Chương 5 - Thiết kế cơ sở dữ liệu phân tán

Thiết kế của một hệ thống máy tính phân tán bao gồm việc ra quyết định về việc đặt dữ liệu và chương trình ở các trạm trên một mạng máy tính, cũng như là khả năng thiết kế chính mạng máy tính đó. Trong trường hợp hệ thống quản lý CSDL phân tán, việc phân tán các ứng dụng gồm 2 điều: sự phân tán của phần mềm hệ thống quản lý CSDL phân tán và sự phân tán của các chương trình ứng dụng chạy trên phần mềm hệ thống đó. Sự phân tán của các phần mềm hệ thống quản lý CSDL phân tán tồn tại trong từng trạm lưu trữ dữ liệu. Ở chương này, chúng ta không quan tâm về vị trí của các chương trình ứng dụng. Hơn nữa, chúng ta giả sử rằng mạng đã được thiết kế, hoặc sẽ được thiết kế ở bước tiếp theo, tuỳ theo quyết định liên quan tới việc thiết kế CSDL phân tán. Chúng ta tập trung vào việc phân tán dữ liệu.

Chúng ta biết rằng tổ chức các hệ thống phân tán có thể được nghiên cứu theo ba chiều trực giao (Levin và Morgan, 1975):

1. Mức độ chia sẻ
2. Hành vi của các thành phần truy cập
3. Mức độ kiến thức trong hành vi của thành phần truy nhập

Hình 5.1 mô tả khá rõ các chiều này. Về mức độ chia sẻ, có 3 khả năng. Đầu tiên là không chia sẻ: mỗi ứng dụng và dữ liệu của nó thực hiện trên một trạm, và không có sự giao tiếp với các chương trình khác hoặc truy cập tới bất kỳ các file dữ liệu nào ở các trạm khác. Điều này là đặc trưng của mạng trong những buổi sơ khai đầu tiên và ngày nay đã không còn phổ biến nữa. Sau đó chúng ta đã phát triển lên mức độ chia sẻ dữ liệu: tất cả các chương trình được lặp lại tại tất cả các trạm, nhưng các file dữ liệu thì không. Như vậy, các yêu cầu của người sử dụng được xử lý tại trạm phát sinh yêu cầu và các file dữ liệu cần thiết được di chuyển khắp trên mạng. Cuối cùng, ở mức chia sẻ chương trình-cộng-dữ liệu, cả dữ liệu và các chương trình đều có thể được chia sẻ, có nghĩa là một

chương trình ở một trạm đã cho có thể yêu cầu một dịch vụ từ một chương trình khác ở một trạm thứ hai, mà trong đó trạm thứ hai có thể phải truy cập file dữ liệu nằm trong trạm thứ ba.

Levin và Morgan vạch ra sự khác biệt giữa chia sẻ dữ liệu và chia sẻ chương trình-cộng-dữ liệu để minh hoạ cho sự khác biệt giữa hệ thống máy tính phân tán đồng nhất và không đồng nhất. Họ chỉ ra rằng trong môi trường không đồng nhất thì thường là rất khó, nếu không muốn nói là không thể, thực hiện một chương trình đã cho trên một phần cứng khác dưới một hệ điều hành khác. Tuy nhiên có thể di chuyển dữ liệu một cách khá dễ dàng.

Với chiều thứ hai về hành vi của thành phần truy nhập, chúng ta có thể xác định hai khả năng. Các thành phần truy nhập của các yêu cầu người sử dụng có thể là tĩnh, nghĩa là chúng không đổi theo thời gian, hoặc là động. Hiển nhiên là người ta có thể lên kế hoạch và quản lý môi trường tĩnh dễ hơn so với trường hợp của các hệ thống phân tán động. Thật không may, rất khó thể tìm ra nhiều ứng dụng phân tán thực sự được phân loại là tĩnh. Do đó, vấn đề quan trọng không phải là việc hệ thống là tĩnh hay động mà là động như thế nào. Ngẫu nhiên là ngoài khía cạnh này còn có mối quan hệ giữa thiết kế CSDL phân tán và việc xử lý truy vấn được nêu lên (xem hình 1.8).

Chiều thứ ba là mức độ kiến thức trong hành vi của thành phần truy nhập. Tất nhiên có khả năng những nhà thiết kế không có bất kỳ thông tin nào về việc người sử dụng truy nhập CSDL như thế nào. Đây là một khả năng về lý thuyết, nhưng rất khó, nếu không muốn nói là không thể, thiết kế một hệ thống quản lý CSDL phân tán mà có thể thích ứng một cách hiệu quả với trường hợp này. Một khía cạnh thực tế hơn là các nhà thiết kế có thông tin đầy đủ, nghĩa là các thành phần truy nhập có thể được dự đoán trước ở mức độ chấp nhận được và không quá sai lệch so với những dự đoán này, và thông tin thành phần, trong đó có những sai lệch so với dự đoán.

Vấn đề về thiết kế CSDL phân tán cần phải được xem xét trong phạm vi chung này. Tất cả các trường hợp được thảo luận ở trên, trừ các trường hợp trong khả năng không chia sẻ, các vấn đề mới sẽ được đưa ra trong môi trường phân tán mà không liên quan tới việc thiết lập tập trung. Trong chương này, mục đích của chúng ta tập trung vào những vấn đề khác thường. Phân bố của chương như sau. Trong phần 5.1, chúng ta thảo luận ngắn gọn hai phương pháp thiết kế CSDL phân tán: phương pháp thiết kế từ trên xuống và từ dưới lên. Chi tiết của phương pháp từ trên xuống được trình bày trong phần 5.3 và 5.4, trong khi đó chi tiết về phương pháp từ dưới lên được trình bày trong một chương khác (chương 14). Trước khi thảo luận các vấn đề này, chúng tôi trình bày các vấn đề trong thiết kế phân tán ở chương 5.2.

5.1. CÁC PHƯƠNG PHÁP THIẾT KẾ CHÍNH

Hai phương pháp thiết kế chủ yếu đã được đưa ra (Ceri et al., 1987) để thiết kế CSDL phân tán là phương pháp từ trên xuống và phương pháp từ dưới lên. Như tên gọi, chúng là các phương pháp rất khác nhau trong quá trình thiết kế. Nhưng như tất cả các nhà thiết kế phần mềm đã biết, các ứng dụng thật sự rất hiếm khi đơn giản đủ để phù hợp với một trong những phương pháp này. Do đó điều quan trọng cần phải nhớ là trong hầu hết các trường hợp thiết kế CSDL, cần phải ứng dụng cả hai phương pháp này để bổ sung cho nhau.

Chung tôi cũng trình bày về một thiết kế hệ thống CSDL sử dụng hệ thống quản lý CSDL phân tán trong phạm vi được thảo luận ở phần 4.3. Hành động này là một chức năng kết hợp của cơ sở dữ liệu, doanh nghiệp, và các nhà quản trị hệ thống ứng dụng (hay về việc các nhà quản trị thực hiện cả 3 vai trò trên).

5.1.1. Quá trình thiết kế từ trên xuống

Khung của quá trình được trình bày trong hình 5.2. Hành động bắt đầu bằng một phân tích yêu cầu mà xác định môi trường của hệ thống và “moi ra cả nhu cầu về dữ liệu và xử lý của tất cả những

người sử dụng CSDL tiềm năng" [Yao et al., 1982a]. Việc nghiên cứu yêu cầu cũng được chỉ rõ trong đó hệ thống cuối cùng phải đáp ứng được các mục tiêu của hệ thống quản lý CSDL phân tán được xác định trong phần 1.2. Các mục tiêu đó là sự hoạt động, tính tin cậy và sẵn có, kinh tế và khả năng mở rộng (tính linh hoạt).

Tài liệu yêu cầu được là dữ liệu đầu vào với hai hành động song song: thiết kế mô hình và thiết kế lý thuyết. Hoạt động thiết kế mô hình quan tâm đến việc xác định giao diện với người dùng cuối. Mặt khác, thiết kế lý thuyết là quá trình kiểm tra doanh nghiệp để xác định loại thực thể và mối quan hệ giữa các thực thể. Người ta có thể chia quá trình này thành hai nhóm hoạt động quan hệ [Davenport, 1980]: phân tích thực thể và phân tích chức năng. Phân tích thực thể quan tâm tới việc xác định thực thể, tính chất của chúng và mối quan hệ giữa các thực thể. Mặt khác, phân tích chức năng quan tâm đến việc xác định các chức năng cơ bản của từng mô hình doanh nghiệp có tham gia. Kết quả của 2 bước này cần được tham khảo lẫn nhau để có được sự hiểu biết rõ hơn về chức năng nào xử lý thực thể nào.

Có một mối quan hệ giữa thiết kế lý thuyết và thiết kế mô hình. Dưới cảm nhận của người khác thì thiết kế lý thuyết có thể được xem như là sự tích hợp các cách nhìn của người dùng. Thậm chí mặc dù hành động tích hợp cách nhìn này là rất quan trọng, mô hình lý thuyết phải hỗ trợ không chỉ các ứng dụng đã có, mà còn cả các ứng dụng tương lai. Tích hợp cách nhìn được sử dụng để đảm bảo rằng thực thể và các yêu cầu quan hệ đối với tất cả các cách nhìn được bảo phủ trong mô hình lý thuyết.

Trong các hoạt động của thiết kế lý thuyết và thiết kế cách nhìn người dùng cần xác định rõ các thực thể dữ liệu và phải quyết định ứng dụng sẽ chạy trên CSDL cũng như là các thông tin thông kê về các ứng dụng này. Thông tin thống kê bao gồm sự xác định rõ tần suất của ứng dụng người dùng, khối lượng các thông tin khác nhau, v.v.. . Lưu ý là từ bước thiết kế lý thuyết tới việc định nghĩa mô hình lý thuyết toàn cục được thảo luận trong phần 4.3. Trong thực tế cho tới thời điểm này chúng ta vẫn chưa xem xét sự ảnh hưởng của môi trường phân tán, quá trình xử lý tương tự như trong thiết kế csdl tập trung.

Mô hình lý thuyết toàn cục (GCS) và thông tin thành phần truy nhập được tập hợp lại như một kết quả của thiết kế quan sát là dữ liệu đầu vào cho bước thiết kế phân tán. Mục tiêu của bước này mà được tập trung trong chương này, là để thiết kế các mô hình lý thuyết cục bộ (LCS) bằng cách phân bố thực thể tới các trạm của hệ thống phân tán. Tất nhiên có thể coi từng thực thể như một đơn vị của phân tán. Giả sử chúng ta sử dụng mô hình quan hệ làm cơ sở thảo luận trong quyển sách này, các thực hiện tương ứng với các mối quan hệ.

Với các quan hệ phân tán, người ta thường chia chúng thành hai quan hệ con, được gọi là phân mảnh, mà sau đó được phân bổ. Do vậy hoạt động thiết kế phân tán bao gồm 2 bước: phân mảnh và phân phối. Đây là những vấn đề chính được xét đến trong chương này, như vậy chúng ta dành việc thảo luận chúng cho tới chương sau.

Bước cuối của quá trình thiết kế là thiết kế vật lý. Đó là sắp lại các mô hình lý thuyết cục bộ trong thiết bị lưu trữ vật lý có sẵn ở các trạm tương ứng. Dữ liệu đầu vào của quá trình này là các mô hình lý thuyết vật lý và thông tin thành phần truy nhập về các phân mảnh trong chúng.

Chúng ta biết là hoạt động thiết kế và phát triển là một quá trình tiếp diễn yêu cầu việc giám sát thường xuyên và sự điều chỉnh có chu kỳ. Do đó chúng tôi đã bao gồm cả việc quan sát và kiểm soát như là một hoạt động chính trong quá trình này. Lưu ý là không chỉ kiểm soát hành vi của việc thực thi csdl mà còn cả khả năng thích ứng của quan điểm người dùng. Kết quả là một số hình thức phản hồi mà có thể dẫn tới việc lặp lại một trong các bước trước đó trong thiết kế.

5.1.2. Quá trình thiết kế từ dưới lên

Thiết là từ trên xuống là một phương pháp thích hợp khi hệ thống csdl được thiết kế từ đầu (không có gì). Tuy nhiên thường thì một số csdl đã có sẵn, và công việc thiết kế bao gồm cả việc tích hợp chúng thành một csdl. Phương pháp từ dưới lên thích hợp với môi trường này. Khởi điểm ban đầu của thiết kế từ dưới lên là các mô hình lý thuyết cục bộ riêng lẻ. Quá trình bao gồm việc tích hợp các phương pháp cục bộ thành một phương pháp lý thuyết toàn cục.

Loại môi trường tồn tại chủ yếu trong nội dụng của csdl không đồng nhất. Những nghiên cứu quan trọng cũng được trình bày trong nội dung này. Do vậy, chúng tôi sẽ trình bày quá trình thiết kế từ dưới lên trong chương 14. Phần còn lại của chương tập trung vào 2 vấn đề cơ bản trong thiết kế từ trên xuống: phân mảnh và phân phối.

5.2. CÁC VẤN ĐỀ THIẾT KẾ PHÂN TÁN

Trong phần trước chúng tôi đã chỉ ra rằng mối quan hệ trong một mô hình csdl thường được phân tách thành các mảnh nhỏ hơn, nhưng chúng tôi không đưa ra bất kỳ một sự điều chỉnh hoặc chi tiết nào cho quá trình này. Mục tiêu của phần này là trình bày các chi tiết đó.

Tập hợp các truy vấn liên quan sau đây bao phủ toàn bộ vấn đề. Do vậy chúng tôi tìm kiếm câu trả lời cho những truy vấn này trong toàn bộ phần còn lại của phần này.

- Tại sao phân mảnh?
- Phân mảnh như thế nào?
- Nên phân mảnh bao nhiêu?
- Có cách nào để kiểm tra tính đúng đắn của việc phân rã?
- Phân phối như thế nào?
- Thông tin cần thiết cho quá trình phân mảnh và phân phối?

5.2.1. Lý do phân mảnh

Theo quan điểm của phân tán dữ liệu, thực sự chẳng có lý do gì để phân mảnh dữ liệu. Thực sự thì trong hệ thống file phân tán, sự phân tán được thực hiện trên cơ sở của toàn bộ các file. Trên thực tế các nghiên cứu trước đây nhằm giải quyết chủ yếu việc phân phối file tới các nốt trên mạng máy tính. Chúng tôi xét các mô hình đó trong phần 5.4.

Về việc phân mảnh, vấn đề quan trọng là số lượng thích hợp cho phân phối. Quan hệ không phải là đơn vị phù hợp, đối với một số nguyên nhân. Đầu tiên, các cách nhìn của ứng dụng thường là tập con các quan hệ. Do vậy. tính cục bộ của sự truy cập của các ứng dụng không được xác định trên toàn bộ mối quan hệ mà là trên các tập con của chúng. Vì thế hiển nhiên là ta cần phải coi các tập con quan hệ là các đơn vị phân phối.

Thứ hai, nếu ứng dụng có cách nhìn được xác định trên một quan hệ đã cho nằm tại một trạm khác, có hai phương thức có thể được thực hiện, với toàn bộ quan hệ là đơn vị phân phối. Hoặc là quan hệ không được lặp lại và được lưu chỉ ở tại một trạm duy nhất, hoặc là nó được lặp lại tại tất cả hoặc một số các trạm có ứng dụng ở đó. Cách thức không lặp lại mối quan hệ sẽ tạo ra một khối lượng lớn các truy cập dữ liệu từ xa không cần thiết. Trong khi đó, cách thức lặp lại mối quan hệ sẽ tạo ra các bản sao không cần thiết, gây ra các vấn đề trong việc thực hiện cập nhật (sẽ được thảo luận sau) và có thể là không mong muốn nếu dung lượng lưu trữ là có hạn.

Cuối cùng, sự phân rã một mối quan hệ thành các phân mảnh, từng phân mảnh được xem như là một đơn vị, cho phép một số các giao dịch thực hiện đồng thời. Hơn nữa, việc phân mảnh mối quan hệ thường dẫn đến việc xử lý song song một truy vấn đơn bằng cách chia nó thành một tập các truy vấn con mà chạy trên các phân mảnh. Do đó việc phân mảnh làm tăng mức độ đồng thời và dẫn tới tăng thời gian xử lý của hệ thống. Hình thái đồng thời này, chúng tôi muốn nói tới truy vấn trong, được xét trong chương 8 và 9, ở phần xử lý truy vấn.

Để đầy đủ hơn, chúng tôi cũng trình bày nhược điểm của phân mảnh. Nếu ứng dụng có mâu thuẫn trong yêu cầu dẫn tới việc ngăn cản sự phân rã quan hệ thành các phân mảnh khác biệt nhau, những ứng dụng có cách nhìn được xác định trên nhiều phân mảnh có thể sẽ bị giảm khả năng hoạt động. Chẳng hạn, cần phải nhận dữ liệu từ hai phân mảnh và sau đó hợp chúng làm một hoặc làm chúng liên kết với nhau sẽ rất tốn chi phí. Tránh điều này chính là vấn đề của phân mảnh cơ bản.

Vấn đề thứ hai liên quan đến việc điều khiển ngữ nghĩa dữ liệu, đặc biệt là đối với việc kiểm tra toàn vẹn. Kết quả của việc phân mảnh là các tính chất liên quan đến tính phụ thuộc có thể bị phân rã thành các thành phần khác nhau và có thể bị phân phối tới các trạm khác nhau. Trong trường hợp này, thậm chí một tác vụ đơn giản như kiểm tra tính phụ thuộc cũng sẽ dẫn đến việc truy tìm dữ liệu trên một số trạm. Trong chương 6, chúng ta sẽ trở lại vấn đề kiểm tra ý nghĩa dữ liệu.

5.2.2. Các phương pháp phân mảnh

Biểu diễn quan hệ chính là các bảng, như vậy vấn đề là phải tìm cách để chia một bảng thành các bảng nhỏ hơn. Rõ ràng là có hai cách là chia theo chiều dọc và chia theo chiều ngang.

Ví dụ 5.1:

Trong chương này chúng tôi sử dụng một phiên bản đã sửa đổi của một mô hình csdl quan hệ được phát triển trong chương 2. Chúng tôi thêm một tính chất mới là quan hệ J (LOC) mà biểu thị nơi chốn của từng dự án. Hình 5.3 mô tả mô hình csdl mà chúng ta sẽ sử dụng. Hình 5.4 biểu diễn quan hệ J của hình 5.3 được chia theo chiều ngang thành 2 quan hệ. Quan hệ con J_1 bao gồm thông tin về các dự án có ngân sách nhỏ hơn $200.000 trong khi J_2 lưu thông tin về các dự án có ngân sách lớn hơn.

Ví dụ 5.2:

Hình 5.5 biểu diễn quan hệ J của hình 5.3 được chia theo chiều dọc thành hai quan hệ con, J_1 và J_2. J_1 chỉ bao gồm thông tin về ngân sách dự án, trong khi J_2 bao gồm tên dự án và địa điểm. Quan trọng là cần phải chú ý đến khoá quan hệ (JNO) được bao gồm trong cả 2 phân mảnh.

Tất nhiên là việc phân mảnh có thể lồng nhau. Nếu lồng các loại phân mảnh khác nhau, ta sẽ có phân mảnh hỗn hợp. Mặc dù chúng ta không coi phân mảnh hỗn hợp là một loại phương pháp phân mảnh chính, hiển nhiên là các quá trình phân mảnh trong thực tế thường là hỗn hợp.

5.2.3. Cấp độ phân mảnh

Quy mô csdl nào nên được phân mảnh là một quyết định quan trọng ảnh hưởng đến việc thực hiện xử lý truy vấn. Trên thực tế, các vấn đề trong phần 5.2.1 quan tâm đến lý do phân mảnh tạo thành một tập con các câu trả lời cho câu hỏi chúng tôi đang trình bày ở đây. Mức độ phân mảnh đi từ thái cực này, đó là hoàn toàn không phân mảnh, tới một thái cực kia, phân mảnh tới mức thành các tuple riêng lẻ (trong trường hợp phân mảnh theo chiều ngang) hoặc mức mức các tính chất riêng lẻ (trong trường hợp phân mảnh theo chiều dọc).

Chúng tôi đã đưa ra các ảnh hưởng trái ngược của các đơn vị rất lớn và rất nhỏ trong phân mảnh. Điều chúng ta cần là tìm ra một mức độ phân mảnh thích hợp cân bằng giữa 2 thái cực. Một mức độ như vậy chỉ có thể được xác định với các ứng dụng sẽ chạy trên csdl. Vấn đề là làm thế nào? Nói

chung, các ứng dụng cần phải được đặc trưng về số tham số. Theo các giá trị của những tham số này, các phân mảnh riêng lẻ có thể được xác định. Trong phần 5.3, chúng tôi mô tả làm thế nào việc đặc trưng này có thể được tiến hành đối với các phân mảnh chủ yếu.

5.2.4. Tính đúng đắn các nguyên tắc phân mảnh

Khi chúng ta nhìn vào sự bình thường hoá trong chương 2, chúng tôi nhắc tới một số các quy tắc để đảm bảo tính nhất quán của csdl. Quan trọng là cần phải lưu ý đến tính tương tự giữa phân mảnh dữ liệu để phân phối (đặc biệt là phân mảnh dọc) và bình thường hoá các mối quan hệ. Do đó các quy tắc phân mảnh tương tự như các nguyên tắc bình thường hoá có thể được xác định.

Chúng ta buộc phải tuân theo 3 quy tắc sau trong qua trình phân mảnh. Các quy tắc này đảm bảo rằng dữ liệu không bị thay đổi ý nghĩa trong phân mảnh.

1. Tính toàn vẹn

   Nếu một biểu diễn quan hệ R được phân rã thành các quan hệ R_1, R_2, …, R_n, mỗi phần dữ liệu được tìm thấy trong R cũng có thể được tìm thấy trong các R_i. Tính chất này, tương tự tính chất phân rã không mất trong bình thường hoá (chương 2), cũng rất quan trọng đối với phân mảnh vì nó đảm bảo rằng dữ liệu trong quan hệ toàn cục được mô tả trong các phân mảnh mà không có bất kỳ sự mất mát nào [Grant, 1984]. Chú ý là trong trường hợp phân mảnh ngan, một "phần" chính là một tuple, trong đó trong trường hợp phân mảnh dọc, nó là tính chất.

2. Cấu trúc lại

   Nếu một quan hệ R được phân rã thành các phân mảnh R_1, R_2, …, R_n, nó phải có thể xác định một toán tử quan hệ ∇ trong đó

   $$R = \nabla R_i, \qquad \forall R_i \in F_R$$

   Toán tử ∇ khác với các hình thái phân mảnh khác: tuy nhiên điều quan trọng là nó có thể được xác định. Tính chất cấu trúc lại quan hệ từ các phân mảnh của nó đảm bảo rằng các ràng buộc được xác định trên dữ liệu ở trạng thái phụ thuộc được duy trì.

3. Tính tách rời

   Nếu một quan hệ R được phân rã theo chiều ngang thành các phân mảnh R_1, R_2, … R_n và các phần dữ liệu d_i nằm trong R_j, nó không được nằm trong trong bất kỳ phân mảnh nào khác R_k ($k \neq j$). Điều kiện này đảm bảo rằng các phân mảnh ngang là tách rời nhau. Nếu quan hệ R được phân rã theo chiều dọc, tính chất khoá chính của nó được lặp lại trong tất cả các phân mảnh của nó. Do vậy, trong trường hợp phân mảnh dọc, tính tách rời được xác định chỉ ở trong các tính chất không phải khoá chính của quan hệ.

5.2.5. Các phương án phân phối

Giả sử csdl được phân mảnh hợp lý, người ta cần phải quyết định sự phân phối các phân mảnh này tới các trạm khác nhau trên mạng. Khi dữ liệu được phân phối, nó hoặc là được sao chép lại hoặc là được bảo trì như một bản ghi riêng lẻ. Lý do sao chép là do tính tin cậy và tính hiệu quả của các truy vấn chỉ đọc. Nếu có nhiều bản sao của một phần dữ liệu, sẽ có khả năng một số bản sao của dữ liệu sẽ có thể truy cập được thậm chí ở những nơi hệ thống hỏng. Hơn nữa các truy vấn chỉ đọc truy cập vào cùng một phần dữ liệu có thể được xử lý song song vì có nhiều bản sao tồn tại trên nhiều trạm. Mặt khác việc thực hiện các truy vấn cập nhật sẽ gặp rắc rối vì hệ thống phải đảm bảo rằng tất cả các bản sao của dữ liệu được cập nhật một cách chính xác. Như vậy quyết định liên quan tới việc sao chép cần sự cân bằng phụ thuộc vào tỉ lệ các truy vấn chỉ đọc so với các truy vấn cập nhật.

Quyết định này ảnh hưởng đến hầu như tất cả các thuật toán và hàm điều khiển trong hệ thống quản lý CSDL phân tán.

Một csdl không lặp (thường được gọi là csdl phân rã) bao gồm các phân mảnh nằm tại các trạm, và chỉ có một bản sao duy nhất của phân mảnh trên mạng. Trong trường hợp lặp, hoặc là csdl tồn tại trong thể toàn vẹn của nó tại từng trạm (csdl lặp hoàn toàn) hoặc là các phân mảnh được phân bổ tới các trạm theo một cách mà các bản sao của phân mảnh có thể nằm ở nhiều trạm (csdl lặp thành phần). Trong csdl lặp thành phần một số bản sao của một phân mảnh có thể là đầu vào của thuật toán phân phối hoặc một tham số quyết định mà giá trị của nó được quyết định bởi thuật toán. Hình 5.6 so sánh ba phương pháp lặp này về các hàm hệ thống quản lý CSDL phân tán khác nhau.

5.2.6. Thông tin yêu cầu

Một khía cạnh của thiết kế phân tán là có quá nhiều nhân tố tạo thành một thiết kế tối ưu. Tổ chức logic của csdl, vị trí của các ứng dụng, các tính chất truy nhập của ứng dụng tới csdl, và các tính chất của hệ thống máy tính tại từng trạm, tất cả đều có ảnh hưởng tới các quyết định phân phối. Điều này làm nó trở nên rất phức tạp tạo thành một vấn đề phân phối.

Thông tin cần cho thiết kế phân tán có thể được chia làm 4 mục: thông tin csdl, thông tin ứng dụng, thông tin mạng truyền thông và thông tin hệ thống máy tính. Hai phần sau về bản chất hoàn toàn là số lượng và được sử dụng trong các mô hình phân phối hơn là trong thuật toán phân mảnh. Chúng tôi không xét chi tiết chúng ở đây. Các thông tin yêu cầu chi tiết của sự phân mảnh và thuật toán phân phối được thảo luận trong các phần tương ứng.

5.3 PHÂN MẢNH

Trong phần này chúng tôi trình bày các phương pháp và thuật toán phân mảnh khác nhau. Như được nhắc tới ở trên, có 2 phương pháp phân mảnh cơ bản: theo chiều dọc và theo chiều ngang. Hơn nữa có khả năng phân mảnh lồng nhau trong mô hình hỗn hợp.

5.3.1. Phân mảnh theo chiều ngang

Như chúng tôi đã giải thích ở trên, phân mảnh theo chiều ngang chia mối quan hệ theo các tuple của nó. Vì vậy từng phân mảnh có một tập con các tuple của quan hệ. Có hai cách phân theo chiều ngang: cơ sở và dẫn xuất. Phân mảnh theo chiều ngang cơ sở của một quan hệ được thực hiện bằng cách sử dụng các tính chất được định nghĩa trên mối quan hệ đó. Trong khi đó, phân mảnh ngang dẫn xuất là sự phân chia mối quan hệ mà được tạo ra từ các tính chất được xác định trên một quan hệ khác.

Ở phần tiếp sau chúng tôi xét một thuật toán thực hiện cả hai cách phân mảnh này. Tuy nhiên đầu tiên chúng ta hãy nghiên cứu các thông tin cần cho việc tiến hành hoạt động phân mảnh theo chiều ngang.

Thông tin yêu cầu cho phân mảnh ngang

Thông tin cơ sở dữ liệu. Thông tin csdl quan tâm đến mô hình lý thuyết toàn cục. Trong nội dung này điều quan trọng là phải lưu ý đến việc làm thế nào các quan hệ csdl được kết nối với nhau, đặc biệt với chỗ giao nhau. Trong mô hình quan hệ, những mối quan hệ này cũng được mô tả như là các quan hệ. Tuy nhiên trong các mô hình dữ liệu khác, như là mô hình thực thể-quan hệ (E-R) [Chen, 1976], những mối quan hệ giữa các đối tượng csdl này được mô tả một cách rõ ràng. Trong [Ceri et al., 1983] mối quan hệ cũng được mô hình hoá một cách cụ thể, trong khung quan hệ, nhằm mục đích thiết kế phân tán. Trong giải thích sau này, các liên kết có hướng được vẽ giữa các quan hệ mà có liên quan với nhau bằng một phép tính equijoin.

Ví dụ 5.3:

Hình 5.7 biểu diễn biểu thức liên kết giữa các quan hệ cơ sở dữ liệu đã cho ở hình 2.4. Lưu ý là hướng của các liên kết biểu thị mối quan hệ một-nhiều. Chẳng hạn, đối với từng tiêu đề có nhiều nhân viên tại tiêu đề đó; vì vậy có một liên kết giữa quan hệ S và E. Dọc theo vẫn tuyến đó, mối quan hệ một-nhiều giữa E và J được biểu diễn bằng hai liên kết tới quan hệ G.

Các liên kết giữa các đối tượng csdl (nghĩa là các liên kết trong trường hợp của chúng ta) khá giống với những liên kết đã gặp trong các mô hình mạng của dữ liệu. Trong mô hình quan hệ chúng được xem như các đồ thị liên kết, mà chúng ta sẽ thảo luận chi tiết trong chương xử lý truy vấn. Chúng tôi trình bày chúng ở đây vì chúng giúp làm đơn giản hoá việc biểu diễn các mô hình phân tán mà chúng ta sẽ thảo luận sau đây.

Mối quan hệ ở cuối của một liên kết được gọi là người chủ của quan hệ và mối quan hệ ở đầu liên kết chúng ta gọi là thành viên [Ceri et al., 1983]. Trong khung quan hệ, các thuật ngữ được sử dụng phổ biến hơn là quan hệ nguồn đối với chủ và quan hệ đích với thành viên. Hãy định nghĩa hai hàm: chủ và thành viên, cả hai cung cấp việc ánh xạ từ tập các liên kết tới tập quan hệ. Do vậy, cho một liên kết, chúng sẽ trả về quan hệ chủ hoặc thành viên.

Thông tin lượng được yêu cầu về csdl là nhân tố của từng quan hệ R, thẻ được biểu thị R.

Thông tin ứng dụng. Như đã trình bày ở trên trong quan hệ ở hình 5.2, cả thông tin chất lượng và số lượng được ứng dụng yêu cầu. Thông tin chất lượng hướng dẫn hành động phân mảnh, trong khi thông tin số lượng được kết hợp chặt chẽ trong mô hình phân phối.

Thông tin chất lượng cơ bản bao gồm các tính chất được sử dụng trong các truy vấn người dùng. Nếu không thể phân tích tất cả các ứng dụng người dùng để xác định những tính chất này, ít nhất ta cũng phải nghiên cứu các ứng dụng "quan trọng" nhất. Người ta cho rằng giống như nguyên tắc ngón tay trái, 20% tài khoản truy vấn người dùng được kích hoạt thường xuyên nhất có thể được sử dụng như nguyên tắc chỉ đạo trong việc tiến hành phân tích này.

Tại điểm này chúng ta quan tâm việc xác định các tính chất đơn giản. Cho một quan hệ $R(A_1, A_2, \ldots, A_n)$, trong đó A_i là một tính chất được định nghĩa trên miền D_i, một tính chất đơn giản p_j được xác định trên R có dạng:

p_j: $A_i \ \theta$ Value

trong đó $\theta \in (=,<,\neq,\leq,>,\geq)$ và Value được chọn từ miền A_i (Value $\in D_i$). Chúng tôi sử dụng Pr_i để ký hiệu tập tất cả các tính chất đơn giản được xác định trên quan hệ R_i. Các thành viên của Pr_i được ký hiệu là p_{ij}.

Ví dụ 5.5

Cho biểu diễn quan hệ J của hình 5.3

PNAME = "Maintenance"

là một tính chất đơn giản, cũng như

BUDGET ≤ 200000

Thậm chí mặc dù các tính chất đơn giản khá dễ xử lý, các truy vấn người dùng thường bao gồm các tính chất phức tạp hơn, là sự phối hợp Boolean của các tính chất đơn giản. Một phối hợp mà chúng tôi đặc biệt lưu tâm, được gọi là tính chất minterm, là sự kết hợp của các tính chất đơn giản. Vì luôn có thể chuyển một biểu thức Boolean về dạng bình thường liên kết, việc sử dụng các tính chất minterm trong thuật toán thiết kế không gây ra bất kỳ sự mất mát nào nói chung.

Cho một tập $Pr_i = \{p_{i1}, p_{i2}, \ldots, p_{im})$ các tính chất đơn giản cho quan hệ R_i, tập minterm các tính chất $M_i=\{m_{i1}, m_{i2}, \ldots, m_{iz}\}$ được xác định:

$$M_i = \{m_{ij} \mid m_{ij} = \bigwedge_{p_{ik} \in Pr_i} p_{ik}^*\}, 1 \le k \le m, 1 \le j \le z$$

trong đó $p_{ik}^* = p_{ik}$ hoặc $p_{ik}^* = \neg p_{ik}$. Vậy mỗi tính chất đơn giản có thể xảy ra trong một tính chất minterm hoặc là trong hình thái ban đầu của nó hay hình thái phủ định của nó.

Cần phải lưu ý một điểm quan trọng ở đây. Việc đề cập tới thể phủ định của một tính chất có nghĩa là các tính chất tương đương có dạng

Tính chất = Giá trị

Đối với các tính chất không tương đương, thể phủ định có thể được coi như phần bù. Chẳng hạn, thể phủ định của tính chất đơn giản

Tính chất ≤ Giá trị

nghĩa là

Tính chất > Giá trị

Ngoài các vấn đề về lý thuyết về phần bù trong tập vô hạn, cũng có vấn đề về thực tế là phần bù có thể rất khó xác định. Chẳng hạn, nếu hai tính chất đơn giản có dạng:

Biên dưới ≤ Tính chất 1

Tính chất 1 ≤ Biên trên

được xác định, các phần bù sẽ là

¬(Biên dưới ≤ Tính chất 1)

và

¬(Tính chất 1 ≤ Biên trên)

Tuy nhiên ban đầu hai tính chất đơn giản có thể được viết như sau

Biên thấp ≤ Tính chất 1 ≤ Biên trên

với phần bù

¬(Biên thấp ≤ Tính chất 1 ≤ Biên trên)

rất khó xác định. Do đó, nghiên cứu trong lĩnh vực này thường chỉ xét các tính chất tương đương đơn giản ([Ceri et al., 1982a] và [Ceri và Pelagatti, 1984]).

Ví dụ 5.6

Xét quan hệ S của hình 5.3. Sau đây là một số các tính chất đơn giản có thể được xác định trên S

p_1: TITLE = “Elect. Eng.”

p_2: TITLE = “Syst. Arial.”

p_3: TITLE = “Mech. Eng.”

p_4: TITLE = “Programmer”

p_5: SAL ≤ 30000

p_6: SAL > 30000

Sau đây là một số tính chất minterm có thể được xác định dựa trên những tính chất đơn giản đó

m_1: TITLE = "Elect. Eng." $\wedge$ SAL $\leq$ 30000

m_2: TITLE = "Elect. Eng." $\wedge$ SAL > 30000

m_3: $\neg$(TITLE = "Elect. Eng.") $\wedge$ SAL $\leq$ 30000

m_4: $\neg$(TITLE = "Elect. Eng.") $\wedge$ SAL > 30000

m_5: TITLE = "Programmer" $\wedge$ SAL $\leq$ 30000

m_6: TITLE = "Programmer" $\wedge$ SAL > 30000

Có 2 điểm cần lưu ý ở đây. Đầu tiên đây không phải là tất cả tính chất minterm mà có thể được xác định; chúng tôi đang trình bày chỉ một ví dụ đại diện. Thứ hai, một số các tính chất này có thể vô nghĩa với các ý nghĩa của tập S. Chúng tôi không trình bày vấn đề đó ở đây. Thêm nữa, chú ý rằng m_3 có thể được viết như sau

m_3: TITLE $\neq$ "Elect. Eng." $\wedge$ SAL $\leq$ 30000

Đối với thông tin số lượng ứng dụng người dùng, chúng ta cần phải có hai tập dữ liệu

1. Chọn lọc tính chất minterm: số các tuple của quan hệ có thể được truy cập bởi một truy vấn người dùng được xác định cụ thể theo một tính chất minterm. Ví dụ, chọn lọc của m_1 các ví dụ 5.6 là 0 vì không có tuple trong S thoả mãn tính chất minterm. Mặt khác chọn lọc của m_3 là 1. Chúng tôi ký hiệu chọn lọc của 1 tính chất m_i là sel(m_i).
2. Tần số truy cập: tần số các ứng dụng người dùng truy cập dữ liệu. Nếu Q={q_1, q_2, …, q_q} là một tập truy vấn người dùng, acc(q_i) biểu thị tần số truy cập của truy vấn q_i trong một khoảng thời gian đã cho.

Lưu ý là tần số truy cập minterm có thể xác định từ tần số truy vấn. Chúng tôi ký hiệu tàn số truy cập của một minterm m_i là acc(m_i).

Phân mảnh ngang chính. Trước khi chúng tôi trình bày một thuật toán chính thức về phân mảnh ngang, chúng tôi sẽ thảo luận quá trình của phân mảnh ngang và phân mảnh dẫn xuất. Một phân mảnh ngang chính được xác định bằng một phép toán chọn trên các quan hệ chủ của một mô hình csdl. Như vậy, cho quan hệ R_i, các phân mảnh ngang của nó được tính theo

$$R_i^j = \sigma_{F_j}(R_i), 1 \leq j \leq w$$

trong đó F_j là công thức chọn được dùng để thu được phân mảnh R_i^j. Chú ý là nếu F_j nằm ở trạng thái nối tiếp bình thường, nó là một tính chất minterm (m_{ij}). Trong thực tế, thuật toán chúng tôi thảo luận sẽ biểu thị rằng F_j là một tính chất minterm.

Ví dụ 5.7

Sự phân tách quan hệ J thành các phân mảnh ngang J_1 và J_2 trong ví dụ 5.1 được xác định như sau:

$J_1 = \sigma_{BUDGET \leq 200000}(J)$

$J_2 = \sigma_{BUDGET > 200000}(J)$

Ví dụ 5.7 trình bày một trong những vấn đề của phân chia ngang. Nếu miền các tính chất tham gia vào các công thức chọn là tiếp diễn và vô hạn, như trong ví dụ 5.7, rất khó để xác định tập công

thức F = {F_1, F_2, …, F_n} mà sẽ phân mảnh quan hệ một cách hợp lý. Một tập hợp các hành động có thể để xác định các miền như chúng ta đã thực hiện trong ví dụ 5.7. Tuy nhiên luôn có vấn đề về xử lý 2 điểm cuối. Chẳng hạn, nếu một tuple mới với một giá trị BUDGET là, ví dụ, $600000 được chèn vào trong J, ta sẽ phải xem lại sự phân mảnh để xác định tuple mới để đưa vào J_2 hoặc nếu các phân mảnh cần được sửa lại và một phân mảnh mới cần được xác định:

$$J_2 = \sigma_{200000<BUDGET\leq400000}(J)$$

$$J_3 = \sigma_{BUDGET>400000}(J)$$

Vấn đề này trong thực tế hiển nhiên có thể giải quyết bằng cách giới hạn miền giá trị theo yêu cầu của ứng dụng.

Ví dụ 5.8

Xét quan hệ J của hình 5.3. Chúng ta có thể xác định các phân mảnh ngang sau đây dựa trên vị trí các dự án. Các phân mảnh đề cập được biểu diễn trong hình 5.6.

$$J_1 = \sigma_{LOC=\text{"Montreal"}}(J)$$

$$J_2 = \sigma_{LOC=\text{"New York"}}(J)$$

$$J_3 = \sigma_{LOC=\text{"Paris"}}(J)$$

(Hình 5.8)

Giờ chúng ta có thể xác định một phân mảnh ngang cẩn thận hơn. Một phân mảnh ngang R_i của quan hệ R bao gồm tất cả các tuple của R mà thoả mãn một tính chất minterm m_i. Vì vậy, cho một tập tính chất minterm M, có bao nhiêu tính chất minterm có bấy nhiêu phân mảnh ngang của quan hệ R. Tập các phân mảnh ngang này cũng thường được xem như tập các phân mảnh minterm.

Chúng tôi đã trình bày các ví dụ về sự phân mảnh ngang chính nhưng vẫn chưa đưa ra một thuật toán mà có thể cung cấp một phân mảnh được cho từ quan hệ R và một tập ứng dụng chạy trên nó. Theo các biện luận ở trên thì hiển nhiên là việc xác định các phân mảnh ngang phu thuộc vào các tính chất minterm. Như vậy bước đầu tiên của bất kỳ một thuật toán phân mảnh nào là xác định một tập các tính chất đơn giản với các thuộc tính xác định.

Một khía cạnh quan trong của các tính chất đơn giản là tính toàn vẹn của nó; một khía cạnh khác là tính tối thiểu của nó. Một tập các tính chất đơn giản Pr được nói là toàn vẹn nếu và chỉ nếu có khả năng cân bằng về truy cập của từng ứng dụng tới bất kỳ hai tuple nào thuộc về bất kỳ phân mảnh nào mà được xác định theo Pr. Do đó rõ ràng là định nghĩa về tính toàn vẹn của một tập các tính chất đơn giản là khác biệt so với nguyên tắc toàn vẹn của sự phân mảnh đã cho trong phần 5.2.4.

Ví dụ 5.9

Xét sự phân mảnh quan hệ J đã cho trong ví dụ 5.8. Nếu duy nhất một ứng dụng truy cập J muốn truy cập các tuple theo vị trí, tập là toàn vẹn vì mỗi tuple của từng phân mảnh J_i (ví dụ 5.8) có cùng khả năng bị truy cập. Tuy nhiên, nếu có một ứng dụng thứ hai chỉ truy cập các tuple dự án mà có kinh phí nhỏ hơn $200000 vậy thì Pr không toàn vẹn. Một số các tuple trong từng J có khả năng bị truy cập cao hơn do ứng dụng thứ hai này. Để làm cho tập các tính chất toàn vẹn, chúng ta cần phải thêm (BUDGET ≤ 200000, BUDGET > 200000) vào Pr:

Pr = {LOC = “Montreal”, LOC = “New York”, LOC = “Paris”,
BUDGET ≤ 200000, BUDGET > 200000}

Lý do tính toàn vẹn là một thuộc tính cần thiết là bởi vì tập các tính chất toàn vẹn cho phép chúng ta xác định một tập tính chất minterm mà theo đó phân mảnh ngang chính có thể được tiến hành. Các phân mảnh thu được theo cách này không chỉ đồng đều về logic trong đó tất cả chúng đều thoả mãn tính chất minterm, mà còn đồng nhất về số.

Có thể xác định tính toàn vẹn một cách chính thức để có thể tự động nhận được tập các tính chất toàn vẹn. Tuy nhiên điều này yêu cầu nhà thiết kế phải xác định khả năng truy cập cho từng tuple của một quan hệ đối với từng ứng dụng được xem xét. Việc này yêu cầu cảm nhận và kinh nghiệm của nhà thiết kế để tạo ra một tập toàn vẹn. Nói một cách ngắn gọn, chúng tôi sẽ trình bày một thuật toán để thu được tập này.

Tính chất cần có thứ hai của tập các tính chất, theo đó các tính chất minterm và các phân mảnh được xác định, là khả năng tối thiểu, mà phụ thuộc rất nhiều vào trực giác. Điều này chỉ ra rằng nếu một tính chất ảnh hưởng tới việc phân mảnh được tiến hành như thế nào (nghia là làm cho một mảnh f có thể được phân mảnh nhỏ hơn nữa, như là f_i và f_j), sẽ có ít nhất một ứng dụng mà truy cập f_i và f_j một cách khác nhau. Nói cách khác, tính chất đơn giản phải có liên quan với việc xác định một phân mảnh. Nếu tất cả các tính chất của tập Pr là liên quan thì Pr là nhỏ nhất.

Một định nghĩa chính thức của quan hệ có thể được đưa ra như sau [Ceri et al., 1982a]. Cho m_i và m_j là hai tính chất minterm xác định theo định nghĩa của chúng, ngoại trừ m_i bao gồm tính chất p_i trong thê ban đầu của nó trong khi m_j bao gồm $\neg p_i$. Cũng thế, cho f_i và f_j là hai mảnh lần lượt được xác định theo m_i và m_j. Vậy thì p_i là liên quan nếu và chỉ nếu

$$\frac{acc(m_i)}{card(f_i)} \neq \frac{acc(m_j)}{card(f_j)}$$

Một lần nữa chúng tôi quan tâm tới trực giác và chuyên môn của nhà thiết kế hơn là áp dụng định nghĩa chính thức.

Ví dụ 5.10

Tập Pr được xác định trong ví dụ 5.9 là toàn vẹn và tối thiểu. Tuy nhiên nếu chúng ta thêm tính chất

JNAME = “Instrumentation”

vào Pr, tập thu được sẽ không còn là tối thiểu vì tính chất mới là không liên quan với Pr. Không có ứng dụng nào sẽ truy cập vào các phân mảnh kết quả một cách khác nhau.

Chúng ta có thể xét một thuật toán lặp mà tạo ra một tập toàn vẹn và tối thiểu Pr’ từ tập các tính chất đơn giản Pr. Thuật toán này, gọi là COM_MIN, được trình bày trong Thuật toán 5.1. Để tránh trình bày dài dòng, chúng tôi đã chấp nhận các định nghĩa sau:

Quy tắc 1: qua tắc cơ bản của tính toàn vẹn và tối thiểu, là một quan hệ hoặc mảnh được phân thành “ít nhất hai phần được truy cập một cách khác nhau bởi ít nhất một ứng dụng”.

f_i và Pr’: mảnh f_i được xác định theo một tính chất minterm được định nghĩa trên các tính chất của Pr’.

Thuật toán 5.1 COM_MIN

input: R: relation; Pr: set of simple predicates

output: Pr’: set of simples predicates

declare

F: set of minterm fragments

```
begin
    find a p_i ∈ Pr such that p_i partitions R according to Rule 1
    Pr' - p_i
    Pr – Pr - p_i
    F – f_i          {f_i is the minterm fragment according to p_i}
    do
        begin
            find a p_j ∈ Pr such that p_j partitions some f_k of Pr' according to Rule 1
            Pr' – Pr' ∪ p_j
            Pr – Pr - p_j
            F – F ∪ f_j
            if ∃p_k ∈ Pr' which is nonrelevant then
            begin
                Pr' – Pr' – p_k
                F – F – f_k
            enf-if
        end-begin
    until Pr' is complete
end. {COM_MIN}
```

Thuật toán bắt đầu bằng việc tìm một tính chất mà có liên quan và phân tách quan hệ. Vòng lặp do-until từ từ thêm các tính chất vào tập này, đảm bảo tính tối thiểu trong từng bước. Như vậy cuối cùng tập Pr' có cả tính toàn vẹn và tính tối thiểu.

Bước thứ hai trong quá trình thiết kế ngang chính là tìm tập các tính chất minterm có thể được xác định trên các tính chất trong tập Pr'. Những tính chất minterm này quyết định các phân mảnh được sử dụng trong bước phân phối tiếp sau. Việc quyết định các tính chất minterm riêng lẻ không quan trọng lắm: khó khăn là tập các tính chất minterm có thể là rất lớn (trong thực tế, là luỹ thừa của số các tính chất đơn giản). Trong bước tiếp theo, chúng tôi tìm các cách để giảm số các tính chất minterm mà cần được xét trong sự phân mảnh.

Bước thứ ba của quá trình thiết kế là huỷ một số các phân mảnh minterm vô nghĩa. Việc huỷ được thực hiện bằng cách xác định những minterm có thể mâu thuẫn với tập quan hệ I. Ví dụ, nếu Pr' = $\{p_1, p_2\}$, trong đó

p_1 : att = value_1

p_2 : att = value_2

và miền của att là {value_1, value_2}, hiển nhiên là I bao gồm hai quan hệ, được biểu thị

i_1: (att = value_1) => ¬(att = value_2)

i_2: ¬(att = value_1) => (att = value_2)

Bốn tính chất minterm sau đây được xác định theo Pr':

m_1: (att = value_1) $\wedge$ (att = value_2)

m_2: (att = value_1) $\wedge \neg$(att = value_2)

m_3: $\neg$(att = value_1) $\wedge$ (att = value_2)

m_4: $\neg$(att = value_1) $\wedge \neg$(att = value_2)

Trong trường hợp này các tính chất minterm m_1 và m_4 là mâu thuẫn với quan hệ I và do đó có thể bị xoá khỏi M.
Thuật toán phân mảnh ngang chính được trình bày trong thuật toán 5.2. Đầu vào của thuật toán PHORIZONTAL là một quan hệ R_i. Quan hệ này là đối tượng cho phân mảnh ngang chính, và Pr_i là tập các tính chất đơn giản đã được xác định theo các ứng dụng được định nghĩa trong quan hệ R_i.

Thuật toán 5.2 PHORIZONTAL

input: R_i: relation; Pr_i: set of simple predicates

output: M_i: set of minterm fragments

begin

Pr_i' - COM_MIN(R_i, Pr_i)

determine the set M_i of minterm predicates

determine the set I_i of implications among $p_i \in Pr'_i$

for each $m_i \in M_i$ do

if m_i is contradictory according to I then

$M_i - M_i - m_i$

end if

end for

end. {PHORIZONTAL}

Ví dụ 5.11

Giờ chúng ta xét thiết kế của mô hình csdl đã cho trong hình 5.7. Điều đầu tiên cần lưu ý là hai quan hệ mà là đối tượng cảu phân mảnh ngang chính: quan hệ S và J.

Giả sử rằng chỉ có duy nhất một ứng dụng truy cập S. Ứng dụng đó kiểm tra thông tin lương và quyết định tăng lương. Giả sử bản ghi nhân viên được quản lý ở hai nơi: một nới xử lý các bản ghi lương nhỏ hơn hoặc bằng $30000, và nơi kia xử lý các bản ghi có lương lớn hơn $30000. Do vậy truy vấn được đưa ra ở hai trạm

Các tính chất đơn giản có thể được dùng để phân chia quan hệ S là:

p_1: SAL $\leq$ 30000

p_2: SAL > 30000

Như vậy cho tập ban đầu các tính chất đơn giản Pr = {p_1, p_2}. Ứng dụng thuật toán COM_MIN chỉ ra rằng Pr là toàn vẹn và tối thiểu. Như thế, Pr' = Pr.

Chúng ta có thể hình thành các tính chất minterm sau như là số hạng của tập M:

m_1: (SAL ≤ 30000) ∧ (SAL > 30000)

m_2: (SAL ≤ 30000) ∧ ¬(SAL >30000)

m_3: ¬(SAL ≤ 30000) ∧ (SAL > 30000)

m_4: ¬(SAL > 30000) ∧ ¬(SAL > 30000)

Giả sử răng miền SALARY có thể được chia thành hai, như giả thiết bởi p_1 và p_2, các quan hệ sau là hiển nhiên:

i_i: (SAL ≤ 30000) => ¬(SAL > 30000)

i_2: ¬(SAL ≤ 30000) => (SAL >30000)

i_3: (SAL > 30000) => ¬(SAL ≤ 30000)

i_4: ¬(SAL > 30000) => (SAL ≤ 30000)

Theo i_1, tính chất minterm m_1 mâu thuẫn; theo i_2, m_4 mâu thuẫn. Do đó, chúng ta có M = {m_2, m_3}.

i_1 và i_4 cùng giảm đặc điểm của m_2 trong p_1 và đồng thời dẫn đến sự giảm của m_3 tới p_2 do i_2 và i_3. Như vậy chúng ta xác định hai mảnh F_i = {S_1, S_2} theo M (Hình 5.9)

Tiếp đó chúng ta xét quan hệ j. Giả sử có hai ứng dụng. Ứng dụng đầu tiên bắt đầu tại ba trạm và tìm tên và ngân quỹ của dự án. Trong cú pháp SQL truy vấn được viết

```
SELECT    JNAME, BUDGET
FROM      J
WHERE     JNO=value
```

Đối với ứng dụng này, các tính chất đơn giản sẽ được sử dụng như sau:

p_1: LOC = “Montreal”

p_2: LOC = “New York”

p_3: LOC = “Paris”

Ứng dụng thứ hai bắt đầu tại hai trạm và phải tính hành với việc quản lý dự án. Những dự án có ngân sách nhỏ hơn $200000 được quản lý ở một trạm trong khí những dự án lớn hơn được quản lý tại trạm thứ 2. Do đó các tính chất đơn giản được sử dụng cho phân mảnh theo ứng dụng thứ hai sẽ là:

p_4: BUDGET ≤ 200000

p_5: BUDGET > 200000

Nếu thuật toán COM_MIN được áp dụng, tập Pr’ = {p_1, p_2, p_3, p_4, p_5} hiển nhiên là toàn viện và nhỏ nhất.

Dựa trên Pr’, sau tính chất minterm hình thành nên M có thể được xác định như sau:

m_1: (LOC = “Montreal”) ∧ (BUDGET ≤ 200000)

m_2: (LOC = “Montreal”) ∧ (BUDGET > 200000)

m_3: (LOC = “New York”) ∧ (BUDGET ≤ 200000)

m_4: (LOC = “New York”) ∧ (BUDGET > 200000)

m_5: (LOC = “Paris”) $\wedge$ (BUDGET $\leq$ 200000)

m_6: (LOC = “Paris”) $\wedge$ (BUDGET > 200000)

Đây không chỉ là các tính chất minterm có thể phát sinh ra. Chẳng hạn có thể xác định các tính chất có dạng:

$$p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5$$

Tuy nhiên, các quan hệ hiển nhiên:

$$i_1 : p_1 \Rightarrow \neg p_2 \wedge \neg p_3$$
$$i_2 : p_2 \Rightarrow \neg p_1 \wedge \neg p_3$$
$$i_3 : p_3 \Rightarrow \neg p_1 \wedge \neg p_2$$
$$i_4 : p_4 \Rightarrow \neg p_5$$
$$i_5 : p_5 \Rightarrow \neg p_4$$
$$i_6 : \neg p_4 \Rightarrow p_5$$
$$i_7 : \neg p_5 \Rightarrow p_4$$

huỷ các tính chất minterm và chúng ta còn lại m_1 và m_6.

Xét csdl trong hình 5.3, ta có thể chỉ ra các liên hệ sau:

i_8: LOC = “Montreal” => ¬(BUDGET > 200000)

i_9: LOC = “Paris” => ¬(BUDGET ≤ 200000)

i_{10}: ¬(LOC = “Montreal”) => BUDGET ≤ 200000

i_{11}: ¬(LOC = “Paris”) => BUDGET > 200000

Tuy nhiên cần phải nhơ rằng các quan hệ cần phải được định nghĩa theo ý nghĩa của csdl chứ không phải theo dữ liệu hiện thời. Một số các phân mảnh được xác định theo M = {m_1, …, m_6} có thể rỗng, nhưng dù sao chúng là các phân đoạn. Không có gì trong ý nghĩa csdl biết các quan hệ i_8 đến i_{11} có gì.

Kết quả của việc phân mảnh ngang chính của J là để hình thành sau phân mảnh F_J = {J_1, J_2, J_3, J_4, J_5, J_6} của quan hệ J theo các tính chất minterm M (hình 5.10). Chúng tôi cũng lưu ý rằng một số các phân mảnh rỗng và do đó không được hiển thị trong hình 5.10.

Phân mảnh ngang dẫn xuất. Một phân mảnh ngang dẫn xuất được xác định trên một quan hệ thành viên của một liên kết theo một công thức chọn lọc cụ thể trên chủ. Quan trọng là phải nhớ hai điểm. Đầu tiên, liên kết giữa chủ và quan hệ thành viên được xác định là liên kết ngang hàng. Thứ hai, một liên kết ngang hàng có thể được thực hiện bởi các bán liên kết. Điểm thứ hai này là đặc biệt quan trọng đối với mục đích của chúng tôi, vì chúng tôi muốn phân chia một quan hệ thành viên theo các phân mảnh của chủ của nó, nhưng chúng tôi cũng muốn phân mảnh tạo ra được định nghĩa chỉ trên các tính chất của quan hệ thành viên.

Theo đó, cho một liên kết L với owner(L) = S và member(L) = R, các phân mảnh ngang dẫn xuất của R được xác định như sau:

(công thức)

trong đó w là số cực đại các phân mảnh sẽ được xác định trên R, và $S_i = \sigma_{Fi}(S)$, trong đó F_i là công thức xác định phân mảnh ngang chính S_i.

Ví dụ 5.12

Xét liên kết L_1 trong hình 5.7, trong đó owner(L_1) = S và member(L_1) = E. Vậy chúng ta có thể nhóm các kỹ sư thành hai nhóm theo lương của họ: những người có lương nhỏ hơn hoặc bằng $30000 và những người có lương lớn hơn 30000. Hai phân mảnh E_1 và E_2 được xác định như sau:

(công thức)

trong đó

(công thức)

Kết quả của phép phân mảnh này được mô tả trong hình 5.11.

(Hình)

Để thực hiện phân mảnh ngang dẫn xuất, cần có 3 đầu vào: tập các thành phần của quan hệ chủ (chẳng hạn S_1 và S_2 trong ví dụ 5.12), quan hệ thành viên, và tập các tính chất bán liên kết giữa chủ và thành viên (như là E.TITLE = S.TITLE trong ví dụ 5.12). Thuật toán phân mảnh khá bìh thường, do vậy chúng tôi không trình bày chi tiết.

Có một vấn đề rắc rối cần chú ý. Trong một mô hình csdl, thường có nhiều hơn hai liên kết trong một quan hệ R (ví dụ như trong hình 5.7, G có hai liên kết từ trong ra). Trong trường hợp này có khả năng có nhiều phân mảnh ngang dẫn xuất của R. Quyết định phân mảnh nào được chọn dựa trên hai điều kiện:

1. Phân mảnh có các đặc điểm liên kết tốt hơn
2. Phân mảnh được sử dụng trong nhiều ứng dụng hơn

Chúng ta hãy thảo luận điều kiện thứ hai trước. Điều này khá rõ ràng nếu chúng ta xét tần số một ứng dụng truy cập vào dữ liệu. Nếu có thể, người ta sẽ cố gắng làm cho việc truy cập của người dùng "nặng" thuận tiên hơn để cho toàn bộ ảnh hưởng của họ trên hệ thống đạt mức nhỏ nhất.

Tuy nhiên việc áp dụng điều kiện đầu tiên không dễ dàng như vậy. Chẳng hạn, xét phân mảnh chúng ta đã thảo luận trong ví dụ 5.12. Hiệu quả (và đối tượng) của phân mảnh này là liên kết của quan hệ E và S để tra lời truy vấn được yêu cầu (1) bằng cách thực hiện nó trong một quan hệ nhỏ hơn (nghĩa là các phân mảnh) và (2) bằng cách thực hiện liên kết trong một hình thức phân tán.

Quan điểm đầu tiên là hiển nhiên. Các phân mảnh của E nhỏ hơn E. Do đó, nó sẽ liên kết với các phân mảnh của S nhanh hơn so với làm việc với chính quan hệ đó. Tuy nhiên quan điểm thứ hai quan trong hơn và là trung tâm của csdl phân tán. Nếu, ngoài việc xử lý một số các truy vấn ở các trạm khác nhau, chúng ta có thể xử lý một truy vấn song song, người ta mong thời gian phản hồi hoặc lượng dữ liệu của hệ thống sẽ được cải thiện. Trong trường hợp các liên kết, điều này có thể xảy ra dưới một số tính trạng cụ thể. Chẳng hạn, xét đồ thị quan hệ (nghĩa là các liên kết) giữa phân mảnh E và S trong ví dụ 5.10 (hình 5.12). Chỉ có duy nhất một liên kết đến và đi ra ngoài một phân mảnh. Một đồ thị liên kết như vậy được gọi là đồ thị đơn giản. Lợi thế của thiết kế trong đó quan hệ liên kết giữa các phân mảnh là đơn giản là chủ và thành viên của liên kết có thể được phân phối tới một trạm và các liên kết giữa các cặp phân mảnh khác nhau có thể được xử lý độc lập và song song.

(hình)

Thật không mayl, thu được các đồ thị liên kết đơn giản không phải lúc nào cũng có thể thực hiện được. Trong trường hợp đó, phương án tiếp theo là có một thiết kế tạo ra một đồ thị liên kết phân

chia. Một đồ thị phân chia bao gồm hai hoặc nhiều các đồ thị con không liên kết với nhau. Các phẩm mảnh thu được có thể không phân tán đối với xử lý song song dễ dàng như những phân mảnh thu được thông qua đồ thị liên kết đơn giản, nhưng việc phân phối vẫn có thể thực hiện được.

Ví dụ 5.13

Chúng ta hãy tiếp tục với thiết kế phân tán csdl chúng ta đã bắt đầu trong ví dụ 5.12. Chúng tôi đã quyết định phân chia quan hệ E theo sự phân chia của quan hệ S (ví dụ 5.12). Giờ chúng ta xét đến G. Giả sử có hai ứng dụng như sau:

1. Ứng dụng đầu tiên tìm tên của các kỹ sử làm việc tại một số nơi nhất định. Nó chạy trên cả 3 trạm và truy nhập trhông tin về các kỹ sư làm việc trong các dự án cục bộ có khả năng cao hơn các dự án ở các địa phương khác.
2. Tại từng trạm điều hành nơi các bản ghi nhân viên được quản lý, người dùng muốn truy cập các dự án mà các nhân viên này làm việc và tìm hiểu xem họ sẽ làm việc với các vấn đề này trong bao lâu.

Ứng dụng đầu tiên tạo ra sự phân mảnh G theo các mảnh J_1, J_3, J_4, và J_6 của J thu được trong ví dụ 5.11. Hãy nhớ là:

(công thức)

Do đó, phân mảnh dẫn xuất của G theo $\{J_1, J_2, J_3\}$ được xác định như sau:

(công thức)

Những mảnh này được biểu diễn trong hình 5.13.

Truy vấn thứ hai được trình bày bằng SQL như sau:

```
SELECT RESP, DUR
FROM G, E
WHERE G.END = E.END;
```

trong đó i =1 và i = 2 phụ thuộc vào việc truy vấn được yêu cầu ở trạm nào. Phân mảnh dẫn xuất của G theo phân mảnh của E được xác định ở dưới và được biểu diễn trong hình 5.14.

(công thức)

Ví dụ trên trình bày hai điều:

1. Phân mảnh dẫn xuất có thể theo chuỗi trong đó một quan hệ bị phân mảnh như một kết quả của thiết kế khác và nó đồng thời cũng làm cho quan hệ khác phân mảnh theo (ví dụ như chuỗi S-E-G).
2. Điển hình là sẽ có nhiều phân mảnh để lựa chọn hơn đối với một quan hệ (chẳng hạn quan hệ G). Lựa chọn mô hình phân mảnh cuối cùng có thể là một vấn đề quyết định được đưa ra trong khi phân phối.

(hình 5.13 và 5.14)

Kiểm tra tính đúng đắn. Giờ chúng ta sẽ kiểm tra thuật toán phân mảnh được thảo luận từ trước tới giờ theo 3 điều kiện về tính đúng đắn đã trình bày trong phân 5.2.4.

Tính toàn vẹn. Tính toàn vẹn của phân mảnh ngang chính dựa trên các tính chất lựa chọn được sử dụng. Chừng nào các tính chất lựa chọn là toàn vẹn, phân mảnh được tạo ra cũng đảm bảo

tính toàn vẹn. Vì cơ sở của thuật toán phân mảnh là một tập các tính chất toàn vẹn và tối thiểu, tính toàn vẹn được bảo đảm chừng nào ta không phậm sai lầm nào trong việc xác định Pr'.

Tính toàn vẹn của phân mảnh ngang dẫn xuất khó xác định hơn. Điều khó khăn là các tính chất trong thực tế quyết định phân mảnh liên quan đến hại quan hệ. Đầu tiên chúng ta hãy xác định quy tắc toàn vẹn và sau đó xét một ví dụ.

Cho R là quan hệ thành viên của một liên kết có chủ là quan hệ S, được phân thành $F_S = \{S_1, S_2, \ldots, S_w\}$. Hơn nữa, cho A là một tính chất liên kết giữa R và S. Vậy thì với mỗi tuple t thuộc R, sẽ có một tuple t' thuộc S mà

(công thức)

Chẳng hạn sẽ không có G tuple mà có một số dự án không được bao gồm trong J. Tương tự, sẽ không có E tuple với giá trị TITLE trong khi giá trị TITLE tương tự không xuất hiện trong S. Quy tắc này được biết như là tính toàn vẹn liên quan và đảm bảo rằng các tuple của bất kỳ phân mảnh nào của quan hệ thành viên cũng nằm trong quan hệ chủ.

Cấu trúc lại. Sự cấu trúc lại một quan hệ toàn cục từ các phân mảnh của nó được thực hiện bằng phép toàn hợp nhất trong cả phân mảnh ngang chính và dẫn xuất. Do vậy đối với một quan hệ R với phân mảnh $F_R = (R_1, R_2, \ldots, R_w\}$,

(công thức)

Không liên kết. Tạo ra tính không liên kết của sự phân mảnh cho phân mảnh ngang chính dễ hơn cho phân mảnh ngang dẫn xuất. Trong trường hợp phân mảnh ngang chính, sự không liên kết được bảo đảm chừng nào các tính chất minterm quyết định sự phân mảnh là duy nhất.

Trong phân mảnh dẫn xuất, có một bán liên kết làm cho phức tạp hơn. Tính không liên kết được đảm bảo nếu đồ thị liên kết là đơn giản. Nếu nó không đơn giản, ta cần phải điều tra các giá trị tuple thực tê. Nói chúng, chúng ta không muốn một tuple của một quan hệ thành viên liên kết với hai hoặc nhiều các tuple của quan hệ chủ trong khi những tuple này nằm trong các phân mảnh khác của chủ. Điều này có lẽ khó thực hiện, và chứng tỏ tại sao người ta luôn mong muôn các mô hình phân mảnh dã xuất tạo ra một đồ thị liên kết đơn giản.

Ví dụ 5.14

Trong việc phân mảnh quan hệ S (ví dụ 5.11), các tính chất minterm $M = \{m_1, m_2\}$ là

(công thức)

Vì m_1 và m_2 là duy nhất, sự phân mảnh của S là không liên quan.

Tuy nhiên đối với quan hệ E chúng ta cần:

1. Mỗi kỹ sư phải có một danh hiệu riêng
2. Mỗi danh hiệu có một giá trị lương liên quan tới nó.

Vì ý nghĩa của csdl tuân theo hai quy tắc này, phân mảnh của E so với S cũng không liên quan.

5.3.2. Phân mảnh dọc

Nhớ là phân mảnh dọc của một quan hệ R tạo ra các phânmảnh $R_1, R_2, \ldots R_n$ trong đó từng mảnh bao gồm một tập con các tính chất của R cúng như là khoá chính của R. Mục đích của phân mảnh dọc là chí một quan hệ thành một tập các quan hệ nhỏ hơn để nhiều ứng dụng người dùng sẽ chỉ chạy trên duy nhất một phân mảnh. Từ đó, một phân mảnh "tối ưu" là phân mảnh mà tạo ra một mô

hình phân mảnh làm giảm tối thiểu thời gian xử lý của ứng dụng người dung chạy trên các mảnh này.

Phân mảnh dọc đã được nghiên cứu trong nội dung của hệ thống csdl tập trung cũng như là trong các hệ thống phân tán. Mục đích của nó trong hệ thống tập trung là một công cụ thiết kế cho phép truy vấn người dùng xử lý các quan hệ nhỏ hơn, và dẫn tới số trang bị truy cập nhỏ hơn [Navathe et al., 1984]. Người ta cũng cho rằng các quan hệ con linh hoạt nhất có thể được xác định và đặt trong các hệ thống con bộ nhớ nhanh hơn trong trường hợp cơ cấu bộ nhớ được hỗ trợ [Eisner và Severance, 1976].

Phân chia theo chiều dọc phức tạp hơn là phân chia ngang. Đó là do tổng số các phương án hiện có. Chẳng hạn, trong phân chia ngang, nếu tổng số các tính chất đơn giản trong Pr là n, có 2^n các tính chất minterm có thể được xác định trên nó. Thêm nữa, chúng ta biết là một số các tính chất này sẽ mâu thuẫn với tính chất có sẵn, do đó làm giảm khá nhiều các mảnh cần xem xét. Tuy nhiên trong trường hợp phân chia dọc, nếu một quan hệ có m các tính chất không phải khoá chính, số các mảnh có thể có bằng B(m), mà là số Bell thứ m [Niamir, 1978]. Đối với các giá trị lớn hơn của m, $B(m) \approx m^m$; chẳng hạn với m = 10, B(m) ≈ 115000, với m = 15, B(m) ≈ 10^9, với m = 30, B(m) = 10^{23} ([Hammer và Niamir, 1979] và [Navathe et al, 1984]).

Những giá trị này chỉ ra rằng nỗ lực nhằm đặt được giải pháp tối ưu đối với các vấn đề chia dọc là vô ích; việc này cần sử dụng cách tự khám phá. Hai phương pháp tự khám phá đối với việc phân mảnh dọc các quan hệ toàn cục là:

1. Nhóm: bắt đầu bằng việc gán từng giá trị cho một mảnh, và ở từng bước, liên kết một số các mảnh cho đến khi thoả mãn một số điều kiện. Lập nhóm đầu tiên được đưa ra trong [Hammer và Niamir, 1979] cho csdl tập trung, và sau đó được sử dụng trong [Sacca và Wiederhold, 1985] cho csdl phân tán.
2. Chia nhỏ: bắt đầu bằng một quan hệ và quyết định phân chia có lợi dựa trên hành vi truy cập của ứng dụng tới tính chất. Kỹ thuật này đầu tiên được thảo luận trong thiết kế csdl tập trung trong [Hofler và Severance, 1975]. Sau đó nó được mở rộng cho môi trường phân tán trong [Navathe et al., 1984].

Trong đây chúng tôi chỉ thảo luận kỹ thuật chia nhỏ, vì nó phù hợp với phương pháp thiết kế từ trên xuống hơn, và như được trình bày trong [Navathe et al., 1984], vì giải pháp tối ưu có thể gần hơn với quan hệ đầy đủ hơn là so với tập phân mảnh mà từng mảnh bao gồm một tính chất đơn giản. Hơn nữa chia nhỏ tạo ra các mảnh không chồng lên nhau trong khi đó việc nhóm thường tạo ra các mảnh chồng nhau. Trong phạm vi các hệ thống csdl phân tán, chúng tôi quan tâm tới các mảnh không chồng. Tất nhiên, không chồng nhau đề cập tới các tính chất không phải khoá chính.

Trước khi chúng ta tiếp tục, hãy phân loại một vấn đề mà chúng ta mới chỉ nhắc tới trong ví dụ 5.2, gọi là việc lặp khoá của quan hệ toàn cục trong các mảnh. Đây là một đặc trưng của phân mảnh dọc cho phép cấu trúc lại quan hệ toàn cục. Do đó, chia nhỏ được coi như là duy nhất cho các tính chất này mà không liên quan tới khoá chính.

Có một lợi ích lớn trong việc lặp lại các tính chất khoá bỏ qua các vấn đề chúng gây ra. Lợi thế này có liên quan tới việc tạo quan hệ bắt buộc được thảo luận trong chương 6. Lưu ý là mọi sự phụ thuộc được trình bày trong chương 2 trên thực tế là ràng buộc cần phải tuân theo trong số các giá trị tính chất chất của quan hệ ở bất kỳ thời điểm nảo. Cũng cần phải nhớ rằng hầu hết sự phụ thuộc này liên quan tới các tính chất khoá của quan hệ. Nếu chúng ta thiết kế csdl để các tính chất khoá là một phần của một mảnh được phân phối tới một trạm, và các tính chất được đề cập tới là một phần của phân mảnh khác mà được phân phối tới trạm thứ hai, mọi yêu cầu cập nhật dẫn tới việc kiểm tra toàn bộ sẽ cần có sự giao tiếp giữa các trạm. Việc lặp các tính chất khoá ở từng mảnh giảm khả

năng xảy ra điều này nhưng không hoàn toàn xoá bỏ nó, vì những sự giao tiếp như vậy có thể là cần thiết do các ràng buộc về toàn vẹn không liên quan đến khoá chính, cũng như là do điều khiển dòng.

Một phương pháp để lặp các tính chất khoá là sử dụng bộ xác nhận tuple (TID). Đó là tuple của quan hệ có các giá trị duy nhất được gán cho hệ thống. Vì TID được bảo trì bởi hệ thống, các mảnh là không liên kết cho đến khi người sử dụng quan tâm tới.

Yêu cầu thông tin của phân mảnh dọc. Thông tin chính được yêu cầu đối với phân mảnh dọc liên quan tới các ứng dụng. Do đó thảo luận sau đây dựa trên những điều cần được xác định về các ứng dụng chạy dựa vào csdl phân tán. Vì việc phân dọc đặt trong một mảnh những tính chất thường được truy cập đồng thời, cần một số phương pháp xác định chính xác hơn khái niệm "sự đồng thời". Phương pháp này là sự tương đồng của các tính chất, biểu thị các tính chất quan hệ với nhau chặt chẽ đến mức nào. Thật không may, việc mong đợi người thiết kế hoặc người dùng có thể xác định điều này một cách dễ dàng đúng là một điều rất khó. Giờ chúng tôi sẽ trình bày một cách mà có thể thu được chúng từ dữ liệu ban đầu.

Yêu cầu dữ liệu chính liên quan tới các ứng dung là tần số truy cập của chúng. Cho Q = $\{q_1, q_2, \ldots, q_q\}$ là một tập các truy vấn người dùng (ứng dụng) mà chạy trên quan hệ $R(A_1, A_2, \ldots, A_n\}$. Sau đó, với từng truy vấn q_i và từng tính chất A_j, chúng ta kết hợp một giá trị sử dụng tính chất, được ký hiệu là use(q_i, A_j) và được xác định như sau:

(công thức)

Các vector use(q_i, •) đối với từng ứng dụng rất dễ xác định nếu nhà thiết kế biết rõ ứng dụng chạy trên csdl. Cũng cần nhớ quy tắc 80-20 được thảo luận trong phần 5.3.1 rất có ích trong công việc này.

Ví dụ 5.15

Xét quan hệ J của hình 5.3. Giả sử các ứng dụng sau đây được xác định để chạy trên quan hệ. Trong từng trường hợp chúng ta cũng đưa ra câu lệnh SQL cụ thể

q_1: tìm ngân sách của một dự án, biết trước số mã của nó

```
SELECT BUDGET
FROM J
WHERE JNO = Value
```

q_2: tim tên và ngân sách của tất cả dự án

```
SELECT JNAME, BUDGET
FROM J
```

q_3: tìm tên của dự án nằm tại thành phố đã cho

```
SELECT JNAME
FROM J
WHERE LOC = Value
```

q_4: tìm tổng ngân sách dự án của mỗi thành phố

```
SELECT SUM(BUDGET)
FROM J
```

WHERE LOC = Value

Theo bốn ứng dụng này, các giá trị sử dụng tính chất có thể được xác định. Để tiện lợi chúng tôi đặt A_1 = JNO, A_2 = JNAME, A_3 = BUDGET, và A_4 = LOC. Giá trị sử dụng được xác định dưới dạng ma trận (hình 5.15) trong đó đầu vào (i,j) ký hiệu cho use(q_i, A_j).

(hình 5.15)

Các giá trị sử dụng tính chất nói chúng không hình thành nên cơ sở của việc phân chia tính chất và phân mảnh. Điều này là do những giá trị này không biểu thị ảnh hưởng của tần số ứng dụng. Phép đo tần số có thể bao gồm trong việc xác định phép đo sự tương đồng của tính chất af f(A_i, A_j), mf các phép đo gắn chặt giữa hai tính chất của quan hệ theo cách chúng được truy cập bởi ứng dụng.

Cách đo sự tương đồng của tính chất giữa 2 tính chất A_i và A_j của quan hệ R(A_1, A_2, …, A_n) với tập các ứng dụng Q = (q_1, q_2, …, q_q) được xác định như sau:

(công thức)

trong đó $ref_i(q_k)$ là số lần truy cập của tính chất (A_i, A_j) đối với từng sự thực thi của ứng dụng q_k trong trạm S_i và $acc_i(q_k)$ là phép đo tần số truy cập ứng dụng đã được định nghĩa trước đó và được sửa đổi để bao gồm tần số tại các trạm khác.

Kết quả của phép tính này là một ma trận n x n, mỗi thành phần đều là một trong các phép đo được tính ở trên. Chúng tôi gọi ma trận này là ma trận tính chất tương đồng (AA).

Ví dụ 5.16

Chúng ta hãy tiếp tục với trường hợp chúng ta đã kiểm tra trong ví dụ 5.15. Để đơn giản hơn, chúng ta hãy giả sử rằng $ref_i(q_k)$ =1 đối với mọi q_k và S_i. Nếu tần số ứng dụng là

(công thức)

vậy thì phép đo tính tương đồng giữa các tính chất A_1 và A_3 có thể được đo như sau:

(công thức)

vì chỉ duy nhất một ứng dụng truy cập vào cả hai tính chất là q_1. Ma trận tính chất tương đồng được biểu diễn trong hình 5.16. Lưu ý là do tính toàn vẹn các giá trị chéo cũng được tính thậm chí mặc dù chúng là vô nghĩa.

Ma trận tính chất tương đồng sẽ được sử dụng trong toàn bộ phần còn lại của chương để hướng dẫn việc phân mảnh. Quá trình bao gồm việc phân nhóm với nhau các tính chất với tính tương đồng cao và sau đó chia mối quan hệ theo sự tương đồng đó.

Thuật toán nhóm. Nhiệm vụ cơ bản trong thiết kế thuật toán phân mảnh dọc là tìm một số phương pháp nhóm các tính chất của quan hệ dựa trên các giá trị tính chất tương đồng trong AA. Theo [Hofler và Severance, 1975] và [Navathe et al., 1984] người ta cho rằng thuật toán liên kết năng lượng (BEA) [McCormick et al, 1972] nên được áp dụng cho mục đích này. Xét các lý do sau [Hofler và Severance, 1975]:

1. Nó được thiết kế một cách cụ thể để xác định các nhóm những mục tương đồng thay vì sắp xết tuyến tính các mục (nghĩa là nó nhóm các tính chất có giá trị tương đồng lớn lại với nhau, và những tính chất có giá trị tương đồng nhỏ lại với nhau).
2. Việc nhòm cuối cùng là ngẫu nhiên so với thứ tự các mục được trình bày trong thuật toán.
3. Thời gian tính của thuật toán là hợp lý [$O(n^2)$, trong đó n là số các tính chất].
4. Mối quan hệ giao nhau thứ cấp giữa các tính chất được nhóm là xác định.

Thuật toán liên kết năng lượng lấy đầu vào là ma trận tính chất tương đồng, hoán vị dòng và cột của nó, và tạo ra ma chận tính tương đồng được nhóm (CA). Việc hoán vị được thực hiện theo cách tối đa hoá các phép đo tương đồng toàn cụ (AM):

(công thức)

trong đó

(công thức)

Tập các điều kiện xử lý các vấn đề trong đó một tính chất được đặt trong CA ở bên trái của tính chất phía trái ngoài cùng hoặc ở bên phải của tính chất phải ngoài cùng trong việc hoán vị cột, và trên của hàng trên cùng, dưới của hàng dưới cùng trong phép hoán vị dòng. Trong những trường hợp này, chúng tôi lấy 0 là giá trị aff giữa tính chát được xét để thay thế và trái hoặc phải (trên hoặc dưới) tính chất bên canh của nó, mà không tồn tại trong CA.

Hàm tối đa hoá chỉ xét những tính chất kề gần nhất, do vậy dẫn đến việc nhóm các giá trị lớn và giá trị nhỏ. Tương tự, ma trận tính chất tương đồng (AA) là đối xứng, làm giảm hàm đối tượng của công thức phía trên thành

(công thức)

Chi tiết về thuật toán liên kết năng lượng được trình bày trong thuật toán 5.5. Việc tạo ra ma trận nhóm tính tương đồng (CA) được thực hiện trong 3 bước:

1. *Khởi đầu.* Đặt và cố định một trong các cột của AA bất kỳ vào CA. Trong thuật toán cột 1 là cột được chọn
2. *Lặp.* Chọn từng cột trong số n-i cột còn lại (trong đó i là số cột đã được đặt vào CA) và cố gắng đặt chúng trong i-1 vị trí còn lại trong ma trận CA. Chọn vị trí sao cho sự ảnh hưởng là lớn nhất đối với phép đo tương đồng toàn cục được trình bày ở trên. Tiếp tục bước này cho đến khi không còn cột nào để đặt.
3. *Sắp xếp hàng.* Một khí thứ tự cột đã được xác định, vị trí của hàng cũng nên được thay đổi để vị trí quan hệ của chúng phù hợp với vị trí quan hệ của cột.

Thuật toán 5.3 BEA

input: AA: attribute affinity matrix

output: CA: clustered affinity matrix

begin

{initialize: remember that AA is an n x n matrix}

CA(•, 1) – AA(•, 1)

CA(•, 2) – AA(•,2)

index – 3

while index ≤ n do (choose the “best” location for attribute AA_{index})

begin

for i from 1 to index – 1 by 1 do

calculate cont(A_{i-1},A_{index},A_i)

end for

calculate cont($A_{index-1}$,A_{index},$A_{index+1}$) (boundary condition)

loc – placement given by maximum cont value

for j from index to loc by –1 do

CA(•, j) – CA(•, j-1)

end for

CA(•, loc) – AA(•, index)

index – index +1

end while

order the rows according to the relative ordering of columns

end. (BEA)

Để bước thứ hai của thuật toán thực hiện được, chúng ta cần định nghĩa ảnh hưởng của một tính chất tới phép đo tương đồng có nghĩa là gì. ảnh hưởng này có thể thu được như sau. Ta đã biết rằng phép đo tương đồng toàn cục AM được xác định là:

(công thức)

mà có thể được viết lại như sau:

(công thức)

Chúng ta hãy xác định mối liên hệ giữa hai tính chất A_x và A_y như sau:

(công thức)

Vậy thì AM có thể được viết lại:

(công thức)

Giờ hãy xét n tính chất sau:

(công thức)

Phép đo tương đồng toàn cục đối với những tính chất này được biểu diễn:

(công thức)

Giờ xét việc đặt một tính chất mới A_k vào giữa tính chất A_i và A_j trong ma trận nhóm tương đồng. Phép đo tương đồng toàn cục mới có thể được biểu diễn dưới dạng:

(công thức)

Do đó mạng ảnh hưởng vào phép đo tương đồng toàn cục của việc đặt tính chất A_k giữa A_i và A_j là:

(công thức)

Ví dụ 5.17

Xét ma trận AA đã cho trong hình 5.16 và nghiên cứu ảnh hưởng của việc di chuyển tính chất A_4 giữa tính chất A_1 và A_2, trình bày bởi công thức:

(công thức)

Tính từng số hạng một, ta có

(công thức)

Do đó

(công thức)

Chú ý là phepé tính mối liên hệ giữa hai tính chất yêu cầu rất nhiều các thành phần tương ứng của hai cột biểu diễn những tính chất này và lấy tổng hàng.

Thuật toán và biện luận của chúng tôi cho tới giờ đều tập trung vào cột của ma trận tính chất tương đồng. Chúng ta có thể đưa ra các lý luận tương tự và thiết kế lại thuật toán để chạy trên hàng. Vì ma trận AA là đối xứng, cả hai phương pháp này sẽ tạo ra kết quả giống nhau.

Một điểm khác về thuật toán 5.3 là để cải thiện tính hiệu quả, cột thứ hai cũng được cố định và đặt kế bên cột đầu tiên trong bước lặp. Điều này có thể chập nhận được vì, theo như thuật toán, A_2 có thể được đặt vào hoặc là bên trái hoặc là bên phải của A_1. Mối liên hệ giữa hai điều này là độc lập với vị trí của chúng.

Cuối cùng chúng tối trình bày vấn đề trong việc tính cont tại điểm cuối. Nếu một tính chất A_i được xét để đặt vào bên trái của tính chất bên trái ngoài cùng, một trong những biểu thức quan hệ được tính là giữa một thành phần bên trái không tồn tại và A_k [nghĩa là bond(A_0, A_k)]. Do đó chúng ta cần đề cập tới điều kiện ràng buộc của định nghĩa phép đo tương đồng toàn cục AM, trong đó CA(0, k) = 0. Điều khác nữa là nếu A_j là tính chất ngoài cùng bên phải mà đã được đặt vào trong ma trận CA và chúng ta đang kiểm tra sự ảnh hưởng của việc đặt tính chất A_k vào bên phải của A_j. Trong trường hợp này bond(k, k÷1) cần được tính. Tuy nhiên vì chưa có tính chất nào được đặt vào cột k÷1 của CA, phép đo tương đồng không được xác định. Do vậy theo điều kiện của điểm cuối, giá trị bond này cũng bằng 0.

Ví dụ 5.18

Xét việc nhóm các tính chất quan hệ J và sử dụng ma trận tính chất tương đồng AA của hình 5.16.

Theo như bước khởi đầu, chúng ta sao chép cột 1 và 2 của ma trận AA vào ma trận CA (hình 5.17a) và bắt đầu với cột 3 (nghĩa là tính chất A_3). Có 3 vị trí thích hợp để đặt cột 3., bên trái của cột 1, dẫn đến thứ tự là (3-1-2), giữa cột 1 và 2, (1-3-2) và bên phải của cột 2, (1-2-3). Lưu ý là để tính sự ảnh hưởng của thứ tự cuối cùng chúng ta phải tính cont(A_2, A_3, A_4) hơn là cont(A_1, A_2, A_3). Hơn nữa trong phần này A_4 đề cập tới vị trí chỉ mục thứ tư trong ma trận CA, là rỗng (hình 5.17c), chứ không đề cập đến cột tính chất A_4 của ma trận. Tính ảnh hưởng tới phép đo tương đồng toàn cục của từng phương pháp.

Thứ tự (0-3-1):

(công thức)

Chúng ta biết

(công thức)

Do đó

(công thức)

Thứ tự (1-3-2):

(công thức)

Do đó

(công thức)

Thứ tự (2-3-4):

(công thức)

Do đó

(công thức)

Vì ảnh hưởng của thứ tự (1-3-2) là lớn nhất, chúng ta chọn vị trí A_3 ở bên phải của A_1 (hình 5.17b). Tính tương tự đối với tính chất A_4 chỉ ra rằng nó cần phải đặt ở bên phải của A_2 (hình 5.17c)

Cuối cùng, các hàng đã được tổ chức theo một thứ tự giống với các cột và được tạo ra như trong hình 5.17d.

Trong hình 5.17d chúng ta nhận ra việc tao ra hai nhóm: một là trong góc trái trên và bao gồm các giá trị tương đồng nhỏ hơn và một nằm ở góc phải dưới và bao gồm các giá trị tương đồng lớn hơn. Việc phân nhóm này biểu thị các tính chất của quan hệ J được chia ra như thế nào. Tuy nhiên nói chúng sự phân biệt giữa cách chia này là không rõ ràng. Khi ma trận CA lớn, thường có nhiều hơn hai nhóm được hình thành và có nhiều sự phân chia thích hợp. Do đó cần một phương pháp để tiếp cận vấn đề này một cách có hệ thống hơn.

Thuật toán phân chia. Mucj didchs của việc chia này là để tìm ra các tập tính chất mà chỉ truy cập, hoặc đối với hầu hết các phần, bởi các tập ứng dụng ban đầu. Chẳng hạn nếu có thể xác định hai tính chất A_1 và A_2 mà chỉ được truy cập bởi ứng dụng q_1, và tính chất A_3 và A_4 chỉ được truy cập bởi hai ứng dụng q_2 và q_3, như vậy làm cho việc xác định mảnh trở nên rất rõ ràng. Nhiệm vụ là phải tìm một thuật toán xác định các nhóm này.

Xét ma trận tính chất nhóm của hình 5.18. Nếu một điểm dọc theo đường chéo là cố định, hai tập tính chất được xác định. Một tập $\{A_1, A_2, \ldots, A_i\{$ ở góc trái trên và tập thứ hai $\{A_{i+1}, \ldots, A_n\}$ ở bên phải và ở đáy của điểm này. Chúng tôi gọi tập đầu tiên là tập đỉnh và tập thứ hai là tập đáy và ký hiệu các tập tính chất lần lượt là TA và BA.

Giờ chúng ta quay sang tập ứng dụng $Q = \{q_1, q_2, \ldots, q_q\}$ và định nghĩa tập ứng dụng mà chỉ truy cập vào TA, hoặc chỉ truy vập vào BA hoặc cả hai. Những tập này được xác định như sau:

(công thức)

Biểu thức đầu tiên định nghĩa tập các tính chất được truy cập bởi ứng dụng q_1: TQ và BQ là các tập ứng dụng chỉ truy cập vào TA hoặc BA, và OQ là tập ứng dụng truy cập vào cả hai.

Có một vấn đề ở đây. Nếu có n tính chất của một quan hệ, có n-1 vị trí có thể mà việc chia các điểm có thể được đặt dọc theo đường chéo của ma trận nhóm tính chất đối với quan hệ đó. Vị trí tốt nhất cho sự phân chia là vị trí tạo ra tập TQ và BQ mà tổng truy cập tới chỉ duy nhất một mảnh đạt tối đa hoá trong khi tổng truy cập tới cả hai mảnh đạt tối thiểu. Do đó chúng ta định nghĩa biểu thức giá trị sau:

(công thức)

Mỗi biểu thức ở trên đếm tổng số lần truy cập tới tính chất của ứng dụng trong lớp tương ứng của nó. Dựa trên những số đo này, vấn đề tối ưu hoá được xác định bằng cách tìm điểm x (1<=x<=n) như công thức:

(công thức)

được tăng cực đại [Navathe et al., 1984]. Tính năng quan trọng của biểu thức này là ở chỗ nó định nghĩa hai mảnh giá CTQ và CBQ gần bằng nhau. Việc này cho phép làm cân bằng quá trình nạp khi các mảnh được phân phối tới các trạm khác nhau. Rõ ràng là thuật toán phân chia có mức độ phức tạp tuyến tính về số các tính chất của quan hệ là O(n).

Có hai sự phức tạp cần được chỉ ra. Đầu tiên là về việc phân chia. Độc giả cân phải lưu ý rằng quá trình phân chia tập các tính chất hai-chiều. Đối với tập tính chất lớn hơn, nó giông như là phân chia m-chiều.

Việc thiết kết một cách phân chia n-chiêu là có thể nhưng rất tốn công về mặt tính toán. Dọc theo đường chéo của ma trận CA, ta cần thử 1, 2, …, n-1 điểm chia, và đối với từng điểm chia này, cần phải kiểm tra vị trí nào là cực đại với x. Do đó độ phức tạp của một thuật toán như vậy là $O(2^n)$. Tất nhiên việc xác định x phải được thay đổi đối với những trường hợp có nhiều điểm chia. Giải pháp thích hợp là ứng dụng thuật toán chia nhị phân đối với từng mảnh thu được trong quá trình lặp trước đó. Ta sẽ tính TQ, Bq và OQ cũng như là số đo truy cập có liên quan với từng phân mảnh, và chia chúng nhỏ hơn.

Sự phức tạp thứ hai liên quan tới địa điểm của khối tính chất hình thành nên một phân mảnh. Biện luận của chúng tối cho tới lúc này giả sử rằng điểm chia là duy nhất và đơn lẻ và chia ma trận CA thành một phần trái trên và phần thứ hai là các tính chất còn lại. Tuy nhiên việc phân chia cũng có thể được hình thành ở giữa ma trận. Trong trường hợp này chúng ta cần thay đổi thuật toán một chút. Cột ngoài cùng bên trái của ma trận CA được chuyển thành cột ngoài cùng bên phải và hàng trên cùng được chuyển thành hàng dưới cùng. Quá trình chuyển đổi tuân theo việc kiểm tra n-1 vị trí đường chéo để tìm ra x cực đại. Ý tưởng nằm sau việc chuyển đổi là di chuyển khối tính chất mà sẽ hình thành nên một nhóm tới gõ trái trên cùng của ma trận, nơi nó có thể được xác định một cách dễ dàng. Với việc thêm vào phép chuyển đổi, độ phức tạp của thuật toán phân chia tăng lên $O(n^2)$.

Giả sử rằng một thủ tục chuyển đổi, gọi là SHIFT, đã được thực hiện, thuật toán phân chia đã cho trong thuật toán 5.4. Đầu vào của PARTITION là ma trận nhóm tính chất CA và quan hệ R được phân mảnh. Đầu ra là tập các mảnh $F_R = \{R_1, R_2\}$, trong đó $R_i \subseteq \{A_1, A_2, \ldots A_n\}$ và $R_1 \cap R_2 =$ các tính chất khoá của quan hệ R. Lưu ý là để chia n-chiều, việc này cần phải được viện dẫn lặp đi lặp lại hoặc được thực thi như một thủ tục đệ quy.

Thuật toán 5.4 PARTITION

input: CA: clustered affinity matrix; R: relation

output: F: set of fragments

```
begin
    {determine the z value for the first column}
    {the subscripts in the cost equations indicate the split point}
    calculate CTQ_{n-1}
    calculate CBQ_{n-1}
    calculate COQ_{n-1}
    best – CTQ_{n-1} * CBQ_{n-1} – (COQ_{n-1})^2
    do          {determine the best partitioning}
        begin
            for i from n-1 to 1 by –1 do
            begin
                calculate CTQ_i
                calculate CBQ_i
```

calculate COQ_i

$z - CTQ*CBQ_i-(COQ_i)^2$

if z >best then

begin

best –z

record the shift position

end if

end for

call SHIFT(CA)

end begin

until no more SHIFT is possible

reconstruct the matrix according to the shift position

R_1 - $\Pi_{TA}(R) \cup K$ {K is the set of primary key attributes of R}

R_2 - $\Pi_{BA}(R) \cup K$

$F - \{R_1, R_2\}$

end. {PARTITION}

Ví dụ 5.19

Khi thuật toán PARTITION được áp dụng cho ma trận CA thu với quan hệ J (ví dụ 5.18), kết quả là sự xác định của các mảnh $F_j = \{J_1, J_2\}$, trong đó $J_1 = \{A_1, A_3\}$ và $J_2 = \{A_1, A_2, A_4\}$. Do đó

(công thức)

Chú ý trong bài tập này chúng tôi thực hiện việc phân mảnh dựa trên toàn bộ tập tính chất chứ không chỉ ở các tính chất không khoá. Lý do là để đơn giản hoá ví dụ. Do vậy, chúng tôi bao gồm cả JNO, khoá của J trong J_2 cũng như là J_1.

Kiểm tra tính đúng đắn. Chúng tôi theo các biện luận tương tự như trong phân chia ngang để chứng minh rằng thuật toán PARTITION tạo ra phân mảnh dọc đúng.

Tính toàn vẹn. Tính toàn vẹn được đảm bảo bởi thuật toán PARTITION vì từng tính chất của quan hệ toàn cục được gán cho một trong các mảnh. Chừng nào tập các tính chất A trên quan hệ R được xác định bao gồm:

$A = TA \cup TB$

tính đúng đắn của phân mảnh dọc được đảm bảo.

Cấu trúc lại. Chúng tôi đã nhắc đến việc cấu trúc lại của quan hệ toàn cục ban đầu là có thể thực hiện bừng phép toán liên kết. Do đó, với quan hệ R theo phân mảnh dọc $F_r = \{R_1, R_2, \ldots, R_r\}$ và tính chất khoá K

(công thức)

Như vậy, chừng nào mỗi R_i là toàn vẹn, phép toán liên kết sẽ cấu trúc lại một cách chính xác R. Một điểm quan trọng khác nữa là mỗi R_i sẽ bao gồm tính chất khoá của R hoặc nó sẽ bao gồm tuple ID được hệ thống gán cho (TID).

Không liên quan. Như chúng tôi đã trình bày ở trên, tính không liên quan của các mảnh không quan trọng lắm trong phân mảnh dọc như ở phân mảnh ngang. Ở đây có hai trường hợp:

1. TID được sử dụng, trong trường hợp các mảnh không liên quan vì TID được lặp lại trong từng mảnh là do hệ thống gán vào và các thực thể quản lý, hoàn toàn vô hình với người dùng.
2. Các tính chất khoá được lặp lại trong từng phân mảnh, trong trường hợp không thể chỉ ra rằng chúng là không liên quan trong sự chặt chẽ của số hạng. Tuy nhiên cần phải nhận ra rằng việc lặp lại các khoá chính được biết và quản lý bởi hệ thống và không có sự liên hệ tương tự như nhân bản tuple trong mảnh được phân ngang. Nói cách khác, chừng nào các mảnh là không liên quan, ngoại trừ các tính chất khoá, chúng ta có thể thoả mãn và coi chúng là không liên quan.

5.3.3. Phân mảnh hỗn hợp

Trong hầu hết các trường hợp phân mảnh dọc hoặc ngang đơn giản của một mô hình csdl sẽ không đủ để thoả mãn yêu cầu của ứng dụng người dùng. Trong trường hợp này một phân mảnh dọc có thể được theo sau bởi một phân mảnh ngang, hoặc ngược lại, tạo ra một phân mảnh cấu trúc cây (hình 5.19). Vì hai phương pháp phânmảnh được ứng dụng lần lượt, phương pháp này được gọi là phân mảnh hỗn hợp. Nó cũng được gọi là phân mảnh lồng hoặc pha trộn.

Một ví dụ cho sự cần thiết của phân mảnh hỗn hợp là quan hệ J, quan hệ mà chúng ta đã thực hiện rất nhiều. Trong ví dụ 5.11 chúng ta phân chia nó thành sáu mảnh ngang dựa trên hai ứng dụng. Trong ví dụ 5.19 chúng ta chia quan hệ đó thành hai theo chiều dọc. Như vậy cái chúng ta có là một tập mảnh ngang, mỗi mang được chia nhỏ hơn thành hai mảnh dọc.

Số mức độ lồng có thể lớn hơn, nhưng có hạn. Trong trường hợp phân mảnh ngang, người ta phải dừng lại khi từng mảnh bao gồm chỉ duy nhất một tuple. Tuy nhiên giới hạn này có tính chất lý thuyết vì mức độ lồng trong hầu hết các ứng dụng thực tế không vượt quá 2. Đây là do các quan hệ toàn cục được bình thường hoá đã có mức độ nhỏ và người ta không thể thực hiện quá nhiều phân mảnh dọc trước khi chi phí liên kết trở nên quá cao.

Chúng tôi sẽ không thảo luận chi tiết các nguyên tắc đúng đắn và điều kiện đối với phân mảnh hỗn hợp vì chúng hiển nhiên tuân theo các quy tắc và điều kiện của phân mảnh ngang và dọc. Chẳng hạn để cấu trúc lại quan hệ toàn cục ban đầu trong trường hợp phân mảnh hỗn hợp, người ta bắt đầu ở các lá của cây phân chia và di chuyển lên bằng cách thực hiện việc liên kết và hợp nhất (hình 5.20). Việc phân mảnh là toàn vẹn nếu các mảnh lá và trung gian toàn vẹn. Tương tự, tính không liên quan được bảo đảm nếu mảnh lá và trung gian là không liên quan.

5.4. PHÂN PHỐI

Việc phân phối tài nguyên trên các nốt của mạng máy tính là một vấn đề đã được nghiên cứu rộng rãi. Tuy nhiên đa số các nghiên cứu này không chỉ ra vấn đề của thiết kế csdl phân tán, mà chỉ có vấn đề đặt các file riêng lẻ trên một mạng máy tính. Chúng tôi sẽ kiểm tra sự khác nhau giữa hai điều này. Đầu tiên chúng ta cần xác định vấn đề phân phối chính xác hơn.

5.4.1. Vấn đề phân phối

Giả sử có một tập mảnh $F = \{F_1, F_2, \ldots, F_n\}$ và một mạng bao gồm các trạm $S = \{S_1, S_2, \ldots, S_m\}$ trên đó có một tập ứng dụng $Q = \{q_1, q_2, \ldots, q_q\}$ đang hoạt động. Vấn đề phân phối liên quan tới việc tìm sự phân phối "tối ưu" của F tới S.

Một trong những vấn đề quan trọng cần được thảo luận là việc định nghĩa tính tối ưu. Tính tối ưu có thể được định nghĩa theo 2 phép đo [Dowdy và Foster, 1982]:

1. *Chi phí nhỏ nhất.* Hàm chi phí bao gồm chi phí lưu trữ từng F_i ở một trạm S_j, chi phí truy vấn F_i tại trạm S_j, chi phí cập nhật F_i ở tất cả các trạm lưu trữ nó, và chi phí truyền dữ liệu. Như vậy, vấn đề phân phối cố gắng tìm một phương án phân phối mà đạt được hàm tổng chi phí nhỏ nhất.
2. *Thực thi.* Phương pháp phân phối được thiết kế để duy trì khả năng thực thi. Hai cách phổ biến nhất là giảm thời gian phản hồi và tăng dữ liệu đầu vào ở từng trạm.

Đa số các mô hình đã được đưa ra cho đến thời điểm này coi điều này là đặc trưng của tính tối ưu. Tuy nhiên, nếu thực sự suy xét vấn đề theo chiều sau, dường như phép đo "tính tối ưu" nên bao gồm cả vấn đề thực thi và các nhân tố chi phí. Nói cách khác, ta nên tìm một phương pháp phân phối trả lời các truy vấn của người dùng trong thời gian tối thiểu trong khi vẫn giữ được chi phí xử lý ở mức nhỏ nhất. Tương tự đối với việc tăng tối đa dữ liệu đầu vào. Có người sẽ hỏi tại sao các mô hình như vậy lại chưa được phát triển. Câu trả lời rất đơn giản: tính phức tạp.

Xét một công thức đơn giản của vấn đề. Cho F và S như định nghĩa ở trên. Do vấn đề về thời gian, chúng ta chỉ xét một mảnh đơn F_k. Chúng tôi đưa ra một số giả thuyết và định nghĩa cho phép mô tả vấn đề phân phối.

1. Giả sử q có thể thay đổi để có thể xác định truy vấn cập nhật và truy vấn chỉ nhận, và xác định như sau đôi svới một mảnh đơn F_k:

   (công thức)

   trong đó t_i là lưu thông chỉ đọc được tạo ra tại trạm S_i đối với F_k, và

   (công thức)

   trong đó u_i là lưu thông cập nhật được tạo ra tại trạm S_i đối với F_k.
2. Giả sử chi phí truyền thông giữa hai cặp bất kỳ của trạm S_i và S_j là cố định đối với đơn vị truyền thông. Hơn nửa, giả sử có sự khác biệt giữa cập nhật và nhận để định nghĩa những điều sau:

   (công thức)

   trong khi cij là chi phí truyền thông của đơn vị đối với các yêu cầu nhận giữa trạm Si và Sj, và c'ij là chi phí truyền thông đơn vị của truy vấn cập nhật giữa Si và Sj.
3. Xét chi phí lưu trữ mảnh ở trạm là d_i. Do đó chúng ta có thể xác định $D = \{d_1, d_2, \ldots, d_m\}$ đối với chi phí lưu trữ mảnh F_k tại tất cả các trạm.
4. Giả sử không có ràng buộc về dung lượng đối với các trạm và các liên kết truyền thông.

Vậy thì vấn đề phân phối có thể xác định như một vấn đề giảm thiểu chi phí trong đó chúng ta cố gắng tìm một tập $I \subseteq S$ chỉ rõ vị trí các bản sao của mảnh được lưu. Dưới đây, x_j là ký hiệu của biến ra quyết định đối với vị trí như là

(công thức)

Việc xác định chính xác như sau:

(công thức)

tuỳ vào

$x_j = 0$ hay 1

Vế thứ hai của công thức tính tổng chi phí lưu trữ tất cả các bản sao của mảnh. Mặt khác vế thứ nhất phản ứng với chi phí truyền thông tin cập nhật tới tất cả các trạm lưu bản sao của mảnh, và với chi phí thực hiện các truy vấn chỉ nhận ở trạm, tạo ra chi phí truyền dữ liệu nhỏ nhất.

Đây là một công thức rất đơn giản không thích hợp với thiết kế csdl phân tán. Nhưng thậm chí như vậy đi nữa, vẫn còn một vấn đề nữa. Công thức này, theo [Casey, 1972], đã được chứng minh cho NP-toàn vẹn [Eswaran, 1974]. Những công thức khác nhau của vấn đề đã được chứng minh chỉ trong 1 năm (chẳng hạn, [Sacca và Wiederhold, 1985] và [Lam và Yu, 1980]. Tất nhiên sự liên hệ với những vấn đề lớn (nghĩa là số mảnh và trạm lớn) là thu được các giải pháp tối ưu là không khả thi về mặt tính toán. Những nghiên cứu đáng kể đã được thực hiện nhằm tìm cách cung cấp các giải phải khả quan.

Có một số lý do tại sao các công thức đơn giản như chúng ta đã thảo luận không phù hợp với phân tích thiết kế hướng đối tượng. Đây lày những vấn đễ đã có trong tất cả các mô hình phân phối file trước đây trong mạng máy tính.

1. Người ta không thể coi manh như là các file riêng biệt được phân phối cô lập tại một thời điểm. Vị trí của một mảnh thường có ảnh hưởng đến các quyết định vị trí của các mảnh khác khi các mảnh này được truy cập cùng nhau vì chi phí truy cập tới các mảnh đã có có thể thay đổi (chẳng hạn do liên kết phân tán). Do đó, mối quan hệ giữa các phải phải được tính đến.
2. Việc truy cập dữ liệu bởi các ứng dụng được mô tả rất đơn giản. Yêu cầu người dùng được đưa tới một trạm và tất cả dữ liệu trả lời nó được truyền tới trạm đó. Trong hệ thống csdl phân tán, truy cập tới dữ liệu phức tạp hơn các mô hình "truy cập file từ xa" đơn giản này. Do đó, mối quan hệ giữa sự phân phối và xử lý truy vấn cần phải được mô tả một cách chính xác.
3. Những mô hình này không xét đến chi phí của liên kết bắt buộc, dù vậy xác định hai mảnh tham gia vào cùng một ràng buộc liên kết tại hai trạm khác nhau có thể rất tốn chi phí.
4. Tương tự, chi phí của cơ cấu điều khiển dòng quan hệ cần phải được xem xét [Rothnie và Goodman, 1977].

Nói tóm lại, chúng ta cần phải nhớ là mỗi quan hệ chéo giữa các vấn đề của csdl phân tán được mô tả trong hình 1.8. Vì việc phân phối là rất tập trung, mối quan hệ của nó với thuật toán được thực hiện trong các lĩnh vực vấn đề khác cần phải được biểu diễn trong mô hình phân phối. Tuy nhiên, đây chính là điều làm nó trở nên rất khó giải quyết các mô hình này. Để tách các vấn đề cũ của phân phối file khỏi phân phối mảnh trong thiết kế csdl phân tán, chúng ta coi phân phối file là *vấn đề phân phối file* (FAP) và phân phối mảnh là *vấn đề phân phối csdl* (DAP).

Không có mô hình chung để có thể được xem như đưa vào một tập mảnh và tạo ra một chủ đề phân phối gần tối ưu đối với các kiểu ràng buộc được trình bày ở đây.

Các mô hình được phát triển tới ngày nay đưa ra một số giả định đơn giản và có thể ap dụng trong một số công thực cụ thể. Như vậy, thay vì trình bày một hoặc nhiều những thuật toán phân phối này, chúng tôi trình bày một mô hình quan hệ chung và sau đó thảo luận một số phương pháp khả thi có thể ứng dụng để giải quyết nó.

5.4.2. Yêu cầu thông tin

Ở bước phân phối chúng ta cần dữ liệu số lượng về csdl, ứng dụng chạy trên nó, mạng truyền thông, khả năng xử lý, và giới hạn lưu trữ của từng trạm trên mạng. Chúng tôi sẽ thảo luận chi tiết từng điều một.

Thông tin csdl. Để thực hiện phân chia ngang, chúng tôi định nghĩa sự lựa chọn các minterm. Giờ chúng tôi mở rộng định nghĩa đó tới các mảnh, và đinh nghĩa sự lựa chọn các mảnh F_j tương ứng với truy vấn q_i. Đây là một số tuple F_j cần được truy cập để xử lý q_i. Giá trị này được ký hiệu là $sel_i(F_j)$.

Một thông tin khác cần cho mảnh csdl là kích cỡ của nó. Kích cỡ của một mảnh F_j được xác định là

(công thức)

trong đó length(F_j) là độ dài (theo byte) của tuple của mảnh F_j.

Thông tin ứng dụng. Đa số các thông tin liên quan đến ứng dụng đã được biên dịch trong khi phân mảnh, nhưng cần một số thông tin nữa trong mô hình phân phối. Hai số đo quan trọng là số lần truy cập để đọc mà một truy vấn q_i thực hiện với mảnh F_j trong quá trình thực thi của nó (ký hiệu RR_{ij}) và số lần tương ứng đối với truy cập cập nhật (UR_{ij}). Chẳng hạn có thể đếm số khối bị truy cập được yêu cầu bởi một truy vấn.

Chúng tôi cũng cần định nghĩa hai ma trận UM và RM, với các phần tử u_{ij} và r_{ij}, lần lượt được xác định như sau:

(công thức)

Vectơ O các giá trị o(i) cũng được định nghĩa, trong đó o(i) biểu thị trạm ban đầu của truy vấn q_i. Cuối cùng, định nghĩa ràng buộc thời gian phản hồi, thời gian phản hồi tối đa cho phép của từng ứng dụng cần phải được xác định cụ thể.

Thông tin trạm. Đối với từng trạm máy tính, chúng ta cần biết về dung lượng lưu trữ và khả năng xử lý của nó. Hiển nhiên những giá trị này có thể được tính bằng các hàm tinh vi hoặc chỉ bằng sự ước lượng đơn giản. Chi phí đơn vị của lưu trữ dữ liệu ở trạm S_k được ký hiêu là USC_k. Cũng cần phải xác định một số đo chi phí LPC_k là chi phí xử lý một đơn vị công việc tại trạm S_k. Đơn vị công việc thường được xác định với số đo RR và UR.

Thông tin mạng. Trong mô hình của chúng tôi, chúng tôi giả sử có sự tồn tại của một mạng đơn giản có chi phí truyền thông được xác định với một khung dữ liệu. Do đó g_{ij} ký hiệu cho chi phí truyền thông đối với từng khung giữa các trạm S_i và S_j. Để cho phép tính toán số thông điệp, chúng tôi sử dụng fsize ký hiệu cho kích cỡ (bằng byte) của một khung. Không có thắc mắc về các mô hình mạng phức tạp về dung lượng kênh, khoảng cách giữa các trạm, giao thức đầu gói tin, v.v.. . Tuy nhiên những điều này vượt quá phạm vi của chương này.

5.4.3. Mô hình phân phối

Chúng tôi thảo luận một mô hình phân phối giảm tối thiểu tổng chi phí xử lý và lưu trữ trong khi cố gắng đáp ứng điều kiện thời gian phản hồi nhất định. Mô hình chúng tôi sử dụng có dạng:

min(Total Cost)

tuỳ vào

ràng buộc thời gian phản hồi

ràng buộc về lưu trữ

ràng buộc trong xử lý

Trong phần còn lại của phần này chúng tôi mở rộng các thành phần của mô hình này dựa trên yêu cầu thông tin trong phần 5.4.2. Biến số quyết định là x_{ij} được xác định như sau:

(công thức)

Tổng chi phí. Hàm tổng chi phí có hai thành phần: xử lý truy vấn và lưu trữ. Do đó nó có thể biểu thị dưới dạng:

(công thức)

trong đó QPC_i là chi phí xử lý truy vấn của ứng dụng q_i, và STC_{jk} là chi phí lưu trữ mảnh F_j ở trạm S_k.

Xét chi phí lưu trữ trước. Nó được cho một cách đơn giản là:

(công thức)

và hai phép tính tổng tìm tổng chi phí lưu trữ tại tất cả các trạm đối với tất cả các mảnh.

Chi phí xử lý truy vấn khó xác định hơn. Đa số các mô hình của vấn đề phân phối file (FAP) tách nó làm hai phần: chi phí xử lý chỉ nhận và chi phí xử lý cập nhật. Chúng tôi chọn một cách tiếp cận khác trong mô hình của chúng tôi về vấn đề phân phối csdl (DAP) xác định nó bao gồm chi phí xử lý (PC) và chi phí truyền tải (TC). Do vậy chi phí xử lý truy vần (QPC) đối với ứng dụng q_i là:

(công thức)

Theo hướng dẫn được trình bày ở phần 5.4.1, bộ phận xử lý, PC, bao gồm 3 nhân tố, chi phí truy cập (AC), chi phí liên kết (IE) và chi phí điều khiển dòng (CC):

(công thức)

Biểu diễn chi tiết của từng nhân tố chi phí này phụ thuộc vào các thuật toán được sử dụng để hoàn thành tác vụ. Tuy nhiên, chúng tôi trình bày AC chi tiết hơn:

(công thức)

Hai số hạng đầu tiên trong công thức trên tính số truy cập của truy vấn người dùng q_i tới mảnh F_j. Lưu ý là $(UR_{ij} \div RR_{ij})$ tính tổng số lần truy nhập nhận và cập nhật. Giả sử chi phí cục bộ của việc xử lý chúng được xác định. Phép tính tổng tính tổng số truy cập cho tất cả các mảnh được nhắc đến.

Phép nhân LPC_k tính chi phí của truy nhập này tại trạm S_k. Chúng tôi lại sử dụng x_{jk} để chọn chỉ các giá trị chi phí đối với các trạm lưu trữ mảnh.

Có một vấn đề rất quan trọng cần phải lưu ý ở đây. Hàm chi phí truy nhập biểu thị rằng quá trình xử lý một truy vấn có liên quan đến việc chia nó thành một tập các truy vấn con, mỗi truy vấn này làm việc trên một mảnh được lưu trên trạm đó, tiếp đó truyền kết quả trở về trạm có truy vấn ban đầu. Như đã thảo luận ở trên, đây là một cách nhìn đơn giản, không xét đến tính phức tạp của xử lý csdl. Chẳng hạn, hàm chi phí không tinh đến chi phí thực hiện liên kết (nếu cần), mà có thể được thực hiện theo một số cách, được nghiên cứu ở chương 9. Trong một mô hình thực tế hơn mô hình chung chung mà chúng tôi đang xét, vấn đề này không nên bỏ qua.

Nhân tố chi phí liên kết có thể được trình bày gần giống như thành phần xử lý, ngoại trừ chi phí xử lý đơn vị cục bộ có thể thay đổi để phản ánh chi phí thực của liên kết. Vì việc kiểm tra liên kết và các phương thức điều khiển dòng được thảo luận ở phần sau của cuốn sách, chúng tôi không cần phải nghiên cứu những thành phần chi phí này sâu hơn ở đây. Độc giả nên quay lại phần này sau khi đọc chương 6 và chương 11 để hiểu về các hàm chi phí thực sự có thể phân chia.

Hàm chi phí truyền tải có thể được biểu diễn bằng công thức dọc theo đường của hàm chi phí truy nhập. Tuy nhiên, phần đâuc ủa truyền tải dữ liệu cho cập nhật và cho các yêu cầu nhận dữ liệu là rất khác nhau. Trong truy vấn cập nhật cần phải thông báo tất cả các trạm có các bản sao, trong khi trong truy vấn nhận dữ liệu, truy cập tới duy nhất một bản sao là đủ. Thêm nữa, ở cuối của yêu cầu

cập nhật, không có việc truyền dữ liệu trở về cho trạm ban đầu ngoài một thông báo xác nhận, trong đó các truy vấn chỉ nhận có thể dẫn tới việc truyền tải dữ liệu đáng kể.

Thành phần cập nhật của hàm truyền tải là

(công thức)

Số hạng đầu tiền là để gửi thông báo cập nhật từ trạm ban đầu o(i) của q_i tới tất cả các mảnh bản sao cần được cập nhật. Số hạng thứ hai là dành cho việc xác nhận.

Chi phí nhận dữ liệu có thể biểu diễn:

(công thức)

Số hạng đầu tiên là TCR biểu diễn chi phí truyền yêu cầu nhận tới những trạm có bản sao các mảnh cần được truy cập. Số hạng thứ hai là cho việc truyền kết quả từ những trạm này tới trạm ban đầu. Biểu thức nêu lên rằng trong số tất cả các trạm với các bản sao của cùng một mảnh chỉ duy nhất một trạm tạo ra tổng chi phí truyền dữ liệu nhỏ nhất được chọn để thực hiện công việc.

Hàm chi phí truyền dữ liệu cho truy vấn q_i có thể được biểu diễn như sau
(công thức)

Công thức trên trình bày đầy đủ hàm tổng chi phí.

Ràng buộc. Các hàm ràng buộc có thể được trình bày chi tiết như vậy. Tuy nhiên thay vì mô tả những hàm này, chúng tôi chỉ đơn giản trình bày chúng trông như thế nào. Ràng buộc về thời gian phản hồi có thể được trình bày:

(công thức)

Để hoàn hảo hơn, số đo chi phí trong hàm đối tượng sẽ được trình bày theo dạng thời gian, khi đưa ra một cách cụ thể về rạng buộc thời gian thực thi.

Ràng buộc về lưu trữ là:

(công thức)

trong khi đó ràng buộc về xử lý là:

(công thức)

Điều này hoàn thành sự phát triển về mô hình phân phối của chúng tôi. Thậm chí mặc dù chúng tối không hoàn toàn phát triển nó, sự chính xác trong một số công thức biểu diễn về việc lập công thức như thế nào đối với một vấn đề như vậy. Thêm vào khía cạnh này, chúng tôi đã đưa ra các vấn đề quan trọng cần chỉ ra trong mô hình phân phối.

5.4.4. Các phương pháp giải quyết

Trong phần trên chúng tôi phát triển một mô hình phân phối chung mà khá phức tạp so với mô hình FAP được trình bày trong phần 5.4.1. Vì mô hình FAP là NP hoàn toàn, ta sẽ mong có một giải pháp cho việc lập công thức vấn đề phân phối csdl (DAP) cúng là NP hoàn toàn. Thậm chí mặc dù chúng tối không chứng minh ước đoán này, nó là có thật. Do đó ta phải tìm các phương pháp giải quyết tạo ra các giải pháp khá tối ưu. Bài kiểm tra "tính chất" trong trường hợp này hiển nhiên là kết quả gần của thuật toán sát với phân phối tối ưu như thế nào.

Một số phương pháp khác đã được áp dụng cho giải pháp mô hình FAP và DAP. Rõ ràng là có một sự tương quan giữa FAP và vấn đề phân phối đã được nghiên cứu trong nghiên cứu các tiến trình. Thực tế, sự cân bằng của FAP đơn giản và vấn đề phân phối sản phẩm nhà kho đã được trình bày [Ramamoorthy và Wah, 1983]. Do đó các phương pháp được phát triển bởi các nhà nghiên cứu tiến

trình nói chung đã được kế thừa để giải quyết vấn đề FAP và DAP. Các ví dụ là giải pháp vấn đề ba lô [Ceri et al., 1982b], …, và thuật toán lưu thông trên mạng [Chang và Liu, 1982].

Có các nỗ lực khác để giảm tính phức tạp của vấn đề. Một phương pháp đã giả sử rằng tất cả các phân chia được chọn đã được quyết định cùng nhau với các chi phí tương ứng của chúng theo xử lý truy vấn. Vậy thì vấn đề được mô tả để lựa chọn sự phân chia tối ưu và vị trí đối với từng quan hệ [Ceri et al., 1983]. Một phương pháp đơn giản khác thường được áp dụng là bỏ qua bản sao và tìm một giải pháp không sao chép tối ưu. Việc sao chép được xử lý ở bước thứ hai bằng cách áp dụng thuật toán tham lam mà bắt đầu bằng giải pháp không sao chép như một giải pháp khả thi ban đầu, và cố gắng cải thiện nó [Ceri et al., 1983] và [Ceri và Pernici, 1985]. Đối với những phương pháp này, có không dủ dữ liệu để xác định kết quả gần với sự tối ưu như thế nào.

5.5. KẾT LUẬN

Trong phần này, chúng tôi đã trình bày các kỹ thuật có thể được sử dụng cho thiết kế csdl phân tán với tầm quan trọng đặc biệt về vấn đề phân mảnh và phân phối. Có một số các nghiên cứu được đưa vào trong thiết kế csdl phân tán. Chẳng hạn Chang đã tự phát triển một lý thuyết phân mảnh [Chang và Cheng, 1980], và phân phối [Chang và Liu, 1982]. Tuy nhiên do tính chín muồi trong sự phát triển của nó, cúng tối chọn phát triển chương này theo hướng được phát triển bởi Ceri, Pelagatti, Navathe, và Wiederhold. Tài liệu tham khảo của chúng tôi với tài liệu của các tác giả này phản ánh điều này rất rõ.

Có một phần thân tài liệu đáng kể trong vấn đề phân phối, tập trung chủ yếu vào vấn đề phân phối file đơn giản hơn. Chúng tôi vẫn không có các mô hình chung hiệu quả có thể xét tất cả các khía cạnh của việc phân tán dữ liệu. Mô hình được trình bày trong phần 5.4 chỉ làm nổi bật các loại vấn đề cần được xem xét. Trong nội dung bài này, xem xét các phương pháp khác nhau để giải quyết vấn để phân phối phân tán có thể là rất hữu ích. Người ta có thể phát triển một tập các quy tắc giả pháp kèm theo các công thức toán học và giảm không gian giải pháp, do đó làm cho việc giải quyết vấn đề khả thi hơn.

Một hướng nghiên cứu rất thú vị trong thiết kế csdl mà không được xem xét đầy đủ là việc sử dụng kỹ thuật mô phỏng. Có một số nghiên cứu theo hướng này phân tích các ảnh hưởng trên việc thực thi csdl của sự thừa file [Muro et al., 1983] và [Muro et al., 1985] và về quyết định phân phối [Yoshida et al., 1985]. Tuy nhiên, vẫn chưa có nghiên cứu nào sử dụng mô phỏng như là một công cụ thiết kế chứ không phải một công cụ phân tích.