Tạp chí Tin học và Điều khiển học, T.23, S.1 (2007), 86–98

PHÂN MẢNH TRÊN GIÁ TRỊ LẶP CỦA CÁC THUỘC TÍNH TRONG CƠ SỞ DỮ LIỆU QUAN HỆ

LÊ HUY THẬP

Viện Công nghệ thông tin, Viện Khoa học và Công nghệ Việt Nam

Abstract. This paper presents a method of fragmentation of the given relational database, in which there's a number of attributes having the repeated values. The fragmentation is based on the principle of the horizontal, vertical and mixed fragmentations. The received fragmentations include the attributes having the repeated values and have the maximum size, then they are replaced by some aliases. The original database may be rebuilt from the subfragmentations and the aliases. The fragmentation of the given relational database by group of the repeated values of the attributes will save the needed memory, increase the processing speed of data. This work has been applied in people's committee of Hatay province and its districts and precints.

Tóm tắt. Bài báo giới thiệu phương pháp phân mảnh cơ sở dữ liệu quan hệ đã cho dựa vào sự lặp lại các giá trị của một số thuộc tính. Việc phân mảnh dựa trên nguyên tắc phân mảnh ngang, dọc và hỗn hợp. Các mảnh thu được chứa các thuộc tính có giá trị lặp và có kích thước lớn nhất, sau đó, các mảnh này được thay bởi các bí danh. Cơ sở dữ liệu gốc được tái thiết lại từ các mảnh con và từ các bí danh. Việc phân mảnh cơ sở dữ liệu quan hệ bằng cách gom nhóm các giá trị lặp của các thuộc tính sẽ tiết kiệm được dung lượng thông tin cần lưu lại trong bộ nhớ đồng thời làm tăng tốc độ xử lý dữ liệu.

1. MỞ ĐẦU

Quan hệ trong CSDL là một bảng - mà tiêu đề cột của bảng chính là thuộc tính, các hàng là những bản ghi. Để tránh dị thường về dữ liệu, người ta tìm cách tách, gộp các lược đồ quan hệ và đưa về các dạng chuẩn 1NF, 2NF, 3NF, Boye-Codd,... Tuy nhiên, trong thực tế, các quan hệ đã được chuẩn hóa nhưng vẫn có một số thuộc tính có thể có giá trị lặp lại nhiều lần (dĩ nhiên không phải là thuộc tính khóa). Chúng ta có thể dùng một bí danh nào đó để đại diện cho nhóm các bộ có các thuộc tính có các giá trị lặp lại đó, phần còn lại sẽ tạo ra các mảnh con. Khi cập nhật ta chỉ cần tái thiết lại bảng ban đầu từ các mảnh đã phân rã và tham chiếu đến bí danh đã đặt, bí danh là mảng mà các phần tử đại diện cho các giá trị trong nhóm và các thuộc tính của nhóm. Điều này làm giảm dung lượng thông tin cần lưu lại trong bộ nhớ và làm tăng tốc độ xử lý dữ liệu (đọc, viết, truy vấn,...).

2. NỘI DUNG

2.1. Khái niệm về phân mảnh

Với hệ quản trị CSDL phân tán, việc phân tán các ứng dụng cần thực hiện hai điều: Phân

mảnh và phân tán CSDL và phân tán các chương trình ứng dụng chạy trên hệ quản trị CSDL đó. Tuy nhiên trong bài báo này ta chỉ xét đến việc phân mảnh một bảng dữ liệu.

Việc phân rã một quan hệ thành nhiều mảnh, mỗi mảnh được xử lý như một đơn vị, sẽ cho phép thực hiện nhiều truy cập đồng thời. Ngoài ra việc phân mảnh các quan hệ sẽ cho phép thực hiện song song các câu truy vấn bằng cách chia nó thành một tập các câu vấn tin hoạt động trên các mảnh. Việc phân mảnh cũng làm tăng khả năng hoạt động đồng thời trên các mảnh, điều đó làm tăng lưu lượng hoạt động của hệ thống, tăng tốc độ xử lý.

Việc phân mảnh cũng có những nhược điểm của nó. Ví dụ có thể xảy ra xung đột khi có nhiều truy vấn xảy ra trên cùng một mảnh hoặc phải truy xuất dữ liệu từ hơn một mảnh rồi nối hoặc hợp chúng lại, do đó làm tăng chi phí truy xuất,... khắc phục điều này là một trong những mục tiêu của kỹ thuật phân mảnh. Ví dụ thứ hai có liên quan đến việc kiểm soát dữ liệu ngữ nghĩa, đặc biệt là việc kiểm tra tính toàn vẹn. Vì phân mảnh nên các thuộc tính tham gia vào một phụ thuộc nào đó có thể bị phân rã vào các mảnh khác nhau và được cấp phát đến các vị trí khác nhau. Trong trường hợp như vậy, việc kiểm tra các phụ thuộc đó cũng phải được lục tìm dữ liệu trên nhiều vị trí khác nhau.

Có ba cách phân mảnh cơ bản là: Phân mảnh ngang, phân mảnh dọc và phân mảnh hỗn hợp (kết hợp giữa phân mảnh ngang và phân mảnh dọc).

2.1.1. Phân mảnh ngang

Phân mảnh ngang chia một quan hệ thành các nhóm, các bộ và vì vậy mỗi mảnh là một quan hệ con của quan hệ đã cho. Có hai loại phân mảnh ngang: Phân mảnh ngang nguyên thủy - đó là việc phân mảnh một quan hệ dựa trên các mệnh đề được định nghĩa trên quan hệ đó và phân mảnh ngang dẫn xuất - là phân mảnh một bảng được quan hệ (child) dựa trên các mệnh đề được định nghĩa trên bảng quan hệ (parent).

Phân mảnh ngang được thực hiện dựa trên các mệnh đề hội sơ cấp, mệnh đề hội sơ cấp là hội của các mệnh đề đơn giản.

Cho quan hệ $R(Key, A_1, A_2, ..., A_n)$, trong đó Key là tập thuộc tính khóa, A_j là một thuộc tính không khóa được định nghĩa trên một miền D_j $(j = 1, ..., n)$ một mệnh đề đơn giản được định nghĩa trên R có dạng

$$\eta_j : A_j \ \theta \ d_j, \ j = \overline{1,n} \tag{1}$$

trong đó $\theta \in \{=, \neq, <, \leqslant, >, \geqslant\}$, $d_j \in D_j = Dom(A_j)$, $j = \overline{1,n}$. Ở đây, ta sẽ chỉ xét $\theta \in \{=, \neq\}$.

Giả thiết rằng trong một số D_j có các giá trị lặp lại (trùng nhau). Ký hiệu:

$$\begin{cases} n_r = Card_r(R), \\ P = \{P_1, P_2, ..., P_n\}, \end{cases} \tag{2}$$

trong đó, n_r là lực lượng hàng của quan hệ R, (là số bộ hiện có), P_j là tập các mệnh đề đơn giản trên thuộc tính A_j, $j = \overline{1,n}$.

Đặt β_{ij} là mệnh đề đơn giản trên thuộc tính j tác động lên giá trị thứ i của thuộc tính j. Ví dụ β_{42} : DanhHieu = "Huân Chương", tức là mệnh đề đơn giản tác động lên thuộc tính thứ 2 có tên là DanhHieu với quan hệ bằng với giá trị thứ 4 của thuộc tính là "Huân chương".

Ký hiệu:
$P'_j = \{\beta_{1j}, \beta_{2j}, \beta_{3j}, ...\} = \{\beta_{ij}\},\ j = \overline{1,n}$,
P' ký hiệu chuyển vị của P.
Hay $P = \{\beta_{ij}\}_{n_r \times n}$ là ma trận cấp $n_r \times n$.
Như vậy số mệnh đề đơn giản có thể có trên quan hệ R là $m = n \times n_r$.
Ký hiệu $Q = \{q_1, q_2, ..., q_n\}$ là tập các tập mệnh đề hội sơ cấp trên R trong đó q_k được tạo ra như sau:
Gọi $K \subseteq \{1, 2, ..., n\}$, với $Card(K) = k,\ k = \overline{1,n}$. Đặt:

$$q_{ik} = \underset{j \in K}{\wedge} \beta_{ij},\ \forall i = \overline{1, n_r}. \tag{3}$$

Như vậy có thể thấy $Card(Q) = \sum_{k=1}^{n} C_m^k$, trong đó $C_m^k = \dfrac{m!}{k!(m-k)!}$.
Khi dùng q_{ik} để phân mảnh ngang thì quan hệ R sẽ được chia thành các mảnh ngang H_{ik} (gồm các bộ của R thỏa mãn các mệnh đề hội sơ cấp q_{ik}). Vì thế nếu cho biết bảng quan hệ R, các mảnh ngang sẽ là H_{ik}được xác định như sau:

$$H_{ik} = \sigma_{Fk}(R), \tag{4}$$

trong đó F_{ik} là các công thức chọn được sử dụng để được mảnh H_{ik}. Thực chất F_{ik} là một mệnh đề hội sơ cấp dạng (3). Do đó nếu ký hiệu $z_{ik} = Card(q_{ik})$ thì số mảnh ngang sẽ là z_{ik}. Như vậy việc phân mảnh ngang phụ thuộc vào các mệnh đề hội sơ cấp. Và vì vậy phải xác định được tập các mệnh đề đơn giản sẽ được sử dụng để tạo ra các mệnh đề hội sơ cấp. Trong khuôn khổ bài báo này, ta chỉ xét $\theta \in \{=, \neq\}$, do đó các mệnh đề đơn giản trong trường hợp này là:

$$P_j : A_j = d_j. \tag{5}$$

Ứng dụng các thuật toán COM_MIN và PHORIZONTAL để phân mảnh ngang quan hệ đã cho là hoàn toàn thực hiện được ([1]).
Có thể coi việc phân mảnh ngang tương ứng với câu lệnh SQL như sau:

SELECT *
FROM R
WHERE <Mệnh đề hội sơ cấp>

Ký hiệu * ở đây để chỉ tất cả thuộc tính của quan hệ R. <Mệnh đề hội sơ cấp> là điều kiện tuyển chọn các bộ giá trị của R.
Mảnh ngang thu được là tập các bộ thu được qua việc thực hiện câu SQL ở trên.

2.1.2. Phân mảnh dọc

Phân mảnh dọc một quan hệ R sinh ra các mảnh $V_1, V_2, ..., V_d$. Mỗi mảnh con chứa một tập con thuộc tính của R và cả tập khóa Key của R. Có thể coi việc phân mảnh dọc tương ứng với câu lệnh SQL như sau:

SELECT <Tìm thuộc tính không khóa của R>
FROM R (6)
WHERE True

Khác với phân mảnh ngang điều kiện True ở đây là mệnh đề logic cho phép chọn ra các thuộc tính của R có giá trị thỏa mãn điều kiện nào đó. Ví dụ True là điều kiện để chọn ra các thuộc tính có độ rộng lớn nhất bé nhất hay các thuộc tính yêu cầu khác, chẳng hạn, Null.

Các mảnh dọc thu được có dạng $Key \cup V_l, l = \overline{1,d}$.

Tuy nhiên phương pháp tổng quát để phân mảnh dọc là sử dụng ma trận liên kết giữa các thuộc tính AA để tìm ra ma trận liên kết tụ CA thông qua thuật toán BEA ([1]) để gom tụ các thuộc tính có liên kết cao lại với nhau rồi tách các quan hệ dựa theo mối liên kết cao đó. Để làm được điều này thì khá phức tạp vì chúng ta phải biết được các ứng dụng, số truy xuất của các ứng dụng đó,... Để đơn giản và mang tính chất ứng dụng, ở đây chúng ta sẽ gom các thuộc tính A_j của quan hệ $R(Key, A_1, A_2, ..., A_n)$ thỏa mãn mệnh đề đơn giản (5), tức là $p_j : A_j = d_j$ =constant và mệnh đề (6) lại với nhau để cùng với các thuộc tính khóa của R tạo ra một mảnh (quan hệ con). Hơn thế nữa nếu việc phân mảnh thực hiện được thì với quan hệ con được tạo thành chỉ cần lưu các thuộc tính khóa và giá trị của chúng. Còn các thuộc tính không khóa và các giá trị của mảnh chỉ cần lưu dưới dạng biến nhớ mảng hai chiều. Với phương pháp này chúng ta có thể tiết kiệm được dung lượng bộ nhớ, thời gian đọc, cập nhật và xử lý - tức là giảm chi phí.

2.1.3. Phân mảnh hỗn hợp

Trường hợp khi phân mảnh kết hợp cả phân mảnh ngang và phân mảnh dọc thì quá trình phân mảnh được gọi là phân mảnh hỗn hợp. Phân mảnh hỗn hợp thường được thực hiện theo trình tự phân mảnh ngang trước sau đó phân mảnh dọc hoặc ngược lại ([1]). Trong bài báo này, ta sẽ chọn theo cách thứ nhất. Có thể coi việc phân mảnh hỗn hợp tương ứng với các câu lệnh SQL như sau:

SELECT *
FROM R
WHERE <Mệnh đề hội sơ cấp>

Mảnh thu được là tập các bộ được tách ra qua việc thực hiện câu SQL ở trên chính là mảnh ngang. Ký hiệu mảnh ngang thu được là H. Bây giờ thực hiện phân mảnh dọc từ các mảnh ngang đã có, dùng câu lệnh SQL như sau:

SELECT <Tìm thuộc tính không khóa của H >
FROM H
WHERE True

Ký hiệu mảnh dọc thu được do câu lệnh trên là V_H, và tập khóa của H là $KeyH$.

Mảnh hỗn hợp thu được có dạng $KeyH \cup V_H$ và được ký hiệu là HV.

2.1.4. Tái thiết quan hệ toàn cục

- Tái thiết một quan hệ toàn cục từ các mảnh ngang được thực hiện bằng toán tử hợp. Tức là nếu quan hệ R được phân thành các mảnh ngang $H_1, H_2, ..., H_h$ thì:

$$R = \bigcup_{i=1}^{h} H_i. \tag{7}$$

- Tái thiết một quan hệ toàn cục từ các mảnh dọc được thực bằng toán tử nối ∇. Tức là nếu quan hệ R được phân thành các mảnh dọc $V_1, V_2, ..., V_v$ thì:

$$R = \nabla_{i=1}^{v} V_i. \tag{8}$$

- Tái thiết một quan hệ toàn cục từ phân mảnh hỗn hợp được thực hiện bằng cách dùng toán tử nối để nối các mảnh dọc sau đó dùng toán tử hợp để hợp các mảnh ngang nếu thứ tự phân mảnh hỗn hợp là phân mảnh ngang sau đó mới phân mảnh dọc hoặc dùng toán tử hợp để hợp các mảnh ngang sau đó dùng toán tử nối để nối các mảnh dọc nếu thứ tự phân mảnh hỗn hợp là phân mảnh dọc trước sau đó mới phân mảnh ngang. Như vậy công việc tái thiết có thứ tự ngược với thứ tự phân mảnh ngang và dọc. Tức là:

$$R = \cup\nabla(HV)_i \text{ hoặc } \nabla \cup (VH)_i \text{ với } \forall(HV)_i \text{ và } \forall(VH)_i. \tag{9}$$

Thứ tự thực hiện phép toán từ phải sang trái, ngược với thứ tự phân mảnh.

Nếu các mảnh $(HV)_i$ thỏa mãn điều kiện $(HV)_i \cap (HV)_j \to (HV)_i - (HV)_j$ hoặc $(HV)_i \cap (HV)_j \to (HV)_j - (HV)_i \ \forall i, j$ thì việc phân mảnh và tái thiết sẽ không mất mát thông tin ([2]).

3. BÀI TOÁN PHÂN MẢNH ỨNG DỤNG

3.1. Bài toán phân mảnh tối ưu

Giả sử việc phân mảnh hỗn hợp quan hệ R sẽ cho ra n mảnh HV_i, mỗi mảnh HV_i sẽ cho hệ số lợi ích là hv_i. Gọi cr_i và ca_i là chi phí để chỉ đọc và để truy cập tương ứng đến mảnh HV_i ([1]). Như vậy có thể đặt vấn đề tìm chiến lược phân mảnh hỗn hợp từ quan hệ đã cho R để có các n mảnh HV_i làm cho hàm lợi ích:

$$U = \sum_{i=1}^{n} (hv_i - cr_i - ca_i) \tag{10}$$

có giá trị lớn nhất.

Các ràng buộc:

$$\begin{cases} \sum\limits_{i=1}^{n} Size(HV)_i \leqslant C, \ C \text{ là khả năng lưu trữ} \\ \sum\limits_{i=1}^{n} Time_{Read}(HV)_i \leqslant T_{read}, \ T_{read} \text{ là thời gian đọc tối đa} \\ \sum\limits_{i=1}^{n} Time_{Write}(HV)_i \leqslant T_{write}, \ T_{write} \text{ là thời gian ghi tối đa} \end{cases} \tag{11}$$

3.2. Bài toán phân mảnh ứng dụng

Để ứng dụng, ta sẽ không giải (10) - (11) vì phức tạp và kém hiệu quả, mà ta sẽ phân mảnh quan hệ R thành các mảnh hỗn hợp với mục đích đã nói ở phần mở đầu, tức là mảnh hỗn hợp chỉ gồm các thuộc tính được tuyển chọn kết hợp lại với nhau, mỗi thuộc tính chỉ nhận giá trị hằng và về nguyên tắc mỗi thuộc tính của mảnh được thay bởi một đại lượng hằng thay cho các miền D của thuộc tính đó. Trước hết tiến hành phân mảnh quan hệ R thành các mảnh ngang có một số thuộc tính có giá trị lặp (các mảnh này có thể giao nhau). Sau đó tập hợp các thuộc tính có giá trị lặp lại với nhau để tạo ra mảnh hỗn hợp.

Ví dụ, cho quan hệ $R \backslash KeyR$, $Size(TenHuyen) = 20$, $Size(DanhHieu) = 15, Size(Muc) = 10$, $Size(LyDoNhanDanhHieu) = 25, ...$

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	<Không lặp>
Thanh Oai	Huân chương	Hạng nhất	Chống Mỹ	
Thanh Oai	Huy chương		Chống Mỹ	
Ba Vì	Huân chương	Hạng nhất	Chống Pháp	
Ba Vì	Huân chương	Hạng nhất	Chống Mỹ	
Ba Vì	Huân chương	Hạng hai	Chống Pháp	
Ba Vì	Huân chương	Hạng nhất	Chống Mỹ	

Các mảnh ngang có thể có là (trong các mảng sau chỉ giữ lại các giá trị lặp).

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Thanh Oai				
Thanh Oai				

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Ba Vì				
Ba Vì				
Ba Vì				
Ba Vì				

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
	Huân chương			
	Huân chương			
	Huân chương			
	Huân chương			
	Huân chương			

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
		Hạng nhất		
		Hạng nhất		
		Hạng nhất		
		Hạng nhất		

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
			Chống Pháp	
			Chống Pháp	

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Thanh Oai			Chống Mỹ	
Thanh Oai			Chống Mỹ	

TenHuyen	DanhHieu	Muc		...
Ba Vì	Huân chương			
Ba Vì	Huân chương			
Ba Vì	Huân chương			
Ba Vì	Huân chương			

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Ba Vì	Huân chương	Hạng nhất		
Ba Vì	Huân chương	Hạng nhất		
Ba Vì	Huân chương	Hạng nhất		

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Ba Vì	Huân chương	Hạng nhất	Chống Mỹ	
Ba Vì	Huân chương	Hạng nhất	Chống Mỹ	

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
	Huân chương	Hạng nhất		
	Huân chương	Hạng nhất		
	Huân chương	Hạng nhất		

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
	Huân chương	Hạng nhất	Chống Mỹ	
	Huân chương	Hạng nhất	Chống Mỹ	

Để việc phân mảnh hiệu quả, một số mảnh ngang sẽ không được tham gia vào phân mảnh dọc. Điều này được thể hiện bởi Định lý 1 sau đây.

Ký hiệu

$$Size(HV) = [Card(HV) - 2] \sum_{j=1}^{hv} Size(A_j) \tag{12}$$

trong đó, $A_j,\ j = \overline{1, hv}$ là các thuộc tính không khóa của HV.

$Card(HV)$ là số bộ có trong mảnh HV.

$Size(A_j)$ là độ rộng của thuộc tính A_j.

Định lý 1. *Phân mảnh hỗn hợp HV hiệu quả khi* $Size(HV) > 0$ *hay* $Card(HV) > 2$.

Chứng minh. Do $\sum_{j=1}^{hv} Size(A_j) > 0$ nên $Size(HV) > 0$ khi $Card(HV) > 2$. Như vậy mảnh HV có ít nhất 3 bộ. Có thể dùng một mảng hai chiều hai hàng và hv cột. Một hàng để lưu trữ các thuộc tính của HV, hàng còn lại lưu trữ các giá trị tương ứng với các thuộc tính của một bộ. Như vậy, chúng ta tiết kiệm được một bộ giá trị không phải lưu trữ. ■

Hệ quả 1. *Ký hiệu* $Card(\beta_{ij})$ *là số bộ thu được khi phân mảnh ngang với mệnh đề đơn giản* $\beta_{ij},\ i = \overline{1, n_r},\ j = \overline{1, n}$ *thì khi phân mảnh chúng không cần quan tâm đến thuộc tính* A_j *mà* $Card(\beta_{ij}) \leqslant 2\ \forall i$*, nghĩa là có thể loại đi tất cả các mệnh đề đơn giản* P_j *trong* P ([2]).

Theo Định lý 1 và Hệ quả 1, thì ngay từ đầu ta đã có thể loại đi các mảnh ngang chỉ có hai bộ, vì vậy các mảnh ngang ở ví dụ trên còn được giữ lại là:

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Ba Vì				
Ba Vì				
Ba Vì				
Ba Vì				

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
	Huân chương			
	Huân chương			
	Huân chương			
	Huân chương			
	Huân chương			

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
		Hạng nhất		
		Hạng nhất		
		Hạng nhất		
		Hạng nhất		

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	
			Chống Mỹ	

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Ba Vì	Huân chương			
Ba Vì	Huân chương			
Ba Vì	Huân chương			
Ba Vì	Huân chương			

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Ba Vì	Huân chương	Hạng nhất		
Ba Vì	Huân chương	Hạng nhất		
Ba Vì	Huân chương	Hạng nhất		

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
	Huân chương	Hạng nhất		
	Huân chương	Hạng nhất		
	Huân chương	Hạng nhất		

Tiến hành phân mảnh hỗn hợp, chính là phân mảnh dọc, trong trường hợp này là phân mảnh dọc của các mảnh ngang còn được giữ lại trên (tức là tách lấy các phần có chữ ở các mảnh ngang trên).

Định lý 2. *Các mảnh hỗn hợp nhận được từ việc phân mảnh dọc các mảnh ngang có số bộ lớn hơn hai, với các thuộc tính tham gia vào mệnh đề hội sơ cấp để có mảnh ngang cùng với các bộ của mảnh ngang đó.*

Chứng minh.

Như chúng ta đã quy định việc phân mảnh hỗn hợp qua hai bước: Đầu tiên phân mảnh ngang, sau đó phân mảnh dọc các mảnh ngang đã nhận được. Số bộ lớn hơn 2 là kết quả của Định lý 1.

Còn các thuộc tính của mảnh hỗn hợp là các thuộc tính tham gia vào mệnh đề hội sơ cấp để có mảnh ngang đó cùng với các bộ của chúng.

Áp Định lý 2 vào ví dụ trên ta có các mảnh hỗn hợp sau:

TenHuyen
Ba Vì
Ba Vì
Ba Vì
Ba Vì

HV_1

DanhHieu
Huân chương
Huân chương
Huân chương
Huân chương
Huân chương

HV_2

Muc
Hạng nhất
Hạng nhất
Hạng nhất
Hạng nhất

HV_3

LyDoNhanDanhHieu
Chống Mỹ
Chống Mỹ
Chống Mỹ
Chống Mỹ

HV_4

TenHuyen	DanhHieu
Ba Vì	Huân chương
Ba Vì	Huân chương
Ba Vì	Huân chương
Ba Vì	Huân chương

HV_5

TenHuyen	DanhHieu	Muc
Ba Vì	Huân chương	Hạng nhất
Ba Vì	Huân chương	Hạng nhất
Ba Vì	Huân chương	Hạng nhất

HV_6

DanhHieu	Muc
Huân chương	Hạng nhất
Huân chương	Hạng nhất
Huân chương	Hạng nhất

HV_7

Định lý 3. *Trong tất cả các mảnh hỗn hợp nhận được thì chọn mảnh HV sao cho $Size(HV)$ lớn nhất là hiệu quả nhất.*

Chứng minh.

Áp dụng Định lý 3 ta tính kích thước của các mảnh hỗn hợp ở ví dụ trên theo công thức

$$Size(HV) = [Card(HV) - 2] \sum_{i=1}^{hv} Size(A_i),$$

với $Size(TenHuyen) = 20$, $Size(DanhHieu) = 15$, $Size(Muc) = 10$,
$Size(LyDoNhanDanhHieu) = 25$

$Size(HV_1) = [4-2] \times Size(TenHuyen) = 2 \times 20 = 40$
$Size(HV_2) = [5-2] \times Size(DanhHieu) = 3 \times 15 = 45$
$Size(HV_3) = [4-2] \times Size(Muc) = 2 \times 10 = 20$
$Size(HV_4) = [4-2] \times Size(LyDoNhanDanHieu) = 2 \times 25 = 50$
$Size(HV_5) = [4-2] \times (Size(TenHuyen) + Size(DanhHieu)) = 2 \times (20+15) = 70$
$Size(HV_6) = [3-2] \times (Size(TenHuyen) + Size(DanhHieu) + Size(Muc))$
$= 20 + 15 + 10 = 45$
$Size(HV_7) = [3-2] \times (Size(DanhHieu) + Size(Muc)) = 15 + 10 = 25$

Chọn mảnh HV_5 vì Size(HV_5) = 70 lớn nhất.

Ghi lại mảnh HV_5 bởi một nhãn.

Làm lại từ đầu với các mảnh:

R_1

TenHuyen	DanhHieu	Muc	LyDoNhanDanhHieu	...
Thanh Oai	Huân chương	Hạng nhất	Chống Mỹ	
Thanh Oai	Huy chương		Chống Mỹ	

R_2

Muc	LyDoNhanDanhHieu	...
Hạng nhất	Chống Pháp	
Hạng nhất	Chống Mỹ	
Hạng hai	Chống Pháp	
Hạng nhất	Chống Mỹ	

■

3.3. Thuật toán

3.3.1. Mô tả thuật toán

Trong quan hệ R:

+ Không xét đến các thuộc tính có tất cả các giá trị lặp nhỏ hơn 3 (như vậy các mảnh ngang đều có số bộ lớn hơn hoặc bằng 3).

+ Không xét đến các thuộc tính có miền giá trị là một hằng (vì có thể thay thuộc tính đó bởi một đại lượng hằng).

+ Tiến hành phân mảnh ngang.

+ Tiến hành phân mảnh dọc các mảnh ngang đã có để được các mảnh hỗn hợp.

+ Tính Size cho tất cả các mảnh nhận được.

+ Chọn mảnh có Size lớn nhất và đánh dấu bằng nhãn (mảng hai chiều) $HV_{\max}$.

+ Phân mảnh bảng CSDL gốc để được các bảng mới như sau:

$R1 = R\backslash\{$ Các bộ tham gia vào mảnh có nhãn $HV_{\max}\}$.

$R_2 = \{$ Mảnh ngang dùng để phân mảnh hỗn hợp để có nhãn $HV_{\max}\}\backslash\{$ Các thuộc có trong mảnh hỗn hợp có nhãn $HV_{\max}\}$

+ Tiến hành phân mảnh lại từ đầu đối với R_1 và R_2 (Tức là thay R bởi R_1, R_2 và lặp lại thuật toán).

3.3.2. Thuật toán

Nhắc lại một số thông tin liên quan thuật toán. Ký hiệu:

$n_r = Card_r(R)$ là lực lượng hàng của quan hệ R (là số bộ hiện có), $P = \{P_1, P_2, ..., P_n\}$, trong đó P_j là tập các mệnh đề đơn giản trên thuộc tính $A_j,\ j = \overline{1,n}$. Cụ thể P_j là tập các mệnh đề so sánh bằng. Tức là $P'_j = \{\beta_{ij}\},\ j = \overline{1,n}$, trong đó $\beta_{ij} = d_{ij} \in Dom(A_j)$, P'_j ký hiệu chuyển vị của P_j hay $P = \{\beta_{ij}\}_{n_r \times n}$ là ma trận cấp $n_r \times n$.

Như vậy số mệnh đề đơn giản có thể có trên quan hệ R là $m = n_r \times n$.

Ký hiệu $Q = \{q_1, q_2, ..., q_n\}$ là tập các tập mệnh đề hội sơ cấp trên R trong đó q_k được tạo ra như sau:

Ứng với mỗi số $k, k = \overline{1,n}$, gọi tập $K \subseteq \{1, 2, ..., n\}$ sao cho $Card(K) = k$. Đặt:

$$q_{ik} = \underset{j \in K}{\Lambda} \beta_{ij},\ \forall i.$$

Như vậy có thể thấy $Card(Q) = \sum\limits_{k=1}^{n} C_m^k$, trong đó $C_m^k = \dfrac{m!}{k!(m-k)!}$.

Bước 1.

$k = 1,\ \forall \beta_{ij} \in q_{i1}$

SELECT *

FROM R

WHERE β_{ij}

Được mảnh ngang H_{ij}.

Bước 2. Nếu $\forall i, j, Card(H_{ij}) \leqslant 2$ đến Bước 7 (vì không có thuộc tính nào có số các giá trị lặp lớn hơn 2). Ngược lại đến Bước 3.

Bước 3. Phân mảnh dọc từ các mảnh ngang H_{j1} trên để có các mảnh hỗn hợp và tìm mảnh hỗn hợp mà $Size(HV_1^*) = \max\ Size(HV_{j1})$. Ký hiệu $HV_{\max} = HV_1^*,\ \max = Size(HV_1^*)$. Sang Bước 4.

Bước 4.

$k = k + 1$

$i = 0$

Sang Bước 5.

Bước 5.

$i = i + 1$

SELECT *

FROM R

WHERE q_{ik}

Được mảnh ngang H_{ik}.

Nếu $Card(H_{ik}) <= 2$ đến Bước 6.

Ngược lại, phân mảnh dọc từ các mảnh ngang H_{jk} trên để có các mảnh hỗn hợp và tìm mảnh hỗn hợp mà $Size(HV_{ik}^*) = \max(HV_{ik})$.

Nếu $\max < Size(HV_{ik}^*)$

$\max = Size(HV_{ik}^*)$

$HV_{\max} = HV_{ik}^*$

Sang Bước 6.

Bước 6.

Nếu $i < n_r$ quay lại Bước 5.

Ngược lại, nếu $k \leqslant n$ quay lại Bước 4

Ngược lại, gán nhãn $HV_{\max}$ và đặt:

$R_1 = R\backslash\{$ Các bộ tham gia vào mảnh có nhăn $HV_{\max}\}$.

$R_2 = \{$ Mảnh ngang dùng để phân mảnh hỗn hợp để có nhãn $HV_{\max}\{$Các thuộc có trong mảnh hỗn hợp có nhãn $HV_{\max}\}$.

Thay R bởi R_1, lặp lại Bước 1.

Thay R bởi R_2, lặp lại Bước 1.

Bước 7. Cho ra các mảnh (nếu có). Kết thúc thuật toán.

Định lý 4.

1) *Số mảnh hỗn hợp có hoặc không có, nếu có thì không vượt quá int*$(Card(R) \times n/3)$*, trong đó n là số thuộc tính không khóa của R.*

2) *Thuật toán kết thúc sau một số hữu hạn bước.*

Chứng minh.

1) Rõ ràng là mảnh hỗn hợp nhỏ nhất là mảnh chỉ có một thuộc tính và một giá trị. Vì vậy số mảnh hỗn hợp nhiều nhất trong trường hợp này là $(Card(R) \times n$. Nhưng theo Định lý 1 thì các mảnh phải có số bộ lớn hơn hoặc bằng 3 mới có hiệu quả. Vì vậy số mảnh nhiều nhất cũng chỉ bằng int$(Card(R) \times n/3)$.

2) Điều này được suy ra từ 1). ■

4. KẾT LUẬN

Việc tiết kiệm không gian lưu trữ thông tin, rút ngắn thời gian truy nhập dữ liệu và cực tiểu hoá chi phí khi cấp phát dữ liệu là mục đích hàng đầu của người lập trình và người sử dụng. Bài báo này giải quyết vấn đề trên bằng cách gom nhóm và dùng bí danh để đại diện cho nhóm các bộ có các thuộc tính có các giá trị lặp lại.

Công trình đã được ứng dụng để quản lý công tác làm báo cáo như: Báo cáo đột xuất, Báo cáo định kỳ của các huyện thị, sở ngành (Tháng, Quý, 6 tháng,...). Quản lý thi đua khen thưởng, người có công cách mạngtrong tỉnh tại các thị xã, huyện và văn phòng UBND tỉnh Hà Tây.

5. HƯỚNG PHÁT TRIỂN

Nghiên cứu mô hình cấp phát để làm giảm chi phí, thời gian xử lý và không gian lưu trữ,... trên cơ sở các mảnh hỗn đã được tạo ra, nghĩa là ta cần nghiên cứu mô hình:

min(Total Cost)

Với các ràng buộc

Thời gian xử lý các câu truy vấn

Không gian lưu trữ

...

TÀI LIỆU THAM KHẢO

[1] M. Tamer Ozsu, Patrick Valduriez, *Nguyên lý các hệ cơ dữ liệu phân tán*, Trần Đức Quang biên dịch, NXB Thống kê, 1999.

[2] Lê Tiến Vương, *Nhập môn cơ sở dữ liệu quan hệ*, NXB Thống Kê, 2000.

[3] Đỗ Xuân Lôi, *Cấu trúc dữ liệu và giải thuật*, NXB Khoa học và Kỹ thuật, 1996.

[4] Robert Sedgewick, *Cẩm nang thuật toán*, Vol.1 and Vol.2, NXB Khoa học và Kỹ thuật, 2001.

Nhận bài ngày 20 - 12 - 2002
Nhận lại sau sửa ngày 15 - 3 -2007