……..o0o……..

Cơ sở dữ liệu nâng cao

CHƯƠNG 1
TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU QUAN HỆ

Trong chương này chúng ta hệ thống lại các khái niệm cơ bản của cơ sở dữ liệu quan hệ. Mục đích là định nghĩa các thuật ngữ, đưa ra bộ khung cơ sở cho các phần sau. Việc chọn mô hình cơ sở dữ liệu quan hệ làm hệ thống cơ sở có nhiều lý do: thứ nhất là cơ sở toán học vững chắc của mô hình quan hệ làm nó trở thành mô hình lý tưởng trong việc giải quyết các vấn đề lý thuyết, hai là phần lớn các vấn đề trình bày trong các chương sau có thể mô tả dễ dàng bởi mô hình quan hệ, ba là thị trường hệ quản trị cơ sở dữ liệu quan hệ rất phát triển và vẫn đang mở rộng và cuối cùng là phần lớn các hệ cơ sở dữ liệu sau này cũng thuộc loại quan hệ.

Mô hình quan hệ có ba đặc trưng cơ bản:

1. Cấu trúc dữ liệu đơn giản. Chúng là các quan hệ (relation), được biểu diễn như các bảng 2 chiều với phần tử là các bản ghi (record, data item). Nó cung cấp đặc tính độc lập ở mức độ cao so với dạng biểu thị vật lý của dữ liệu.
2. Mô hình quan hệ cung cấp một nền tảng vững chắc, bảo đảm được tính nhất quán dữ liệu (data consistency). Thiết kế cơ sở dữ liệu được hỗ trợ bởi quá trình chuẩn hoá, giúp loại bỏ các bất thường dữ liệu. Các trạng thái nhất quán của một cơ sở dữ liệu có thể được định nghĩa và duy trì một cách thống nhất qua các quy tắc toàn vẹn cơ sở dữ liệu (integrity rules).
3. Mô hình quan hệ cho phép các thao tác trên quan hệ theo kiểu tập hợp. Đặc tính này đã dẫn đến việc phát triển các ngôn ngữ phi thủ tục mạnh mẽ với nền tảng là lý thuyết tập hợp (đại số quan hệ) hoặc lôgic (phép tính quan hệ).

1. Khái niệm cơ sở dữ liệu

1.1. Cơ sở dữ liệu

Dữ liệu được lưu trữ trên các thiết bị lưu trữ theo một cấu trúc nào đó để có thể phục vụ cho nhiều người sử dụng với nhiều mục đích khác nhau gọi là *cơ sở dữ liệu.*

1.2. Hệ quản trị cơ sở dữ liệu

Phần mềm cho phép một hoặc nhiều người tạo lập, lưu trữ, cập nhật và khai thác cơ sở dữ liệu gọi là *hệ quản trị cơ sở dữ liệu* (DataBase Management Systems - DBMS).

Vai trò chính của hệ quản trị cơ sở dữ liệu là cho phép người dùng thao tác với dữ liệu thông qua các thuật ngữ trìu tượng, khác với việc máy tính lưu trữ dữ liệu. Theo nghĩa này hệ quản trị cơ sở dữ liệu có nhiệm vụ như là một bộ thông dịch (interpreter) với ngôn ngữ bậc cao nhằm giúp người dùng sử dụng hệ thống mà không cần quan tâm đến cách biểu diễn dữ liệu trong máy hoặc các thuật toán chi tiết. Ví dụ người dùng không cần biết hệ quản trị cơ sở dữ liệu *Access* tổ chức dữ liệu theo kiểu hàm băm, kiểu file chỉ mục hay kiểu cây cân bằng, và cũng không cần biết thuật toán thực hiện lệnh sắp xếp là Quick Sort, thuật toán nổi bọt hay sắp xếp nhị phân ...

Một cơ sở dữ liệu gồm một hoặc nhiều tập tin được thiết kế theo một cấu trúc nhất định và có quan hệ chặt chẽ với nhau. Cơ sở dữ liệu được dùng chung cho

nhiều người và nhiều mục đích khác nhau, vì vậy sẽ tiết kiệm được tài nguyên, giảm thiểu sự trùng lặp thông tin, bảo đảm tính nhất quán thông tin.

1.3. Các đặc trưng của phương pháp cơ sở dữ liệu

+ *Chia sẻ dữ liệu.* Mục đích chính của cách tiếp cận cơ sở dữ liệu là dữ liệu được chia sẻ bởi nhiều người dùng hợp pháp.

+ *Giảm thiểu dư thừa dữ liệu.* Dữ liệu dùng chung cho nhiều bộ phận, thay vì được lưu trữ phân tán trùng lặp nay được lưu trữ tập trung một chỗ theo một cấu trúc thống nhất.

+ *Tính tương thích dữ liệu.* Việc loại bỏ sự dư thừa dữ liệu kéo theo hệ quả là sự tương thích dữ liệu. Ví dụ khi địa chỉ nhân viên thay đổi thì tất cả các bộ phận đều được cập nhật địa chỉ mới.

+ *Tính toàn vẹn dữ liệu* (data integrity). Mỗi cơ sở dữ liệu cần đảm bảo một số loại ràng buộc toàn vẹn (integrity constraints). Đặc biệt khi người dùng thực hiện các thao tác như chèn, xoá hay sửa đổi dữ liệu thì các ràng buộc đó cần phải được kiểm tra một cách chặt chẽ.

+ *Bảo mật dữ liệu (data security).* Khi có nhiều người cùng chia sẻ dữ liệu, việc bảo đảm an toàn dữ liệu và bảo mật thông tin là tối quan trọng. Cần phải có cơ chế bảo mật như mật khẩu (*password*) và phân quyền truy cập dữ liệu (*data access rights*).

+ *Tính đồng bộ dữ liệu* (synchronization). Thông thường cơ sở dữ liệu được nhiều người dùng truy cập đồng thời, gây nên sự cạnh tranh dữ liệu. Vì vậy cần có cơ chế bảo vệ chống sự không tương thích bởi các thao tác cùng lúc lên dữ liệu.

+ *Tính độc lập dữ liệu.* Sự tách biệt cấu trúc mô tả dữ liệu khỏi chương trình ứng dụng sử dụng dữ liệu gọi là độc lập dữ liệu. Điều này cho phép phát triển tổ chức dữ liệu mà không cần sửa đổi chương trình ứng dụng. Sự độc lập dữ liệu là một trong mục tiêu chính của cách tiếp cận cơ sở dữ liệu.

Tương tự như một phần mềm, *vòng đời* của cơ sở dữ liệu gồm có các giai đoạn chính sau:

- Lập kế hoạch cơ sở dữ liệu (database planning).
- Khảo sát, phân tích (study and analysis).
- Thiết kế cơ sở dữ liệu (database design).
- Cài đặt cơ sở dữ liệu (database implementation).
- Bảo trì cơ sở dữ liệu (post-implementation).

1.4. Lược đồ dữ liệu và thể hiện dữ liệu

Khi thiết kế cơ sở dữ liệu ta tạo ra cấu trúc cơ sở dữ liệu, cái đó gọi là *lược đồ dữ liệu.* Ví dụ lược đồ dữ liệu hồ sơ nhân sự gồm các thành phần sau:

Họ tên, ngày sinh, hệ số lương

Các thành phần của lược đồ dữ liệu gọi là *thuộc tính* hoặc *trường*.

Khi sử dụng cơ sở dữ liệu thì ta làm việc với dữ liệu thật sự, đó là sự *thể hiện dữ liệu*. Ví dụ đối với lược đồ trên ta có thể có các thể hiện dữ liệu sau:

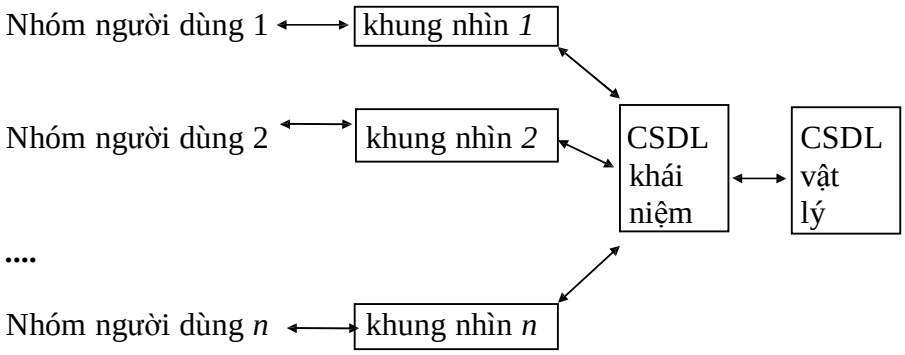
Nguyễn Văn A, 20/08/1970, 3.40

Trần Thị B, 15/05/1962, 4.12

Mỗi thể hiện dữ liệu gọi là *bộ* hay *bản ghi*.

1.5. Các mức trìu tượng dữ liệu

Giữa máy tính thao tác với các bit, người thiết kế cơ sở dữ liệu và người dùng có cách nhìn khác nhau đối với dữ liệu, đó chính là các *mức trìu tượng dữ liệu*. Sơ đồ chuẩn về các mức trìu tượng như sau:



a. Cơ sở dữ liệu vật lý: nằm cố định trong các thiết bị lưu trữ như đĩa và băng từ. Bản thân cơ sở dữ liệu vật lý cũng có nhiều mức trìu tượng khác nhau: từ mức bản ghi và file trong ngôn ngữ lập trình như PASCAL, qua mức bản ghi lôgic được hỗ trợ bởi hệ điều hành, đến mức các bit và địa chỉ vật lý trong các thiết bị lưu trữ.

b. Cơ sở dữ liệu khái niệm (schema): là sự trìu tượng thế giới thực đối với một đối tượng nào đó. Hệ quản trị cơ sở dữ liệu cung cấp ngôn ngữ thiết kế dữ liệu để thiết kế sơ đồ khái niệm. Đây là ngôn ngữ bậc cao cho phép mô tả cơ sở dữ liệu khái niệm bằng ngôn ngữ "mô hình dữ liệu". Một ví dụ điển hình là đồ thị có hướng trong mô hình mạng, trong đó các nút biểu diễn các đơn thể và các cung biểu diễn quan hệ.

c. Khung nhìn hoặc *lược đồ con (subschema)*: là mô hình trìu tượng một phần của cơ sở dữ liệu khái niệm. Có những hệ trang bị công cụ gọi là ngôn ngữ thiết kế khung nhìn cho phép khai báo khung nhìn, và công cụ gọi là ngôn ngữ thao tác dữ liệu khung nhìn để diễn tả câu hỏi và các thao tác đối với khung nhìn.

Theo một nghĩa nào đó, khung nhìn là cơ sở dữ liệu khái niệm và cùng mức trìu tượng như cơ sở dữ liệu khái niệm. Mặt khác khung nhìn có thể trìu tượng hơn theo nghĩa dữ liệu của nó được suy ra từ cơ sở dữ liệu khái niệm.

d. Độc lập dữ liệu

Độc lập dữ liệu dữ liệu giữa các mức trìu tượng có ý nghĩa rất quan trọng đối với cơ sở dữ liệu.

- *Độc lập dữ liệu mức vật lý*: Sự thay đổi lược đồ dữ liệu vật lý không làm thay đổi lược đồ dữ liệu mức khái niệm và mức khung nhìn.

Ta cần hiểu rằng sự thay đổi tổ chức vật lý dữ liệu có thể làm ảnh hưởng đến hiệu quả chương trình ứng dụng. Sự độc lập dữ liệu mức vật lý đảm bảo không phải viết lại chương trình chỉ vì lý do thay đổi cách tổ chức dữ liệu. ý nghĩa của tính độc lập dữ liệu mức vật lý là nó cho phép ta tinh chỉnh cơ sở dữ liệu mức vật lý để tăng hiệu quả sử dụng trong khi các chương trình ứng dụng vẫn chạy bình thường như không có vấn đề gì xảy ra.

- *Độc lập dữ liệu lôgic*: Sự thay đổi lược đồ dữ liệu khái niệm không làm thay đổi khung nhìn.

Trong quá trình sử dụng cơ sở dữ liệu có thể ta phải sửa đổi hiệu chỉnh lược đồ khái niệm, chẳng hạn thêm thông tin về thực thể mà cơ sở dữ liệu mô tả.

1.6. Ngôn ngữ dữ liệu

Mỗi hệ quản trị cơ sở dữ liệu cần phải có ngôn ngữ riêng của mình. Có hai loại ngôn ngữ cơ sở dữ liệu.

a. Ngôn ngữ mô tả dữ liệu (Data Definition Language - DDL). Gồm các lệnh cho phép khai báo, hiệu chỉnh cấu trúc cơ sở dữ liệu, mô tả các mối quan hệ của dữ liệu cũng như các quy tắc áp đặt lên dữ liệu. Ngôn ngữ mô tả dữ liệu được xây dựng dựa trên loại mô hình dữ liệu (mô hình quan hệ, mô hình mạng, mô hình phân cấp, mô hình hướng đối tượng ...) mà hệ quản trị cơ sở dữ liệu tương ứng được thiết kế.

b. Ngôn ngữ thao tác dữ liệu (Data Manipulation Language - DML). Là bộ lệnh cho phép người dùng thực hiện các công việc:

- Cập nhật dữ liệu như thêm, sửa, xoá.
- Truy vấn, tổng hợp dữ liệu.
- Các hàm tính toán.
- Bảo mật dữ liệu.

* *Giao tiếp ngôn ngữ chủ* : là khả năng cho phép chương trình viết trong các ngôn ngữ bậc cao như COBOL, C , ... có thể truy cập xử lý dữ liệu trong cơ sở dữ liệu.

* *Ngôn ngữ truy vấn SQL* : là ngôn ngữ có cú pháp tiếng Anh, giúp người dùng có thể thao tác dữ liệu dễ dàng mà không cần lập trình. Đây là loại ngôn ngữ thế hệ thứ 4 với đặc trưng *phi thủ tục*.

2. Khái niệm cơ sở dữ liệu quan hệ

2.1. Miền

Miền là tập hợp các giá trị. Người ta thường dùng chữ hoa để ký hiệu miền.

◊ *Ví dụ*. Các tập hợp sau là các miền:

- Tập các số nguyên.
- Tập các xâu ký tự độ dài không quá 30 ký tự.
- A = {0,1}.

2.2. Tích Đề-các

Tích Đề-các của các miền $D_1, D_2,..., D_n$, ký hiệu là

$$D_1 \times D_2 \times ... \times D_n,$$

là tập hợp tất cả *n-bộ* $(v_1, v_2, ..., v_n)$ thoả mãn

$$v_1 \in D_1, v_2 \in D_2, ..., v_n \in D_n.$$

Tức là

$$D_1 \times D_2 \times ... \times D_n = \{(v_1, v_2,..., v_n) \mid v_1 \in D_1, v_2 \in D_2, ..., v_n \in D_n \}.$$

◊ *Ví dụ.* Cho $D_1 = \{0,1\}$, $D_2 = \{a,b,c\}$. Khi đó tích đề-các

$$D_1 \times D_2 = \{(0,a),(0,b),(0,c),(1,a),(1,b),(1,c)\}$$

2.3. Quan hệ

Quan hệ là tập con của tích đề-các của 1 hoặc nhiều miền. Quan hệ có thể có hữu hạn hoặc vô hạn số phần tử. Trong giáo trình này ta giả thiết rằng quan hệ có hữu hạn phần tử.

◊ *Ví dụ*

Cho $D_1 = \{0,1\}$, $D_2 = \{a,b,c\}$. Tập **r** $= \{(0,a),(1,b),(1,c)\} \subset D_1 \times D_2$, vậy **r** là quan hệ trên D_1 và D_2.

Ta nói quan hệ **r** có *bậc n* nếu **r** là tập con của tích đề-các của *n* miền.

Mỗi phần tử của quan hệ gọi là *bộ*. Mỗi bộ của quan hệ bậc *n*, còn gọi là *n-bộ*, có *n thành phần*. Mỗi thành phần của bộ là *nguyên tố*, có nghĩa không thể phân tách được thành các thành phần nhỏ hơn.

Để trực quan ta có thể coi quan hệ như một *bảng* trong đó mỗi *hàng* là một bộ và mỗi *cột* ứng với một thành phần.

Dữ liệu được tổ chức dưới dạng các quan hệ có liên quan với nhau gọi là *cơ sở dữ liệu quan hệ.*

Mỗi cột của quan hệ được gán một tên gọi là *thuộc tính*.

Tập hợp tất cả các tên thuộc tính của quan hệ gọi là *lược đồ quan hệ*.

Tập hợp các lược đồ quan hệ của một cơ sở dữ liệu gọi là *lược đồ cơ sở dữ liệu quan hệ.*

· ***Các tính chất của quan hệ***

- Các giá trị trên cột phải cùng một miền giá trị và đơn trị.

Giá trị trên giao của cột và hàng đơn trị, không chấp nhận nhiều giá trị.

- Mỗi hàng là duy nhất.

Không cho phép hai hàng hoàn toàn giống nhau.

◊ *Ký hiệu*

Lược đồ quan hệ R có các thuộc tính $A_1, A_2, ..., A_n$ ký hiệu là

$$R = (A_1, A_2, ..., A_n)$$

Quan hệ **r** với lược đồ $R = (A_1, A_2, ..., A_n)$ có thể viết

r(R) hoặc **r**(A_1, A_2, ..., A_n)

Cho **r** là quan hệ với lược đồ R = (A_1, A_2, ..., A_n), *t* là một bộ của **r**, A ⊂ {A_1, A_2, ..., A_n}. Khi đó *t*(A) ký hiệu bộ các thành phần của *t* ứng với các thuộc tính trong tập A. Nếu A là 1 thuộc tính thì *t*(A) chính là giá trị thành phần ứng với thuộc tính A.

◊ *Ví dụ*

Đây là ví dụ sẽ dùng làm cơ sở dữ liệu mẫu để mô hình hoá một công ty. Các thực thể được mô hình hoá là:

- Các *nhân viên*, ký hiệu EMP, viết tắt từ *employee*.
- Các *dự án*, ký hiệu PROJ, viết tắt từ *project*.

Đối với mỗi nhân viên chúng ta muốn theo dõi các thông tin sau:

- *Mã số nhân viên*, ký hiệu *ENO*, viết tắt từ *employee number*.
- *Tên nhân viên*, ký hiệu *ENAME*, viết tắt từ *employee name*.
- *Chức vụ*, ký hiệu *TITLE*.
- *Lương*, ký hiệu *SAL*, viết tắt từ *salary*.
- *Mã số dự án*, ký hiệu *PNO*, viết tắt từ *project number*.
- *Nhiệm vụ dự án*, ký hiệu *RESP*, viết tắt từ *responsibility*.
- *Thời gian làm việc trong dự án*, ký hiệu *DUR*, viết tắt từ *duration*.

Đối với mỗi dự án chúng ta muốn theo dõi các thông tin sau:

- *Mã số dự án*, ký hiệu *PNO*, viết tắt từ *project number*.
- *Tên dự án*, ký hiệu *PNAME*, viết tắt từ *project name*.
- *Kinh phí dự án*, ký hiệu *BUDGET*.

Các lược đồ quan hệ (relation scheme) cho cơ sở dữ liệu này có thể định nghĩa như sau:

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)

PROJ(*PNO, PNAME, BUDGET*)

Lược đồ EMP có 7 thuộc tính (attribute): *ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*.

Giá trị của *ENO* lấy tự miền chứa các mã số nhân viên, giả sử là D_1, Giá trị của *ENAME* lấy tự miền chứa các tên nhân viên hợp lệ, giả sử là D_2, ...

Đây là một thể hiện cơ sở dữ liệu mẫu của chúng ta gồm hai bảng như sau:

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E2	M.Smith	Syst.Anal.	34000	P1	Analyst	24
E2	M.Smith	Syst.Anal.	34000	P2	Analyst	6
E3	A.Lee	Mech.Eng.	27000	P3	Consultant	10
E3	A.Lee	Mech.Eng.	27000	P4	Engineer	48
E4	J.Miller	Programmer	24000	P2	Programmer	18
E5	B.Casey	Syst.Anal.	34000	P2	Manager	24

E6	L.Chu	Elect.Eng.	40000	P4	Manager	48
E7	R.David	Mech.Eng.	27000	P3	Engineer	36
E8	J.Jones	Syst.Anal.	34000	P3	Manager	40

PROJ

PNO	*PNAME*	*BUDGET*
P1	Instrumentation	150000
P2	Database Development	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Các cột của bảng tương ứng các thuộc tính của quan hệ. Các thông tin nhập theo hàng tương ứng với các bộ hay bản ghi. Dòng trên cùng biểu diễn lược đồ quan hệ của bảng.

Một giá trị của thuộc tính, chẳng hạn lương của nhân viên, thời gian tham gia dự án,... tại thời điểm nào đó có thể chưa được xác định. Khi đó có nhiều cách diễn giải khác nhau như "chưa được biết" hay "chưa áp dụng được". "Giá trị đặc biệt" thường được gọi là ***null***. Giá trị ***null*** phải khác các giá trị trong miền thuộc tính, và cũng cần phân biệt nó với giá trị *zero* (với thuộc tính kiểu số) hay giá trị *rỗng* (với thuộc tính kiểu ký tự).

2.4. Khoá

a) Siêu khoá

Cho lược đồ quan hệ $R=(A_1, A_2, ..., A_n)$ và tập con $S \subset \{ A_1, A_2, ..., A_n \}$. Tập S gọi là *siêu khoá* (superkey) của lược đồ R nếu các thuộc tính của S xác định duy nhất các bộ của mỗi quan hệ của lược đồ R, tức là với mọi quan hệ **r** của lược đồ R phải thoả mãn:

$$\forall\, t_1, t_2 \in \mathbf{r}: t_1 \neq t_2 \Rightarrow \exists\, A \in S: t_1(A) \neq t_2(A)$$

Lưu ý rằng theo định nghĩa, mỗi bộ là duy nhất nên đối với mỗi lược đồ quan hệ, *tập hợp tất cả thuộc tính là siêu khoá*. Siêu khoá là cơ sở để phân biệt 2 bộ khác nhau trong 1 quan hệ. Một lược đồ có thể có nhiều siêu khoá. Tính chất của siêu khoá là quy luật được xác định trong quá trình phân tích thiết kế cơ sở dữ liệu.

◊ *Ghi chú*

(1) Tập tất cả thuộc tính R là siêu khoá (tầm thường).

(2) $S \subset T \subset R$ & S là siêu khoá $\Rightarrow$ T là siêu khoá.

b) Khoá

Tập K các thuộc tính của lược đồ R là *khoá (key)* nếu K là siêu khoá *cực tiểu*, tức là mọi tập con thực sự của K không phải là siêu khoá.

· *Mệnh đề*: Mọi lược đồ quan hệ luôn có khóa.

Chứng minh. Mệnh đề suy ra từ sự tồn tại phần tử cực tiểu trong tập có quan hệ thứ tự.

◊ *Ví dụ*

Lược đồ PROJ có khoá là *PNO*.

Lược đồ EMP có khoá là (*ENO, PNO*).

Các thuộc tính thuộc khoá nào đó gọi là *thuộc tính khoá* hay *thuộc tính nguyên tố*.

Thuộc tính không phải thuộc tính khoá gọi là *thuộc tính không khoá*.

Mỗi quan hệ có ít nhất một khoá. Trường hợp có nhiều khoá thì gọi các khoá đó là *khoá dự tuyển* (candidate key), trong đó có một khoá là *khoá chính* (primary key).

c) Khoá ngoại

Cho lược đồ R và lược đồ Q. Tập con H các thuộc tính của R gọi là *khoá ngoại* của R *tham chiếu* đến lược đồ Q, nếu Q có khoá K gồm các thuộc tính (có thể dưới tên khác) của H thoả mãn:

Với mọi quan hệ ***r*** và ***q*** là các quan hệ của 1 cơ sở dữ liệu ứng với lược đồ R và Q ta có:

$$\forall x \in \mathbf{r}\ \exists y \in \mathbf{q}: x(H) = y(K)$$

◊ *Ví dụ*

Lược đồ EMP có khoá ngoại *PNO* tham chiếu đến khoá *PNO* của lược đồ PROJ.

Ở dạng bảng cơ sở dữ liệu mẫu của chúng ta gồm 2 bảng như sau

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
				P1		

PROJ

PNO	*PNAME*	*BUDGET*
P1		

3. Quy tắc toàn vẹn

Qui tắc toàn vẹn (*integrity rule*) là các ràng buộc đảm bảo trạng thái nhất quán của cơ sở dữ liệu. Chúng thường được diễn tả như là các ràng buộc toàn vẹn.

Có các loại qui tắc toàn vẹn sau: *Toàn vẹn thực thể* (Entity integrity), *Miền giá trị* (Domains integrity), *Toàn vẹn tham chiếu* (Referential integrity), *Thao tác bẫy* (Triggering operations).

3.1. Qui tắc toàn vẹn thực thể

Qui tắc *toàn vẹn thực thể* yêu cầu thực thể phải có khoá chính, các thuộc tính khoá phải có giá trị duy nhất và khác *null*. Qui tắc này không cho phép hai bản ghi trùng khoá.

◊ *Ví dụ*. Xét cơ sở dữ liệu

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)
PROJ(*PNO, PNAME, BUDGET*)

Qui tắc toàn vẹn thực thể ràng buộc:

- Trong lược đồ PROJ thuộc tính khoá *PNO* không thể nhận giá trị *null* và có giá trị không trùng nhau.

- Trong lược đồ EMP cặp thuộc tính khoá (*EMP, PNO*) không thể nhận giá trị *null* và có giá trị không trùng nhau.

3.2. Qui tắc miền giá trị

Đây là loại ràng buộc lên các giá trị hợp lệ của thuộc tính. *Miền giá trị* là tập hợp tất cả các loại dữ liệu và phạm vi giá trị được thuộc tính thừa nhận. Định nghĩa miền giá trị xác định các tham số đặc trưng của thuộc tính: kiểu dữ liệu (data type), độ dài (length), khuôn dạng (format), phạm vi (range), giá trị cho phép (allowable values), ý nghĩa (meaning), tính duy nhất (uniqueness), chấp nhận giá trị *null* (null support).

◊ *Ví dụ*. Xét quan hệ

PROJ(*PNO, PNAME, BUDGET*)

Các thuộc tính *PNAME* và *BUDGET* có ràng buộc miền giá trị như sau

Tên thuộc tính	:	*PNAME*	*BUDGET*
Ý nghĩa	:	Tên dự án	Kinh phí dự án
Kiểu dữ liệu	:	Ký tự (Character)	Số (numeric)
Độ dài	:	20	10
Format	:		9,999,999
Phạm vi	:		> 1000 & <10,000,000
Giá trị cho phép	:		
Duy nhất	:	Có	Không
Null support	:	Non-null	Null

3.3. Qui tắc toàn vẹn tham chiếu

Toàn vẹn tham chiếu là ràng buộc đảm bảo tính hợp lệ của sự tham chiếu của một đối tượng trong cơ sở dữ liệu (gọi là đối tượng tham chiếu) đến đối tượng khác (gọi là đối tượng được tham chiếu) trong cơ sở dữ liệu đó. Các thuộc tính tương ứng gọi *thuộc tính cặp ghép* của ràng buộc tham chiếu.

◊ *Ví dụ*. Xét các quan hệ

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)

PROJ(*PNO, PNAME, BUDGET*)

Với mỗi bộ **e** ∈ EMP phải tồn tại bộ **p**∈ PROJ sao cho

$$\mathbf{e}(PNO) = \mathbf{p}(PNO)$$

Thuộc tính *PNO* của quan hệ EMP là khoá ngoại tham chiếu đến khoá chính *PNO* của quan hệ PROJ: EMP.PNO→PROJ.PNO

Quan hệ EMP là quan hệ tham chiếu và quan hệ PROJ là quan hệ được tham chiếu, với thuộc tính cặp ghép là (*EMP.PNO*, *PROJ.PNO*).

Qui tắc toàn vẹn tham chiếu được xét đến trong khi cập nhật quan hệ tham chiếu hoặc quan hệ được tham chiếu. Ta xét các qui tắc con sau.

· ***Qui tắc chèn***: Không thể chèn hàng mới vào quan hệ tham chiếu nếu quan hệ được tham chiếu chưa có dữ liệu thuộc tính cặp ghép tương ứng.

◊ *Ví dụ*. Xét các quan hệ EMP và PROJ. Giả sử ta muốn chèn bản ghi

e = (E1, J.Doe, Elect.Eng., 40000, P5, Manager, 20)

trong đó P5 là dự án Elect.Commerce (thương mại điện tử) với kinh phí 500000 USD.

Khi đó, nếu quan hệ PROJ chưa có bản ghi

p = (P5, Elect.Commerce, 500000)

thì qui tắc chèn đảm bảo không thể chèn bản ghi **e** vào quan hệ EMP được.

Muốn chèn bản ghi **e** vào quan hệ EMP, trước hết ta phải chèn bản ghi **p** vào quan hệ PROJ.

· ***Qui tắc xoá***: Không thể xoá hàng của quan hệ được tham chiếu nếu hàng đó có dữ liệu thuộc tính cặp ghép tương ứng trong quan hệ tham chiếu.

◊ *Ví dụ*. Xét các quan hệ EMP và PROJ.

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E2	M.Smith	Syst.Anal.	34000	P1	Analyst	24
E2	M.Smith	Syst.Anal.	34000	P2	Analyst	6
...	...	...	...	...	...	...

PROJ

PNO	*PNAME*	*BUDGET*

P1	Instrumentation	150000
P2	Database Development	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Giả sử dự án P1 đã kết thúc và ta muốn xoá nó khỏi quan hệ PROJ. Tuy nhiên các bản ghi một và hai của EMP tham chiếu đến dự án P1, vì thế ta không thể xoá dự án P1 được.

Tuy nhiên, qui tắc toàn vẹn tham chiếu cho phép các phương án lựa chọn sau:

1) *Restrict*: Không cho xoá.
2) *Nullify*: Gán giá trị *null* cho thuộc tính cặp ghép của các bản ghi của quan hệ tham chiếu tham chiếu đến bản ghi bị xoá.
3) *Cascade*: Xoá tất cả các bản ghi của quan hệ tham chiếu tham chiếu đến bản ghi bị xoá.

3.4. Qui tắc thao tác bẫy

Qui tắc thao tác bẫy là qui tắc yêu cầu tính hợp pháp của dữ liệu trong các tác nghiệp cập nhật như xoá, chèn và sửa.

Thao tác bẫy có thể liên quan đến các thuộc tính của một quan hệ hoặc nhiều quan hệ. Các ràng buộc phức tạp thường được phát biểu dạng thao tác bẫy.

Một thao tác bẫy thường có các thành phần sau:

1) *Qui tắc người dùng*: là yêu cầu ngắn gọn của ràng buộc.
2) *Sự kiện*: là các thao tác xử lý dữ liệu (chèn, sửa hoặc xoá) kích hoạt thao tác bẫy.
3) *Tên quan hệ*: tên các quan hệ liên quan.
4) *Điều kiện*: là các lý do dẫn đến việc kích hoạt thao tác bẫy.
5) *Hành động*: là công việc thực thi khi thao tác bẫy được kích hoạt.

◊ *Ví dụ*. Cho quan hệ

NHANVIEN(*Manv, HoTen, NgaySinh, NgayBC, ...*).

Hiển nhiên là *NgayBC* (ngày vào biên chế) không được sớm hơn *NgaySinh*. Ta có thể đảm bảo điều kiện này bằng thao tác bẫy sau:

Qui tắc người dùng: *NgayBC* không sớm hơn *NgaySinh.*
Sự kiện: Chèn, Sửa.
Tên quan hệ: NHANVIEN
Điều kiện: *NgayBC < NgaySinh.*
Hành động: Phủ nhận thao tác cập nhật.

◊ *Ví dụ*. Xét hai quan hệ

KHACH(*Makhach, TenKhach, TaiKhoan, SoDu*)
THANHTOAN(*MaKhach, SoTien*)

Ta thấy rằng *SoTien* của THANHTOAN không thể vượt quá *SoDu* của KHACH. Ta có thể đảm bảo điều kiện này bằng thao tác bẫy sau:

Qui tắc người dùng: *SoTien* không lớn hơn *SoDu*.
Sự kiện: Chèn, Sửa.
Tên quan hệ: THANHTOAN, KHACH
Điều kiện: THANHTOAN.*SoTien* > KHACH.*SoDu*
Hành động: Phủ nhận thao tác cập nhật.

4. Các ngôn ngữ dữ liệu quan hệ

Các ngôn ngữ thao tác dữ liệu được phát triển cho mô hình quan hệ (thường gọi là *ngôn ngữ vấn tin*, query language) được chia làm hai nhóm căn bản: Các ngôn ngữ dựa trên đại số quan hệ (relational algebra) và các ngôn ngữ dựa trên phép tính quan hệ (relational calculus). Khác biệt giữa chúng là cách thức người sử dụng đưa ra câu vấn tin. Đại số quan hệ thuộc loại thủ tục (procedural), trong đó người dùng cần phải đặc tả, nhờ một số toán tử, bằng cách nào đạt được kết quả. Ngược lại phép tính quan hệ thuộc loại phi thủ tục (nonprocedural), người dùng chỉ cần đặc tả các mối liên hệ cần phải đảm bảo trong kết quả. Cả hai ngôn ngữ được Codd đưa ra năm 1970 và ông đã chứng minh rằng chúng tương đương về khả năng diễn tả.

4.1. Đại số quan hệ

Đại số quan hệ có một tập các phép toán trên các quan hệ. Chúng có nguồn gốc từ lý thuyết tập hợp (mỗi quan hệ thực chất là một tập hợp). Mỗi toán tử nhận một hoặc hai quan hệ làm toán hạng và cho ra một quan hệ mới (quan hệ kết quả), đến lượt nó quan hệ kết quả có thể dùng làm toán hạng cho một toán tử khác. Những phép toán này cho phép vấn tin và cập nhật cơ sở dữ liệu quan hệ.

Có năm phép toán đại số cơ bản và năm phép toán khác có thể định nghĩa theo các phép toán cơ bản.

Các phép toán cơ bản là: *phép chọn, phép chiếu, phép hợp, phép hiệu* và *tích Descartes*. Hai phép toán đầu thuộc loại một ngôi, và ba phép toán sau thuộc loại hai ngôi.

Các phép toán bổ sung có thể định nghĩa bởi các phép toán cơ bản là: *phép giao, phép nối, phép nối tự nhiên, phép bán nối*, và *phép thương*.

Trong thực hành đại số quan hệ được mở rộng để có thể nhóm hoặc sắp xếp kết quả và có thể thực hiện các phép gộp phần (aggregation) hoặc các phép tính số học. Một số phép toán khác như nối ngoài (outer join) cũng được hỗ trợ.

Các toán hạng của một số phép toán hai ngôi phải *ứng hợp* (*union compartible*), tức là chúng cùng bậc (cùng số thuộc tính) và các thuộc tính tương ứng có cùng miền giá trị.

a. Phép chiếu

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$. Cho S là lược đồ con của R, $S \subset R$, S có m thuộc tính ($m < n$). Chiếu của **r** lên lược đồ S, ký hiệu $\pi_S(\mathbf{r})$, được định nghĩa là quan hệ với lược đồ S gồm các m-bộ u sao cho tồn tại n-bộ $v \in \mathbf{r}$ thoả mãn

$$v(S) = u$$

tức là

$$\pi_S(\mathbf{r}) = \{\text{m-bộ } u : \exists\, v \in \mathbf{r}, v(S) = u \}$$

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C. Sau đây là ví dụ cụ thể về chiếu của **r** lên hai thuộc tính A và C.

r

A	**B**	**C**
a	b	c
d	a	f
c	b	d
c	f	d

⇒

$\pi_{A,C}(\mathbf{r})$

A	**C**
a	c
d	f
c	d

◊ *Ví dụ*: Xét quan hệ PROJ

PROJ

PNO	*PNAME*	*BUDGET*
P1	Instrumentation	150000
P2	Database Development	135000
P3	CAD/CAM	250000
P4	Maintenance	310000

Chiếu của PROJ lên các thuộc tính PNO và BUDGET có kết quả như sau

$\pi_{PNO,BUDGET}$(PROJ)

PNO	*BUDGET*
P1	150000
P2	135000
P3	250000
P4	310000

b. Phép chọn

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1,..., A_n)$ và F là biểu thức lôgic bao gồm

(i) Các toán hạng là hằng hoặc (số hiệu) thành phần,

(ii) Các phép toán so sánh : <, =, >, ≠ , ≤, ≥

(iii) Các phép toán logic : &, ∧ (và), ∨ (hoặc) , ¬ (phủ định).

Phép chọn của **r** theo biểu thức F, ký hiệu $\sigma_F(\mathbf{r})$, là quan hệ với lược đồ R gồm tất cả các bộ t trong **r** sao cho khi thay các thành phần của t vào biểu thức F thì ta có giá trị đúng. Tức là

$$\sigma_F(\mathbf{r}) = \{ t \in \mathbf{r} : F(t) = true \}$$

◊ *Ví dụ:* Cho quan hệ **r** với các thuộc tính A,B,C. Sau đây là ví dụ cụ thể về chọn của **r** theo biểu thức B=b

r

A	**B**	**C**
a	b	c

⇒

$\sigma_{B=b}(\mathbf{r})$

A	**B**	**C**
a	b	c

d	a	f		c	b	d
c	b	d			
c	f	d			

◊ *Ví dụ*: Xét quan hệ EMP

EMP

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E2	M.Smith	Syst.Anal.	34000	P1	Analyst	24
E2	M.Smith	Syst.Anal.	34000	P2	Analyst	6
E3	A.Lee	Mech.Eng.	27000	P3	Consultant	10
E3	A.Lee	Mech.Eng.	27000	P4	Engineer	48
E4	J.Miller	Programmer	24000	P2	Programmer	18
E5	B.Casey	Syst.Anal.	34000	P2	Manager	24
E6	L.Chu	Elect.Eng.	40000	P4	Manager	48
E7	R.David	Mech.Eng.	27000	P3	Engineer	36
E8	J.Jones	Syst.Anal.	34000	P3	Manager	40

Các bộ của các kỹ sư điện (electrical engineer) được biểu diễn bằng phép chọn như sau

$\sigma_{TITLE = \text{'Elect. Eng.'}}(EMP)$

ENO	*ENAME*	*TITLE*	*SAL*	*PNO*	*RESP*	*DUR*
E1	J.Doe	Elect.Eng.	40000	P1	Manager	12
E6	L.Chu	Elect.Eng.	40000	P4	Manager	48

c. Phép hợp

Cho quan hệ **r**, **s** với lược đồ quan hệ R=(A_1,..., A_n).

Hợp của **r** và **s**, ký hiệu **r** ∪ **s** , là quan hệ với lược đồ R gồm tất cả các bộ thuộc **r** hoặc thuộc **s**.

◊ *Ví dụ*

Cho quan hệ **r** và **s** với các thuộc tính A,B,C . Sau đây là ví dụ cụ thể về hợp của **r** và **s**.

r

A	**B**	**C**
a	b	c
d	a	f
c	b	d

s

A	**B**	**C**
b	g	a
d	a	f

⇒

r ∪ **s**

A	**B**	**C**
a	b	c
d	a	f
c	b	d
b	g	a

d. Phép hiệu

Cho quan hệ **r**, **s** với lược đồ quan hệ R=(A_1,..., A_n).

Hiệu của **r** và **s**, ký hiệu **r** − **s**, là quan hệ với lược đồ R gồm tất cả các bộ thuộc **r** nhưng không thuộc **s**.

◊ *Ví dụ*

Cho quan hệ **r** và **s** với các thuộc tính A,B,C . Sau đây là ví dụ cụ thể về hiệu của **r** và **s**.

r

A	B	C
a	b	c
d	a	f
c	b	d

s

A	B	C
b	g	a
d	a	f

⇒ **r − s**

A	B	C
a	b	c
c	b	d

e. Tích Đề-các

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(B_1,..., B_m)$.

Tích Đề các của **r** và **s**, ký hiệu **r** × **s**, là quan hệ với lược đồ $=(A_1, ..., A_n, B_1,..., B_m)$ gồm tất cả các (n+m)-bộ, trong đó n thành phần đầu là bộ thuộc **r** và m thành phần sau là bộ thuộc **s**.

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính D,E,F. Sau đây là ví dụ cụ thể về tích Đề-các của **r** và **s**.

r

A	B	C
a	b	c
d	a	f
c	b	d

s

D	E	F
b	g	a
d	a	f

⇒ **r × s**

A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

f. Phép giao

Cho quan hệ **r**, **s** với lược đồ quan hệ $R=(A_1,..., A_n)$.

Giao của **r** và **s**, ký hiệu **r** ∩ **s**, là quan hệ với lược đồ R gồm tất cả các bộ thuộc **r** và thuộc **s**.

◆ *Công thức*. Phép giao suy ra từ phép hiệu

$$\mathbf{r} \cap \mathbf{s} = \mathbf{r} - (\mathbf{r} - \mathbf{s}) = \mathbf{s} - (\mathbf{s} - \mathbf{r})$$

◊ *Ví dụ*

Cho quan hệ **r** và **s** với các thuộc tính A,B,C . Sau đây là ví dụ cụ thể về giao của **r** và **s**.

r

A	B	C
a	b	c
d	a	f
c	b	d

s

A	B	C
b	g	a
d	a	f

⇒ **r ∩ s**

A	B	C
d	a	f

g. Phép nối

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(B_1, ..., B_m)$.

Cho biểu thức lôgic F. Phép *nối* của quan hệ **r** và quan hệ **s** theo điều kiện nối F, ký hiệu

$$\mathbf{r} \underset{F}{><} \mathbf{s}$$

là quan hệ với lược đồ $=(A_1, ..., A_n, B_1, ..., B_m)$ gồm tất cả các (n+m)-bộ (t,u) thoả $t \in \mathbf{r}$, $u \in \mathbf{s}$ và khi thế các giá trị của *t* và u vào F thì ta được giá trị đúng.

Ở đây F là biểu thức lôgic gồm các toán hạng dạng $A\theta B$, trong đó A là thuộc tính của **r** và B là thuộc tính **s** và θ là phép toán so sánh.

◆ *Công thức.* Ta có thể biểu diễn

$$\mathbf{r} \underset{F}{><} \mathbf{s} = \sigma_F(\mathbf{r} \times \mathbf{s})$$

◆ *Phép đẳng nối*

Nếu các toán tử so sánh trong biểu thức F đều là phép bằng (=), thì phép nối theo F được gọi là phép *đẳng nối* (*equijoin*).

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính D,E. Sau đây là ví dụ cụ thể về phép nối và đẳng nối của **r** và **s**.

r

A	B	C
1	2	3
4	5	6
7	8	9

s

D	E
3	1
6	2

$\Rightarrow$ $\mathbf{r} \underset{B<D}{><} \mathbf{s}$

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

r

A	B	C
1	2	3
4	5	6
7	8	9

s

D	E
3	1
6	2

$\Rightarrow$ $\mathbf{r} \underset{C=D}{><} \mathbf{s}$

A	B	C	D	E
1	2	3	3	1
4	5	6	6	2

◊ *Ví dụ.* Xét các quan hệ

EMP(*ENO, ENAME, TITLE*)

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

PAY(*TITLE, SAL*)

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000
Operator	15000

với ràng buộc toàn vẹn tham chiếu EMP.*TITLE*→PAY.*TITLE*.

Phép nối EMP với PAY theo EMP.*TITLE*=PAY.*TITLE* có kết quả sau

EMP >< PAY
EMP.*TITLE*=PAY.*TITLE*

ENO	*ENAME*	*EMP.TITLE*	*PAY.TITLE*	*SAL*
E1	J.Doe	Elect.Eng.	Elect.Eng.	40000
E2	M.Smith	Syst.Anal.	Syst.Anal.	34000
E3	A.Lee	Mech.Eng.	Mech.Eng.	27000
E4	J.Miller	Programmer	Programmer	24000
E5	B.Casey	Syst.Anal.	Syst.Anal.	34000
E6	L.Chu	Elect.Eng.	Elect.Eng.	40000
E7	R.David	Mech.Eng.	Mech.Eng.	27000
E8	J.Jones	Syst.Anal.	Syst.Anal.	34000

h. Phép nối tự nhiên

Nối tự nhiên giữa 2 quan hệ **r** và **s**, ký hiệu là **r** >< **s**, là phép đẳng nối trên các thuộc tính cụ thể có cùng miền giá trị. Tuy nhiên khác với đẳng nối là các thuộc tính dùng để nối tự nhiên chỉ xuất hiện một lần trong bảng kết quả.

◆ *Cách tính*.

Cho các quan hệ **r** và **s** với các cột được đặt tên theo thuộc tính. *Nối tự nhiên* **r** >< **s** được tính như sau:

(i) Tính tích Đề-các **r** × **s**.

(ii) Với mỗi thuộc tính A có cả trong **r** và **s**, chọn các bộ trong **r** × **s** có R.A = S.A, trong đó R.A (S.A) là tên cột của R × S tương ứng với cột A của R (S).

(iii) Với mỗi thuộc tính A như trên ta loại bỏ cột S.A.

Một cách hình thức, giả sử A_1, A_2, ... , A_k là tên các thuộc tính dùng chung cho cả R và S. Gọi i_1, i_2, ... , i_m là danh sách các thành phần của R × S , trừ các thuộc tính $S.A_1$, $S.A_2$, ..., $S.A_k$. Khi đó ta có thể biểu diễn nối tự nhiên bằng công thức sau.

◆ *Công thức*

$$\mathbf{r} >< \mathbf{s} = \pi_{i_1, i_2, ..., i_m} \sigma_{R.A_1=S.A_1 \wedge ... \wedge R.A_k=S.A_k} (\mathbf{r} \times \mathbf{s})$$

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính B,C,D. Sau đây là ví dụ cụ thể về phép nối tự nhiên của **r** và **s**.

r			**s**				**r** >< **s**			
A	**B**	**C**	**B**	**C**	**D**	⇒	**A**	**B**	**C**	**D**
a	b	c	b	c	d		a	b	c	d
d	b	c	b	c	e		a	b	c	e
b	b	f	a	d	b		d	b	c	d

c	a	d		d	b	c	e
				c	a	d	b

◊ *Ví dụ.* Các quan hệ

EMP(*ENO, ENAME, TITLE*)

PAY(*TITLE, SAL*)

cho ở ví dụ trước, có thuộc tính chung là *TITLE*.

Nối tự nhiên của hai quan hệ cho ở bảng sau

EMP >< PAY

ENO	*ENAME*	*TITLE*	*SAL*
E1	J.Doe	Elect.Eng.	40000
E2	M.Smith	Syst.Anal.	34000
E3	A.Lee	Mech.Eng.	27000
E4	J.Miller	Programmer	24000
E5	B.Casey	Syst.Anal.	34000
E6	L.Chu	Elect.Eng.	40000
E7	R.David	Mech.Eng.	27000
E8	J.Jones	Syst.Anal.	34000

i. Phép bán nối

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1,..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(B_1, ..., B_m)$.

Ký hiệu θ là toán tử so sánh (<, =, >, ≠ , ≤, ≥). Cho F là biểu thức lôgic gồm các toán hạng dạng AθB, trong đó A là thuộc tính của **r** và B là thuộc tính **s**.

Phép *bán nối* của quan hệ **r** và quan hệ **s** theo điều kiện nối F, ký hiệu

$$\mathbf{r} \underset{F}{><} \mathbf{s}$$

là quan hệ với lược đồ R gồm các bộ của **r** có tham gia vào nối của **r** và **s** theo F**.**

◆ *Công thức.* Ta có thể biểu diễn

$$\mathbf{r} \underset{F}{><} \mathbf{s} = \pi_R(\mathbf{r} \underset{F}{><} \mathbf{s})$$

Ưu điểm của phép bán nối là giảm số lượng các bộ cần xử lý để thực hiện nối. Trong hệ cơ sở dữ liệu phân tán điều này có ý nghĩa rất quan trọng vì nó làm giảm số lượng dữ liệu cần truyền giữa các vị trí để ước lượng câu vấn tin.

◆*Bán nối tự nhiên*

$$\mathbf{r} >< \mathbf{s} = \pi_R(\mathbf{r} >< \mathbf{s})$$

◊ *Ví dụ.* Cho quan hệ **r** với các thuộc tính A,B,C và **s** với các thuộc tính D,E. Sau đây là ví dụ cụ thể về phép bán nối của **r** và **s**.

r			**s**			$\mathbf{r} \underset{B<D}{><} \mathbf{s}$		
A	**B**	**C**	**D**	**E**	⇒	**A**	**B**	**C**
1	2	3	3	1		1	2	3

4	5	6		6	2		4	5	6
7	8	9							

r **s** **r** $\overset{A=E}{><}$ **s**

A	**B**	**C**		**D**	**E**	⇒	**A**	**B**	**C**
1	2	3		3	1		1	2	3
4	5	6		6	2				
7	8	9							

◊ *Ví dụ*. Xét các quan hệ

EMP(*ENO, ENAME, TITLE*)

PAY(*TITLE, SAL*)

ở ví dụ trước. Phép bán nối của hai quan hệ theo EMP.*TITLE*=PAY.*TITLE* có kết quả sau

EMP >< PAY

EMP.*TITLE*=PAY.*TITLE*

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

PAY >< EMP

EMP.*TITLE*=PAY.*TITLE*

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000

j. Phép chia (thương)

Cho quan hệ **r** với lược đồ quan hệ $R=(A_1, ..., A_n)$ và quan hệ **s** với lược đồ quan hệ $S=(A_1, ..., A_m)$, trong đó $m < n$.

Ta định nghĩa *phép chia* **r** ÷ **s** là quan hệ với lược đồ $(A_{m+1}, ..., A_n)$ gồm tất cả (n-m)-bộ *v* sao cho với mọi m-bộ *u* thuộc **s**, bộ (*u*,*v*) thuộc **r**.

Để lập công thức cho phép chia ta ký hiệu

$$\mathbf{t} = \pi_{A_{m+1},...,A_n}(r)$$

Khi đó

$$(\mathbf{s} \times \mathbf{t}) - \mathbf{r}$$

là tập hợp các n-bộ không thuộc **r**, tạo ra bằng cách lấy các phần tử của **s** kết hợp với n-m thành phần của các phần tử thuộc **r**. Đặt

$$\mathbf{q} = \pi_{A_{m+1},\ldots,A_n}\left((s \times t) - r\right)$$

Như vậy **q** là tập tất cả các (n−m)-bộ *v* gồm n−m thành phần cuối của các phần tử của **r** và tồn tại phần tử $u \in \mathbf{s}$ sao cho bộ (*u*,*v*) không thuộc **r**. Suy ra

◆ *Công thức*

$$\mathbf{r} \div \mathbf{s} = \mathbf{t} - \pi_{A_{m+1},\ldots,A_n}\left((s \times t) - r\right)$$

◊ *Ví dụ*

Cho quan hệ **r** với các thuộc tính A,B,C,D và **s** với các thuộc tính C,D. Sau đây là ví dụ cụ thể về phép chia của **r** cho **s**.

r				**s**			**r ÷ s**	
A	**B**	**C**	**D**	**C**	**D**	⇒	**A**	**B**
a	b	c	d	c	d		a	b
a	b	e	f	e	f		e	d
b	c	e	f					
e	d	c	d					
e	d	e	f					
a	b	d	e					

◊ *Ví dụ*. Xét các quan hệ sau:

ASG'

ENO	*PNO*	*PNAME*	*BUDGET*
E1	P1	Instrumentation	150000
E2	P1	Instrumentation	150000
E2	P2	Database Deve.	135000
E3	P3	CAD/CAM	250000
E3	P4	Maintenance	310000
E4	P2	Database Deve.	135000
E5	P2	Database Deve.	135000
E6	P4	Maintenance	310000
E7	P3	CAD/CAM	250000
E8	P3	CAD/CAM	250000

PROJ'

PNO	*PNAME*	*BUDGET*
P3	CAD/CAM	250000
P4	Maintenance	310000

Để tìm mã số nhân viên của những nhân viên được phân vào tất cả dự án trong PROJ', ta phải thực hiện phép chia ASG' cho PROJ' và được kết quả là

ASG' ÷ PROJ'

ENO
E3

· ***Các chương trình đại số quan hệ***

Vì tất cả các phép toán đại số đều nhận quan hệ làm đối biến và sinh ra các quan hệ kết quả, chúng ta có thể lồng ghép các phép toán này bằng các dấu ngoặc đơn và sinh ra các *chương trình đại số quan hệ*.

Dưới đây là một số ví dụ minh hoạ sử dụng các quan hệ

EMP(*ENO, ENAME, TITLE*)
PAY(*TITLE, SAL*)
PROJ(*PNO, PNAME, BUDGET*)
ASG(*ENO, PNO, RESP, DUR*)

◊ *Ví dụ*. Sau đây là chương trình tìm tên tất cả nhân viên đang làm việc cho dự án CAD/CAM

$$\pi_{ENAME}(((\sigma_{PNAME='CAD/CAM'}(PROJ)) >< ASG) >< EMP)$$

Thứ tự thực hiện như sau: thực hiện phép chọn trên PROJ, sau đó nối tự nhiên với ASG, theo sau là nối tự nhiên với EMP, và cuối cùng là chiếu lên ENAME.

Một chương trình tương đương nhưng kích thước quan hệ trung gian nhỏ hơn là

$$\pi_{ENAME}(EMP >< (\pi_{ENO}(ASG >< (\sigma_{PNAME='CAD/CAM'}(PROJ))))$$

◊ *Ví dụ*. Sau đây là chương trình tăng lương tất cả các lập trình viên (programmer) lên 25000 USD.

$$(PAY - (\sigma_{TITLE='Programmer'}(PAY)) \cup (<'Programmer',25000>)$$

4.2. Đại số quan hệ

Trong các ngôn ngữ dựa trên phép tính quan hệ, thay vì xác định xem phải làm thế nào để thu được kết quả, chúng ta sẽ xác định xem kết quả là gì bằng cách đưa ra mối liên hệ được giả sử là đúng đối với kết quả.

Ngôn ngữ phép tính quan hệ được phân làm 2 nhóm: *phép tính quan hệ bộ (tuple relational calculus)* và *phép tính quan hệ miền (domain relational calculus)*. Sự khác biệt giữa chúng là ở các biến nguyên thuỷ được dùng khi xác định các câu vấn tin.

Ngôn ngữ phép tính quan hệ có cơ sở lý thuyết vững chắc bởi vì chúng xây dựng trên logic vị từ bậc nhất. Ngữ nghĩa được gán cho các công thức bằng cách diễn giải chúng như các phán đoán trên cơ sở dữ liệu. Một cơ sở dữ liệu quan hệ có thể xem như tập các bộ hoặc tập các miền. Phép tính quan hệ bộ diễn giải các biến trong công thức như một bộ của quan hệ, còn phép tính quan hệ miền diễn giải biến như giá trị của miền.

a) Phép tính quan hệ bộ (Codd 1970)

Biến nguyên thuỷ dùng trong phép tính quan hệ bộ là *biến bộ* (tuple variable), biểu thị một bộ của quan hệ. Nói cách khác biến này biến thiên trên các bộ của quan hệ.

Trong phép tính quan hệ bộ, câu vấn tin được đặc tả là

$$\{t \mid F(t)\}$$

trong đó t là biến bộ và F là công thức chỉnh dạng. Công thức nguyên tử có hai dạng:

(1) *Biểu thức kiểu phần tử biến bộ.*

Nếu t là một biến bộ biến thiên trên các bộ của một quan hệ R, biểu thức "*bộ t thuộc quan hệ R*" là công thức nguyên tử, và được viết là $R.t$ hoặc $R(t)$.

(2) *Điều kiện.*

Loại công thức nguyên tử này có thể định nghĩa như sau:

(i) $s[A]\ \theta\ t[B]$, trong đó s và t là các biến bộ và A và B là các thành phần tương ứng của s và t , θ là một trong các toán tử so sánh $<, >, =, \leq, \geq$ và $\neq$.

(ii) $s[A]\ \theta\ c$, trong đó s , A và θ định nghĩa giống như trên và c là hằng.

Hiện có nhiều ngôn ngữ dựa trên phép tính quan hệ bộ, và ngôn ngữ thông dụng nhất là SQL và QUEL. SQL hiện là chuẩn quốc tế (duy nhất) với các phiên bản chuẩn hoá đã được đưa ra năm 1986 (SQL1), năm 1992 (SQL2) và năm 1998 (SQL3).

· ***Ngôn ngữ truy vấn SQL***

SQL thuộc loại *ngôn ngữ thế hệ thứ tư (4GL)* được nghiên cứu nhiều năm và trở thành tiêu chuẩn quốc tế về kiểm soát dữ liệu. SQL kế thừa *tính phi thủ tục* của 4GL: Xử lý đồng thời hàng loạt câu lệnh. Người dùng chỉ cần nêu ra yêu cầu về dữ liệu mà không cần biết máy tính xử lý bên trong như thế nào. Người dùng có thể truy xuất nhanh chóng với những CSDL lớn, yêu cầu những xử lý phức tạp tinh vi mà không cần lập trình.

SQL là *ngôn ngữ có cấu trúc*. Trong câu lệnh của SQL có một số mệnh đề tuân theo những cú pháp riêng của nó. Có 4 loại lệnh trong SQL :

- Các lệnh truy vấn dữ liệu.
- Các lệnh định nghĩa dữ liệu (DDL).
- Các lệnh xử lý cập nhật dữ liệu (DML).
- Các lệnh kiểm soát dữ liệu.

◆ Các lệnh truy vấn dữ liệu, gọi là câu vấn tin có cú pháp tổng quát như sau

```
SELECT [DISTINCT] <biểu thức 1> AS <tên 1>] [,...] | *
FROM  <bảng 1> [<bí danh 1>] [,...]
[INTO <dbf đích>]
[WHERE  <điều kiện nối >  [AND | OR <điều kiện lọc>]]
[GROUP BY  <cột nhóm 1> [,...]
   [HAVING <điều kiện nhóm>]]
[ORDER BY <biểu thức sắp xếp 1> [ASC | DESC] [,...]]
[UNION | INTERSECT | MINUS  < câu truy vấn khác>]
```

Dưới đây là một số ví dụ minh họa sử dụng các quan hệ

EMP(*ENO, ENAME, TITLE*)
PAY(*TITLE, SAL*)
PROJ(*PNO, PNAME, BUDGET*)
ASG(*ENO, PNO, RESP, DUR*)

◊ *Ví dụ*. Tìm tên tất cả nhân viên đang làm việc cho dự án CAD/CAM

```
Select   EMP.ENAME
From     EMP, ASG, PROJ
Where   (EMP.ENO = ASG.ENO)
         AND (ASG.PNO = PROJ.PNO)
         AND (PROJ.PNAME = "CAD/CAM")
```

◊ *Ví dụ*. Tìm tên tất cả nhân viên đang quản lý dự án (Manager)

```
SELECT      ENAME
FROM        EMP, ASG
WHERE       (EMP.ENO = ASG.ENO)
        AND    (RESP = "Manager")
```

◊ *Ví dụ*. Tìm tên tất cả nhân viên đang làm việc trong dự án P3 và P4. (bài tập)

◆ Các lệnh cập nhật dữ liệu gồm có lệnh UPDATE (hiệu chỉnh), INSERT (thêm) và DELETE (xoá).

◊ *Ví dụ.* Tăng lương các lập trình viên (programmer) lên 25000 USD.

```
UPDATE  PAY
SET     SAL = 25000
WHERE   PAY.TITLE = "Programmer"
```

◊ *Ví dụ.* Thêm nhân viên mới vào EMP

```
INSERT INTO EMP
   VALUE ('E10', 'John Smith', 'Programmer')
```

◊ *Ví dụ.* Xoá dự án 'P1'

```
DELETE FROM PROJ
   WHERE  PNO = 'P1'
```

b) Phép tính quan hệ miền (Lacroix, Pirotte 1977)

Biến nguyên thuỷ dùng trong phép tính quan hệ miền là *biến miền* (*domain variable*), xác định một thành phần của bộ biến thiên trong tập giá trị của miền. Nói cách khác, miền xác định của biến miền bao gồm các miền trên đó quan hệ được định nghĩa. Câu vấn tin có dạng sau:

$$x_1, \ldots, x_n \mid F(x_1, \ldots, x_n)$$

trong đó F là công thức chỉnh dạng còn $x_1, \ldots, x_n$ là các biến tự do.

Thành công của ngôn ngữ phép tính quan hệ miền chủ yếu do QBE (Zloof, 1977) đem lại. Đây là ứng dụng kiểu trực quan của phép tính miền. QBE (Query by example) được thiết kế dành cho kiểu làm việc tương tác từ thiết bị đầu cuối trực quan và thân thiện.

Khái niệm cơ bản là *example*: người sử dụng đưa ra các câu vấn tin bằng cách cung cấp một example có thể có của câu trả lời. Hành động gõ tên quan hệ sẽ kích hoạt việc hiển thị các lược đồ của chúng lên màn hình. Sau đó bằng cách cung cấp các từ khoá trong các cột (miền), người dùng đặc tả câu vấn tin.

Chẳng hạn các thuộc tính của quan hệ chiếu được cho bằng từ P (Project).

Theo mặc định tất cả các câu vấn tin đều là kiểu truy xuất. Câu vấn tin cập nhật đòi hỏi phải có đặc tả U dưới tên quan hệ cần cập nhật.

◊ *Ví dụ.* Tìm tên tất cả nhân viên đang làm việc cho dự án CAD/CAM

EMP	*ENO*	*ENAME*	*TITLE*
	E2	P	

ASG	*ENO*	*PNO*	*RESP*	*DUR*
	E2	P3		

PROJ	*PNO*	*PNAME*	*BUDGET*

P3	CAD/CAM	

◊ *Ví dụ.* Tăng lương các lập trình viên (programmer) lên 25000 USD.

PAY	TITLE	SAL
	Programmer	U.25000

5. Thiết kế cơ sở dữ liệu quan hệ

5.1. Dư thừa dữ liệu

Khi thiết kế cơ sở dữ liệu quan hệ ta thường đứng trước vấn đề lựa chọn giữa các lược đồ quan hệ: lược đồ nào tốt hơn ? Tại sao ? Mục này sẽ nghiên cứu một số tiêu chuẩn đánh giá lược đồ quan hệ và các thuật toán giúp chúng ta xây dựng được lược đồ cơ sở dữ liệu quan hệ có cấu trúc tốt.

Có thể nói tổng quát một lược đồ quan hệ có *cấu trúc tốt* là lược đồ không chứa đựng sự *dư thừa dữ liệu*, tức là là sự trùng lặp thông tin trong cơ sở dữ liệu.

5.1.1. Sự dư thừa dữ liệu

Dư thừa dữ liệu là sự trùng lặp thông tin trong cơ sở dữ liệu.

◊ *Ví dụ*

Xét quan hệ EMP(*ENO*, *ENAME, TITLE, SAL*, *PNO*, *RESP*, *DUR*). Nếu một nhân viên tham gia trong nhiều dự án, thì các dữ liệu như *ENAME, TITLE, SAL* phải lặp lại nhiều lần và kéo theo dư thừa dữ liệu.

Ngoài việc gây lãng phí dung lượng lưu trữ, sự dư thừa dữ liệu có thể gây ra những hậu quả nghiêm trọng đối với dữ liệu khi người dùng cập nhật dữ liệu làm cho dữ liệu không tương thích, bất định hoặc mất mát. Các sự cố như vậy gọi là những *dị thường*.

5.1.2. Các dị thường cập nhật dữ liệu

Ta sẽ minh hoạ các dị thường bằng các lược đồ

EMP(*ENO*, *ENAME, TITLE, SAL*, *PNO*, *RESP, DUR*)
PROJ(*PNO*, *PNAME, BUDGET*)

a. Dị thường do dữ liệu lặp: Một số thông tin có thể được lặp lại một cách vô ích.

◊ *Ví dụ*: Trong quan hệ EMP tên (*ENAME*), chức vụ (*TITLE*), và lương (*SAL*) của nhân viên được lặp lại trong mỗi dự án mà họ tham gia. Điều này rõ ràng là làm lãng phí chỗ lưu trữ và đối nghịch với các nguyên lý của cơ sở dữ liệu.

b. Dị thường chèn bộ: Không thể chèn bộ mới vào quan hệ, nếu không có đầy đủ dữ liệu.

◊ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên mới được nhận vào công ty và chưa được phân công vào dự án nào cả. Khi đó chúng ta không thể nhập các thông tin về tên, chức vụ, lương của nhân viên này vào quan hệ, vì khoá của EMP là (*ENO, PNO*).

c. Dị thường xoá bộ: Trường hợp này ngược với dị thường chèn bộ. Việc xoá bộ có thể kéo theo mất thông tin.

◊ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên làm việc trong một dự án duy nhất. Khi dự án chấm dứt, chúng ta không thể xoá thông tin về dự án đó trong EMP được, vì nếu làm thế ta sẽ mất luôn thông tin về nhân viên đó.

d. Dị thường sửa bộ: Việc sửa đổi dữ liệu dư thừa có thể dẫn đến sự không tương thích dữ liệu.

◊ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên làm việc trong nhiều dự án. Khi có sự thay đổi về lương, rất nhiều bộ phải cập nhật sự thay đổi này. Điều đó gây lãng phí thời gian công sức và là nguy cơ gây ra sự không thống nhất dữ liệu.

Trong các ví dụ trên ta thấy tác hại của sự dư thừa dữ liệu và sự cần thiết phải loại bỏ chúng khỏi các lược đồ quan hệ. Quá trình từng bước thay thế một lược đồ quan hệ bằng các tập lược đồ quan hệ đơn giản và chuẩn tắc hơn gọi là ***chuẩn hoá***. Mục đích của chuẩn hoá là loại bỏ các dị thường (hoặc các khía cạnh không mong muốn khác) để có những quan hệ tốt hơn.

Cơ sở lý thuyết của việc thiết kế lược đồ cơ sở dữ liệu quan hệ tốt là khái niệm *phụ thuộc dữ liệu*. Phụ thuộc dữ liệu biểu diễn các quan hệ nhân quả giữa các thuộc tính trong quan hệ. Ví dụ trong bảng EMP, thuộc tính SAL phụ thuộc vào thuộc tính ENO, vì mỗi nhân viên chỉ có một lương duy nhất.

Cũng dựa trên khái niệm phụ thuộc dữ liệu người ta định nghĩa các *dạng chuẩn* của lược đồ dữ liệu quan hệ. Mỗi dạng chuẩn đáp ứng một yêu cầu nhất định đối với lược đồ quan hệ.

Quá trình biến đổi một lược đồ thành lược đồ *tương đương* (bảo toàn thông tin và phụ thuộc dữ liệu) thoả mãn dạng chuẩn gọi là quá trình *chuẩn hoá lược đồ quan hệ*.

Khái niệm phụ thuộc dữ liệu sẽ được nghiên cứu chi tiết ở phần sau.

5.2. Cấu trúc phụ thuộc dữ liệu

Có ba dạng phụ thuộc dữ liệu, *phụ thuộc hàm* (*functional dependancy*- FD) , *phụ thuộc đa trị* (*multivalued dependancy* - MVD) và *phụ thuộc chiếu nối* (*projection-join dependancy* - PJD)

a. Phụ thuộc hàm

Cho lược đồ quan hệ R=(A_1, A_2, ..., A_n) và X, Y là các tập con của {A_1, A_2, ..., A_n}. Ta nói rằng X *xác định hàm* Y hay Y *phụ thuộc hàm* X, ký hiệu $X \rightarrow Y$, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

$$\forall u, v \in \mathbf{r} : u(X) = v(X) \Rightarrow u(Y) = v(Y)$$

Cần nhấn mạnh rằng tính chất phụ thuộc hàm phải thoả với mọi quan hệ **r** của lược đồ R. Ta không thể chỉ xét một quan hệ đặc biệt (quan hệ rỗng chẳng hạn) rồi quy nạp cho toàn lược đồ. Nhưng ta có thể phủ nhận phụ thuộc hàm qua một quan hệ cụ thể nào đó.

Phụ thuộc hàm $X \rightarrow Y$ gọi là phụ thuộc hàm *tầm thường* nếu $Y \subset X$ (hiển nhiên là nếu $Y \subset X$ thì theo định nghĩa ta có $X \rightarrow Y$).

Phụ thuộc hàm $X \rightarrow Y$ gọi là phụ thuộc hàm *nguyên tố* nếu không có tập con thực sự $Z \subset X$ thoả $Z \rightarrow Y$.

Tập thuộc tính $K \subset R$ gọi là *khoá* nếu nó xác định hàm tất cả các thuộc tính và $K \rightarrow R$ là phụ thuộc hàm nguyên tố.

◊ *Ví dụ*: Xét quan hệ PROJ. Ta có thể chấp nhận rằng mỗi dự án có tên và kinh phí xác định. Vậy có thể khẳng định

$$PNO \rightarrow (PNAME, BUDGET)$$

Trong quan hệ EMP ta có

$$(ENO, PNO) \rightarrow (ENAME, TITLE, SAL, RESP, DUR)$$
$$ENO \rightarrow (ENAME, TITLE, SAL)$$

Hoàn toàn hợp lý khi chúng ta khẳng định rằng lương của mỗi chức vụ là cố định, do đó sẽ tồn tại phụ thuộc hàm

$$TITLE \rightarrow SAL$$

b. Phụ thuộc đa trị

Cho lược đồ quan hệ $R=(A_1, A_2, ..., A_n)$ và X, Y là các tập con của $\{A_1, A_2, ..., A_n\}$. Ta nói rằng X *xác định đa trị* Y hay Y *phụ thuộc đa trị* vào X, ký hiệu $X \rightarrow\rightarrow Y$, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

Ứng với mỗi giá trị của miền giá trị các thuộc tính trong X, có một tập giá trị các thuộc tính trong Y liên quan và tập này độc lập với các thuộc tính trong $Z=R\backslash(X \cup Y)$, tức là:

$$\forall x \in D(X)\ \forall y, y' \in D(Y)\ \forall z, z' \in D(Z): (x,y,z), (x,y',z') \in \mathbf{r} \Rightarrow (x,y,z'), (x,y',z) \in \mathbf{r}$$

với D(X), D(Y) và D(Z) là miền giá trị của X, Y và Z.

· Chú ý rằng *phụ thuộc hàm là trường hợp riêng của phụ thuộc đa trị*, tức là

$$X \rightarrow Y \quad \Rightarrow \quad X \rightarrow\rightarrow Y$$

Thật vậy, nếu

$$(x,y,z), (x, y', z') \in \mathbf{r} \text{ và } X \rightarrow Y$$

thì y=y', và kéo theo

$$(x, y, z'), (x, y', z) \in \mathbf{r}$$

◊ *Ví dụ*

Trở lại ví dụ đang xét. Giả sử ta muốn duy trì thông tin về tập nhân viên và về tập dự án có liên quan đến công ty cũng như về chi nhánh (*PLACE*) thực hiện dự án. Yêu cầu này có thể được thực hiện bằng cách định nghĩa quan hệ

SKILL**(***ENO, PNO, PLACE***)**

Ta giả sử (có thể không thực tế cho lắm) (1) mỗi nhân viên đều có thể làm việc cho mọi dự án, (2) mỗi nhân viên đều có thể làm việc tại mọi chi nhánh và (3) mỗi dự án đều có thể được thực hiện tại bất kỳ chi nhánh nào. Một quan hệ mẫu thoả các điều kiện này cho ở bảng sau:

SKILL

ENO	*PNO*	*PLACE*
E1	P1	Toronto
E1	P1	New York
E1	P1	London
E1	P2	Toronto
E1	P2	New York
E1	P2	London
E2	P1	Toronto
E2	P1	New York
E2	P1	London
E2	P2	Toronto
E2	P2	New York
E2	P2	London

Chú ý rằng không có phụ thuộc hàm không tầm thường nào trong quan hệ SKILL; tất cả thuộc tính là thuộc tính khoá. Quan hệ SKILL có hai phụ thuộc đa trị

$$ENO \rightarrow\rightarrow PNO$$
$$ENO \rightarrow\rightarrow PLACE$$

c. Phụ thuộc chiếu-nối

Cho lược đồ quan hệ $R=(A_1, A_2, ..., A_n)$ và $R_1, R_2, ..., R_k$ là các tập con của $\{A_1, A_2, ..., A_n\}$. Ta nói rằng $\{R_1, R_2, ..., R_k\}$ xác định một *phụ thuộc chiếu-nối* của R, nếu mọi quan hệ **r** của R là nối tự nhiên của các chiếu của nó lên $R_1, R_2, ..., R_k$, tức là

$$\mathbf{r} = \pi_{R1}(\mathbf{r}) >< \pi_{R2}(\mathbf{r}) >< \ ... >< \pi_{Rk}(\mathbf{r})$$

· Chú ý rằng *phụ thuộc đa trị là trường hợp riêng của phụ thuộc chiếu-nối*, tức là

$$X \rightarrow\rightarrow Y \quad \Rightarrow \quad \{X \cup Y, X \cup Z\}$$ xác định một phụ thuộc chiếu-nối,

trong đó $Z=R\backslash(X \cup Y)$.

Thật vậy, cho quan hệ **r** trên lược đồ R thỏa phụ thuộc đa trị X→→Y. Hiển nhiên $\pi_{XY}(\mathbf{r}) >< \pi_{XZ}(\mathbf{r}) \supset \mathbf{r}$. Ta chỉ cần chứng minh $\pi_{XY}(\mathbf{r}) >< \pi_{XZ}(\mathbf{r}) \subset \mathbf{r}$. Cho $(x,y) \in \pi_{XY}(\mathbf{r})$ và $(x,z) \in \pi_{XZ}(\mathbf{r})$. Khi đó tồn tại z' và y' thỏa $(x,y,z') \in \mathbf{r}$ và $(x,y',z) \in \mathbf{r}$ (theo định nghĩa phép chiếu), kéo theo $(x,y,z) \in \mathbf{r}$ (vì X→→Y). Từ đó suy ra $\pi_{XY}(\mathbf{r}) >< \pi_{XZ}(\mathbf{r}) \subset \mathbf{r}$.

◊ *Ví dụ*

Xét quan hệ SKILL(*ENO, PNO, PLACE*) ở trên. Do *ENO*→→*PNO*, nên {(*ENO,PNO*), (*ENO,PLACE*)} xác định phụ thuộc chiếu nối.
Ta có

$\pi_{ENO,PNO}$(SKILL)

ENO	*PNO*
E1	P1
E1	P2
E2	P1
E2	P2

$\pi_{ENO,PLACE}$(SKILL)

ENO	*PLACE*
E1	Toronto
E1	New York
E1	London
E2	Toronto
E2	New York
E2	London

Suy ra

$$\text{SKILL} = \pi_{ENO,PNO}(\text{SKILL}) >< \pi_{ENO,PLACE}(\text{SKILL})$$

5.3. Phụ thuộc đa trị

Trong phần này chúng ta sẽ nghiên cứu sâu hơn về phụ thuộc đa trị, mối quan hệ giữa phụ thuộc đa trị và phụ thuộc hàm.

5.3.1. Định nghĩa

Ta nhắc lại định nghĩa phụ thuộc đa trị.

Cho lược đồ quan hệ $R(A_1, A_2, ..., A_n)$ và X, Y là các tập con của $\{A_1, A_2, ..., A_n\}$. Ta nói rằng X *xác định đa trị* Y hay Y *phụ thuộc đa trị* vào X, ký hiệu X→→Y, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

Ứng với mỗi giá trị của miền giá trị các thuộc tính trong X, có một tập giá trị các thuộc tính trong Y liên quan và tập này độc lập với các thuộc tính trong Z=R\(X∪Y), tức là:

$$\forall x \in D(X) \forall y,y' \in D(Y)\ \forall z, z' \in D(Z): (x,y,z), (x,y',z') \in \mathbf{r} \Rightarrow (x,y,z'), (x,y',z) \in \mathbf{r}$$

◊ *Ví dụ*

Xét lược đồ CTHRSG=(*C,T,H,R,S,G*), trong đó C (Course) là môn học, T (Teacher) là giáo viên, H (Hour) là tiết học, R (Room) là phòng học, S (Student) là sinh viên và G (Grade) là điểm số. Một quan hệ mẫu cho ở bảng sau

C	*T*	*H*	*R*	*S*	*G*
CS101	Trần	M9	222	Hùng	9
CS101	Trần	W9	333	Hùng	9
CS101	Trần	F9	222	Hùng	9
CS101	Trần	M9	222	Dũng	7
CS101	Trần	W9	333	Dũng	7
CS101	Trần	F9	222	Dũng	7

Trong ví dụ đơn giản này ta thấy *môn học* có nhiều giờ học, trong các phòng học khác nhau, trong tuần. Mỗi *sinh viên* có một bản ghi cho mỗi giờ học và điểm của sinh viên cũng được lặp lại tương ứng.

Như vậy ta suy ra phụ thuộc đa trị $C \rightarrow\rightarrow H,R$, tức là sẽ có tập *giờ-phòng* ứng với mỗi môn học, độc lập với các thuộc tính khác. Ví dụ, cho hai bản ghi

t = (CS101, Trần, M9, 222, Hùng, 9)
s = (CS101, Trần, W9, 333, Dũng, 7)

chúng ta chờ đợi rằng có thể hoán chuyển (M9, 222) của *t* với (W9, 333) của *s* để nhận được các bộ sau

u = (CS101, Trần, M9, 222, Dũng, 7)
v = (CS101, Trần, W9, 333, Hùng, 9)

Và ta thấy *u, v* cũng có mặt trong quan hệ trên.

Cần nhấn mạnh rằng, $C \rightarrow\rightarrow H,R$ đúng bởi vì với mỗi môn học *c*, nếu tồn tại các bộ

(c, h1, r1, t1, s1, g1)

và

(c, h2, r2, t2, s2, g2)

thì cũng sẽ tồn tại

(c, h1, r1, t2, s2, g2) và (c, h2, r2, t1, s1, g1).

Lưu ý rằng $C \rightarrow\rightarrow H$ và $C \rightarrow\rightarrow R$ không đúng, bởi vì, nếu ngược lại, từ *t* và *s* suy ra bản ghi

(CS101, Trần, M9, 333, Dũng, 7)

phải có trong quan hệ trên.

Tồn tại nhiều phụ thuộc đa trị khác như $C \rightarrow\rightarrow S,G$ và $H,R \rightarrow\rightarrow S,G$ (bài tập).

5.3.2. Các tiên đề phụ thuộc đa trị và phụ thuộc hàm

Ta sẽ trình bày tập hợp đầy đủ các tiên đề phụ thuộc hàm và phụ thuộc đa trị trên tập thuộc tính U. Các tiên đề Armstrong được nhắc lại vì tính hệ thống.

(A1) ***Quy tắc phản xạ phụ thuộc hàm***:

$$Y \subset X \subset U \Rightarrow X \rightarrow Y$$

(A2) ***Quy tắc tăng trưởng phụ thuộc hàm*:**

$$X \rightarrow Y \ \& \ Z \subset U \Rightarrow X \cup Z \rightarrow Y \cup Z$$

(A3) ***Quy tắc bắc cầu phụ thuộc hàm***:

$$X \rightarrow Y \ \& \ Y \rightarrow Z \Rightarrow X \rightarrow Z$$

(M1) ***Quy tắc bù phụ thuộc đa trị***:

$$X \rightarrow\rightarrow Y \Rightarrow X \rightarrow\rightarrow (U \setminus (X \cup Y))$$

(M2) ***Quy tắc tăng trưởng phụ thuộc đa trị***:

$$X \rightarrow\rightarrow Y \ \& \ V \subset W \Rightarrow X \cup W \rightarrow\rightarrow Y \cup V$$

(M3) ***Quy tắc bắc cầu phụ thuộc đa trị***:

$$X \rightarrow\rightarrow Y \ \& \ Y \rightarrow\rightarrow Z \Rightarrow X \rightarrow\rightarrow (Z \setminus Y)$$

(M4) ***Quy tắc phụ thuộc hàm-đa trị***:

$$X \rightarrow Y \Rightarrow X \rightarrow\rightarrow Y$$

(M5) ***Quy tắc phụ thuộc đa trị-hàm***:

$$X \rightarrow\rightarrow Y \ \& \ Z \subset Y \ \& \ W \rightarrow Z \ \& \ W \cap Y = \varnothing \Rightarrow X \rightarrow Z$$

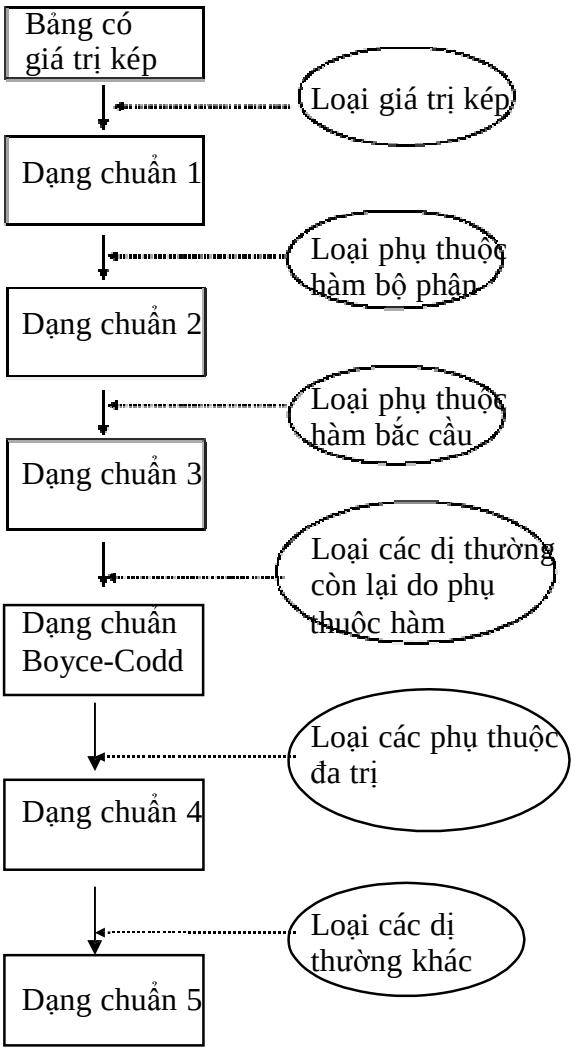
· ***Định lý*:** Các tiên đề A1-A3 và M1-M5 là đúng và đủ cho phụ thuộc hàm và phụ thuộc đa trị. Tức là, nếu D là tập hợp các phụ thuộc hàm và phụ thuộc đa trị trên tập thuộc tính U, và D^+ là tập hợp các phụ thuộc hàm và phụ thuộc đa trị suy diễn lôgic từ D (theo nghĩa mỗi quan hệ thoả D thì cũng thoả D^+), thì D^+ chính là tập hợp các phụ thuộc hàm và phụ thuộc đa trị suy ra từ D bằng các tiên đề trên.

Chứng minh. (công nhận)

5.4. Chuẩn hoá lược đồ quan hệ

Chúng ta đã chỉ ra rằng sự dư thừa dữ liệu là nguyên nhân của các dị thường khi cập nhật dữ liệu dẫn đến sự không tương thích dữ liệu và các hậu quả nghiêm trọng khác. Một lược đồ cơ sở dữ liệu được cho là *tốt* là phải loại bỏ được sự dư thừa dữ liệu. Tuy nhiên ta cần đưa ra định nghĩa chính xác thế nào là lược đồ cơ sở dữ liệu tốt cùng với quá trình thiết kế chúng. Quá trình biến đổi một lược đồ cơ sở dữ liệu thành *lược đồ tương đương*, tức phải bảo toàn thông tin và bảo toàn phụ thuộc dữ liệu, thoả mãn những tiêu chuẩn nhất định gọi là quá trình *chuẩn hoá lược đồ quan hệ*.

Chuẩn hoá lược đồ quan hệ thường được thực hiện qua các giai đoạn tương ứng với các dạng chuẩn (xem sơ đồ dưới). *Dạng chuẩn* là trạng thái quan hệ được xác định bằng cách áp dụng các quy tắc đối với phụ thuộc hàm của quan hệ.



5.4.1. Dạng chuẩn thứ nhất (1NF)

Quan hệ gọi là ở *dạng chuẩn thứ nhất* hay *quan hệ chuẩn hoá* nếu miền giá trị của mỗi thuộc tính chỉ chứa những giá trị *nguyên tử*, tức là không phân chia được nữa. Như vậy mỗi giá trị trong quan hệ cũng là nguyên tử.

Dạng chuẩn 1 chỉ có ý nghĩa ở mức thể hiện của lược đồ quan hệ, vì chỉ liên quan đến giá trị các thuộc tính của các bộ trong một quan hệ được định nghĩa trên lược đồ quan hệ đó.

5.4.2. Dạng chuẩn thứ 2 (2NF)

Thuộc tính A gọi là *phụ thuộc đầy đủ* vào tập thuộc tính X, nếu $X \rightarrow A$ là phụ thuộc hàm nguyên tố.

Giả sử K là khoá của lược đồ R. Khi đó mọi thuộc tính không khoá A của R đều phụ thuộc hàm vào khoá K: $K \rightarrow A$. Nếu A không phụ thuộc đầy đủ vào K thì tồn tại tập con thực sự H của K xác định hàm A, tức $H \rightarrow A$. Khi đó phụ thuộc hàm $H \rightarrow A$ gọi là *phụ thuộc hàm bộ phận*.

Một lược đồ quan hệ gọi là *ở dạng chuẩn thứ 2* nếu nó ở dạng chuẩn thứ 1 và không có phụ thuộc hàm bộ phận, tức là mọi thuộc tính không khoá đều phụ thuộc đầy đủ vào các khoá của lược đồ.

◊ *Ví dụ*

- Xét các quan hệ sau:

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)
PROJ(*PNO, PNAME, BUDGET*)

Lược đồ của EMP có khoá là (*ENO, PNO*).

Phụ thuộc hàm *ENO→*(*ENAME, TITLE*) là phụ thuộc hàm bộ phận vì vế phải là tập con thực sự của khoá. Vậy EMP không ở dạng chuẩn thứ 2.

Lược đồ của PROJ không có phụ thuộc hàm bộ phận, vậy nó ở dạng chuẩn 2.

- Xét quan hệ KHO_HANG(*Kho, Hang, QuayHang, NhanVien*). Lược đồ của quan hệ này có hai phụ thuộc hàm sau:

Kho,Hang→QuayHang: Mỗi mặt hàng ở mỗi kho chỉ được bán ở 1 quầy hàng;

Kho,QuayHang→NhanVien: Mỗi quầy hàng của mỗi kho chỉ có 1 nhân viên phụ trách.

Khoá của lược đồ này là (*Kho, Hang*).

Vậy lược đồ này ở dạng chuẩn thứ 2 vì không có phụ thuộc hàm bộ phận.

5.4.3. Dạng chuẩn thứ 3 (3NF)

Phụ thuộc hàm X→A gọi là *phụ thuộc hàm bắc cầu*, nếu nó là phụ thuộc hàm nguyên tố, A là thuộc tính không khoá, A∉X, và X chứa thuộc tính không khoá.

Khi đó với mọi khoá K ta có các phụ thuộc hàm không tầm thường K→X & X→A. Mặt khác không thể có X→K, vì X chứa các thuộc tính không khoá và không chứa khoá (vì X→A là nguyên tố).

Nói một cách khác phụ thuộc hàm bắc cầu là sự phụ thuộc không tầm thường giữa các thuộc tính không khoá.

Một lược đồ quan hệ gọi là *ở dạng chuẩn thứ 3* nếu nó ở dạng chuẩn thứ 2 và không có phụ thuộc hàm bắc cầu.

◊ *Ví dụ*

- Lược đồ của quan hệ

EMP(*ENO, ENAME, TITLE, SAL, PNO, RESP, DUR*)

có khoá là (*ENO, PNO*).

Phụ thuộc hàm *TITLE→SAL* là phụ thuộc hàm bắc cầu. Vậy EMP không ở dạng chuẩn thứ 3.

- Lược đồ của quan hệ

PROJ(*PNO, PNAME, BUDGET*)

không có phụ thuộc hàm bắc cầu, vậy nó ở dạng chuẩn 3.

- Xét quan hệ KHO_HANG(*Kho, Hang, QuayHang, NhanVien*). Ta có hai phụ thuộc hàm sau:

Kho,Hang→QuayHang: Mỗi mặt hàng ở mỗi kho chỉ được bàn ở 1 quầy hàng;

Kho,QuayHang→NhanVien: Mỗi quầy hàng của mỗi kho chỉ có 1 nhân viên phụ trách.

Khoá của lược đồ này là (*Kho, Hang*).

Phụ thuộc hàm thứ hai là phụ thuộc hàm bắc cầu, vì thế lược đồ không ở dạng chuẩn thứ 3, mặc dù nó ở dạng chuẩn thứ 2.

5.4.4. Dạng chuẩn Boyce-Codd (BCNF)

Một lược đồ quan hệ gọi là ở *dạng chuẩn Boyce-Codd* nếu mọi phụ thuộc hàm không tầm thường đều có vế trái là siêu khoá

◊ *Ví dụ*:

- Lược đồ của quan hệ

PROJ(*PNO, PNAME, BUDGET*)

chỉ có phụ thuộc hàm duy nhất *PNO→*(*PNAME, BUDGET*), vậy nó ở dạng chuẩn Boyce-Codd.

- Xét lược đồ

LOPHOC(*Lop,MonHoc,GiaoVien*) với 2 phụ thuộc hàm sau:

$$GiaoVien \rightarrow MonHoc \quad \text{và} \quad (Lop,MonHoc) \rightarrow GiaoVien$$

Lược đồ có 2 khoá

$$K_1 = (Lop, MonHoc) \text{ và } K_2 = (Lop, GiaoVien),$$

nên tất cả thuộc tính đều là thuộc tính khoá. Như vậy lược đồ ở dạng chuẩn thứ 3. Tuy nhiên lược đồ không ở dạng chuẩn Boyce-Codd vì phụ thuộc hàm

$$GiaoVien \rightarrow MonHoc$$

không thoả yêu cầu vế trái phải là siêu khoá.

Sự dị thường khi thêm bộ hay sửa bộ thể hiện ở chỗ nếu một giáo viên dạy nhiều lớp (cùng một môn học) thì thông tin về giáo viên đó lặp lại nhiều lần gây dư thừa dữ liệu.

Sự dị thường khi xoá bộ thể hiện ở chỗ nếu giáo viên T chỉ dạy lớp C nào đó, thì thông tin về giáo viên T (môn học mà giáo viên đó dạy) sẽ bị mất nếu ta xoá bản ghi tương ứng (chẳng hạn vì giáo viên T thôi không dạy lớp C nữa).

5.4.5. Dạng chuẩn thứ 4 (4NF)

Một quan hệ R được gọi là *ở dạng chuẩn thứ 4*, nếu với mỗi phụ thuộc đa trị X→→Y trong R, X cũng xác định hàm tất cả thuộc tính của R.

Như vậy, *nếu quan hệ ở dạng chuẩn BCNF và các phụ thuộc đa trị cũng là phụ thuộc hàm thì quan hệ này ở dạng chuẩn 4*.

◊ *Ví dụ*: Xét quan hệ

SKILL(*ENO, PNO, PLACE*)

ENO	*PNO*	*PLACE*
E1	P1	Toronto
E1	P1	New York
E1	P1	London
E1	P2	Toronto
E1	P2	New York
E1	P2	London
E2	P1	Toronto
E2	P1	New York
E2	P1	London
E2	P2	Toronto
E2	P2	New York
E2	P2	London

Chú ý rằng không có phụ thuộc hàm nào trong quan hệ SKILL; tất cả thuộc tính là thuộc tính khoá. Quan hệ SKILL có hai phụ thuộc đa trị

ENO→→PNO
ENO→→PLACE

Vì quan hệ không có phụ thuộc hàm nên nó ở dạng BCNF. Tuy nhiên nó không ở dạng chuẩn 4, vì *ENO* không phải là khoá.

Để đạt dạng chuẩn 4, cần phân rã SKILL thành hai quan hệ

EP(*ENO, PNO*) và EL(*ENO, PLACE*)

5.4.6. Dạng chuẩn thứ 5 (5NF)

Một quan hệ R được gọi là *ở dạng chuẩn thứ 5*, còn gọi là *dạng chuẩn chiếu-nối* PJNF, nếu mỗi phụ thuộc chiếu nối được xác định bởi các khoá của R.

◊ *Ví dụ*

Với quan hệ PROJ(*PNO, PNAME, BUDGET*) ta có phụ thuộc chiếu-nối

{(*PNO, PNAME*), (*PNO, BUDGET*)}

và mỗi thành phần đều có khoá chính PNO. Vì vậy PROJ ở dạng chuẩn 5.

CHƯƠNG 2

CƠ SỞ DỮ LIỆU PHÂN TÁN

1. Hệ quản trị cơ sở dữ liệu phân tán

1.1. Khái niệm hệ quản trị cơ sở dữ liệu phân tán

Công nghệ các *hệ quản trị cơ sở dữ liệu phân tán* (*distributed database management system - distributed DBMS*) là sự hợp nhất của hai hướng tiếp cận đối với quá trình xử lý dữ liệu: *Công nghệ cơ sở dữ liệu* và *công nghệ mạng máy tính*. Xử lý dữ liệu chuyển từ hệ thống xử lý file cổ điển sang dạng cơ sở dữ liệu, quản lý tập trung. Điều này dẫn đến tính *độc lập dữ liệu*, nghĩa là các ứng dụng được "miễn nhiễm" đối với những thay đổi về tổ chức lôgic hoặc vật lý của dữ liệu và ngược lại.

Một trong những động lực chủ yếu thúc đẩy sử dụng cơ sở dữ liệu là nhu cầu tích hợp các dữ liệu hoạt tác của một xí nghiệp và cho phép truy xuất tập trung. Công nghệ mạng máy tính đặc trưng ở chỗ phi tập trung hoá thiết bị. Tuy nhiên điểm mấu chốt của ý tưởng cơ sở dữ liệu phân tán là *tích hợp* (*integration*), chứ không phải *tập trung hoá* (*centralization*). Cần hiểu rằng có thể tích hợp mà không cần tập trung. Và đây chính là mục tiêu của công nghệ cơ sở dữ liệu phân tán.

Tại sao chúng ta phải thực hiện phân tán ? Câu trả lời kinh điển cho câu hỏi này là việc xử lý phân tán nhằm thích ứng tốt hơn với việc phân bố ngày càng rộng rãi các công ty, xí nghiệp. Nhiều ứng dụng hiện tại của công nghiệp máy tính được phân tán. Thương mại điện tử, các ứng dụng đa phương tiện, giáo dục từ xa, chữa bệnh từ xa, điều khiển sản xuất từ xa, ... là các ví dụ minh hoạ.

Tuy nhiên từ góc độ tổng quát hơn, xử lý phân tán là một biến thể của qui tắc "*chia để trị*" nhằm giải quyết tốt hơn các bài toán lớn và phức tạp. Từ quan điểm kinh tế, cách tiếp cận này có ưu điểm cơ bản là việc tính toán phân tán tận dụng sức mạnh của nhiều bộ phận xử lý một cách tối ưu.

- ***Hệ cơ sở dữ liệu phân tán là gì ?***

Chúng ta có thể định nghĩa một *cơ sở dữ liệu phân tán* là tập hợp nhiều cơ sở dữ liệu có liên quan lôgic và được phân bổ trên một mạng máy tính.

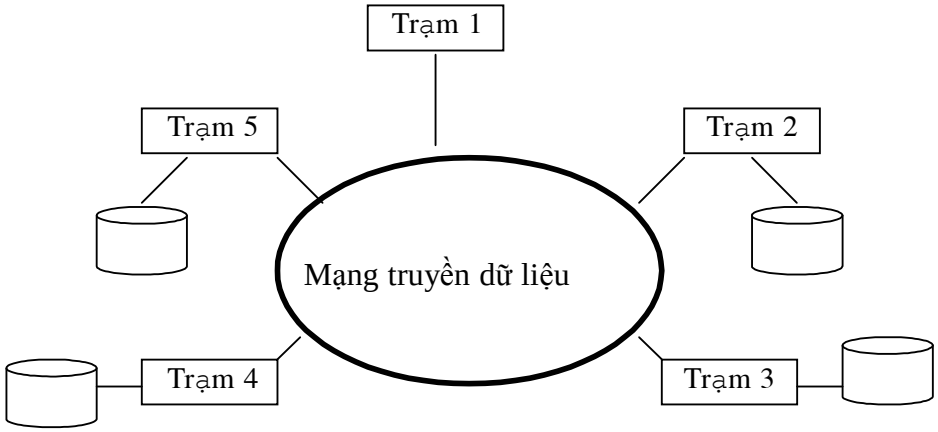
Hệ quản trị cơ sở dữ liệu phân tán (*distributed database management system - distributed DBMS*) là hệ thống phần mềm cho phép quản lý các hệ cơ sở dữ liệu phân tán và làm cho việc phân tán trở nên *vô hình* đối với người sử dụng.

Lưu ý rằng một hệ cơ sở dữ liệu phân tán không phải là tập hợp các tập tin lưu trữ riêng rẽ tại các nút của mạng máy tính. Để tạo ra một hệ cơ sở dữ liệu phân tán, các tập tin không chỉ liên đới lôgic mà còn phải có cấu trúc chung và được truy xuất qua một giao diện chung. Cũng cần phân biệt một hệ cơ sở dữ liệu phân tán với các *dữ liệu bán cấu trúc* (*semi-structrured data*) được lưu trên Internet (chẳng hạn như các trang Web).

Định nghĩa trên cũng loại bỏ các hệ thống đa bộ xử lý dùng chung bộ nhớ trong (shared memory) hoặc dùng chung bộ nhớ thứ cấp (shared disk).

Ngoài ra một hệ cơ sở dữ liệu phân tán không phải là hệ thống mà trong đó, mặc dù có sự hiện diện của mạng máy tính, cơ sở dữ liệu chỉ nằm tại một nút của mạng.

Môi trường của một hệ CSDL phân tán có thể biểu diễn bằng sơ đồ sau



• ***Các đặc trưng của cơ sở dữ liệu phân tán***

Đặc tính *vô hình* là sự tách biệt về ngữ nghĩa ở mức độ cao của hệ thống với các vấn đề cài đặt ở cấp độ thấp. Ưu điểm của hệ cơ sở dữ liệu vô hình là không cho người dùng "nhìn thấy" các chi tiết cài đặt, hỗ trợ phát triển các ứng dụng phức tạp.

Độc lập dữ liệu là dạng vô hình cơ bản cần có trong một hệ cơ sở dữ liệu. Sự độc lập dữ liệu liên quan đến khả năng "miễn nhiễm" của các ứng dụng đối với những thay đổi trong định nghĩa và tổ chức dữ liệu, và ngược lại.

Vô hình kết mạng. Trong môi trường phân tán, hệ thống mạng là một loại tài nguyên quan trọng cần quản lý. Thông thường, người dùng cần được tách khỏi mọi chi tiết hoạt động của mạng, thậm chí người ta mong muốn che dấu sự tồn tại của mạng, nếu được. Khi đó đối với người dùng sẽ không có sự khác biệt giữa các ứng dụng chạy trên cơ sở dữ liệu tập trung và các ứng dụng chạy trên cơ sở dữ liệu phân tán. Kiểu vô hình này gọi là *vô hình kết mạng* (*network transparency*) hoặc *vô hình phân bổ* (*distribution transparency*).

Vô hình nhân bản. Vì những lý do về hiệu năng (performance), độ tin cậy (reliability) và tính sẵn sàng (availability), người ta mong muốn có thể nhân dữ liệu thành nhiều bản (nhân bản) trên các máy mạng. Việc nhân bản giúp tăng hiệu năng vì những yêu cầu sử dụng có xung đột và nằm rải rác có thể đáp ứng kịp thời. Thí dụ, dữ liệu thường được một người truy xuất có thể được đặt tại máy của người đó và trên máy của những người khác có cùng nhu cầu truy xuất, như thế sẽ làm tăng khu vực truy xuất. Ngoài ra nếu một máy phải ngưng hoạt động, một bản sao khác của dữ liệu vẫn có sẵn trên máy khác của mạng. Tuy nhiên việc nhân bản sẽ gây khó khăn khi cập nhật cơ sở dữ liệu. Vì vậy việc nhân bản và qui mô nhân bản do các ứng dụng quyết định.

Vô hình phân mảnh. Phân hoạch dữ liệu cho các vị trí khác nhau là yêu cầu tất yếu của hệ phân tán. Quá trình này gọi là *quá trình phân mảnh* (*fragmentation*). Có hai kiểu phân mảnh. *Phân mảnh ngang* (*horizontal fragmentation*), trong đó mỗi quan hệ được phân hoạch thành tập các quan hệ con, mỗi quan hệ con này chứa một tập con các bộ của quan hệ ban đầu. *Phân mảnh dọc* (*vertical fragmentation*), trong đó mỗi quan hệ được phân hoạch thành tập các quan hệ

con, mỗi quan hệ con này được định nghĩa trên một tập con các thuộc tính của quan hệ ban đầu).

Khi các đối tượng cơ sở dữ liệu bị phân mảnh, chiến lược xử lý vấn tin là dựa trên các mảnh chứ không phải quan hệ. Như vậy *câu vấn tin toàn cục* (*global query*) phải được dịch thành *câu vấn tin theo mảnh* (*fragment query*).

1.2. Mô hình kiến trúc hệ quản trị cơ sở dữ liệu phân tán

Kiến trúc của một hệ thống xác định cấu trúc của nó. Tức là các thành phần của hệ thống được xác định, chức năng mỗi thành phần được mô tả, các mối tương liên (interrelationship) và tương tác (interaction) giữa các thành phần được định nghĩa.

Chúng ta hãy xem xét một số cách kết hợp nhiều cơ sở dữ liệu lại để dùng chung cho nhiều hệ quản trị cơ sở dữ liệu. Ta phân loại hệ thống theo các đặc điểm (1) *tính tự trị* (*autonomy*) của các hệ thống cục bộ, (2) *tính phân tán* (*distribution*) của chúng, (3) *tính đa chủng* (*heterogeneity*) của chúng.

1.2.1. Tính tự trị

Tính tự trị (*autonomy*) muốn nói đến sự *phân bổ quyền điều khiển*, chứ không phải phân bố dữ liệu. Tính tự trị chỉ ra mức độ hoạt tác độc lập của từng hệ quản trị cơ sở dữ liệu. Tính tự trị biểu hiện qua một số yếu tố sau:

- Các hệ thống thành viên có trao đổi thông tin với nhau không.
- Các hệ thống thực hiện các giao dịch một cách độc lập hay không.
- Các hệ thống có được sửa đổi hay không.

Từ đó người ta xây dựng các yêu cầu đối với một hệ thống tự trị. Chẳng hạn hệ thống tự trị phải thoả mãn:

(1) Các hoạt động cục bộ của từng hệ quản trị cơ sở dữ liệu không bị ảnh hưởng bởi sự tham gia của chúng vào trong phức hệ cơ sở dữ liệu (multidatabase system).

(2) Phương thức xử lý và tối ưu hoá vấn tin trong từng hệ quản trị cơ sở dữ liệu không bị ảnh hưởng bởi việc thực hiện các câu truy vấn toàn cục truy xuất nhiều cơ sở dữ liệu.

(3) Tính nhất quán và hoạt động của hệ thống không bị ảnh hưởng khi từng hệ quản trị cơ sở dữ liệu riêng rẽ tham gia hoặc tách ra khỏi liên minh cơ sở dữ liệu.

Ở bình diện khác, tính tự trị thể hiện ở các khía cạnh sau:

(1) *Tự trị thiết kế* (*design autonomy*): Mỗi hệ quản trị cơ sở dữ liệu tự do sử dụng các mô hình dữ liệu và các kỹ thuật quản lý giao dịch thích hợp.

(2) *Tự trị truyền thông* (*communication autonomy*): Mỗi hệ quản trị cơ sở dữ liệu tự do quyết định loại thông tin cung cấp cho hệ quản trị cơ sở dữ liệu khác hoặc cho các phần mềm điều khiển hoạt động toàn cục.

(3) *Tự trị thực thi* (*execution autonomy*): Mỗi hệ quản trị cơ sở dữ liệu có thể thực hiện các giao dịch theo phương thức của mình.

Tính tự trị có thể chia làm ba cấp độ sau:

(**0**) *Tích hợp mật thiết* (*tight integration*): Chỉ tồn tại một hình ảnh duy nhất về toàn bộ hệ thống cơ sở dữ liệu cho người dùng muốn dùng chung thông tin trong

nhiều cơ sở dữ liệu. Một trong các bộ quản lý dữ liệu (data manager) nắm quyền kiểm soát việc xử lý yêu cầu của người dùng, ngay cả khi yêu cầu đó phải được nhiều bộ quản lý dữ liệu tham gia xử lý.

(**1**) *Hệ thống bán tự trị* (*semiautonomous system*): Bao gồm các hệ quản trị cơ sở dữ liệu có thể hoạt tác độc lập, nhưng quyết định tham gia vào liên minh nhằm chia sẻ dữ liệu cục bộ của chúng. Mỗi hệ quản trị cơ sở dữ liệu phải xác định những phần cơ sở dữ liệu nào của riêng chúng mà các hệ quản trị cơ sở dữ liệu khác được truy xuất. Chúng không phải là hệ thống tự trị hoàn toàn mà cần sửa đổi lại để có thể trao đổi thông tin với những hệ thống khác.

(**2**) *Hệ thống cô lập*: Các hệ quản trị cơ sở dữ liệu hoạt động cô lập, không có giao tiếp chia sẻ dữ liệu với nhau.

1.2.2. Tính phân tán

Tính phân tán (*distribution*) chỉ khả năng *phân bố dữ liệu* ở những vị trí khác nhau. Có hai loại kiến trúc phân tán:

(**1**) *Phân tán khách/chủ* (*client/server*): Đây là hình thức phân tán chức năng. Thành phần chủ (server) chịu trách nhiệm quản trị dữ liệu; thành phần khách (client) chịu trách nhiệm cung cấp môi trường ứng dụng, kể cả giao diện người dùng.

Nhiệm vụ truyền thông được chia sẻ giữa chủ và khách.

(**2**) *Phân tán ngang hàng* (*peer-to-peer*): Còn gọi là *phân tán hoàn toàn*. Không có sự phân biệt giữa máy chủ và khách, mỗi máy đều có đầy đủ chức năng của một hệ quản trị cơ sở dữ liệu và có thể trao đổi thông tin với máy khác để thực hiện vấn tin và giao dịch.

1.2.3. Tính đa chủng

Tính đa chủng (*heterogeneous*) thể hiện dưới nhiều hình thái khác nhau trong các hệ phân tán, từ khác biệt về phần cứng, các giao thức kết nối mạng đến sự khác biệt của các bộ quản lý dữ liệu.

Tính đa chủng liên quan đến sự khác biệt các mô hình dữ liệu (data model), ngôn ngữ vấn tin (query language) và nghi thức quản lý giao dịch (transaction management protocol).

1.2.4. Các kiểu kiến trúc

Ta tổ hợp các mức độ tự trị, phân tán và đa chủng để nghiên cứu các mô hình kiến trúc khác nhau.

Ký hiệu

A0 ... hệ thống tự trị tích hợp
A1 ... hệ thống bán tự trị
A2 ... hệ thống cô lập
D0 ... hệ thống không phân tán (tập trung)
D1 ... hệ thống phân tán khách/chủ
D2 ... hệ thống phân tán ngang hàng
H0 ... hệ thống đồng chủng

H1 ... hệ thống đa chủng

• Mô hình (A0,D0,H0): *Hệ thống tích hợp, không phân tán, đồng chủng*. Được gọi là *hệ thống phức hợp* (composite system), bao gồm nhiều hệ quản trị cơ sở dữ liệu được tích hợp về mặt lôgic. Kiến trúc này phù hợp với những hệ thống đa bộ xử lý và mọi tài nguyên dùng chung.

• Mô hình (A0,D0,H1): *Hệ thống tích hợp, không phân tán, đa chủng*. Nó có nhiều hệ quản trị cơ sở dữ liệu đa chủng, nhưng cung cấp một hình ảnh tích hợp cho người dùng. Chẳng hạn trên cơ sở dữ liệu mạng cùng hiện diện cả cơ sở dữ liệu phân cấp và cơ sở dữ liệu quan hệ.

• Mô hình (A0,D1,H0): *Hệ thống tích hợp, phân tán khách/chủ, đồng chủng*. Hệ thống cung cấp một hình ảnh tích hợp cho người dùng.

• Mô hình (A0,D2,H0): *Hệ thống tích hợp, phân tán hoàn toàn, đồng chủng*. Hệ thống không phân biệt khách chủ. Mỗi vị trí đều trang bị đầy đủ chức năng.

• Mô hình (A1,D0,H0): *Hệ thống bán tự trị, không phân tán, đồng chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu liên bang* (*federated DBMS*).

Các hệ thống thành viên đều có quyền tự trị nhất định trong các hoạt động của chúng. Chúng tự do hiệp đồng với những hệ thống khác khi thực hiện các yêu cầu của người dùng truy xuất đến nhiều cơ sở dữ liệu.

◊ *Ví dụ*. Nhiều bản cài đặt một hệ quản trị cơ sở dữ liệu "*mở*" trên cùng một máy. "Mở" ở đây có nghĩa là hệ quản trị cơ sở dữ liệu có khả năng tham gia vào liên bang.

• Mô hình (A1,D0,H1): *Hệ thống bán tự trị, không phân tán, đa chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu liên bang đa chủng* (heterogeneous federated DBMS).

◊ *Ví dụ*. Một hệ quản trị cơ sở dữ liệu quan hệ lo quản lý dữ liệu có cấu trúc, một hệ quản trị cơ sở dữ liệu đồ hoạ lo quản lý hình ảnh tĩnh, và một máy chủ cung cấp các hình video. Nếu chúng ta muốn cung cấp một hình ảnh tích hợp cho người dùng thì cần phải "che dấu" tính tự trị và đa chủng của các hệ thống thành viên và thiết lập giao diện chung.

• Mô hình (A1,D1,H1): *Hệ thống bán tự trị, phân tán khách/chủ, đa chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu liên bang đa chủng phân tán* (heterogeneous federated distributed DBMS). Dữ liệu được phân tán trên các máy khác nhau.

• Kiến trúc (A2,D0,H0): *Hệ thống cô lập, không phân tán, đồng chủng*, được gọi dưới cái tên *hệ quản trị cơ sở dữ liệu phức hệ* (multidatabase system-MDBS). Các thành viên không có khái niệm hiệp đồng. Đây thực chất là tập các cơ sở dữ liệu tự trị và được kết nối lại.

• Mô hình (A2,D0,H1): *Hệ thống cô lập, không phân tán, đa chủng*. Hệ thống này được dùng để xây dựng các ứng dụng truy xuất dữ liệu từ nhiều hệ thống lưu trữ khác nhau với các đặc tính khác nhau. Một số hệ thống có thể không phải là hệ quản trị cơ sở dữ liệu.

• Mô hình (A2,D1,H1) và (A2,D2,H1): Ta xét chung hai trường hợp này do tính tương tự của những vấn đề do chúng sinh ra. Cả hai đều biểu diễn cho trường hợp các cơ sở dữ liệu thành viên tạo ra phức hệ phân tán trên một số vị trí – chúng được gọi là các *phức hệ cơ sở dữ liệu phân tán* (distributed MDBS). Khác biệt chính giữa hai kiến trúc này là ở phân tán khách/chủ, phần lớn các công việc tương tác được trao cho *hệ thống trung gian* (middleware system), tạo ra *kiến trúc ba tầng* (three layer architecture).

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

Chúng ta sẽ xem xét chi tiết ba kiến trúc hệ thống trong số các kiến trúc giới thiệu ở phần trước. Đó là các hệ thống khách/chủ (Ax,D1,Hy), các hệ phân tán (A0,D2,H0) và các phức hệ (A2,Dx,Hy).

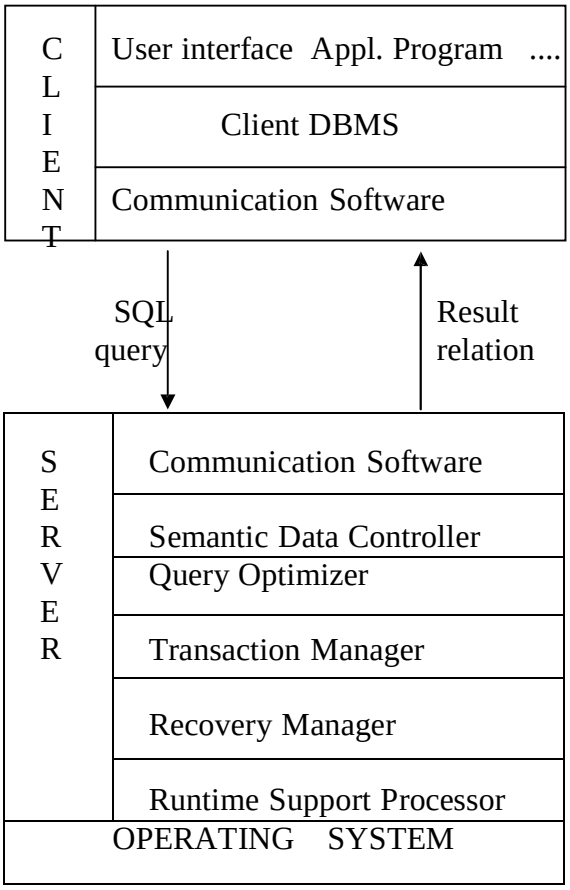
1.3.1. Các hệ khách/chủ

Các hệ quản trị cơ sở dữ liệu *khách/chủ* xuất hiện đầu những năm 90 và có ảnh hưởng lớn đến công nghệ DBMS và phương thức xử lý tính toán. ý tưởng tổng quát hết sức đơn giản và rõ ràng: phân biệt các chức năng cần được cung cấp và chia những chức năng này thành hai lớp: *chức năng chủ* (server function) và *chức năng khách* (client function). Nó cung cấp một *kiến trúc hai tầng* (*two-level architecture*), tạo điều kiện dễ dàng cho việc quản lý mức độ phức tạp của các hệ quản trị cơ sở dữ liệu hiện đại và độ phức tạp của việc phân tán dữ liệu.

Máy chủ thực hiện phần lớn công việc quản lý dữ liệu: xử lý tối ưu hoá vấn tin, quản lý giao dịch, quản lý thiết bị lưu trữ.

Máy khách quản lý các ứng dụng, giao diện, hệ quản trị cơ sở dữ liệu của khách, chịu trách nhiệm quản lý dữ liệu được gửi cho khách và có thể cả quản lý các khoá chốt giao dịch.

Kiến trúc này được mô tả trong hình sau.



kiến trúc tham chiếu khách/chủ

Kiến trúc này thông dụng trong các hệ cơ sở dữ liệu quan hệ, ở đó việc giao tiếp giữa khách và chủ nằm ở mức câu lệnh SQL. Khách hàng chuyển câu vấn tin cho máy chủ mà không cần biết nó thực hiện và tối ưu hoá như thế nào. Máy chủ thực hiện hầu hết công việc và gửi quan hệ kết quả về cho khách.

Có một số kiến trúc khách/chủ khác nhau:

• *Kiến trúc nhiều khách, một chủ.*

Từ góc độ quản lý, loại này không khác nhiều so với cơ sở dữ liệu tập trung, vì CSDL được lưu trên một máy chủ duy nhất và cũng có phần mềm quản lý.

Có một số khác biệt quan trọng ở cách thực hiện các giao dịch và quản lý bộ nhớ cache.

• *Kiến trúc nhiều khách, nhiều chủ*:

Có hai chiến lược quản lý:

- Mỗi máy khách tự quản lý kết nối của nó với các máy chủ khác nhau. Lối tiếp cận này làm đơn giản chương trình ở máy chủ nhưng đặt gánh nặng lên máy khách cùng với các trách nhiệm khác.
- Mỗi máy khách chỉ quan hệ với máy chủ đại diện của mình và giao tiếp với các máy chủ khác thông qua đại diện khi cần.

1.3.2. Các hệ phân tán ngang hàng

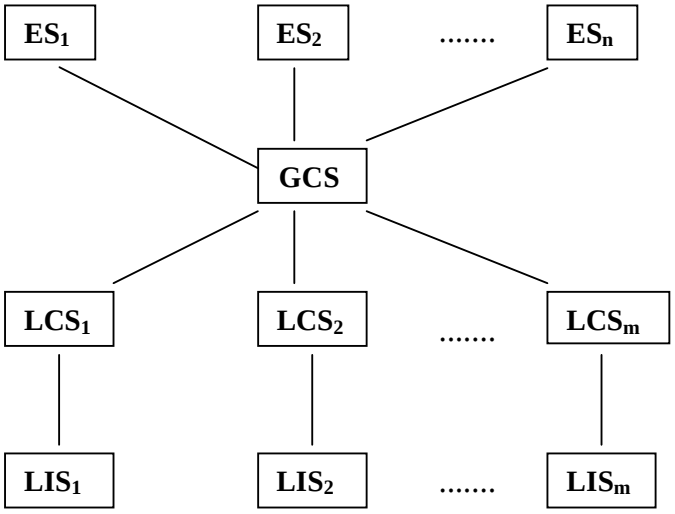
Trong kiến trúc này tổ chức dữ liệu vật lý trên mỗi máy có thể rất khác nhau. Điều này dẫn đến việc định nghĩa cấu trúc dữ liệu riêng cho mỗi vị trí, gọi là *lược đồ nội tại cục bộ* LIS (local internal schema). Cấu trúc lôgic của dữ liệu ở mọi vị trí được mô tả bằng *lược đồ khái niệm toàn cục* GCS (global conceptual schema).

Để mô tả tổ chức lôgic của dữ liệu tại mỗi vị trí cần phải có *tầng thứ ba* trong kiến trúc gọi là *lược đồ khái niệm cục bộ* LCS (local conceptual schema).

Lược đồ khái niệm toàn cục lúc này là *hợp* của các lược đồ khái niệm cục bộ.

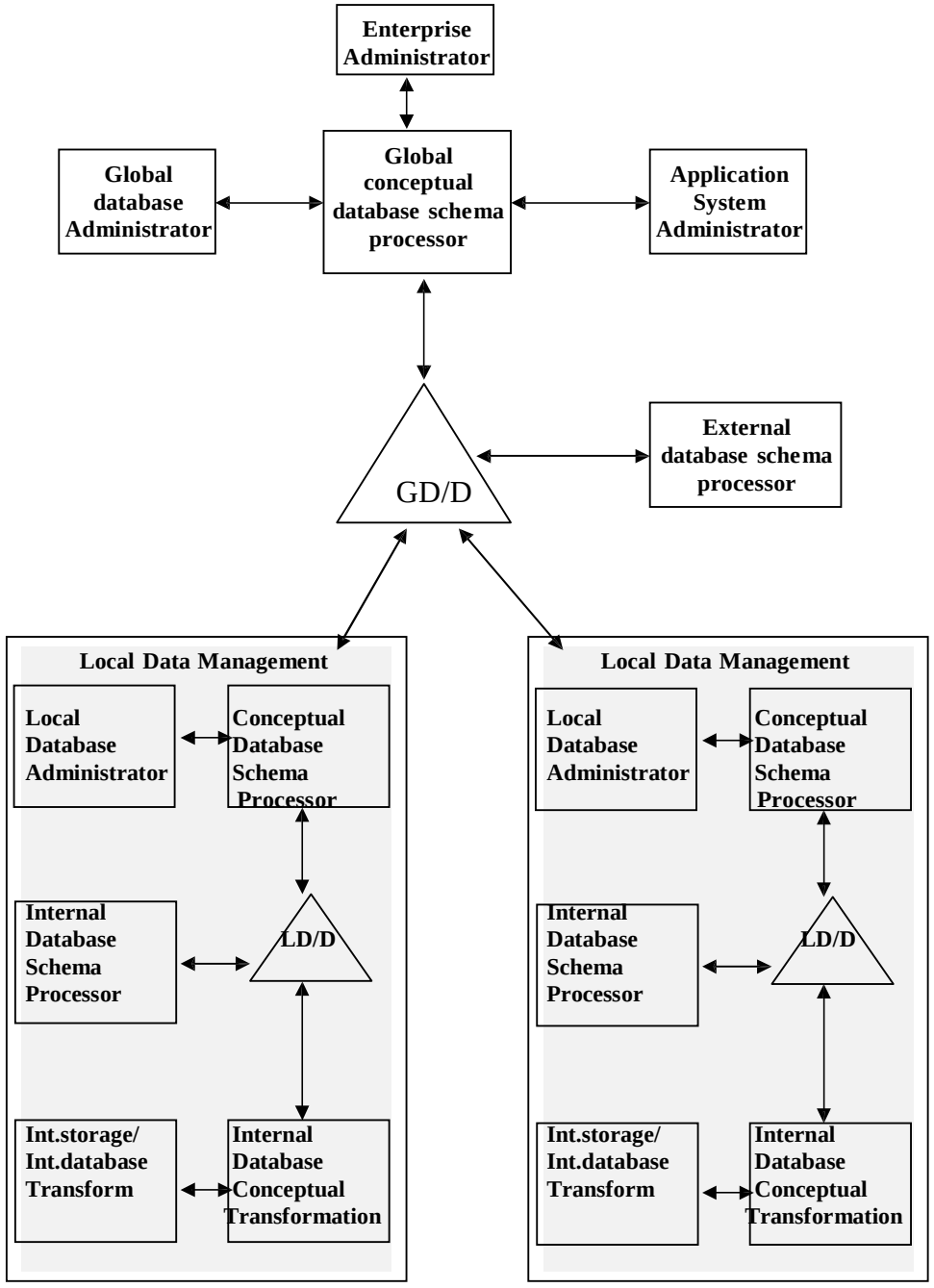
Các ứng dụng và việc truy xuất cơ sở dữ liệu được hỗ trợ qua các *lược đồ ngoại giới* ES (external schema), được định nghĩa là một tầng nằm trên tầng lược đồ khái niệm toàn cục.

Kiến trúc này được mô tả trong hình sau.



kiến trúc tham chiếu CSDL phân tán ngang hàng

Các chức năng của hệ phân tán ngang hàng được biểu diễn bằng sơ đồ sau:



sơ đồ chức năng của hệ quản trị CSDL phân tán ngang hàng

Trong hệ thống này các *từ điển/thư mục dữ liệu*, viết tắt là D/D, có vai trò trung tâm, vừa xử lý lược đồ dữ liệu vừa cung cấp các ánh xạ giữa chúng ở các cấp độ toàn cục và cục bộ.

• *Thư mục/từ điển toàn cục* GD/D (global directory/dictionary) chứa các định nghĩa lược đồ toàn cục và thực hiện các ánh xạ toàn cục.

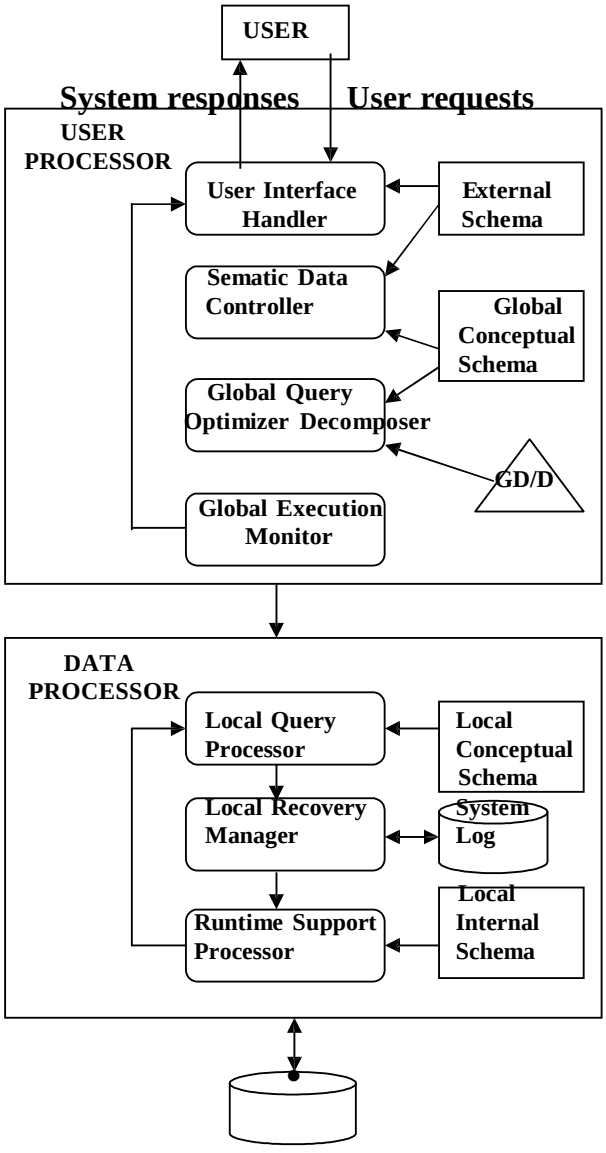
• *Thư mục/từ điển cục bộ* LD/D (local directory/dictionary) chứa các định nghĩa lược đồ cục bộ và thực hiện các ánh xạ cục bộ. Nó cũng có thể chứa các số liệu thống kê về các ứng dụng, các thông tin kiểm soát truy xuất,...

Các thành phần quản lý cơ sở dữ liệu cục bộ được tích hợp nhờ các chức năng của hệ quản trị cơ sở dữ liệu toàn cục.

Trong sơ đồ này, lược đồ khái niệm cục bộ là ánh xạ của lược đồ khái niệm toàn cục vào mỗi vị trí. Hơn nữa, những cơ sở dữ liệu loại này thường được thiết kế theo kiểu từ trên xuống, vì thế tất cả định nghĩa khung nhìn đều có phạm vi toàn cục.

Mỗi vị trí cũng có một quản trị viên cơ sở dữ liệu thể hiện mong muốn có được khả năng điều khiển cục bộ đối với hoạt động quản trị cơ sở dữ liệu.

Các thành phần của hệ quản trị cơ sở dữ liệu phân tán được trình bày trong hình sau:



Hệ thống có hai thành phần chính: Một thành phần lo xử lý mọi tương tác với người dùng gọi là *bộ phận phục vụ người dùng* (user processor), còn thành phần thứ hai lo việc lưu trữ dữ liệu, gọi là *bộ phận xử lý dữ liệu* (data processor).

- *Bộ phận phục vụ người dùng* (user processor): bao gồm bốn phần:
  (1) *Bộ phận giao tiếp* (user interface handler) chịu trách nhiệm diễn dịch các yêu cầu của người dùng (user requests) và định dạng dữ liệu kết quả để chuyển cho người dùng.
  (2) *Bộ phận kiểm soát dữ liệu ngữ nghĩa* (sematic data controller): sử dụng các *ràng buộc toàn vẹn* (integrity constraints) và thông tin quyền hạn (authorization), được định nghĩa như thành phần của lược đồ khái niệm toàn cục, để kiểm tra xem các câu vấn tin có thể xử lý được hay không.
  (3) *Bộ phận phân rã và tối ưu hoá vấn tin* (global query optimizer and decomposer) xác định chiến lược hoạt động nhằm giảm thiểu chi phí, phiên dịch các câu vấn tin toàn cục thành các câu vấn tin cục bộ bằng cách sử dụng các lược đồ khái niệm toàn cục, lược đồ khái niệm cục bộ và các thư mục toàn cục. Bộ phận tối ưu vấn tin toàn cục, ngoài những nhiệm vụ khác, còn chịu trách nhiệm tạo ra chiến lược thực thi tốt nhất cho các phép nối phân tán.
  (4) *Bộ phận theo dõi hoạt động phân tán* (distributed execution monitor) điều phối việc thực hiện phân tán các yêu cầu người dùng và cũng được gọi là *bộ quản lý giao dịch phân tán* (distributed transaction manager). Khi thực hiện các vấn tin phân tán, các bộ phận tại các vị trí có thể giao tiếp với nhau.

- *Bộ phận xử lý dữ liệu* (data processor): bao gồm ba phần.
  (1) *Bộ phận xử lý câu vấn tin cục bộ* (local query processor): hoạt động như *bộ chọn đường truy xuất* (access path selector), chịu trách nhiệm chọn ra một đường truy xuất thích hợp nhất để truy xuất các mục dữ liệu.
  (2) *Bộ phận khôi phục cục bộ* (local recovery manager): bảo đảm cho các cơ sở dữ liệu cục bộ vẫn duy trì được tính nhất quán ngay cả khi có sự cố xảy ra.
  (3) *Bộ phận hỗ trợ thực thi* (run-time support processor): truy xuất cơ sở dữ liệu tuỳ vào các lệnh trong *lịch biểu* (schedule) do bộ phận tối ưu vấn tin sinh ra. Nó chính là giao diện với hệ điều hành và chứa *bộ quản lý vùng đệm cơ sở dữ liệu* (database buffer manager), chịu trách nhiệm quản lý vùng đệm và việc truy xuất dữ liệu.

Lưu ý rằng việc sử dụng thuật ngữ *Bộ phận phục vụ người dùng* và *Bộ phận xử lý dữ liệu* không phải là sự phân chia chức năng giống như các hệ khách/chủ. Sự phân chia này chỉ thể hiện khía cạnh tổ chức và không bắt buộc phải đặt trên các máy khác nhau. Trong các hệ thống ngang hàng, người ta mong muốn có cả môdun phục vụ người dùng và môđun xử lý dữ liệu trên cùng một máy.

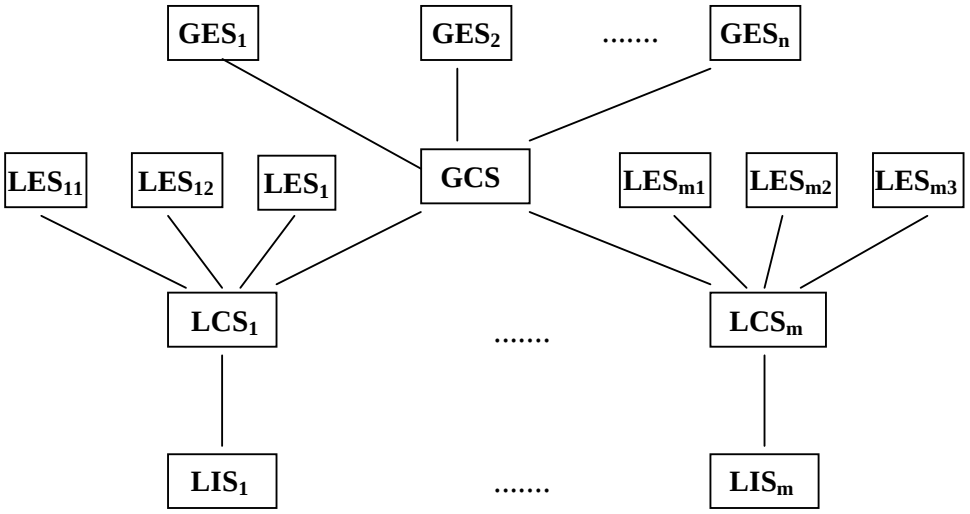
1.3.3. Các phức hệ cơ sở dữ liệu

Hệ quản trị phức hệ cơ sở dữ liệu phân tán khác biệt với hệ quản trị cơ sở dữ liệu phân tán về

- mức độ tự trị: phản ánh trong các mô hình kiến trúc
- định nghĩa lược đồ khái niệm toàn cục: trong hệ phức hợp lược đồ khái niệm toàn cục chỉ là một tập con bao gồm một số cơ sở dữ liệu cục bộ mà mỗi hệ quản trị CSDL muốn dùng chung, còn trong hệ phân tán cơ sở dữ liệu toàn cục là hợp các cơ sở dữ liệu cục bộ.

a) Các mô hình sử dụng lược đồ khái niệm toàn cục

Trong phức hệ cơ sở dữ liệu, lược đồ khái niệm toàn cục GCS được định nghĩa bằng cách tích hợp các lược đồ ngoài của các cơ sở dữ liệu tự trị hoặc các thành phần của lược đồ khái niệm cục bộ của chúng, xem hình sau



kiến trúc phức hệ CSDL với một lược đồ khái niệm toàn cục

Người dùng của hệ quản trị cơ sở dữ liệu cục bộ sẽ định nghĩa khung nhìn riêng (LES) của họ trên cơ sở dữ liệu cục bộ và không cần thay đổi các ứng dụng hiện có nếu họ không truy xuất dữ liệu của cơ sở dữ liệu khác. Đây chính là khía cạnh tự trị của kiến trúc này.

Thiết kế lược đồ khái niệm toàn cục trong phức hệ cơ sở dữ liệu bao gồm việc tích hợp các lược đồ khái niệm cục bộ (ánh xạ đi từ dưới lên, từ lược đồ khái niệm cục bộ lên lược đồ khái niệm toàn cục, ngược lại với hệ phân tán) hoặc tích hợp các lược đồ ngoài cục bộ (ánh xạ đi theo chiều từ trên xuống).

Một khi đã thiết kế xong GCS, các khung nhìn trên lược đồ có thể định nghĩa cho người dùng cần truy xuất ở phạm vi toàn cục. Các lược đồ ngoại giới toàn cục GES và lược đồ khái niệm toàn cục GCS không nhất thiết sử dụng cùng một mô hình và cùng ngôn ngữ, chúng không cần xác định xem hệ thống *đồng chủng* hay *đa chủng*.

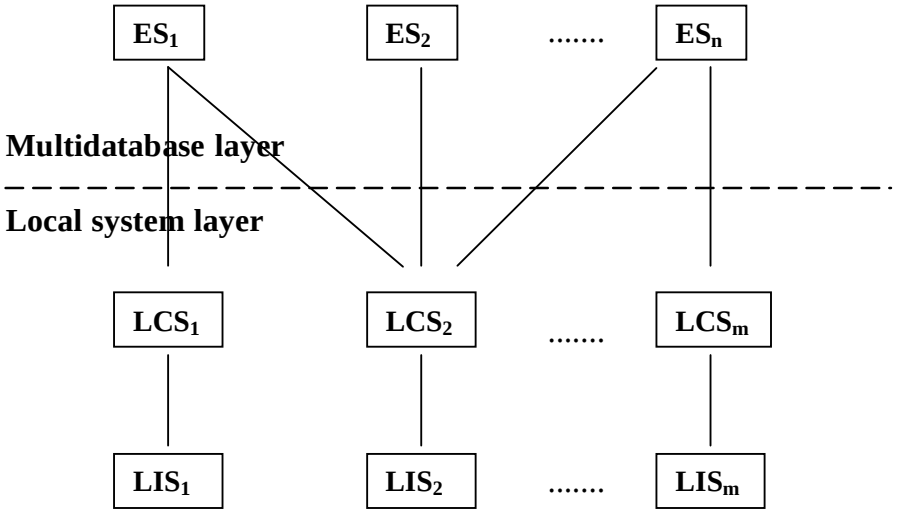
Trong trường hợp đa chủng, ta có hai lựa chọn cài đặt: *đơn ngôn* (unilingal) hoặc *đa ngôn* (multilingal).

Trong phức hệ cơ sở dữ liệu đơn ngôn, khi truy xuất toàn cục người dùng sử dụng chung một lược đồ ngoại giới toàn cục và một ngôn ngữ xử lý toàn cục.

Trong phức hệ cơ sở dữ liệu đa ngôn, khi truy xuất toàn cục mỗi người dùng sử dụng lược đồ ngoại giới toàn cục được định nghĩa bằng ngôn ngữ của hệ quản trị cơ sở dữ liệu cục bộ của mình và các câu vấn tin toàn cục cũng được tạo bằng ngôn ngữ của hệ quản trị cơ sở dữ liệu cục bộ.

b) Các mô hình không có lược đồ khái niệm toàn cục

Kiến trúc của phức hệ cơ sở dữ liệu không có lược đồ khái niệm toàn cục được trình bày trong hình sau



kiến trúc phức hệ CSDL không có lược đồ khái niệm toàn cục

Kiến trúc này có hai tầng: *tầng hệ thống cục bộ* và *tầng phức hệ CSDL* phía trên.

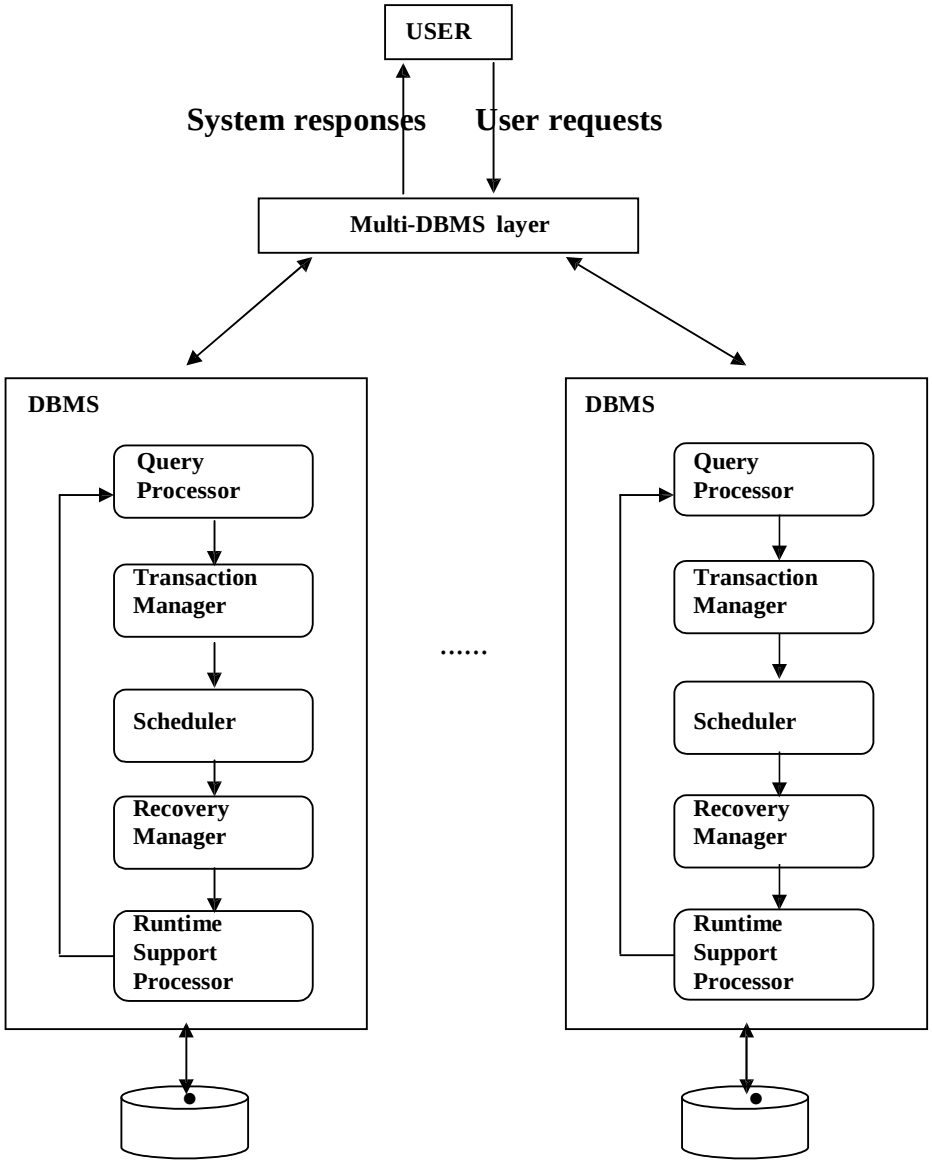
Tầng hệ thống cục bộ bao gồm một số hệ quản trị cơ sở dữ liệu, với chức năng là giới thiệu cho tầng phức hệ cơ sở dữ liệu các thành phần của cơ sở dữ liệu cục bộ có thể dùng chung với những cơ sở dữ liệu khác. Dữ liệu dùng chung này được trình bày qua lược đồ khái niệm cục bộ thực sự hoặc qua lược đồ ngoại giới cục bộ. Nếu có vấn đề đa chủng, mỗi lược đồ LCS_i có thể sử dụng mô hình dữ liệu khác nhau.

Tầng phức hệ cơ sở dữ liệu gồm các lược đồ ngoại giới, trong đó mỗi khung nhìn được định nghĩa trên một hay nhiều lược đồ khái niệm cục bộ. Vì vậy trách nhiệm cung cấp quyền truy xuất đến nhiều cơ sở dữ liệu (có thể đa chủng) được trao cho ánh xạ giữa lược đồ ngoại giới và lược đồ khái niệm cục bộ. Đây là điểm khác biệt cơ bản so với kiến trúc sử dụng lược đồ khái niệm toàn cục, trong đó quan hệ giữa lược đồ ngoại giới và lược đồ khái niệm cục bộ được thực hiện bởi các ánh xạ thông qua lược đồ khái niệm toàn cục.

Kiến trúc cơ sở dữ liệu liên bang cũng không sử dụng lược đồ khái niệm toàn cục. Trong hệ thống này mỗi hệ quản trị cơ sở dữ liệu cục bộ định nghĩa một

lược đồ xuất (export schema), trong đó định nghĩa dữ liệu muốn chia sẻ với các DBMS khác, mỗi ứng dụng truy xuất đến cơ sở dữ liệu toàn cục qua định nghĩa của *lược đồ nhập* (import schema), thực chất là hình ảnh ngoại giới toàn cục.

Các thành phần của hệ quản trị phức hợp khác biệt nhiều so với hệ quản trị cơ sở dữ liệu phân tán. ở đây có một tầng phần mềm chạy bên trên những hệ quản trị cơ sở dữ liệu riêng biệt và cung cấp cho người dùng những tiện ích để truy xuất nhiều cơ sở dữ liệu khác nhau. Tuỳ thuộc vào sự tồn tại hay không lược đồ khái niệm toàn cục và vấn đề đa chủng mà nội dung phần mềm sẽ thay đổi cho phù hợp.



các thành phần của phức hệ CSDL

1.4. Tổ chức thư mục toàn cục

Các vấn đề *tổ chức thư mục toàn cục* chỉ được đề cập đến trong các hệ phân tán và phức hệ có sử dụng lược đồ khái niệm toàn cục.

Thư mục toàn cục cũng là cơ sở dữ liệu chứa dữ liệu về các dữ liệu thực sự lưu trữ trong cơ sở dữ liệu (còn gọi là *meta dữ liệu* hay *siêu dữ liệu*). Vì thế những kỹ thuật thiết kế cơ sở dữ liệu phân tán cũng áp dụng cho việc quản lý thư mục. Như vậy thư mục có thể *toàn cục* đối với toàn bộ cơ sở dữ liệu hoặc *cục bộ* đối với từng vị trí. Nghĩa là có thể có một thư mục duy nhất chứa các thông tin về tất cả dữ liệu trong cơ sở dữ liệu, hoặc có một số thư mục, mỗi thư mục chứa thông tin được lưu ở các vị trí khác nhau. Trong trường hợp sau chúng ta có thể xây dựng hệ phân cấp thư mục để dễ dàng khi tìm kiếm hoặc cài đặt một chiến lược tìm kiếm phân tán cho phép trao đổi giữa các vị trí lưu trữ thư mục.

Vấn đề thứ hai liên quan đến vị trí chứa thư mục. Thư mục có thể được duy trì tập trung tại một vị trí hoặc phân tán đến một số vị trí. Giữ thư mục tại một vị trí làm cho việc quản lý được dễ dàng, nhưng có thể làm tăng tải trọng tại đó, gây ùn tắc lưu lượng thông báo ở vị trí đó. Ngược lại, phân tán thư mục trên nhiều vị trí làm giảm tải trọng tại một điểm nhưng sẽ làm cho vấn đề quản lý thư mục thêm phức tạp. Trong các phức hệ cơ sở dữ liệu, sự lựa chọn sẽ phụ thuộc vào vấn đề hệ thống có phân tán hay không. Nếu có, thư mục sẽ được phân tán, ngược lại nó được quản lý tập trung.

Vấn đề thứ ba là nhân bản thư mục. Có thể có một bản thư mục duy nhất hoặc nhiều bản. Có nhiều bản thư mục sẽ làm tăng độ tin cậy của hệ thống do khả năng truy xuất được một bản thư mục sẽ cao hơn. Hơn nữa thời gian trễ khi truy xuất thư mục sẽ giảm đi do ít xảy ra tranh chấp và khoảng cách đến các bản sao sẽ ngắn hơn. Tuy nhiên, việc duy trì cập nhật các bản sao cũng phức tạp và tốn kém. Vì vậy sự lựa chọn sẽ phụ thuộc vào môi trường hệ thống và phải cân bằng các yếu tố như thời gian đáp ứng, kích thước thư mục, khả năng của máy ở mỗi vị trí, yêu cầu khả tín, mức độ thay đổi của thư mục.

2. Thiết kế cơ sở dữ liệu phân tán

Thiết kế một hệ thống máy tính phân tán cần phải chọn *vị trí đặt dữ liệu* và *chương trình* trên một mạng máy tính, rất có thể phải kể luôn cả việc thiết kế mạng. Đối với hệ quản trị cơ sở dữ liệu phân tán cần phải thực hiện hai điều: *phân tán cơ sở dữ liệu* và *phân tán các chương trình ứng dụng* chạy trên hệ đó. ở đây chúng ta chỉ tập trung vào việc phân tán dữ liệu.

Việc tổ chức các hệ phân tán có thể được nghiên cứu dựa theo ba trục không gian

- *Mức độ chia sẻ dữ liệu* (level of sharing)
- *Kiểu mẫu truy xuất* (behavior of access pattern)
- *Mức độ hiểu biết về kiểu mẫu truy xuất*

(1) Theo mức độ chia sẻ có ba khả năng xảy ra:

- *Không chia sẻ dữ liệu*: mỗi ứng dụng và dữ liệu của nó thực thi tại một vị trí, không có trao đổi hoặc giao tiếp với những chương trình khác hoặc truy xuất dữ liệu ở những vị trí khác. Hình thức này đặc trưng cho các kết nối mạng ở thời kỳ sơ khai.
- *Chia sẻ dữ liệu*: tất cả chương trình đều được nhân bản cho mỗi vị trí, nhưng không nhân bản dữ liệu. Theo đấy các yêu cầu của người dùng được xử lý tại mỗi vị trí và dữ liệu cần thiết được chuyển đi trên mạng.
- *Chia sẻ dữ liệu – chương trình*: cả chương trình và dữ liệu được dùng chung. Nghĩa là chương trình nằm tại một vị trí có thể yêu cầu dịch vụ từ một chương trình nằm ở vị trí thứ hai, và đến lượt nó, chương trình này có thể truy xuất dữ liệu nằm tại vị trí thứ ba.

Ở đây cần phân biệt giữa *Chia sẻ dữ liệu* và *Chia sẻ dữ liệu - chương trình*, đặc biệt đối với hệ phân tán đa chủng. Trong môi trường đa chủng rất khó khăn, có khi không thể được, cho thực thi một chương trình trên một phần cứng khác và trong hệ điều hành khác.

(2) Theo kiểu mẫu truy xuất có hai kiểu lựa chọn:

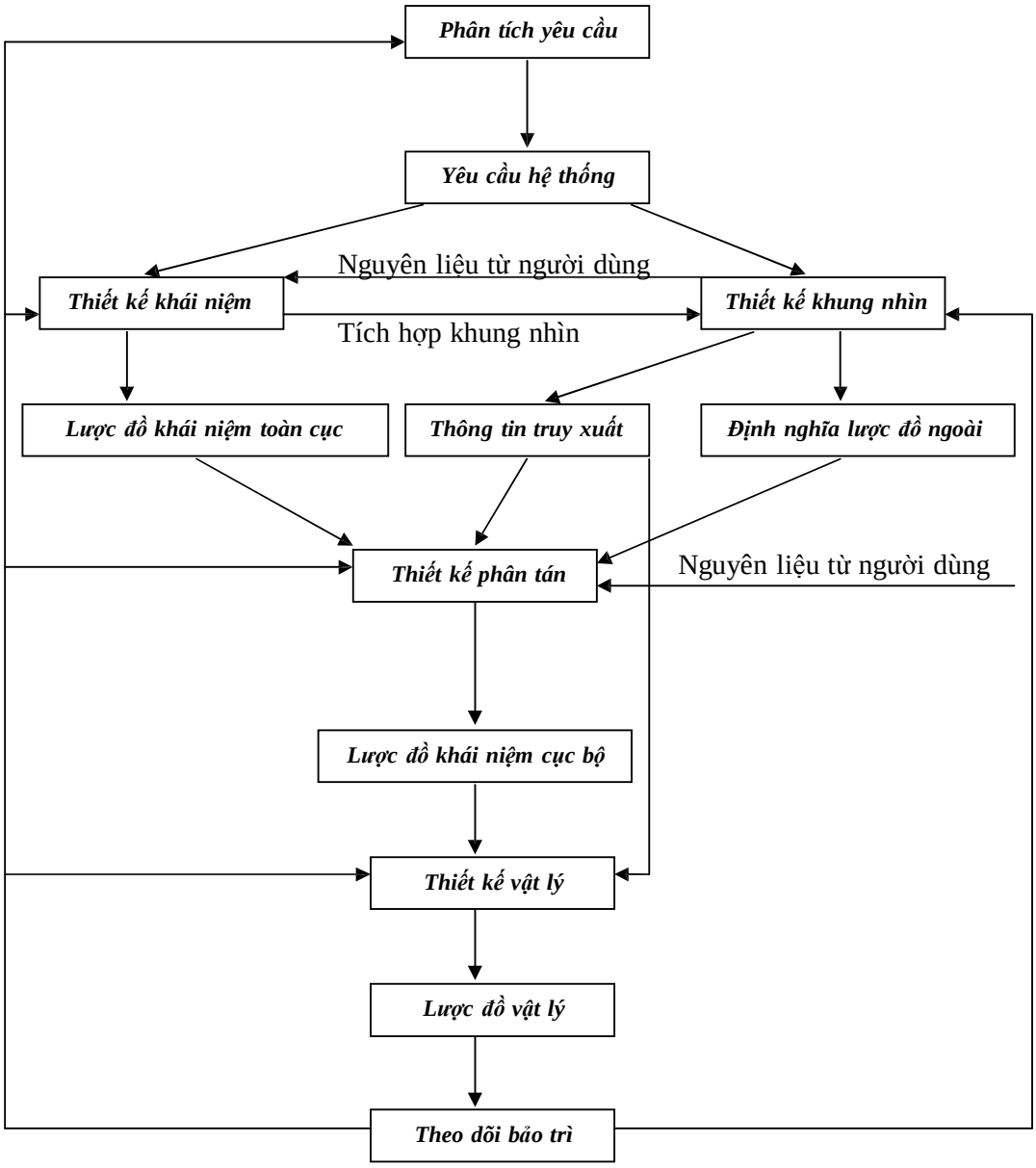
- Loại *tĩnh*, không thay đổi theo thời gian
- Loại *động*, thay đổi theo thời gian.

2.1. Các chiến lược thiết kế

Hai chiến lược chính trong việc thiết kế cơ sở dữ liệu phân tán là *tiếp cận từ trên xuống* và *tiếp cận từ dưới lên*. Trong thực tế rất hiếm các ứng dụng đơn giản để chỉ sử dụng một cách tiếp cận, vì vậy trong phần lớn thiết kế cả hai cách tiếp cần đều được áp dụng bổ sung nhau.

2.1.1. Quá trình thiết kế từ trên xuống

Sơ đồ quá trình thiết kế từ trên xuống biểu diễn trong hình sau:



Việc *phân tích yêu cầu* nhằm định nghĩa môi trường hệ thống và thu nhập các nhu cầu xử lý của tất cả người dùng, đồng thời cũng xác định *yêu cầu hệ thống.*

Hồ sơ ghi chép các yêu cầu là nguyên liệu cho hai hoạt động song song: *Thiết kế khung nhìn* (view design) và *Thiết kế khái niệm* (conceptual design).

Thiết kế khung nhìn định nghĩa các giao diện cho *người dùng đầu cuối* (end-user).

Thiết kế khái niệm là quá trình xem xét tổng thể đối tượng – xí nghiệp, nhằm xác định các loại thực thể và mối liên hệ giữa chúng với nhau. Ta có thể chia quá trình này thành 2 nhóm bao gồm các hoạt động liên quan tới nhau: *Phân tích thực*

thể (entity analysis) và *Phân tích chức năng* (functional analysis). Phân tích thực thể có liên quan đến việc xác định các thực thể, các thuộc tính và các mối liên hệ giữa chúng. Phân tích chức năng đề cập đến việc xác định các chức năng cơ bản có liên quan đến xí nghiệp cần được mô hình hoá. Kết quả của hai quá trình này cần được đối chiếu qua lại, giúp chúng ta biết được chức năng nào sẽ hoạt tác trên những thực thể nào.

Có sự liên hệ khăng khít giữa thiết kế khái niệm và thiết kế khung nhìn. Theo nghĩa nào đó thiết kế khái niệm được coi như là sự tích hợp các khung nhìn. Tuy nhiên mô hình khái niệm cần phải hỗ trợ không chỉ những ứng dụng hiện có mà còn cả những ứng dụng trong tương lai. Tích hợp khung nhìn nhằm đảm bảo các yêu cầu về thực thể và các mối liên hệ giữa các khung nhìn đều phải được bao quát trong lược đồ khái niệm.

Trong các hoạt động thiết kế khái niệm và thiết kế khung nhìn, người thiết kế cần phải đặc tả các thực thể dữ liệu và phải xác định các ứng dụng chạy trên cơ sở dữ liệu cũng như các thông tin thống kê về những ứng dụng này. Thông tin thống kê bao gồm đặc tả về tần số ứng dụng, khối lượng thông tin khác nhau,...

Lược đồ khái niệm toàn cục GCS và thông tin về kiểu mẫu truy xuất thu được trong thiết kế khung nhìn sẽ là nguyên liệu (input) cho bước *thiết kế phân tán*. Mục tiêu của giai đoạn này là thiết kế các lược đồ khái niệm cục bộ LCS bằng cách phân tán các thực thể cho các vị trí của hệ thống phân tán.

Ta chia quan hệ thành nhiều quan hệ nhỏ hơn gọi là các *mảnh* (*fragment*) và phân tán các mảnh này. Hoạt động thiết kế phân tán gồm hai bước: *Phân mảnh* (*fragmentation*) và *cấp phát* (*allocation*). Ta sẽ thảo luận về vấn đề này trong các phần sau.

Thiết kế vật lý là ánh xạ lược đồ khái niệm cục bộ sang các thiết bị lưu trữ vật lý có sẵn tại các vị trí tương ứng. Nguyên liệu cho quá trình này là lược đồ khái niệm cục bộ và thông tin về kiểu mẫu truy xuất các mảnh.

Hoạt động phát triển và thiết kế luôn là quá trình liên tục, đòi hỏi theo dõi hiệu chỉnh thường xuyên. Vì thế chúng ta đưa vấn đề quan sát và theo dõi như một hoạt động chính trong quá trình này. Cần chú ý rằng chúng ta không chỉ theo dõi vấn đề cài đặt cơ sở dữ liệu, mà còn quan sát theo dõi tính thích hợp của các khung nhìn của người dùng. Kết quả này có tác dụng phản hồi, tạo cơ sở cho việc tái thiết kế về sau.

2.1.2. Quá trình thiết kế từ dưới lên

Thiết kế từ trên xuống thích hợp cho những cơ sở dữ liệu được thiết kế từ đầu. Tuy nhiên trong thực tế cũng có khi đã có sẵn một số cơ sở dữ liệu, và chúng ta phải tích hợp chúng thành một cơ sở dữ liệu chung. Tiếp cận từ dưới lên sẽ thích hợp cho tình huống này. Khởi điểm của thiết kế từ dưới lên là các lược đồ khái niệm cục bộ, sẽ phải được tích hợp thành lược đồ khái niệm toàn cục.

2.2. Các vấn đề thiết kế phân tán

Trong mục này chúng ta sẽ trả lời các câu hỏi sau:

- Tại sao cần phân mảnh
- Làm thế nào để thực hiện phân mảnh
- Phân mảnh nên thực hiện đến mức độ nào
- Cách thức kiểm tra tính đúng đắn của phân mảnh
- Cách thức cấp phát dữ liệu
- Những thông tin nào cần thiết cho phân mảnh và cấp phát

2.2.1. Các lý do phân mảnh

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

Thứ hai là, nếu các ứng dụng có các khung nhìn được định nghĩa trên một quan hệ cho trước lại nằm tại những vị trí khác nhau thì chỉ có hai cách chọn lựa với đơn vị phân tán là toàn bộ quan hệ, khi không có phân mảnh: Hoặc quan hệ không được nhân bản mà được lưu ở một vị trí, hoặc quan hệ được nhân bản cho tất cả hoặc một số vị trí có chạy ứng dụng. Chọn lựa đầu gây ra một số lượng lớn truy xuất không cần thiết đến dữ liệu ở xa. Còn chọn lựa sau có thể dẫn đến nhân bản không cần thiết, gây khó khăn khi cập nhật và lãng phí không gian lưu trữ.

Cuối cùng việc phân rã quan hệ thành nhiều mảnh, mỗi mảnh xử lý như một đơn vị, sẽ cho phép thực hiện nhiều giao dịch đồng thời. Việc phân mảnh quan hệ cho phép thực hiện song song câu vấn tin, bằng cách chia nó thành một tập câu vấn tin con hoạt tác trên các mảnh. Vì thế việc phân mảnh sẽ làm tăng mức độ hoạt động đồng thời và kéo theo tăng lưu lượng hoạt động của hệ thống. Kiểu hoạt động này gọi là *đồng thời nội vấn tin (intraquery concurency)*, sẽ được phân tích trong các phần sau.

2.2.2. Các kiểu phân mảnh

Có hai kiểu phân mảnh: *phân mảnh theo chiều dọc* và *phân mảnh theo chiều ngang*.

◊ *Ví dụ*

Chúng ta sử dụng lược đồ cơ sở dữ liệu đã phát triển trong chương trước. Ta thêm vào lược đồ PROJ thuộc tính LOC (vị trí) để chỉ nơi thực hiện dự án. Sau đây là một thể hiện cơ sở dữ liệu sẽ được dùng:

EMP

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

ASG

ENO	*PNO*	*RESP*	*DUR*
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

PROJ

PNO	*PNAME*	*BUDGET*	*LOC*
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

PAY

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000

Trong hình sau trình bày quan hệ PROJ được tách ngang thành 2 quan hệ $PROJ_1$ chứa các thông tin về dự án có kinh phí dưới 200000 USD, và $PROJ_2$ chứa các thông tin về dự án có kinh phí lớn hơn 200000 USD.

$PROJ_1$

PNO	*PNAME*	*BUDGET*	*LOC*
P1	Instrumentation	150000	Montreal
P2	Database Develop.	135000	New York

$PROJ_2$

PNO	*PNAME*	*BUDGET*	*LOC*
P3	CAD/CAM	250000	New York
P4	Maintenance	310000	Paris

Còn trong hình sau trình bày quan hệ PROJ được tách dọc thành 2 quan hệ $PROJ_1$ chứa các thông tin về kinh phí dự án, và $PROJ_2$ chứa các thông tin về tên và vị trí dự án.

$PROJ_1$

PNO	*BUDGET*
P1	150000
P2	135000
P3	250000
P4	310000

$PROJ_2$

PNO	*PNAME*	*LOC*
P1	Instrumentation	Montreal
P2	Database Develop.	New York
P3	CAD/CAm	New York
P4	Maintenamce	Paris

Việc phân mảnh có thể lồng ghép, vừa phân mảnh ngang vừa phân mảnh dọc, thành *phân mảnh tổng hợp* (*hybrid fragmentation*).

2.2.3. Các qui tắc phân mảnh đúng đắn

Có ba qui tắc trong khi phân mảnh đảm bảo cơ sở dữ liệu sẽ không thay đổi ngữ nghĩa khi phân mảnh.

1) *Tính đầy đủ* (*completeness*): Cho quan hệ **r** bất kỳ. Giả sử **r** được phân rã thành các mảnh. Khi đó tính đầy đủ yêu cầu mỗi mục dữ liệu trong **r** cũng phải được lưu trữ trong một hoặc vài mảnh nào đó.

2) *Tính tái thiết* (*reconstruction*): Cho quan hệ **r** bất kỳ. Giả sử **r** được phân rã thành các mảnh $\mathbf{r_1}$, ..., $\mathbf{r_n}$. Khi đó tính tái thiết yêu cầu "hợp" các phân mảnh của quan hệ **r** trả lại đầy đủ dữ liệu ban đầu của quan hệ **r**. Khái niệm "hợp" ở đây là toán tử quan hệ Δ sao cho

$$\mathbf{r} = \Delta_{i=1}^{n} r_i$$

Toán tử Δ thay đổi tuỳ theo từng loại phân mảnh.

Khả năng tái thiết một quan hệ từ các mảnh của nó cũng phải đảm bảo rằng các ràng buộc định nghĩa theo phụ thuộc dữ liệu sẽ được bảo toàn.

3) *Tính tách biệt* (*disjointness*): Cho quan hệ **r** bất kỳ. Giả sử **r** được phân rã thành các mảnh $\mathbf{r_1}$, ..., $\mathbf{r_n}$. Khi đó tính tách biệt yêu cầu một mục dữ liệu *d* nào đó một khi đã xuất hiện trong mảnh $\mathbf{r_i}$ thì sẽ không xuất hiện trong mảnh $\mathbf{r_k}$ khác. Tiêu chuẩn này đảm bảo các mảnh ngang sẽ tách biệt nhau. Còn trong phân mảnh dọc thì các thuộc tính khoá chính phải được lặp lại trong mỗi mảnh, vì vậy tính tách biệt chỉ áp dụng với các thuộc tính không khoá.

2.2.4. Các kiểu cấp phát

Giả sử cơ sở dữ liệu đã được phân mảnh thích hợp và cần phải quyết định cấp phát các mảnh cho các vị trí trên mạng. Khi dữ liệu được cấp phát, nó có thể được nhân bản hoặc chỉ duy trì một bản duy nhất.

Một cơ sở dữ liệu không nhân bản, gọi là *cơ sở dữ liệu phân hoạch*, có chứa các mảnh được cấp phát cho các vị trí, trong đó chỉ tồn tại một bản duy nhất cho mỗi mảnh trên mạng.

Kiểu *cơ sở dữ liệu nhân bản* có hai dạng:

- *Cơ sở dữ liệu nhân bản hoàn toàn*, trong đó toàn bộ cơ sở dữ liệu đều có bản sao ở mỗi vị trí.
- *Cơ sở dữ liệu nhân bản một phần*, trong đó các mảnh được phân tán đến các vị trí, mỗi mảnh có thể có nhiều bản sao nằm ở các vị trí khác nhau. Số lượng các bản sao của các mảnh có thể là tham số (input) cho các thuật toán cấp phát (allocation algorithm) hoặc là biến quyết định (dicision variable) mà giá trị của nó được xác định bằng thuật toán này.

Lý do nhân bản là nhằm đảm bảo được độ tin cậy và hiệu quả cho các câu vấn tin chỉ đọc. Nếu có nhiều bản sao của một mục dữ liệu thì chúng ta vẫn có cơ hội truy xuất được dữ liệu đó ngay cả khi hệ thống có sự cố. Hơn nữa các câu vấn tin chỉ đọc truy xuất đến cùng một mục dữ liệu có thể cho thực hiện song song vì các bản sao có mặt tại nhiều vị trí.

Ngược lại, trong cơ sở dữ liệu nhân bản các câu vấn tin cập nhật có thể gây nhiều rắc rối vì phải đảm bảo các bản sao phải được cập nhật chính xác.

Vì vậy quyết định nhân bản cần được cân nhắc và phụ thuộc vào tỉ lệ giữa các câu vấn tin chỉ đọc và câu vấn tin cập nhật. Quyết định này hầu như ảnh hưởng đến tất cả các thuật toán của hệ quản trị cơ sở dữ liệu phân tán và các chức năng kiểm soát khác.

2.3. Phương pháp phân mảnh

Trong phần này chúng ta sẽ bàn đến các chiến lược và thuật toán phân mảnh. Có hai chiến lược phân mảnh cơ bản: *phân mảnh ngang* (*horizontal fragmentation*) và *phân mảnh dọc* (*vertical fragmentation*). Ngoài ra còn có khả năng phân mảnh hỗn hợp.

2.3.1. Phân mảnh ngang

Phân mảnh ngang chia quan hệ theo các bộ. Mỗi mảnh là một tập con của quan hệ. Có hai loại phân mảnh ngang: *phân mảnh nguyên thuỷ* (*primary horizontal fragmentation*), thực hiện dựa trên các vị từ định nghĩa trên chính

quan hệ đó, và *phân mảnh dẫn xuất* (*derived horizontal fragmentation*), dựa trên các vị từ định nghĩa trên quan hệ khác.

Trước khi thực hiện phân mảnh, chúng ta cần thu thập thông tin cần thiết.

a) Yêu cầu thông tin

• *Thông tin về cơ sở dữ liệu*

Thông tin này bao gồm lược đồ khái niệm toàn cục, các liên kết giữa các quan hệ, đặc biệt là phép nối.

Trong mô hình quan hệ, các mối liên hệ được biểu thị bằng các quan hệ. Tuy nhiên trong các mô hình khác, như mô hình thực thể-quan hệ, các mối liên hệ được biểu diễn tường minh. Với mục đích thiết kế phân tán, các mối liên hệ cũng được mô hình hoá trong bộ khung quan hệ. Theo cách này chúng ta sẽ vẽ các đường nối (L) có hướng giữa các quan hệ (R, S) ràng buộc nhau qua phép đẳng nối dạng

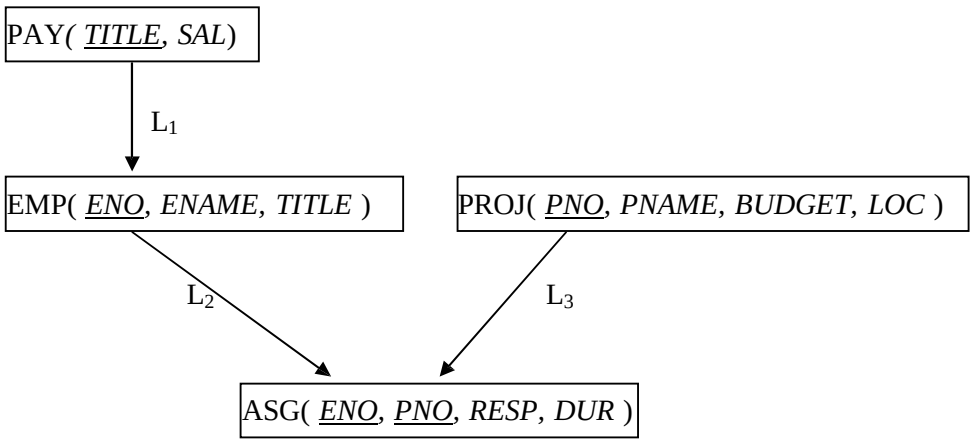
$$R \xrightarrow{L} S$$

trong đó R gọi là quan hệ chủ, S gọi là quan hệ thành viên. Người ta dùng hàm *owner* và *member* để phân biệt các quan hệ này:

$$owner(L) = R \quad \text{và} \quad member(L) = S$$

◊ *Ví dụ*

Hình sau trình bày một cách biểu diễn các đường nối giữa các quan hệ PAY, EMP, PROJ và ASG.



ở đây có ba đường nối L_1, L_2, L_3. Ta có

$owner(L_1) = PAY$ và $member(L_1) = EMP$

$owner(L_2) = EMP$ và $member(L_2) = ASG$

$owner(L_3) = PROJ$ và $member(L_3) = ASG$

Thông tin định lượng về cơ sở dữ liệu, tức là *lực lượng* (*cardinality*) của mỗi quan hệ R, ký hiệu *card*(R).

• *Thông tin về ứng dụng*

Yêu cầu thông tin định tính lẫn định lượng về các ứng dụng. Thông tin định tính định hướng cho hoạt động phân mảnh, còn thông tin định lượng sử dụng cho các mô hình cấp phát.

Những thông tin định tính cơ bản gồm các vị từ được dùng trong các câu vấn tin. Nếu không thể phân tích được hết tất cả các ứng dụng để xác định những vị từ này thì ít nhất cũng phải nghiên cứu được các ứng dụng quan trọng nhất. Một hướng dẫn quan trọng, gọi là *qui tắc 80/20*, là "*20% câu vấn tin sẽ chiếm đến 80% truy xuất dữ liệu*".

Cho quan hệ $R(A_1, ..., A_n)$, trongđó A_i là một thuộc tính được định nghĩa trên miền giá trị D_i. Một *vị từ đơn giản* p được định nghĩa trên R có dạng:

$$p: A_i \, \theta \; Value$$

Trong đó $\theta \in \{ =, <, \leq, \neq, >, \geq \}$ và *Value* là giá trị được chọn từ miền D_i.

◊ *Ví dụ*

Xét bảng quan hệ PROJ. Các biểu thức

$$PNAME = \text{"Maintenance"} \quad \text{và} \quad BUDGET \leq 200000$$

là các vị từ đơn giản.

Các câu vấn tin thường chứa nhiều vị từ phức tạp, là tổ hợp các vị từ đơn giản. Một tổ hợp cần đặc biệt chú ý được gọi là *tiểu hạng* (minterm predicate), là *hội* (conjuction) của các vị từ đơn giản. Bởi vì ta luôn có thể biến đổi một biểu thức bool thành *dạng chuẩn hội* (conjuctive normal form), việc sử dụng tiểu hạng trong thuật toán thiết kế không làm mất tính tổng quát.

Cho một tập các vị từ đơn giản $Pr = \{p_1, ..., p_m\}$ trên quan hệ R. Tập các tiểu hạng $M=\{m_1, ..., m_z\}$ gồm các vị từ dạng

$$m_j = \bigwedge_{p_k \in Pr} p_k^* \text{ , với } 1 \leq k \leq m, \; 1 \leq j \leq z$$

trong đó $p_k^* = p_k$ hoặc $p_k^* = \neg p_k$, $\forall k=1,...,m$. Như vậy mỗi vị từ đơn giản có thể xuất hiện trong vị từ tiểu hạng dạng khẳng định hoặc phủ định.

◊ *Ví dụ*

Xét bảng quan hệ PAY. Sau đây là các vị từ đơn giản định nghĩa trên PAY.

p_1 : *TITLE* = "Elect. Eng."
p_2 : *TITLE* = "Syst. Anal."
p_3 : *TITLE* = "Mech. Eng."
p_4 : *TITLE* = "Programmer"
p_5 : *SAL* ≤ 30000
p_6 : *SAL* > 30000

Từ các vị từ đơn giản trên có thể định nghĩa các vị từ tiểu hạng sau

m_1 : *TITLE* = "Elect. Eng." ∧ *SAL* ≤ 30000
m_2 : *TITLE* = "Elect. Eng." ∧ *SAL* > 30000
m_3 : ¬(*TITLE* = "Elect. Eng.") ∧ *SAL* ≤ 30000
m_4 : ¬(*TITLE* = "Elect. Eng.") ∧ *SAL* > 30000
m_5 : *TITLE* = "Programmer" ∧ *SAL* ≤ 30000
m_6 : *TITLE* = "Programmer" ∧ *SAL* > 30000

Theo những thông tin định lượng về các ứng dụng, chúng ta cần biết hai tập dữ liệu.

(1) *Độ tuyển tiểu hạng* (*minterm selectivity*): Số lượng các bộ quan hệ sẽ được truy xuất bởi câu vấn tin được đặc tả theo một vị từ tiểu hạng đã cho. Ta ký hiệu độ tuyển của vị từ tiểu hạng *m* là *sel(m).*

(2) *Tần số truy xuất* (*access frequency*): tần số ứng dụng truy xuất dữ liệu. Cho

$$Q = \{ q_1, q_2, ..., q_k \}$$

là tập các câu vấn tin, ký hiệu $acc(q_i)$ biểu thị tần số truy xuất của q_i trong một khoảng thời gian đã cho. Ta cũng ký hiệu tần số truy xuất của một vị từ tiểu hạng *m* là *acc(m).*

b) Phân mảnh ngang nguyên thuỷ

Phân mảnh ngang nguyên thuỷ được định nghĩa bằng một phép toán chọn trên các quan hệ chủ của một lược đồ CSDL.

Cho quan hệ **r**, các mảnh ngang của **r** là các quan hệ con r_i, i = 1, ..., k, với

$$\mathbf{r_i} = \sigma_{F_i}(r), \text{ i = 1, ..., k}$$

trong đó F_i , i = 1, ..., k, là công thức chọn để có mảnh $\mathbf{r_i}$.

◊ *Ví dụ*

Phân rã quan hệ PROJ thành các mảnh ngang $PROJ_1$ và $PROJ_2$ trong ví dụ trên có thể được định nghĩa như sau:

$$PROJ_1 = \sigma_{BUDGET \le 200000}(PROJ)$$
$$PROJ_2 = \sigma_{BUDGET > 200000}(PROJ)$$

Ta cũng có thể định nghĩa các mảnh ngang sau đây

$$PRJ_1 = \sigma_{LOC="Montreal"}(PROJ)$$
$$PRJ_2 = \sigma_{LOC="New\ York"}(PROJ)$$
$$PRJ_3 = \sigma_{LOC="Paris"}(PROJ)$$

PRJ_1

PNO	*PNAME*	*BUDGET*	*LOC*
P1	Instrumentation	150000	Montreal

PRJ_2

PNO	*PNAME*	*BUDGET*	*LOC*
P2	Database Develop.	135000	New York
P3	CAD/CAM	250000	New York

PRJ_3

PNO	*PNAME*	*BUDGET*	*LOC*
P4	Maintenamce	310000	Paris

Bây giờ ta có thể định nghĩa một mảnh ngang chặt chẽ hơn. Một mảnh ngang $\mathbf{r_i}$ của quan hệ **r** có chứa tất cả các bộ của **r** thoả vị từ tiểu hạng m_i. Như vậy, cho tập M các vị từ tiểu hạng, số mảnh ngang bằng số các vị từ tiểu hạng trong tập M. Tập các mảnh ngang này gọi là tập các *mảnh tiểu hạng* (minterm fragment).

Theo như đã phân tích, việc định nghĩa các mảnh ngang phụ thuộc vào các vị từ tiểu hạng. Vì thế bước đầu tiên của mọi thuật toán phân mảnh là xác định tập các vị từ đơn giản sẽ cấu thành các vị từ tiểu hạng.

Các vị từ đơn giản cần có các tính chất *đầy đủ* và *cực tiểu*.

• *Tính đầy đủ*. Tập các vị từ đơn giản *Pr* gọi là *đầy đủ* nếu và chỉ nếu xác suất mỗi ứng dụng truy xuất đến một bộ bất kỳ thuộc về một mảnh tiểu hạng nào đó được định nghĩa theo *Pr* đều bằng nhau.

◊ *Ví dụ*. Xét phân mảnh PRJ_1, PRJ_2, PRJ_3 ở ví dụ trước. Nếu ứng dụng duy nhất truy xuất PROJ muốn truy xuất các bộ theo vị trí, thì tập vị từ (ứng dụng)

$$Pr = \{LOC=\text{"Montreal"}, LOC=\text{"New York"}, LOC=\text{"Paris"}\}$$

là đầy đủ bởi vì mỗi bộ của mỗi mảnh PRJ_i đều có xác suất truy xuất như nhau.

Tuy nhiên, nếu có ứng dụng thứ hai chỉ truy xuất các bộ dự án có ngân sách nhỏ hơn 200000 USD, thì tập vị từ *Pr* không còn đầy đủ nữa. Một số bản ghi trong mỗi mảnh $PROJ_i$ được ứng dụng thứ hai truy xuất với xác suất cao hơn.

Để cho tập vị từ đầy đủ ta phải thêm các vị từ $BUDGET \leq 200000$, $BUDGET > 200000$ vào *Pr*, tức là

$$Pr = \{LOC=\text{"Montreal"}, LOC=\text{"New York"}, LOC=\text{"Paris"}, BUDGET \leq 200000, BUDGET > 200000\}$$

Lý do cần phải đảm bảo tính đầy đủ là vì các mảnh thu được theo tập vị từ đầy đủ sẽ nhất quán về mặt logic do tất cả chúng đều thoả vị từ tiểu hạng. Chúng cũng đồng nhất về mặt thống kê theo cách mà các ứng dụng truy xuất chúng. Vì vậy chúng ta sẽ sử dụng tập vị từ đầy đủ làm cơ sở cho phân mảnh ngang nguyên thủy.

• *Tính cực tiểu*

Cho tập các vị từ đơn giản *Pr*. Ta nói một vị từ là *có liên đới* (*relevant*) trong việc xác định phân mảnh, nếu nó ảnh hưởng đến việc mảnh f nào đó bị phân thành các mảnh con f_1 và f_2 thì phải có ít nhất một ứng dụng truy xuất đến f_1 và f_2 theo cách khác nhau.

Tập *Pr* gọi là *cực tiểu* nếu mọi vị từ trong nó là *có liên đới* và không có vị từ tương đương.

◊ *Ví dụ*. Tập vị từ *Pr* trong ví dụ trước là đầy đủ và cực tiểu. Tuy nhiên nếu chúng ta thêm vị từ

$$PNAME=\text{"Instrumentation"}$$

vào *Pr*, thì *Pr* không còn là cực tiểu nữa bởi vì vị từ trên không có liên đới ứng với *Pr*. Không có ứng dụng truy xuất khác nhau đến các mảnh ảnh hưởng bởi vị từ trên .

Bây giờ chúng ta sẽ trình bày một thuật toán lặp sinh ra một tập các vị từ đầy đủ và cực tiểu. Ta sẽ sử dụng qui tắc cơ bản về tính đầy đủ và cực tiểu gọi tắt là qui tắc 1.

♦ *Qui tắc 1*. Một quan hệ hoặc một mảnh được phân hoạch thành ít nhất hai phần và chúng được truy xuất khác nhau bởi một ứng dụng.

◊ Ký hiệu *f(p)* là mảnh sinh bởi vị từ *p* và *F* là tập các mảnh, f_k là mảnh sinh bởi vị từ p_k.

• ***Thuật toán: COM_MIN***

Đầu vào: Quan hệ **r** và tập các vị từ đơn giản *Pr*.
Đầu ra: Tập các vị từ *Pr'* đầy đủ và cực tiểu.
Khai báo: *F* là tập các mảnh tiểu hạng.
begin
Tìm vị từ $p \in Pr$ sao cho *p* phân hoạch **r** theo qui tắc 1
$Pr' \leftarrow p$
$Pr \leftarrow Pr - \{p\}$
$F \leftarrow f\{= f(p)\}$
repeat
begin
Tìm vị từ $p \in Pr$ sao cho *p* phân hoạch một mảnh f_k nào đó của *F* theo qui tắc 1 :
$Pr' \leftarrow Pr' \cup \{p\}$
$Pr \leftarrow Pr - \{p\}$
$F \leftarrow F \cup \{f\}$
if tồn tại $p_k \in Pr'$ không liên đới hoặc có vị từ tương đương **then**
begin
$Pr' \leftarrow Pr' - \{p_k\}$
$F \leftarrow F - \{f_k\}$
end
end
until *Pr'* đầy đủ
end. {COM_MIN}

Bước thứ hai trong quá trình thiết kế phân mảnh ngang nguyên thuỷ là suy dẫn ra các tập vị từ tiểu hạng có thể được định nghĩa trên các vị từ trong *Pr'*. Các vị từ tiểu hạng này xác định các mảnh "ứng cử viên" cho bước cấp phát.

Việc xác định các vị từ tiểu hạng không khó. Khó khăn chính là tập này rất lớn (thực sự chúng tỉ lệ hàm mũ theo số lượng vị từ đơn giản), cần phải giảm số lượng.

Bước thứ ba là loại bỏ một số mảnh vô nghĩa. Điều này được thực hiện bằng cách xác định những vị từ mâu thuẫn với tập các *phép kéo theo*, kí hiệu là I.

Chẳng hạn, nếu $Pr' = \{p_1, p_2\}$, trong đó

$$p_1 : att = value1 \quad \& \quad p_2 : att = value2$$

và miền giá trị của thuộc tính *att* là {*value1*,*value2*}. Như vậy sẽ phát sinh hai phép kéo theo:

$$i_1 : p_1 \Rightarrow \neg p_2 \qquad \text{và} \qquad i_2 : \neg p_1 \Rightarrow p_2$$

Bốn vị từ tiểu hạng của *Pr'* được định nghĩa như sau:

$$m_1 : (att=value1) \wedge (att=value2)\ m_2 : (att=value1) \wedge \neg(att=value2)\ m_3 : \neg(att=value1) \wedge (att=value2)\ m_4 : \neg(att=value1) \wedge \neg(att=value2)$$

Trong trường hợp này các vị từ tiểu hạng m_1 và m_4 mâu thuẫn với các phép kéo theo i_1 và i_2 , và vì thế chúng bị loại khỏi tập các vị từ tiểu hạng M.

Thuật toán phân mảnh ngang nguyên thuỷ được trình bày như sau:

- ***Thuật toán: PHORIZONTAL***

Đầu vào: Quan hệ **r** và tập các vị từ đơn giản *Pr*.
Đầu ra: Tập các vị từ tiểu hạng M.
begin
Đặt *Pr'*:=COM_MIN(**r**, *Pr*);
Xác định tập M các vị từ tiểu hạng;
Xác định tập I các phép kéo theo giữa các vị từ trong *Pr'*;
Loại vị từ tiểu hạng mâu thuẫn:
for mỗi vị từ tiểu hạng $m \in$ M do
if *m* mâu thuẫn với I **then**
M $\leftarrow$ M − {*m*} { loại *m* khỏi M}
End-if
End-for
End.

◊ *Ví dụ*

Xét thiết kế lược đồ CSDL: EMP, ASG, PROJ, PAY cho ở phần trên. Có hai quan hệ cần phân mảnh ngang nguyên thuỷ là PAY và PROJ.

Giả sử có một ứng dụng truy xuất PAY, ứng dụng này kiểm tra thông tin lương và xác định số lương sẽ tăng. Giả sử rằng các mẩu tin nhân viên được quản lý ở hai nơi. Một nơi xử lý các mẩu tin có lương ≤ 30000 USD, và nơi khác xử lý các mẩu tin của nhân viên có lương >30000 USD. Vì thế câu vấn tin được sử dụng ở cả hai nơi.

Tập vị từ đơn giản được sử dụng để phân hoạch PAY là

$$p_1 : \text{SAL} \leq 30000 \qquad \& \qquad p_2 : \text{SAL} > 30000$$

Từ đó ta có tập vị từ đơn giản khởi đầu là

$$Pr = \{p_1, p_2\}$$

Áp dụng thuật toán COM_MIN ta có tập khởi đầu *Pr*' = $\{p_1\}$. Đây là tập đầy đủ và cực tiểu vì p_2 không phân hoạch mảnh $f_1 = f(p_1)$ theo qui tắc 1. Chúng ta có thể tạo ra các vị từ tiểu hạng cho tập M:

$$m_1 : (\text{SAL} \leq 30000) = p_1 \quad \text{và} \quad m_2 : \neg(\text{SAL} \leq 30000) = (\text{SAL} > 30000) = p_2$$

Sau đó ta định nghĩa hai mảnh $F_{PAY} = \{\text{PAY}_1, \text{PAY}_2\}$ theo M:

PAY_1

TITLE	*SAL*
Mech.Eng.	27000
Programmer	24000

PAY_2

TITLE	*SAL*
Elect.Eng.	40000
Syst.Anal.	34000

Bây giờ ta xét quan hệ PROJ. Giả sử có hai ứng dụng.

Ứng dụng thứ nhất được đưa ra tại ba vị trí và cần tìm tên và ngân sách của các dự án khi biết vị trí. Theo ký pháp SQL, câu vấn tin được viết là

```
SELECT   PNAME, BUDGET
FROM     PROJ
WHERE    LOC = Value
```

Đối với ứng dụng này, các vị từ đơn giản có thể dùng là

$$p_1 : LOC = \text{"Montreal"}, \quad p_2 : LOC = \text{"New York"}, \quad p_3 : LOC = \text{"Paris"}$$

Ứng dụng thứ hai được đưa ra tại hai vị trí và liên quan tới việc điều hành dự án. Những dự án có ngân sách ≤ 200000 USD được quản lý tại một vị trí, còn những dự án có ngân sách > 200000 USD được quản lý tại vị trí thứ hai. Vì thế các vị từ đơn giản phải được sử dụng để phân mảnh theo ứng dụng thứ hai là

$$p_4 : BUDGET \leq 200000 \quad \text{và} \quad p_5 : BUDGET > 200000$$

Nếu kiểm tra bằng thuật toán COM_MIN, tập

$$\text{Pr'} = \{p_1, p_2, p_3, p_4, p_5\}$$

rõ ràng là đầy đủ và cực tiểu.

Dựa trên *Pr*' chúng ta có thể định nghĩa sáu vị từ tiểu hạng sau tạo nên M.

$$m_1 : (LOC = \text{"Montreal"}) \wedge (BUDGET \leq 200000)$$
$$m_2 : (LOC = \text{"Montreal"}) \wedge (BUDGET > 200000)$$

m_3 : (*LOC* = "New York") $\wedge$ (*BUDGET* $\leq$ 200000)
m_4 : (*LOC* = "New York") $\wedge$ (*BUDGET* > 200000)
m_5 : (*LOC* = "Paris") $\wedge$ (*BUDGET* $\leq$ 200000)
m_6 : (*LOC* = "Paris") $\wedge$ (*BUDGET* > 200000)

Đây không phải là tất cả các vị từ tiểu hạng có thể được tạo ra. Chẳng hạn có thể định nghĩa vị từ

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

Tuy nhiên, các phép kéo theo hiển nhiên là

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 : $\neg p_5 \Rightarrow p_4$

cho phép loại bỏ những vị từ tiểu hạng khác và chúng ta còn lại m_1 đến m_6.
Kết quả của phân mảnh ngang nguyên thuỷ cho PROJ là tạo ra sáu mảnh

$$F_{PROJ} = \{PROJ_1, PROJ_2, PROJ_3, PROJ_4, PROJ_5, PROJ_6\}$$

của quan hệ PROJ tương ứng theo các vị từ tiểu hạng m_1 đến m_6 trong M. Chú ý rằng các mảnh $PROJ_2$ và $PROJ_5$ rỗng.

$PROJ_1$

PNO	PNAME	BUDGET	LOC
P1	Instrumentation	150000	Montreal

$PROJ_3$

PNO	PNAME	BUDGET	LOC
P2	Database Develop.	135000	New York

$PROJ_4$

PNO	PNAME	BUDGET	LOC
P3	CAD/CAM	250000	New York

$PROJ_6$

PNO	PNAME	BUDGET	LOC
P4	Maintenance	310000	Paris

c) Phân mảnh ngang dẫn xuất

Phân mảnh ngang dẫn xuất được định nghĩa trên một quan hệ thành viên của một đường nối dựa theo phép toán chọn trên quan hệ chủ nhân của đường nối đó. Ta cần lưu ý hai điểm sau. Trước tiên đường nối giữa quan hệ chủ và quan hệ thành viên được định nghĩa bằng một phép đẳng nối. Thứ hai, đẳng nối có thể

được cài đặt nhờ các phép *bán nối*. Điểm này rất quan trọng vì ta muốn phân hoạch quan hệ thành viên theo phân mảnh của quan hệ chủ, nhưng cũng muốn mảnh thu được chỉ định nghĩa trên các thuộc tính của quan hệ thành viên.

Muốn thực hiện phân mảnh ngang dẫn xuất, ta cần ba yếu tố đầu vào: tập các phân hoạch của quan hệ chủ, quan hệ thành viên, và tập các vị từ bán nối giữa quan hệ chủ và quan hệ thành viên.

Như thế, nếu cho trước đường nối L, trong đó *owner*(L)=S và *member*(L)=R, các mảnh ngang dẫn xuất của R được định nghĩa là

$$R_i = R >< S_i \text{ , } 1 \le i \le n$$

trong đó *n* là số lượng mảnh được định nghĩa trên R, và $S_i = \sigma_{F_i}(S)$ với F_i là công thức định nghĩa mảnh nguyên thuỷ S_i.

◊ *Ví dụ*. Xét đường nối L_1 trong ví dụ mục a). Ta có *owner*(L_1)=PAY và *member*(L_1)=EMP. Ta có thể nhóm các kỹ sư (engineer) thành hai nhóm tuỳ theo lương: nhóm có lương từ 30000 USD trở xuống và nhóm có lương từ 30000 USD trở lên. Hai mảnh EMP_1 và EMP_2 được định nghĩa như sau

$$EMP_1 = EMP >< PAY_1 \quad \text{và} \quad EMP_2 = EMP >< PAY_2$$

trong đó

$$PAY_1 = \sigma_{SAL \le 30000}(PAY) \quad \text{và} \quad PAY_2 = \sigma_{SAL > 30000}(PAY)$$

Kết quả được trình bày ở các bảng sau

EMP_1

ENO	*ENAME*	*TITLE*
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E7	R.David	Mech.Eng.

EMP_2

ENO	*ENAME*	*TITLE*
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E8	J.Jones	Syst.Anal.

Cũng có vấn đề phức tạp cần chú ý. Trong lược đồ CSDL chúng ta hay gặp nhiều đường nối đến quan hệ R (chẳng hạn có hai đường nối đến quan hệ ASG trong ví dụ trên). Như thế có thể có nhiều cách phân mảnh ngang dẫn xuất cho R. Quyết định chọn cách phân mảnh nào dựa trên 2 tiêu chuẩn.

(1) Phân mảnh nào có đặc tính nối tốt hơn.
(2) Phân mảnh nào được sử dụng trong nhiều ứng dụng hơn.

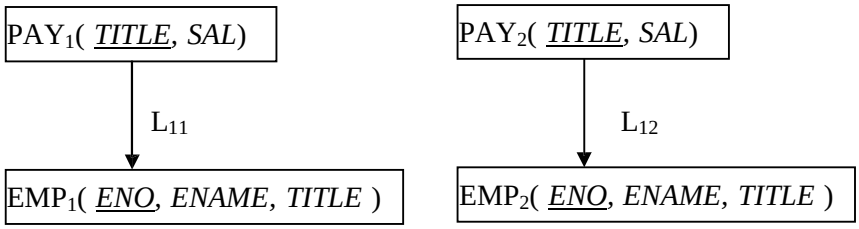
Chúng ta sẽ bàn về tiêu chuẩn thứ 2 trước. Tiêu chuẩn rất đơn giản nếu xét đến tần số truy xuất dữ liệu của các ứng dụng. Nếu được, ta nên ưu tiên những người dùng quan trọng để tác động của họ lên hiệu năng hệ thống là nhỏ nhất.

Tuy nhiên, việc áp dụng tiêu chuẩn thứ nhất không đơn giản. Chẳng hạn thử xét phân mảnh đã thảo luận trong ví dụ trước. Tác động (và mục tiêu) của phân mảnh này là nối của các quan hệ EMP và PAY để trả lời các câu vấn tin được hỗ

trợ (a) bằng cách thực hiện nó trên các quan hệ nhỏ hơn (tức các mảnh) và (b) bằng cách thực hiện các nối theo lối phân tán.

Điểm (a) là hiển nhiên. Các mảnh của EMP nhỏ hơn bản thân EMP. Vì thế khi nối một mảnh của PAY với một mảnh của EMP sẽ nhanh hơn là nối các quan hệ này.

Tuy nhiên điểm (b) lại quan trọng hơn và là trọng tâm của CSDL phân tán. Nếu ngoài việc thực hiện được nhiều câu vấn tin tại nhiều vị trí khác nhau, chúng ta có thể cho thực hiện song song một câu vấn tin, thời gian đáp ứng và lưu lượng hệ thống sẽ được cải thiện. Chẳng hạn xét đồ thị nối giữa các mảnh EMP và PAY lấy từ ví dụ trên. Mỗi mảnh chỉ có 1 đường nối đến hoặc đi khỏi.



Một đồ thị nối như thế gọi là *đồ thị đơn giản*. Thiết kế, trong đó mối liên hệ nối giữa các mảnh là đơn giản, có ưu điểm là quan hệ thành viên và quan hệ chủ có thể được cấp phát cho một vị trí, và các nối giữa các cặp mảnh khác nhau có thể tiến hành độc lập và song song.

Tuy nhiên, không phải lúc nào cũng thu được các đồ thị nối đơn giản. Khi đó thiết kế cần tạo ra *đồ thị nối phân hoạch*. Một đồ thị phân hoạch chứa 2 hoặc nhiều đồ thị con và không có đường nối giữa chúng. Các mảnh như thế không dễ phân tán để thực hiện song song như trong các đồ thị đơn giản, nhưng vẫn có thể cấp phát.

◊ *Ví dụ*. Xét tiếp ví dụ trên. Ta đã phân mảnh EMP theo phân mảnh của PAY. Bây giờ ta xét quan hệ ASG. Giả sử có hai ứng dụng sau:

(1) Ứng dụng thứ nhất tìm tên các kỹ sư có làm việc tại một nơi nào đó. Ứng dụng này chạy ở cả ba trạm và truy xuất thông tin về các kỹ sư làm việc trong các dự án tại chỗ với xác suất cao hơn các kỹ sư làm việc ở những vị trí khác.

(2) Ứng dụng thứ hai là tại mỗi trạm quản lý, nơi lưu các mẩu tin nhân viên, người dùng muốn truy xuất đến các dự án đang được các nhân viên này thực hiện và cần biết xem họ sẽ làm việc với dự án đó trong bao lâu.

Ứng dụng thứ nhất dẫn đến việc phân mảnh ASG theo các mảnh $PROJ_1$, $PROJ_3$, $PROJ_4$ và $PROJ_6$ của PROJ thu được trong ví dụ phân mảnh ngang nguyên thuỷ, trong đó

$$PROJ_1 = \sigma_{LOC=\text{"Montreal"} \wedge BUDGET \leq 200000}(PROJ)$$
$$PROJ_3 = \sigma_{LOC=\text{"New York"} \wedge BUDGET \leq 200000}(PROJ)$$
$$PROJ_4 = \sigma_{LOC=\text{"New York"} \wedge BUDGET > 200000}(PROJ)$$

$$PROJ_6 = \sigma_{LOC=\text{"Paris"} \wedge BUDGET > 200000}(PROJ)$$

Vì thế phân mảnh dẫn xuất của ASG theo $PROJ_1$, $PROJ_3$, $PROJ_4$ và $PROJ_6$ được định nghĩa như sau:

$$ASG_1 = ASG >< PROJ_1$$
$$ASG_2 = ASG >< PROJ_3$$
$$ASG_3 = ASG >< PROJ_4$$
$$ASG_4 = ASG >< PROJ_6$$

Thể hiện các mảnh này được trình bày ở các bảng sau:

ASG_1

ENO	*PNO*	*RESP*	*DUR*
E1	P1	Manager	12
E2	P1	Analyst	24

ASG_3

ENO	*PNO*	*RESP*	*DUR*
E3	P3	Consultant	10
E7	P3	Engineer	36
E8	P3	Manager	40

ASG_2

ENO	*PNO*	*RESP*	*DUR*
E2	P2	Analyst	6
E4	P2	Programmer	18
E5	P2	Manager	24

ASG_4

ENO	*PNO*	*RESP*	*DUR*
E3	P4	Engineer	48
E6	P4	Manager	48

Ứng dụng thứ hai là câu vấn tin được viết bằng SQL như sau:

```
SELECT   RESP, DUR
FROM     ASG, EMP_i
WHERE    ASG.ENO = EMP_i.ENO
```

trong đó *i* =1 hoặc *i* =2 tuỳ thuộc nơi đưa ra câu vấn tin. Phân mảnh dẫn xuất của ASG theo phân mảnh của EMP được định nghĩa dưới đây và cho ở bảng sau:

$$ASG_1 = ASG >< EMP_1$$
$$ASG_2 = ASG >< EMP_2$$

ASG_1

ENO	*PNO*	*RESP*	*DUR*
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E7	P3	Engineer	36

ASG_2

ENO	*PNO*	*RESP*	*DUR*
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E5	P2	Manager	24
E6	P4	Manager	48
E8	P3	Manager	40

Ví dụ này minh hoạ 2 điều:

(1) Phân mảnh dẫn xuất có thể xảy ra dây chuyền, trong đó một quan hệ được phân mảnh như hệ quả của phân mảnh quan hệ khác, và đến lượt nó làm cho các quan hệ khác phân mảnh (chẳng hạn như dây chuyền PAY→EMP→ASG).

(2) Thông thường một quan hệ có nhiều cách phân mảnh (chẳng hạn như ASG). Chọn một lược đồ phân mảnh là bài toán quyết định và sẽ được phân tích trong phần cấp phát.

d) Kiểm định tính đúng đắn

Bây giờ ta cần kiểm tra các tiêu chuẩn đối với các thuật toán.

• *Tính đầy đủ*

Tính đầy đủ của phân mảnh ngang nguyên thuỷ dựa vào các vị từ chọn được dùng. Với điều kiện các vị từ chọn là đầy đủ, phân mảnh thu được cũng đảm bảo đầy đủ. Bởi vì cơ sở của thuật toán phân mảnh là tập các vị từ cực tiểu và đầy đủ *Pr'*, tính đầy đủ sẽ được đảm bảo.

Tính đầy đủ của phân mảnh ngang dẫn xuất có phức tạp hơn. Trước tiên chúng ta định nghĩa lại qui tắc đầy đủ.

Cho R là quan hệ thành viên của đường nối với quan hệ chủ S. Quan hệ S được phân mảnh thành $F_S = \{S_1, ..., S_k\}$. Gọi A là thuộc tính nối giữa R và S. Ký hiệu $\{R_1, ..., R_k\}$ là tập phân mảnh dẫn xuất của R theo F_S. Khi đó với mỗi bộ t của R_i thì phải có 1 bộ t' của S_i thoả t[A]=t'[A], với mọi i =1, ..., k.

Tính đầy đủ ở đây thực chất là đảm bảo *ràng buộc toàn vẹn tham chiếu*.

• *Tính tái thiết*

Tái thiết một quan hệ toàn cục từ các mảnh được thực hiện bằng toán tử hợp trong cả phân mảnh ngang nguyên thuỷ lẫn dẫn xuất.

$$R = \mathop{\mathrm{U}}_{R_i \in F} R_i$$

• *Tính tách rời*

Trong phân mảnh ngang nguyên thuỷ tính tách rời được đảm bảo nếu các vị từ tiểu hạng xác định phân mảnh có tính loại trừ tương hỗ.

Tuy nhiên phân mảnh dẫn xuất có hàm chứa các bán nối có phức tạp hơn. Tính tách rời được đảm bảo nếu đồ thị nối thuộc loại đơn giản. Nếu đồ thị nối không đơn giản thì phải xem xét các giá trị thực sự của phân mảnh.

◊ *Ví dụ*. Khi phân mảnh quan hệ PAY ở ví dụ trước, các vị từ tiểu hạng $M=\{m_1,m_2\}$ là

$$m_1 = SAL \leq 30000$$
$$m_2 = SAL > 30000$$

Vì m_1 và m_2 loại trừ tương hỗ, phân mảnh của PAY là tách biệt.

Tuy nhiên, với quan hệ EMP ta đòi hỏi

(i) Mỗi kỹ sư có một chức vụ duy nhất.
(ii) Mỗi chức vụ có một giá trị lương duy nhất đi kèm.

Vì hai qui tắc này suy ra từ ngữ nghĩa của CSDL, phân mảnh của EMP ứng với PAY cũng tách rời.

2.3.2. Phân mảnh dọc

Một phân mảnh dọc của quan hệ R là tập các mảnh R_1, ..., R_k, trong đó mỗi mảnh chứa tập con thuộc tính của R và các khoá của R. Mục đích của phân mảnh dọc là phân hoạch một quan hệ thành tập các quan hệ nhỏ hơn để nhiều ứng dụng chỉ cần chạy trên 1 mảnh. Phân mảnh tối ưu cho phép giảm tối đa thời gian thực thi các ứng dụng chạy trên các mảnh đó.

Phân mảnh dọc phức tạp hơn so với phân mảnh ngang, do số khả năng phân mảnh dọc rất lớn. Trong phân mảnh ngang, nếu tổng số vị từ đơn giản trong *Pr* là *n* thì có 2^n vị từ tiểu hạng có thể định nghĩa trên nó. Ngoài ra có thể loại bỏ một số lớn trong đó mâu thuẫn với phép kéo theo hiện có, như vậy sẽ làm giảm đi số mảnh dự tuyển. Trong trường hợp phân mảnh dọc, nếu quan hệ có *m* thuộc tính không khoá, thì số mảnh có thể bằng *B*(*m*), *số Bell* thứ *m*. Với *m* lớn $B(m) \approx m^m$, chẳng hạn với $B(10) \approx 115000$, $B(15) \approx 10^9$, $B(30) \approx 10^{23}$.

Những giá trị này cho thấy, nỗ lực tìm lời giải tối ưu cho bài toán phân hoạch dọc không hiệu quả; vì thế phải dùng đến các phương pháp *heuristic*. Chúng ta nêu ở đây hai loại heuristic cho phân mảnh dọc các quan hệ toàn cục.

(1) *Nhóm thuộc tính*: Bắt đầu bằng cách gán mỗi thuộc tính cho một mảnh, và tại mỗi bước nối số mảnh lại cho đến khi thoả tiêu chuẩn nào đó.

(2) *Tách mảnh*: Bắt đầu bằng một quan hệ và quyết định cách phân hoạch có lợi dựa trên hành vi truy xuất của các ứng dụng trên các thuộc tính.

Trong phần tiếp chúng ta chỉ thảo luận kỹ thuật tách mảnh, vì nó thích hợp phương pháp thiết kế từ trên xuống hơn và giải pháp tối ưu gần với quan hệ đầy đủ hơn là tập các mảnh chỉ có 1 thuộc tính. Hơn nữa kỹ thuật tách mảnh sinh ra các mảnh với các thuộc tính không khoá không chồng nhau.

Việc nhân bản khoá chính cho các mảnh là đặc trưng của phân mảnh dọc, cho phép tái thiết quan hệ toàn cục. Vì thế phương pháp tách mảnh chỉ đề cập đến các thuộc tính không khoá.

Mặc dù có những vấn đề phát sinh, nhân bản các thuộc tính khoá có ưu điểm nổi bật là duy trì tính toàn vẹn ngữ nghĩa.

Một chọn lựa khác đối với nhân bản các thuộc tính khoá là sử dụng một *mã định bộ* (TID - tuple identifier). Đây là giá trị duy nhất được hệ thống gán cho mỗi bộ của quan hệ.

a) Các yêu cầu thông tin của phân mảnh dọc

Những thông tin chính cần cho phân mảnh dọc có liên quan tới các ứng dụng. Vì phân mảnh dọc đặt vào một mảnh các thuộc tính thường được truy xuất chung với nhau, ta cần xác định một độ đo khái niệm "chung với nhau". Số đo này gọi là *ái lực* (*affinity*) của thuộc tính, chỉ mức độ liên đới giữa các thuộc tính.

Thông số quan trọng về dữ liệu có liên quan đến các ứng dụng là *tần số truy xuất* (*access frequency*) của chúng. Gọi $Q = \{q_1, ..., q_k\}$ là tập các vấn tin của người dùng sẽ chạy trên quan hệ $R(A_1, ..., A_n)$. Với mỗi câu vấn tin q_i và mỗi thuộc tính A_j ta định nghĩa *giá trị sử dụng thuộc tính* (*attribute usage value*), ký hiệu $use(q_i, A_j)$, như sau

$$use(q_i, A_j) = \begin{cases} 1, \text{ nếu } q_i \text{ tham chiếu thuộc tính } A_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

Các vectơ $use(q_i, \bullet)$ cho mỗi ứng dụng sẽ được xác định nếu nhà thiết kế biết được các ứng dụng chạy trên cơ sở dữ liệu. Cần nhắc lại rằng *qui tắc 80/20* rất có ích cho công việc này.

◊ *Ví dụ.* Xét quan hệ PROJ của ví dụ phần trước. Giả sử các ứng dụng sau đây chạy trên quan hệ đó.

q_1 : Tìm ngân sách của dự án, cho biết mã số dự án.

```
SELECT   BUDGET
FROM     PROJ
WHERE    PNO = Value
```

q_2 : Tìm tên và ngân sách của tất cả dự án.

```
SELECT   PNAME, BUDGET
FROM     PROJ
```

q_3 : Tìm tên của các dự án được thực hiện ở thành phố đã cho

```
SELECT   PNAME
FROM     PROJ
WHERE    LOC = Value
```

q_4 : Tìm tổng ngân sách các dự án được thực hiện ở thành phố đã cho

```
SELECT   SUM(BUDGET)
FROM     PROJ
WHERE    LOC = Value
```

Dựa theo 4 ứng dụng này ta có thể xác định được giá trị sử dụng các thuộc tính. Để cho đơn giản ta ký hiệu

$$A_1 = PNO,\ A_2 = PNAME,\ A_3 = BUDGET \text{ và } A_4 = LOC.$$

Giá trị sử dụng cho ở ma trận sau (phần tử ô [i, j] chính là $use(q_i, A_j)$) :

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

Giá trị sử dụng thuộc tính không đủ thông tin để làm cơ sở cho việc tách và phân mảnh. Lý do là chúng không biểu thị độ lớn của tần số ứng dụng. Số đo tần số có thể được chứa trong định nghĩa về số đo ái lực thuộc tính $aff(A_i, A_j)$, biểu thị cho *cầu nối* (*bond*) giữa hai thuộc tính của một quan hệ theo cách chúng được các ứng dụng truy xuất.

Cho quan hệ R(A_1, ..., A_n) và tập ứng dụng Q={q_1, ..., q_k}. Với mỗi cặp thuộc tính (A_i, A_j) ký hiệu

S_l là vị trí thứ l, l = 1, ..., L

$ref_l(q_k)$ là số lần truy xuất đến A_i, A_j cho mỗi lần thực hiện ứng dụng q_k tại vị trí S_l.

$acc_l(q_k)$ là số đo tần số thực hiện ứng dụng q_k tại vị trí S_l.

$Q(i, j) = \{k \mid use(q_k, A_i) = 1 \ \& \ use(q_k, A_j) = 1\}$

Số đo ái lực thuộc tính giữa hai thuộc tính A_i và A_j của quan hệ R(A_1 , ..., A_n) ứng với tập ứng dụng Q={q_1, ..., q_k} được xác định theo công thức

$$aff(A_i, A_j) = \sum_{k \in Q(i,j)} \sum_{l=1}^{L} ref_l(q_k).acc_l(q_k)$$

Kết quả là ma trận vuông cấp n gọi là *ma trận ái lực thuộc tính* (*AA - attribute affinity matrix*).

◊ *Ví dụ.* Ta tiếp tục ví dụ trên. Để đơn giản ta giả sử rằng có 3 vị trí S_1, S_2 và S_3 và 4 ứng dụng và $ref_l(q_k)$ =1 với mọi S_l và mọi q_k. Cho tần số sử dụng như sau

$$\begin{array}{lll} acc_1(q_1) = 15; & acc_2(q_1) = 20; & acc_3(q_1) = 10; \\ acc_1(q_2) = 5; & acc_2(q_2) = 0; & acc_3(q_2) = 0; \\ acc_1(q_3) = 25; & acc_2(q_3) = 25; & acc_3(q_3) = 25; \\ acc_1(q_4) = 3; & acc_2(q_4) = 0; & acc_3(q_4) = 0; \end{array}$$

Vì chỉ có ứng dụng q_1 truy xuất đến cả hai thuộc tính A_1 và A_3, nên ta có

$$aff(A_1, A_3) = acc_1(q_1) + acc_2(q_1) + acc_3(q_1) = 45$$

Tương tự ta tính được ma trận ái lực sau

$$AA = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{pmatrix} \end{array}$$

b) Thuật toán tụ nhóm

Nhiệm vụ cơ bản trong việc xây dựng thuật toán phân mảnh dọc là tìm tiêu chí nào đó để nhóm các thuộc tính của quan hệ dựa trên ma trận ái lực. *Thuật toán năng lượng nối BEA* (*bond energy algorithm*) là thích hợp vì những lý do sau:

(1) BEA gom các thuộc tính có cùng độ lớn ái lực lại với nhau.

(2) Các kết quả tụ nhóm không bị ảnh hưởng bởi thứ tự đưa các thuộc tính vào thuật toán.

(3) Độ phức tập tính toán $O(n^2)$, với n là số thuộc tính, là chấp nhận được.

(4) Có thể xác định mối liên hệ giữa các nhóm thuộc tính.

Thuật toán năng lượng nối BEA hoán vị các hàng và cột ma trận ái lực để tạo thành *ma trận ái lực tụ CA* (*clustered affinity matrix*). Hoán vị được thực hiện để cực đại *số đo ái lực chung AM* (*global affinity measure*):

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

trong đó

$$aff(A_0, A_j) = aff(A_i, A_0) = aff(A_{n+1}, A_j) = aff(A_i, A_{n+1}) = 0 \quad \forall i,j$$

Vì ma trận ái lực *AA* đối xứng nên hàm mục tiêu *AM* có thể rút gọn như sau

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

Quá trình sinh *ma trận ái lực tụ CA* được thực hiện qua 3 bước.

(1) *Khởi tạo*. Đặt cố định một trong các cột của *AA* vào *CA*, chẳng hạn cột 1. Gán số cột trong *CA* i := 1.

(2) *Thực hiện lặp*. Lấy lần lượt một trong $n-i$ cột còn lại trong *AA* đặt vào $i+1$ vị trí giữa i cột trong *CA* (kể cả hai đầu). Chọn nơi đặt sao cho nó làm tăng nhiều nhất số đo ái lực được mô tả ở trên. Tăng $i := i+1$.
Tiếp tục bước này cho đến khi $i = n$.

(3) *Sắp thứ tự hàng*. Một khi thứ tự các cột đã được xác định, các hàng cũng sắp thứ tự lại tương ứng.

Để bước (2) của thuật toán hoạt động chúng ta cần định nghĩa xem *đóng góp* (*contribution*) của thuộc tính vào số đo ái lực mang ý nghĩa gì.
Ta có

$$\begin{aligned} AM &= \sum_{i=1}^{n}\sum_{j=1}^{n} aff(A_i, A_j)[aff(A_i, A_{j-1}) + aff(A_i, A_{j+1})] \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} \left[aff(A_i, A_j)aff(A_i, A_{j-1}) + aff(A_i, A_j)aff(A_i, A_{j+1})\right] \end{aligned}$$

Ta định nghĩa *cầu nối* (*bond*) giữa hai thuộc tính A_x và A_y như sau

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

Như vậy *AM* có thể viết lại là

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

Bây giờ xét n thuộc tính sau

$$\underbrace{A_1, A_2, \ldots, A_{i-1}}_{AM'} \underbrace{A_i A_j A_{j+1}, \ldots, A_n}_{AM''}$$

Số đo ái lực chung cho các thuộc tính này có thể viết như sau:

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

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

Bây giờ xét đến việc đặt thuộc tính mới A_k vào giữa thuộc tính A_i và A_j trong ma trận ái lực tụ:

$$\underbrace{A_1, A_2, \ldots, A_{i-1}}_{AM'} \underbrace{A_i A_k A_j A_{j+1}, \ldots, A_n}_{AM''}$$

Số đo ái lực chung mới có thể biểu diễn tương tự như sau:

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

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

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

Sau đây là *thuật toán năng lượng nối* BEA

• **Thuật toán BEA**
Đầu vào: Ma trận ái lực *AA* cấp *n*.
Đầu ra: Ma trận ái lực tụ *CA*.
{*loc* là biến lưu vị trí nơi có giá trị đóng góp *cont* lớn nhất}
Begin
{khởi tạo}
CA(•,1) := *AA*(•,1); {đưa cột thứ nhất của *AA* vào cột thứ nhất của *CA*}
CA(•,2) := *AA*(•,2); {đưa cột thứ hai của *AA* vào cột thứ hai của *CA*}
index := 3;
while *index* ≤ *n* **do** {chọn vị trí tốt nhất cho cột AA_{index}}
begin
for *i:=1* to *index−1* **do**
begin
tính $cont(A_{i-1}, A_{index}, A_i)$
end {for}
tính $cont(A_{index-1}, A_{index}, A_{index+1})$

```
        loc := nơi có giá trị cont lớn nhất.
        for  j := index downto loc do
            CA(•, j) := CA(•, j−1)
            CA(•, loc) := AA(•, index)
        index := index + 1
      end;
  {sắp thứ tự các hàng theo thứ tự các cột}
End; {BEA}
```

◊ *Ví dụ*

Xét ma trận *AA* tính ở ví dụ trước

$$AA = \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} 45 & 0 & 45 & 0 \\ 0 & 80 & 5 & 75 \\ 45 & 5 & 53 & 3 \\ 0 & 75 & 3 & 78 \end{pmatrix} \end{array}$$

Ta tính phần đóng góp khi di chuyển A_4 vào giữa các thuộc tính A_1 và A_2 được cho bằng công thức

$$cont(A_1, A_4, A_2) = 2bond(A_1, A_4) + 2bond(A_4, A_2) - 2bond(A_1, A_2)$$
$$= 2*135 + 2*11865 - 2*225 = 23550$$

vì

$$bond(A_1, A_4) = 45*0 + 0*75 + 45*3 + 0*78 = 135$$
$$bond(A_4, A_2) = 0*0 + 75*80 + 3*5 + 78*75 = 11865$$
$$bond(A_1, A_2) = 45*0 + 0*80 + 45*5 + 0*75 = 225$$

Trường hợp thuộc tính A_k đặt vào tận cùng bên trái hoặc tận cùng bên phải ta có

$$bond(A_0, A_k) = bond(A_k, A_{k+1}) = 0$$

◊*Ví dụ*

Ta xét tiếp ví dụ trên. Bây giờ ta gom tụ các thuộc tính của quan hệ PROJ và dùng ma trận ái lực *AA*.

Theo bước khởi đầu ta đưa cột 1 và cột 2 của *AA* vào *CA*

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} & & 45 & 0 \\ & & 5 & 75 \\ & & 53 & 3 \\ & & 3 & 78 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array} \begin{array}{c} \begin{array}{cc} A_1 & A_2 \end{array} \\ \begin{pmatrix} 45 & 0 \\ 0 & 80 \\ 45 & 5 \\ 0 & 75 \end{pmatrix} \end{array}$$

Tiếp theo ta di chuyển cột 3 của AA sang CA. Có 3 nơi có thể đặt cột 3:

- Bên trái cột 1 (0-3-1): ta có

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

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

suy ra

$cont(A_0,A_3,A_1) = 2\ bond(A_0,A_3) + 2bond(A_3,A_1) - 2\ bond(A_0,A_1) = 8820$

- Giữa cột 1 và 2 (1-3-2): ta có

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

$bond(A_3,A_2) = 890$

$bond(A_1,A_2) = 225$

suy ra

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

- Bên phải cột 2 (2-3-4): ta có

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

$bond(A_1,A_4) = 890$

suy ra

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

Vì đóng góp của trường hợp thứ 2 là lớn nhất, ta chèn A_3 vào giữa A_1 và A_2, và có

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array}\begin{array}{c} \begin{array}{cccc} A_1 & A_2 & A_3 & A_4 \end{array} \\ \begin{pmatrix} & & & 0 \\ & & & 75 \\ & & & 3 \\ & & & 78 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array}\begin{array}{c} \begin{array}{ccc} A_1 & A_3 & A_2 \end{array} \\ \begin{pmatrix} 45 & 45 & 0 \\ 0 & 5 & 80 \\ 45 & 53 & 5 \\ 0 & 3 & 75 \end{pmatrix} \end{array}$$

Tính toán tương tự cho A_4 chỉ ra rằng cần đặt nó bên phải A_2 .

$$\begin{array}{c} \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{array}\begin{array}{c} \begin{array}{cccc} A_1 & A_3 & A_2 & A_4 \end{array} \\ \begin{pmatrix} 45 & 45 & 0 & 0 \\ 0 & 5 & 80 & 75 \\ 45 & 53 & 5 & 3 \\ 0 & 3 & 75 & 78 \end{pmatrix} \end{array}$$

Cuối cùng ta hoán chuyển thứ tự các hàng và được ma trận kết quả

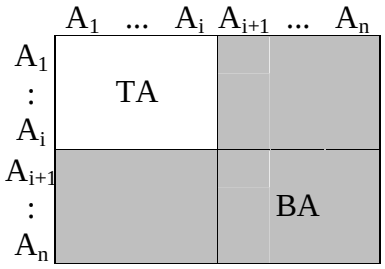
$$CA = \begin{array}{c} \\ A_1 \\ A_3 \\ A_2 \\ A_4 \end{array}\begin{array}{c} \begin{array}{cccc} A_1 & A_3 & A_2 & A_4 \end{array} \\ \begin{pmatrix} 45 & 45 & 0 & 0 \\ 45 & 53 & 5 & 3 \\ 0 & 5 & 80 & 75 \\ 0 & 3 & 75 & 78 \end{pmatrix} \end{array}$$

Ta thấy quá trình tạo ra hai tụ: một ở góc trên bên trái chứa các ái lực nhỏ, còn tụ kia ở góc dưới bên phải chứa các ái lực cao.

c) Thuật toán phân hoạch

Mục đích của hành động tách thuộc tính là tìm tập thuộc tính được truy xuất cùng nhau, hoặc với phần lớn các thuộc tính, bởi các tập ứng dụng riêng biệt. Thí dụ, nếu biết hai thuộc tính A_1 và A_2 chỉ được ứng dụng q_1 truy xuất và các thuộc tính A_3 và A_4 được các ứng dụng khác truy xuất, chẳng hạn q_2 và q_3, thì quyết định phân mảnh là đương nhiên. Vấn đề là tìm được thuật toán để xác định các nhóm này.

Xét ma trận thuộc tính tụ *CA*. Nếu một điểm nằm trên đường chéo được cố định, hai tập thuộc tính sẽ được xác định. Tập $\{A_1, ..., A_i\}$ ở góc trên trái và tập $\{A_{1+1}, ..., A_n\}$ ở góc dưới phải. Tập thứ nhất gọi là *đỉnh* (top) và ký hiệu *TA* , tập thứ hai gọi là *đáy* (bottom) và ký hiệu *BA*.



Bây giờ xét tập ứng dụng Q = $\{q_1, ..., q_k\}$ và định nghĩa các tập ứng dụng chỉ truy xuất TA, BA hoặc cả hai. Những tập này định nghĩa như sau

$$
\begin{aligned}
AQ(q_i) &= \{A_j \mid use(q_i, A_j) = 1\} \\
TQ &= \{q_j \mid AQ(q_i) \subseteq TA\} \\
BQ &= \{q_j \mid AQ(q_i) \subseteq BA\} \\
OQ &= Q - (TQ \cup BQ)
\end{aligned}
$$

Phương trình đầu định nghĩa tập thuộc tính được truy xuất bởi ứng dụng q_i ; TQ và BQ tương ứng là các tập ứng dụng chỉ truy xuất TA và BA, còn OQ là tập ứng dụng truy xuất cả hai tập TA và BA.

Ở đây nảy sinh bài toán tối ưu hoá. Nếu có *n* thuộc tính trong quan hệ, thì sẽ có $n-1$ vị trí khả hữu có thể là điểm phân chia trên đường chéo của ma trận thuộc tính tụ cho quan hệ đó. Vị trí tốt nhất để phân chia là vị trí sinh các tập TQ và BQ sao cho tổng các *truy xuất chỉ một mảnh* là lớn nhất, còn tổng các *truy xuất cả hai mảnh* là nhỏ nhất. Vì thế ta định nghĩa các phương trình chi phí như sau:

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

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

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

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

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

$$z = CTQ*CBQ - COQ^2$$

lớn nhất.

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

Để tăng thêm số phương án xét, tức hiệu quả thuật toán, chúng ta cần hiệu chỉnh thuật toán bằng thủ tục *xê dịch* (SHIFT) thuộc tính như sau. Sau mỗi bước lặp tìm vị trí tốt nhất cho một thứ tự thuộc tính, ta dịch cột tận trái sang thành cột tận phải, và hàng trên cùng xuống cuối cùng. Với thao tác xê dịch như thế này ta chỉ cần kểm tra $n-1$ vị trí trên đường chéo để tìm trị z lớn nhất. Độ phức tạp của thuật toán sẽ tăng thêm n lần, tức là $O(n^2)$.

Với giả thiết đã có thủ tục xê dịch SHIFT, ta có thuật toán phân hoạch PARTITION sau:

• **Thuật toán PARTITION**

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

Đầu ra: tập các mảnh $F_R = \{R_1, R_2\}$. với $R_1 \cap R_2$ là khoá.

Begin
{xác định giá trị z cho cột thứ nhất}
{các chỉ mục trong phương trình chi phí chỉ ra điểm tách}
Tính CTQ_{n-1} ;
Tính CBQ_{n-1} ;
Tính COQ_{n-1} ;
gán *best* := $CTQ_{n-1} * CBQ_{n-1} - (COQ_{n-1})^2$;
do {xác định cách phân hoạch tốt nhất}
begin
for i := n–2 downto 1 do
begin
Tính CTQ_i ;
Tính CBQ_i ;

Tính COQ_i ;
gán $z := CTQ_i * CBQ_i - (COQ_i)^2$;
if $z > best$ *then*
begin
$best := z$;
ghi nhận điểm tách vào trong hành động xê dịch
end;
end;{for}
gọi SHIFT(CA);
end;
until không thể thực hiện SHIFT được nữa.
xây dựng lại ma trận theo vị trí xê dịch
$R_1 := \pi_{TA}(R) \cup K$ {K là tập thuộc tính khoá chính của R}
$R_2 := \pi_{BA}(R) \cup K$
$F := \{R_1, R_2\}$;
end.

◊ *Ví dụ*

Áp dụng thuật toán PARTITION cho ma trận *CA* từ quan hệ PROJ ở thí dụ trước. Kết quả là định nghĩa các mảnh $F_{PROJ} = \{PROJ_1, PROJ_2\}$, trong đó

$PROJ_1 = \{A_1, A_3\} = \{ PNO, BUDGET \}$
$PROJ_2 = \{A_1, A_2, A_4\} = \{ PNO, PNAME, LOC \}$

d) Kiểm tra tính đúng đắn

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

♦ *Tính tái thiết*: đảm bảo

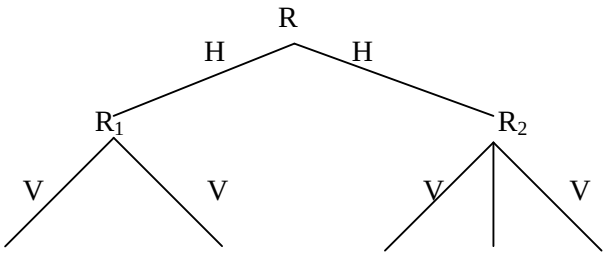
$$R = \ ><_K R_i \ \forall R_i \in F_R .$$

nếu mỗi R_i là đầy đủ và chứa thuộc tính khoá của R hoặc chứa *mã định bộ* (TID - tuple identifier) được gán bởi hệ thống.

♦ *Tính tách biệt*: đảm bảo tách biệt đối với các thuộc tính không khoá.

2.3.3. Phân mảnh hỗn hợp

Trong đa số trường hợp, phân mảnh ngang hoặc phân mảnh dọc đơn giản cho một lược đồ CSDL không đủ đáp ứng yêu cầu các ứng dụng. Khi đó phân mảnh dọc có thể được thực hiện sau một phân mảnh ngang hoặc ngược lại. Chiến lược này sinh ra một lối phân hoạch có cấu trúc cây và gọi là *phân mảnh hỗn hợp* (hybrid fragmentation).



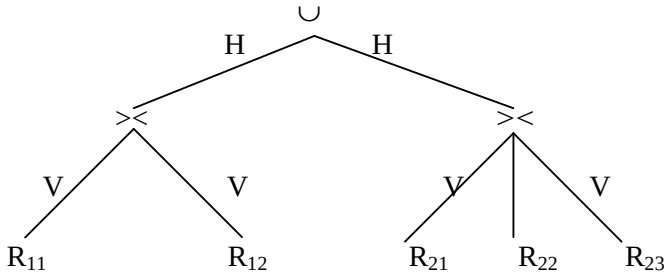
R_{11} R_{12} R_{21} R_{22} R_{23}

Phân mảnh hỗn hợp

Một ví dụ điển hình minh hoạ cho sự cần thiết phải có phân mảnh hỗn hợp là quan hệ PROJ. Trong một ví dụ trước ta đã phân hoạch nó thành sáu mảnh ngang dựa vào hai ứng dụng. Trong một ví dụ sau đó, chúng ta lại phân mảnh dọc PROJ thành hai mảnh. Như thế chúng ta có một tập các mảnh ngang, mỗi mảnh ngang lại được phân tiếp thành hai mảnh dọc.

Số mức lồng ghép có thể khá lớn, nhưng hữu hạn. Trong thực tế, do các quan hệ toàn cục đã chuẩn hoá và chi phí nối nhiều mảnh có thể quá cao, nên phần lớn người ta chỉ thực hiện tối đa hai mức lồng ghép.

Tính đúng đắn của phân mảnh hỗn hợp suy ra từ tính đúng đắn của phân mảnh ngang và phân mảnh dọc. Thí dụ để tái thiết quan hệ toàn cục, người ta bắt đầu tại các nút lá của cây phân hoạch và di chuyển lên trên bằng cách thực hiện các phép nối và hợp như ở hình sau



Tính tái thiết của phân mảnh hỗn hợp

2.4. Cấp phát

Cấp phát tài nguyên cho các nút của mạng máy tính là bài toán đã được nghiên cứu rộng rãi.

2.4.1. Bài toán cấp phát

Giả sử có tập các mảnh $FF = \{F_1, ..., F_n\}$ và mạng bao gồm các vị trí $SS = \{S_1,..., S_m\}$ trên đó có tập ứng dụng $QQ = \{q_1 ,..., q_l\}$ đang chạy.

Bài toán cấp phát (allocation problem) là tìm phân phối tối ưu của FF cho SS.

Một trong các điểm quan trọng là định nghĩa khái niệm *tối ưu* (*optimality*). Khái niệm tối ưu có thể định nghĩa ứng với hai số đo:

(1) *Chi phí nhỏ nhất*: Hàm chi phí gồm có chi phí lưu mỗi mảnh F_i tại vị trí S_j, chi phí vấn tin F_i tại vị trí S_j , chi phí cập nhật F_i tại tất cả mọi vị trí chứa nó và chi phí truyền dữ liệu. Bài toán cấp phát cố gắng tìm lược đồ cấp phát với hàm chi phí tổ hợp thấp nhất.

(2) *Hiệu năng*: Chiến lược cấp phát được thiết kế nhằm duy trì hiệu năng hữu lượng. Các chiến lược đã biết là hạ thấp thời gian đáp ứng và tăng tối đa lưu lượng hệ thống tại mỗi vị trí.

Một lược đồ cấp phát hoàn hảo là trả lời câu vấn tin trong thời gian ngắn nhất mà vẫn duy trì chi phí xử lý thấp nhất. Do tính phức tạp của bài toán, nên người ta mới chỉ giải quyết những trường hợp đơn giản.

Chúng ta xét một trường hợp đơn giản. Cố định mảnh $F \in FF$. Chúng ta đưa ra một số giả thiết và định nghĩa nhằm mô hình hoá bài toán cấp phát này.

(1) Giả sử có thể sửa đổi QQ để xác định được các *vấn tin cập nhật* (update query) và *vấn tin chỉ đọc* (retrieval-only query), và để định nghĩa các đại lượng sau đây cho mảnh F

$$T = \{t_1,...,t_m\}$$

với t_i là lưu lượng chỉ đọc được sinh ra tại S_i cho mảnh F và

$$U = \{u_1,...,u_m\}$$

với u_i là lưu lượng cập nhật được sinh ra tại S_i cho mảnh F .

(2) Giả sử chi phí truyền giữa hai vị trí S_i và S_j là cố định với mỗi đơn vị truyền. Ngoài ra cũng giả sử rằng chúng khác nhau khi cập nhật và khi truy xuất chỉ đọc. Ta định nghĩa

c_{ij} là chi phí truyền 1 đơn vị đối với các yêu cầu *chỉ đọc* giữa vị trí S_i và S_j ($i,j=1, ..., m$);

c'_{ij} là chi phí truyền 1 đơn vị đối với các yêu cầu *cập nhật* giữa vị trí S_i và S_j ($i, j=1, ..., m$);

và ký hiệu

$$C(T) = \{c_{12}, ..., c_{1m}, ..., c_{m-1,m}\}$$
$$C'(U) = \{c'_{12}, ..., c'_{1m}, ..., c'_{m-1,m}\}$$

(3) Gọi chi phí lưu trữ mảnh F tại vị trí S_i là d_i và ký hiệu

$$D = \{d_1, ..., d_m\}$$

(4) Giả thiết không có ràng buộc về khả năng lưu trữ cho các vị trí hoặc cho các đường truyền.

Khi đó bài toán cấp phát có thể được đặc tả là bài toán cực tiểu hoá chi phí, trong đó ta tìm tập $I \subseteq SS$ các vị trí lưu các bản sao của mảnh F. Ký hiệu x_i là *biến quyết định* (decision variable) chọn nơi đặt sao cho

$$x_i = \begin{cases} 1, \text{ nếu } F \text{ phân cho } S_i \\ 0, \text{ nếu ngược lại} \end{cases}$$

Bài toán cấp phát được phát biểu chính xác như sau

$$\min_{I \subset SS} \left[\sum_{i=1}^{m} \left(\sum_{S_j \in I} x_j u_j c_{ij} + t_i \min_{S_j \in I} c_{ij} \right) + \sum_{S_j \in I} x_j d_j \right]$$

Số hạng thứ nhất tương ứng với chi phí truyền các cập nhật từ các vị trí (S_i) đến mọi vị trí có giữ bản sao của mảnh F và với chi phí thực hiện các yêu cầu chỉ đọc tại vị trí đó (S_i) nhưng với chi phí truyền dữ liệu thấp nhất. Số hạng thứ hai tính tổng chi phí lưu tất cả bản sao của mảnh F.

Cách đặt vấn đề hết sức đơn giản và không phù hợp lắm với thực tế thiết kế cơ sở dữ liệu phân tán. Đây thực chất là mô hình *bài toán cấp phát tập tin* - FAP (*file allocation problem*) cho mạng máy tính. Để phân biệt với bài toán trên, ta gọi bài toán cấp phát mảnh trong thiết kế cơ sở dữ liệu phân tán là *bài toán cấp phát cơ sở dữ liệu* - DAP (*database allocation problem*). Trong thực tế, vì bài toán quá phức tạp nên ta chỉ nghiên cứu các *phương pháp heuristic* để tìm lời giải gần tối ưu.

2.4.2. Yêu cầu về thông tin

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

a) Thông tin về cơ sở dữ liệu

Cho mảnh F_j và ứng dụng q_i. Độ tuyển của F_j ứng với ứng dụng q_i, ký hiệu $sel_i(F_j)$, là số lượng các bộ của F_j mà q_i truy xuất để xử lý.

Kích thước của mảnh F_j cũng được định nghĩa thông qua lực lượng $card(F_j)$ và độ dài $length(F_j)$ như sau

$$size(F_j) = card(F_j) * length(F_j)$$

b) Thông tin về ứng dụng

Phần lớn các thông tin về ứng dụng đều đã được biên dịch trong khi thực hiện phân mảnh. Hai số liệu quan trọng là

RR_{ij} số truy xuất chỉ đọc bởi câu vấn tin q_i thực hiện trên mảnh F_j,
UR_{ij} số truy xuất cập nhật bởi câu vấn tin q_i thực hiện trên mảnh F_j.

Ta cũng định nghĩa hai ma trận UM và RM với các phần tử tương ứng u_{ij} và r_{ij} như sau

$$u_{ij} = \begin{cases} 1, \text{ nếu } q_i \text{ cập nhật } F_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

$$r_{ij} = \begin{cases} 1, \text{ nếu } q_i \text{ cần đọc } F_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

Tiếp theo, vectơ $O = \{o(1),..., o(k)\}$ gồm các thành phần $o(i)$ đặc tả vị trí đưa ra câu vấn tin q_i (i = 1, .., k).

Cuối cùng để định nghĩa ràng buộc thời gian đáp ứng, thời gian đáp ứng tối đa được phép của mỗi ứng dụng cũng cần được đặc tả.

c) Thông tin về vị trí

Với mỗi vị trí, chúng ta cần biết về khả năng lưu trữ và xử lý của nó. Ta ký hiệu

USC_k là chi phí lưu 1 đơn vị dữ liệu tại S_k.
LPC_k là chi phí xử lý 1 đơn vị dữ liệu tại S_k.

d) Thông tin về mạng

Chi phí truyền dữ liệu được định nghĩa theo đơn vị là *bó dữ liệu* (frame). Ký hiệu

g_{ij} là chi phí truyền 1 bó dữ liệu giữa hai vị trí S_i và S_j.

2.4.3. Mô hình cấp phát

Chúng ta sẽ thảo luận một mô hình cấp phát có mục tiêu là giảm thiểu tổng chi phí xử lý và lưu trữ trong khi vẫn cố gắng đáp ứng được các đòi hỏi về thời gian đáp ứng. Mô hình có dạng sau

min (Tổng chi phí)

với ràng buộc

ràng buộc thời gian đáp ứng, ràng buộc lưu trữ, ràng buộc xử lý.

Ta sẽ khai triển các thành phần của mô hình này. Ký hiệu biến quyết định

$$x_{ij} = \begin{cases} 1, \text{ nếu } F_i \text{ phân cho } S_j \\ 0, \text{ nếu ngược lại} \end{cases}$$

- ***Tổng chi phí***

Hàm tổng chi phí có hai thành phần: phần xử lý vấn tin và phần lưu trữ. Vì vậy nó có thể biểu diễn là

$$TOC = \sum_{q_i \in QQ} QPC_i + \sum_{S_k \in SS} \sum_{F_j \in FF} STC_{jk}$$

với

QPC_i là chi phí xử lý câu vấn tin của ứng dụng q_i

STC_{jk} là chi phí lưu mảnh F_j tại vị trí S_k.

Ta xét chi phí lưu trữ trước. Nó được cho bởi

$$STC_{jk} = USC_k * size(F_j) * x_{jk}$$

và hai ký hiệu tổng là tìm các tổng chi phí lưu trữ tại tất cả vị trí cho tất cả các mảnh.

Chi phí xử lý vấn tin khó xác định hơn. Ta tách QPC_i thành hai thành phần chi phí xử lý *PC* và chi phí truyền *TC*

$$QPC_i = PC_i + TC_i$$

Thành phần xử lý *PC* gồm 3 hệ số chi phí: chi phí truy xuất *AC,* chi phí toàn vẹn *IE* và chi phí điều khiển đồng thời *CC*:

$$PC_i = AC_i + IE_i + CC_i$$

Mô tả chi tiết cho mỗi hệ số chi phí phụ thuộc thuật toán dùng để hoàn tất các tác vụ đó. Chi tiết về chi phí truy xuất AC như sau

$$AC_i = \sum_{S_k \in SS} \sum_{F_j \in FF} \left(u_{ij} * UR_{ij} + r_{ij} * RR_{ij} \right) * x_{jk} * LPC_k$$

Hai số hạng đầu trong công thức trên tính số truy xuất của vấn tin q_i đến mảnh F_j. Chú ý rằng

$$\left(u_{ij} * UR_{ij} + r_{ij} * RR_{ij}\right)$$

là tổng số các truy xuất cập nhật và đọc mảnh F_j. Giả thiết chi phí xử lý là giống nhau với mọi mảnh. Ký hiệu tổng cho biết tổng số các truy xuất cho tất cả các mảnh được q_i tham chiếu. Nhân với LPC_k cho ra chi phí của truy xuất này tại vị trí S_k. Ta dùng x_{jk} để tách giá trị chi phí cho các vị trí có lưu các mảnh.

Hệ số chi phí duy trì tính toàn vẹn cơ sở dữ liệu có thể mô tả giống thành phần xử lý ngoại trừ chi phí xử lý cục bộ một đơn vị cần được thay đổi nhằm phản ánh chi phí thực sự để duy trì tính toàn vẹn. Ta sẽ quay lại vấn đề này ở chương sau.

Hàm chi phí truyền dữ liệu có thể biểu diễn giống hàm chi phí truy xuất. Tuy nhiên tổng chi phí truyền dữ liệu cho cập nhật và cho yêu cầu chỉ đọc sẽ khác nhau hoàn toàn. Trong vấn tin cập nhật, ta cần cho tất cả mọi vị trí biết nơi có các bản sao, còn trong vấn tin chỉ đọc thì chỉ cần truy xuất đến 1 trong các bản sao là đủ. Ngoài ra vào lúc kết thúc yêu cầu vấn tin cập nhật thì không cần truyền dữ liệu ngược lại cho vị trí đưa ra vấn tin ngoài một thông báo xác nhận, còn trong vấn tin chỉ đọc có thể phải có nhiều thông báo truyền dữ liệu.

Thành phần cập nhật hàm truyền dữ liệu là

$$TCU_i = \sum_{S_k \in SS} \sum_{F_j \in FF} u_{ij} * x_{jk} * g_{o(i),k} + \sum_{S_k \in SS} \sum_{F_j \in FF} u_{ij} * x_{jk} * g_{k,o(i)}$$

Số hạng thứ nhất để gửi thông báo cập nhật từ vị trí gốc $o(i)$ của q_i đến tất cả bản sao cần cập nhật. Số hạng thứ hai dành cho thông báo xác nhận.

Thành phần chi phí chỉ đọc là

$$TCR_i = \sum_{F_j \in FF} \min_{S_k \in SS} \left(r_{ij} * x_{jk} * g_{o(i),k} + r_{ij} * x_{jk} * \frac{sel_i(F_j) * length(F_j)}{fsize} * g_{k,o(i)} \right)$$

Số hạng thứ nhất trong TCR biểu thị chi phí truyền yêu cầu chỉ đọc đến những vị trí có bản sao của mảnh cần truy xuất. Số hạng thứ hai biểu thị chi phí để truyền các kết quả từ những vị trí này đến vị trí yêu cầu. Phương trình này khẳng định rằng trong số các vị trí có bản sao của cùng 1 mảnh, chỉ vị trí có tổng chi phí truyền thấp nhất mới được chọn để thực hiện thao tác này.

Bây giờ hàm chi phí truyền dữ liệu của câu vấn tin q_i có thể được tính là

$$TC_i = TCU_i + TCR_i$$

- ***Ràng buộc***

Các hàm ràng buộc có thể được đặc tả tương tự. Tuy nhiên thay vì mô tả những hàm này kỹ lưỡng, chúng ta chỉ nêu những ý tổng quát.

Ràng buộc thời gian đáp ứng cần được đặc tả là

$$\textit{thời gian thực thi của } q_i \leq \textit{thời gian đáp ứng lớn nhất của } q_i\ , \forall q_i \in QQ.$$

Người ta thường đặc tả số đo chi phí của hàm theo thời gian vì nó đơn giản.

Ràng buộc lưu trữ là

$$\sum_{F_j \in FF} STC_{jk} \leq \textit{khả năng lưu trữ tại vị trí } S_k, \forall S_k \in SS$$

Ràng buộc xử lý là

$$\sum_{q_i \in QQ} \textit{tải trọng xử lý của } q_i \textit{ tại } S_k \leq \textit{khả năng xử lý tại vị trí } S_k, \forall S_k \in SS$$

Mô hình bài toán cấp phát có lời giải phi đa thức, vì thế người ta tìm các phương pháp *heuristic* cho lời giải gần tối ưu. Có sự tương ứng giữa bài toán cấp phát và bài toán chọn vị trí đặt thiết bị đã được khám phá trong các nghiên cứu về quá trình điều hành sản xuất.

BÀI TẬP

1. Cho quan hệ

EMP

ENO	ENAME	TITLE
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

Ký hiệu p_1 là vị từ TITLE < "Programmer" và p_2 là vị từ TITLE > "Programmer". Giả thiết chuỗi ký tự được sắp theo thứ tự từ điển.

a) Thực hiện phân mảnh ngang cho quan hệ EMP ứng với $\{p_1, p_2\}$.

b) Giải thích tại sao phân mảnh kết quả $\{EMP_1, EMP_2\}$ không đáp ứng quy tắc đúng đắn của phân mảnh.

c) Sửa lại các vị từ p_1 và p_2 để chúng phân hoạch EMP theo các quy tắc đúng đắn của phân mảnh.

Hướng dẫn. Sửa các vị từ bằng cách xây dựng các vị từ tiểu hạng và suy ra các phép kéo theo tương ứng với thực hiện phân mảnh ngang của EMP dựa trên các vị từ tiểu hạng này. Cuối cùng chứng tỏ kết quả có tính đầy đủ, tính tái thiết và tính tách biệt.

2. Xét quan hệ

ASG

ENO	PNO	RESP	DUR
E1	P1	Manager	12
E2	P1	Analyst	24
E2	P2	Analyst	6
E3	P3	Consultant	10
E3	P4	Engineer	48
E4	P2	Programmer	18
E5	P2	Manager	24
E6	P4	Manager	48
E7	P3	Engineer	36
E8	P3	Manager	40

Giả sử có hai ứng dụng truy xuất ASG. ứng dụng thứ nhất được đưa ra tại 5 vị trí và muốn tìm thời gian phân công các nhân viên khi biết mã số nhân viên. Giả sử rằng nhà quản lý (Manager), nhà tư vấn (Consultant), kỹ sư (Engineer) và lập trình viên (Programmer) được lưu tai 4 vị trí khác nhau. ứng dụng thứ hai được đưa ra tại 2 vị trí trong đó các nhân viên có thời gian phân công dưới 20 tháng được quản lý tại một vị trí còn nhứng người trên 20 tháng được quản lý tại vị trí thứ hai. Hãy phân mảnh ngang nguyên thủy cho ASG theo các thông tin trên.

3. Xét các quan hệ sau

EMP

ENO	ENAME	TITLE
E1	J.Doe	Elect.Eng.
E2	M.Smith	Syst.Anal.
E3	A.Lee	Mech.Eng.
E4	J.Miller	Programmer
E5	B.Casey	Syst.Anal.
E6	L.Chu	Elect.Eng.
E7	R.David	Mech.Eng.
E8	J.Jones	Syst.Anal.

PAY

TITLE	SAL
Elect.Eng.	40000
Syst.Anal.	34000
Mech.Eng.	27000
Programmer	24000

Giả sử EMP và PAY được phân mảnh ngang như sau:

$EMP_1 = \sigma_{TITLE="Elect.Eng."}(EMP)$
$EMP_2 = \sigma_{TITLE="Syst.Anal"}(EMP)$
$EMP_3 = \sigma_{TITLE="Mech.Eng."}(EMP)$
$EMP_4 = \sigma_{TITLE="Programmer."}(EMP)$
$PAY_1 = \sigma_{SAL >=30000}(PAY)$
$PAY_2 = \sigma_{SAL < 30000}(PAY)$

Vẽ đồ thị nối nửa của $E ><_{TITLE} PAY$. Đây là đồ thị đơn giản hay phân hoạch ? Nếu nó phân hoạch, hãy sửa lại phân mảnh của EMP hoặc PAY để có đồ thị nối nửa $E ><_{TITLE} PAY$ đơn giản.

4. Xét quan hệ PAY và EMP ở bài tập trên. Ký hiệu p_1 là vị từ SAL < 30000 và p_2 là vị từ SAL >= 30000. Thực hiện phân mảnh ngang cho PAY ứng với p_1 và p_2 để có được hai mảnh PAY_1 và PAY_2. Sử dụng phân mảnh của PAY thực hiện phân mảnh ngang dẫn xuất cho EMP. Chứng tỏ tính đầy đủ, tính tái thiết và tính tách biệt của phân mảnh EMP.

5. Xét các quan hệ EMP, ASG ở các bài tập trên. Giả sử định nghĩa khung nhìn sau

```
CREATE   VIEW    EMPVIEW(ENO, ENAME, PNO, RESP)
AS       SELECT  EMP.ENO, EMP.ENAME, ASG.PNO, ASG.RESP
         FROM    EMP, ASG
         WHERE   EMP.ENO = ASG.ENO
           AND   ASG.DUR = 24
```

được truy xuất bởi ứng dụng q_1 nằm tại các vị trí 1 và 2 với tần suất lần lượt là 10 và 20. Giả sử tiếp rằng có một vấn tin q_2 khác được định nghĩa là

```
SELECT   ENO, DUR
FROM     ASG
```

chạy tại các vị trí 2 và 3 với tần số tương ứng là 20 và 10. Dựa trên những thông tin này hãy xây dựng ma trận $use(q_i, A_j)$ cho các thuộc tính của cả hai quan hệ

EMP và ASG. Xây dựng ma trận ái lực chứa tất cả các thuộc tính của EMP và ASG. Cuối cùng, hãy biến đổi ma trận ái lực để có thể dùng nó khi tách các quan hệ này thành hai mảnh dọc bằng một heuristic hoặc thuật toán BEA.

6. Giả sử môi trường làm việc như ở bài tập 5. Giả sử 60% truy xuất của q_1 là cập nhật đến PNO và RESP của khung nhìn EMPVIEW, trong khi ASG.DUR không được cập nhật qua khung nhìn EMPVIEW. Ngoài ra, giả sử rằng tốc độ truyền dữ liệu giữa vị trí 1 và vị trí 2 chỉ bằng một nữa tốc độ truyền dữ liệu giữa vị trí 2 và vị trí 3. Dựa vào những thông tin trên hãy tìm một phân mảnh hợp lý của ASG và EMP và cách nhân bản và vị trí đặt dữ liệu tối ưu cho các mảnh, giả thiết rằng chi phí lưu trữ không đáng kể, nhưng các bản sao phải nhất quán.

Hướng dẫn. Xét phân mảnh ngang cho ASG dựa trên vị từ DUR=24 và phân mảnh ngang dẫn xuất tương ứng cho EMP. Cần xemét ma trận ái lực nhận được trong bài tập 5 cho EMP và ASG và xét xem nó có ý nghĩa gì không trong việc thực hiện phân mảnh dọc cho ASG.

7. Cho thí dụ về ma trận CA, trong đó điểm tách không duy nhất và phân hoạch nằm ngay giữa ma trận. Cho biết số các thao tác xê dịch cần thực hiện để có điểm tách duy nhất.

8. Gọi $Q = \{q_1, q_2, q_3, q_4, q_5\}$ là tập vấn tin, $A = \{A_1, A_2, A_3, A_4, A_5\}$ là tập thuộc tính và $S = \{S_1, S_2, S_3\}$ là tập vị trí. Ma trận giá trị sử dụng các thuộc tính và ma trận tần số truy xuất ứng dụng lần lượt là

	A_1	A_2	A_3	A_4	A_5
q_1	1	1	1	0	1
q_2	1	1	1	0	1
q_3	1	0	0	1	1
q_4	0	0	1	0	0
q_5	1	1	1	0	0

và

	S_1	S_2	S_3	
q_1	10	20	0	q_1
q_2	5	0	10	q_2
q_3	0	35	5	q_3
q_4	0	10	0	q_4
q_5	0	15	0	q_5

Giả thiết $ref_l(q_k) = 1$ cho mọi q_k và S_l, A_1 là thuộc tính khóa. Sử dụng các thuật toán năng lượng nối và phân hoạch dọc để có được một phân mảnh dọc cho tập thuộc tính trong A.