ÁP DỤNG THUẬT TOÁN TỰ BẢO TRÌ KHUNG NHÌN TRONG KHO DỮ LIỆU VÀO BÀI TOÁN NGÂN HÀNG

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 [1,3]. Khi xảy ra một truy vấn, nó sẽ được thi hành trực tiếp ở Dwh. Có khá nhiều thuật toán bảo trì Dwh* online *để bảo đảm dữ liệu tại Dwh luôn nhất quán với nguồn thông tin, chia thành 2 nhóm: nhóm thứ nhất bảo trì theo cách gửi truy vấn ngược lại nguồn [3,4], nhóm thứ hai tự bảo trì ngay tại Dwh bằng cách thêm vào các dữ liệu hỗ trợ [1,2]. Bài báo này giới thiệu: hệ thuật toán tự bảo trì Dwh với các dữ liệu hỗ trợ thêm vào tối thiểu do nhóm nghiên cứu thuộc trường ĐH Standford, Mỹ phát triển [1,2], một đóng góp mở rộng thuật toán [6] và việc ứng dụng Dwh trong bài toán ngân hàng nhiều chi nhánh [7]. Sản phẩm được thực hiện bằng ngôn ngữ Java trên mô hình 3 lớp Java-CORBA/RMI-Oracle.*

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 sau khi đã tính toán và kèm theo cấu trúc chỉ mục.

-Bảo trì MV: là quá trình cập nhật MV sao cho MV nhất quán với những thay đổi dữ liệu diễn ra ở nguồn.

-Tự bảo trì: cho một MV được định nghĩa trên một tập các quan hệ nguồn R. Gọi δMV là những thay đổi trên MV ứng với δR là những thay đổi trên R. Nếu δMV có thể được tính mà chỉ cần dùng MV và δR thì MV được gọi là tự bảo trì. Nếu MV không tự bảo trì, ta quan tâm đến việc tìm một tập các khung nhìn hỗ trợ (Auxiliary MV- MV_A) tối thiểu sao cho tập $\{MV\} \cup MV_A$ là tự bảo trì.

b) Đặt vấn đề: 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 [6,7]:

(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. Mỗi RBTVTC từ S.B -> R.A nghĩa là với mỗi một bộ $s \in S$ sẽ tồn tại

một bộ r ∈ R sao cho s.B = r.A. Giả sử tại Dwh cần tạo một MV có nội dung như sau: "Tất cả giao dịch xảy ra trên những tài khoản ngân hàng thuộc nhóm loại tiền 'USD' với khách hàng thuộc nhóm nước 'TW' và loại tài khoản khách hàng là 'DS' tại chi nhánh Hà nội". Thông tin đưa ra gồm: số giao dịch, ngày giao dịch, số tài khoản ngân hàng, số tài khoản khách hàng, tên khách hàng, số tiền.

CREATE VIEW MV_1 *AS*
SELECT Sogd, Ngaygd, Tknh, Tkkh,Tenkh, Sotien *FROM* K, Kt, G, Nt *WHERE* K.Makh=Kt.Makh *AND* Kt.Tkkh=G.Tkkh *AND* Nt.Tknh=G.Tknh *AND* K.Quoctich='TW' *AND* Kt.Loaitk='DS'*AND* Nt.Matien= 'USD'

Vấn đề đặt ra là: với MV như trên thì cần lưu thêm MV_A nào tại Dwh để chúng tự bảo trì?

Bảng 1 trình bày những phát biểu SQL cho một tập gồm 3 MV_A , tập này đủ để bảo trì MV_1 khi xẩy ra cập nhật *thêm và xóa* ở mỗi quan hệ nguồn, và bản thân cũng có thể tự bảo trì. Việc sử dụng các MV_A như *bảng 1* dẫn đến một sự tiết kiệm đáng kể so với việc cụ thể toàn bộ những quan hệ nguồn như minh họa trong *bảng 2*. Giả sử rằng chúng ta có :

-Số bộ của mỗi quan hệ nguồn được cho trong *cột 2 bảng 2*.

-Tỷ lệ các bộ có K.Quoctich= 'TW' là 0.02, Kt.Loaitk= 'DS' là 0.25 và Nt.Matien= 'USD' là 0.05.

- Độ phân tán là giống nhau thì số bộ thoả điều kiện chọn cục bộ được cho trong *cột 3 bảng 2*.

Một phương pháp bảo trì MV khi xảy ra cập nhật *thêm* trên các quan hệ nguồn là: thực hiện phép chiếu và điều kiện chọn cục bộ trên quan hệ nguồn và chỉ giữ lại tại Dwh những bộ, thuộc tính của quan hệ nguồn thỏa mãn các phép chọn và chiếu đó. Đối với ví dụ trên, số bộ mà phương pháp này thu được để lưu trong kho dữ liệu được trình bày trong *cột 3 bảng 2*.

Quan hệ nguồn	Số bộ trong quan hệ nguồn	Số bộ thỏa mãn điều kiện chọn cục bộ	Số bộ trong các MV_A *của bảng 1*
K	2.000	40	40
Kt	8.000	2.000	400
G	10.000	10.000	0
Nt	500	25	25
Tổng	20.500	12.065	465

Bảng 2: Số lượng các bộ trong quan hệ nguồn và MV_A (qua kiểm tra thử nghiệm).

```
CREATE VIEW ht_K AS
SELECT Makh,Tenkh FROM K
WHERE Quoctich= 'TW'
CREATE VIEW ht_Kt AS
SELECT Tkkh, Makh FROM Kt WHERE LoaiTk=
'DS' AND Makh IN (SELECT Makh FROM ht_K)
CREATE VIEW ht _Nt AS
SELECT Tknh FROM Nt
WHERE Matien= 'USD'
```

Bảng 1 : Các MV_A được dùng để bảo trì MV_1

Ta *cải tiến* phương pháp trên bằng cách sử dụng khóa và các RBTVTC [5]. Xét ví dụ sau với các loại cập nhật:

- Cập nhật *thêm*: không cần lưu bất kỳ bộ nào trong *G*, do khóa và các RBTVTC đảm bảo những bộ tồn tại trong *G* không thể kết với những bộ được thêm vào các quan hệ khác. Có thể loại bỏ các bộ trong *Kt* không thể kết với bất kỳ bộ nào trong *K* có Quoctich = 'TW', vì các bộ có sẵn trong *Kt* sẽ không kết với những bộ mới được thêm vào *K*. Cách này giảm rất

nhiều số lượng các bộ trong MV_A mà chỉ cần thực hiện một số phép chọn (xem minh hoạ *cột 4 bảng 2*).

- Cập nhật *xóa*: tương tự sử dụng các ràng buộc khóa để quản lý cập nhật xoá trên những quan hệ nguồn mà không cần quan tâm đến tình trạng các quan hệ nguồn. Ta có thể xác định ảnh hưởng của việc xoá các bộ trong quan hệ *Kt, G và Nt* mà không phải tham chiếu đến những quan hệ nguồn nào khác bởi vì MV_1 đã bao gồm khóa của những quan hệ này. Chỉ cần kết những bộ bị xóa với MV_1 dựa trên những khóa thích hợp. Và đặc biệt là dù MV_1 không chứa khoá của *K*, nhưng $K \triangleright\triangleleft_{makh} Kt$, vì thế ảnh hưởng của cập nhật *xóa* trong *K* có thể được xác định bằng cách kết những bộ bị xoá với *Kt* rồi *kết* kết quả này với MV_1 trên khóa của *Kt*.
- Cập nhật *sửa*: việc sửa trên các quan hệ nguồn *rất ít xảy ra* mà thường chỉ là những cập nhật *thêm*. Tuy nhiên nếu việc sửa các quan hệ nguồn trong ví dụ trên không làm thay đổi giá trị của các thuộc tính có liên quan đến các điều kiện chọn trong MV_1, thì các MV_A trong *bảng 1* là đầy đủ. Ngược lại, nếu việc sửa quan hệ *Kt* có thể thay đổi thuộc tính *Loaitk* thì cần phải sử dụng thêm MV_A sau đây (có 500 bộ = 10000 x 0.05 là tỷ lệ số bộ thỏa điều kiện Nt.Matien= 'USD') :

CREATE VIEW ht_G *AS SELECT* Sogd,Ngaygd, Tknh, Tkkh, Sotien
FROM G *WHERE* Tknh *IN (SELECT* Tknh *FROM* ht_Nt)

Nghĩa là cần phải giữ tất cả các giao dịch liên quan đến những tài khoản ngân hàng có Matien= 'USD' trong trường hợp thuộc tính Loaitk của các bộ trong *Kt* tương ứng bị sửa thành 'DS'. Dĩ nhiên, nếu những cập nhật sửa có thể thay đổi Matien của quan hệ *Nt* thành 'USD' thì phải giữ tất cả các bộ của quan hệ *G* để bảo trì khung nhìn. Trong thực tế các thuộc tính xuất hiện trong những điều kiện chọn của MV có khuynh hướng là những thuộc tính không bị sửa. Như thế, khi những cập nhật sửa như vậy không xảy ra thì việc tự bảo trì chỉ cần rất ít thông tin hỗ trợ thêm vào. Vì thế, việc nhận biết những cập nhật *sửa* nào được cho phép là một điểm quan trọng trong thuật toán.

II THUẬT TOÁN TỰ BẢO TRÌ [1,2]

a) Chú thích, thuật ngữ và giả định:

Ta có các định nghĩa về quan hệ nguồn R khi xảy ra một trong ba kiểu cập nhật *sửa*:

Loại 1: là những cập nhật *sửa* trên R có thể thay đổi giá trị các thuộc tính liên quan đến điều kiện chọn (cục bộ hay kết) trong MV. Loại 1 có thể thêm những bộ mới vào MV và cũng có thể xóa một số bộ khỏi MV vì thế ta xem loại 1 như là cập nhật *xóa* những bộ mang giá trị cũ và cập nhật *thêm* những bộ mang giá trị mới.

Ví dụ: cho $MV=\sigma_{R.A=10}R \triangleright\triangleleft S$ nếu giá trị của R.A của một bộ trong R thay đổi từ 9 thành 10 thì kết quả là một số bộ mới sẽ được thêm vào MV.

Loại 2: là những cập nhật *sửa* trên R không làm thay đổi giá trị các thuộc tính có liên quan đến điều kiện chọn nhưng làm thay đổi giá trị của những thuộc tính trong MV. Loại 2 chỉ làm thay đổi giá trị một số thuộc tính của những bộ đang tồn tại trong MV.

Loại 3: trường hợp còn lại. Loại 3 không làm ảnh hưởng gì đến MV, vì thế chúng không cần phải truyền về Dwh. Do đó thuật toán chỉ xét những cập nhật sửa *loại 1* và *loại 2*.

b) Thuật toán xác định MV_A:

Cho trước MV, chúng ta trình bày thuật toán xác định tập các MV_A sao cho sự kết hợp MV và MV_A là tự bảo trì. Mỗi MV_A ký hiệu A_{Ri} là một biểu thức có dạng : $A_{Ri}=(\pi\sigma R_i)\triangleright\!<A_{Rj1}\triangleright\!<A_{Rj2}\triangleright\!<\ldots\triangleright\!<A_{Rjn}$. Một số định nghĩa khác:

- <R,ε> : là một đồ thị kết $\mathcal{G}$(MV) có hướng, trong đó : R - là tập các quan hệ tham chiếu trên MV - trở thành các đỉnh của đồ thị. Nếu MV chứa điều kiện kết $R_i.B = R_j.A$, và A là khóa của R_j thì sẽ tồn tại một cạnh có hướng $e(R_i,R_j) \in \varepsilon$ đi từ R_i đến R_j. Cạnh được chú thích là RI nếu có một RBTVTC từ $R_i.B$ vào $R_j.A$. Giả sử rằng đồ thị không có *chu trình* và *khuyên*.

-$Dep(R_i,\mathcal{G})=\{R_j \mid \exists\, e(R_i,R_j)$ trong <R,ε> được chú thích RI, và R_j không có cập nhật *loại 1*}

-$Dep^+(R_i,\mathcal{G})$ là bao đóng bắc cầu của $Dep(R_i,\mathcal{G})$: *$Dep^+(R_i,\mathcal{G})$ giúp xác định A_{Ri} nào sẽ được giữ tại Dwh để bảo trì MV hay sẽ bị loại bỏ.*

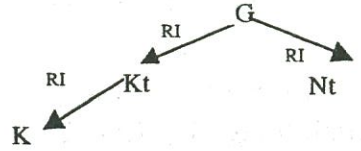
$$\text{-Need}(R_i,\mathcal{G})=\begin{cases}\varnothing & \text{: Nếu khóa của } R_i \text{ được lưu trong trong MV}\\ \{R_j\}\cup \text{Need}(R_j,\mathcal{G}) & \text{: Nếu khóa của } R_i \text{ không được lưu trong MV nhưng tồn tại một } R_j \text{ sao cho } e(R_j,R_i)\in\mathcal{G}(\text{MV}).\\ R-\{R_i\} & \text{: Trường hợp khác}\end{cases}$$

$Need(R_i,\mathcal{G})$ cũng được sử dụng để xác định xem có cần thiết phải lưu các MV_A hay không. A_{Rj} cần phải lưu lại nếu R_j xuất hiện trong Need của R_i nào đó. Nếu khóa của R_i được lưu trong MV, thì cập nhật xóa và cập nhật sửa loại 2 trên R_i có thể được truyền đến MV bằng cách kết trực tiếp chúng với MV dựa trên khóa của R_i. Ngược lại, nếu khóa của R_i không được lưu trong MV nhưng R_i được kết với một quan hệ R_j khác trên khóa của R_i và MV có khóa của R_j thì cập nhật xóa và cập nhật sửa loại 2 trên R_i có thể được gởi vào MV bằng cách kết chúng với R_j và lấy kết quả đó kết với MV. Trong trường hợp này, R_j có trong Need của R_i và do đó A_{Rj} là cần thiết.

Tổng quát: nếu khóa của R_i không hiện diện trong MV nhưng R_i kết với R_j trên khóa của R_i thì MV_A cho R_j và mỗi quan hệ trong $Need(R_j,\mathcal{G})$ là cần thiết cho việc truyền cập nhật xóa và cập nhật sửa *loại 2* trên R_i. Cuối cùng, nếu tất cả những điều kiện trên đều không thỏa thì những MV_A cho mọi quan hệ tham chiếu trong MV trừ R_i đều cần thiết.

Để minh họa cho những định nghĩa trên, ta xem lại MV_1. Ta có đồ thị $\mathcal{G}(MV_1)$ như trong *bảng 3*, và giả sử tất cả các quan hệ nguồn đều có cập nhật sửa *loại 2*, ta có Dep, Dep^+ và Need như trong *bảng 4*.

Thuật toán tìm MV hỗ trợ tối thiểu trình bày trong *bảng 6*. Những MV hỗ trợ sinh ra từ thuật toán chính là những MV được cho trong *bảng 1 mục I*, ta viết lại dưới dạng đại số quan hệ (đsqh) như trong *bảng 5*.



Bảng 3: Đồ thị $G(MV_1)$

$Dep(K,\mathcal{G})=\varnothing$	$Dep^+(K,\mathcal{G})=\varnothing$
$Dep(Kt,\mathcal{G})=\{K\}$	$Dep^+(Kt,\mathcal{G})=\{K\}$
$Dep(Nt,\mathcal{G})=\varnothing$	$Dep^+(Nt,\mathcal{G})=\varnothing$
$Dep(G,\mathcal{G})=\{Kt,Nt\}$	$Dep^+(G,\mathcal{G})=\{K,Nt,Kt\}$
$Need(K,\mathcal{G})=Kt$	$Need(Kt,\mathcal{G})=\varnothing$
$Need(Nt,\mathcal{G})=\varnothing$	$Need(G,\mathcal{G})=\varnothing$

Bảng 4: Các hàm Dep và Need cho các quan hệ nguồn

-$A_K=\pi_{makh,tenkh}\sigma_{quoctich=TW}\,K$

-$A_{Kt}=\pi_{tkkh,makh}\sigma_{loaitk=DS}\,Kt \triangleright\!<_{makh} A_K$ -$A_{Nt}=\pi_{tknh}\,\sigma_{matien=USD}\,Nt$

Bảng 5: Tập khung nhìn hỗ trợ cho việc bảo trì khung nhìn MV_1

Thuật toán F xác định khung nhìn hỗ trợ cực tiểu [1,2]:
Vào: MV Ra: Tập các MV_A
Phương pháp:
Gán R= tập các quan hệ nguồn được tham chiếu trong MV.
Xây dựng đồ thị $\mathcal{G}$(MV).
For each $R_i \in R$ {
xây dựng $Dep(R_i, \mathcal{G})$, $Dep^+(R_i, \mathcal{G})$ và $Need(R_i, \mathcal{G})$
}
For each $R_i \in R$ {
If $Dep^+(R_i, \mathcal{G}) = R-\{R_i\}$ và $!\exists R_j \in R$ sao cho $R_i \in Need(R_j, \mathcal{G})$
Then A_{Ri} là không cần thiết
Else $A_{Ri}=(\pi_P\sigma_S R_i) \triangleright\!<_{C1} A_{Rk1} \triangleright\!<_{C2} A_{Rk2} \triangleright\!<_{C3} ... \triangleright\!<_{Cm} A_{Rkm}$
}
Trong đó: -P là tập những thuộc tính trong R_i mà được lưu trong MV, xuất hiện trong các điều kiện kết, hoặc là một khóa của R_i
-S là tập các điều kiện chọn cục bộ có thể tồn tại trên R_i
-C_l là điều kiện kết $R_i.B=R_{kl}.A$ với A là khóa của R_{kl}
-$Dep(R_i, \mathcal{G}) = \{R_{k1}, R_{k2}, \ldots, R_{km}\}$

Bảng 6: Thuật toán suy dẫn những khung nhìn hỗ trợ.

Định lý: Tập MV_A kết quả của thuật toán F là tập khung nhìn tối thiểu duy nhất có thể gộp vào MV mà từ đó $\{MV\} \cup MV_A$ là tự bảo trì [1,2].

c) Bảo trì khung nhìn có sử dụng những khung nhìn hỗ trợ[1,2]

Trong phần này, ta sẽ thấy một tập MV_A được chọn bởi thuật toán F nói trên là đủ để bảo trì MV bằng cách chuyển biểu thức bảo trì (***BTBT***) khung nhìn (được viết dưới dạng những thay đổi và các quan hệ nguồn) thành biểu thức BTBT tương đương (được viết dưới dạng những thay đổi, MV và MV_A).

c.1) Xử lý cập nhật thêm

Gọi ΔR là các *cập nhật thêm* trên R. Thay thế các quan hệ R bởi ΔR trong biểu thức đsqh, ta thu được BTBT cho MV. Ví dụ xảy ra một cập nhật *thêm* trên *G*, gọi là ΔG, thì BTBT tương ứng cho MV là:

$\Delta MV_G=\pi_{makh,tenkh}\sigma_{quoctich='TW'} K \triangleright\!\triangleleft_{makh} \pi_{tkkh,makh}\sigma_{loaitk='DS'} Kt \triangleright\!\triangleleft_{tkkh} \Delta G \triangleright\!\triangleleft_{nhkh} \pi_{tknh}\sigma_{matien='USD'} Nt$

Tương tự cho các BTBT ΔMV_K, ΔMV_{Kt}, ΔMV_{Nt}.

Tính chất 1): nếu có một RBTVTC từ $R_j.B$ đến $R_i.A$ ($R_j.B$ là khóa ngoại), A là khóa của R_i và R_i không có cập nhật sửa *loại 1* thì :

$$\pi\sigma R_1 \triangleright\!\triangleleft .... \triangleright\!\triangleleft \pi\sigma R_i \triangleright\!\triangleleft_{\Delta Ri.A=Rj.B} R_j \triangleright\!\triangleleft ... \triangleright\!\triangleleft \pi\sigma R_n = \varnothing$$

Luật 1): BTBT dùng để tính toán các ảnh hưởng trên MV của phép thêm vào quan hệ nguồn R_i có thể được loại bỏ nếu tồn tại R_j sao cho $R_i \in Dep(R_j, \mathcal{G})$.

=> Luật 1) sẽ loại bỏ một số BTBT và cho phép ta sử dụng những MV_A thay cho những quan hệ nguồn trong các BTBT còn lại đối với cập nhật thêm. Ví dụ khi áp dụng luật 1), các biểu thức bảo trì ΔMV_K, ΔMV_{Kt}, ΔMV_{Nt} có thể bị loại bỏ do tồn tại các RBTVTC (mục I.b). Biểu thức bảo trì ΔMV_G được viết lại sử dụng những MV_A như sau:

$\Delta MV_G = A_K \triangleright\!\triangleleft_{makh} A_{Kt} \triangleright\!\triangleleft_{tkkh} \Delta G \triangleright\!\triangleleft_{nhkh} A_{Nt}$ (A_K, A_{Kt}, A_{Nt} được cho trong *bảng 6*)

c.2) Xử lý cập nhật xóa

Gọi ∇R là các cập nhật xóa trên R, thay thế các quan hệ R bởi ∇R trong biểu thức đsqh, ta thu được BTBT cho MV. Ví dụ xảy ra một cập nhật xóa ∇G trên G, thì BTBT tương ứng cho MV là:

$\nabla MV_G = \pi_{makh,tenkh}\sigma_{quoctich='TW'}K \triangleright\triangleleft_{makh}\pi_{tkkh,makh}\sigma_{loaitk='DS'}Kt \triangleright\triangleleft_{tkkh}\nabla G \triangleright\triangleleft_{nhkh}\pi_{tknh}\sigma_{matien='USD'}Nt$

Tương tự cho các BTBT ∇MV_K, ∇MV_{Kt}, ∇MV_{Nt}.

Tính chất 2): cho $MV=\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_i \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$, nếu khóa của R_i được lưu trong MV thì ta có: $\nabla MV_{Ri} = \pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma \nabla R_i \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n = MV \triangleright\triangleleft_{Key(Ri)} \nabla R_i$

Tính chất 3): cho $MV=\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_i \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$ thoả mãn các điều kiện sau:

MV chứa điều kiện kết: $R_i.A=R_{i+1}.B$, $R_{i+1}.A=R_{i+2}.B$, ..., $R_{i+k-1}.A=R_{i+k}.B$

Thuộc tính A là khóa của R_{i+j} $(0<=j<=k)$

$R_{i+k}.A$ được lưu trong MV

Thì ta có : $\nabla MV_{Ri} = \pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma \nabla R_i \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$

$= MV \triangleright\triangleleft_{Ri+k.A} \pi\sigma R_{i+k} \triangleright\triangleleft_{Ri+k.B=Ri+k-1.A} \pi\sigma R_{i+k-1} \triangleright\triangleleft ... \triangleright\triangleleft_{Ri+1.B=Ri.A} \triangleright\triangleleft \nabla R_i$

Luật 2): Cho $\mathcal{G}(MV)$ và $Need(R_i,\mathcal{G})=\{R_{i+1},..., R_{i+k}\}$, $k\geq 0$: BTBT cho MV ứng với cập nhật xóa trên R_i có thể được đơn giản theo *tính chất 3)* nếu $Need(R_i,\mathcal{G})$ không chứa tất cả quan hệ của MV trừ R_i.

=> *Luật 2) cho phép ta viết lại BTBT cho cập nhật xóa bằng cách thay $\pi\sigma R_i$ trong BTBT thành A_{Ri} tương ứng.* Ví dụ BTBT ∇MV_G được viết lại như sau: $\nabla MV_G = MV \triangleright\triangleleft_{sogd} \nabla G$.

Vấn đề: trong trường hợp tổng quát, liệu việc dùng những MV_A có đủ để thi hành các BTBT (đã đơn giản hóa) cho cập nhật *xóa* ? Vấn đề này đã được chứng minh trong [1,2].

c.3) Xử lý cập nhật sửa loại 2

Ta sử dụng 2 BTBT dùng để tính những thay đổi trên MV do cập nhật sửa *loại 2* trên quan hệ nguồn R:

- Biểu thức trả về những bộ sẽ bị *xóa* khỏi MV (kí hiệu $\nabla_\mu MV_R$): bằng cách thay quan hệ nguồn R bởi $\pi^{old}{}_\mu R$ ta thu được BTBT cho MV dùng cho việc tính những bộ sẽ bị xóa khỏi MV. Trong đó $\pi^{old}{}_\mu R$ là những *giá trị cũ* của những bộ được *sửa* trong R.
- Biểu thức trả về những bộ sẽ được *thêm* vào MV (kí hiệu $\Delta_\mu MV_R$): bằng cách thay quan hệ nguồn R bởi $\pi^{new}{}_\mu R$ ta thu được BTBT cho MV dùng cho việc tính những bộ sẽ được thêm vào MV. Trong đó $\pi^{new}{}_\mu R$ là những *giá trị mới* của những bộ được *sửa* trong R.

Ví dụ: BTBT tính những bộ bị *xóa* khỏi MV_1 ứng với cập nhật *sửa loại 2* trên quan hệ *G*:

$\nabla_\mu MV_G = \pi_{makh,tenkh}\sigma_{quoctich='TW'}\mathbf{K} \triangleright\triangleleft_{makh} \pi_{tkkh,makh}\sigma_{loaitk='DS'}\mathbf{Kt} \triangleright\triangleleft_{tkkh}\pi^{old}\nabla_\mu G \triangleright\triangleleft_{nhkh} \pi_{tknh}\sigma_{matien='USD'}\mathbf{Nt}$

Tương tự cho $\nabla_\mu MV_K$, $\nabla_\mu MV_{Kt}$, $\nabla_\mu MV_{Nt}$. Và thay $\pi^{old}\nabla_\mu G$ thành $\pi^{new}\Delta_\mu G$, ta được $\Delta_\mu MV_G$ (tương tự cho $\Delta_\mu MV_K$, $\Delta_\mu MV_{Kt}$, $\Delta_\mu MV_{Nt}$)

Tính chất 4): cho $MV=\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$. Nếu khóa của quan hệ R_i được lưu trong MV thì ta có: $\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_{i-1} \triangleright\triangleleft \pi^{old}\sigma_\mu R_i \triangleright\triangleleft \pi\sigma R_{i+1} \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n = MV \triangleright\triangleleft_{oldkey(Ri)} \pi^{old}{}_\mu R_i$

trong đó $\triangleright\triangleleft_{oldkey(Ri)}$: là phép kết trên giá trị cũ của khóa.

Và : $\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_{i-1} \triangleright\triangleleft \pi^{new}\sigma_\mu R_i \triangleright\triangleleft \pi\sigma R_{i+1} \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n = MV \triangleright\triangleleft_{oldkey(Ri)} \pi^{new}{}_\mu R_i$

Tính chất 5) (Trường hợp tổng quát của *tính chất 4*)

Cho $MV=\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$ thoả mãn các điều kiện sau:

1. MV chứa điều kiện kết: $R_i.A=R_{i+1}.B$, $R_{i+1}.A=R_{i+2}.B$, ..., $R_{i+k-1}.A=R_{i+k}.B$
2. Thuộc tính A là khóa của R_{i+j} $(0<=j<=k)$
3. $R_{i+k}.A$ được lưu trong MV

Thì ta có biểu thức tương đương sau (ngay cả khi không có RBTVTC):

$\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_{i-1} \triangleright\triangleleft \pi^{old}\sigma_\mu R_i \triangleright\triangleleft \pi\sigma R_{i+1} \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$

$= MV \triangleright\triangleleft_{Ri+k.A} \pi\sigma R_{i+k} \triangleright\triangleleft_{Ri+k.B=Ri+k-1.A} \pi\sigma R_{i+k-1} \triangleright\triangleleft ... \triangleright\triangleleft_{Ri+1.B=Ri.A} \pi^{old}{}_\mu R_i$

Và : $\pi\sigma R_1 \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_{i-1} \triangleright\triangleleft \pi^{new}\sigma_\mu R_i \triangleright\triangleleft \pi\sigma R_{i+1} \triangleright\triangleleft ... \triangleright\triangleleft \pi\sigma R_n$

$= MV \triangleright\triangleleft_{Ri+k.A} \pi\sigma R_{i+k} \triangleright\triangleleft_{Ri+k.B=Ri+k-1.A} \pi\sigma R_{i+k-1} \triangleright\triangleleft ... \triangleright\triangleleft_{Ri+1.B=Ri.A} \pi^{new}{}_\mu R_i$

Xét đồ thị kết $\mathcal{G}(MV)$, theo *tính chất 5*: nếu MV lưu khóa của một vài quan hệ R_{i+k} và R_i kết với R_{i+k} dựa trên một loạt khóa (nghĩa là: $Need(R_i, \mathcal{G}) = \{R_{i+1},...,R_{i+k}\}$ và không bao gồm tất cả quan hệ nguồn của MV), thì ta có thể tính những thay đổi trên MV của cập nhật sửa *loại 2* trên R_i bằng cách kết μR_i với dãy quan hệ trên cho đến quan hệ R_{i+k} và rồi kết R_{i+k} với MV.

Luật 3) : cho đồ thị $\mathcal{G}(MV)$ và $Need(R_i, \mathcal{G})=\{R_{i+1},..., R_{i+k}\}$, $k\geq 0$. BTBT để tính các thay đổi trên MV của *cập nhật sửa loại 2* trên R_i có thể được đơn giản theo *tính chất 5* nếu $Need(R_i, \mathcal{G})$ không chứa tất cả quan hệ của MV trừ R_i.

=>Luật 3) được dùng để đơn giản BTBT cho $\nabla_\mu MV_R$ và $\Delta_\mu MV_R$ để chỉ sử dụng nội dung của MV và các MV_A bằng cách thay $\pi\sigma R_i$ trong BTBT thành A_{Ri}.

Ví dụ: Ví dụ BTBT $\nabla_\mu MV_G$ và $\Delta_\mu MV_G$ có thể được viết lại lần lượt như sau:

$\nabla_\mu MV_G = MV \triangleright\triangleleft_{old\_sogd} \pi^{old}{}_\mu G$, $\Delta_\mu MV_G = MV \triangleright\triangleleft_{old\_sogd} \pi^{new}{}_\mu G$

Vấn đề: liệu trong trường hợp tổng quát, việc sử dụng những khung nhìn hỗ trợ có đủ cho việc ước lượng các biểu thức bảo trì ? Vấn đề này đã được chứng minh trong [1,2].

d) Bảo trì MV_A

Ta sẽ thấy tập các MV_A là tự bảo trì. Nhắc lại, những MV_A nhận được từ thuật toán F có dạng: $A_R = (\pi_P\sigma_S R_i) \triangleright\!<_{C1} A_{R1} \triangleright\!<_{C2} A_{R2} \triangleright\!<_{C3} ... \triangleright\!<_{Cm} A_{Rm}$

Trong đó C_i kết bằng một khóa ngoại của R với khóa tương ứng của quan hệ R_i. Bởi vì các phép kết là dọc theo các RBTVTC về khóa ngoại, nên mỗi phép kết nửa có thể được thay thế bởi một phép kết. Mỗi phép kết trong MV_A cũng là phép kết trong khung nhìn gốc, nên đồ thị $\mathcal{G}(MV_A)$ là một đồ thị con của $\mathcal{G}(MV)$. Vì vậy, những thông tin cần để bảo trì MV là cần cho việc bảo trì MV_A tương ứng.

d.1) Xử lý cập nhật thêm

Với A_R cho như trên, BTBT cho cập nhật *thêm* là:

$\Delta A_R = (\pi_P\sigma_S\Delta R) \triangleright\!<_{C1} A_{R1} \triangleright\!<_{C2} A_{R2} \triangleright\!<_{C3} \triangleright\!<_{Cm} A_{Rm}$

Theo *luật 1)*, một bộ *t* được thêm vào bất kỳ A_{Ri} cũng sẽ không kết với bất kỳ bộ nào trong R do RBTVTC từ R đến R_i. Vì vậy, cập nhật *thêm* trên A_{Ri} không cần truyền đến A_R.

d.2) Xử lý cập nhật xóa

Với A_R cho như trên, BTBT cho cập nhật xóa là:

$\nabla A_R = (A_R \triangleright\!<_{key(R)} (\pi_P\sigma_S\nabla R)) \cup (A_R \triangleright\!<_{C1} \nabla A_{R1}) \cup ... \cup (A_R \triangleright\!<_{Cm} \nabla A_{Rm})$

Cập nhật xóa trên A_{Ri} có thể được kết trực tiếp với A_R vì A_R chứa các khóa của mỗi A_{Ri}.

d.2) Xử lý cập nhật sửa loại 2

Để tính những thay đổi trên A_R của *cập nhật sửa loại 2* trên quan hệ nguồn R, ta đưa ra 2 BTBT: BTBT trả lại những bộ bị xóa khỏi A_R : $\nabla_\mu A_R = \pi^{old}(A_R \triangleright\!<_{oldkey(R)\,\mu} R)$ và BTBT trả lại những bộ sẽ được thêm vào A_R: $\Delta_\mu A_R = \pi^{new}(A_R \triangleright\!<_{oldkey(R)\,\mu} R)$ (các cập nhật sửa *loại 2* trên các A_{R1}, $A_{R\,2}$,..., A_{Rm} sẽ không ảnh hưởng đến A_R vì A_R chỉ chứa những thuộc tính trong quan hệ R).

III MỞ RỘNG [6]

1) Các cập nhật *thêm* cùng lúc gửi đến *Dwh:*

Nếu cùng lúc các cập nhật *thêm* được truyền đến Dwh cho cả 4 quan hệ, thì kết quả sẽ sai nếu ta thi hành 4 BTBT như ở mục d1) rồi hội kết quả lại. Do đó phải giả định: khi một cập nhật *thêm* thứ i được truyền đến Dwh, thì các cập nhật *thêm* j < i đã phải được xử lý. Tốc độ xử lý sẽ bị chậm do phải xử lý tuần tự. Một bước cải tiến sau đây cho phép thực hiện BTBT MV *đồng thời* cho các cập nhật *thêm*. Giả sử có các cập nhật *thêm* xảy ra trên tất cả các quan hệ nguồn và được nhận đồng thời tại *Dwh. Bảng 7* cho ta thấy một BTBT MV trong trường hợp này. Để đơn giản, ta bỏ qua các phép chọn và chiếu.

$$\begin{aligned}\Delta MV = {} & (\Delta K \triangleright\triangleleft Kt \triangleright\triangleleft G \triangleright\triangleleft Nt) \uplus ((\Delta K \uplus K) \triangleright\triangleleft \Delta Kt \triangleright\triangleleft G \triangleright\triangleleft Nt) \\ & \uplus ((\Delta K \uplus K) \triangleright\triangleleft (\Delta Kt \uplus Kt) \triangleright\triangleleft \Delta G \triangleright\triangleleft Nt) \\ & \uplus ((\Delta K \uplus K) \triangleright\triangleleft (\Delta Kt \uplus Kt) \triangleright\triangleleft (\Delta G \uplus G) \triangleright\triangleleft \Delta Nt) \\ & \uplus ((\Delta K \uplus K) \triangleright\triangleleft (\Delta Kt \uplus Kt) \triangleright\triangleleft (\Delta G \uplus G) \triangleright\triangleleft (\Delta B \uplus Nt))\end{aligned}$$

Bảng 7

Loại trừ một số phép kết (do *tính chất 1)*, ta có thể đơn giản BTBT trong bảng *7* thành :

$$\Delta MV = (((\Delta K \uplus K) \triangleright\triangleleft \Delta Kt) \uplus (K \triangleright\triangleleft Kt)) \triangleright\triangleleft \Delta G \triangleright\triangleleft (\Delta Nt \uplus Nt)$$

Thay các quan hệ nguồn bằng các MV_A, ta được :

$$\Delta MV = (((\Delta K \uplus A_K) \triangleright\triangleleft \Delta Kt) \uplus (A_K \triangleright\triangleleft A_{Kt})) \triangleright\triangleleft \Delta G \triangleright\triangleleft \Delta Nt \uplus A_{Nt})$$

2) Trường hợp đồ thị có chu trình và khuyên:

Trong qua trình xây dựng tập Need của thuật toán F ở *bảng 6*, một giả thiết là đồ thị không có chu trình và khuyên. Các bước cải tiến sau đây cho phép thuật toán có thể áp dụng được với đồ thị có chu trình và khuyên:

Trường hợp đồ thị có chu trình, hàm Need(R_i,G) được tạo lại như sau :

Bước 1 :

$$\text{Need}(R_i,\mathcal{G}) = \begin{cases} \varnothing & : \text{Nếu khóa của } R_i \text{ được lưu trong trong MV} \\ \{R_j\} \cup \text{Need}(R_j,\mathcal{G}) & : \text{Nếu khóa của } R_i \text{ không được lưu trong MV nhưng } \exists R_j \text{ sao cho } e(R_j,R_i) \in \mathcal{G}(\text{MV}), \text{ và } e(R_j,R_i) \text{ không thuộc một chu trình.} \\ \textit{chưa biết} & : \text{Trường hợp khác} \end{cases}$$

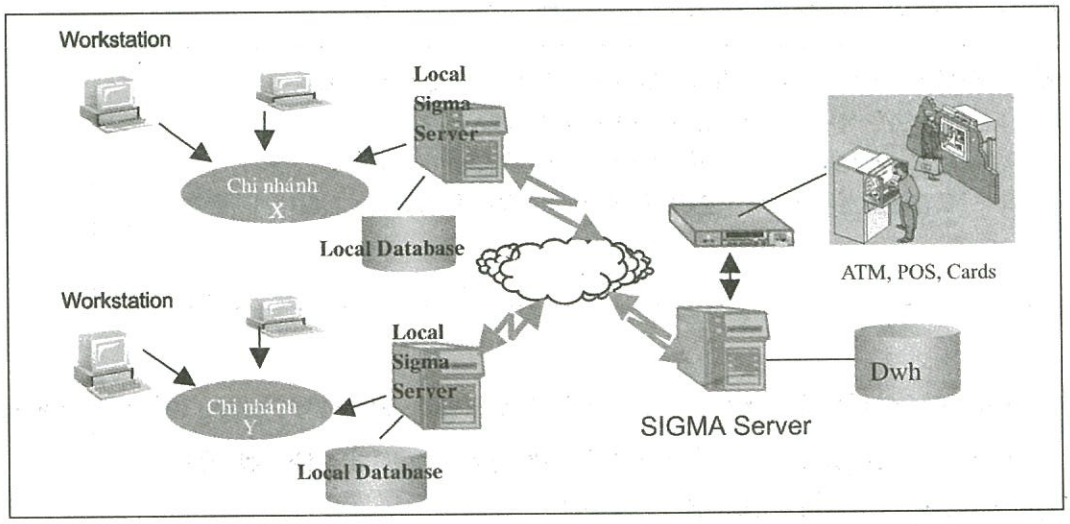
Bước 2 : Với mỗi chu trình, lần lượt xét các đỉnh thuộc chu trình đó. Nếu mọi đỉnh k đều có tập Need là *"chưa biết"* thì gán $\text{Need}(R_k,\mathcal{G}) = \{R\} - R_k$, ngược lại nếu $\exists$ đỉnh k nào có tập Need đã xác định thì thêm tập Need(R_k,G) vào tập Need của tất cả các đỉnh còn lại.

Trường hợp đồ thị có khuyên:

Đồ thị có khuyên tức là trong MV có một quan hệ T được kết với chính nó. Tình huống này có thể tránh được bằng gán nhãn mỗi thể hiện của T xảy ra trong phép kết, ví dụ T_a và T_b. Sau đó áp dụng thuật toán F để tìm A_{Ta} và A_{Tb}.

IV ỨNG DỤNG [7]

Thông tin trong lĩnh vực ngân hàng rất đa dạng và phân tán, nhu cầu đặt một *Dwh* tại hội sở trung ương để theo dõi các biến động thông tin từ các chi nhánh là rất cần. Ngoài chức năng thực hiện các báo cáo tổng hợp nhanh chóng và kịp thời trên *Dwh*, các dịch vụ ngân hàng tiên tiến như máy rút tiền tự động ATM, E-bank, các giao dịch đa chi nhánh, hệ thống thanh toán bù trừ, ... đều có thể được ứng dụng rộng rãi. Hiện tại sản phẩm bài toán ngân hàng n-chi nhánh (SIGMA) ứng dụng các kỹ thuật Dwh đã được thực hiện tại công ty BIT, số 2 Phùng Khắc Khoan, phòng G15 xây dựng trên công nghệ 3 lớp Java-CORBA/RMI-Oracle, mô hình như sau:



Lớp 1 (**Client**) : Dùng Java xây dựng các màn hình nhập liệu, in báo cáo, truy vấn.

Lớp 2 (**Middle-ware server**) : sử dụng công nghệ Corba/RMI đảm nhiệm các xử lý dữ liệu do *client* gửi đến, cũng là nơi thực hiện các thuật toán sản sinh và tự bảo trì khung nhìn. Các thuật toán về *Dwh* được áp dụng vào Sigma, do đó bảo đảm tốc độ đáp ứng thông tin trong tình trạng cơ sở hạ tầng mạng còn yếu kém. Hệ thống Sigma cho phép khai báo các chuỗi *server* vật lý để tăng tốc độ xử lý, tránh được hiện tượng nút cổ chai.

Lớp 3 (**Database server**) : sử dụng Oracle database, tại mỗi chi nhánh là CSDL của chi nhánh đó, tại hội sở trung ương là một *Dwh* lưu các MV và MV_A. Người sử dụng có thể thiết lập các báo cáo trên Dwh bằng những công cụ hỗ trợ của Sigma.

KẾT LUẬN

Ta đã nghiên cứu việc thiết lập các MV tự bảo trì bằng cách lưu bổ sung tập các MV_A tại Dwh. Mặc dù tập các quan hệ nguồn (mà MV được định nghĩa trên đó) cũng là tập các MV_A nhưng phương pháp của ta là tìm được tập các MV_A nhỏ hơn nhiều so với việc lưu trữ toàn bộ các quan hệ nguồn. Do khuôn khổ bài báo, người viết không thể đề cập đến một số trường hợp mở rộng khác của thuật toán, ví dụ như cách giải quyết trong trường hợp khung nhìn có phép tổng.

APPLICATION OF SELF-MAINTAINABLE MATERIALIZED VIEWS ALGORITHM ON DATA WAREHOUSE FOR BANKING PROBLEM

Nguyen Dang Cao

ABSTRACT: A warehouse is a repository of integrated information drawn from remote data source. Since a warehouse effectively implements materialized views, we must maintain the views when the data sources are updated. There are two kinds of algorithms to maintain the view in consistency with source: one has to send queries back to source and one stores a set of auxiliary views at warehouse and together with the view, they are *self-maintainable*. This paper introduces a technique applied to simplify auxiliary views by exploiting key and referential integrity constraints to solve the commercial banking problem with n-branch. The software was written by model 3-tiers Java-CORBA/RMI-Oracle.

TÀI LIỆU THAM KHẢO

[1] D. Quass, A. Gupta, I. Mumick, J. Widom. *Making Views Self-Maintainable for Data Warehousing*. http://db-standford.edu/datawarehouse

[2] Jun Yang, Jennifer Widom. *Temporal View Self-Maintenance in a Warehousing Environment.* http://db-standford.edu/datawarehouse

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

[4] Y. Zhuge, H. Molina, J. Wiener. *The Strobe Algorithms for Multi-Source Data Warehouse Consistency.* http://db-standford.edu/datawarehouse

[5] David Maier, *The Theory Of Relational Databases.* Computer Science Press,1988

[6] 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.

[7] BIT© , *Sigma Technical Help – http://www.bitvn.com*