Tạp chí Tin học và Điều khiển học, T. 14, S. 4 (1998) (19–24)

THUẬT TOÁN TÍNH TỔNG THEO MIỀN TRONG XỬ LÝ PHÂN TÍCH TRỰC TUYẾN ĐỐI VỚI CÁC KHO DỮ LIỆU

ĐOÀN VĂN BAN

Abstract. A range query applies an aggregation operation over all selected cells an OLAP data cube where the selection is specified by providing ranges of values for numeric dimensions. We present fast algorithm for range queries for the most popular aggregation operation: SUM.

1. GIỚI THIỆU

Kho dữ liệu (Data Warehouse) được xây dựng từ các cơ sỡ dữ liệu (CSDL) tác nghiệp (khác nhau) nhằm hỗ trợ cho việc ra quyết định. *Kho dữ liệu thường rất lớn, lưu trữ những dữ liệu có tính lịch sử theo thời gian, theo chủ điểm, không thay đổi (không được cập nhật) và được tích hợp từ nhiều nguồn dữ liệu khác nhau* ([1], [4], [6]). Mô hình thích hợp cho các kho dữ liệu là *khối dữ liệu* (Data Cube) hay còn gọi là *CSDL nhiều chiều* (những phương diện khác nhau hay là những thuộc tính độc lập) ([1], [7]). Các kho dữ liệu lớn, đa chiều thường có chứa nhiều thông tin (tri thức) ẩn kín mà những công cụ truyền thống như kỹ thuật truy vấn SQL rất khó và nhiều khi không phát hiện ra được. Ví dụ ông Giám đốc xí nghiệp muốn biết: "Những mặt hàng nào bán chạy nhất, trong vùng nào, tháng nào trong năm và những tầng lớp khách hàng (nhóm tuổi, ngành nghề, v.v...) nào tiêu thụ nhiều nhất ?". Không dễ gì có được câu trả lời cho những câu hỏi nhiều chiều (4 chiều trong câu hỏi trên: mặt hàng, vùng, thời gian, và nhóm tuổi) bằng cách sử dụng các kỹ thuật truy vấn truyền thống trong các mô hình dữ liệu quan hệ, ví dụ SQL, mà phải dựa trên những kết quả *phân tích dữ liệu nhiều chiều*. Hơn thế nữa, yêu cầu của người sử dụng lại thường xuyên thay đổi, đòi hỏi câu trả lời phải được xử lý theo thứ tự khác nhau: lúc thì theo vùng, khi thì theo thời gian, khi khác lại theo nhóm tuổi, v.v... Do vậy đòi hỏi phải thực hiện *xử lý phân tích trực tuyến* (OLAP - On-Line Analytical Processing do Codd đề xuất [3]) trên những dữ liệu lớn và hỗn hợp. Chúng ta có hai cách chính truy nhập trực tiếp vào kho dữ liệu. *Cách thứ nhất* là truy nhập trực tiếp vào dữ liệu trong kho thông qua các *khung nhìn* (view) *nhiều chiều* và thể hiện nó như là cấu trúc nhiều chiều phục vụ cho việc phân tích và lập báo cáo ở các trạm làm việc. Để thực hiện hiệu quả OLAP trên các khung nhìn dữ liệu, người ta thường tập trung xây dựng các thuật toán để chọn tự động các bảng tổng hợp và chỉ số hóa các khung nhìn ([5], [8], [9]). *Cách thứ hai* là phân tích trực tiếp các khối dữ liệu nhiều chiều được tạo lập từ các kho dữ liệu và tạo ra khả năng tổng hợp, gộp chung, hỗ trợ cho việc ra quyết định về dự báo, phân tích xu thế phát triển và phân tích thống kê ([2], [3], [7]).

Trong bài báo này chúng tôi giới thiệu một số thuật toán thực hiện những phép toán sử dụng thường xuyên trong OLAP, phép toán thực hiện để trả lời cho những truy vấn dữ liệu theo miền như phép SUM xác định tổng theo miền dữ liệu của khối dữ liệu.

2. TRUY VẤN THEO MIỀN TRONG OLAP VỚI KHỐI DỮ LIỆU

Mô hình phổ biến cho các ứng dụng của OLAP là CSDL nhiều chiều hay còn gọi là *khối dữ liệu* (Data Cube). Từ các *kho dữ liệu* (Data Warehouse), ta có thể chọn ra một số thuộc tính cần thiết để tạo ra khối dữ liệu phục vụ cho việc phân tích, khai thác thông tin hỗ trợ cho việc ra quyết định. Một số trong những thuộc tính đó chứa những giá trị đo đếm được mà người sử dụng quan tâm nên thường được gọi là thuộc tính độ đo (metric), còn những thuộc tính khác thường được gọi là thuộc tính chiều (hay thuộc tính chức năng). Những thuộc tính độ đo cho phép thực hiện được

những phép kết hợp, gộp chung (ví dụ tính tổng) hay so sánh được để xác định những giá trị cực đại, cực tiểu trong từng vùng dữ liệu. Như vậy khối dữ liệu có thể xem như là các bảng d-chiều, mà các chỉ số của nó tương ứng với những thuộc tính về chiều, còn các giá trị của mỗi phần tử là giá trị thuộc tính độ đo tương ứng ([1], [6], [7]).

Ví dụ 1: Xét khối dữ liệu BH được xây dựng từ các kho dữ liệu của hãng bảo hiểm BV được tổ chức theo 4 chiều: tuổi, năm, tỉnh_thành và loại_hình_BH. Tuổi có giá trị từ 1 đến 100, năm tính từ năm 1985 đến 1998, tỉnh_thành bao gồm 61 tỉnh thành của cả nước còn loại_hình_BH thì gồm: bảo hiểm bất động sản, ô tô và y tế. Khối dữ liệu BH sẽ có $101 * 14 * 61 * 3$ ô lưu trữ dữ liệu về tiền đóng bảo hiểm tương ứng với tổ hợp của 4 thuộc tính trên. Chúng ta có thể thực hiện *phép gộp* (aggregation operation) như SUM - tính tổng và MAX - xác định giá trị cực đại để trả lời cho những câu hỏi truy vấn theo miền, ví dụ như: "Tìm số tiền đóng bảo hiểm về ô tô của khách hàng tuổi từ 30 đến 50, trong những năm 1990 đến 1995 ở trong cả nước?"

Khi thực hiện phân tích dữ liệu để hỗ trợ ra quyết định, chúng ta thường phải thực hiện những *truy vấn theo miền* (range queries) để trả lời cho những cây hỏi tương tự như trên. Mục tiêu của chúng ta là xây dựng những thuật toán hiệu quả, thực hiện nhanh những phép truy vấn theo miền (phép tính SUM, MAX theo miền) với không gian lưu trữ là tuyến tính.

2.1. Mô hình dữ liệu

Trước tiên chúng ta hãy nghiên cứu mô hình của khối dữ liệu và dạng biểu diễn của các câu hỏi truy vấn theo miền. $D = \{1, 2, ..., d\}$ tập các chỉ số tương ứng với thuộc tính chiều. Khối dữ liệu d-chiều được biểu diễn bằng bảng d-chiều A có kích thước $n_1 \times n_2 \times \ldots \times n_d$, với $2 \leq n_j$, $j \in D$. Giả thiết rằng các chỉ số của bảng luôn bắt đầu từ 0. Để cho tiện chúng ta gọi mỗi phần tử của bảng là một ô. Kích thước của bảng A sẽ là

$$N = \prod_{j=1}^{d} n_j \,. \tag{0}$$

Chúng ta phân tích bảng A với kích thước N (d-chiều) để có được câu trả lời cho những câu hỏi (truy vấn) liên quan đến nhiều chiều. Trong thực tế, mỗi chiều của A sẽ là một miền phạm vi tương ứng với thuộc tính của khối dữ liệu. Để cho đơn giản về ký hiệu và công thức, chúng ta có thể giả thiết (không mất tính tổng quát) rằng tồn tại một ánh xạ đơn giản từ miền xác định của thuộc tính (không nhất thiết phải liên tục) sang miền chỉ số $(0, ..., n_k,\ k \in D)$.

Các kỹ thuật thực hiện truy vấn theo miền có thể áp dụng cho những toán tử nhị nguyên $\oplus$ có toàn tử đảo Θ, nghĩa là $a \oplus b \,\Theta\, b = a$, với mọi a, b trong miền xác định, ví dụ: phép $+$ có phép đảo là $-$, phép $*$ có phép đảo là $/$, v.v... Ở đây chúng ta quan tâm đến phép tính tổng SUM theo miền dựa trên các toán tử $(+, -)$, bởi vì nó là phép gộp thường xuyên sử dụng nhất trong các ứng dụng của OLAP (thuật toán cũng sẽ hoàn toàn tương tự đối với những phép toán $\oplus$ có toán tử đảo Θ như ở trên).

2.2. Thuật toán tính tổng SUM theo miền

Vấn đề tính tổng SUM theo miền của các chiều được đánh chỉ số từ l_j đến h_j, $j \in D$ trong khối dữ liệu A có thể định nghĩa một cách hình thức như sau:

$$\mathrm{SUM}_A(l_1 : h_1, ..., l_d : h_d) = \sum_{i_1=l_1}^{h_1} \ldots \sum_{i_d=l_d}^{h_d} A[i_1, \ldots, i_d] \,, \tag{1}$$

trong đó $0 \leq l_i \leq h_i \leq n_i$, $i \in D$.

Theo công thức (1) chúng ta có thể dễ dàng tính được $\mathrm{SUM}_A\ (l_1 : h_1, \ldots, l_d : h_d)$ với độ phức tạp về thời gian là $O(N)$, N kích thước của A.

Để tăng tốc độ xử lý phép gộp theo miền chúng ta có thể sử dụng *tổng phần đầu* (Prefix_sum), phép toán đã được xây dựng trong các CSDL ([2]) và ở đây sẽ được phát triển, mở rộng cho khối dữ liệu.

Bảng P được gọi là bảng tính trước các phần đầu của khối dữ liệu (bảng A) và được xác định theo công thức sau:

$$P[x_1, ..., x_d] = \text{SUM}_A(0 : x_1, ..., 0 : x_d) = \sum_{i_1=0}^{x_1} \cdots \sum_{i_d=0}^{x_d} A[i_1, ..., i_d]. \quad (2)$$

$P[x_1, ..., x_d]$ là tổng của tất cả các phần tử đầu trong các miền xác định bởi: thuộc tính thứ nhất từ 0 đến x_1, thuộc tính thứ 2 từ 0 đến x_2,..., thuộc tính cuối cùng từ 0 đến x_d.

Ví dụ 2: khi $d = 2$, bảng dữ liệu A có $n_1 = 8$ và $n_2 = 4$ được xác định như sau:

Bảng A

Chỉ số	0	1	2	3	4	5	6	7
0	2	5	4	1	6	2	1	5
1	5	2	6	7	4	1	8	9
2	1	3	6	8	4	2	7	10
3	4	5	7	6	12	5	6	2

Chúng ta tính trước các tổng phần đầu của A theo (2) và lưu vào bảng P.

$$P[x, y] = \text{SUM}_A(0 : x, 0 : y) = \sum_{i=0}^{x} \sum_{j=0}^{y} A[i, j].$$

Bảng P

Chỉ số	0	1	2	3	4	5	6	7
0	2	7	11	12	18	20	21	26
1	7	14	24	32	42	45	54	68
2	8	18	34	50	64	69	85	109
3	12	27	50	72	98	108	130	156

Định lý sau giúp chúng ta có được thuật toán tính tổng SUM theo miền của A từ 2^d phần tử tương ứng của P.

Trước tiên chúng ta qui định: $P[x_1, ..., x_d] = 0$ nếu có $k \in D$ và $x_k = -1$;

$$\forall j \in D \quad s(j) = \begin{cases} 1 & \text{nếu } x_j = h_j \\ -1 & \text{nếu } x_j = l_j - 1 \end{cases}$$

Định lý.

$$\text{SUM}_A(l_1 : h_1, ..., l_d : h_d) = \sum_{\forall x_j \in \{l_j - 1, h_j\}, j \in D} \left\{ \left(\prod_{i=1}^{d} s(i) \right) * P[x_1, ..., x_d] \right\} \quad (3)$$

với $0 \le l_i \le h_i \le n_i$, $i \in D$.

Chứng minh. Để chứng minh (3) chúng ta chứng minh công thức tổng quát sau:

$$\text{SUM}_A(l_1 : h_1, ..., l_t : h_t, 0 : x_{t+1}, ..., 0 : x_d)$$
$$= \sum_{\forall x_j \in \{l_j - 1, h_j\}, 1 \le j \le t} \left\{ \left(\prod_{i=1}^{t} s(i) \right) * P[x_1, ..., x_d] \right\} \quad (4)$$

Chúng ta nhận thấy ngay rằng (3) là trường hợp đặc biệt của (4) khi $t = d$. Công thức (4) có thể được chứng minh thông qua qui nạp toán học theo t.

Với $t = 1$, chúng nhận thấy (4) được suy trực tiếp từ định nghĩa (2). Giả thiết rằng (4) đúng với $t = k$ và $1 \le k \le d$, nghĩa là:

$$\begin{aligned} &\mathrm{SUM}_A(l_1 : h_1, ..., l_k : h_k, 0 : x_{k+1}, ..., 0 : x_d) \\ &= \sum_{\forall x_j \in \{l_j - 1, h_j\}, 1 \le j \le k} \Big\{ \Big(\prod_{i=1}^{k} s(i) \Big) * P[x_1, ..., x_d] \Big\} \end{aligned} \tag{5}$$

Chúng ta cần chứng minh (4) đúng với $t = k + 1$.

Thay $x_{k+1} = l_{k+1}$ vào (5) chúng ta có

$$\begin{aligned} &\mathrm{SUM}_A(l_1 : h_1, ..., l_k : h_k, 0 : l_{k+1}, 0 : x_{k+2}, ..., 0 : x_d) \\ &= \sum_{\forall x_j \in \{l_j - 1, h_j\}, 1 \le j \le k} \Big\{ \Big(\prod_{i=1}^{k} s(i) \Big) * P[x_1, ..., x_d] \Big\} \end{aligned} \tag{7}$$

Tương tự thay $x_{k+1} = h_{k+1}$ vào (5) chúng ta sẽ có

$$\begin{aligned} &\mathrm{SUM}_A(l_1 : h_1, ..., l_k : h_k, 0 : h_{k+1}, 0 : x_{k+2}, ..., 0 : x_d) \\ &= \sum_{\forall x_j \in \{l_j - 1, h_j\}, 1 \le j \le k} \Big\{ \Big(\prod_{i=1}^{k} s(i) \Big) * P[x_1, ..., x_d] \Big\} \end{aligned} \tag{8}$$

Để đơn giản, chúng ta ký hiệu những hạng thức trong các công thức từ (6) đến (8) lần lượt là $T1$, $T2$, $T3$, $T4$. Từ đó chúng ta có

$$\begin{aligned} &\mathrm{SUM}_A(l_1 : h_1, ..., l_k : h_k, l_{k+1} : h_{k+1}, 0 : x_{k+2}, ..., 0 : x_d) = T3 - T1 = T4 - T2 \\ &= \sum_{\forall x_j \in \{l_j - 1, h_j\}, 1 \le j \le k+1} \Big\{ \Big(\prod_{i=1}^{k+1} s(i) \Big) * P[x_1, ..., x_d] \Big\}, \end{aligned}$$

nghĩa là (4) cũng đúng với $t = k + 1$. Đó là điều cần phải chứng minh.

Ví dụ 3: Áp dụng định lý trên để tính $\mathrm{SUM}_A(2 : 4, 1 : 2)$ của bảng A và P từ ví dụ 2:

$$\mathrm{SUM}_A(2 : 4, 1 : 2) = P[4, 2] - P[4, 0] - P[1, 2] + P[1, 0] = 64 - 18 - 18 + 7 = 35.$$

Theo định lý nêu trên chúng ta có thể xây dựng được thuật toán thực hiện phép gộp theo miền SUM với độ phức tạp $O(2^d)$ tốt hơn thuật toán thực hiện theo định nghĩa ban đầu (với độ phức tạp $O(N)$), trong đó N là kích thước của khối dữ liệu và thường là rất lớn. Lưu ý là để có cơ sở so sánh độ phức tạp về thời gian thực hiện của các thuật toán, chúng ta xem tốc độ truy nhập vào các phần tử của bảng A và các bảng P là như nhau.

Để áp dụng được định lý trên thì chúng ta phải tính bảng P (tổng các phần đầu của A) và thuật toán tính P theo định nghĩa (2) có độ phức tạp $O(N^2)$. Chúng ta xây dựng thuật toán mới thực hiện phép gộp theo miền với độ phức tạp $O(d * N)$ như sau.

Thuật toán Prefix_sum /* Thực hiện trong d bước */

Input: d /* Số chiều của khối */

n_k, $1 \le k \le d$ /* Chỉ số của mỗi chiều */

$A[x_1, ..., x_d]$, $0 \le x_k \le n_k$, $1 \le k \le d$ /* Khối dữ liệu */

Output: $P[x_1, ..., x_d]$, $0 \le x_k \le n_k$, $1 \le k \le d$ /* Tổng phần đầu của A */

/* Tính Prefix_sum để thực hiện cộng dồn theo chiều thứ nhất và lưu kết quả vào P (ký hiệu P_1) */

$$P[0, x_2, ..., x_d] = A[0, x_2, ..., x_d],\ 0 \le x_k \le n_k,\ 2 \le k \le d$$

$$P[i, x_2, ..., x_d] = A[i, x_2, ..., x_d] + P[i-1, x_2, ..., x_d],\ 0 \le x_k \le n_k,\ 2 \le k \le d$$

và $1 \le i \le n_1$

/* Bước thứ k, $2 \le k \le d$, tính Prefix_sum để thực hiện cộng dồn theo chiều thứ k và lưu kết quả vào P (ký hiệu P_k) */

for $k = 2$ to d do

$$P[x_1, x_2, ..., x_{k-1}, i, x_{k+1}, ..., x_d] = P[x_1, x_2, ..., x_{k-1}, i, x_{k+1}, ..., x_d] + P[x_1, x_2, ..., x_{k-1}, i-1, x_{k+1}, ..., x_d],$$

$$0 \le x_m \le n_m,\ 1 \le m \le k-1,\ k+1 \le m \le d \text{ và } 1 \le i \le n_k$$

Nhận xét:

a/ Thuật toán trên thực hiện chính xác bảng P, bảng tổng các phần đầu của A sau d bước thực hiện. Trong mỗi bước, thuật toán thực hiện N (kích thước của bảng A) phép cộng dồn, nên thuật toán Prefix_sum có độ phức tạp tính toán là $O(d * N)$.

b/ Sau khi tính được bảng P, chúng ta có thể xóa bảng A vì có thể sử dụng P để tính lại theo định lý như sau:

$$A[x_1, ..., x_d] = \text{SUM}_A(x_1 : x_1, ..., x_d : x_d) = \sum_{\forall y_k \in \{x_k - 1, x_k\},\ 1 \le k \le d} \left\{ \prod_{i=1}^{d} s(i) * P[y_1, y_2, ..., y_d] \right\}, \quad (10)$$

$A[x_1, ..., x_d]$ được tính lại từ P với 2^d phép cộng (hay trừ). Do vậy khi các khối dữ liệu có số chiều tương đối nhỏ thì thuật toán Prefix_sum với cách lưu trữ P thay cho khối dữ liệu A là rất hiệu quả cả về không gian lưu trữ lẫn tốc độ thực hiện.

Ví dụ 4: Xây dựng bảng P từ bảng A ở ví dụ 2 theo thuật toán trên

/* Tính Prefix_sum để thực hiện cộng dồn theo chiều thứ nhất (cộng dồn theo hàng) và lưu kết quả vào P (ký hiệu P_1) */

Bảng P_1

Chỉ số	0	1	2	3	4	5	6	7
0	2	7	11	12	18	20	21	26
1	5	7	13	20	24	25	33	42
2	1	4	10	18	22	24	31	41
3	4	9	16	22	34	39	45	47

/* Tính Prefix_sum để thực hiện cộng dồn theo chiều thứ hai (cộng dồn theo cột) và lưu kết quả vào P (ký hiệu P_2) */

Bảng P_2

Chỉ số	0	1	2	3	4	5	6	7
0	2	7	11	12	18	20	21	26
1	7	14	24	32	42	45	54	68
2	8	18	34	50	64	69	85	109
3	12	27	50	72	98	108	130	156

Dễ dàng nhận thấy P_2 cũng chính là bảng P ở ví dụ 2.

3. KẾT LUẬN

Đối với các kho dữ liệu (CSDL nhiều chiều) chúng ta phải sử dụng phương pháp OLAP mới phát hiện được nhanh, chính xác những thông tin (tri thức) cần thiết cho việc ra quyết định. Từ các bảng dữ liệu nhiều chiều thường xuyên sử dụng để xử lý phân tích (trực tuyến) dữ liệu tổng hợp từ các kho dữ liệu được lưu trữ nhằm trả lời nhanh cho những câu hỏi của người sử dụng và hỗ trợ cho việc ra những quyết định, chúng ta có thể sử dụng thuật toán nêu trên để tính các bảng tổng phần đầu tương ứng (với độ phức tạp tính toán là tuyến tính - $O(d * N)$). Mặt khác, các dữ liệu của kho dữ liệu có tính lịch sử, không thay đổi, không được cập nhật nên chúng ta có thể xóa ngay bảng A (nhận xét b/) sau khi tính được bảng P tổng các phần đầu của nó. Do vậy độ phức tạp về không gian bộ nhớ của hai cách thực hiện trên là như nhau, nhưng theo thuật toán trên chúng ta sẽ có được những kết quả theo yêu cầu nhanh hơn, giảm từ $O(N)$ xuống $O(2^d)$, với d là số nhiều , thường là tương đối nhỏ (thuật toán có hiệu quả khi $d < 10$), còn N là kích thước của khối dữ liệu, thường là rất lớn theo công thức (0). Theo cách tiếp cận trên, chúng ta có thể xây dựng được các thuật toán thực hiện hiệu quả những phép truy xuất khác như: phép xác định phần tử cực đại MAX_INDEX, phép tính liệt kê số phần tử COUNT, phép lấy trung bình cộng AVERAGE, v.v... theo miền trong các khối dữ liệu.

TÀI LIỆU THAM KHẢO

[1] Adriaans P., Zantinge D., *Data Mining*, Adison Wesley Longman, 1996.

[2] Chen M. C. and McName L. P., The Data Model and Access Method of Summary Data Managemnt, *IEEE Transactions on Knowledge and Data Engineering* **1** (4) (1989) 519-529.

[3] Codd E. F., *Providing OLAP (On-Line Anatytical Processing) to User-analysts: An IT mandate*, Technical Report, E. F. Codd and Associates, 1993.

[4] Đoàn Văn Ban," Phương pháp thiết kế và khai thác kho dữ liệu", Đề tài Trung tâm KHTN & CNQG TT96-9704, Viện Công nghệ thông tin, 1997, 8-17.

[5] Đoàn Văn Ban, Nguyễn Tuệ, Thuật toán chỉ số hóa khung nhìn trong xử lý phân tích trực tuyến kho dữ liệu, *Tuyển tập các công trình Hội nghị khoa học Đại học Khoa học tự nhiên*, Hà Nội, 4-1998, (48-61).

[6] Inmon W. H., Hackathorn R. D., *Using the Data Warehouse*, John Wisley & Sons, 1994.

[7] Mumick I. S., Quass D., Mumick B. S., Maintenance of Data Cube and Summary Tables in a Warehouse, *Proceeding ACM SIGMOD, Inter. Confer. on Management of Data*, May 13-15, 1997, (100-111).

[8] Patrick O'Neil and Quass D., Improved Query Performance with Variant Indexes, *Proceeding ACM SIGMOD, Inter. Confer. on Management of Data*, May 13-15, 1997, (38-49).

[9] Quass D., Widom J., On-Line Warehouse View Maintenance, *Proceeding ACM SIGMOD, Inter. Confer. on Managemment of Data*, May 13-15, 1997.

Nhận bài ngày 20 - 1 - 1998

Viện Công nghệ thông tin, Trung tâm KHTN và CNQG.