Chương VI

CƠ SỞ DỮ LIỆU PHÂN TÁN

Chương này đề cập đến các khái niệm cơ bản về hệ cơ sở dữ liệu phân tán, hệ quản trị cơ sở dữ liệu phân tán và bài toán thiết kế cơ sở dữ liệu phân tán. Bài toán thiết kế cơ sở dữ liệu phân tán rất phức tạp và do trọng tâm của giáo trình này trình bày kiến thức về hệ cơ sở dữ liệu quan hệ nên các phương pháp thiết kế phân tán không được trình bày chi tiết cho đến các giải thuật mà chủ yếu là nêu vấn đề. Mặt khác đây là hệ cơ sở dữ liệu phân tán dựa trên mô hình quan hệ nên trước khi thiết kế phân tán các cơ sở dữ liệu phải được thiết kế theo chuẩn của hệ cơ sở dữ liệu quan hệ tập trung.

I. CÁC KHÁI NIỆM VỀ HỆ CƠ SỞ DỮ LIỆU PHÂN TÁN.

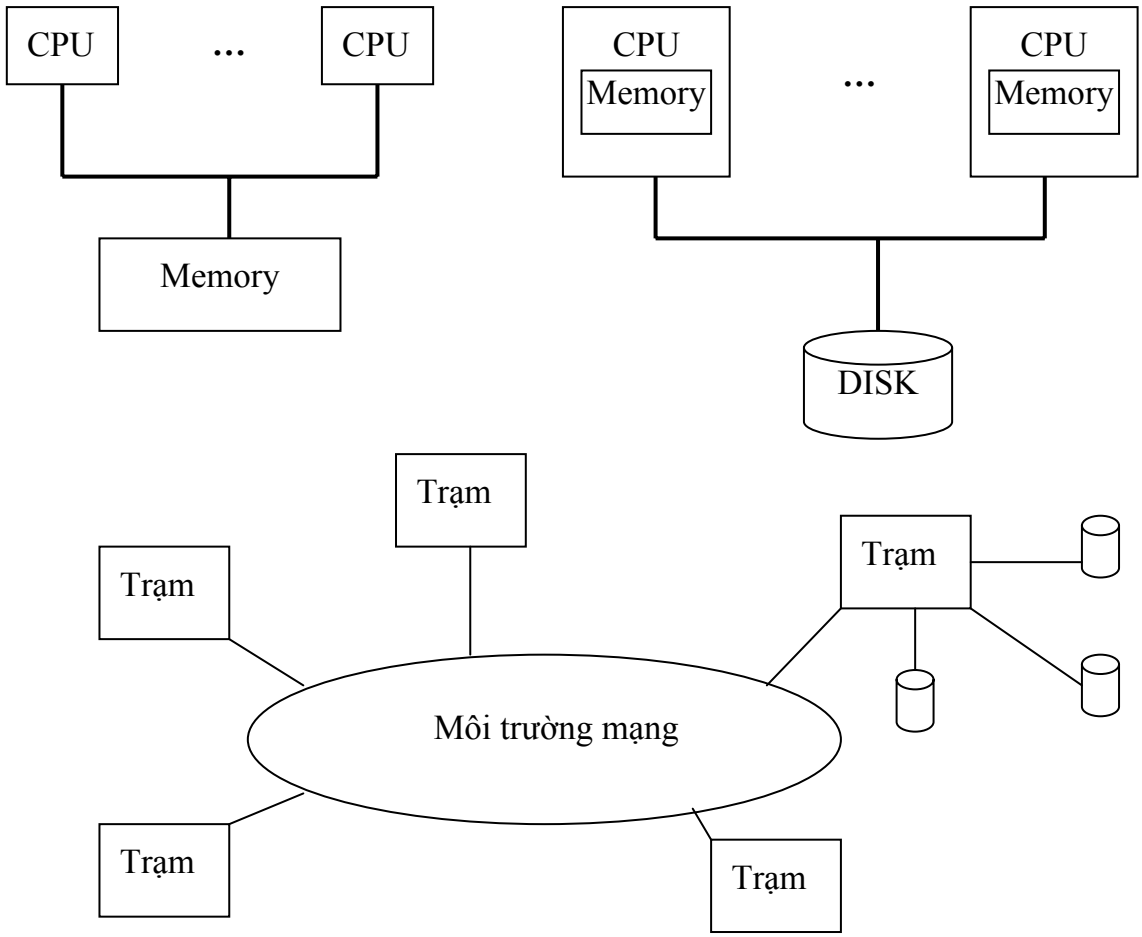
Hệ cơ sở dữ liệu phân tán (*Distributed DataBase System*) được xây dựng dựa trên hai công nghệ cơ bản là cơ sở dữ liệu và mạng máy tính. Hệ cơ sở dữ liệu phân tán được mô tả như là tập hợp nhiều cơ sở dữ liệu có liên quan logic đến nhau và được phân bố trên mạng máy tính.

Trong khái niệm được mô tả về cơ sở dữ liệu phân tán ở trên có hai đặc trưng cơ bản là "*liên quan logic*" và "*phân bố trên mạng*". Trong cơ sở dữ liệu phân tán các tập tin dữ liệu được lưu trữ độc lập trên các nút của mạng máy tính và phải có liên quan đến nhau về mặt logic và còn hơn thế nữa còn đòi hỏi chúng phải được truy xuất đến qua một giao diện chung, thống nhất.

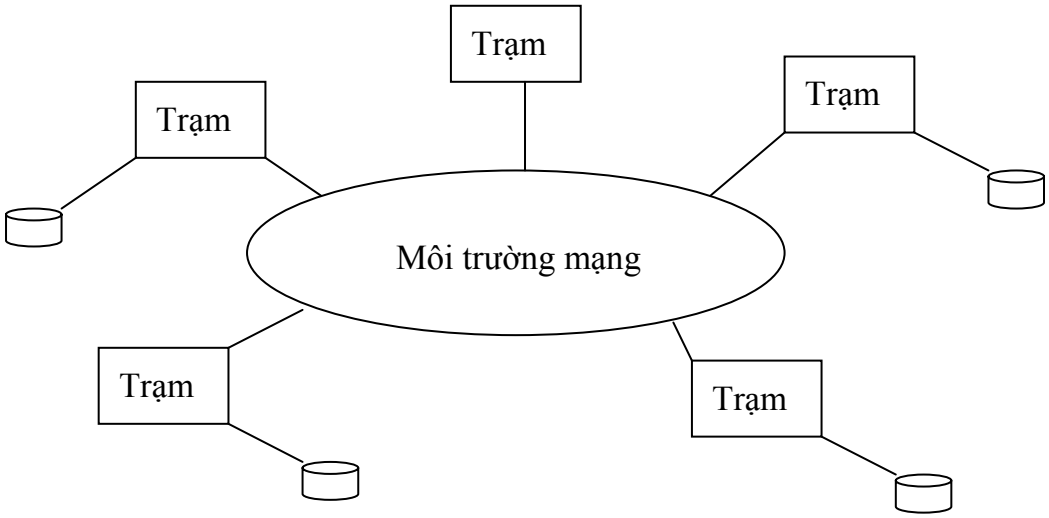
Hiện nay khái niệm xử lý phân tán (Distributed procesing), tính toán phân tán (Distributed computing) hoặc các thuật ngữ có từ "phân tán" hay được dùng để chỉ các hệ thống rải rác như các hệ thống máy tính có đa bộ xử lý (multiprocessor system) hay là các xử lý trên mạng máy tính. Cơ sở dữ liệu phân tán là một khái niệm không bao gồm các trường hợp xử lý dữ liệu trong các hệ thống sử dụng bộ nhớ chung, kể cả bộ nhớ trong hay bộ nhớ thứ cấp (đĩa từ), nhất thiết phải là một hệ có sử dụng giao tiếp mạng với các trạm làm việc độc lập.

Hệ Quản trị cơ sở dữ liệu phân tán (*Distributed DBMS*) là hệ thống phần mềm cho phép quản lý các hệ cơ sở dữ liệu phân tán và làm cho sự phân tán trở nên "*trong suốt*" đối với người sử dụng.

Khái niệm "trong suốt" – "transparent" để chỉ sự tách biệt ở cấp độ cao của hệ thống với các vấn đề cài đặt ở cấp độ thấp của hệ thống. Có các dạng "trong suốt" như sau:



Hình 6.1. Các mô hình không phải Hệ CSDL phân tán



Hình 6.2. Mô hình hệ CSDL phân tán

+ "Trong suốt" về phân tán. Do tính chất phân tán của hệ thống nên các dữ liệu được lưu trữ tại các nút có vị trí địa lý khác nhau, phần mềm sẽ đáp ứng các yêu cầu

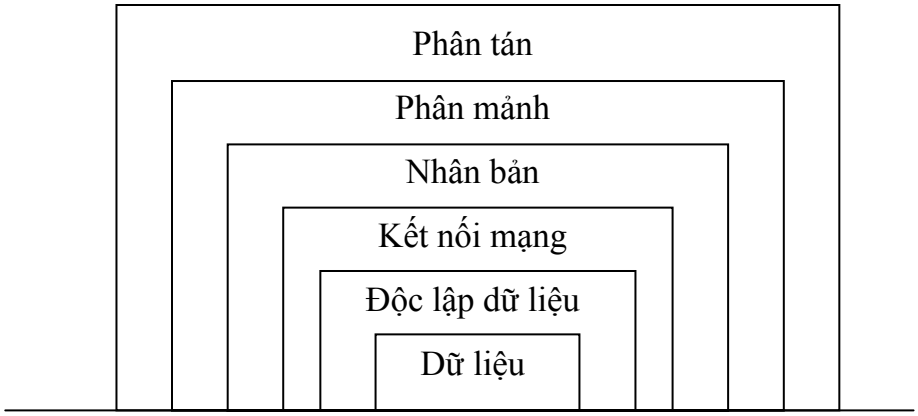
của người sử dụng sao cho người sử dụng không cần phải biết vị trí địa lý của dữ liệu.

+ "Trong suốt" về phân hoạch (Partition). Do dữ liệu phân tán và do nhu cầu của công việc dữ liệu cần được phân hoạch và mỗi phân hoạch được lưu trữ tại một nút khác nhau (đây gọi là quá trình phân mảnh – fragmentation). Quá trình phân mảnh hoàn toàn tự động bởi hệ thống và người sử dụng không cần phải can thiệp.

+ "Trong suốt" về nhân bản (Replication). Vì lí do "hiệu năng", "tin cậy" nên dữ liệu còn được sao chép một phần ở những vị trí khác nhau.

+ "Trong suốt" về độc lập dữ liệu.

+ "Trong suốt" về kết nối mạng. Người sử dụng không cần biết về sự có mặt của giao tiếp mạng.



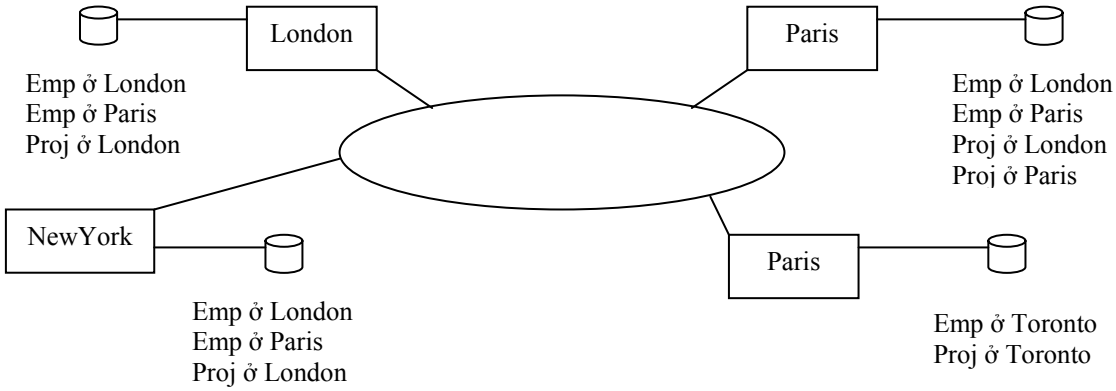
Hình 6.3. Các tầng trong suốt của hệ thống.

Ví dụ 6.1: Một công ty có các văn phòng ở Paris, London, NewYork, Toronto. Công ty này có các cơ sở dữ liệu sau đây:

Cơ sở dữ liệu về nhân viên:	EMP (ENo, EName, Title)
Cơ sở dữ liệu về các dự án:	PROJ (PNo, PName, Budget, Loc)
Cơ sở dữ liệu về lương:	PAY (Title, Sal)
Cơ sở dữ liệu về phân công:	ASG (ENo, PNo, Dur, Resp)

Giải thích về các thuộc tính: *Sal: Lương; Title: Chức vụ; Budget: Ngân sách của dự án; Loc: Địa điểm; Dur: Duration – Thời hạn; Resp: Responsibility – Trách nhiệm*

Do tính phân tán của các văn phòng nên tại mỗi văn phòng có lưu trữ dữ liệu tác nghiệp của chính các văn phòng đó, có thể là các nhân viên tại đó và các dự án mà văn phòng đó đang quản lý. Ta có sơ đồ lưu trữ đã phân tán và phân mảnh giả định như sau:



Hình 6.4. Sơ đồ lưu trữ phân tán

II. MÔ HÌNH KIẾN TRÚC CỦA HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU PHÂN TÁN.

Trong các mô hình kiến trúc của hệ QTCSDL phân tán được trình bày trong các tài liệu kinh điển ta xét 3 mô hình phổ biến.

1.1. Mô hình kiến trúc của hệ phân tán khách/đại lý – client/server.

Đặc trưng của hệ này là chức năng của hệ thống được chia làm hai lớp:

+ Chức năng đại lý – server function

+ Chức năng khách hàng – client function.

Trong hệ thống khách/đại lý các thao tác xử lý dữ liệu đáp ứng yêu cầu của khách hàng đều được thực hiện bởi các chức năng đại lý, chỉ có kết quả được gửi trả cho khách hàng. Ta có mô hình chức năng như sơ đồ ở hình 7.5.

Hệ khách có các tầng:

+ Giao diện tương tác với người sử dụng (User Interface), các chương trình ứng dụng (Application Program), ...

+ Hệ quản trị cơ sở dữ liệu khách hàng (Client DBMS).

+ Các phần mềm mạng có chức năng truyền tin (Communication Software).

Hệ đại lý có các tầng:

+ Các phần mềm mạng có chức năng truyền tin.

+ Tầng kiểm soát ngữ nghĩa của dữ liệu (Semantic Data Controler).

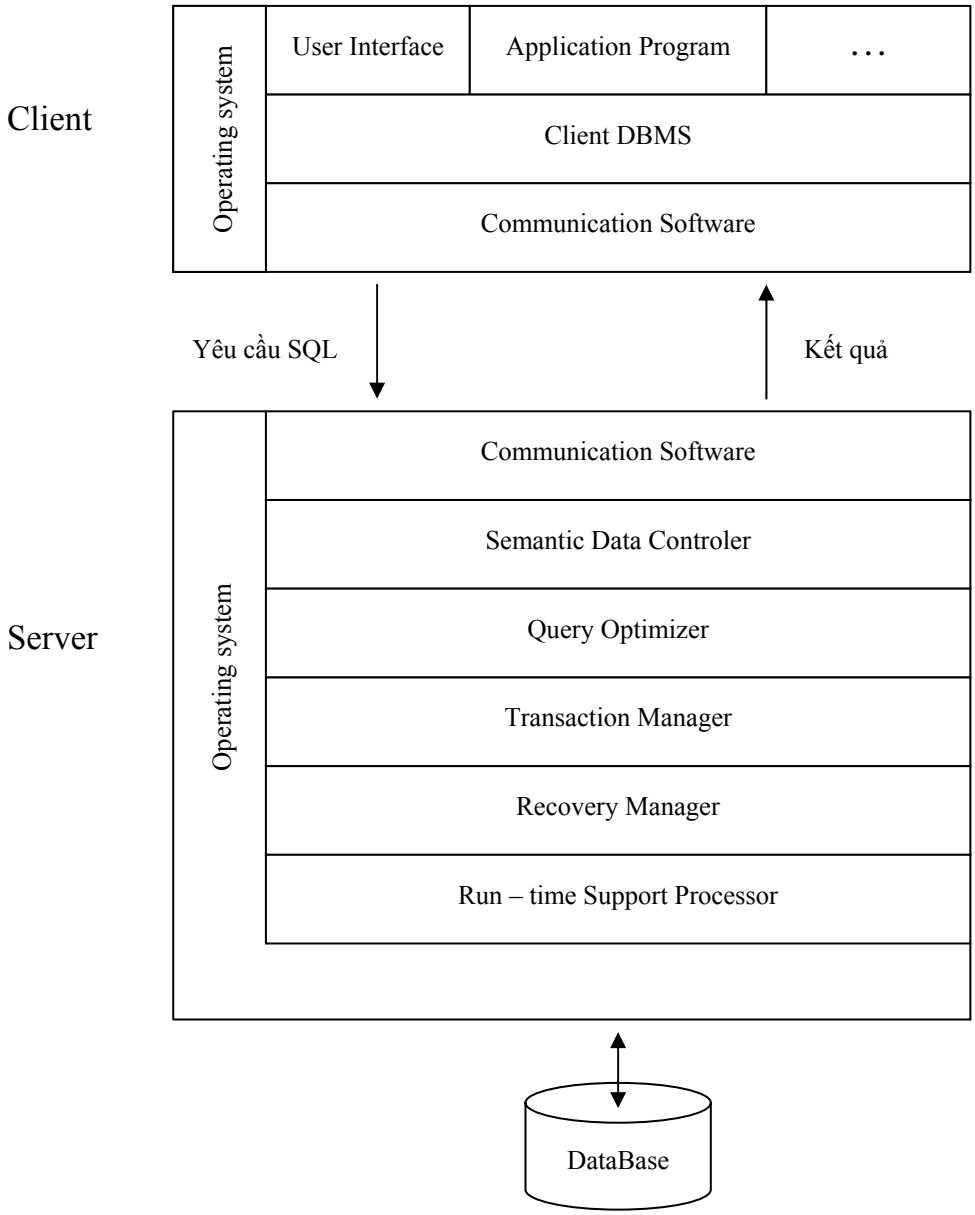
+ Tầng tối ưu hóa câu hỏi (Query Optimizer).

+ Tầng quản lý các giao tác (Transaction Manager).

+ Tầng quản lý khôi phục (Recovery Manager).

+ Tầng hỗ trợ thực thi (Run – time Support Processor) .

+ Hệ điều hành quản lý chung và giao tiếp với cơ sở dữ liệu vật lý.

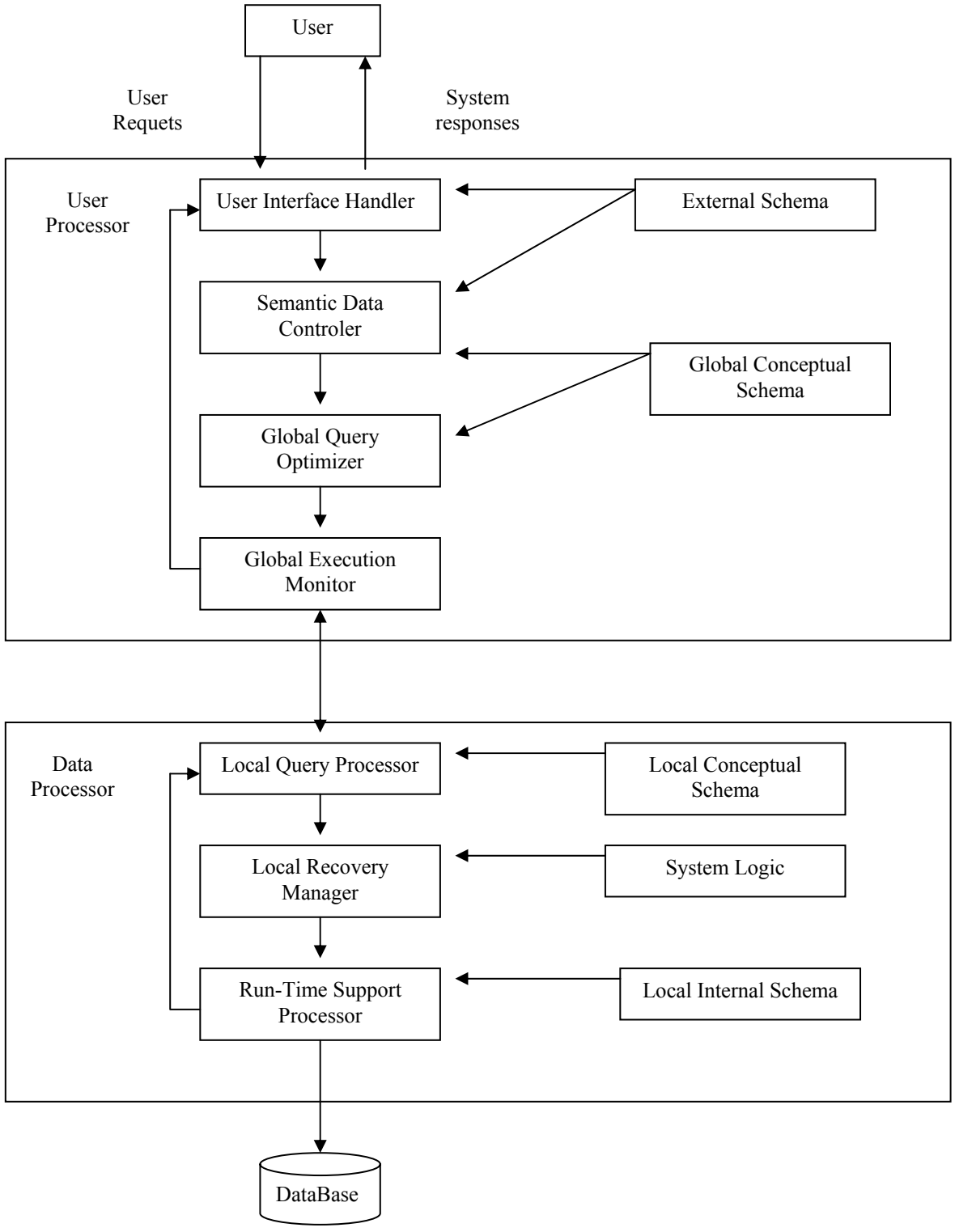


Hình 6.5. Sơ đồ hệ phân tán client/server

Hệ client/server có ưu điểm là xử lý dữ liệu tập trung, trên đường truyền chỉ có các gói tin yêu cầu (câu hỏi) và các kết quả đáp ứng câu hỏi, giảm tải được khối lượng

truyền tin trên mạng kết hợp với thiết bị tại đại lý rất mạnh sẽ tăng tốc độ xử lý dữ liệu của cả hệ thống.

1.2. Mô hình hệ phân tán ngang hàng.



Hình 6.6. Sơ đồ kiến trúc của hệ phân tán ngang hàng

Đặc điểm nổi bật của hệ thống này là dữ liệu được tổ chức ở các nút có chức năng như nhau, đồng thời sự tổ chức dữ liệu ở các nút này lại có thể rất khác nhau, từ đó cần phải có:

+ Định nghĩa dữ liệu tại mỗi vị trí: tại mỗi nút phải xây dựng lược đồ dữ liệu cục bộ LIS (*Local Internal Schema*)

+ Mô tả cấu trúc logic toàn cục: Lược đồ khái niệm toàn cục GCS (*Global Conceptual Schema*).

+ Mô tả cấu trúc logic tại mỗi vị trí, điều nảy xảy ra do nhân bản và phân mảnh, gọi là lược đồ khái niệm cục bộ LCS (*Local Conceptual Schema*).

+ Mô tả cấu trúc dữu liệu của các ứng dụng gọi là lược đồ ngoại giới ES (*External Schema*).

Cấu trúc của hệ thống bao gồm hai thành phần chính: Bộ phận tiếp nhận người dùng (*User Processor*) và bộ phận xử lý dữ liệu (*Data Processor*). Hai modun này được đặt chung trên mỗi máy chứ không tách biệt như hệ thống khách/đại lý.

Các chức năng cơ bản của từng modun như sau:

+ *User Interface Handler* - Giao tiếp người sử dụng: Diễn dịch yêu cầu, định dạng kết quả.

+ *Semantic Data Controler* - Kiểm soát dữ liệu ngữ nghĩa: Dựa vào lược đồ khái niệm toàn cục để kiểm tra câu vấn tin có thực hiện được hay không.

+ *Global Query Optimizer* - Tối ưu hóa câu hỏi toàn cục: Định ra chiến lược thực thi tốt nhất trên các nút.

+ *Global Execution Monitor* – Điều khiển thực thi câu vấn tin toàn cục.

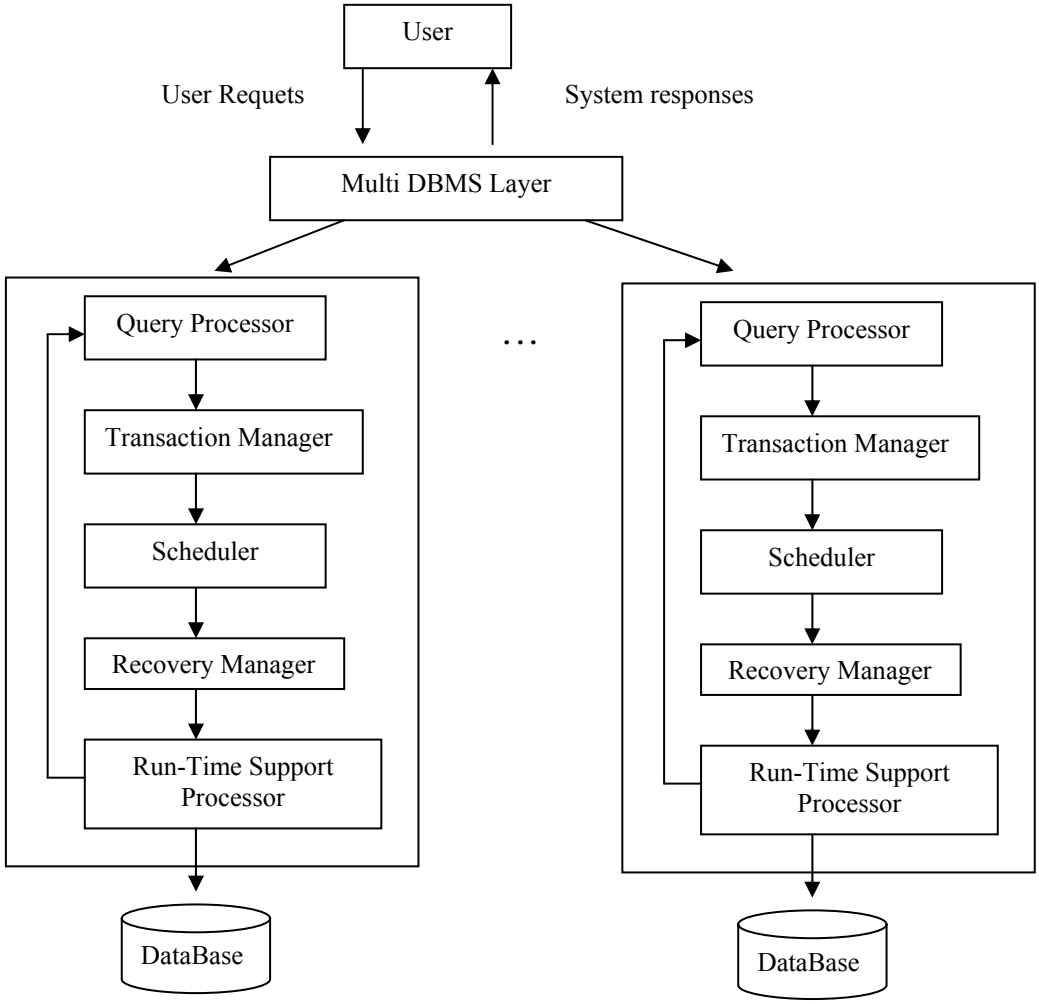
+ *Local Query Processor* – Xử lý câu hỏi cục bộ

+ *Local Recovery Manager* – Quản lý khôi phục cục bộ: Quản lý sự nhất quán khi có sự cố.

+ *Run-Time Support Processor* - Bộ phận hỗ trợ thực thi: Quản lý truy xuất cơ sở dữ liệu.

1.2. Mô hình hệ phân tán phức hợp.

Sự khác biệt cơ bản so với hệ phân tán ngang hàng là ở chỗ phức hệ không có (hoặc có không đầy đủ) một lược đồ khái niệm toàn cục.



Hình 6.7. Sơ đồ kiến trúc của hệ phân tán phức hợp

Trong 3 mô hình nêu ở trên thì mô hình khách/đại lý đang được phát triển và chứng tỏ các ưu điểm của nó về tính đơn giản và hữu hiệu trên mạng.

II. THIẾT KẾ CƠ SỞ DỮ LIỆU PHÂN TÁN.

Một cơ sở dữ liệu phân tán dựa trên mô hình quan hệ trước hết phải tuân thủ các quy tắc về chuẩn hóa cho cơ sở dữ liệu quan hệ. Để phân tán cơ sở dữ liệu có hai hoạt động chính đó là: *Phân mảnh các quan hệ* và *Phân tán các quan hệ* (cấp phát các mảnh).

2.1. Các kiểu phân mảnh.

Trong phần này ta đang xét hệ ccơ sở dữ liệu phân tán dựa trên các lược đồ quan hệ, tức là các bảng, như vậy sự phân mảnh chính là là hoạt động chia một bảng thành các bảng nhỏ hơn. Để phân tích sự phân mảnh ta lấy các quan hệ EMP, PROJ, PAY, ASG đã mô tả ở phần trên.

EMP

ENo	*EName*	*Title*
E1	John	Ks. Điện
E2	Mary	Ks. Hệ thống
E3	Bill	Ks. Cơ khí
E4	Bush	Ks. Lập trình
E5	Blair	Ks. Hệ thống
E6	Tom	Ks. Điện
E7	Algor	Ks. Cơ khí
E8	David	Ks. Điện

ASG

ENo	*PNo*	*Resp*	*Dur*
E1	P1	Quản lý	12 (tháng)
E2	P1	Phân tích HT	24
E2	P2	Phân tích HT	6
E3	P3	Tư vấn	10
E3	P4	Kỹ thuật	48
E4	P2	Lập trình	18
E5	P2	Quản lý	24
E6	P4	Quản lý	48
E7	P3	Kỹ thuật	36
E8	P3	Quản lý	40

PROJ

PNo	*PName*	*Budget*	*Loc*
P1	Thiết bị	150000	Toronto
P2	CSDL	125000	NewYork
P3	Games	75000	NewYork
P4	CAD	100000	Paris

PAY

Title	*Sal*
Ks. Điện	4000
Ks. Hệ thống	7000
Ks. Cơ khí	3500
Ks. Lập trình	2000

2.1.1. Phân mảnh ngang.

Giả sử ta có một yêu cầu phân mảnh quan hệ PROJ thành hai bảng PROJ1 và PROJ2 sao cho một bảng chứa các dự án có ngân sách lớn hơn 100000 và cái kia chứa các dự án có ngân sách nhỏ hơn 100000.

PROJ1

PNo	*PName*	*Budget*	*Loc*
P1	Thiết bị	150000	Toronto
P2	CSDL	125000	NewYork

PROJ2

PNo	*PName*	*Budget*	*Loc*
P3	Games	75000	NewYork
P4	CAD	100000	Paris

2.1.2. Phân mảnh dọc

Cũng quan hệ PROJ ta phân mảnh thành hai bảng PROJ3 và PROJ4, khóa của quan hệ PNo có mặt ở cả hai bảng con.

PROJ3

PNo	*PName*	*Loc*
P1	Thiết bị	Toronto
P2	CSDL	NewYork
P3	Games	NewYork
P4	CAD	Paris

PROJ4

PNo	*Budget*
P1	150000
P2	125000
P3	75000
P4	100000

Trong thực tế sự phân mảnh sẽ xảy ra việc kết hợp cả hai loại phân mảnh và ta gọi là sự phân mảnh hỗn hợp. Mức độ phân mảnh tùy theo yêu cầu của ứng dụng, phân mảnh quá lớn hay quá nhỏ đều gây ra các hiệu ứng phụ khó kiểm soát.

2.1.3. Các quy tắc phân mảnh.

Các quy tắc này nhằm đảm bảo tính nhất quán của cơ sở dữ liệu, đặc biệt về ngữ nghĩa của dữ liệu.

q1. Tính đầy đủ.

Nếu một quan hệ R được phân mảnh thành các mảnh con R_1, R_2, ..., R_n thì mỗi mục dữ liệu phải nằm trong một hoặc nhiều các mảnh con. Ở đây trong phân ngang thì mục dữ liệu được hiểu là các bộ còn trong phân mảnh dọc là các thuộc tính. Quy tắc này đảm bảo không bị mất dữ liệu khi phân mảnh.

q2. Tính tái thiết được.

Nếu một quan hệ R được phân mảnh thành các mảnh con R_1, R_2, ..., R_n thì phải định nghĩa được một toán tử quan hệ ∇ sao cho $R = \nabla_{i=1}^{n} R_i$

q3. Tính tách biệt.

Giả sử một quan hệ R được phân mảnh thành các mảnh con R_1, R_2, ..., R_n.

Đối với phân mảnh ngang mục d_i đã nằm trong mảnh R_j thì nó sẽ không nằm trong mảnh R_k với $k \neq j$.

Đối với phân mảnh dọc thì khóa chính phải được lặp lại trong các mảnh con, còn các thuộc tính khác phải tách rời.

2.2. Phương pháp phân mảnh ngang.

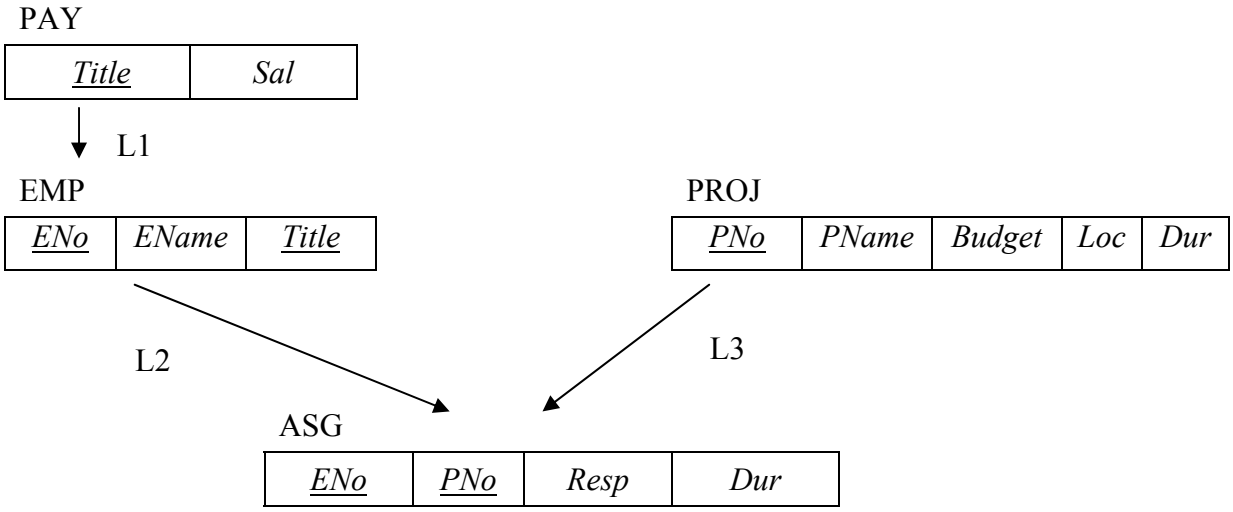
2.2.1. Các yêu cầu về thông tin.

Để phục vụ cho các hoạt động phân mảnh ta cần có các loại thông tin sau đây:

a1) *Thông tin về cơ sở dữ liệu.*

Đây là thông tin về lược đồ dữ liệu toàn cục, chỉ ra các mối liên kết giữa các quan hệ. Ta mô hình hóa sự liên kết này bằng một đồ thị có hướng, các cung chỉ một liên hệ kết nối bằng, mỗi nút là một lược đồ quan hệ. Quan hệ ở đầu đường nối gọi là quan hệ chủ nhân (Owner) còn quan hệ ở cuối đường nối gọi là quan hệ thành viên (Member). ta định nghĩa hai hàm *Owner* và *Member* từ tập các đường nối đến tập các quan hệ.

Ví dụ 6.2:



Ta có các hàm Owner và Member xác định như sau:

Owner (L1) = PAY, Member (L1) = EMP

Owner (L2) = EMP, Member (L2) = ASG

Owner (L3) = PROJ, Member (L3) = ASG

a2) *Thông tin về ứng dụng.*

Thông tin về ứng dụng có hai loại: *Thông tin định tính* dùng để phân mảnh và *thông tin định lượng* dùng để cấp phát.

Thông tin định tính về cơ bản là các vị từ dùng trong câu vấn tin, các vị từ này được xây dựng dựa trên sự phân tích các ứng dụng.

Định nghĩa *vị từ đơn giản*: Cho lược đồ $R = (A_1, A_2, ..., A_n)$ với thuộc tính A_i có miền xác định D_i ta có vị từ đơn giản

$$p: A_i \theta \text{ Value} \qquad \theta \in \{<, \leq, =, \geq, >, \neq\} \text{ và Value} \in D_i$$

Tập P_{Ri} chứa các vị từ đơn giản trên quan hệ R_i. Ví dụ với quan hệ PROJ ở trên ta có tập vị từ đơn giản sau: P_{PROJ} = { PName = 'Xây dựng', Budget ≤ 100000 }

Định nghĩa *vị từ hội sơ cấp*: Cho tập $P_{Ri} = \{p_{i1}, p_{i2}, ..., p_{im}\}$ chứa các vị từ đơn giản trên R_i. Ta định nghĩa tập các vị từ hội sơ cấp $M_i = \{m_{i1}, m_{i2}, ..., m_{it}\}$ như sau:

$$m_{ij} = \bigwedge_{p_{ik} \in P_{Ri}} p^*_{ik}$$

trong đó p^*_{ik} là p_{ik} hoặc $\neg p_{ik}$.

Ở đây vì lí do phức tạp nên ta chỉ xét đến các phủ định của vị từ đẳng thức đơn giản.

Ví dụ 6.3: Ta xét quan hệ PAY có các vị từ đơn giản.

p_1: Title = 'Ks. Điện'

p_2: Title = 'Ks. Hệ thống'

p_3: Title = 'Ks. Cơ khí'

p_4: Title = 'Ks. Lập trình'

p_5: Sal ≤ 3500

p_6: Sal > 3500

từ đó ta xây dựng được các vị từ hội sơ cấp như sau:

m_1: Title = 'Ks. Điện' ∧ Sal ≤ 3500

m_2: Title = 'Ks. Điện' ∧ Sal > 3500

m_3: ¬(Title = 'Ks. Điện') ∧ Sal ≤ 3500

m_4: ¬(Title = 'Ks. Điện') ∧ Sal > 3500

. . .

Tất nhiên các vị từ đơn giản cũng được coi là một bộ phận của các vị từ hội sơ cấp và thực ra m_3 và m_4 có thể viết bằng cách sử dụng một vị từ tương đương, chẳng hạn: *Title ≠ 'Ks. Điện' ∧ Sal ≤ 3500*. Nếu ta cứ xây dựng vị từ hội một cách máy móc thì có một số trường hợp có thể vô nghĩa đối với quan hệ.

Các thông tin có liên quan đến vị từ hội sơ cấp là *độ tuyển hội sơ cấp* và *tần số truy xuất*. Độ tuyển hội sơ cấp đo số lượng các bộ của quan hệ được truy xuất bởi câu vấn tin sử dụng vị từ hội sơ cấp đó. Tần số truy xuất để chỉ tần số các ứng dụng truy xuất dữ liệu có sử dụng câu vấn tin sử dụng vị từ hội sơ cấp.

2.2.2. Phân mảnh ngang nguyên thủy.

Phân mảnh ngang nguyên thủy là một phép chọn trên quan hệ chủ của một lược đồ R.

$$R_i = \delta_{Fi}(R) \qquad i = 1, ..., t$$

ở đây F_i là một công thức chọn sử dụng một vị từ hội sơ cấp m_i.

Ví dụ 6.4: Ta có thể phân rã quan hệ PROJ thành PROJ1 và PROJ2 sử dụng các vị từ cơ bản *Budget ≤100000* và *Budget > 100000*

$$PROJ1 = \delta_{Budget \le 100000}(PROJ)$$

$$PROJ2 = \delta_{Budget > 100000}(PROJ)$$

Một vấn đề phức tạp là tập các vị từ hội sơ cấp dùng để phân mảnh có thể thay đổi khi các ứng dụng hoạt động, sẽ gặp rất nhiều khó khăn khi miền xác định của thuộc tính là vô hạn và liên tục. Chẳng hạn khi thêm một bộ mới vào PROJ có budget là 500000 lúc đó ta phải xem xét là đặt nó vào PROJ2 hay phải xây dựng thêm PROJ3 và hạn chế PROJ2.

$$PROJ2 = \delta_{100000<Budget \le 400000}(PROJ)$$

$$PROJ3 = \delta_{Budget > 400000}(PROJ)$$

Ví dụ 6.5: Ta phân mảnh PROJ dựa vào vị trí của dự án.

$$PROJ1 = \delta_{Loc = \text{'Toronto'}}(PROJ)$$

$$PROJ2 = \delta_{Loc = \text{'NewYork'}}(PROJ)$$

$$PROJ3 = \delta_{Loc = \text{'Paris'}}(PROJ)$$

Như vậy từ tập M_i các vị từ hội sơ cấp ta có tập mảnh ngang tương ứng R_i i=1..t được xây dựng trên phép chọn sử dụng m_i, ta gọi tập $\{R_i\}$ là tập các mảnh hội sơ cấp. Số các mảnh ngang phụ thuộc vào tập vị từ hội sơ cấp, như vậy để phân mảnh cần xác định tập các vị từ đơn giản sẽ tạo ra các vị từ hội sơ cấp. Một tập vị từ đơn giản sẽ phải là một tập vị từ có tính đầy đủ và cực tiểu. Tính đầy đủ được hiểu là xác suất mỗi ứng dụng truy xuất đến một bộ nào đó trong một mảnh hội sơ cấp nào đó được sinh ra nhờ tập vị từ đơn giản đó đều bằng nhau. Tính cực tiểu được hiểu là một vị từ là thừa nếu không có ứng dụng nào truy xuất đến mảnh do nó sinh ra.

Ví dụ 6.6: Với ví dụ 6.5 ta xác định tập các P_{PROJ} = {Loc = 'Toronto', Loc = 'NewYork', Loc = 'Paris'}.

- Nếu truy xuất theo vị trí thì P_{PROJ} là đầy đủ
- Nếu có thêm ứng dụng truy các bộ theo Budget ≤100000 thì P_{PROJ} không đầy đủ. Trong trường hợp đó ta phải bổ xung để cho P_{PROJ} trở thành đầy

đủ và lúc này thì ta có P_{PROJ} = {Loc = 'Toronto', Loc = 'NewYork', Loc = 'Paris', Budget ≤100000, Budget >100000}

Ví dụ 6.7: Xét PROJ có các ứng dụng truy xuất theo vị trí và ngân sách như trong ví dụ 6.6

- Tập P_{PROJ} như ví dụ 6.6 là đơn giản và cực tiểu.
- Nếu bổ xung thêm vị từ đơn giản PName = 'Xây dựng' vào P_{PROJ} thì làm cho tập vị từ này không cực tiểu vì không có ứng dụng nào truy xuất đến mảnh do nó sinh ra.

Khi một tập vị từ là cực tiểu thì tất cả các vị từ trong đó đều sinh ra phân mảnh được truy xuất bởi ít nhất một ứng dụng, ta gọi những vị từ đó là ***có liên đới***.

Bước 1: Thuật toán tìm tập vị từ đầy đủ và cực tiểu.

Quy tắc cơ bản về DD&CT: Một quan hệ hoặc một mảnh được phân hoạch thành ít nhất hai phần và chúng được truy xuất khác nhau bởi ít nhất một ứng dụng.

Ta gọi f_i của P_R là mảnh f_i được sinh ra từ một vị từ hội sơ cấp trong P_R.

Thuật toán COM_MIN

Đầu vào R là quan hệ; P_R là tập vị từ đơn giản.

Đầu ra $P_{R'}$ là tập vị từ đơn giản và cực tiểu.

Begin

Tìm một vị từ $p_i \in P_R$ sao cho p_i phân hoạch R theo quy tắc cơ bản DD&CT;

$P_{R'} = \{p_i\}$;

$P_R = P_R - \{p_i\}$;

F = $\{f_i\}$ /* f_i là mảnh hội sơ cấp sinh ra bởi p_i*/

Do

Begin

Tìm một $p_j \in P_R$ sao cho p_j phân hoạch một mảnh f_k của $P_{R'}$ theo quy tắc cơ bản DD&CT ;

$P_{R'} = P_{R'} \cup \{p_j\}$;

$P_R = P_R - \{p_j\}$;

$F = F \cup \{f_j\}$; /* f_j sinh ra bởi p_j */

If $\exists p_k \in P_{R'}$ là một vị từ không có liên đới **then**

Begin

$P_{R'} = P_{R'} - \{p_k\}$;

$F = F - \{f_k\}$

End ;

End ;

Until $P_{R'}$ là đầy đủ

End.

Bước 2 : Tính tập vị từ hội sơ cấp từ tập đầy đủ và cực tiểu.

Việc tính toán này rất dễ nhưng hay dẫn đến những tập vị từ hội sơ cấp rất lớn do việc tính máy móc. Việc giản ước tập vị từ hội sơ cấp được thực hiện ở bước thứ 3.

Bước 3 : Loại bỏ những vị từ hội sơ cấp vô nghĩa.

Việc này đầu tiên phải xác định những vị từ mâu thuẫn với tập các phép kéo theo. Ví dụ , $P_{R'} = \{p_1, p_2\}$ với Att là thuộc tính và $\{V_1, V_2\}$ là miền thuộc tính của Att ta có thể giả thiết :

$p_1 : Att = V_1$ và $p_2 : Att = V_2$

Vậy ta có các phép kéo theo

$(Att = V_1) \Rightarrow \lceil (Att = V_2)$

$\lceil (Att = V_1) \Rightarrow (Att = V_2)$

Ta có các vị từ hội sơ cấp được tính theo quy tắc:

$m_1: (Att = V_1) \wedge (Att = V_2)$

$m_2: (Att = V_1) \wedge \lceil (Att = V_2)$

$m_3: \lceil (Att = V_1) \wedge (Att = V_2)$

$m_4: \lceil (Att = V_1) \wedge \lceil (Att = V_2)$

Các vị từ m_1 và m_4 mâu thuẫn với các phép kéo theo chúng ta đã xác định ở trên và vì thế chúng ta sẽ loại nó ra khỏi $P_{R'}$.

Bước 4: Thuật toán tìm tập vị từ hội sơ cấp có nghĩa.

Thuật toán PHORIZONTAL

Đầu vào R là môt quan hệ

Đầu ra M là tập các vị từ hội sơ cấp có nghĩa.

Begin

$P_{R'}$ = COM_MIN (R, P_R) ;

Tính tập M các vị từ hội sơ cấp từ $P_{R'}$;

Tính tập các I các phép kéo theo giữa các $p_i \in P_{R'}$;

For mỗi $m_i \in M$ **Do**

If m_i mâu thuẫn với I **then** M = M – {m_i}

End.

Ví dụ 6.8 : Giả sử có quan hệ PAY và PROJ phải phân mảnh ngang nguyên thủy

Giả sử chỉ có một ứng dụng truy xuất PAY, mẫu tin nhân viên được lưu trữ tại hai nơi. Một nơi quản lý thông tin của các nhân viên có lương cao hơn 3500 và nơi kia là các nhân viên có lương từ 3500 trở xuống. Vì vậy câu vấn tin của ứng dụng sẽ được truy xuất ở cả hai nơi. Tập vị từ đơn giản dùng để phân hoạch PAY :

p_1 : Sal ≤ 3500

p_2 : Sal > 3500

Từ đó ta có tập vị từ đơn giản khởi đầu là P_R = {p_1, p_2}. Áp dụng thuật toán COM_MIN với khởi đầu i=1 ta có $P_{R'}$ = {p_1}. Đây chính là tập đầy đủ và cực tiểu vì p_2 không phân hoạch f_1 là mảnh hội sơ cấp được sinh ra từ p_1. Vậy chúng ta có các vị từ hội sơ cấp sau :

m_1 : Sal ≤ 3500

m_2 : ⌈(Sal ≤ 3500) tương đương với (Sal > 3500)

Cuối cùng chúng ta có các mảnh ngang nguyên thủy của PAY được phân hoạch theo m_1 và m_2:

PAY2

Title	*Sal*
Ks. Điện	4000
Ks. Hệ thống	7000

PAY1

Title	*Sal*
Ks. Cơ khí	3500
Ks. Lập trình	2000

Giả sử có hai ứng dụng truy xuất đến PROJ. Một ứng dụng truy xuất theo vị trí, chẳng hạn ta có câu truy vấn:

```
Select    PName, Budget
From      PROJ
Where     Loc = Value
```

Từ đó ta có:

P_{PROJ} = { p_1: Loc = ‘Toronto’; p_2: Loc = ‘New York’; p_3: Loc = ‘Paris’}

Một ứng dụng khác truy xuất theo ngân sách, ta thêm vào các vị từ:

p_4: Budget ≤ 100000 và p_5: Budget > 100000

Cuối cùng ta có tập các vị từ đơn giản là:

P_{PROJ} = { p_1: Loc = ‘Toronto’; p_2: Loc = ‘New York’; p_3: Loc = ‘Paris’; p_4: Budget ≤ 100000; p_5: Budget > 100000}

Thực hiện thuật toán COM_MIN với P_{PROJ} ta thấy tập P_{PROJ} là đầy đủ và cực tiểu.

Xây dựng tập vị từ hội sơ cấp M:

m_1: Loc = ‘Toronto’ ∧ Budget ≤ 100000

m_2: Loc = ‘Toronto’ ∧ Budget > 100000

m_3: Loc = ‘New York’ ∧ Budget ≤ 100000

m_4: Loc = ‘New York’ ∧ Budget > 100000

m_5: Loc = ‘Paris’ ∧ Budget ≤ 100000

m_6: Loc = ‘Paris’ ∧ Budget > 100000

m7: $p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5$

. . .

Các phép kéo theo hiển nhiên:

i_1: $p_1 \Rightarrow \lceil p_2 \wedge \lceil p_3$

i_2: $p_2 \Rightarrow \lceil p_1 \wedge \lceil p_3$

i_3: $p_3 \Rightarrow \lceil p_1 \wedge \lceil p_2$

i_4: $p_4 \Rightarrow \lceil p_5$

i_5: $p_5 \Rightarrow \lceil p_4$

i_6: $\lceil p_4 \Rightarrow p_5$

i_7: $\lceil p_5 \Rightarrow p_4$

Xét trong tập M chỉ có các vị từ hội chứa hai vị từ đơn giản là không xảy ra mâu thuẫn với các phép kéo theo, vì vậy chúng ta loại bỏ hết các vị từ hội chỉ giữ lại các vị từ đó. Như vậy chúng ta có tập vị từ hội sơ cấp cuối cùng là:

M = {m_1, m_2, m_3, m_4, m_5, m_6}

Phân mảnh ngang nguyên thủy quan hệ PROJ theo M (lưu ý là PROJ1 và PROJ6 là rỗng) chúng ta có:

PROJ2

PNo	*PName*	*Budget*	*Loc*
P1	Thiết bị	150000	Toronto

PROJ3

PNo	*PName*	*Budget*	*Loc*
P2	CSDL	125000	NewYork

PROJ4

PNo	*PName*	*Budget*	*Loc*
P3	Games	75000	NewYork

PROJ5

PNo	*PName*	*Budget*	*Loc*
P4	CAD	100000	Paris

2.2.3. Phân mảnh ngang dẫn xuất.

Phân mảnh ngang dẫn xuất được định nghĩa dựa trên một sự phân mảnh ngang một quan hệ thành viên của một đường nối dựa theo phép toán chọn trên quan hệ chủ nhân của đường nối đó, hay ta còn gọi đó là sự phân mảnh quan hệ thành viên dựa trên cơ sở phân mảnh quan hệ chủ nhân.

Cho trước một đường nối L, ta có: Owner (L) = S và Member (L) = R. Định nghĩa các mảnh ngang dẫn xuất của R như sau:

$$R_i = R \ \theta \ S_i \quad \text{với } i=1..s$$

trong đó s là số lượng các mảnh ngang trên R, $S_i = \delta_{Fi}(S)$ là mảnh ngang nguyên thủy được xây dựng từ vị từ hội sơ cấp F_i, θ là phép liên kết bằng trên khóa kết nối của chủ nhân và thành viên.

Ví dụ 6.9: xét sơ đồ quan hệ PAY với EMP có đường kết nối L1, ta định nghĩa phép liên kết bằng θ: *PAY.Title = EMP.Title.*

Trên quan hệ PAY ta có M_{PAY} = { m_1: Sal ≤ 3500; m_2: Sal >3500} từ đó PAY được chia thành các mảnh hội sơ cấp PAY1 và PAY2.

$$PAY1 = \delta_{Sal \le 3500}(PAY)$$

$$PAY2 = \delta_{Sal > 3500}(PAY)$$

Ta có các mảnh ngang dẫn xuất: EMP1 = EMP θ PAY1

EMP2 = EMP θ PAY2

EMP1

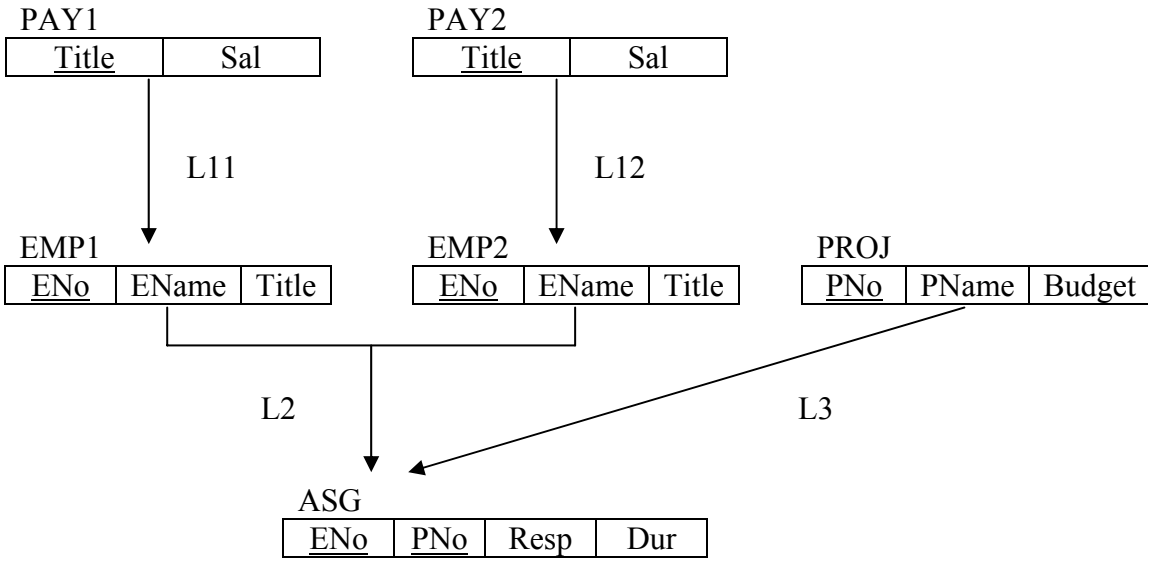
ENo	*EName*	*Title*
E1	John	Ks. Điện

EMP2

ENo	*EName*	*Title*
E4	Bush	Ks. Lập trình

E2	Mary	Ks. Hệ thống
E3	Bill	Ks. Cơ khí
E5	Blair	Ks. Hệ thống
E6	Tom	Ks. Điện
E7	Algor	Ks. Cơ khí
E8	David	Ks. Điện

Sơ đồ liên kết của cơ sở dữ liệu sau khi phân mảnh:



Ta có một số nhận xét quan trọng sau đây:

+ Thuật toán phân mảnh dẫn xuất cần có tập các phân hoạch quan hệ chủ nhân - thành viên, tập vị từ liên kết quan hệ giữa chủ nhân và thành viên.

+ Nếu một quan hệ là thành viên của nhiều hơn một chủ nhân thì vấn đề sẽ trở nên phức tạp hơn.

+ Phân mảnh dẫn xuất sẽ gây nên phân mảnh lan truyền.

2.3. Phương pháp phân mảnh dọc.

Ý nghĩa của phân mảnh dọc là tạo ra các quan hệ nhỏ hơn để sao cho giảm tối đa thời gian thực hiện của các ứng dụng chạy trên mảnh đó. Việc phân mảnh dọc là hoạt động chia một quan hệ R thành các mảnh con R_1, R_2, ..., R_n sao cho mỗi mảnh con chứa tập con thuộc tính và chứa cả khóa của R. Với cách đặt vấn đề như vậy thì

việc phân mảnh dọc không chỉ là bài toán của hệ cơ sở dữ liệu phân tán mà còn là bài toán của ngay cả hệ cơ sở dữ liệu tập trung.

Phân mảnh dọc là một bài toán hết sức phức tạp, người ta đã chứng minh được rằng nếu quan hệ có *m* thuộc tính không phải là thuộc tính khóa thì số lượng các mảnh dọc được phân ra là số Bell thứ *m* (kí hiệu B(m)), số này tăng rất nhanh với số *m* lớn và đạt đến m^m. Chẳng hạn m=10 thì B(m)≈115.000, với m=15 thì B(m)≈10^9, với m=30 thì B(m)≈10^{23}. Vì vậy bài toán phân mảnh dọc phải sử dụng đến các thuật giải heuristic. Có hai phương pháp chính đã được nghiên cứu đó là *phương pháp nhóm* và *phương pháp tách*, trong hai phương pháp thì phương pháp tách tỏ ra có sự tối ưu hơn.

Phương pháp nhóm: Khởi đầu bằng tập các mảnh, mỗi mảnh có một thuộc tính, tại mỗi bước ghép một số mảnh lại cho đến khi thỏa mãn một tiêu chuẩn nào đó.

Phương pháp tách: Tại mỗi bước tìm một phân hoạch có lợi cho việc truy xuất của ứng dụng trên các thuộc tính của nó.

Thông tin dùng để phân mảnh dọc có liên quan đến các ứng dụng, một mảnh dọc thường chứa các thuộc tính thường xuyên được truy xuất chung bởi một ứng dụng, người ta tìm cách lượng hóa khái niệm này bằng một số đo gọi là "ái lực" (affinity – ái lực hoặc sự lôi cuốn). Số đo này có thể tính được khi ta tính được tần số truy xuất tới các thuộc tính đó của ứng dụng. Trên cơ sở khái niệm "ái lực" và tính được *độ sử dụng* thuộc tính của các câu vấn tin của ứng dụng người ta đã xây dựng được giải thuật tách rất hữu hiệu.

Gọi Q = {q_1, q_2, . . ., q_t} là tập các câu vấn tin mà ứng dụng sẽ truy xuất trên quan hệ R(A_1, A_2, . . ., A_n). Với mỗi câu vấn tin q_i và thuộc tính A_j chúng ta sẽ đưa ra một giá trị sử dụng thuộc tính, kí hiệu là use (q_i, A_j) được định nghĩa như sau:

$$\text{use}(q_i, A_j) = \begin{cases} 1 & \text{nếu } A_j \text{ được vấn tin } q_i \text{ sử dụng} \\ 0 & \text{trong trường hợp ngược lại.} \end{cases}$$

các giá trị use (q_i, *) rất dễ xác định nếu chúng ta biết được các ứng dụng chạy trên CSDL.

Ví dụ 6.10: Xét quan hệ PROJ, giả sử các ứng dụng sử dụng câu vấn tin SQL truy xuất đến nó:

q1: Tìm ngân sách của dự án theo mã số.

```
SELECT    Budget
FROM      PROJ
WHERE     PNo = V
```

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

```
SELECT    PName, Budget
FROM      PROJ
```

q3: Tìm tên của dự án theo vị trí.

```
SELECT    PName
FROM      PROJ
WHERE     Loc = V
```

q4: Tìm tổng ngân sách dự án tại mỗi vị trí.

```
SELECT    Sum (Budget)
FROM      PROJ
WHERE     Loc = V
```

Để thuận tiện ta kí hiệu A_1 = PNo, A_2 = PName; A_3 = Budget; A_4 = Loc. Chúng ta có ma trận sau đây

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Ta nhận xét rằng giá trị sử dụng không chứa thông tin về độ lớn của tần số ứng dụng, số đo này nằm trong định nghĩa về số đo ái lực thuộc tính aff (A_i, A_j)

$$\text{aff}(A_i, A_j) = \sum_{k:use(q_k, A_i)=1 \wedge use(q_k, A_j)=1} \sum_{\forall S_l} ref(q_k).acc_l(q_k)$$

trong đó $ref(q_k)$ là số truy xuất đến các thuộc tính (A_i, A_j) cho mỗi ứng dụng của q_k tại vị trí S_l và $acc(q_k)$ là kí hiệu số đo tần số truy xuất ứng dụng. Kết quả tính toán được một ma trận vuông nxn và ta gọi nó là ma trận ái lực thuộc tính AA.

Ví dụ 6.11: Tiếp tục với ví dụ trên và để cho đơn giản chúng ta giả sử $ref(q_k) = 1$ cho tất cả q_k và S_l. Số đo tần số truy xuất ứng dụng giả thiết như sau:

$acc_1(q_1) = 15$	$acc_2(q_1) = 20$	$acc_3(q_1) = 10$
$acc_1(q_2) = 5$	$acc_2(q_2) = 0$	$acc_3(q_2) = 0$
$acc_1(q_3) = 25$	$acc_2(q_3) = 25$	$acc_3(q_3) = 25$
$acc_1(q_4) = 3$	$acc_2(q_4) = 0$	$acc_3(q_4) = 0$

Như vậy chúng ta tính số đo ái lực giữa các thuộc tính A_1 và A_3 và bởi vì ứng dụng duy nhất truy xuất đến cả hai thuộc tính này là q_1 nên ta có:

$$aff(A_1, A_3) = \sum_{k=1}^{1}\sum_{l=1}^{3} acc_l(q_k) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

Ma trận ái lực thuộc tính đầy đủ như sau:

AA =		A_1	A_2	A_3	A_4
	A_1	45	0	45	0
	A_2	0	80	5	75
	A_3	45	5	53	3
	A_4	0	75	3	78

Thuật toán tụ nhóm.

Mục tiêu của thuật toán này là tìm một phương pháp nào đó để nhóm các thuộc tính của một quan hệ lại dựa trên các giá trị ái lực thuộc tính trong AA. Ý tưởng chính của thuật toán là từ một ma trận ái lực thuộc tính AA sinh ra một ma trận ái lực tụ CA dựa trên các hoán vị hàng và cột, hoán vị được thực hiện sao cho số đo ái lực chung AM là lớn nhất.

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

trong đó $aff(A_0,A_j) = aff(A_i,A_0) = aff(A_{n+1},A_j) = aff(A_i,A_{n+1}) = 0$ là các điều kiện biên khi một thuộc tính được đặt vào CA vào bên trái của thuộc tính cận trái hoặc về bên phải của thuộc tính cận phải trong các hóan vị cột, tương tự cận trên dưới đối với hoán vị hàng. Vì ma trân ái lực AA có tính đối xứng nên công thức trên có thể thu gọn:

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

Chúng ta định nghĩa cầu nối (bond) giữa hai thuộc tính A_x và A_y là:

$$bond(A_x,A_y) = \sum_{z=1}^{n} aff(A_z,A_x).aff(A_z,A_y)$$

dựa vào định nghĩa đó chúng ta có thể viết lại AM như sau:

$$AM = \sum_{j=1}^{n} [bond(A_j,A_{j-1}) + bond(A_j,A_{j+1})]$$

Bây giờ chúng ta xét dãy thuộc tính như sau:

$$\underbrace{A_1 \dots A_{i-1}}_{AM'} \quad A_i\, A_j \quad \underbrace{A_{j+1} \dots A_n}_{AM''}$$

số đo ái lực chung cho các thuộc tính này là:

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

$= AM'+AM\ ''+ bond(A_{i-1},A_i)+bond(A_j,A_{j+1})+ 2bond(A_i,A_j)$

Khi đặt một thuộc tính mới A_k giữa các thuộc tính A_i và A_j thì số đo ái lực chung mới là:

$AM_{new} = AM'+AM\ ''+ bond(A_{i-1},A_i)+$
$bond(A_i,A_k)+bond(A_k,A_i)+bond(A_k,A_j)+bond(A_j,A_k)+bond(A_j,A_{j+1})$
$= AM'+AM\ ''+ bond(A_{i-1},A_i)+bond(A_j,A_{j+1})+$
$2bond(A_i,A_k) + 2bond(A_k,A_j)$

Đóng góp thực cho số đo ái lực chung khi đặt A_k giữa A_i và A_j là:

$cont(A_i,A_k,A_j) = AM_{new} – AM_{old} = 2bond(A_i,A_k) + 2bond(A_k,A_j) - 2bond(A_i,A_j)$

Ví dụ 6.12: Với ma trận AA được tính ở trên, tính đóng góp thực khi chuyển thuộc tính A_4 vào giữa các thuộc tính A_1 và A_2:

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

Ta có:

$bond(A_1,A_4) = 45*0 + 0*75 + 45*3 + 0*78 = 135$

$bond(A_4,A_2) = 11865$

$bond(A_1,A_2) = 225$

vì vậy: $cont(A_1,A_4,A_2) = 2*135 + 2*11865 – 2*225 = 23550$

Thuật toán năng lượng nối BEA (Bond Energy Algorithm)

Thuật toán năng lượng nối được thực hiện qua ba bước.

B1. Khởi gán. Đặt và cố định một trong các cột của AA vào trong CA. Cột 1 được chọn trong thuật toán này.

B2. Thực hiện lặp. Lấy lần lượt một trong n-i cột còn lại (i là số cột đã đặt vào trong CA) và thử đặt chúng vào i+1 vị trí còn lại trong ma trận CA. Nơi đặt được chọn sao cho nó đóng góp nhiều nhất cho số ái lực chung được mô tả ở trên. Việc lặp được kết thúc khi không còn cột nào để đặt

B3. Sắp thứ tự hàng. Một khi thứ tự cột đã được xác định, các hàng cũng cần được đặt lại để các vị trí tương đối của chúng phù hợp với các vị trí tương đối của cột

Thuật toán BEA

Đầu vào: AA ma trận ái lực thuộc tính

Đầu ra: CA ma trận ái lực tụ.

Begin

/* Khởi gán */

CA(*,1) := AA(*,1) ;

CA(*,2) := AA(*,2) ;

index := 3 ;

While index <= n **Do** /*Chọn vị trí tốt nhất cho thuộc tính AA_{index} */

Begin

For i :=1 **To** index -1 **Do** tính cont (A_{i-1},A_{index},A_i);

tính cont ($A_{index-1}$, A_{index}, $A_{index+1}$);

loc := nơi đặt được chọn bởi giá trị cont lớn nhất

For j := index **DownTo** loc **Do** CA(*,j) := AA(*,j-1);

CA(*,loc) := AA(*,index);

index := index + 1

End

Sắp thứ tự các hàng theo thứ tự tương đối của cột.

End.

Ví dụ 6.13. Tiếp tục với những kết quả tính toán ở những ví dụ trên, chúng ta xem xét quá trình gom tụ các thuộc tính của quan hệ PROJ.

Khởi đầu chúng ta đặt cột 1 và 2 của AA vào ma trận CA. Tiếp theo chúng ta xét cột 3 (thuộc tính A_3), có ba cách đặt mô tả theo vị trí là 3-1-2, 1-3-2 hoặc 1-2-3. Chúng ta tính đóng góp cho số đo ái lực chung của mỗi khả năng này :

Thứ tự 0-3-1 :

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

chúng ta biết rằng $bond(A_0,A_1) = bond(A_0,A_3) = 0$, vì vậy:

$cont(A_0,A_3,A_1) = 2bond(A_3,A_1) = 2(45*48+5*0+53+45+3*0) = 8820$

Thứ tự 1-3-2 :

$bond(A_1,A_3) = bond(A_3,A_1) = 4410$

$bond(A_3,A_2) = 890$

$bond(A_1,A_2) = 225$

$cont(A_1,A_3,A_2) = 2bond(A_1,A_3) + 2bond(A_3,A_2) - 2bond(A_1,A_2) = 10150$

Thứ tự 2-3-4 :

$bond(A_1,A_4) = 890$

$bond(A_3,A_4) = bond(A_2,A_4) = 0$

$cont(A_2,A_3,A_4) = 2bond(A_2,A_3) + 2bond(A_3,A_4) - 2bond(A_2,A_4) = 1780$

Trong những cách tính toán trên lưu ý rằng cột A_0 và cột A_4 là các vị trí rỗng của ma trận CA trong ngữ cảnh hiện tại, không được nhầm lẫn với thuộc tính A_4.Ta thấy thứ tự 1-3-2 có số đóng góp lớn nhất nên vị trí này được chọn.

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

(a)

	A_1	A_3	A_2
A_1	45	45	0
A_2	0	5	80
A_3	45	53	5
A_4	0	3	75

(b)

	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_2	0	5	80	75
A_3	45	53	5	3
A_4	0	3	75	78

(c)

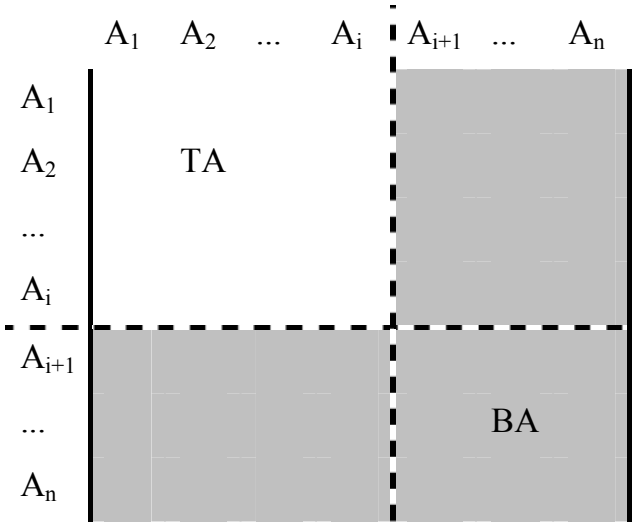
	A_1	A_3	A_2	A_4
A_1	45	45	0	0
A_3	45	53	5	3
A_2	0	5	80	75
A_4	0	3	75	78

(d)

Trong bảng (d) ở trên ta thấy ma trận có hai tụ, góc trên trái bao gồm các giá trị ái lực nhỏ, góc dưới phải có các giá trị ái lực lớn, tuy nhiên trên thực tế sự tách biệt này không hoàn toàn rõ ràng. Nếu ma trận CA lớn ta sẽ thấy có nhiều tụ hơn vì vậy sẽ dẫn đến có nhiều phân hoạch để lựa chọn hơn.

Thuật toán phân hoạch thuộc tính.

Xét ma trận tụ, một điểm nằm trên đường chéo sẽ xác định hai tập thuộc tính. Giả sử điểm đó nằm ở cột i thì các tập đó là $\{A_1, \ldots, A_i\}$ và $\{A_{i+1}, \ldots, A_n\}$, ta gọi là tập đỉnh (top) TA và tập đáy (bottom) BA.



Xét tập ứng dụng $Q = \{q_1, q_2, \ldots, q_t\}$, ta định nghĩa các tập ứng dụng chỉ truy xuất TA, chỉ truy xuất BA hoặc cả hai. $AQ(q_i)$ tập thuộc tính được truy xuất bởi ứng dụng q_i, TQ và BQ là tập ứng dụng chỉ truy xuất TA và BA, OQ là tập ứng dụng truy xuất cả hai.

$$AQ(q_i) = \{A_j \mid use(q_i, A_j) = 1\} \qquad TQ = \{q_i \mid AQ(q_i) \subseteq TA\}$$

$$BQ = \{q_i \mid AQ(q_i) \subseteq BA\} \qquad OQ = Q - \{TQ \cup BQ\}$$

Giả sử có n thuộc tính thì chúng ta có n-1 vị trí có thể chọn cho điểm phân chia. Vị trí tốt nhất để chọn sao cho tổng các truy xuất chỉ một mảnh là lớn nhất còn tổng truy xuất cả hai mảnh là nhỏ nhất. Chúng ta định nghĩa phương trình chi phí như sau:

$$CQ = \sum_{q_i \in Q} \sum_{\forall S_j} ref_j(q_i) \cdot acc_j(q_i) \qquad CTQ = \sum_{q_i \in TQ} \sum_{\forall S_j} ref_j(q_i) \cdot acc_j(q_i)$$

$$CBQ = \sum_{q_i \in BQ} \sum_{\forall S_j} ref_j(q_i) \cdot acc_j(q_i) \qquad COQ = \sum_{q_i \in OQ} \sum_{\forall S_j} ref_j(q_i) \cdot acc_j(q_i)$$

Phương trình tối ưu hóa xác định điểm x ($1 \le x \le n$) sao cho:

$$z = CTQ * CBQ - COQ^2 \rightarrow \max$$

Để chọn được x theo phương trình tối ưu hóa chúng ta phải xét tất cả n-1 trường hợp. Để cho đơn giản chúng ta chỉ xét trường hợp điểm z là duy nhất và tụ nằm ở góc trên trái và góc dưới phải của ma trận CA. Điểm z chia quan hệ R thành hai mảnh R_1 và R_2 sao cho $R_1 \cap R_2 = K$ (tập thuộc tính khóa chính)

Thuật toán PARTITION

Đầu vào: CA ma trận ái lực tụ, R quan hệ, *ref* ma trận sử dụng thuộc tính, *acc* ma trận tần số truy xuất, K tập thuộc tính khóa chính của R

Đầu ra: F tập các mảnh dọc.

Begin

z là vị trí thuộc cột thứ nhất;

tính CTQ_1;

tính CBQ_1;

tính COQ_1;

best := $CTQ_1 * CBQ_1 - COQ^2_1$

For i := 2 **To** n-1 **Do**

Begin

tính CTQ_i;

tính CBQ_i;

tính COQ_i;

z := $CTQ_i * CBQ_i - COQ^2_i$;

If z > best **Then** best := z

End;

$R_1 := \prod_{TA}(R) \cup K$;

$R_2 := \prod_{BA}(R) \cup K$;

$F := R_1 \cup R_2$

End.

Ví dụ 6.14 : Tiếp tục với các tính toán ở trên và với những dữ liệu đã có.

$acc_1(q_1) = 15$ $acc_2(q_1) = 20$ $acc_3(q_1) = 10$

$acc_1(q_2) = 5$ $acc_2(q_2) = 0$ $acc_3(q_2) = 0$

$acc_1(q_3) = 25$ $acc_2(q_3) = 25$ $acc_3(q_3) = 25$

$acc_1(q_4) = 3$ $acc_2(q_4) = 0$ $acc_3(q_4) = 0$

	A_1	A_2	A_3	A_4
q_1	1	0	1	0
q_2	0	1	1	0
q_3	0	1	0	1
q_4	0	0	1	1

Ở vị trí 1 : TA = $\{A_1\}$, TQ = {}, BQ = $\{q_2, q_3, q_4\}$, OQ = $\{q_1\}$

$CTQ_1 = 0$

$CBQ_1 = acc_1(q_2) + acc_2(q_2) + acc_3(q_2) + acc_1(q_3) + acc_2(q_3) + acc_3(q_3) + acc_1(q_4) + acc_2(q_4) + acc_3(q_4) = 83$

$COQ_1 = 45$

$z = -2025$

Ở vị trí 2 : TA = $\{A_1, A_3\}$, TQ = $\{q_1\}$, BQ = $\{q_3\}$, OQ = $\{q_2, q_4\}$

$CTQ_2 = 45$

$CBQ_2 = 75$

$COQ_2 = 8$

$z = 3311$

Ở vị trí 3 : TA = $\{A_1, A_3, A_2\}$, TQ = $\{q_1, q_2\}$, BQ = {}, OQ = $\{q_3, q_4\}$

$CTQ_3 = 50$

$CBQ_3 = 0$

$COQ_3 = 78$

$z = -6084$

Ta chọn vị trí 3 làm điểm phân chia vì tại vị trí này giá trị chi phí là cao nhất. Như vậy chúng ta có PROJ1 = $\{A_1, A_3\}$ và PROJ2 = $\{A_1, A_2, A_4\}$. Tức là PROJ1 = {PNo, Budget} và PROJ2 = {PNo, PName, Loc}

2.4. Phân tán tài nguyên

Phân tán tài nguyên không phải là một bài toán chỉ của hệ cơ sở dữ liệu phân tán, mặc dù đối với cơ sở dữ liệu phân tán nó có những đặc trưng riêng, bài toán này đã được biết đến trong các lý thuyết mạng với cách đặt vấn đề về cấp phát các tập tin.

Giả sử rằng ta có một tập các mảnh $F = \{F_1, F_2, ..., F_n\}$ và một mạng bao gồm các nút $S = \{S_1, S_2, ..., S_m\}$, trên mạng đó có một tập các ứng dụng đang chạy $App = \{A_1, A_2, ..., A_q\}$. Bài toán cấp phát yêu cầu tìm một cách phân phối tối ưu các mảnh F_i cho các nút trên S.

Tính tối ưu được hiểu là cấp phát với chi phí nhỏ nhất và với hiệu năng cao nhất. Các chi phí bao gồm không gian lưu trữ mỗi mảnh F_i tại vị trí S_j, chi phí vấn tin, chi phí cập nhật, chi phí cho truyền dữ liệu. Hiệu năng được tính trên cơ sở giảm thời gian đáp ứng và tăng tối đa lưu lượng của hệ thống tại mỗi vị trí.

Đây là một bài toán hết sức phức tạp, người ta phải giải quyết bài toán này bằng các phương pháp heuristic và cũng chưa có một phương pháp nào thật sự hữu hiệu để giải bài toán tối ưu tổng quát.

TÓM TẮT CHƯƠNG

- *Hệ cơ sở dữ liệu phân tán*: Hệ cơ sở dữ liệu phân tán được mô tả như là tập hợp nhiều cơ sở dữ liệu có liên quan logic đến nhau và được phân bố trên mạng máy tính.
- *Hệ Quản trị cơ sở dữ liệu phân tán*: Hệ thống phần mềm cho phép quản lý các hệ cơ sở dữ liệu phân tán và làm cho sự phân tán trở nên "*trong suốt*" đối với người sử dụng
- *Kiến trúc Hệ Quản trị cơ sở dữ liệu phân tán*: Mô hình phổ biến và có nhiều ưu điểm là mô hình client/server.
- *Thiết kế phân tán gồm ba bài toán chủ yếu:* Phân mảnh ngang để chia quan hệ theo các bộ, phân mảnh dọc để chia quan hệ theo thuộc tính và phân tán tài nguyên để cấp phát các tập tin dữ liệu.

CÂU HỎI VÀ BÀI TẬP.

1. Hãy phân tích các đặc trưng cơ bản của hệ cơ sở dữ liệu phân tán. Các khái niệm về "Xử lý phân tán", "Hệ phân tán", "Cơ sở dữ liệu phân tán" giống nhau và khác nhau ở điểm nào?

2. Hãy phân tích các ưu điểm của kiến trúc hệ quản trị cơ sở dữ liệu phân tán theo mô hình khách/đại lý.
3. Hãy mô tả các chức năng của từng tầng trong kiến trúc hệ quản trịc CSDL phân tán theo mô hình khách/đại lý.
4. Phân tích ý nghĩa của các quy tắc phân mảnh trong thiết kế hệ cơ sở dữ liệu phân tán.
5. Hãy nêu và giải thích ý nghĩa của các bài toán phân mảnh và phân tán tài nguyên.
6. Cho các ví dụ để nêu bật ý tưởng cơ bản của các phương pháp phân mảnh ngang.
7. Ý tưởng cơ bản của phương pháp phân mảnh dọc là gì? Cho ví dụ minh họa.