ÁP DỤNG THUẬT TOÁN TỐI ƯU VIỆC LỰA CHỌN KHUNG NHÌN TRONG KHO DỮ LIỆU

Nguyễn Đăng Cao
Trường Đại học Kinh tế Tp. HCM
(Bài nhận ngày 24 tháng 01 năm 2003)

TÓM TẮT: *Với cách tiếp cận kho dữ liệu (data warehouse-**Dwh**) trên một cơ sở dữ liệu (CSDL) phân tán, thông tin từ mỗi nguồn liên quan sẽ được rút trích và lưu lại trong một Dwh [1]. Khi xảy ra một truy vấn, nó sẽ được thi hành trực tiếp ở Dwh. Bài báo này giới thiệu : một thuật toán do J.Ullman và nhóm cộng sự xây dựng để lựa chọn tối ưu các khung nhìn cần lưu tại Dwh sao cho không gian lưu trữ (**KGLT**) dữ liệu giảm xuống mà thời gian truy vấn (**TGTV**) gần như không đổi [3], một số cải tiến của thuật toán [4]. Chương trình thử nghiệm được thực hiện bằng ngôn ngữ Java.*

I. MỞ ĐẦU

a) Các khái niệm [1,2]:

-Khung nhìn (View-***V***): là một quan hệ được định nghĩa như một hàm trên một tập quan hệ cơ sở từ nhiều nguồn. Hàm này có nhiệm vụ tính toán lại mỗi khi V được tham khảo.

-Khung nhìn đã cụ thể (Materialized View - ***MV***): là V nhưng lưu lại các bộ tại *Dwh* sau khi đã tính toán và kèm theo cấu trúc chỉ mục.

-Khối dữ liệu (Data cube –***DC***): được biểu diễn trong một không gian n-chiều, mỗi chiều biểu diễn một thuộc tính của MV. Tại mỗi đỉnh trong khối biểu diễn một hàm tổng (sum, avg,...) trên 1 hay nhiều thuộc tính của MV.

Ví dụ 1: giả sử một hệ thống ngân hàng sử dụng lược đồ CSDL như sau để quản lý số dư tài khoản khách hàng và các giao dịch xảy ra trong ngày [4]:

(1) Khach (**Makh**, Tenkh, Quoctich, Diachi) (***K***): quản lý thông tin về khách hàng.

(2) KhTk(**Tkkh**, Loaitk, Makh, Sodu, Psco, Psno) (***Kt***): quản lý thông tin về tài khoản của mỗi khách hàng: số tài khoản, loại tài khoản, mã khách, số dư , phát sinh nợ, có.

(3) Giaodich(**Sogd**, Ngaygd, Loaigd, Tkkh, Tknh, Sotien) (***G***): quản lý các giao dịch xảy ra, bao gồm số thứ tự giao dịch, loại giao dịch, số tài khoản khách hàng, số tài khoản ngân hàng, số tiền.

(4) NhTk(**Tknh**,Matien,Sodu, Psco, Psno)(***Nt***): quản lý thông tin tài khoản của ngân hàng.

Thuộc tính đầu tiên của mỗi quan hệ là khóa của quan hệ. Ta có các ràng buộc toàn vẹn tham chiếu (***RBTVTC***): (1) Kt.Makh->K.Makh (2) G.Tkkh->Kt.Tkkh (3)G.Tknh->Nt.Tknh

Các quan hệ được phân mảnh ngang theo mã chi nhánh, nghĩa là mỗi chi nhánh chỉ lưu thông tin của chi nhánh đó.

Xét một giao dịch xảy ra tại một chi nhánh với các thông tin gồm: loại tiền (*matien*), tài khoản ngân hàng (*tknh*), tài khoản khách hàng (*tkkh*) với một số tiền (*sotien*) xác định. Ta quan tâm đến 3 đại lượng (ba chiều) là : *matien, tknh* và *tkkh*. Vì thế mỗi ô (m,*n*,*t*) trong *DC* 3 chiều này là tổng trị giá đã giao dịch của loại tiền *m* được thực hiện trên tài khoản ngân hàng *n* với tài khoản khách hàng đối ứng *t*.

Người sử dụng cũng có thể hỏi tổng trị giá của loại tiền *m* đã giao dịch trên tài khoản khách hàng *t* (không quan tâm đến tài khoản ngân hàng). Để thực hiện câu hỏi này, ta có thể đưa thêm vào những ô có giá trị *ALL* ứng với mỗi chiều. Ví dụ ta có ô (*m,ALL,t*) được thêm vào chiều *tknh.*

Một số nhận xét:

- Trong *DC*, giá trị của nhiều ô có thể được tính từ những ô khác. Ta gọi những ô như thế là những *ô phụ thuộc*. Ví dụ, ta có thể tính ô (*m,ALL,t*) ở trên bằng cách cộng tất cả những ô (m,n_1,t),..., $(m,n_{N_{tknh}},t)$, trong đó N_{tknh} là số tài khoản ngân hàng tương ứng. Như vậy, càng nhiều ô được cụ thể thì tốc độ truy vấn càng nhanh. *Tuy nhiên không gian bộ nhớ không cho phép ta lưu hết tất cả các ô trong DC, do đó điều quan trọng là chọn những ô nào để lưu lại.*
- Ta sẽ sử dụng mô hình quan hệ để lưu *DC*. Tập các ô của *DC* được lưu trong các quan hệ. Tất cả những ô khớp vị trí của giá trị ALL được đặt trong cùng một tập. Mỗi tập chuyển thành một truy vấn SQL. Ví dụ giá trị trong tập các ô (_,ALL,_) được thể hiện bởi truy vấn sau:

SELECT matien, tkkh, *SUM*(sotien) *AS* Tong *FROM Giaodich* GROUP BY matien, tkkh

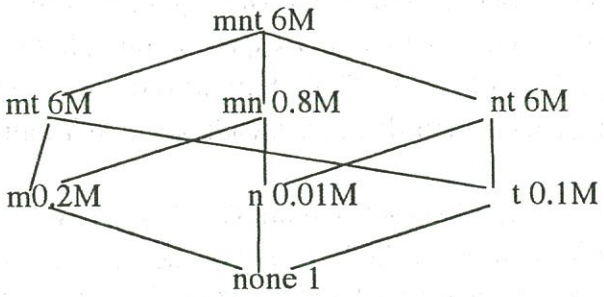
Trong mệnh đề GROUP BY sẽ không có thuộc tính ứng với giá trị 'ALL', do đó các truy vấn SQL của tập các ô khác nhau chỉ phân biệt thông qua những thuộc tính được *group by*. Ta tiến đến một khái niệm là: *quyết định chọn tập ô nào cần lưu tương đương với việc chọn MV nào để cụ thể.*

b) Đặt vấn đề

Ví dụ 2: Xét 3 thuộc tính: *matien, tknh, tkkh*. Như thế có 8 khả năng nhóm các thuộc tính. Ta liệt kê các MV và số bộ giả định :

1. matien, tknh,tkkh	(6 triệu bộ - 6M)	2. *matien,tkkh*	(6M)
3. matien,tknh	(0.8M)	4. *tknh,tkkh*	(6M)
5. matien	(0.2M)	6. *tknh*	(0.01M)
7. tkkh	(0.1M)	8.*none*	(1)

none : không có thuộc tính nào được nhóm trong mệnh đề GROUP BY. *Hình 1* thể hiện một lưới (ta sẽ bàn đến khái niệm lưới trong *mục II*) gồm 8 MV này.



Hình 1: *8 MV có thể tạo bởi nhóm các thuộc tính matien (m), tknh (n), tkkh (t).*

Giả sử một truy vấn yêu cầu thông tin giao dịch nhóm theo thuộc tính *matien*. Ta có thể sử dụng MV 5 (thời gian là 0.2M) hoặc MV 2 (thời gian là 6M) để trả lời. Qua đó ta thấy:

1.Nên cụ thể bao nhiêu MV thì hợp lý ?

2.Với một không gian S cho trước, chọn những MV nào để cụ thể nhằm giảm tối thiểu thời gian truy vấn.

3.Ngược lại, nếu thời gian truy vấn có thể gia tăng một sai số X% so với thời gian truy vấn trung bình, thì không gian S cần sử dụng là bao nhiêu ?
Chẳng hạn ở ví dụ 2, nếu ta cụ thể toàn bộ DC thì KGLT sẽ khoảng 19M. Nhận xét:
-Ta bắt buộc phải cụ thể MV1 (mnt), vì nó không thể suy được từ MV khác.
-Nếu cụ thể MV 2 (mt) thì một truy vấn sử dụng MV này đòi hỏi phải xử lý 6M bộ. Trong khi đó ta có thể sử dụng MV1 và cũng chỉ xử lý 6M bộ => không cần cụ thể MV 2.
-Tương tự, ta cũng không cần cụ thể MV 4 (nt)
-Các MV còn lại cần cụ thể vì KGLT không đáng kể để cho TGTV nhanh.
=> KGLT sẽ giảm đáng kể (chỉ còn khoảng 7M, giảm hơn 60%) mà TGTV tương đương.

II KHUNG LƯỚI [2,3]

Những truy vấn được đặc trưng bởi các thuộc tính trong mệnh đề GROUP BY, do đó ta biểu thị một truy vấn bằng cách đặt những thuộc tính nhóm trong dấu ngoặc đơn.Ví dụ truy vấn có thuộc tính nhóm là *matien* và *tkkh* được biểu thị là (*matien,tkkh*).

a) Quan hệ phụ thuộc trên những truy vấn

Xét 2 truy vấn Q_1 và Q_2, ta nói $Q_1 \preceq Q_2$ nếu và chỉ nếu Q_1 có thể được trả lời chỉ bằng kết quả của Q_2. Khi đó ta nói Q_1 phụ thuộc Q_2. Ví dụ ở *mục I.b*, truy vấn (*matien*) có thể được trả lời chỉ bằng kết quả của truy vấn (*matien, tkkh*). Như thế (*matien*) $\preceq$ (*matien,tkkh*).

b) Các chú thích

Ký hiệu $(L,\preceq)$ là một lưới L với các thành phần là những MV và quan hệ phụ thuộc $\preceq$. Với thành phần a và b của $(L,\preceq)$, $a \prec b$ nghĩa là $a \preceq b$ và $a \neq b$.

Tổ tiên: tổ tiên của một thành phần trong $(L,\preceq)$ được định nghĩa như sau :

ancestor(a) = {b / $a \preceq b$}

Con cháu: con cháu một thành phần trong $(L,\preceq)$ được định nghĩa như sau :

descendant (a) = {b / $b \preceq a$}

Mỗi thành phần của lưới sẽ là tổ tiên và con cháu của chính nó.

Tập *next*(a) = {b / $a \prec b$, $!\exists c, a \prec c , c \prec b$} : là tập những thành phần là tổ tiên trực tiếp của thành phần a.

c) Đồ thị

Lưới $(L,\preceq)$ được biểu diễn như một đồ thị, mỗi thành phần trong lưới là một đỉnh và tồn tại một cạnh từ a lên b nếu và chỉ nếu b thuộc về next(a).

d) Sự phân cấp

Trong các ứng dụng thực tế, chiều của một *DC* thường có hơn một thuộc tính, trường hợp này mỗi chiều được tổ chức thành một phân cấp các thuộc tính. Ví dụ chiều *date* trong một *DC* có thể được phân cấp thành : *day, month, year.*

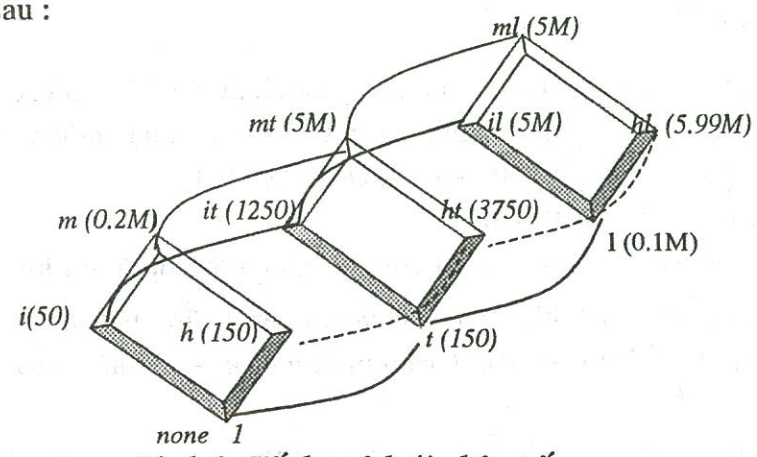
e) Kết hợp nhiều lưới

Ta có 2 loại phụ thuộc:
-Phụ thuộc bởi chiều (thuộc tính) (như *ví dụ 2*)
-Phụ thuộc bởi sự phân cấp trong một chiều (*II.d*)
Thực hiện kết hợp các loại phụ thuộc trên bằng cách sau: mỗi phụ thuộc loại 1 sẽ được mở rộng bằng cách thực hiện các *tổ hợp* nhóm các thuộc tính phân cấp của loại 2, ví dụ: xét 2 chiều *matien* và *tkkh*. Mỗi chiều này được tổ chức phân cấp thành nhiều thuộc tính :

- chiều *tkkh* được phân cấp thành : tài khoản khách hàng (*t*), loại tài khoản (*l*).
- chiều *matien* được phân cấp thành : mã loại tiền *(m)*, tỉ giá *(i)*, nhóm *(h)*.

Hình 2: Sự phân cấp của chiều matien và tkkh

Có thể có những truy vấn kết hợp xảy ra như : nhóm theo loại tài khoản (t), nhóm theo tài khoản khách hàng, nhóm tiền (*an*) ... Do đó ta có *lưới kết hợp* được từ 2 lưới phân cấp trong *hình 2* như sau :



Hình 3: Kết hợp 2 lưới phân cấp

Cách tiếp cận *lưới* cũng cho ta biết một thứ tự cụ thể các MV. Ví dụ cho một tập MV cần cụ thể $S = \{Q_1, ..., Q_N\}$. Xây dựng một lưới $(S, \preceq)$ với các thành phần Q_i rồi thực hiện một phép sắp xếp *topo* trên toán tử $\preceq$ theo thứ tự giảm dần. Giả sử kết quả là $Q_{1s}, ... Q_{Ns}$. Ta thực hiện việc cụ thể theo thứ tự này. Để cụ thể các MV đầu tiên (không có tổ tiên) thì phải truy xuất vào dữ liệu nguồn. Tuy nhiên những MV kế tiếp có thể được cụ thể bằng cách sử dụng những MV đã cụ thể mà nó phụ thuộc=> giảm TGTV một cách đáng kể.

III. TỐI ƯU KHUNG LƯỚI [3]

Cho trước số MV cần cụ thể là *k*, cần xác định MV nào sẽ được chọn để TGTV ít nhất. Thuật toán sử dụng là giải thuật '*greedy*' : tiến hành chọn tập gồm k các MV sao cho mỗi MV được chọn là tốt nhất. Ta cũng sẽ chứng minh thuật toán này gần tối ưu.

a) Thuật toán Greedy [3]

Cho trước một khung lưới :

gọi C(v) là kích thước (số bộ) của MV v.

gọi *k* là số MV cần cụ thể.

gọi S là tập MV được tuyển chọn (có chứa MV mà không có tổ tiên), ta định nghĩa hàm B(v,S) -hàm lợi ích của việc chọn tiếp MV v đối với S- như sau:

Với mỗi $w \preceq v$, đại lượng B_w được định nghĩa:

Gọi *u* là MV có kích thước nhỏ nhất trong S sao cho $w \preceq u$.

Nếu $C(v) < C(u)$ thì $B_w = C(u) - C(v)$, ngược lại $B_w = 0$

$B(v,S) = \sum_{w \preceq v} B_w$

Ý nghĩa của hàm B(v,S) : muốn chọn MV v, ta xác định các MV w $\preceq$ v , sau đó so sánh chi phí thi hành w mà sử dụng v và chi phí thi hành w mà sử dụng một MV u có chi phí nhỏ nhất trong S (w $\preceq$ u). Nếu muốn chọn v thì C(v) phải nhỏ hơn C(u), độ lệch đó có nghĩa là *lợi ích* nếu chọn v. Cộng tất cả các độ lệch lại ta có được hàm lợi ích B(v,S). MV v nào có hàm lợi ích lớn nhất sẽ được tuyển vào tập S. *Ta có thuật toán Greedy như sau:*

Vào:

- Khung lưới (L, $\preceq$).
- *k* là số MV xác định để cụ thể.

Ra: - tập S là tập các MV được chọn gần tối ưu.

Thuật toán :

S= {MV không có tổ tiên trong lưới}

For i=2 to k do {

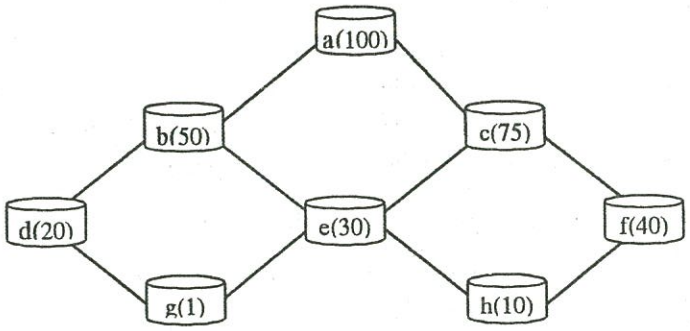
Chọn MV v không nằm trong S sao cho B(v,S) lớn nhất ;

S = S U {v} ;

}

Ví dụ 3: cho lưới như *hình 4* gồm 8 MV từ a -> h, cần chọn k=4 MV sao cho TGTV ít nhất:

Hình 4 : Khung lưới với kích thước cho trước.



- MV *a* với kích thước 100 bắt buộc phải được tuyển vào S, ta cần chọn thêm 3 MV nữa:
- Với i=1 : ta có bảng phân tích lợi ích nếu chọn một MV trong lưới như sau:

chọn b : B(b,S) =$B_b + B_d + B_e + B_g + B_h$ = (C(a)-C(b)) x 5 = 50 x 5 =250.

chọn c : B(c,S) = $B_c + B_e + B_f + B_g + B_h$ = (C(a)-C(c)) x 5 = 25 x 5 =125.

chọn d : B(d,S) = $B_d + B_e$ = (C(a)-C(d)) x 2 = 80 x 2 =160.

chọn e : B(e,S) = $B_e + B_g + B_h$ = (C(a)-C(e)) x 3 = 70 x 3 =210.

chọn f : B(f,S) = $B_f + B_h$ = (C(a)-C(f)) x 2 = 60 x 2 =120.

chọn g : B(g,S) = B_g= C(a)-C(g) = 99.

chọn h : B(h,S) = B_h = C(a)-C(h) = 90.

Như vậy , nếu chọn b sẽ có lợi ích lớn nhất, nên S={a,b}

Tương tự *với i=2,* nếu chọn f sẽ có lợi ích lớn nhất: B(f,S)=(C(a)-C(f)) + (C(b)-C(f)) = 60 + 10 =70 nên S={a,b,f} và *với i=3,* nếu chọn d sẽ có lợi ích lớn nhất : B(d,S) =(C(b)-C(d)) x 2 = 30 x 2 =60 nên S={a,b,f,d}

Tập MV được chọn để cụ thể là S={a,b,f,d}, rõ ràng nếu chỉ cụ thể MV a thì TGTV sẽ là 800 so với TGTV nếu cụ thể tập S là : 100 (của a) + 50 (của b) + 40 (của f) + 20 (của d) + 100 (của c, dựa vào a) + 50 (của e, dựa vào b) + 20 (của g, dựa vào d) + 40 (của h, dựa vào f) = 420

Ví dụ 4: thực hiện thuật toán greedy với khung lưới như trong *hình 3*. Ta có kết quả như *hình 5*. Ví dụ này chứng tỏ tại sao ta không cụ thể toàn bộ MV trên lưới (KGLT chiếm tới 22.3M trong khi đó TGTV giảm không đáng kể là 22.3M).

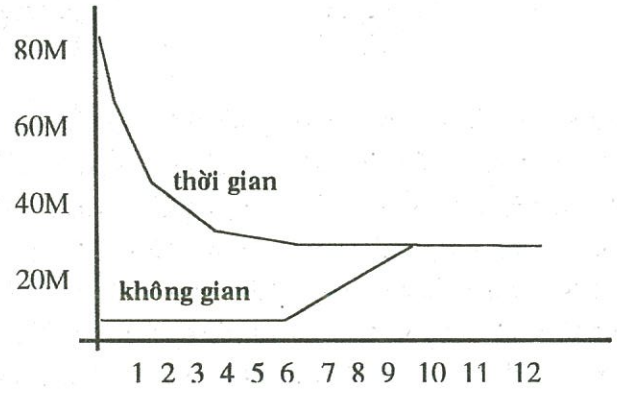
b) Phân tích thuật toán

Gọi $v_1, v_2, \ldots, v_k$ là tập k MV được chọn bởi thuật toán greedy, a_i là giá trị lợi ích của việc chọn v_i, với i=1,...,k. Định nghĩa $A=\sum a_i$

Gọi $w_1, w_2, \ldots, w_k$ là tập k MV tối ưu, nghĩa là tổng giá trị lợi ích sẽ là cực đại. Gọi b_i là giá trị lợi ích của việc chọn w_i, với i=1,...,k . Định nghĩa $B=\sum b_i$.

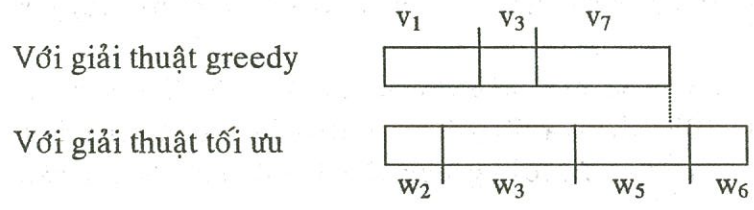
Số MV k	MV được chọn	Lợi ích	TGTV	KGLT
1	tm	Không xđ	72M	6M
2	li	~24M	~48M	~6M
3	lh	~24M	~36M	~6M
4	t	~5.9M	~30.1M	~6.1M
5	m	~5.8M	~24.3M	~6.3M
6	ti	~1M	~23.3M	~11.3M
7	lm	~1M	~22.3M	~16.3M
8	th	~0.01M	~22.3M	~22.3M
9	h	~rất nhỏ	~22.3M	~22.3M
10	l	~rất nhỏ	~22.3M	~22.3M
11	i	~rất nhỏ	~22.3M	~22.3M
12	none	~rất nhỏ	~22.3M	~22.3M

Hình 5 : *Kết quả của thuật toán Greedy cho lưới hình 3.*



Hình 6: *Đồ thị tương quan giữa TGTV vàKGLT*

Ta muốn so sánh việc cải tiến thời gian truy vấn của một MV u bất kỳ do các MV v_i và w_i. *Hình 7* cho thấy sự so sánh này :



Hình 7

Ví dụ: trong *ví dụ 3* bằng thuật toán greedy, thời gian truy vấn *g* được cải tiến từ 100 xuống 20. Trong đó : 50 do chọn b, 30 do chọn d.

- Gọi x_{ij} là giá trị qui chiếu lợi ích của w_i thành v_j . Ta có bất đẳng thức sau :

$\sum_{i=1}^{k} x_{ij} \le a_j$: tổng các phần qui chiếu của tất cả các w_i thành v_j thì không thể vượt quá a_j

$b_i \le a_1$, $\forall i$. Thật vậy, nếu không thì w_i sẽ được chọn đầu tiên bởi giải thuật greedy thay vì v_1 . Tương tự, lần lựa chọn thứ 2 của thuật toán greedy cho ta kết quả:

$b_i - x_{i1} \le a_2$: lý do là lợi ích của các w_i cho lần chọn thứ 2 là b_i trừ cho phần đã qui chiếu vào v_1 , tức là x_{i1}. Và theo thuật toán thì a_2 phải có giá trị lớn nhất trong lần lựa chọn thứ 2. Tổng quát, ta có bất đẳng thức sau cho lần thứ j:

$b_i - x_{i1} - x_{i1} - \ldots - x_{i,j-1} \le a_j$

Tổng hai vế của các bất đẳng thức trên, ta được :

(1) $\sum_{i=1}^{k} b_i \le k a_1$

(2) $\sum_{i=1}^{k} b_i \le k a_2 + \sum_{i=1}^{k} x_{i1}$

.....

(k) $\sum_{i=1}^{k} b_i \le k a_k + \sum_{i=1}^{k} x_{i1} + \sum_{i=1}^{k} x_{i2} + \ldots + \sum_{i=1}^{k} x_{i,k-1}$

Do $\sum_{i=1}^{k} b_i = B$ và $\sum_{i=1}^{k} x_{ij} \le a_j$ nên ta có kết quả sau:

$B \le k a_1$

$B \le k a_2 + a_1$

...

(k) $B \le k a_k + a_1 + a_2 + \ldots + a_{k-1}$ *Hình 8:Hệ bất đẳng thức*

Vì B không lớn hơn giá trị nhỏ nhất của các vế phải của các bất đẳng thức trên, nên giá trị cận trên gần nhất của B xảy ra khi các vế phải của các bất đẳng thức bằng nhau. Hiệu của 2 vế phải bất đẳng thức thứ i và i+1 là $k a_{i+1} - (k-1)a_i = 0$,

=> $a_i = a_{i+1} \frac{k}{k-1}$

=> $A = \sum_{i=0}^{k-1} \left(\frac{k}{k-1}\right)^i a_k$ (Từ định nghĩa $A=\sum a_i$) (*)

=> $B \le k \left(\frac{k}{k-1}\right)^{k-1} a_k$ (Từ hệ bất đẳng thức hình 8) (**)

Từ (*) và (**) : $A / B \ge \frac{1}{k} \sum_{i=0}^{k-1} \left(\frac{k}{k-1}\right)^{i-k+1}$

$\ge \frac{1}{k}\left(1 + \frac{k-1}{k} + \left(\frac{k-1}{k}\right)^2 + \ldots + \left(\frac{k-1}{k}\right)^{k-1}\right)$

$\ge 1 - \left(\frac{k-1}{k}\right)^k$

Với k=2 thì A/B ≥ 3/4 , có nghĩa là thuật toán greedy tối thiểu đạt 3/4 hiệu quả của thuật toán tối ưu.

Do $\lim_{k->\infty}\left(\frac{k-1}{k}\right)^k = 1/e$, vì thế A/B ≥ 1- 1/e = 0.63 : thuật toán greedy luôn đạt hiệu quả không thấp hơn 63% thuật toán tối ưu đối với mọi khung lưới.

Nếu a_1 lớn hơn rất nhiều so với các giá trị a_i khác, thì thuật toán greedy gần như tối ưu. Bởi vì xét bất đẳng thức cuối cùng trong *hình 8* coi như B ≤ a_1, và vì A xấp xỉ a_1 với giả thiết này nên B ≤ A.

Nếu các a_i bằng nhau, thì thuật toán greedy tối ưu. Thật vậy, xét bất đẳng thức đầu tiên trong *hình 8* : B ≤ ka_1 , do A=ka_1 nên B ≤ A.

IV. CẢI TIẾN THUẬT TOÁN [4]

Thuật toán được cải tiến theo các hướng sau:

- Hàm C(v) sẽ được thay bằng E(v), với E(v) = C(v)* S(v). Trong đó C(v) là kích thước của MV v, và S(v) là tần suất truy vấn của MV v. Bởi vì có thể có một MV có kích thước nhỏ nhưng do tần suất truy vấn lớn nên lợi ích do MV đó nếu được tuyển vẫn mang lại nhiều hơn.
- Thuật toán không cần lệ thuộc tham số k, khi đó thuật toán chỉ quan tâm đến việc chọn tập *n* các MV sao cho KGLT giảm xuống cực tiểu mà TGTV không tăng lên bao nhiêu. Trong trường hợp này, người viết thử cài đặt một hàm *heuristic* để chọn số *n* bằng cách xác định tỉ số lớn nhất giữa độ lệch của KGLT và độ lệch của TGTV giữa lần chọn thứ i và i+1, kết quả tương đối tốt qua các mẫu thử nghiệm.

Thuật toán cải tiến:

- Tính lại hàm B(v,s) :

Với mỗi w ⪯ v, đại lượng B_w được định nghĩa:

Gọi *u* là MV có kích thước nhỏ nhất trong S sao cho w ⪯ u.

Nếu E(v) < E(u) thì B_w = E(u) – E(v) , ngược lại B_w = 0

2) B(v,S) = $\sum_{w \preceq v} B_w$

- Hàm T(S) : là TGTV tất cả các MV trong khung lưới (L, ⪯) khi chọn được tập S.
- Hàm Q(S) : là KGLT các MV có mặt trong tập S.

Vào:

- Khung lưới (L, ⪯).

Ra: - tập S là tập các MV được chọn gần tối ưu.

Thuật toán :

S= {MV không có tổ tiên trong lưới}
T'=T(S); Q'=Q(S);
Δ=0; n=1;
For i=2 to *size(L)* do

Chọn MV v không nằm trong S sao cho B(v,S) lớn nhất ;
    S = S *U* { v_i } ; // đánh số khung nhìn v được thêm vào S
    If (| T(S)-T' | / | Q(S)-Q'| > Δ) then
    {
        Δ=| T(S)-T' | / | Q(S)-Q'|)
        n=i

```
        }
    T'=T(S); Q'=Q(S);
}
S=S\ {v_i}_{∀i>n}
```

APPLICATION OF THE GREEDY ALGORITHM FOR SELECTING MATERIALIZED VIEWS IN DATA WAREHOUSE

Nguyen Dang Cao

ABSTRACT: A warehouse is a repository of integrated information drawn from remote data sources. Since a warehouse effectively implements materialized views, we must maintain the views as data sources are updated. A lattice framework is used to express dependencies among views. This paper introduces a greedy algorithm that works off this lattice and determines a good set of views to materialize. A program written by Java illustrated the algorithm.

TÀI LIỆU THAM KHẢO

[1] Y. Zhuge, H. Garcia-Molina, J. Hammer, J.Widom. *View Maintenance in a warehousing environment.* In Carey and Schneider, 1997.

[2] I. Mumick, D. Quass, B. Mumick. *Maintenance of Data Cubes and Summary Tables in a Data Warehouse.* http://db-standford.edu/datawarehouse

[3] V. Harinarayan, A. Rajaraman, and J.Ullman. *Implementing data cubes efficiently.* In Jagadish and Mumick, 1997.

[4] Nguyễn Đăng Cao, Luận văn thạc sĩ ngành công nghệ thông tin*: Các kỹ thuật bảo trì khung nhìn trên kho dữ liệu và cài đặt thử nghiệm* –trường ĐH KHTN, ĐH Quốc gia,2000.