm quá trình tải truyền thông giữa các node. Việc truy cập đĩa chia sẻ có thể gây ra hiện tượng “node cổ chai”.

5.4.4. Kiến trúc không chia sẻ

Trong kiến trúc không chia sẻ, mỗi bộ xử lý truy cập độc lập đến bộ nhớ chính và đơn vị ổ đĩa. Vì vậy mỗi node có thể được xem như một site cục bộ (về cơ sở dữ liệu và phần mềm) trong một hệ cơ sở dữ liệu phân tán. Vì vậy phần lớn các giải pháp được thiết kế cho các hệ phân tán như phân đoạn dữ liệu, quản lý phân tán giao dịch và xử lý truy vấn phân tán có thể được áp dụng. Các ví dụ về các hệ thống song song không chia sẻ DBC của Teradata và NonStopSQL của Tandem. Các sản phẩm truyền thống như GRACE, EDS, GAMMA, DUBBA, PRISMA cũng hiệu quả.

Các sản phẩm kiến trúc không chia sẻ có ba ưu điểm: giá thành, khả năng mở rộng và tính sẵn sàng. Ưu điểm về giá thành của phương pháp này cũng như kiến trúc chia sẻ đĩa. Hệ cơ sở dữ liệu phân tán được cài đặt trong kiến trúc này có thể dễ dàng tăng thêm hiệu năng khi thêm các node mới, khả năng mở rộng tốt hơn (có thể lên tới hàng ngàn node). Ví dụ hệ thống DBC của Teradata có thể cung cấp 1024 bộ xử lý. Với các phân vùng dữ liệu có ích được đặt trên nhiều đĩa. Tốc độ tăng lên theo tuyến tính và phạm vi tăng tuyến tính có thể đạt được khối lượng công việc đơn giản. Việc tạo các bản sao dữ liệu trên nhiều node có thể tăng tính sẵn sàng dữ liệu.

Kiến trúc không chia sẻ phức tạp hơn kiến trúc chia sẻ bộ nhớ bởi vì sự cần thiết phải cài đặt các chức năng phân tán dữ liệu tại nhiều vị trí khác nhau. Không như kiến trúc chia sẻ bộ nhớ và chia sẻ đĩa, độ cân bằng tính quyết định vị trí dữ liệu và tải không hiện thực của hệ thống, hơn nữa khi thêm các node mới vào hệ thống có thể yêu cầu phải tổ chức lại dữ liệu vì có liên quan đến vấn đề độ cân bằng tải.



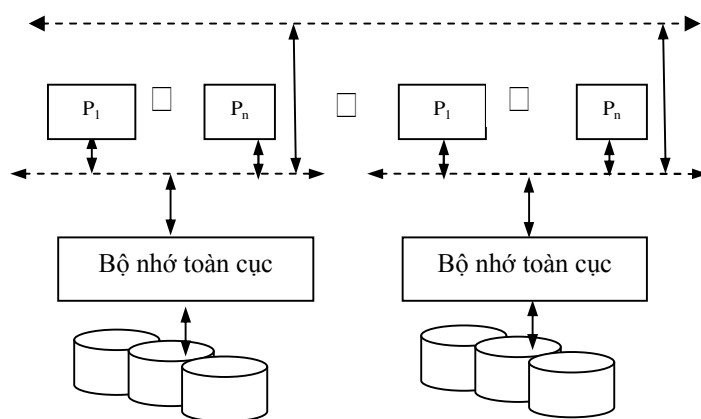
Hình 5.4. Kiến trúc không chia sẻ

5.4.5. Kiến trúc phân cấp

Kiến trúc phân cấp (Hierarchical Architectures) hay còn gọi là kiến trúc nhóm (Cluster Architecture) là sự kết hợp giữa kiến trúc không chia sẻ và kiến trúc chia sẻ bộ nhớ. Kiểu kiến trúc không chia sẻ, nhưng các node được thiết kế có kiến trúc chia sẻ chung một bộ nhớ. Kiến trúc này được đề xuất bởi Bhide, sau đó là Priahesh và Boral. Một mô tả chi tiết được đề xuất bởi Graefe.

Ưu điểm của kiến trúc phân cấp là hiển nhiên. Nó kết hợp đặc điểm linh hoạt và hiệu năng của thành phần chia sẻ bộ nhớ với khả năng mở rộng của thành phần không chia sẻ. Trong mỗi node chia sẻ chung một bộ nhớ giao tiếp được thực thi có hiệu quả bởi thành phần chia sẻ bộ nhớ của kiến trúc, do đó hiệu năng tăng lên. Độ cân bằng tải cũng tăng bởi thành phần chia sẻ bộ nhớ.

Mặc dù kiến trúc phân cấp có hiệu năng và tính khả dụng cao, nhưng hiệu năng hệ thống phụ thuộc vào hiệu quả của mỗi node và trao đổi thông tin giữa chúng. Sự không đồng bộ giữa các node sẽ dễ dàng dẫn đến sự mất cân bằng của hệ thống và ảnh hưởng rất lớn đến hiệu năng, đặc biệt là những chênh lệch về tốc độ và bộ nhớ giữa các node. Mặt khác, khi phát sinh lỗi, khả năng cô lập của hệ thống không linh hoạt, vì việc sử dụng bộ nhớ chung, ảnh hưởng hiệu năng của hệ thống.



Hình 5.5. Kiến trúc phân cấp

5.5. Kỹ thuật hệ quản trị cơ sở dữ liệu song song

Việc cài đặt các hệ CSDL song song thường dựa trên các kỹ thuật cơ sở dữ liệu phân tán, là những vấn đề về sắp xếp dữ liệu, truy vấn song song xử lý dữ liệu song song và tối ưu hóa truy vấn song song. Giải pháp cho những vấn đề này phức tạp hơn khi số lượng các node tăng. Các kỹ thuật hệ quản trị cơ sở dữ liệu song song được thiết kế trên kiến trúc không chia sẻ, là trường hợp tổng quát nhất và các kỹ thuật cài đặt của nó cũng có thể áp dụng cho các kiến trúc khác.

5.5.1. Sắp đặt dữ liệu

Khái niệm sắp đặt dữ liệu trong hệ cơ sở dữ liệu song song tương tự như các khái niệm phân mảnh ngang và phân mảnh dọc trong cơ sở dữ liệu phân tán. Việc phân mảnh sẽ làm tăng tính song song và cân bằng tải. Trong hệ cơ sở dữ liệu song song, khái niệm sắp đặt dữ liệu là khái niệm partition tương ứng với khái niệm phân mảnh ngang và phân mảnh dọc trong cơ sở dữ liệu phân tán.

Có hai khác biệt quan trọng so với cách tiếp cận cơ sở dữ liệu phân tán đó là:

- Không cần tăng tối đa việc xử lý cục bộ tại mỗi node, vì người sử dụng không được liên kết với một node cụ thể nào.
- Việc cân bằng tải rất khó đạt được khi số lượng các node quá lớn. Mặt khác khi giải quyết các vấn đề tranh chấp tài nguyên, dẫn đến sụp đổ toàn bộ hệ thống, ví dụ một node xử lý chiếm tất cả các tài nguyên, trong khi các node khác không được thực hiện. Việc chọn đặt dữ liệu là một vấn đề rất quan trọng về hiệu năng.

Việc sắp đặt dữ liệu phải nhằm làm tăng tối đa hiệu năng hệ thống bằng việc tổ hợp công việc thực hiện hệ thống và thời gian đáp ứng cho các câu truy vấn. Việc tăng tối đa thời gian đáp ứng của các truy vấn thực hiện song song làm cho tổng chi phí thời gian trao đổi giữa các node tăng lên và tổng công việc cũng tăng lên. Nếu làm tự tất cả các dữ liệu cần thiết cho một chương trình, sẽ làm giảm tổng lượng công việc được thực hiện cho một chương trình, sẽ làm giảm tổng lượng công việc được thực hiện bởi hệ thống khi thực hiện chương trình đó. Tóm lại vấn đề đặt dữ liệu theo hướng phân mảnh ngang dữ liệu sẽ làm tăng tối đa thời gian đáp ứng hoặc thực hiện song song truy vấn. Nếu theo hướng làm tự dữ liệu sẽ làm giảm thiểu tổng lượng công việc. Bài toán này trong cơ sở dữ liệu phân tán được giải quyết với các phương thức tĩnh. Người quản trị cơ sở dữ liệu có trách nhiệm kiểm tra theo định kỳ về tần số tham chiếu mảnh và khi cần phải di chuyển và tổ chức cấp phát lại các mảnh trên mạng máy tính.

Giải pháp lựa chọn cho việc sắp đặt dữ liệu là phân vùng toàn bộ. Trong đó mỗi một quan hệ được phân mảnh ngang tới tất cả các node trong hệ thống. Phân vùng toàn bộ được sử dụng trong DBC/1012, GAMMA, Nonstop SQL. Dưới đây là ba chiến lược cơ bản cho việc phân vùng dữ liệu: Round-Robin (xoay vòng), Hashing (hàm băm), Interval (khoảng).

Phân mảnh xoay vòng là chiến lược đơn giản nhất, đảm bảo sự phân tán dữ liệu được đồng nhất. Với n vùng Partition, hàng thứ i được chèn vào vùng thứ $i \bmod n$. Chiến lược này cho phép truy cập tuần tự tới một quan hệ được thực hiện song song. Tuy nhiên, khả năng truy cập tới các hàng riêng lẻ dựa trên việc truy cập đến các yêu cầu, thuộc tính của toàn bộ quan hệ.

Khi sử dụng kỹ thuật xoay vòng, kết quả của các phân mảnh sẽ đưa ra chính xác. Phương thức của phân mảnh cung cấp một phân phối dữ liệu trên nhiều đĩa, không cần chọn khóa phân mảnh hoặc khóa băm. Tuy nhiên, vì các dòng được phân phối một cách ngẫu nhiên, không thực hiện các thực thi trên những phân mảnh đã được xác định và các phép toán kết nối.

Phân mảnh băm: hàm băm H-function được hiểu như sau: Nếu mỗi bản ghi có một khóa là giá trị số (x), hàm băm $h(x)$ nhận một giá trị trong khoảng $[0-k]$ với k là một giá trị nguyên dương, k thường là một số nguyên tố, ví dụ $h(x) = x \bmod k$.

Chiến lược phân mảnh băm áp dụng hàm băm cho một vài thuộc tính và tạo ra một số Partition. Chiến lược này cho phép một node nhất định được xử lý tất cả các truy vấn khác một cách song song. Để thực hiện phân mảnh hàm băm, phân chia tập hợp các bản ghi của tệp dữ liệu thành các cụm (Buckets). Mỗi cụm bao gồm một hoặc nhiều khối (Block). Mỗi khối chứa một số lượng cố định các bản ghi. Việc tổ chức lưu trữ dữ liệu trong mỗi cụm được áp dụng theo đồng, tức là các bản ghi được lưu trữ kế tiếp nhau trong các khối và không tuân theo một thứ tự đặc biệt nào và không có một tổ chức đặc biệt nào được áp dụng với các khối. Với mỗi tệp được lưu trữ theo phương pháp này cần có một hàm băm $h(x)$ lấy đối số là giá trị khóa của tệp và sinh ra một số nguyên từ 0 đến một giá trị tại cực đại $B-1$, B là số cụm của tổ chức tệp băm. Nếu x là một giá trị khóa, $h(x)$ xác định chỉ số của cụm mà bản ghi có giá trị khóa này được tìm thấy nếu nó được chứa trong tệp dữ liệu.

Phân mảnh theo khoảng: phân mảnh các bộ dựa trên các khoảng giá trị của một thuộc tính. Phân mảnh theo khoảng rất thích hợp cho truy vấn phân mảnh băm và các truy vấn khoảng. Ví dụ, mỗi truy vấn “ A nằm giữa A_1 và A_2 ” có thể được xử lý trên các node chứa các bộ giá trị nằm trong đoạn $[A_1, A_2]$. Tuy nhiên phân mảnh khoảng có thể dẫn đến một sự khác biệt rất lớn về kích thước các phân hoạch. Phân mảnh khoảng cho phép người sử dụng xác định các khoảng dữ liệu không bị trùng và sau đó đặt mỗi dòng vào những vùng thích

hợp, dựa trên giá trị của những cột đặc thù trong dòng. Vì vậy, yêu cầu chọn cột được gọi là “khóa dành riêng (Partition key)” bởi những phần dữ liệu sẽ được chỉ ra vị trí vật lý rõ ràng. Ví dụ, trong quan hệ khách hàng, có thể chọn cột “last_name” như một khóa riêng xác định đánh dấu khoảng A-E trên một đĩa, khoảng từ F-J trên một đĩa khác...

Muốn xác định khóa dành riêng tại cột thường xuyên, sử dụng mệnh đề Where của câu lệnh Select. Ví dụ nếu thường xuyên quan tâm đến kiểu của các loại xe trong cơ sở dữ liệu Car- Sales, sử dụng cột Type _Car là một lựa chọn hợp lý làm khóa dành riêng. Sau đó, tạo phân mảnh bảng Car_Sales.

```
CREATE TABLE Car_Sale(  
    type CHAR  
    model CHAR  
    price NUM  
    dealer CHAR  
    sale_date DATE)  
PARTITION BY VALUE  
    type = 'Mercedes' IN partition_1,  
    type = 'Porsche' IN partition_2,  
    type = 'BMW' IN partition_3,  
    type = 'Volvo' IN partition_4;
```

Giả sử thực thi câu truy vấn sau:

```
SELECT type, model, price, dealer, sale_date  
FROM Car_Sales  
WHERE type= 'BMW' AND  
    price BETWEEN (300000 and 45000)
```

Dữ liệu được yêu cầu (xe BMW) chỉ nằm ở trong partition_3, vì thế không cần phải sử dụng bất kỳ tài nguyên nào để truy vấn dữ liệu ngoài một đĩa và một CPU để phục vụ quá trình tìm kiếm này. Thay vì việc sử dụng cả 4 CPU và cả 4 đĩa, chỉ cần sử dụng một mảnh đã được xác định nghĩa là chỉ cần đến 1 CPU và 1 đĩa, và như vậy quá trình tìm kiếm đã làm giảm được 75% thời gian tìm dữ liệu.

Phân mảnh khoảng có thể tạo ra những câu truy vấn có hiệu quả rõ rệt bằng những phân mảnh đã được xác định. Cần chú ý về kích thước dữ liệu, khi dữ liệu không được chia bằng nhau giữa các phân mảnh.

Phân mảnh khoảng và băm rất có ích khi thực thi có sử dụng việc kết nối theo cụm (Clustered) và trong kiến trúc MPP. Phân mảnh được xác định trong những node riêng biệt và những node đó có thể kết nối với tốc độ cao và có thể tối thiểu hóa dữ liệu chuyển trong mạng để đạt được tối ưu hóa độ mềm dẻo. Sử dụng phân mảnh băm hoặc phân mảnh khoảng trong khóa nối cho mọi bản ghi với những câu truy vấn có nối, đảm bảo nối luôn luôn tìm thấy trong cùng một phân mảnh và trong cùng một node. Dữ liệu chuyển trong một node sẽ được tối thiểu hóa vì mọi phép nối được thực hiện nội bộ và vì thế hiệu năng thực thi sẽ tăng lên.

So sánh ba chiến lược:

Round-robin partitioning	Hash partitioning	Range partitioning
Dữ liệu phân chia đều	Dữ liệu được phân chia đều, giả sử có một hàm băm tốt	Phù hợp với truy vấn vùng trong một thuộc tính phân mảnh
Phù hợp cho kiểm tra truy vấn quan hệ đầy đủ	Phù hợp truy vấn điểm và kết nối bằng trong thuộc tính phân mảnh	Chọn thuộc tính để phân mảnh là quan trọng
Không phù hợp với truy vấn vùng	Không phù hợp với truy vấn khoảng.	Lựa chọn thuộc tính chia sẽ dẫn đến dữ liệu phân chia không đều và thực thi cũng không đều

Hiệu năng phân mảnh hoàn toàn được so sánh với hiệu năng làm tự các quan hệ vào một đĩa duy nhất. Các kết quả chỉ ra rằng với nhiều loại tải trong phân mảnh luôn tốt hơn. Tuy nhiên, làm tự có thể trội hơn khi xử lý các vấn đề phức tạp, ví dụ các phép toán nối. Lưu lượng của một hệ thống khi có sự phân mảnh hoàn toàn được chứng minh là tăng một cách tuyến tính với 32 node.

Mặc dù phân mảnh hoàn toàn có những ưu điểm về hiệu năng, việc thực thi song song nhiều cũng dẫn đến chi phí lớn về những câu truy vấn phức tạp có kết nối. Ví dụ, trong một kiến trúc 1024 node, số lượng thông báo trong trường hợp xấu nhất cho một kết nối hai ngôi không có chọn sẽ lên đến 1024². Hơn nữa

phân mảnh hoàn toàn không thích hợp cho các quan hệ nhỏ trải dài qua một số ít khối trên đĩa. Những nhược điểm này cần tìm ra sự cân đối giữa làm tụ và phân mảnh hoàn toàn.

Người ra đề xuất một giải pháp cho vấn đề sắp xếp dữ liệu bằng phương pháp phân mảnh thay đổi. Mức độ phân hoạch, tức là số node mà một quan hệ được phân mảnh, là một hàm theo kích thước và tần số truy xuất của quan hệ. Chiến lược này phức tạp hơn so với phân mảnh hoàn toàn và kỹ thuật tụ. Những thay đổi về phân phối dữ liệu có thể dẫn đến việc phải tổ chức lại.

Phân mảnh thay đổi nếu được sử dụng trong một hệ thống song song mức cao, việc tổ chức lại theo định kỳ để cân bằng tải có vai trò quyết định và cần được thực hiện thường xuyên trừ khi tải trọng khá ổn định và chỉ có một số ít được cập nhật. Việc tổ chức lại phải được vô hình đối với các chương trình đã được biên dịch cần được giữ độc lập với vị trí dữ liệu. Sự độc lập có thể đạt được nếu hệ thống lúc chạy có hỗ trợ phương thức truy xuất kết hợp đến dữ liệu phân tán. Điều này khác với một cơ sở dữ liệu phân tán, trong đó truy xuất kết hợp có thể đạt được vào lúc biên dịch nhờ thể xử lý truy vấn bằng cách dùng thư mục dữ liệu.

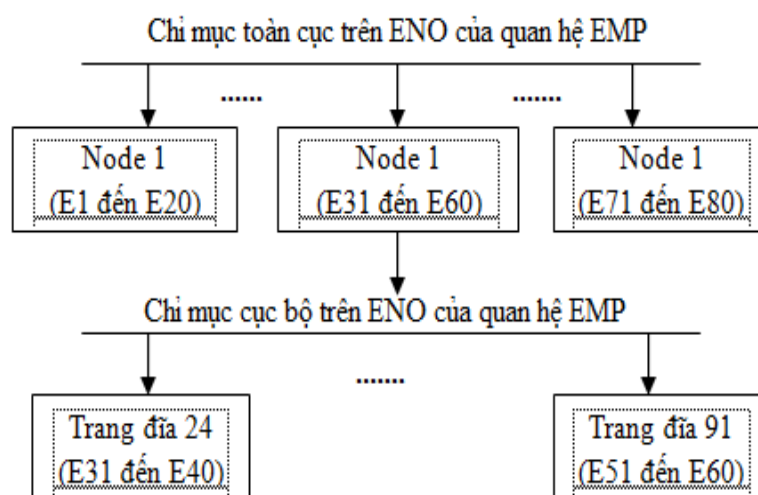
Cơ chế chỉ mục: Một giải pháp cho vấn đề truy xuất kết hợp là dùng một cơ chế chỉ mục toàn cục được nhân bản tại mỗi node. Chỉ mục toàn cục sẽ chỉ ra chỗ đặt một quan hệ trên một tập node. Về mặt khái niệm, chỉ mục toàn cục là một chỉ mục hai mức với mức tụ chính theo tên quan hệ và mức tụ phụ trên một thuộc tính nào đó của quan hệ. Chỉ mục toàn cục hỗ trợ cho phân mảnh thay đổi, ở đó mỗi quan hệ có một phân mảnh khác nhau. Cấu trúc chỉ mục dựa trên kỹ thuật băm hoặc trên một tổ chức B-tree. Trong cả hai trường hợp, các truy vấn giống hệt được xử lý bằng một truy xuất node. Tuy nhiên với kỹ thuật băm, các truy vấn khoảng được xử lý hiệu quả bằng cách truy xuất tất cả các node có chứa dữ liệu từ quan hệ đang được vấn tin. Sử dụng chỉ mục B-tree cho phép xử lý các vấn tin khoảng hiệu quả hơn, trong đó chỉ các node chứa dữ liệu trong khoảng được yêu cầu mới được truy xuất.

Ví dụ 5.1: Xét quan hệ EMP (ENO, ENAME, DEPT, TITLE).

Giả sử muốn xác định các phần tử trong quan hệ EMP có giá trị ENO là E50. Chỉ mục mức thứ nhất trên tập tên ánh xạ tên EMP vào chỉ mục trên thuộc tính ENO của quan hệ EMP cung cấp các kết quả thử nghiệm cho phân mảnh

thay đổi với một tải trọng được tạo ra do trộn lẫn các giao dịch ngắn và các giao dịch phức tạp. Kết quả chỉ ra rằng, khi phân mảnh tăng lên, lưu lượng tiếp tục tăng với các giao dịch ngắn. Tuy nhiên, với các giao dịch phức tạp cần thực hiện nhiều phép nối, phân mảnh tiếp tục sẽ làm giảm lưu lượng do chi phí truyền giao dịch tăng lên.

Sắp đặt dữ liệu cần giải quyết các vấn đề phân phối dữ liệu lệch có thể dẫn đến việc phân mảnh không thống nhất và ảnh hưởng đến cân bằng tải. Ảnh hưởng do lệch của phân mảnh khoảng thường cao hơn so với phân mảnh xoay vòng hoặc băm. Một giải pháp xử lý vấn đề phân phối dữ liệu lệch của phân mảnh không thống nhất là phân mảnh tiếp tục cho các phân mảnh lớn, tách biệt giữa các node logic và vật lý, vì một node logic có thể tương ứng với nhiều node vật lý.



Hình 5.6. Ví dụ về chỉ mục toàn cục và cục bộ

Nhân bản dữ liệu: Nhân bản dữ liệu nhằm đảm bảo độ khả dụng cao. Giải pháp đơn giản là duy trì hai bản sao của cùng một dữ liệu, một bản chính và một bản dự phòng trên hai máy riêng biệt. Đây là kiến trúc đĩa ảnh (Mirrored Disk) như đã được vận dụng trong hệ thống nonStop SQL của Tandem. Tuy nhiên trong trường hợp một node bị sự cố tải trọng có thể nhân đôi tại node có bản sao, vì thế sẽ ảnh hưởng đến cân bằng tải. Để tránh vấn đề này, nhiều chiến lược nhân bản dữ liệu đã được đề xuất cho các cơ sở dữ liệu song song. Một giải pháp khác là phân mảnh đan xen của Teradata, phân mảnh bảng dữ liệu trên một số node. Khi có sự cố, tải trọng của bảng chính sẽ được cân đối giữa các

node có bản sao. Nếu cả hai node xảy ra sự cố thì quan hệ đó không truy xuất được, làm ảnh hưởng đến độ khả dụng. Xây dựng lại bản chính từ bản sao dự phòng có thể tốn nhiều chi phí. Trong tình huống bình thường, để duy trì tính nhất quán cho các bản có thể có chi phí cao.

Bảng 5.1. Ví dụ phân mảnh đơn xen

Node	1	2	3	4
Bản chính	R_1	R_2	R_3	R_4
Bản dự phòng	$r_{2,3}$ $r_{3,2}$	$r_{1,1}$ $r_{3,3}$	$r_{1,2}$ $r_{2,1}$	$r_{1,3}$ $r_{2,2}$ $r_{3,1}$

Phân mảnh đơn xen chéo mắt xích của Gamma, bản chính và bản dự phòng được lưu trên hai node kế nhau. Phân mảnh sao cho xác suất bị sự cố hai node kế nhau nhỏ hơn xác suất bị sự cố hai node bất kỳ. Trong trường hợp bị sự cố, tải của node bị sự cố và các node dự phòng được cân đối cho các node còn lại bằng cách sử dụng node bản chính và bản dự phòng. Chi phí duy trì tính nhất quán các bản thấp hơn. Việc chọn đặt dữ liệu có xem xét đến nhân bản dữ liệu, tương tự như việc cấp phát mảnh trong cơ sở dữ liệu phân tán, được xem xét như một bài toán tối ưu hóa.

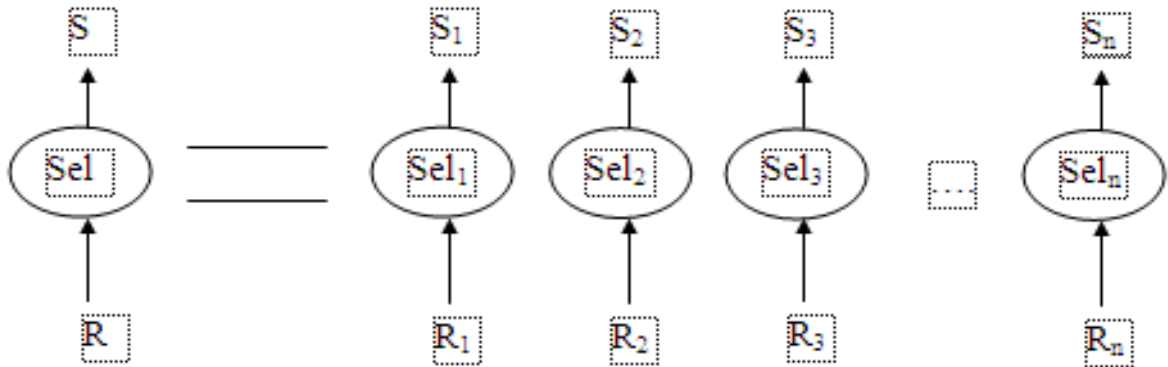
Bảng 5.2. Ví dụ phân mảnh mắt xích

Node	1	2	3	4
Bản chính	R_1	R_2	R_3	R_4
Bản dự phòng	r_4	r_1	r_2	r_3

5.5.2. Truy vấn song song

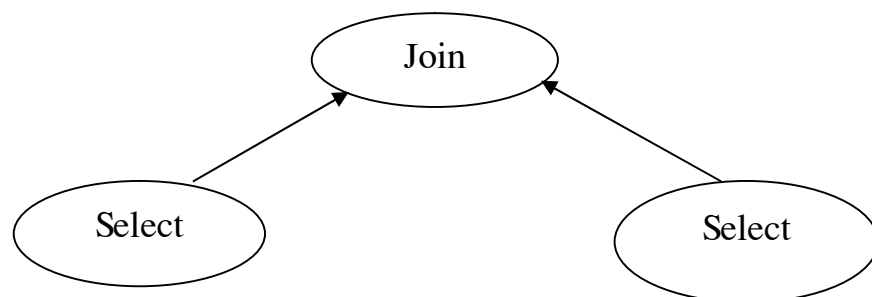
Truy vấn song song theo nghĩa thực hiện song song, đồng thời nhiều câu vấn tin được sinh ra bởi nhiều giao dịch đồng thời. Truy vấn song song làm tăng lưu lượng giao dịch. Trong một câu truy vấn (song hành nội truy vấn), song hành nội toán tử và liên toán tử được sử dụng để giảm thời gian đáp ứng. Song hành liên toán tử có được bằng cách cho thực thi song song nhiều toán tử của cấu trúc cây vấn tin. Trên nhiều bộ xử lý trong khi đó song hành nội toán tử, một toán tử sẽ được nhiều bộ xử lý trong khi đó song hành nội toán tử, một toán tử sẽ được nhiều bộ xử lý thực hiện, mỗi bộ xử lý thao tác trên một tập con dữ liệu.

Song hành nội toán tử dựa trên việc phân rã một toán tử thành tập con các toán tử độc lập, được gọi là thể hiện toán tử (operator Instance), bằng cách sử dụng kỹ thuật phân mảnh tĩnh hoặc động cho các quan hệ. Mỗi thể hiện toán tử sẽ thực hiện một phân mảnh quan hệ, được gọi là lô (Batch). Để minh họa cho việc song hành nội toán tử, xét một truy vấn chọn – nối đơn giản. Toán tử chọn (Select) có thể phân rã trực tiếp thành nhiều toán tử chọn, mỗi toán tử thao tác trên một phân mảnh khác nhau không cần phải thực hiện tái phân phối.



Hình 5.7. Ví dụ song hành nội toán tử

Song hành liên toán tử có thể được sử dụng với song hành ống dẫn (Pipeline Parallelism) nhiều toán tử với một đường nối sản xuất – tiêu dùng được thực thi song song. Ví dụ toán tử Select được thực thi song song với toán tử nối (Join) kế tiếp. Ưu điểm của thực thi theo phương pháp này là kết quả trung gian không phải cụ thể hóa (không phải lưu lại), vì thế tiết kiệm bộ nhớ và truy xuất đĩa. Tuy nhiên, nó chỉ có thể xảy ra với cách thực thi nhiều nhánh và đòi hỏi nhiều tài nguyên hơn.



Hình 5.8. Ví dụ về một song hành liên toán tử

5.5.3. Xử lý dữ liệu song song

Cơ sở cho việc thực hiện truy vấn dữ liệu song song là việc phân mảnh dữ liệu và cấp phát dữ liệu. Việc cấp phát các mảnh dữ liệu có vai trò rất quan trọng

trong việc thiết kế các thuật toán song song và điều khiển việc dữ liệu một cách hiệu quả bằng các toán tử đại số quan hệ. Thực hiện câu truy vấn dữ liệu gồm nhiều toán tử. Vì vậy, cần phải đảm bảo sự cân bằng giữa tính song song và chi phí cho quá trình trao đổi thông tin. Thuật toán song song cho các toán tử quan hệ đại số được xây dựng thành các khối cần thiết cho việc xử lý truy vấn song song.

Xử lý dữ liệu song song có thể sử dụng nhiều phép toán song hành nội toán tử. Xử lý toán tử Setlet trong ngữ cảnh sắp xếp dữ liệu phân mảnh cũng như xử lý toán tử Select trong cơ sở dữ liệu phân tán. Nếu chỉ mục được tổ chức dưới dạng cấu trúc B-tree khi đó toán tử Select với một kích cỡ xác định có thể chỉ được thực hiện bởi những node lưu trữ dữ liệu thích hợp.

Việc xử lý dữ liệu song song cho toán tử kết nối Join phức tạp hơn nhiều so với toán tử chọn Select. Tính sẵn sàng của toàn bộ chỉ mục tại thời gian chạy cung cấp đem lại nhiều thuận lợi cho việc thực hiện song song một cách có hiệu quả. Sau đây là ba thuật toán kết nối song song cơ bản cho việc phân mảnh dữ liệu. Thuật toán vòng lặp lồng song song (The Paralled Nested Loop – PNL), thuật toán nối kết hợp song song (The Paralled Associative Join – PAJ) và thuật toán nối băm song song (The Paralled Hash Join – PHJ).

Giả sử quan hệ R được phân mảnh thành m mảnh và cấp phát trên m vị trí. Quan hệ S đã được phân mảnh n và cấp phát trên n vị trí. Giả thiết m node và n node tách rời nhau. Các node chứa các mảnh của quan hệ R được ký hiệu là R – node và tương tự, node chứa mảnh dữ liệu của S ký hiệu là S-node.

Thuật toán vòng lặp lồng song song (Paralled nested Loop – PNL) tạo ra song song tích Đề các của các quan hệ R và S. Vì vậy, có thể hỗ trợ cho các vị trí nối phức tạp. Thuật toán INGRES phân tán là phiên bản của thuật toán này. Kết quả kết nối sinh ra tại các S-node. Sau đây là thuật toán:

Thuật toán PNL

Input: R_1, R_2, \dots, R_n : Các mảnh của quan hệ R;

S_1, S_2, \dots, S_n : Các mảnh của quan hệ S;

JP: Vị trí xác nhận kết nối;

Output: T_1, T_2, \dots, T_n : Các đoạn kết quả;

Begin

For i from 1 to m do in parallel


```

    Gửi  $R_1$  đến mỗi node có chứa một mảnh của S.
end for
for j from 1 to n do in parallel {Thực hiện nối tại mỗi S-node}
begin
 $R \leftarrow \bigcup R_i$  {Nhận  $R_1$  từ các R-node; R được nhân bản trên các S-node}
 $T_j \leftarrow \text{JOIN}(R, S_j, JP)$ 
End-for
End. {PNL}

```

Giải thích hoạt động thuật toán:

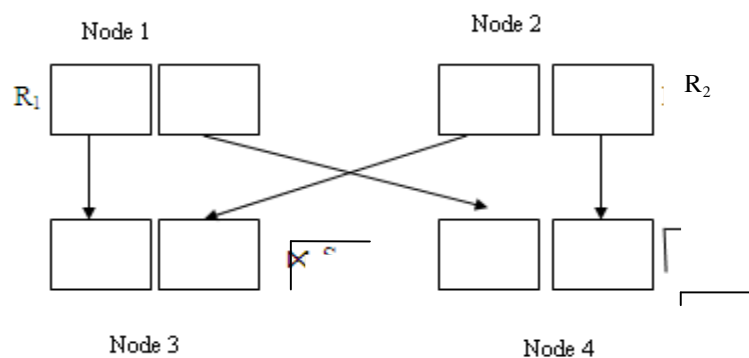
- Mỗi mảnh của R được gửi và nhân bản tại mỗi S-node (có n node). Vì R có m mảnh nên sẽ có m node được thực hiện song song và mỗi mảnh được truyền theo phương thức quảng bá (Broadcast) đến n node. Như vậy tổng chi phí cho m thông báo.

- Sau khi các S-node nhận tất cả các mảnh quan hệ R, thực hiện đồng thời, song song việc kết nối R với mảnh S tại các S-node. Việc thực hiện kết nối cục bộ xảy ra như trong môi trường tập trung. Có thể xử lý kết nối ngay khi mỗi node nhận dữ liệu. Nếu sử dụng thuật toán nối vòng lặp lồng PNL, việc xử lý có thể được thực hiện theo kiểu đường ống ngay khi bộ dữ liệu của R đi vào. Trong trường hợp áp dụng thuật toán nối sắp xếp trộn SMJ, tất cả dữ liệu phải nhận được trước khi kết nối của các quan hệ đã sắp.

Thuật toán vòng lặp lồng song song PNL thay thế toán tử $R \bowtie S$ bằng:

$$\bigcup (R_i \bowtie S_i)$$

Ví dụ 5.2: Xét thuật toán lặp lồng song song PNL mới $m = n = 2$ trên 4 node.



Hình 5.9. Ví dụ về thuật toán vòng lặp lồng song song PNL

Thuật toán nối kết hợp song song (PAJ): Giả sử vị từ kết nối là thuộc tính A của R và B của S. Giả sử quan hệ S thì được phân mảnh theo hàm băm h và được kết nối theo thuộc tính B, nghĩa là tất cả bộ dữ liệu của S có cùng giá trị h(B) sẽ được đặt tại một node. Áp dụng thuật toán nối kết hợp song song PAJ sẽ sinh ra kết quả kết nối tại các S-node.

Input: R_1, R_2, \dots, R_n : Các mảnh của quan hệ R;

S_1, S_2, \dots, S_n : Các mảnh của quan hệ S;

JP: Vị từ kết nối;

Output: T_1, T_2, \dots, T_n : Các mảnh kết quả;

Begin {R.A=S.B và quan hệ S là các phân mảnh theo hàm h(B)}

For i from 1 to m do in parallel

Begin

Rij apply h(A) to R_i ($j = 1, \dots, n$)

For j from 1 to n do

Send Rij to the node storing S_j

End-for

End-for

For j from 1 to n do in parallel {thực hiện việc nối tại S-node}

Begin

$R_j \leftarrow R_{ij}, i = 1, \dots, m$

$b_j \leftarrow \text{JOIN}(R_j, S_j, \text{JP})$

end-for

End. (PAJ)

Thuật toán thực hiện như sau:

- Quan hệ R truyền và kết hợp với các S-node dựa trên hàm h đã được áp dụng cho thuộc tính A. Điều này đảm bảo rằng một bộ dữ liệu của R cùng với giá trị băm v chỉ được gửi tới S-node chứa các bộ với giá trị hàm băm v. Giai đoạn này thực hiện song song bởi m R-node. Không như thuật toán PNL, các bộ dữ liệu của R được phân tán nhưng không được nhân bản trong các S-node.

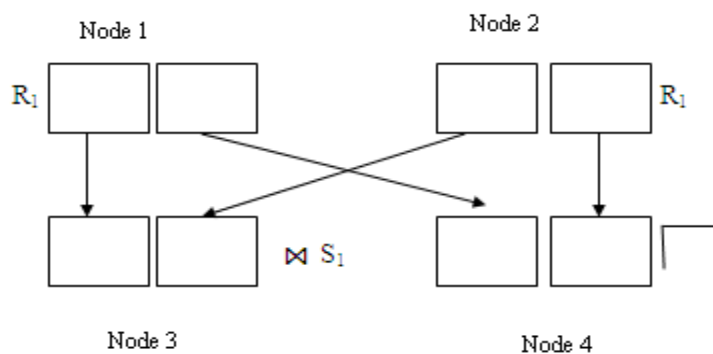
- S-node nhận song song các mảnh R và kết nối tại các mảnh S-node, xử lý kết nối cục bộ được thực hiện như trong thuật toán vòng lặp lồng song song PNL.

Tóm lại, thuật toán nối kết hợp song song PAJ thay thế toán tử R \bowtie S bằng:

$$\bigcup (R_1 \bowtie S_i)$$

Ví dụ 5.3: Thuật toán nối kết hợp song song với $n=m=2$ và trên 4 node. Các hình vuông có cùng màu là các mảnh với hàm băm giống nhau.

Thuật toán nối băm song song PHJ (Parallel Hash Join) là sự tổng quát hóa của thuật toán nối kết hợp song song, được áp dụng trong trường hợp kết nối bằng nhau không cần sự phân mảnh cụ thể nào của quan hệ toán hạng. Ý tưởng cơ bản của thuật toán là phân mảnh cụ thể nào của quan hệ toán hạng. Ý tưởng cơ bản của thuật toán là phân mảnh các quan hệ R và S thành p mảnh: R_1, R_2, \dots, R_p và S_1, S_2, \dots, S_p sao cho: $R \bowtie S = \bigcup (R_i \bowtie S_i)$.



Hình 5.10. Ví dụ về thuật toán nối kết hợp song song PAJ

5.6. Tối ưu hóa truy vấn song song

5.6.1. Không gian tìm kiếm

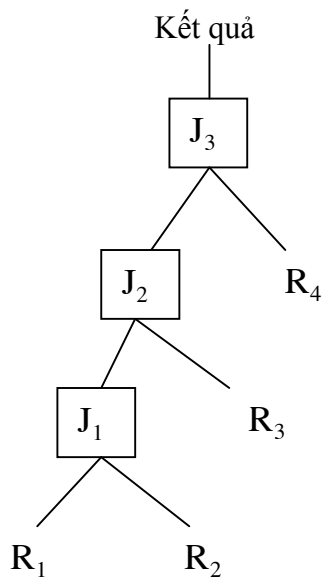
Tập các phương án thực thi tương đương một biểu diễn cho một câu truy vấn, theo nghĩa sinh ra cùng một kết quả nhưng khác nhau về thứ tự thực hiện các toán tử và cách cài đặt chúng. Phương án thực thi được trừu tượng qua cây toán tử, định nghĩa toán tử được thực hiện qua cây toán tử. Hai toán tử liên tiếp có thể thực thi theo kiểu ống dẫn hay phụ trữ lưu lại.

Phụ trữ ống dẫn và phụ trữ lưu lại ràng buộc sự xếp lịch của các phương án thực thi. Tách cây toán tử thành cây con không gối chồng lên nhau gọi là các pha. Các toán tử trong ống dẫn được thực thi trong cùng một pha. Phụ trữ lưu lại sẽ thiết lập ranh giới của một pha và pha tiếp theo. Một số toán tử và thuật toán đòi hỏi một toán hạng sẽ được lưu lại.

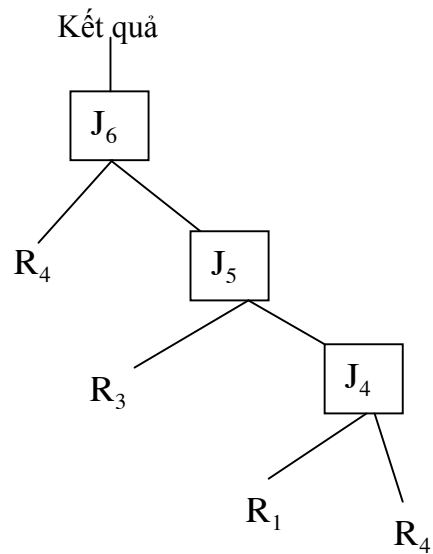
Thuật toán nối băm song song gồm hai pha liên tiếp: pha xây dựng và pha dò tìm. Trong pha xây dựng, một bảng băm được xây dựng trên thuộc tính nối

quan hệ nhỏ nhất. Trong pha dò tìm, quan hệ lớn nhất được quét tuần tự và bảng băm được tham vấn cho mỗi bộ của nó.

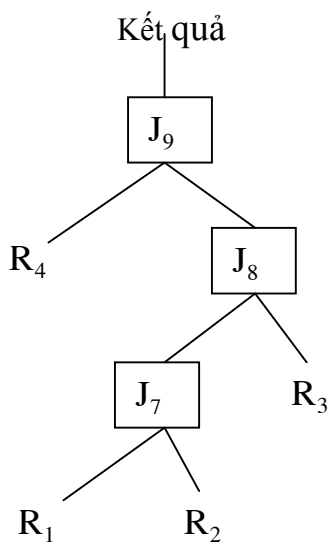
Tập hợp các node lưu trữ quan hệ gọi là vị chủ của nó. Vị chủ của một toán tử là tập các nodenơi nó thực thi và phải là vị chủ của các toán hạng để toán tử đó truy xuất toán hạng của nó. Với các toán tử hai ngôi như kết nối, có thể kéo theo việc phân hoạch lại một trong hai toán hạng. Thẻ tối ưu hóa cũng có khi cần phải tái phân hoạch cả hai toán hạng. Cây phụ trú thực thi chỉ ra việc phân hoạch.



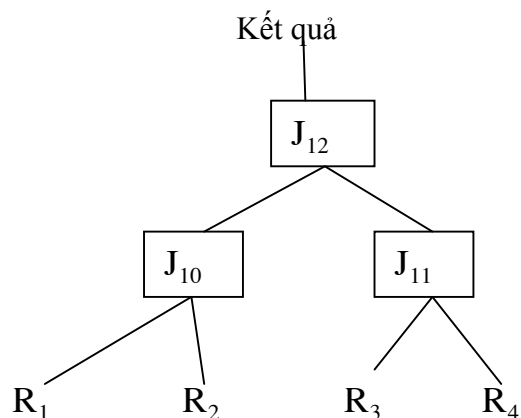
a) Xâu trái



b) Xâu phải



c) Zigza



d) Xum xuê

Hình 5.11. Các cây toán tử biểu diễn các phương án thực thi

Một cây toán tử là cây nhị phân có nhân, node lá là các quan hệ của câu truy vấn và node không phải là node lá là một node toán tử (ví dụ kết nối, phép hợp...), kết quả là một quan hệ trung gian. Một node nối thực hiện nối giữa các toán hạng của. Các cung có hướng (hoặc vô hướng) được hiểu là quan hệ trung gian sinh ra bởi một node được sử dụng theo kiểu ống dẫn (hay lưu lại) bởi node tiếp theo. Cây toán tử cũng có thể thuộc loại tuyến tính, nghĩa là ít nhất một toán hạng của mỗi node nối là một quan hệ cơ sở hoặc thuộc loại xum xuê. Trong hình 5.11, có 4 cây toán tử. Cây toán tử xâu trái là kiểu phụ trữ lưu lại và cây toán tử xâu phải là kiểu phụ trữ ống dẫn, hai cây toán tử còn lại là các quan hệ trung gian.

Sẽ rất tiện lợi khi biểu diễn các quan hệ ống dẫn như nguyên liệu bên phải của một toán tử với việc cây xâu phải biểu diễn kiểu ống dẫn hoàn toàn và cây xâu trái phải biểu diễn kiểu ống dẫn hoàn toàn và cây xâu trái thể hiện vật chất hóa hoàn toàn các kết quả trung gian, thì cây xâu phải sẽ thực hiện hiệu quả hơn các cây xâu trái tương ứng nhưng có xu hướng chi phí nhiều bộ nhớ hơn để lưu lại các quan hệ bên trái.

Các kiểu cây song song cũng rất đáng chú ý. Cây xum xuê ở trên là những cây duy nhất cho song hành độc lập. Song hành độc lập có hiệu quả khi các quan hệ thực hiện phân hoạch trên các vị chủ rời nhau. Các nối J_1 và J_2 có thể thực thi song song một cách độc lập bởi tập các node cấu tạo nên.

Có sự thực hiện song hành giữa phụ trữ và ống dẫn, các cây zic zắc là các quan hệ trung gian giữa cây xâu phải và cây xâu trái, sử dụng bộ nhớ chính tốt. Một Heuristic hợp lý là ưu tiên chọn các cây xâu phải hoặc zic zắc khi các quan hệ phân mảnh một phần trên các vị chủ tách biệt và các quan hệ trung gian khá lớn. Trong trường hợp này thì các cây xum xuê thường dùng nhiều pha và mất nhiều thời gian thực thi hơn. Đảo lại khi các quan hệ trung gian nhỏ, thì ống dẫn không hiệu quả, vì khó cân bằng tải giữa các giai đoạn của ống dẫn.

5.6.2. Mô hình chi phí

Để xác định chính xác mô hình chi phí thực thi của một phương án cần phải biết được môi trường thực thi song song. Mô hình chi phí của thể tối ưu hóa đánh giá chi phí của một phương án thực thi, gồm 2 phần: Phần phụ thuộc kiến trúc và phần độc lập kiến trúc. Phần độc lập kiến trúc được cấu tạo bởi các hàm chi phí của các thuật toán tử, ví dụ thuật toán vòng lặp lồng cho các nối và truy

xuất tuần tự cho phép chọn. Nếu bỏ qua hoạt động đồng thời, thì chỉ có hàm chi phí phân hoạch lại dữ liệu và chi phí bộ nhớ là khác nhau và cấu tạo nên phần phụ thuộc kiến trúc. Việc phân hoạch lại các bộ của một quan hệ trong hệ thống kéo theo việc di chuyển dữ liệu qua các kênh kết nối, để trở thành hành động bấm trong các hệ thống bộ nhớ chung, chi phí bộ nhớ trong hệ thống riêng rất phức tạp bởi sự song hành liên toàn từ. Trong hệ thống bộ nhớ chung tất cả toán tử đọc và ghi thông qua bộ nhớ toàn cục và rất dễ kiểm tra xem có đủ không gian để thực hiện song song hay không, nghĩa là tổng số chi phí nhỏ hơn bộ nhớ còn lại.

Trong các hệ thống riêng, mỗi bộ xử lý có riêng một bộ nhớ, có thể biết được toán tử nào thực thi song song trên cùng một bộ xử lý. Vì thế để đơn giản, giả thiết rằng các bộ xử lý được gán để thực thi các toán tử sẽ không gối chồng lên nhau, nghĩa là phần giao của tập các bộ xử lý là rỗng hoặc đồng nhất.

Một phương án có khả năng thực thi song song được định nghĩa gồm ba thành phần tổng công việc W (Total Work), thời gian đáp ứng RT (Response Time) và sự chi phí bộ nhớ MC (Memory Consumption). WT và RT tính theo giây và MC tính theo Kbyte. Thành phần TW và RT diễn tả việc hoán đổi thời gian giữa đáp ứng và lưu lượng. Thành phần thứ ba biểu diễn kích thước bộ nhớ cần cho việc thực thi phương án. Hàm chi phí là tổ hợp của hai thành phần đầu tiên và các phương án cần nhiều bộ nhớ sẵn có sẽ bị loại bỏ.

5.6.3. Chiến lược tìm kiếm

Khám phá không gian tìm kiếm và chọn lựa ra phương án thứ tự nào tốt nhất, không nhất thiết phải khác nhau trong truy vấn phân tán hay tập trung.

Các thể tối ưu thường sử dụng chiến lược tìm kiếm quy hoạch động với tính chất đơn định, bằng cách xây dựng các hoạch định bắt đầu từ quan hệ cơ sở, kết nối thêm nhiều quan hệ tại mỗi bước cho đến khi thu được tất cả các hoạch định khả hữu. Quy hoạch động xây dựng tất cả kế hoạch khả hữu theo hướng ngang (Breadth First) trước khi chọn ra hoạch định tốt nhất. Để hạ thấp chi phí tối ưu hóa, các hoạch định từng phần rất có thể không dẫn tới khả năng dẫn tới một hoạch định tối ưu đều được xén bỏ ngay khi có thể. Ngược lại, một chiến lược đơn định khác là thuật toán thiên cận chỉ xây dựng một hoạch định theo chiều sâu (Depth First).

Bản chất của quy hoạch động là thực hiện vét cạn và đảm bảo tìm ra được các hoạch định. Nó phải trả một chi phí thời gian và theo không gian thấp, khi số

quan hệ trong câu truy vấn không nhiều. Tuy nhiên kiểu tiếp cận này có chi phí khá cao khi số quan hệ lớn hơn 5. Vì vậy, người ta chỉ quan tâm vào chiến lược ngẫu nhiên (Randomize Strategy) để làm giảm độ phức tạp của tối ưu hóa nhưng không đảm bảo tìm được hoạch định tốt nhất. Không như đơn định các chiến lược ngẫu nhiên hóa cho phép thể tối ưu hóa đánh đổi thời gian thời gian tối ưu hóa với thời gian thực thi. Chiến lược ngẫu nhiên hóa tìm kiếm lời giải tối ưu xung quanh một số đặc điểm nào đó, không đảm bảo thu được lời giải tốt nhất nhưng tránh được chi phí quá cao tính theo sự sử dụng bộ nhớ và thời gian. Trước hết, một hoạch định được xây dựng bắt đầu bằng việc xây dựng chiến lược thiên cận (Neighbor) của nó. Một lân cận thu được bằng cách áp dụng một biến đổi ngẫu nhiên cho một hoạch định. Ví dụ một biến đổi ngẫu nhiên của hoạch định, chứng tỏ rằng các chiến lược ngẫu nhiên hóa có hiệu năng tốt hơn các chiến lược đơn định khi truy vấn có chứa khá nhiều quan hệ.

Hệ quản trị cơ sở dữ liệu song song là sự mở rộng hệ quản trị cơ sở dữ liệu phân tán trên mạng máy tính song song với hai thuật toán song hành nội toán tử và song hành liên toán tử. Cách tiếp cận chia sẻ dữ liệu trên các node làm tăng lưu lượng mạng. Các thể tối ưu gồm ba thành phần: không gian tìm kiếm, mô hình chi phí, chiến lược tìm kiếm và thực hiện theo kiểu phụ trú có ống dẫn hay phụ trú lưu lại hoặc mô hình trung gian phụ thuộc vào máy tính. Với kiểu phụ trú có ống dẫn đòi hỏi phải có bộ nhớ lớn để thực hiện các hàm băm và đối với kiểu lưu lại thì hạn chế về tốc độ thực thi, vì đòi hỏi vật chất hóa các trạng thái trung gian.

Đối với truy vấn song song không gian tìm kiếm có xu hướng lớn hơn nhiều vì có nhiều phương án thực thi song song hơn. Vì thế chiến lược tìm kiếm ngẫu nhiên nói chung thực thi tốt hơn nhiều so với chiến lược đơn định.

Danh mục tài liệu tham khảo

1. Phạm Thế Quế (2004). *Giáo trình Cơ sở dữ liệu: Lý thuyết và thực hành*. Nhà xuất bản Bưu điện, Hà Nội.
2. Phạm Thế Quế (2010). *Giáo trình Cơ sở dữ liệu phân tán*. Nhà xuất bản Thông tin và Truyền thông, Hà Nội.
3. Nguyễn Bá Tường (2005). *Nhập môn cơ sở dữ liệu phân tán*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
4. Lê Tiến Vương (1996). *Nhập môn cơ sở dữ liệu quan hệ*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
5. M.Tamer Ozsü And Patrice Valduriez (1999). *Principles of Distributed Database Systems*, Second Edition. Prentice Hall Upper Saddle River.

Câu hỏi ôn tập cuối chương

1. Hiểu thế nào là cơ sở dữ liệu song song?
2. Hiểu thế nào là một hệ quản trị cơ sở dữ liệu song song?
3. Trình bày mục tiêu của xử lý song song.
4. Trình bày các ưu điểm cơ sở dữ liệu song song.
5. Trình bày bộ quản lý phiên SM (Session Manager).
6. Trình bày bộ quản lý yêu cầu RM (Request Manager)
7. Trình bày bộ quản lý dữ liệu DM (Data Manager).
8. Trình bày kiến trúc chia sẻ bộ nhớ (Shared-Memory).
9. Trình bày kiến trúc chia sẻ đĩa (Shared-Disk)
10. Trình bày kiến trúc không chia sẻ.
11. Trình bày các kiến trúc phân cấp. (Hierarchical Architectures).
12. Trình bày kỹ thuật sắp đặt dữ liệu.
13. So sánh ba chiến lược Round-robin partitioning, Hash partitioning và Range partitioning.
14. Trình bày truy vấn song song.
15. Trình bày xử lý dữ liệu song song.
16. Trình bày khái niệm tối ưu hóa truy vấn song song.
17. Trình bày không gian tìm kiếm.
18. Trình bày mô hình chi phí.
19. Trình bày chiến lược tìm kiếm (Search Strategy).

Chương 6

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ĐỐI TƯỢNG PHÂN TÁN

Chương 6 trình bày một số khái niệm cơ bản về hệ cơ sở dữ liệu đối tượng phân tán. Mô hình hướng đối tượng cung cấp một tập các chức năng, các công cụ hỗ trợ cho các ứng dụng phân tán. Quản lý, định danh đối tượng, điều chế con trỏ, di trú đối tượng, xóa đối tượng, thực thi phương thức và tác vụ quản lý chỗ dự trữ tại server. Việc lưu trữ dữ liệu đối tượng phân tán dựa trên khái niệm định danh logic và định danh vật lý. Nội dung của chương bao gồm:

- Khái niệm cơ bản về đối tượng và mô hình dữ liệu đối tượng;
- Thiết kế phân tán đối tượng;
- Các mô hình kiến trúc đối tượng phân tán;
- Quản lý đối tượng;
- Xử lý truy vấn đối tượng.

6.1. Giới thiệu

Một hệ quản trị cơ sở dữ liệu thực hiện các chức năng quản lý dữ liệu về mặt vật lý và về mặt logic. Có ba loại cơ sở dữ liệu phổ biến: cơ sở dữ liệu phân cấp, cơ sở dữ liệu quan hệ và cơ sở dữ liệu đối tượng. Tuy nhiên, phổ biến hơn là cơ sở dữ liệu đối tượng vì nó phù hợp với môi trường đa người dùng trong sự phát triển rất nhanh của môi trường mạng Internet. Các hệ quản trị cơ sở dữ liệu đối tượng phân tán xuất hiện giúp cho việc quản lý và tổ chức cơ sở dữ liệu phân tán trên mạng máy tính, ví dụ như các hệ quản trị Oracle, SQL Server, MySQL...

Với các hệ thống vừa và nhỏ cho các doanh nghiệp. SQL Server là phù hợp. Nhưng với các hệ thống thông tin trong các doanh nghiệp lớn, trong môi trường Internet, các hệ quản trị cơ sở dữ liệu như Oracle là công cụ phù hợp để quản trị cơ sở dữ liệu nhiều chủng loại, đa phương tiện, không lồ và phức tạp.

Hệ quản trị cơ sở dữ liệu đối tượng phân tán được xây dựng trên khái niệm đối tượng (Object). Một đối tượng bao gồm dữ liệu và các thao tác trên dữ liệu đó.

6.2. Khái niệm cơ bản về đối tượng và mô hình dữ liệu đối tượng

Hệ quản trị cơ sở dữ liệu đối tượng là một hệ thống sử dụng “đối tượng” làm đơn vị cơ bản để mô hình hóa và truy xuất dữ liệu. Nói chung không có một

mô hình đối tượng nào được thừa nhận rộng rãi và được đặc tả một cách hình thức như trong hệ quản trị cơ sở dữ liệu. Phần lớn các đặc tả của mô hình có một số đặc trưng chung, nhưng trong mỗi mô hình ngữ nghĩa có các đặc trưng này khác nhau. Một số đặc tả mô hình ngữ nghĩa có các đặc trưng này khác nhau.

Một số đặc tả mô hình đối tượng chuẩn xuất hiện như một phần của các chuẩn ngôn ngữ, tuy nhiên nó không được thừa nhận rộng rãi. Trước tiên, hãy xem xét khái niệm đối tượng trong hệ quản trị cơ sở dữ liệu đối tượng phân tán.

Đối tượng biểu diễn một thực thể xác định trong hệ thống đang được mô hình hóa. Đơn giản nhất nó được biểu diễn bằng một cặp (OID, trạng thái), trong đó OID là một định danh đối tượng (Object Identifier) và trạng thái tương ứng là một biểu diễn nào đó cho trạng thái hiện tại của đối tượng. Định danh đối tượng là một tính chất bất biến của đối tượng, có thể phân biệt các đối tượng với nhau về mặt logic và vật lý, không phụ thuộc vào trạng thái của nó. Điều này cho phép dùng chung đối tượng bằng các tham chiếu và là cơ sở để hỗ trợ như các cấu trúc hợp phần và cấu trúc phức. Trong một số mô hình, sự bằng nhau của OID là một so sánh nguyên thủy duy nhất còn với một số mô hình khác, người định nghĩa kiểu sẽ đặc tả ngữ nghĩa của phép so sánh. Theo dõi hai đối tượng được xem là đồng nhất (Identical) nếu chúng có cùng OID và bằng nhau (Equal) nếu chúng có cùng OID và trạng thái.

Trạng thái (State) của một số mô hình đối tượng được định nghĩa bằng một giá trị nguyên tử hoặc một giá trị được xây dựng, Gọi D là tập các miền do hệ thống định nghĩa và miền của các kiểu dữ liệu trừu tượng ADT (Abstract Data Type) do người sử dụng định nghĩa. Gọi I là miền các định danh đặt tên các đối tượng, và gọi A là miền tên các thuộc tính. Mỗi giá trị sẽ được định nghĩa như sau:

- Một phần tử của D là một giá trị và được gọi là trị nguyên tử (Atomic Value).
- $[a_1:v_1, \dots, a_n:v_n]$ được gọi là trị bộ (Tuple Value), trong đó a_i là một phần tử của A và v_i là một giá trị hoặc là một phần tử của I. $[]$ được gọi là toán tử đồng bộ (Tuple Construction).
- $\{v_1, \dots, v_n\}$ được gọi là trị tập hợp hoặc trị tập (Set Value), với v_i là một giá trị hoặc là một phần tử của I. $\{\}$ được gọi là toán tử dựng tập (Set Construction).

Những mô hình này là định dạng của đối tượng như giá trị. Tập và bộ là các toán tử xây dựng dữ liệu và có vai trò chủ yếu trong xây dựng các ứng dụng cơ sở dữ liệu. Những toán tử xây dựng khác, như danh sách hoặc mảng, cũng sẽ được đưa vào để làm tăng sức mạnh của mô hình.

6.3. Thiết kế phân tán đối tượng

Trong các cơ sở dữ liệu đối tượng phân tán, bài toán thiết kế phân tán bằng cách xem xét việc phân mảnh và cấp phát trong ngữ cảnh mô hình đối tượng. Giả sử một mô hình đối tượng không phân biệt giữa lớp và kiểu. Thiết kế phân tán trong thế giới đối tượng dẫn đến những lớp và kiểu. Thiết kế phân tán trong thế giới đối tượng dẫn đến những vấn đề phức tạp mới do việc bao gói các phương thức cùng với trạng thái đối tượng. Sẽ khó khăn hơn, vì các phương thức được cài đặt theo kiểu và được dùng chung bởi tất cả các đối tượng thể hiện bởi kiểu đó. Vì thế phải xem xét có nên phân mảnh trên các thuộc tính và nhân các phương thức ra cho mỗi mảnh hay phân mảnh luôn cả các phương thức. Vấn đề cũng tương tự đối với vị trí các đối tượng tương ứng với kiểu của các thuộc tính. Ngoài ra, miền các thuộc tính có thể là lớp khác. Phân mảnh các lớp ứng với mỗi thuộc tính có thể ảnh hưởng đến những lớp khác. Cuối cùng nếu phân mảnh được thực hiện ứng với các phương thức, cần phân biệt giữa các phương thức đơn giản và phương thức phức. Các phương thức đơn giản là các phương thức không kích hoạt các phương thức khác.

Tương tự như trong cơ sở dữ liệu quan hệ, có ba kiểu phân mảnh cơ bản: phân mảnh ngang, phân mảnh dọc và phân mảnh lai. Ngoài ra, có thể định nghĩa phân hoạch ngang dẫn xuất (Derived Horizontal Partitioning), phân hoạch ngang kết hợp (Associated Horizontal Partitioning), phân hoạch đường dẫn (Path Partitioning). Phân hoạch ngang dẫn xuất có ý nghĩa tương tự như trong cơ sở dữ liệu quan hệ. Phân hoạch ngang kết hợp tương tự phân hoạch ngang dẫn xuất ngoại trừ không có “Mệnh đề vị từ” ràng buộc các thể hiện đối tượng. Để cho đơn giản, giả sử rằng mô hình đối tượng dựa trên lớp và không phân biệt giữa kiểu và lớp.

6.3.1. Phân hoạch ngang lớp

Có nhiều điểm giống nhau giữa phân mảnh ngang trong CSDL đối tượng và trong CSDL Quan hệ. Phân mảnh ngang nguyên thủy trong trường hợp cơ sở dữ liệu đối tượng như trong trường hợp quan hệ. Tuy nhiên, phân mảnh dẫn xuất

có nhiều khác biệt. Trong cơ sở dữ liệu đối tượng, phân mảnh ngang dẫn xuất có thể xảy ra một số vấn đề như sau:

1. Khi phân mảnh một lớp chuyên biệt hơn, sẽ phải phân mảnh các lớp con của nó. Vì thế, kết quả của phân mảnh phải được phản ánh trong trường hợp tổng quát hơn. Phân mảnh theo một lớp con có thể gây xung đột với phân mảnh theo những lớp con khác, vì vậy có thể bắt đầu phân mảnh lớp chuyên biệt nhất và di chuyển dần lên trong lớp, phản ánh tác dụng của nó trên những lớp cha.

2. Phân mảnh của một thuộc tính phức hợp có thể ảnh hưởng đến việc phân mảnh lớp chứa nó.

3. Phân mảnh của một lớp dựa trên một dãy kích hoạt phương thức từ một lớp đến một lớp khác có thể cần phản ánh trong thiết kế.

Ví dụ 6.1: Xét định nghĩa lớp Engine

Class Engine as Object

Attributes

No_cylinder: Integer

Capacity: Real

Horsepower: Integer

Trong định nghĩa Engine ở trên, tất cả các thuộc tính đều là giản đơn. Xét các vị từ phân hoạch sau:

P_1 : horsepower ≤ 150

P_2 : horsepower > 150

Khi đó, Engine phân mảnh thành hai lớp Engine 1 và Engine2, kế thừa tất cả các tính chất của lớp Engine, được định nghĩa là một lớp trừu tượng. Các đối tượng của lớp Engine 1 và Engine 2 dựa trên các giá trị của thuộc tính horsepower (mã lực).

Phân mảnh ngang nguyên thủy các lớp được áp dụng cho tất cả các lớp cần phải phân mảnh trong hệ thống. Quá trình sẽ thu được các lược đồ phân mảnh cho mỗi lớp. Tuy nhiên, các lược đồ này không phản ánh tác dụng của phân mảnh dẫn xuất như kết quả của phân mảnh lớp con (giống như ví dụ 6.1). Vì thế, tiếp theo cần tạo ra một tập các mảnh dẫn xuất cho mỗi lớp cha bằng cách sử dụng tập vị từ ở bước trước đó. Điều này chỉ đòi hỏi phải lan truyền các quyết định phân mảnh của các lớp con đến các lớp cha. Thành phần của bước

này là tập các mảnh nguyên thủy được tạo ra trong bước hai và tập các mảnh dẫn xuất từ bước ba.

Các mảnh ngang của một lớp được cấu tạo bởi các đối tượng được truy xuất bởi các ứng dụng chỉ chạy trên một lớp và những ứng dụng chạy trên các lớp con của nó. Vì vậy, phải xác định mảnh nguyên thủy thích hợp nhất để trộn với mỗi mảnh dẫn xuất của mỗi lớp. Có thể dùng nhiều Heuristic đơn giản, chẳng hạn chọn mảnh nguyên thủy lớn nhất hoặc nhỏ nhất, hoặc mảnh nguyên thủy gói chồng nhiều nhất với mảnh dẫn xuất. Những Heuristic này dù đơn giản và trực quan, cũng chỉ bao hàm các thông tin định lượng về cơ sở dữ liệu đối tượng phân tán. Phương pháp pháp triển dựa trên số đo lực hút giữa các mảnh cho kết quả là các mảnh được khi kết nối lại với các mảnh có lực hút với chúng cao nhất.

Xét phân mảnh ngang một lớp với các biến thể hiện dựa trên đối tượng, nghĩa là miền biến thiên của một số biến thể hiện là một lớp khác, với các phương thức đơn giản. Trong trường hợp này, mỗi liên hệ hợp phần (Composition Relationship) giữa các lớp đã có tác dụng. Theo một nghĩa nào đó, mỗi liên hệ hợp phần thiết lập mỗi liên hệ chủ nhân – thành viên. Nếu lớp C1 có một thuộc tính A1 với miền biến thiên của nó là C2, thì C1 là chủ nhân và C2 là thành viên. Vì thế phân mảnh C1 tuân theo nguyên tắc như phân mảnh ngang dẫn xuất.

Với các phương thức đơn giản, phân mảnh ứng với các thuộc tính. Khi xét các phương thức cần phải cẩn trọng hơn. Ví dụ xét trường hợp tất cả các thuộc tính đơn giản nhưng các phương thức lại phức tạp, khi đó việc phân mảnh có thể dựa trên thuộc tính đơn giản để thực hiện như mô tả ở trên. Tuy nhiên, với các phương thức, điều cần thiết vào lúc biên dịch là cần xác định đối tượng được truy xuất bởi một kích hoạt phương thức. Điều này, có thể thực hiện với những phân tích tĩnh. Rõ ràng, hiệu năng tối ưu sẽ có được nếu các phương thức cần kích hoạt chung với phương thức kích hoạt trong cùng một mảnh. Tối ưu hóa đòi hỏi phải đặt các đối tượng cần truy xuất cục bộ liên quan với nhau và giảm thiểu các truy xuất cục bộ không liên quan đến nhau.

Trong trường hợp một lớp có các thuộc tính phức và các phương thức và các phương thức phức, mỗi liên hệ kiểu con, mỗi liên hệ hợp phần và mỗi liên hệ các kích hoạt phương thức cần phải được xem xét. Vì thế phương pháp phân

mảnh là hợp các phương pháp trên. Xét duyệt qua các lớp nhiều lần, sinh ra một số mảnh, rồi dùng phương pháp dựa trên lực hút trộn chúng lại.

6.3.2. Phân hoạch dọc lớp

Phân hoạch dọc một lớp C cho trước thành các mảnh C_1, \dots, C_m , sinh ra một số lớp, mỗi lớp chứa một số thuộc tính và một số phương thức. Mỗi mảnh sẽ được định nghĩa ít hơn so với lớp ban đầu. Cần giải quyết các vấn đề mối liên hệ của các lớp phân mảnh với nhau và vị trí các phương thức. Nếu tất cả các phương thức đều đơn giản thì chúng có thể được phân hoạch dễ dàng. Ngược lại, nếu chúng phức tạp thì vị trí của chúng trở thành một vấn đề cần giải quyết.

6.3.3. Phân hoạch đường dẫn

Đồ thị hợp phần là một biểu diễn các đối tượng hợp phần. Nhiều ứng dụng cần truy xuất toàn bộ đối tượng hợp phần. Phân hoạch đường dẫn (Path Partitioning) là một khái niệm mô tả việc làm từ tất cả các đối tượng tạo ra một đối tượng hợp phần vào một phân hoạch. Một phân hoạch đường dẫn bao gồm việc nhóm các đối tượng thuộc tất cả các lớp tương ứng với tất cả các biến thể hiện trong cây con có gốc tại đối tượng hợp phần đó. Một phân hoạch đường dẫn có thể được biểu diễn bằng một phân cấp với các node tạo thành một chỉ mục cấu trúc. Mỗi node của chỉ mục trỏ đến các đối tượng thuộc lớp miền của đối tượng thành phần. Vì thế, chỉ mục chứa các tham chiếu đến tất cả các đối tượng thành phần của một đối tượng hợp phần. Các thể hiện của chỉ mục cấu trúc là một tập các OID chỉ đến tất cả các đối tượng thành phần của một lớp hợp phần. Chỉ mục cấu trúc là một cấu trúc cắt ngang lược đồ cơ sở dữ liệu đối tượng, nó nhóm tất cả các OID của các đối tượng thành phần của một đối tượng hợp phần vào một lớp chỉ mục cấu trúc.

6.3.4. Các thuật toán phân hoạch

Mục đích của phân hoạch lớp là cải thiện hiệu năng các câu truy vấn và ứng dụng bằng cách làm giảm các truy xuất dữ liệu không cần thiết. Phân hoạch lớp là kỹ thuật thiết kế cơ sở dữ liệu logic, trong đó cấu trúc lại lược đồ cơ sở dữ liệu đối tượng dựa trên ngữ nghĩa ứng dụng. Phân hoạch lớp là một bài toán NP – complete. Các thuật toán phân hoạch lớp dựa trên cách tiếp cận lực hút và chi phí.

a) Phương pháp dựa vào lực hút

Lực hút giữa các thuộc tính được sử dụng để phân mảnh dọc các quan hệ. Tương tự, lực hút giữa các biến thể hiện và phương thức, lực hút giữa các

phương thức có thể được sử dụng để phân hoạch ngang và dọc lớp. Các thuật toán phân hoạch ngang và dọc lớp dựa trên việc phân loại các biến thể hiện và phương thức theo kiểu đơn giản hoặc phức tạp. Một thể hiện phức là biến thể hiện dựa trên đối tượng và là bộ phận của cây phân cấp hợp phân lớp. Mở rộng thể ước lượng phân hoạch (Partition Evaluator) của các cơ quan dữ liệu quan hệ cho các cơ sở dữ liệu hướng đối tượng và thuật toán sử dụng cách liệt kê vét cạn, phát triển thành một phương pháp thu được lược đồ phân hoạch. Áp dụng ngữ nghĩa phương thức và sinh các mảnh làm các phương thức phù hợp với các yêu cầu dữ liệu.

b) Phương pháp dựa theo chi phí

Các tiếp cận dựa trên lực hút cung cấp các lược đồ phân hoạch một cách trực quan, người ta chứng minh rằng các lược đồ phân hoạch không phải lúc nào cũng có thể giảm số lần truy xuất đĩa khi xử lý một tập các ứng dụng. Một mô hình chi phí theo số lượng truy xuất đĩa xử lý các truy vấn lẫn phương thức trên một cơ sở dữ liệu hướng đối tượng. Hơn nữa một Heuristic “leo đồi” sử dụng phương pháp lực hút cho giải pháp khởi đầu và phương pháp chi phí, để điều chỉnh cũng đã được đề xuất. Phát triển các phân cấp chỉ mục nối cấu trúc cho các truy xuất đối tượng phức hợp, và tính hiệu quả của nó đối với phương pháp duyệt trở và những phương pháp khác, như phân cấp chỉ mục nối cấu trúc, quan hệ đa chỉ mục và hỗ trợ truy xuất. Mỗi phân cấp chỉ mục nối và một dạng vật chất hóa của phân mảnh đường dẫn, tạo dễ dàng cho việc truy xuất trực tiếp đến một đối tượng phức hợp và các đối tượng thành phần của nó.

6.3.5. Cấp phát

Cấp phát dữ liệu trong cơ sở dữ liệu đối tượng gồm cấp phát phương thức và cấp phát lớp. Bài toán cấp phát phương thức quan chặt chẽ đến bài toán cấp phát lớp do vấn đề bao gói. Vì thế cấp phát các lớp kéo theo cấp phát phương thức các lớp chủ tương ứng của chúng. Nhưng vì các ứng dụng trên các cơ sở dữ liệu hướng đối tượng có kích hoạt phương thức, việc cấp phát các phương thức ảnh hưởng đến hiệu năng các ứng dụng. Tuy nhiên việc cấp phát các phương thức cần truy xuất nhiều lớp nằm ở những vị trí khác nhau lại là một bài toán chưa được giải quyết. Có bốn lựa chọn được đưa ra:

1. *Hành vi cục bộ - đối tượng cục bộ*: Là trường hợp tầm thường, tạo ra trường hợp cơ bản. Hành vi, đối tượng mà nó áp dụng và các đối tượng, tất cả

đều ở chung một vị trí. Vì thế không cần đến một cơ chế đặc biệt xử lý trường hợp này.

2. *Hành vi cục bộ - đối tượng xa*: Đây là một trong những trường hợp hành vi và đối tượng mà nó áp dụng nằm tại các vị trí khác nhau. Một giải pháp di chuyển đối tượng ở xa về vị trí có chứa hành vi. Giải pháp thứ hai là di chuyển cài đặt hành vi về vị trí của đối tượng.

3. *Hành vi ở xa - đối tượng cục bộ*: Đây là trường hợp đảo ngược của (2).

4. *Hành vi ở xa - đối tượng xa*: Đây là trường hợp ngược lại của (1).

Các thuật toán lực hút cấp phát tĩnh các mảnh sử dụng kỹ thuật phân hoạch đồ thị không áp dụng để cấp phát phương thức và không xét đến sự phụ thuộc qua lại giữa các phương thức và lớp. Vấn đề này được xem xét bằng một giải pháp cho việc cấp phát phương thức và lớp.

6.3.6. Nhân bản

Nhân bản gây một khó khăn mới cho bài toán thiết kế. Đối tượng, lớp các đối tượng hoặc tập thể đối tượng đều có thể là các đơn vị nhân bản. Quyết định ít nhất cũng phụ thuộc một phần vào mô hình đối tượng. Đặc tả kiểu nào cũng được đặt tại mỗi vị trí hay không đều có thể được xem là một bài toán nhân bản.

6.4. Các mô hình kiến trúc đối tượng phân tán

Một phương pháp phát triển hệ phân tán là phương pháp khách/chủ (Client/Server). Phần lớn các hệ cơ sở dữ liệu đối tượng hiện nay là hệ khách/chủ. Khi xem xét các mô hình kiến trúc đối tượng phân tán, cần lưu ý những đặc điểm sau:

- Dữ liệu và thủ tục được bao gói thành các đối tượng. Đơn vị dữ liệu trao đổi giữa máy khách và máy chủ có thể là một trang, một hoặc một nhóm đối tượng.

- Đối tượng không phải là những dữ liệu thụ động, mà có liên quan đến các vị trí, nơi mà các phương thức của đối tượng được thực thi.

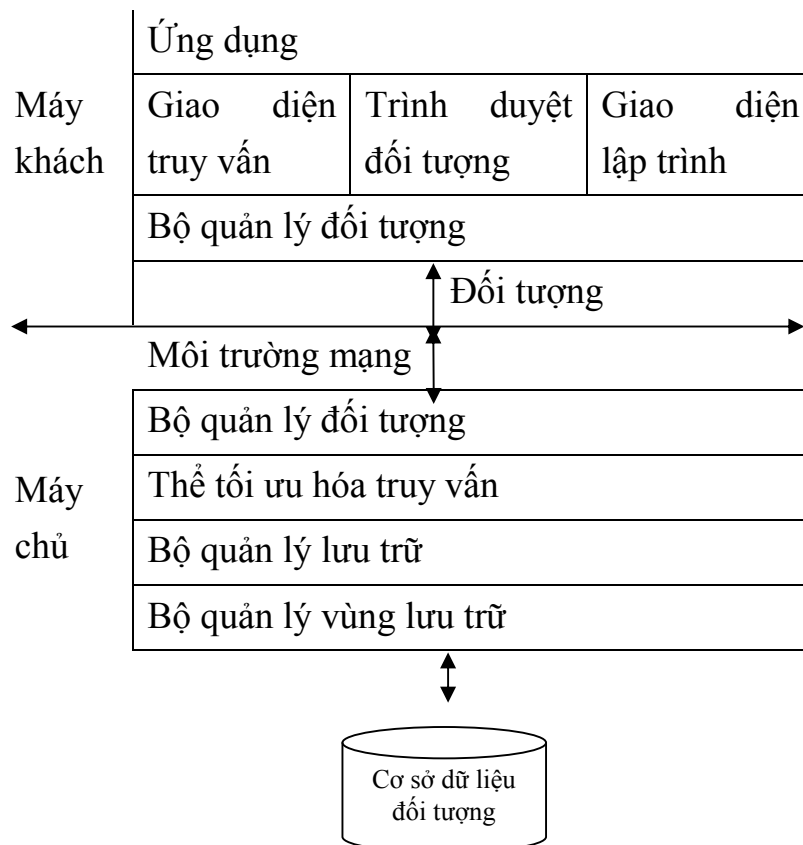
- Trong mô hình khách/chủ, máy khách gửi các truy vấn đến máy chủ tiếp nhận và thực hiện rồi trả lời kết quả về cho máy khách, được gọi là chuyển gửi chức năng (Function Shipping). Trong các hệ quản trị cơ sở dữ liệu DBMS khách/chủ đối tượng, việc duyệt qua các cấu trúc đối tượng phức hoặc hợp phần của chương trình ứng dụng, dữ liệu phải được chuyển cho máy khách. Phần lớn

các DBMS đối tượng thương mại đều sử dụng phương pháp khóa chốt để điều khiển đồng thời. Vấn đề kiến trúc căn bản là việc đặt các khóa chốt và có nên trữ các khóa chốt cho máy khách hay không.

- Đối tượng có thể thuộc loại hợp phần hoặc phức, có khả năng gửi các đối tượng thành phần, Các hệ khách/chủ quan hệ thường không phải gửi trước dữ liệu.

6.4.1. Các kiểu kiến trúc máy khách/chủ

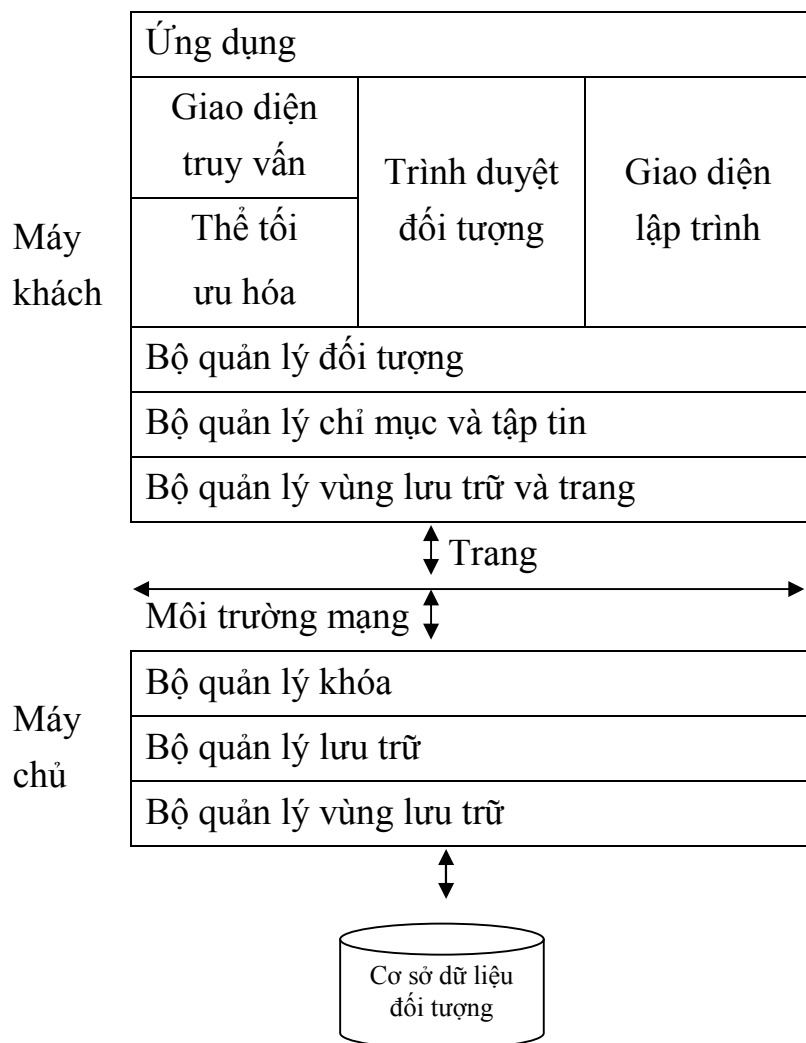
Có 2 kiểu kiến trúc máy khách/chủ: Máy chủ đối tượng (Object Server) và máy chủ trang (Page Server). Phân biệt hai kiểu kiến trúc dựa trên độ mịn (Granularity) của dữ liệu được chuyển đi giữa máy máy khách và máy chủ các chức năng được cung cấp cho máy khách và máy chủ.



Hình 6.1. Kiến trúc máy khách/ chủ đối tượng

Máy khách yêu cầu máy chủ cung cấp các “đối tượng” và máy chủ truy tìm từ cơ sở dữ liệu rồi trả chúng về cho máy khách. Những hệ thống này được gọi là máy chủ đối tượng. Trong các máy chủ đối tượng máy chủ thực hiện phần lớn các dịch vụ của hệ quản trị cơ sở dữ liệu DBMS, máy khách về cơ bản chỉ cung cấp môi trường thực thi cho các ứng dụng và một mức độ nào đó về chức

năng quản lý đối tượng. Tầng quản lý đối tượng được nhân đôi ở cả bên máy khách và bên máy chủ, cho phép cả hai có thể thực hiện các chức năng đối tượng. Bộ quản lý đối tượng cung cấp một số chức năng, trước hết nó cung cấp một ngữ cảnh để thực thi các phương thức. Nhân bản bộ quản lý đối tượng ở cả bên máy chủ và bên máy khách cho phép các phương thức có thể chạy được ở cả hai bên. Thực thi các phương thức ở bên máy khách có thể kích hoạt thực thi các phương thức không được chuyển đến máy chủ cùng các đối tượng. Tối ưu hóa việc thực thi là bài toán quan trọng. Bộ quản lý đối tượng cũng giải quyết với việc cài đặt các định danh đối tượng (logic, vật lý hoặc ảo) và việc xóa các đối tượng (xóa thực sự hoặc dọn rác). Ở trên máy chủ, nó còn hỗ trợ việc làm tự đối tượng và truy xuất phương thức. Cuối cùng các bộ quản lý đối tượng bên máy khách và bên máy chủ cài đặt một vùng trữ đối tượng (Cache) ngoài vùng lưu trữ trang ở máy chủ.



Hình 6.2. Kiến trúc máy chủ trang đối tượng

6.4.2. Lưu trữ đối tượng phân tán

Việc lưu trữ đối tượng phân tán có hai vấn đề: Làm tụ đối tượng và dọn rác phân tán. Các đối tượng và hợp phần cung cấp nhiều phương pháp làm tụ dữ liệu trên đĩa, sao cho chi phí xuất nhập đĩa cần vào việc truy xuất chúng được giảm thiểu. Dọn rác là bài toán trong các cơ sở dữ liệu đối tượng, cho phép chia sẻ dữ liệu trên tham chiếu. Vì vậy việc xóa đối tượng và tái sử dụng đòi hỏi phải được chú ý đặc biệt.

a) Làm tụ đối tượng

Mô hình đối tượng cung cấp tính độc lập dữ liệu vật lý ở mức độ cao. Ảnh xạ mô hình khái niệm thành mô hình lưu trữ vật lý là bài toán kinh điển của cơ sở dữ liệu. Có 2 loại mối liên hệ giữa các kiểu: mối liên hệ sinh kiểu con và mối liên hệ hợp phần, bằng cách cung cấp một cách thức tốt đến các đối tượng.

Làm tụ đối tượng nghĩa là nhóm các đối tượng trong các vật chứa vật lý, theo các tính chất chung chẳng hạn theo giá trị giống nhau của một thuộc tính hoặc theo các đối tượng con của cùng một đối tượng. Có thể truy xuất nhanh đến các đối tượng được làm tụ.

Một đồ thị lớp có ba mô hình lưu trữ cơ bản làm tụ đối tượng:

1. Mô hình lưu trữ phân rã DSM (Decomposition Storage Model): Phân hoạch lớp đối tượng vào các quan hệ hai ngôi (OID, attribute) dựa trên định danh logic. Mô hình loại này đơn giản.

2. Mô hình lưu trữ chuẩn tắc NSM (Normalize Storage Model): Lưu mỗi lớp như một quan hệ riêng biệt. Có thể sử dụng với OID logic và vật lý. Tuy nhiên nó chỉ có định danh logic cho phép phân hoạch dọc các đối tượng theo mối liên hệ kế thừa.

3. Mô hình lưu trữ trực tiếp (Direct Storage Model) cho phép làm tụ nhiều lớp đối tượng phức dựa trên mối quan hệ hợp phần. Mô hình này tổng quát hóa các kỹ thuật của các cơ sở dữ liệu phân cấp và mạng và các hoạt động tốt với định danh vật lý. Rất ưu việt, khi các kiểu mẫu truy xuất đã được xác định.

Mô hình lưu trữ phân rã và mô hình lưu trữ chuẩn tắc đều đơn giản khi sử dụng phân mảnh ngang. DSM cung cấp tính linh hoạt, và hiệu năng khi sử dụng một bộ nhớ và vùng trữ lớn. Cài đặt mô hình lưu trữ trực tiếp trong một kiến trúc lưu trữ đơn cấp phân tán, trong đó mỗi đối tượng có một định danh vật lý có

giá trị đối với toàn hệ thống. Cơ chế nhóm Eos dựa trên khái niệm các đường nối hợp phần có liên quan nhất và giải quyết được bài toán về đối tượng dùng chung có nhiều cha. Khi một đối tượng di chuyển đến một node khác, nó nhận một định danh mới. Để tránh sự giám định các đối tượng đã di chuyển, các tham chiếu đến đối tượng được thay đổi trong quá trình dọn rác mà không mất thêm chi phí. Cơ chế nhóm được thực hiện theo phương thức động để đạt được sự cân bằng tải và thích ứng được với những thay đổi của đồ thị đối tượng.

b) Dọn rác phân tán

Các đối tượng có thể tham chiếu đến nhau qua định danh đối tượng. Khi các chương trình sửa đổi và xóa bỏ các tham chiếu, một đối tượng trường tồn khi mà không còn đối tượng nào tham chiếu đến nó, đối tượng đó được gọi là “Rác” và cần được thu hồi bởi thể dọn rác. Trong các DMBS quan hệ, việc cập nhật liên tục cũng được xem như là hình thái đơn giản của dọn rác “thủ công”. Trong ngữ cảnh các hệ điều hành hoặc ngữ cảnh ngôn ngữ lập trình gây ra lỗi do đó cần phải có cơ chế dọn rác phân tán tự động trong các hệ đối tượng phân tán. Các thuật toán dọn rác cơ bản được phân chia thành loại đếm tham chiếu và loại dò vết.

6.5. Quản lý đối tượng

6.5.1. Quản lý định danh đối tượng

Định danh đối tượng được hệ thống phát sinh và được dùng để xác định một cách duy nhất một đối tượng (ngắn hạn hay trường tồn do hệ thống tạo hay người dùng tạo) trong hệ thống việc cài đặt các đối tượng trường tồn nói chung khác hẳn so với cài đặt các đối tượng ngắn hạn.

a) Định danh cho các đối tượng trường tồn

Có hai giải pháp dựa trên định danh logic và định danh vật lý:

- Phương pháp dùng định danh vật lý và phương pháp định danh đối tượng bằng địa chỉ của đối tượng, có thể là địa chỉ trang và một địa chỉ offset. Ưu điểm có thể nhận ra đối tượng một cách các đối tượng cha và chỉ mục đều phải được cập nhật mỗi khi một đối tượng được di chuyển đến một trang khác.

- Phương pháp dùng định danh logic: Gồm có cấp phát một định danh đối tượng (OID) duy nhất cho mỗi đối tượng trên toàn bộ hệ thống. Vì OID là bất biến nên không phải trả chi phí nào khi di chuyển đối tượng.

Nói chung cả hai phương pháp có thể xem giá trị đối tượng như định danh của chúng. Các hệ cơ sở dữ liệu hướng đối tượng dùng phương pháp định danh logic và nó hỗ trợ các môi trường động tốt hơn.

b) Định danh cho các đối tượng ngắn hạn

- Định danh vật lý: Có thể là địa chỉ ảo hay thực của đối tượng, tùy thuộc vào việc bộ nhớ ảo có được cung cấp hay không. Phương pháp này có hiệu quả nhất, nhưng không cho phép di chuyển các đối tượng.
- Định danh logic: Xử lý các đối tượng thông qua bảng giám định có tính cục bộ đối với sự thực thi của chương trình. Bấy nay liên kết một định danh logic, được coi là con trỏ hướng đối tượng OPP (Object Oriented Pointer) trong Smalltalk, chỉ đến định danh vật lý của đối tượng. Việc di chuyển đối tượng được phép, nhưng phải mất một bước tìm kiếm cho mỗi truy xuất đối tượng.

6.5.2. Quản lý con trỏ

Trong các hệ quản trị cơ sở dữ liệu đối tượng, có thể duyệt một đối tượng này đến một đối tượng khác bằng cách dùng các biểu thức đường dẫn có chứa các thuộc tính với các giá trị của chúng dựa trên đối tượng. Về cơ bản chúng đều là con trỏ. Thông thường trên đĩa định danh đối tượng được dùng để biểu diễn con trỏ này. Tuy nhiên, trong bộ nhớ, dùng con trỏ nội nhớ để duyệt từ đối tượng này đến đối tượng khác. Quá trình chuyển con trỏ trên đĩa thành con trỏ nội nhớ được gọi là “điều chế con trỏ”. Các lược đồ trên phần cứng và dựa trên phần mềm là hai loại cơ chế điều chế con trỏ.

- Lược đồ phần cứng: Cơ chế khuyết trang của hệ điều hành được dùng khi một trang được đưa vào bộ nhớ. Tất cả các con trỏ trong đó đều được điều chế và chúng chỉ đến các khung nhớ ảo dành riêng. Các trang dữ liệu tương ứng với các khung ảo này chỉ được tải vào bộ nhớ khi có truy xuất đến chúng.
- Lược đồ mềm: Một bảng đối tượng được dùng cho mục đích điều chế con trỏ. Nghĩa là một con trỏ được điều chế chỉ đến một vị trí trong bảng tương đối. Có các biến thể “hẫng hái” và “trễ nải” của các lược đồ mềm, tùy thuộc vào con trỏ cần được điều chế. Vì thế mỗi truy xuất đối tượng đều có một mức độ giám định đi kèm với nó.

Ưu điểm của lược đồ phần cứng là nó tốt hơn khi phải duyệt lặp đi lặp lại một cây phân cấp đối tượng vì không có sự hiện diện của một mức độ giám định

cho mỗi truy xuất đối tượng. Tuy nhiên khi chỉ có một ít các đối tượng cần truy xuất mỗi trang, chi phí cao cho cơ chế khuyết trang làm cho lược đồ phân cứng không tối ưu.

6.5.3. Di trú đối tượng

Trong hệ phân tán theo thời gian, các đối tượng cần di chuyển giữa các vị trí, vì vậy nảy sinh ra một số vấn đề về đơn vị di trú. Trong các hệ thống có trạng thái tách rời phương thức, có thể di chuyển trạng thái của đối tượng mà không di chuyển phương thức. Tương ứng của tình huống này trong các hệ chỉ thuần phân mảnh một đối tượng theo các hành vi của nó. Nếu đối tượng là đơn vị di trú, việc tái định vị trí của chúng có thể đưa chúng xa khỏi các đặc tả kiểu và người ta quyết định xem các kiểu của nó có được nhân bản tại mỗi vị trí có chứa các thể hiện hay phải truy xuất kiểu từ xa khi các hành vi hoặc phương thức được áp dụng cho các đối tượng. Có 3 khả năng được xem xét khi di trú các lớp (kiểu):

- Mã nguồn được di chuyển và được biên dịch lại tại vị trí đích.
- Phiên bản đã biên dịch của một lớp được di trú như các đối tượng khác.
- Mã nguồn của định nghĩa lớp được di chuyển nhưng không di chuyển các thao tác đã được biên dịch của nó mà sẽ sử dụng chiến lược di chuyển muộn.

Việc di chuyển đối tượng phải được theo vết để có thể có thể tìm ra những chỗ mới của chúng. Một cách thường dùng để theo vết là để lại các đại diện hoặc các đối tượng ủy nhiệm. Đây là những đối tượng giữ chỗ được để lại tại vị trí cũ của đối tượng và chỉ đến vị trí mới. Truy xuất đến các đối tượng ủy nhiệm được hệ thống chuyển hướng một cách vô hình đến đúng các đối tượng tại vị trí mới.

Di trú đối tượng có thể thực hiện trên các trạng thái hiện hành của chúng. Đối tượng có thể ở một trong 4 trạng thái sau:

- Sẵn sàng: Các đối tượng sẵn sàng hiện không được kích hoạt, hoặc chưa nhận được thông báo, nhưng sẵn sàng được kích hoạt khi nhận được một thông báo.
- Đang hoạt (Active): Các đối tượng đang hoạt có mặt trong một hoạt động đáp lại một kích hoạt hoặc một thông báo.
- Chờ (Waiting): Đối tượng chờ đã được kích hoạt, hoặc đã gửi một thông báo cho một đối tượng khác và đang chờ trả lời.

- Treo (Suspended): Các đối tượng treo, tạm ngừng hiện tại, tạm thời không sẵn sàng đối với kích hoạt.

Các đối tượng đang trong trạng thái đương hoạt hoặc chờ không được di trú vì hoạt động mà chúng đang tham gia sẽ bị phá vỡ. Việc di trú bao gồm các bước:

1. Chuyển tải đối tượng từ nguồn đến đích và
2. Tạo ra một ủy nhiệm tại nguồn để thể hiện đối tượng ban đầu.

Các vấn đề liên quan đến việc bảo trì thư mục hệ thống.

Vấn đề 1: Khi đối tượng di chuyển thư mục hệ thống phải được cập nhật để phản ánh vị trí mới. Điều này được thực hiện muộn khi một đối tượng đại diện hoặc một ủy nhiệm định hướng lại một kích hoạt vào lúc di chuyển.

Vấn đề 2: Trong một môi trường năng động, nơi mà các đối tượng di chuyển thường xuyên, xâu đại diện hoặc xâu ủy nhiệm có thể rất dài. Hệ thống khi đó cần “cô đặc” những xâu này lại theo thời gian. Tuy nhiên, kết quả “cô đặc” phải được phản ánh vào trong thư mục và nó có thể không được thực hiện muộn.

Một vấn đề quan trọng khác nảy sinh của di trú khi di chuyển đối tượng hợp phần. Chuyển tải một đối tượng hợp phần có thể phải chuyển tải cả những đối tượng được đối tượng hợp phần tham chiếu đến, giải quyết bằng cách tổng hợp đối tượng.

6.6. Xử lý truy vấn đối tượng

Các hệ quản trị cơ sở dữ liệu quan hệ được xây dựng dựa trên cơ sở toán học rất chặt chẽ và tập các đại số nguyên thủy được sử dụng phổ biến. Các hệ quản trị cơ sở dữ liệu đối tượng được xây dựng dựa trên khả năng truy vấn khai báo như một đặc trưng cơ bản của DBMS đối tượng. Phần lớn các bộ xử lý truy vấn đối tượng đều dùng các kỹ thuật tối ưu hóa của các hệ thống quan hệ. Tuy nhiên, có một số vấn đề xử lý tối ưu hóa truy vấn trong các DBMS đối tượng phức tạp hơn nhiều.

Kết quả của một phép toán đại số đối tượng có thể là một tập hợp các đối tượng. Nếu ngôn ngữ truy vấn đối tượng được đóng bên dưới các phép toán đại số, tập các đối tượng hỗn tạp có thể là toán hạng cho phép toán khác. Điều này đòi hỏi phải phát triển các lược đồ suy diễn kiểu chi tiết để xác định những phương thức nào có thể áp dụng cho tất cả các đối tượng trong một tập như thế.

Hơn nữa, các phép đại số đối tượng thường thao tác trên các kiểu tập hợp (ví dụ: set, bag, list). Chính vì vậy đòi hỏi phải xác định rõ kiểu kết quả của các phép toán trên các tập của nhiều kiểu đối tượng có các kiểu khác nhau.

Việc đóng gói các phương thức với dữ liệu mà chúng thao tác trong hệ quản trị cơ sở dữ liệu đối tượng làm nảy sinh nhiều vấn đề khó khăn. Trước hết ước lượng chi phí thực thi các phương thức khó khăn hơn nhiều so với tính toán chi phí truy xuất một thuộc tính theo một đường truy xuất. Tối ưu hóa cần phải quan tâm việc thực thi phương thức mà việc này thì không phải là một bài toán dễ vì các phương thức có thể được viết bằng một ngôn ngữ lập trình tổng quát. Thứ hai, việc đóng gói làm nảy sinh vấn đề có liên quan đến tính khả năng truy xuất thông tin bởi các thể xử lý truy vấn thông tin. Một số hệ thống giải quyết khó khăn này bằng cách xen thể tối ưu hóa truy vấn như một ứng dụng đặc biệt, có thể phá bỏ sự đóng gói và truy xuất trực tiếp các thông tin. Những hệ thống khác đề xuất một cơ chế mà qua đó các đối tượng “bộc lộ” chi phí của chúng như một phần trong giao diện của chúng.

Đối tượng có cấu trúc phức tạp, qua đó trạng thái của một đối tượng tham chiếu đến một đối tượng khác. Truy xuất các đối tượng phức tạp như thế phải chứa cả biểu thức đường dẫn. Tối ưu hóa biểu thức đường dẫn là một vấn đề quan trọng trong ngôn ngữ truy vấn đối tượng. Mặt khác, giữa các đối tượng còn có quan hệ phân cấp kế thừa. Tối ưu hóa việc truy xuất các đối tượng với xử lý truy vấn quan hệ.

Xử lý và tối ưu hóa truy vấn thông tin đối tượng đã là chủ thể của nhiều hoạt động nghiên cứu. Nhưng phần lớn các nghiên cứu này không được mở rộng cho các hệ thống đối tượng phân tán. Vì thế, những phần tiếp theo chỉ tập trung vào một số các vấn đề quan trọng: Kiến trúc bộ xử lý truy vấn thông tin đối tượng; tối ưu hóa truy vấn thông tin đối tượng và các chiến lược thực thi truy vấn.

6.6.1. Kiến trúc xử lý truy vấn đối tượng

Tối ưu hóa truy vấn đối tượng được mô hình hóa một số bài toán tối ưu mà lời giải là sự lựa chọn trạng thái tối ưu với mỗi biểu thức truy vấn đại số dựa trên một hàm chi phí trong một không gian tìm kiếm biểu diễn cho một họ các biểu thức truy vấn đại số tương đương. Về mặt kiến trúc, bộ xử lý truy vấn khác nhau ở cách thức chung mô hình hóa những thành phần này.

Nhiều thể tối ưu hóa hệ quản trị Cơ sở dữ liệu hướng đối tượng được cài

đặt như thành phần của bộ quản lý đối tượng bên trên một hệ thống lưu trữ hoặc như các mô-đun máy khách trong kiến trúc máy khách/chủ. Thẻ tối ưu hóa dựa trên quy tắc cung cấp một số khả năng mở rộng bằng cách cho phép định nghĩa những quy tắc biến đổi mới. Tuy nhiên, chúng không cho phép mở rộng theo những trục khác. Trong mục này, sẽ thảo luận một số đề xuất mới về khả năng mở rộng trong các hệ quản trị cơ sở dữ liệu hướng đối tượng.

Thẻ tối ưu hóa truy vấn Open OODB định nghĩa một bộ khung kiến trúc mở cho các hệ quản trị cơ sở dữ liệu hướng đối tượng và mô tả không gian thiết kế cho những hệ thống này. Mô-đun truy vấn là một tính năng mở rộng. Thẻ tối ưu hóa truy vấn xây dựng bằng cách dùng bộ sinh thẻ tối ưu hóa Volcano có thể mở rộng ứng với các toán tử đại số, các quy tắc biến đổi logic, các thuật toán thực thi, các quy tắc cài đặt, hàm đánh giá chi phí. Sự tách biệt giữa các cấu trúc phân tích cú pháp của ngôn ngữ truy vấn với đồ thị toán tử được thẻ tối ưu hóa truy vấn cho phép thay thế chính ngôn ngữ hay chính thẻ tối ưu hóa. Sự tách biệt giữa các toán tử đại số và các thuật toán thực thi cho phép khám phá những phương pháp cài đặt khác nhau cho các toán tử đại số. Thẻ xử lý truy vấn Open OODB gồm có một động cơ thực thi truy vấn chứa những cài đặt hiệu quả của các thao tác quét, quét chỉ mục, nối bấm lại và tổng hợp đối tượng phức hợp.

Thẻ tối ưu hóa truy vấn EPOP là một cách tiếp cận khác về vấn đề mở rộng khả năng tối ưu hóa vấn tin. Không gian tìm kiếm được.

6.6.2. Các vấn đề xử lý truy vấn đối tượng

a) Tối ưu hóa đại số

Trong cơ sở dữ liệu quan hệ, các phép toán đại số cũng tương tự như trong cơ sở dữ liệu đối tượng. Tối ưu hóa đại số dựa trên các biến đổi truy vấn dạng phép tính quan hệ. Trong quá trình này việc áp dụng các quy tắc biến đổi cho phép đơn giản hóa truy vấn nhờ loại bỏ các biểu thức con chung và các biểu thức không có hiệu lực. Các truy vấn có được từ quá trình phân rã và cục bộ hóa dữ liệu có thể cho thực hiện ở dạng thức đó bằng cách đưa thêm các giao tiếp nguyên thủy một cách có hệ thống. Tuy nhiên, việc hoán vị thứ tự các phép toán trong câu truy vấn có thể cho ra nhiều chiến lược tương đương. Việc tìm ra một cách sắp xếp tối ưu các phép toán cho một câu truy vấn chính là nhiệm vụ chủ yếu của tầng tối ưu hóa vấn tin, hoặc nói ngắn gọn là thẻ tối ưu hóa (optimizer). Tối ưu hóa đại số phải đảm bảo các phép toán thực hiện đúng, sau đó xét đến

thuật toán chọn đường để cho ra phương pháp tối ưu nhất. Có thể dựa theo đánh giá hàm heuristic. Các phép toán đại số thì cũng tương tự như trong cơ sở dữ liệu quan hệ, chẳng hạn như phép hợp, giao, chọn... Tuy nhiên, trong cơ sở dữ liệu đối tượng thì việc áp dụng các phép biến đổi tương đương để cho ra phương pháp tối ưu đòi hỏi phụ thuộc vào nhiều tham số.

Thế tối ưu hóa đại số phải đảm bảo lựa chọn làm sao để thực thi truy vấn nhanh gọn, tốn ít bộ nhớ máy tính lưu trữ. Việc tối ưu đại số phải dựa trên cả tham số về không gian tìm kiếm và các quy tắc biến đổi.

b) Không gian tìm kiếm và các quy tắc biến đổi

Tối ưu hóa truy vấn là quá trình sinh ra một hoạch định thực thi truy vấn biểu thị cho chiến lược thực thi truy vấn. Hoạch định được chọn phải hạ thấp tối đa hàm chi phí. Không gian tìm kiếm là tập các hoạch định thực thi biểu diễn cho câu truy vấn. Những hoạch định này là tương đương theo nghĩa là chúng sinh ra cùng một kết quả, nhưng khác nhau ở thứ tự thực hiện các thao tác và cách thức cài đặt các thao tác này vì thế khác nhau về hiệu năng. Không gian tìm kiếm thu được bằng cách áp dụng quy tắc biến đổi. Nếu biểu diễn các đối tượng trong biểu thức truy vấn theo cây nhị phân, trong đó các node là các biểu thức thực hiện truy vấn thì việc duyệt cây sẽ cho ra cùng một kết quả nhưng phương pháp duyệt có thể khác nhau.

Ưu điểm của tối ưu đại số là dùng các biện pháp tính giao hoán, bắc cầu, phân phối... để tìm ra biểu thức đại số có thời gian thực thi truy vấn với thời gian nhỏ nhất. Các quy tắc biến đổi phụ thuộc nhiều vào từng đại số đối tượng và cho các tổ hợp của chúng. Việc không định nghĩa một tiêu chuẩn đại số đối tượng gây khó khăn cho các nghiên cứu và các vấn đề liên quan. Các xem xét nói chung cho việc định nghĩa các quy tắc biến đổi và các thao tác các biểu thức truy vấn là khá giống với các hệ thống quan hệ, chỉ có một sự khác biệt quan trọng đó là các biểu thức truy vấn quan hệ được định nghĩa trên các quan hệ ngang bằng, trong khi đó các truy vấn đối tượng được định nghĩa bằng các lớp (hoặc tập hợp các đối tượng) mà có các quan hệ con và quan hệ kết hợp giữa chúng. Do đó, có thể sử dụng ngữ nghĩa của mối quan hệ này trong việc tùy chọn truy vấn đối tượng để có thêm những phép biến đổi khác.

Xem xét ví dụ, có 3 toán tử đại số đối tượng là phép hợp (union), ký hiệu là \cup , phép giao ký hiệu là \cap và lựa chọn theo tham số ký hiệu là $P_{\sigma_F} < Q_1, Q_2, \dots, Q_k$

>. Phép hợp và phép giao có ngữ nghĩa trên tập lý thuyết thông thường, và phép chọn sẽ chọn ra các đối tượng từ một tập P sử dụng các đối tượng Q_1, Q_2, \dots, Q_k như là các tham số. Kết quả của các phép toán này là một tập hợp các đối tượng. Sau đây là một số quy tắc biến đổi có thể được áp dụng trong suốt quá trình tối ưu hóa cho các biểu thức truy vấn tương đương. Ký hiệu Qset là Q_1, Q_2, \dots, Q_k .

$$(1) (P\sigma_{F1} < QSet >) \sigma_{F1} < RSet > \Leftrightarrow (P\sigma_{F1} < QSet >) \sigma_{F1} < QSet >$$

$$(2) (P \cup Q) \sigma_F < RSet > \Leftrightarrow (P\sigma_F < RSet >) \cup (Q\sigma_F < RSet >)$$

$$(3) (P\sigma_{F1} < QSet >) \sigma_{F2} < RSet > \Leftrightarrow (P\sigma_{F1} < QSet >) \cap (P\sigma_{F2} < RSet >)$$

Quy tắc (1) biểu hiện tính chất giao hoán của select còn quy tắc (2) biểu diễn phép select phân phối trên phép union. Quy tắc (3) là một quy tắc đồng nhất thể hiện select chỉ hạn chế đầu vào của nó và trả về một tập con trong đối số đầu tiên của nó. Do vậy, dựa vào các phép biến đổi tương đương trên, có thể áp dụng cho từng bài toán cụ thể với các phép đại số cụ thể dựa trên một số đối tượng cho biết trước.

Hai quy tắc đầu tiên khá tổng quát, trong đó chúng thể hiện cho các hệ thức tương đương kế thừa từ lý thuyết tập hợp. Quy tắc thứ ba là quy tắc biến đổi đặc biệt cho một toán tử đại số đối tượng cụ thể được định nghĩa trong một ngữ nghĩa cụ thể. Tuy nhiên, cả ba quy tắc trên đều có cú pháp giống nhau xét về bản chất. Xem xét các quy tắc biến đổi cho sau đây:

$$C_1 \cap C_2 = \emptyset \quad \text{Nếu } c_1 \neq c_2$$

$$C_1 \cup C_2^n = C_2^n \quad \text{Nếu } c_1 \text{ là lớp con của } c_2$$

$$(P\sigma_{F1} < QSet >) \cup R\sigma_{F2} \Leftrightarrow (P\sigma_{F1} < QSet >) \cap (R\sigma_{F2} < QSet >)$$

$$\Leftrightarrow P \cap (R\sigma_F < QSet >)$$

Trong các biến đổi ở trên, C_1 ký hiệu là tập hợp các đối tượng trong phạm vi lớp C_i và C_j^n là ký hiệu các đối tượng trong phạm vi sâu hơn của lớp C_j . Các quy tắc biến đổi này có bản chất ngữ nghĩa, bởi vì chúng phụ thuộc vào mô hình đối tượng và các đặc tả mô hình truy vấn. Ví dụ quy tắc đầu tiên là đúng vì mô hình đối tượng giới hạn mỗi đối tượng chỉ thuộc về một lớp. Quy tắc thứ hai đúng vì mô hình truy vấn cho phép tìm ra các đối tượng trong các mở rộng của

lớp đích. Cuối cùng quy tắc thứ ba dựa trên kiểu quy tắc đồng nhất kiểu đối với khả năng áp dụng của nó cũng như một điều kiện F' đồng nhất với F loại trừ mỗi sự kiện của p được thay thế bằng r .

Vì ở phần trước, ý tưởng của việc biến đổi truy vấn chúng ta đã xét nên sẽ không trình bày chi tiết ở đây. Các thảo luận trên đây chỉ minh họa tính tổng quát và những điểm nổi bật cần được xem xét trong các đại số đối tượng.

6.6.3. Thực thi truy vấn đối tượng

Các hệ quản trị cơ sở dữ liệu sẽ hiệu quả từ sự thích hợp đóng giữa các toán tử đại số quan hệ và các truy cập ban đầu vào hệ thống kho chứa. Do đó, tính khái quát hóa của kế hoạch thực thi đối với một biểu thức truy vấn cơ bản quan tâm tới việc lựa chọn và cài đặt của các thuật toán hiệu quả để thực hiện các toán tử đại số riêng và sự kết hợp của chúng. Trong các hệ quản trị cơ sở dữ liệu đối tượng vấn đề này phức tạp hơn mặc dù sự khác nhau ở các cấp độ của các đối tượng được định nghĩa một cách hành vi và kho chứa của chúng. Sự đóng góp của các đối tượng là cái ẩn đi trong các chi tiết cài đặt của chúng và kho chứa của các phương pháp cùng với các đối tượng thách thức một vấn đề thiết kế hấp dẫn, điều mà có thể được xác định ở trạng thái như sau: “Ở thời điểm nào khi xử lý truy vấn thì thể truy vấn nên truy cập thông tin quan tâm đến kho chứa các đối tượng?”. Do đó, kế hoạch thực thi truy vấn được sinh ra từ biểu thức truy vấn được kế hoạch vào cuối bước viết lại truy vấn bằng cách ánh xạ biểu thức truy vấn vào một tập hợp các lời gọi giao diện quản lý đối tượng được định nghĩa một cách rõ ràng. Giao diện quản lý đối tượng thực ra bao gồm một tập hợp các thuật toán thực thi. Mục này nói về vấn đề thực thi truy vấn, cần xem lại một số thuật toán thực thi có thể là các máy thực thi truy vấn đối tượng có khả năng thực thi cấp độ cao.

Một máy thực thi truy vấn yêu cầu 3 lớp thuật toán cơ bản dựa trên tập hợp các đối tượng: quét tập hợp, quét chỉ số, và so sánh tập hợp (set matching). Quét tập hợp là một tập hợp là một thuật toán truyền thẳng truy cập tuần tự tất cả các đối tượng bên trong tập hợp, khi tìm đối tượng sẽ sử dụng phương pháp quét mọi phần tử trong tập hợp. Quét chỉ số cho phép truy cập hiệu quả để lựa chọn các đối tượng trong một tập hợp thông qua một chỉ số. Mỗi đối tượng lưu trong file hoặc trong một tập hợp được đánh dấu với một chỉ số cho trước. Các chỉ số này đảm bảo tính duy nhất, có nghĩa là không được phép trùng nhau về

chỉ số mà lại có các đối tượng khác nhau. Có thể sử dụng một trường của đối tượng hoặc các giá trị trả về bằng một số cách như là một khóa học cho một chỉ số.

Thuật toán so sánh tập hợp sẽ lấy tập hợp các đối tượng làm đầu vào và đưa ra các đối tượng được tích hợp có liên quan bởi một số chuẩn mực, điều kiện nào đó. Kết nối, tập giao và tổng hợp (assembly) là các ví dụ của các thuật toán loại này. Các chỉ số đường dẫn có liên quan trực tiếp đến việc thực thi truy vấn đối tượng.

6.7. Quản lý giao dịch đối tượng phân tán

6.7.1. Các tiêu chuẩn quản lý

Một đặc trưng của mô hình dữ liệu quan hệ là không có ngữ nghĩa rõ ràng về thao tác cập nhật. Mô hình đã chỉ rõ cách truy tìm dữ liệu trong một cơ sở dữ liệu quan hệ qua các phép toán đại số quan hệ, nhưng thực sự không mô tả một hành động cập nhật có ý nghĩa gì. Hệ quả là các định nghĩa nhất quán và các kỹ thuật quản lý giao dịch không có liên hệ gì với mô hình dữ liệu. Người ta áp dụng các kỹ thuật tương tự cho các hệ quản trị cơ sở dữ liệu DBMS phi quan hệ, thậm chí cho các hệ thống lưu trữ không phải cơ sở dữ liệu.

Mô hình dữ liệu quan hệ có thể được xem là mô hình có nhiều ưu điểm về tính độc lập. Vì thế có thể được lan truyền sang các ứng dụng khác nhau. Nghiên cứu quản lý giao tác trên các DBMS đối tượng đã tận dụng sự độc lập này bằng cách áp dụng các kỹ thuật đã biết cho những cấu trúc mới. Đặc điểm của các DBMS hướng đối tượng chẳng hạn như các cấu trúc kiểu (lớp), các đối tượng hợp phần và các nhóm đối tượng cần được xem xét nhưng các kỹ thuật cơ bản là như nhau.

Trong các hệ quản trị cơ sở dữ liệu DBMS hướng đối tượng có tính quyết định là việc mô hình hóa các ngữ nghĩa cập nhật vào trong mô hình đối tượng. Các lập luận này chỉ ra rằng giao tác nên được xem như là đối tượng:

- Trong các DBMS đối tượng, dữ liệu và các thao tác trên dữ liệu, như phương thức, hành vi, các thao tác trong nhiều mô hình đối tượng khác nhau đều được lưu trữ. Các câu truy vấn cơ sở dữ liệu đối tượng tham chiếu đến những thao tác như thành phần trong các vị từ của chúng. Nói cách khác, việc thực thi các truy vấn này khởi hoạt nhiều thao tác khác nhau được định nghĩa trên các lớp. Để đảm bảo tính an toàn của các biểu thức truy vấn các phương pháp xử lý truy vấn tín hiệu hạn chế những thao tác này để không có các hiệu ứng phụ với

tác dụng ngăn chúng cập nhật cơ sở dữ liệu. Đây là hạn chế cần được giảm bớt bằng cách kết hợp ngữ nghĩa cập nhật vào các định nghĩa an toàn truy vấn.

- Các DBMS đối tượng có ảnh hưởng đến dàn kiểu, có một mối liên hệ trực tiếp giữa việc phát triển lược đồ động và việc quản lý giao tác. Nhiều kỹ thuật sử dụng khóa chốt và khóa nhiều độ mịn hạn chế mạnh khả năng hoạt động đồng thời.

- Có một số mô hình đối tượng xử lý tất cả các thực thể của hệ thống như các đối tượng. Theo cách tiếp cận này, mô hình hóa các giao tác như đối tượng. Về cơ bản giao tác là các kết cấu làm thay đổi trạng thái của cơ sở dữ liệu, có ảnh hưởng trên đối tượng. Vì vậy, giao tác là các kết cấu làm thay đổi trạng thái của cơ sở dữ liệu phải được miêu tả rõ ràng. Trong ngữ cảnh này cũng cần chú ý đến miền ứng dụng cần đến các dịch vụ của DBMA đối tượng có xu hướng đặt ra các yêu cầu quản lý giao tác hơi khác, theo cả mô hình giao tác và ràng buộc nhất quán.

Mô hình hóa giao tác như các đối tượng cho phép áp dụng các kỹ thuật đối tượng đã biết (chuyên biệt hóa và sinh kiểu con) để tạo ra các loại hệ thống quản lý giao tác khác nhau. Điều này có khả năng mở rộng hệ thống.

- Một số yêu cầu cần đến sự hỗ trợ bằng quy tắc và khả năng của các đối tượng. Về cơ bản, giao tác là các kết cấu làm thay đổi trạng thái của cơ “năng động”. Bản thân các quy tắc được thực thi như giao tác mà chúng lại sinh ra các giao tác mới. Thế nên cho rằng các quy tắc cần được mô hình hóa như đối tượng và như thế một số giao tác cũng cần phải được mô hình hóa như đối tượng.

6.7.2. Mô hình giao dịch và cấu trúc đối tượng

Trong các mục trên đã trình bày về các mô hình giao tác, từ các giao tác phẳng đến các hệ thống lưu công, nhưng trong các mục đó không xét đến độ mịn của các đối tượng cơ sở dữ liệu mà các giao dịch thao tác, đơn giản được gọi là đơn vị khóa. Trong mục này sẽ xét một số khả năng để chọn mô hình giao tác, cung cấp một không gian hai chiều để thiết kế cài đặt các hệ thống giao tác.

Theo chiều cấu trúc đối tượng, việc xác định các đối tượng đơn giản, như tập tin, trang, bản ghi... là thể hiện của kiểu dữ liệu trừu tượng ADT (Abstract DataType), các đối tượng trưởng thành (Full Fledged) và các đối tượng năng động (Active) theo độ phức tạp tăng dần. Một số hệ thống cung cấp khả năng hoạt động đồng thời ở mức bản ghi với chi phí cao và không có tính nguyên tử,

đòi hỏi sự đồng bộ hóa tại mức trang. Đặc trưng của mức này là các thao tác trên các đối tượng đơn giản không tận dụng ngữ nghĩa của các đối tượng. Khi cập nhật một trang được xem như một thao tác ghi (write) trên trang đó mà không xem đối tượng nào được lưu trên trang đó.

Từ góc độ xử lý giao tác, các kiểu dữ liệu trừu tượng ADT đặt ra một nhu cầu giải quyết với các thao tác trừu tượng, bản thân các thao tác trừu tượng dẫn đến sự gắn kết ngữ nghĩa của chúng vào trong định nghĩa về tiêu chuẩn đúng đắn. Thực thi các giao tác trên kiểu dữ liệu trừu tượng ADT có thể đòi hỏi một cơ chế nhiều mức. Trong các hệ thống như thế, từng giao tác riêng lẻ thể hiện mức trừu tượng cao nhất. Các thao tác trừu tượng được cấu tạo từ mức trừu tượng thấp hơn và được phân rã thành các thao tác read – write tại mức thấp nhất. Dù tiêu chuẩn đúng đắn là gì thì nó cũng phải được áp dụng cho từng mức.

Cần phân biệt các đối tượng là thể hiện của kiểu dữ liệu trừu tượng và các đối tượng trưởng thành để thấy là các đối tượng trưởng thành có một cấu trúc phức hợp, kiểu của chúng tham gia trong một dàn kiểu con. Chúng phải được xử lý một cách riêng rẽ vì:

- Thực hiện một giao tác trên một đối tượng có thể sinh ra các giao tác trên các đối tượng thành phần. Điều này bắt buộc phải có một lồng ẩn (Implicit Nesting) trên chính giao tác. Các thao tác trong các kiểu lồng này đều trừu tượng và cần phải xử lý như các giao tác nhiều mức.
- Việc sinh kiểu con/kế thừa hàm chứa việc chia sẻ hành vi và trạng thái giữa các đối tượng. Vì thế ngữ nghĩa của việc truy xuất một đối tượng tại một mức nào đó trong dàn phải tính đến điều này.

Cũng phải phân biệt các đối tượng thụ động (Passive) và các đối tượng chủ động (Active). Phương pháp tiếp cận đối với việc quản lý các đối tượng chủ động có khác nhau, tất cả các đề xuất đều giống nhau ở chỗ các đối tượng chủ động có khả năng đáp ứng lại các biến cố bằng cách kích hoạt thực hiện các hành động khi một số điều kiện nào đó được thỏa mãn. Các biến cần được theo dõi, điều kiện phải được đáp ứng và hành động được thực thi đáp trả. Vì biến cố có thể được phát hiện khi đang thực thi giao tác trên đối tượng, thực thi quy tắc tương ứng có thể phát sinh như một giao tác lồng. Tùy theo cách kết hợp quy tắc với giao tác ban đầu, các kiểu lồng khác nhau có thể xảy ra. Giao tác được phát sinh có thể được thực thi ngay, có thể để lại đến cuối giao tác hoặc thực thi trong một giao tác riêng rẽ khác.

6.7.3. Quản lý giao dịch trong các hệ quản trị đối tượng phân tán

Như đã trình bày trong mục trên, quản lý giao tác trong các DBMS đối tượng phải giải quyết với đồ thị hợp phần trong đó trình bày cấu trúc đối tượng hợp phần và dàn kiểu biểu diễn mối quan hệ giữa các đối tượng.

Đồ thị phối hợp đòi hỏi các phương pháp xử lý việc đồng bộ hóa các truy xuất đến các đối tượng có chứa các đối tượng khác làm thành phần. Dàn kiểu đòi hỏi bộ quản lý giao tác phải xem xét các vấn đề về phát triển lược đồ.

Ngoài những cấu trúc này, DBMS đối tượng lưu trữ phương thức cùng các dữ liệu. Đồng bộ hóa truy xuất chung đến các đối tượng phải xem xét đến việc thực thi phương thức. Cụ thể là các giao tác kích hoạt các phương thức rồi đến lượt chúng lại kích hoạt các phương thức khác. Vì vậy, nếu mô hình giao tác là phẳng thì việc thực hiện các giao tác này có thể trở thành hình thái lồng các trạng thái hoạt động.

Các định nghĩa tương tranh trong các môi trường khác không thể áp dụng DBMS đối tượng. Định nghĩa tương tranh kinh điển dựa trên tính giao hoán của các thao tác truy xuất đến cùng một đối tượng. Trong các DBMS đối tượng, có thể có các tương tranh giữa các thao tác truy xuất đến các đối tượng khác nhau. Điều này là do sự tồn tại của đồ thị hợp phần và dàn kiểu. Xét một thao tác M1 truy xuất đến đối tượng x có một đối tượng y khác làm một trong những thành phần của nó, có nghĩa x là một đối tượng hợp phần hoặc đối tượng phức. Có thể có một thao tác M2 khác, giả thiết M1 và M2 thuộc về các giao tác khác nhau, truy xuất đến y. Theo định nghĩa điển hình về tương tranh, không xem M1 và M2 tương tranh vì chúng truy xuất đến các đối tượng khác nhau. Tuy nhiên M1 xem y là thành phần của x và có thể muốn truy xuất y khi đang truy xuất x, gây ra một tương tranh với M2.

Các lược đồ được phát triển cho các DBMS đối tượng phải xem xét đến những vấn đề này. Sau đây là một số giải pháp:

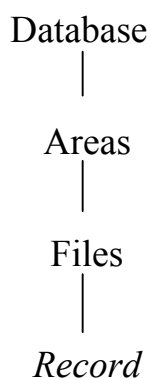
1. *Đồng bộ hóa truy xuất đến các đối tượng*: Lồng cổ hữu trong các kích hoạt phương thức có thể được sử dụng phát triển các thuật toán dựa trên khóa hai pha lồng 2PL và thuật toán sắp xếp thứ tự dấu thời gian. Sự thực hiện song song trong đối tượng có thể được dung để cải thiện tính đồng thời. Nói cách khác, thuộc tính của một đối tượng có thể được mô hình hóa như các phần tử dữ liệu trong cơ sở dữ liệu, còn các phương thức được mô hình như các giao tác,

cho phép kích hoạt nhiều phương thức của một đối tượng hoạt động cùng một lúc. Điều này làm tăng mức đồng thời nếu các giao thức đồng bộ đặc biệt trong đối tượng có thể được sửa lại nhằm duy trì tính tương thích của các quyết định đồng bộ hóa tại mỗi đối tượng.

Khi thực thi một phương thức đã được mô hình hóa như một giao tác, trên mỗi đối tượng gồm một số bước cục bộ (Local Step) tương ứng với việc thực thi các thao tác cục bộ cùng với các kết quả trả về và các bước phương thức (Method Step), đó là việc kích hoạt các phương thức cùng với các giá trị trả về. Một thao tác cục bộ là một thao tác nguyên tử, chẳng hạn như đọc, ghi, tăng, giảm... có tác động lên biến của đối tượng. Sự thực hiện một phương thức định nghĩa một thứ tự bộ phận giữa các bước này theo cách thức thông thường.

Sự độc lập hoàn toàn các đối tượng trong cách chúng đạt được sự đồng bộ hóa trong đối tượng đó, đòi hỏi các thực thi “đúng đắn”, nghĩa là chúng phải khá tuần tự dựa trên tính giao hoán. Ủy quyền đồng bộ hóa nội đối tượng từng đối tượng, thuật toán điều khiển đồng thời tập trung thực hiện đồng bộ hóa liên đối tượng.

Khóa chốt nhiều độ mịn trong Orion với nhiều phân cấp độ mịn khác nhau. Khóa chốt nhiều độ mịn được định nghĩa bằng một cây phân cấp cho các hạt cơ sở dữ liệu có thể khóa được, được gọi là “phân cấp độ mịn”. Cây phân cấp độ mịn mô tả như hình 6.3.



Hình 6.3. Cây phân cấp độ mịn

Trong các hệ quản trị cơ sở dữ liệu DBMS quan hệ, tập tin tương ứng với quan hệ và các bản ghi tương ứng với các bộ. Trong các hệ quản trị cơ sở dữ liệu DBMS đối tượng, tập tin tương ứng với lớp và các bản ghi tương ứng với đối tượng thể hiện. Sự phân cấp này đã giải quyết được các nhược điểm của khóa

hạt thô và hạt mịn. Khóa hạt thô, tại mức tập tin hoặc cao hơn, có chi phí khóa thấp, chỉ một số các khóa được đặt, làm giảm nhiều khả năng đồng thời. Khóa hạt mịn thì ngược lại.

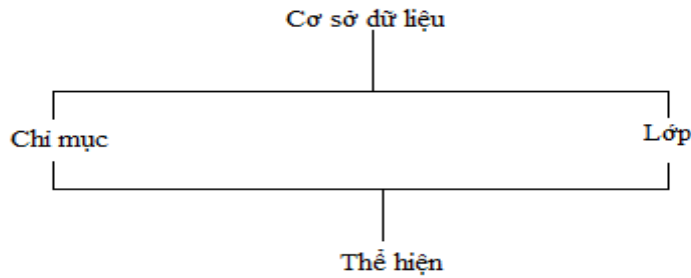
Mục tiêu của khóa chốt nhiều độ mịn đó là một giao tác lấy khóa tại một mức thô ngậm khóa tất cả các đối tượng tương ứng ở mức mịn hơn. Ví dụ khóa tường minh tại mức tập tin gồm khóa ngậm tất cả các bản ghi trong tập tin đó, bằng cách sử dụng ngoài khóa dung chung là S (Shared) và độc quyền X (Exclusive), sử dụng thêm hai loại khóa: khóa dung chung ẩn IS (Implicit Shared) và khóa độc quyền ẩn IX (Implicit Exclusive). Một giao tác muốn đặt một khóa S hoặc IS trên một đối tượng trước tiên phải đặt khóa IS trên các tổ tiên của nó. Tương tự, một giao tác muốn đặt khóa X hoặc khóa IX trên tất cả các đối tượng của nó. Khóa dung chung ẩn không được giải phóng trên một đối tượng nếu các hậu duệ của đối tượng đó hiện đang bị khóa.

Một khó khăn nảy sinh khi một giao tác muốn đọc một đối tượng ở một độ mịn nào đó và sửa đổi một số các đối tượng của nó ở một độ mịn hơn. Trong trường hợp này, cả khóa S và IX phải được đặt trên đối tượng đó. Ví dụ một giao tác có thể đọc một tập tin và cập nhật một số bản ghi nào đó trong tập tin, bằng cách đưa khóa độc quyền dùng chung ẩn SIX (Shared Intention Exclusive), tương đương với việc giữ một khóa S và IX trên đối tượng đó. Ma trận tương thích cho khóa nhiều độ mịn được trình bày trong hình 6.4.

	S	X	IS	IX	SIX
S	+	-	+	-	-
X	-	-	-	-	-
IS	+	-	+	+	+
IX	-	-	+	+	-
SIX	-	-	+	-	-

Hình 6.4. Bảng tương thích cho khóa nhiều độ mịn

Cây phân cấp nhiều độ mịn trong Orion như hình 6.5.



Hình 6.5. Cây phân cấp nhiều độ mịn trong Orion

Các đối tượng thể hiện chỉ bị khóa với thể thức S và X, còn các đối tượng lớp có thể bị khóa với cả năm thể thức. Diễn giải các khóa này trên các lớp như sau:

- Định nghĩa lớp bị khóa bởi thể thức S và tất cả các thể hiện của nó ngầm bị khóa ở thể thức S, y ngăn không cho một giao tác khác cập nhật các thể hiện.
- Định nghĩa lớp bị khóa ở thể thức X và tất cả các thể hiện của nó ngầm bị khóa ở thể thức X. Vì thế định nghĩa lớp và tất cả các thể hiện của nó có thể được đọc hoặc cập nhật.
- Định nghĩa lớp bị khóa ở thể thức IS và các thể hiện bị khóa ở thể thức S.
- Định nghĩa lớp bị khóa ở thể thức IX và các thể hiện có thể bị khóa ở thể thức S hoặc X nếu cần.
- Định nghĩa lớp bị khóa ở thể thức S và tất cả các thể hiện của nó ngầm bị khóa ở thể thức S. Các thể hiện cần phải cập nhật được ngầm khóa ở thể thức X khi giao tác cập nhật chúng.

2. *Quản lý dàn kiểu*: Sự tồn tại của các thao tác thay đổi lược đồ cùng với các câu truy vấn và giao tác thông thường, các mối quan hệ kế thừa định nghĩa giữa các lớp làm phức tạp hơn cho việc quản lý các lớp, các đối tượng phức tạp thêm.

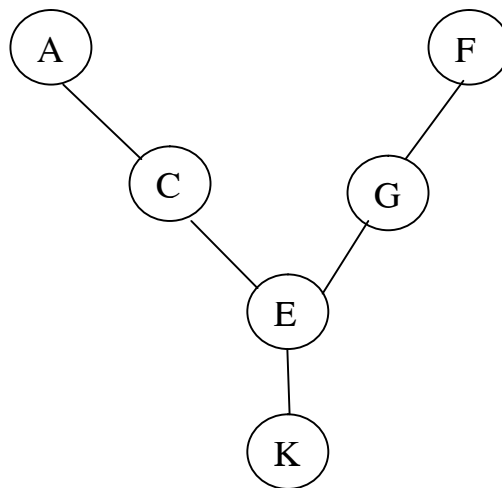
- Một truy vấn giao tác có thể không chỉ truy xuất đến các thể hiện của một lớp nhưng cũng có thể truy xuất đến các thể hiện của lớp con của lớp đó.
- Trong một đối tượng hợp phần, miền của một thuộc tính cũng có thể là một lớp, vì thế truy xuất đến một thuộc tính trong một lớp có thể phải truy xuất đến các đối tượng trong dàn kiểu con có gốc tại lớp miền của thuộc tính đó.

Giải quyết hai vấn đề trên sử dụng khóa nhiều độ mịn trong Orion. Mở

rộng đơn giản cho khóa nhiều độ mịn, ở đó lớp được truy xuất và những lớp con của nó sẽ được khóa ở thể thức thích hợp. Tuy nhiên các tiếp cận này không hiệu quả khi các lớp nằm gần gốc được truy xuất vì nó hàm chứa quá nhiều khóa. Khắc phục nhược điểm này bằng cách đưa ra các thể thức khóa dọc trên dàn R và ghi trên dàn W. Không chỉ khóa lớp đích thực ở thể thức S và X tương ứng mà còn ngầm khóa tất cả các lớp con của lớp đó tương ứng ở các thể thức S và X. Tuy nhiên giải pháp này không làm việc được với tính hướng đa kế thừa.

Đa kế thừa là một lớp có nhiều lớp cha có thể bị khóa ở các thể thức không tương ứng thích bởi hai giao tác có đặt các khóa R và W trên lớp cha khác nhau. Vì thể thức trên lớp chung đều thuộc kiểu ẩn, không có một khóa trên lớp đó. Vì vậy cần kiểm tra các lớp cha của một lớp đang bị khóa. Orion giải quyết bằng cách đặt các “khóa hiển”, không dùng khóa ẩn trên các lớp con. Xét dàn kiểu đơn giản như sau:

Nếu một giao tác T1 đặt trên một khóa IR trên lớp A và một khóa R trên C nó cũng sẽ đặt một khóa hiển thị trên E. Khi một giao tác T2 khác đặt một khóa IW trên F và một khóa W trên G, nó sẽ thử đặt một khóa hiển thị W trên E. Tuy nhiên vì đã có một khóa hiển R trên E nên yêu cầu này bị từ chối.



Hình 6.6. Ví dụ một số dàn kiểu

Cách tiếp cận của Orion là đặt các khóa hiển lên các lớp con của lớp đang được sửa đổi. Một phương pháp khác là đặt khóa ở một độ mịn hơn, sử dụng phương pháp dùng chung có thứ tự. Theo một nghĩa nào đó, thuật toán này là một sự mở rộng của lối tiếp cận dựa trên giao hoán của Weihl đối với các DBMS đối tượng bằng cách sử dụng một mô hình giao tác lồng.

Các phương pháp có thể được định nghĩa để hoạt tác trên các đối tượng lớp:

- Add (m) đưa phương thức m vào lớp,
- Del (m) xóa phương thức m ra khỏi lớp,
- Rep (m) thay thế cài đặt của phương thức m bằng một phương thức khác,
- Use (m) cho thực thi phương thức m.

Tương tự, các thao tác nguyên tử cũng được định nghĩa để truy xuất đến các thuộc tính của một lớp. Định nghĩa thao tác use (a) cho thuộc tính a chỉ ra rằng việc truy xuất của một giao tác đến thuộc tính a trong một thực thi phương thức sẽ được thông qua thao tác use(a), đòi hỏi mỗi phương thức phải liệt kê tất cả các thuộc tính mà nó truy xuất. Sau đây là một dãy các bước một giao tác cần tuân theo khi cho thực thi một phương thức m;

1. Giao tác T đưa ra thao tác use (m).
2. Với mỗi thuộc tính a mà phương thức m truy xuất, T đưa ra thao tác use (a).
3. Giao tác T khởi hoạt phương thức m.

Các bảng giao hoán được định nghĩa cho các thao tác phương thức và thuộc tính. Dựa trên các bảng giao hoán, bảng khóa dùng chung có thứ tự cho mỗi thao tác nguyên tử p có một mối liên hệ dùng chung với tất cả các khóa được đi kèm với các thao tác mà p có mối liên hệ không tương tranh, trong khi đó nó mối liên hệ dùng chung có thứ tự ứng với tất cả các khóa đi kèm với những thao tác mà p có một liên hệ tương tranh.

Dựa trên các bảng khóa này, một thuật toán khóa 2PL lỏng được dùng với những điều cần xem xét như sau:

1. Các giao tác nhận biết quy tắc 2PL nghiêm ngặt và duy trì các khóa của chúng cho đến khi kết thúc.
2. Khi một giao tác hủy bỏ, nó giải phóng tất cả các khóa của nó.
3. Việc kết thúc một giao tác bắt buộc phải chờ sự kết thúc của các con của nó. Khi một giao tác ủy thác, các khóa của nó được kế thừa bởi cha của nó,
4. Quy tắc ủy thác có thứ tự: Ví dụ hai thao tác là T_i và T_j sao cho T_i đang đợi T_j , T_i không thể ủy thác các thao tác của nó. Trên một đối tượng bất kì cho đến khi T_j kết thúc. T_i được xem là đang đợi T_j nếu:

- T_i không phải là gốc của giao tác lỏng và T_i đã được trao một khóa với một mối liên hệ dùng chung có thứ tự ứng với một khóa được giữ bởi T_j trên một đối tượng sao cho T_j là một hậu duệ của T_i .

• T_i là gốc của giao tác lồng và T_i đang giữ một khóa được kế thừa hoặc được trao trên một đối tượng trong mỗi liên hệ dùng chung có thứ tự ứng với một khóa mà T_j hoặc các hậu duệ của nó đang giữ.

5. *Quản lý đồ thị hợp phần*: Yêu cầu các hệ quản trị cơ sở dữ liệu DBMS đối tượng mô hình hóa các đối tượng hợp phần với một phương pháp hiệu quả bằng cách dựa vào cách tiếp cận của Orion đối việc quản lý cây phân cấp hợp phần dựa trên khóa nhiều độ mịn. Nghi thức khóa nhiều độ mịn không xem đối tượng hợp phần là một đơn vị khóa. Giải quyết vấn đề này, có ba thể thức khóa mới: ISO, IXO và SIXO tương ứng với IS, IX và SIX. Các thể thức khóa này được dùng để khóa các lớp thành phần của một đối tượng hợp phần. Tính tương thích của chúng được trình bày trong hình 6.7.

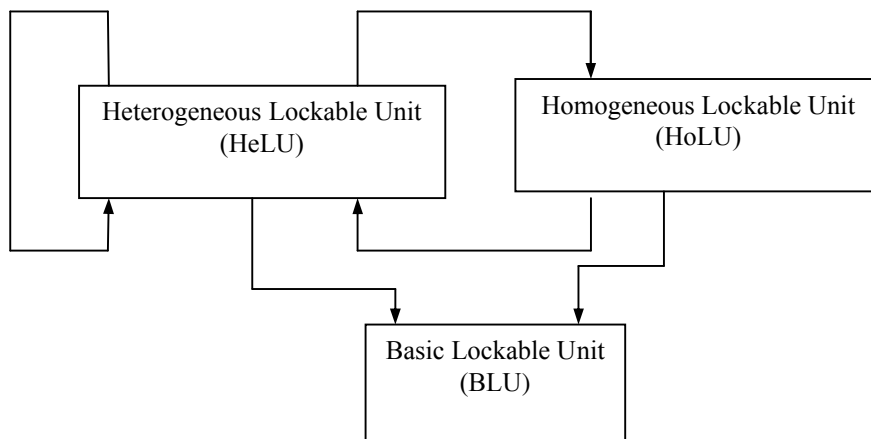
	S	X	IS	IX	SIX	ISO	IXO	SIXO
S	+	-	+	-	-	+	-	-
X	-	-	-	-	-	-	-	-
IS	+	-	+	+	+	+	-	-
IX	-	-	+	+	-	-	-	-
SIX	-	-	+	-	-	-	-	-
ISO	+	-	+	+	-	+	+	+
IXO	-	-	-	-	-	+	+	-
SIXO	N	N	N	N	N	Y	N	N

Hình 6.7. Ma trận tương thích cho các đối tượng hợp phần

Nghi thức hoạt động như sau: Để khóa một đối tượng hợp phần, lớp gốc bị khóa ở thể thức X, IS, IX hoặc SIX và mỗi lớp thành phần của cây phân cấp đối tượng hợp phần bị khóa lần lượt ở thể thức X, ISO, IXO và SIXO.

Các thuật toán điều khiển đồng thời trong Orion dựa trên khóa nhiều độ mịn sẽ cường chế tính khả tuần tự. Thuật toán khôi phục được dùng là ghi nhật ký bằng phương pháp no-fix/flush, chỉ đòi hỏi phải hồi lại nhưng không thực hiện lại.

Một mở rộng của khóa nhiều độ mịn để giải quyết với đồ thị hợp phần là thay thế một đồ thị khóa tĩnh duy nhất bằng một phân cấp các đồ thị đi kèm với mỗi kiểu và truy vấn. Có một đồ thị khóa tổng quát chịu trách nhiệm điều khiển toàn bộ quá trình này. Đồ thị khóa đó như hình 6.8.



Hình 6.8. Đồ thị khóa tổng quát

Đơn vị nhỏ nhất có thể khóa gọi là đơn vị khóa cơ bản BLU. Một số các BLU có thể tạo ra một đơn vị khóa đồng chủng HoLU cấu tạo bởi các kiểu dữ liệu có cùng kiểu. Tương tự có thể cấu tạo một khóa đa chủng HeLU cấu tạo bởi các đối tượng thuộc về các kiểu khác nhau. HeLU có thể chứa các HeLU khác hoặc các HoLU, chỉ ra rằng các đối tượng thành phần không phải là nguyên tử. Tương tự các HoLU có thể chứa các HeLU hoặc các HoLU khác với điều kiện chúng phải cùng kiểu. Phân biệt giữa HoLU và HeLU là nhằm để tối ưu các yêu cầu khóa. Ví dụ một tập danh sách các số nguyên, trên quan điểm bộ quản lý khóa, sẽ được xử lý như một HoLU do nhiều HoLU tạo ra, đến lượt nó lại được cấu tạo từ các BLU. Kết quả là có khóa toàn bộ tập, một trong số danh sách hoặc thậm chí một số nguyên.

Khi định nghĩa kiểu, một đồ thị khóa cụ thể cho một đối tượng được tạo ra và tuân theo đồ thị khóa tổng quát. Cũng như vậy một thành phần thứ ba, một đồ thị khóa cụ thể cho một truy vấn được tạo ra trong khi phân tích truy vấn. Khi thực hiện truy vấn, đồ thị khóa cụ thể cho truy vấn được dùng để yêu cầu khóa bộ quản lý khóa và nó sử dụng đồ thị khóa cụ thể cho đối tượng để đưa ra quyết định. Các thể thức khóa được dùng là các thể thức khóa chuẩn, nghĩa là: IS, IX, S, X.

Danh mục tài liệu tham khảo

1. Lê Tiến Vương (1996). *Nhập môn cơ sở dữ liệu quan hệ*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
2. Nguyễn Bá Tường (2005). *Nhập môn cơ sở dữ liệu phân tán*. Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.
3. Phạm Thế Quế (2004). *Giáo trình Cơ sở dữ liệu: Lý thuyết và thực hành*. Nhà xuất bản Bưu điện Hà Nội.
4. Phạm Thế Quế (2010). *Giáo trình Cơ sở dữ liệu phân tán*. Nhà xuất bản Thông tin và Truyền thông, Hà Nội.
5. M.Tamer Ozsü And Patrice Valduriez (1999). *Principles of Distributed Database Systems*, Second Edition. Prentice Hall Upper Saddle River.

Câu hỏi ôn tập và bài tập cuối chương

1. Trình bày các loại cơ sở dữ liệu phổ biến.
2. Trình bày khái niệm cơ bản về đối tượng và mô hình dữ liệu đối tượng.
3. Trình bày khái niệm chung về thiết kế phân tán đối tượng.
4. Trình bày phân hoạch ngang lớp phân tán đối tượng.
5. Trình bày phân hoạch dọc lớp phân tán đối tượng.
6. Trình bày phân hoạch đường dẫn.
7. Trình bày các thuật toán phân hoạch.
8. Trình bày cấp phát dữ liệu phân tán đối tượng.
9. Trình bày nhân bản phân tán đối tượng.
10. Trình bày các mô hình kiến trúc phân tán đối tượng.
11. Trình bày các kiểu kiến trúc máy khách/chủ phân tán đối tượng.
12. Trình bày lưu trữ đối tượng phân tán.
13. Trình bày quản lý đối tượng.
14. Trình bày quản lý con trỏ.
15. Trình bày di trú đối tượng.
16. Trình bày di trú đối tượng.
17. Trình bày xử lý truy vấn đối tượng.
18. Trình bày kiến trúc xử lý truy vấn đối tượng.
19. Trình bày các vấn đề xử lý truy vấn đối tượng.
20. Trình bày thực thi truy vấn đối tượng.
21. Trình bày quản lý giao dịch đối tượng phân tán.
22. Trình bày các tiêu chuẩn quản lý.
23. Trình bày các mô hình giao dịch và cấu trúc đối tượng.

BÀI TẬP

1. Hãy liệt kê các bài toán phân tán đối tượng nảy sinh trong các DBMS đối tượng, là những vấn đề không có mặt trong các DBMS quan hệ ứng với việc phân mảnh, di trú và nhân bản.
2. Trình bày mối quan hệ giữa làm tụ và phân mảnh. Minh họa việc làm tụ có thể làm giảm hoặc cải thiện hiệu năng của các truy vấn trên các CSDL đối tượng có phân hoạch hay không.
3. Vì sao các DBMS đối tượng máy khách/chủ thường sử dụng kiến trúc chuyển và gửi dữ liệu, còn các DBMS quan hệ lại sử dụng các chuyển/nhận chức năng.
4. Xét bài toán đặt vé máy bay, hãy định nghĩa một lớp kiểu Reservation và đưa ra các ma trận giao hoán tới và lùi con trỏ.

MỤC LỤC

LỜI NÓI ĐẦU.....	3
Chương 1. TỔNG QUAN VỀ CSDL PHÂN TÁN.....	5
1.1. Sơ lược về mạng máy tính.....	5
1.2. Các hình thức tổ chức hệ thống phân tán.....	7
1.3. Các đặc trưng của các loại hình hệ thống phân tán.....	8
1.3.1. Đặc trưng của hệ thống File server và kiến trúc Client/Server.....	9
1.3.2. Các chức năng của kiến trúc client/server.....	9
1.4. Cơ sở dữ liệu phân tán (Distributed Database).....	10
1.4.1. Định nghĩa:.....	10
1.4.2. Hệ quản trị cơ sở dữ liệu phân tán (DDBMSs).....	12
1.4.3. So sánh CSDL phân tán và CSDL tập trung.....	12
1.4.4. Ưu và nhược điểm của hệ phân tán.....	14
1.5. Các loại truy xuất CSDL phân tán.....	15
1.5.1. Truy xuất từ xa thông qua các tác vụ cơ bản:.....	15
1.5.2. Truy xuất từ xa thông qua chương trình phụ trợ.....	15
1.6. Kiến trúc cơ bản của CSDL phân tán.....	16
1.7. Các đặc điểm chính của hệ phân tán.....	18
1.7.1. Chia sẻ tài nguyên.....	18
1.7.2. Tính mở.....	19
1.7.3. Khả năng song song.....	19
1.7.4. Khả năng mở rộng.....	19
1.7.5. Khả năng thứ lỗi.....	20
1.7.6. Đảm bảo tin cậy và nhất quán.....	20
1.8. Trong suốt phân tán.....	20
1.8.1. Tính trong suốt.....	20
1.8.2. Trong suốt phân đoạn (fragmentation transparency):.....	20
1.8.3. Tính trong suốt về vị trí (location transparency):.....	21
1.8.4. Trong suốt ánh xạ địa phương (local mapping transparency):.....	22
Danh mục tài liệu tham khảo.....	23
Câu hỏi cuối chương.....	23
Chương 2. THIẾT KẾ CSDL PHÂN TÁN.....	24
2.1. Nội dung thiết kế một hệ thống phân tán.....	24
2.1.1. Các công việc cần phải làm để thiết kế HT phân tán:.....	24

2.1.2. Các sản phẩm yêu cầu sau khi phân tích thiết kế	24
2.2. Các chiến lược phân tán dữ liệu	25
2.2.1. Tập trung dữ liệu:	25
2.2.2. Chia nhỏ dữ liệu:.....	25
2.2.3. Sao lãp dữ liệu:	25
2.2.4. Phương thức lai:	26
2.3. Phương pháp thiết kế CSDL phân tán.....	26
2.3.1. Sơ đồ thiết kế tổng thể cơ sở dữ liệu phân tán.....	26
2.3.2. Các phương pháp thiết kế CSDL phân tán	27
2.4. Phân mảnh dữ liệu.....	28
2.4.1. Các lý do phân mảnh.....	30
2.4.2. Các kiểu phân mảnh.....	30
2.4.3. Các yêu cầu của việc phân mảnh.....	31
2.5. Thông tin cấp phát	33
Danh mục tài liệu tham khảo	35
Câu hỏi và bài tập cuối chương	36
Chương 3. XỬ LÝ TRUY VẤN TRONG CSDL PHÂN TÁN.....	39
3.1. Giới thiệu về xử lý truy vấn	39
3.2. Xử lý truy vấn trong môi trường tập trung	39
3.2.1. So sánh xử lý truy vấn tập trung và phân tán	40
3.2.2. Chiến lược tối ưu trong CSDL tập trung	40
3.3. Xử lý truy vấn trong môi trường phân tán	44
3.3.1. Phân rã truy vấn.....	45
3.3.2. Định vị dữ liệu phân tán-Tối ưu hóa cục bộ.....	53
3.4. Tối ưu hóa truy vấn trong CSDL phân tán.....	60
3.4.1. Mô hình chi phí của bộ tối ưu hóa truy vấn.....	60
3.4.2. Các thống kê dữ liệu	62
3.4.3. Lực lượng của các kết quả trung gian	64
Danh mục tài liệu tham khảo	67
Câu hỏi và bài tập cuối chương	67
Chương 4. QUẢN LÝ GIAO DỊCH VÀ ĐIỀU KHIỂN ĐỒNG THỜI PHÂN TÁN .	72
4.1. Tổng quan về giao dịch.....	72
4.1.1. Điều kiện kết thúc giao dịch.....	75
4.1.2. Đặc tính của giao dịch	76

4.1.3. Đặc trưng hóa khái niệm giao dịch.....	76
4.2. Các tính chất của giao dịch	78
4.2.1. Tính nguyên tố.....	78
4.2.2. Tính nhất quán.....	79
4.2.3. Tính cô lập.....	80
4.2.4. Tính bền vững.....	83
4.3. Các loại giao dịch.....	83
4.3.1. Các loại cấu trúc theo thời gian hoạt động	83
4.3.2. Các loại giao dịch dựa trên việc tổ chức các hành động đọc và ghi.....	83
4.3.3. Luồng công việc – WorkFlows.....	84
4.4. Điều khiển các giao dịch đồng thời phân tán	86
4.4.1. Đặt vấn đề	86
4.4.2. Tính khả tuần tự lịch biểu.....	87
4.4.3. Phân loại các cơ chế điều khiển đồng thời	93
4.5. Các thuật toán điều khiển đồng thời bằng khóa chốt	95
4.5.1. Thuật toán quản lý khóa cơ bản.....	96
4.5.2. Thuật toán khóa chốt hai pha 2PL.....	98
4.5.3. Thuật toán quản lý giao dịch 2PL tập trung (C2PL TM)	102
4.5.4. Thuật toán 2PL bản chính.....	107
4.5.5. Thuật toán 2PL phân tán.....	108
4.6. Các thuật toán điều khiển đồng thời bằng nhãn thời gian.....	108
4.6.1. Đặt vấn đề.....	108
4.6.2. Thuật toán bộ quản lý giao dịch TO cơ bản	109
4.6.3. Thuật toán TO bảo toàn	114
4.6.4. Thuật toán TO đa phiên bản	116
4.7. Các thuật toán điều khiển đồng thời lạc quan	116
4.8. Quản lý bế tắc	119
4.8.1. Ngăn chặn bế tắc.....	120
4.8.2. Tránh bế tắc.....	121
4.8.3. Phát hiện và giải tỏa bế tắc.....	122
Danh mục tài liệu tham khảo.....	124
Câu hỏi và bài tập cuối chương.....	125
Chương 5. CÁC HỆ CƠ SỞ DỮ LIỆU SONG SONG.....	128
5.1. Mục tiêu của xử lý song song.....	128

5.2. Ưu điểm của các cơ sở dữ liệu song song.....	128
5.2.1. Hiệu năng cao	129
5.2.2. Tính sẵn sàng	129
5.2.3. Khả năng mở rộng.....	130
5.3. Kiến trúc hệ cơ sở dữ liệu song song	130
5.3.1. Bộ quản lý phiên.....	130
5.3.2. Bộ quản lý yêu cầu	130
5.3.3. Bộ quản lý dữ liệu	130
5.4. Các kiến trúc hệ thống song song	131
5.4.1. Tổng quan về kiến trúc song song và hệ thống song song.....	131
5.4.2. Kiến trúc chia sẻ bộ nhớ	131
5.4.3. Kiến trúc chia sẻ đĩa	133
5.4.4. Kiến trúc không chia sẻ.....	134
5.4.5. Kiến trúc phân cấp.....	135
5.5. Kỹ thuật hệ quản trị cơ sở dữ liệu song song.....	135
5.5.1. Sắp đặt dữ liệu.....	136
5.5.2. Truy vấn song song	142
5.5.3. Xử lý dữ liệu song song.....	143
5.6. Tối ưu hóa truy vấn song song.....	147
5.6.1. Không gian tìm kiếm	147
5.6.2. Mô hình chi phí	149
5.6.3. Chiến lược tìm kiếm	150
Danh mục tài liệu tham khảo	152
Câu hỏi ôn tập cuối chương.....	153
Chương 6. HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU ĐỐI TƯỢNG PHÂN TÁN	154
6.1. Giới thiệu.....	154
6.2. Khái niệm cơ bản về đối tượng và mô hình dữ liệu đối tượng	154
6.3. Thiết kế phân tán đối tượng	156
6.3.1. Phân hoạch ngang lớp:.....	156
6.3.2. Phân hoạch dọc lớp.....	159
6.3.3. Phân hoạch đường dẫn:.....	159
6.3.4. Các thuật toán phân hoạch:.....	159
6.3.5. Cấp phát	160

6.3.6. Nhân bản	161
6.4. Các mô hình kiến trúc đối tượng phân tán	161
6.4.1. Các kiểu kiến trúc máy khách/chủ.....	162
6.4.2. Lưu trữ đối tượng phân tán.....	164
6.5. Quản lý đối tượng.....	165
6.5.1. Quản lý định danh đối tượng	165
6.5.2. Quản lý con trỏ:	166
6.5.3. Di trú đối tượng.....	167
6.6. Xử lý truy vấn đối tượng	168
6.6.1. Kiến trúc xử lý truy vấn đối tượng.	169
6.6.2. Các vấn đề xử lý truy vấn đối tượng	170
6.6.3. Thực thi truy vấn đối tượng.....	173
6.7. Quản lý giao dịch đối tượng phân tán	174
6.7.1. Các tiêu chuẩn quản lý.....	174
6.7.2. Mô hình giao dịch và cấu trúc đối tượng.....	175
6.7.3. Quản lý giao dịch trong các hệ quản trị đối tượng phân tán	177
Danh mục tài liệu tham khảo.....	185
Câu hỏi ôn tập và bài tập cuối chương.....	186