CHƯƠNG 3 **THIẾT KẾ CƠ SỞ DỮ LIỆU PHÂN TÁN**

MỤC TIÊU

Chương này đề cập đến hai vấn đề chính sau:

1. Một số tiêu chuẩn thiết kế về cách thức phân tán dữ liệu một cách hợp lý.
2. Nền tảng toán học hỗ trợ cho nhà thiết kế xác định sự phân tán dữ liệu.

Chương này chia làm ba phần:

- Phần thứ nhất giới thiệu mô hình thiết kế cơ sở dữ liệu phân tán với hai tiếp cận từ trên xuống và từ dưới lên.
- Phần thứ hai trình bày sự thiết kế phân mảnh ngang, phân mảnh dọc và phân mảnh hỗn hợp.
- Phần thứ ba trình bày sự cấp phát các phân mảnh. Vấn đề này nhắm đến sự ánh xạ các phân mảnh đến các ảnh vật lý.

3.1. Mô hình thiết kế cơ sở dữ liệu phân tán

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

- Thiết kế lược đồ quan niệm.

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

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

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

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

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

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

- Site mà ứng dụng được đưa ra (còn được gọi là site gốc của ứng dụng)
- Tần số hoạt động của ứng dụng (nghĩa là số lượng yêu cầu hoạt động trong một đơn vị thời gian); trong trường hợp tổng quát các ứng dụng có thể được

đưa ra từ nhiều sites, chúng ta cần biết tần số hoạt động của mỗi ứng dụng tại mỗi site.

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

3.1.1. **Các mục tiêu của việc thiết kế phân tán dữ liệu**

a. Sự truy xuất cục bộ

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

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

b. Tính sẵn sàng và khả tín của các dữ liệu phân tán

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

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

c. Sự phân bố tải

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

d. Chi phí lưu trữ

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

3.1.2 **Các tiếp cận từ trên - xuống và từ dưới - lên để thiết kế sự phân tán dữ liệu**

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

Trong cách tiếp cận từ trên xuống, chúng ta bắt đầu 1. thiết kế lược đồ phổ quát; 2. thiết kế sự phân mảnh cơ sở dữ liệu và sau cùng 3. cấp phát các mảnh đến các sites, tạo các ảnh vật lý của chúng.

Cách tiếp cận này thích hợp nhất đối với các hệ thống được phát triển từ đầu và nó cho phép thiết kế một cách hợp lý. Trong chương này chúng ta không đề cập đến cách thiết kế lược đồ phổ quát và lược đồ vật lý vì nó không riêng biệt gì đối với cơ

sở dữ liệu phân tán mà tập trung vào sự thiết kế phân mảnh và cấp phát các phân mảnh.

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

- Chọn một mô hình cơ sở dữ liệu chung để mô tả lược đồ phổ quát của cơ sở dữ liệu.
- Chuyển dịch mỗi lược đồ cục bộ vào trong mô hình dữ liệu chung.
- Tổ hợp lại lược đồ cục bộ vào trong lược đồ phổ quát chung.

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

3.2. Thiết kế sự phân mảnh dữ liệu

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

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

Ví dụ 3.1:

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

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

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

3.2.1 Sự phân mảnh nguyên thủy

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

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

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

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

Thuộc tính = giá trị

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

$$y = V (p^*_i)$$

Với $p^*_i = p_i$ hoặc p^*_i= not p_i và y = true

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

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

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

DEPT = 1 AND JOB = ”P”

DEPT = 1 AND JOB ≠ “P”

DEPT ≠ 1 AND JOB = “P”

DEPT ≠ 1 AND JOB ≠ “P”

Tất cả các vị từ đơn giản trên là thích hợp, trong khi, ví dụ, SAL > 50 không là một vị từ thích hợp;

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

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

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

2. Tập các vị từ đơn giản P_r được gọi là *cực tiểu* nếu tất cả các vị từ của nó thích hợp (nghĩa là các phân mảnh tương ứng được tham khảo đến bởi ít nhất một ứng dụng)

Ví dụ 3:

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

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

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

P_3 = {DEPT = 1, JOB = "P", SAL >50 }là đầy đủ nhưng không cực tiểu vì SAL > 50 không thích hợp.

Sự phân mảnh có thể thực hiện như sau:

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

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

Ví dụ 4:

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

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

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

Xét một ví dụ tổng quát :

Ví dụ này dựa trên cơ sở dữ liệu ở chương 2 gồm có các quan hệ EMP, DEPT, SUPPLIER, SUPPLY. Giả sử cơ sở dữ liệu phân tán của công ty ở California có ba sites tại San Francisco (site 1), Fresno (site 2), và Los Angeles (site 3); Fresno nằm giữa San Francisco và Los Angeles. Có tất cả 30 phòng ban được nhóm lại như sau: 10 phòng ban đầu tiên ở gần San Francisco, các phòng ban từ 11 đến 20 ở gần Fresno và các phòng ban trên 20 thì ở gần Los Angeles. Tất cả các nhà cung cấp ở San Francisco hoặc ở Los Angeles. Ngoài ra công ty cũng được chia theo khái niệm miền: San Francisco ở miền Bắc, Los Angeles ở miền nam còn Fresno nằm giữa hai miền đó nên một số phòng ban nằm gần Fresno sẽ rơi vào miền bắc hoặc miền nam.

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

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

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

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

Chúng ta đưa các vị từ sau:

p_1 : CITY = "SF"
P_2: CITY = "LA"

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

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

- Các vị từ thích hợp mô tả cho phân mảnh này không thể được suy ra bằng cách phân tích mã lệnh của ứng dụng.
- Quan hệ mật thiết giữa các vị từ giảm đi số lượng phân mảnh. Trong trường hợp này chúng ta nên xem xét những vị từ tương ứng với các vị từ sơ cấp sau:

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

Nhưng chúng ta đã biết rằng:

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

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

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

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

DEPT(<u>DEPTNUM</u>, NAME, AREA, MGRNUM)

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

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

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

Chúng ta đưa ra các vị từ sau:

p_1: DEPTNUM <= 10
p_2: 10 < DEPTNUM <= 20
p_3: DEPTNUM > 20
p_4: AREA = “North”
p_5: AREA = “South”

Có một số quan hệ giữa các vị từ trên như AREA= “North” kéo theo DEPTNUM > 20 là sai; vì thế sự phân mảnh giảm còn 4 phân mảnh:

y_1: DEPTNUM <=10
y_2: (10 < DEPTNUM <= 20) AND (AREA = “North”)
y_3: (10 < DEPTNUM <= 20) AND (AREA = “South”)
y_4: DEPTNUM > 20

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

Hình 4.2 Sự phân mảnh của quan hệ DEPT

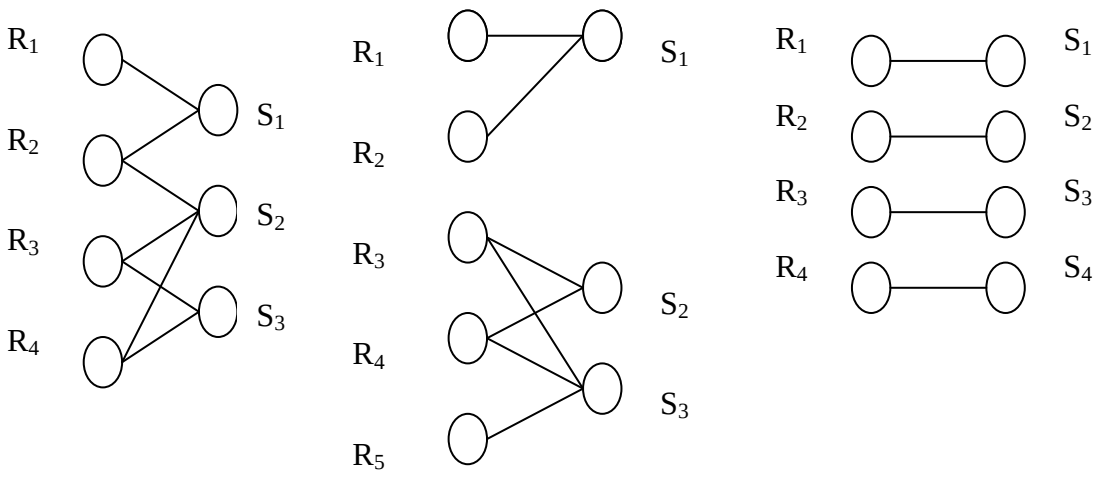
Một nhận xét cuối cùng là sự cấp phát phân mảnh cũng dễ dàng thấy qua sự phân mảnh này. Các phân mảnh tương ứng với vị từ y_1 và y_4 được lưu trữ tại site 1 và site 3; các phân mảnh ứng với các vị từ y_2 hoặc y_3 thể hiện các ứng dụng quản trị thì được phân bố tại site 1 hoặc 3 và các phân mảnh ứng về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

3.2.2 **Sự phân mảnh dẫn xuất ngang**

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

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

Kết nối phân tán được biểu diễn một cách hiệu quả bằng cách dùng đồ thị kết nối. Đồ thị kết nối G của kết nối phân tán R và S là một đồ thị (N,E) với các nút N thể hiện các phân mảnh của R và S và các cạnh vô hướng E biểu diễn các kết nối không rỗng giữa các phân mảnh. Để đơn giản, chúng ta không chứa trong các phân mảnh nào của R và S mà có kết nối rỗng. Đồ thị kết nối được minh họa ở hình I.10.



Hình I.10 Các đồ thị kết nối

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

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

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

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

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

$$R_i = R \ SJ_f \ S_t$$

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

Ví dụ tổng quát (tiếp theo)

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

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

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

Giả sữ quan hệ phổ quát DEPT được phân mảnh ngang theo DEPTNUM và quan hệ SUPPLIER phân mảnh ngang theo DEPTNUM. Có hai phân mảnh ngang dẫn xuất

cho quan hệ SUPPLY bằng phép nửa kết với các phân mảnh SUPPLIER và với các phân mảnh DEPT.

3.2.3 Sự phân mảnh dọc

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

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

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

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

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

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

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

Việc phân mảnh dọc đã nói lên sự nhân bản trong các phân mảnh. Sự nhân bản có ảnh hưởng khác nhau trong các ứng dụng cập nhật hay ứng dụng chỉ đọc. Sự nhân bản có lợi cho các ứng dụng chỉ đọc vì nó được tham khảo cục bộ. Nhưng đối với các ứng dụng cập nhật thì nó lại không phù hợp vì chúng ta phải cập nhật tất cả các bản sao để bảo đảm tính nhất quán.

Ví dụ tổng quát (tiếp theo)

Xét quan hệ phổ quát:

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

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

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

Các ứng dụng về công việc được quản lý tại các phòng ban yêu cầu thông tin về NAME, MGRNUM và DEPTNUM của các nhân viên; các ứng dụng này có thể được

gọi tại tất cả các sites và tham khảo các bộ nhân viên trong cùng một nhóm của các phòng ban với xác suất 80%.

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

EMP_1(EMPNUM, *NAME*, TAX, SAL)

EMP_2(EMPNUM, *NAME*, MGRNUM,DEPT)

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

3.2.4 **Sự phân mảnh hỗn hợp**

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

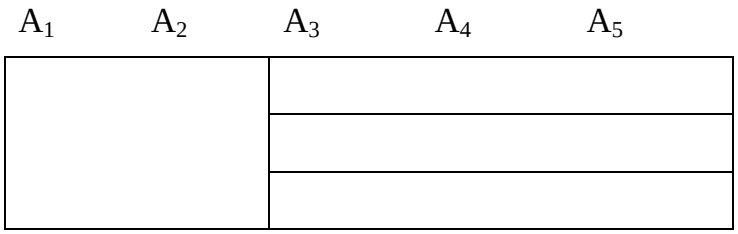
1. Áp dụng sự phân mảnh ngang đối với các phân mảnh dọc.
2. Áp dụng sự phân mảnh dọc đối với các phân mảnh ngang.

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

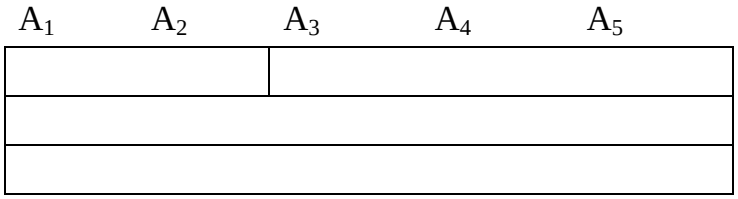
Hình 4.4 thể hiện thứ tự của sự phân mảnh như sau:

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

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



Sự phân mảnh dọc rồi sau đó phân mảnh ngang



Sự phân mảnh ngang sau đó phân mảnh dọc

Hình 4.4 Sự phân mảnh hỗn hợp của quan hệ R(A_1, A_2, A_3, A_4, A_5)

Ví dụ tổng quát

Xét lại quan hệ phổ quát EMP được phân mảnh dọc thành EMP_1 và EMP_2. Giả sử các ứng dụng về công việc được điều hành tại các phòng ban mà sử dụng phân mảnh EMP_2 tham khảo đến xác suất 80% các bộ của các phòng ban lân cận với site mà các

ứng dụng đó được gọi. Vì thế EMP2 có thể được phân mảnh ngang tiếp tục theo nhóm các phòng ban.

3.3. Sự cấp phát các phân mảnh

Bài toàn cấp phát các phân mảnh không giống như bài toán cấp phát hệ thống file vì :

- Các phân mảnh không được xem như là các files.
- Có nhiều phân mảnh hơn các quan hệ cục bộ.
- Mô hình hoá hoạt động ứng dụng của hệ thống file đơn giản hơn ứng dụng trong cơ sơ dữ liệu phân tán.

Tiêu chuẩn chung cho sự cấp phát phân mảnh: Trong việc xác định sự cấp phát các phân mảnh, điều quan trọng là phải xác định chúng ta đang thiết kế sự cấp phát không dư thừa hay sự cấp phát dư thừa (tức là có sự nhân bản dữ liệu không)

Sự nhân bản dữ liệu sinh ra nhiều phức tạp hơn trong thiết kế, vì:

- Mức độ nhân bản của mỗi phân mảnh là một biến của bài toán.
- Mô hình hoá các ứng dụng chỉ đọc sẽ phức tạp hơn bởi các ứng dụng phải chọn ra site nào để truy xuất dữ liệu.

Để xác định việc nhân bản dữ liệu, có hai phương pháp sau:

- Xác định một tập của tất cả các site mà lợi ích của việc cấp phát một bản sao nhiều hơn chi phí bỏ ra.
- Đầu tiên tiến hành không nhân bản dữ liệu sau đó bắt đầu nhân bản dữ liệu đến các site mà có lợi ích cao nhất.