Tạp chí Tin học và Điều khiển học, T. 16, S. 2 (2000), 19–24

MỘT THUẬT TOÁN LẬP LỊCH BIỂU ĐỂ ĐIỀU KHIỂN CÁC GIAO TÁC THEO MÔ HÌNH ĐỌC VÀ ĐỌC–GHI

NGUYỄN XUÂN HUY, TRỊNH MỸ BÌNH

Abstract. The article proposes a method of scheduling for controlling transactions accessing databases concurently in read-and-write model. This schedule is serializable, that it is equivalent to a seri of all given transactions.

1. MỘT SỐ KHÁI NIỆM CHUNG

1.1. Định nghĩa mô hình

Mô hình đọc và đọc–ghi (read and read-and-write [1, 2]) là mô hình trong đó mỗi đơn vị dữ liệu (ĐVDL) của cơ sở dữ liệu (CSDL) có thể được các giao tác chiếm giữ bởi hai dạng khóa sau:

- Khóa đọc (R): Khi một giao tác lấy được khóa R của ĐVDL A, nó chỉ được phép đọc giá trị của A. Đây là khóa ở chế độ dùng chung, nghĩa là cho phép nhiều giao tác cùng chiếm giữ khóa R của một ĐVDL tại cùng một thời điểm. Biểu diễn $T : R(A)$ cho biết tại thời điểm quan sát giao tác T xin đọc ĐVDL A.

- Khóa đọc–ghi (W): Khi một giao tác lấy được khóa W của ĐVDL A, nó được phép đọc và ghi giá trị vào A. Đây là khóa ở chế độ dùng riêng, nghĩa là cho phép một giao tác cùng chiếm giữ khóa W của một ĐVDL khi ĐVDL đó không bị khóa bởi một giao tác nào khác. Biểu diễn $T : W(A)$ cho biết tại thời điểm quan sát giao tác T xin chiếm giữ ĐVDL A để thực hiện giao tác đọc–ghi.

Từ định nghĩa trên ta thấy rằng, hệ thống có thể cho phép nhiều giao tác cùng chỉ đọc một ĐVDL. Nhưng để một giao tác có thể đọc–ghi vào ĐVDL nào đó thì tại thời điểm đó ĐVDL phải không bị giao tác nào khác chiếm giữ khóa.

1.2. Lịch biểu và tính khả tuần tự của lịch biểu

Một kỹ thuật cơ bản dùng để điều khiển đồng thời một tập giao tác cùng truy nhập CSDL là xây dựng một *lịch biểu* (schedule). Theo thứ tự đó, các bước cơ bản của các giao tác được thực hiện. Các thao tác của một giao tác bất kỳ phải xuất hiện trong lịch biểu theo đúng thứ tự mà chúng xuất hiện trong giao tác. Một lịch biểu S được gọi là *khả tuần tự* (serializable) nếu tác động cuối cùng của nó lên CSDL tương đương với một lịch biểu tuần tự của các giao tác (là lịch biểu được lập bằng cách cho thực hiện lần lượt và trọn vẹn theo thứ tự nào đó từng giao tác một) [1, 2].

1.3. Kiểm tra tính khả tuần tự của lịch biểu

Bộ lập lịch biểu phải đảm bảo mọi lịch biểu được lập ra khả tuần tự. Do vậy, cần có một phép kiểm tra đơn giản về tính khả tuần tự của lịch biểu. Phương pháp truyền thống được mô tả một cách hình thức trong thuật toán sau [1].

Thuật toán 1. *Kiểm tra tính khả tuần tự của một lịch biểu.*

Vào: Một lịch biểu S của một tập các giao tác $T_1, T_2, ..., T_n$.

Ra: Xác định xem S có khả tuần tự không, nếu có đưa ra thứ tự tuyến tính của các giao tác tương đương với S.

Phương pháp:

Bước 1. Tạo một đồ thị định hướng G (gọi là đồ thị đợi) có tập đỉnh là tập các giao tác. Các cung của đồ thị G được xây dựng như sau:

1. Nếu có thao tác dạng $T_j : R(A)$ trong S thì tìm xem giá trị của A hiện tại do giao tác T_i nào đó đã ghi. Nếu có và $T_i \neq T_j$, vẽ một cung từ T_i đến T_j.

2. Nếu có thao tác dạng $T_j : W(A)$ trong S thì tìm xem những giao tác T_i nào đã đọc hoặc ghi giá trị hiện tại của A. Nếu có, thì với mỗi trường hợp $T_j (T_i \neq T_j)$ vẽ một cung từ T_i đến T_j.

Ý nghĩa của cung từ T_i đến T_j là giao tác T_i đứng trước giao tác T_j trong thứ tự tuyến tính tương đương của lịch biểu S.

Bước 2. Kiểm tra đồ thị G, nếu G có chu trình thì lịch biểu S không khả tuần tự, ngược lại S là khả tuần tự và thứ tự topo sinh trong thủ tục sẽ chính là thứ tự tuyến tính cho các giao tác. Thứ tự trước sau của các giao tác được xác định qua cung của đồ thị G, cụ thể là giao tác T_i được xem là đứng trước giao tác T_j nếu trong G có cung từ T_i đến T_j. Thứ tự này luôn xác định được bởi thuật toán sắp topo.

1.4. Mục đích bài báo

Trong [1] chỉ trình bày các thuật toán kiểm tra tính khả tuần tự của một lịch và chứng minh rằng một lịch lập trên các giao tác hai pha thì luôn khả tuần tự. Do tính phức tạp của việc lập lịch, trong thực tiễn, hầu hết các hệ thống điều khiển truy nhập đồng thời đều yêu cầu các giao tác phải được viết dưới dạng hai pha. Một giao tác được gọi là *hai pha* nếu trong giao tác đó mọi thao tác lấy khóa được đặt trước mọi giao tác giải phóng khóa. Mục đích của bài báo là trình bày một thuật toán cho phép xây dựng lịch biểu khả tuần tự của một tập các giao tác bất kỳ. Về thực chất, thuật toán lập lịch trình bày trong bài này tập trung vào việc duy trì bất biến "khả tuần tự" cho lịch biểu động tức là lịch biểu được bổ sung dần dần từng thao tác theo tiến độ của thuật toán. Cụ thể là, một thao tác của một giao tác nào đó sẽ được đưa vào lịch biểu nếu thao tác đó không làm mất tính khả tuần tự của lịch biểu.

2. NỘI DUNG THUẬT TOÁN LLBII

2.1. Đặt vấn đề

Cho $T_1, T_2, ..., T_n$ là các giao tác theo mô hình chỉ đọc và đọc-ghi. Yêu cầu đặt ra là thiết kế một thuật toán xây dựng lịch biểu S khả tuần tự để thực hiện đồng thời các giao tác trên.

2.2. Tư tưởng của thuật toán

Thuật toán được xây dựng trên cơ sở duyệt lần lượt các giao tác. Mỗi lần chọn một thao tác từ giao tác hiện hành để xem xét có thể bổ sung thao tác đó vào lịch S hiện hành hay không. Thao tác đang xét có thể bổ sung vào lịch S hiện hành nếu nó không tạo thành một chu trình trong đồ thị G được lập theo thuật toán 1. Như vậy điều quan trọng là tổ chức một thủ tục hiệu quả để kiểm tra chu trình trong của đồ thị G. Điều này tương đương với việc tổ chức sao cho có thể kết luận nhanh chóng xem liệu một thao tác nào đó được bổ sung vào S có làm mất tính khả tuần tự của S tại thời điểm này hay không. Điều quan trọng thứ hai là xác định hành vi của thuật toán trong trường hợp thao tác đang xét không thể bổ sung vào lịch hiện hành. Trường hợp này đòi hỏi một thủ tục sửa lịch nhanh. Giao tác gây ra chu trình sẽ được đưa ra khỏi lịch S và trả về hàng đợi để sắp xếp lại.

2.3. Một số quy ước

Để tiện trình bày, chúng ta xét một số khái niệm. Giả sử ta cần lập một lịch khả tuần tự S cho n giao tác $T_1, T_2, ..., T_n$ trong mô hình chỉ đọc và đọc–ghi.

Định nghĩa 1. Ta kí hiệu S^i là trạng thái của lịch S sau khi kết nạp i thao tác theo tiến độ của thuật toán.

Định nghĩa 2. *Ma trận đặc trưng* của lịch biểu S là ma trận vuông M cấp n được khởi trị và cập nhật tại mỗi thời điểm hoạt động của thuật toán như sau.

Khởi trị: Mọi phần tử của M được gán trị 0.

Mỗi khi bổ sung một thao tác vào lịch biểu S ta xác định trật tự trước sau các giao tác và gán

$M(i,j) = 1$ nếu T_i đứng trước T_j,

$M(i,j) = 0$ trong các trường hợp còn lại.

Ký hiệu z là một dòng (cột) của ma trận M. Biểu thị z dưới dạng

$$z = (z_1, z_2, ..., z_n),$$

trong đó mọi z_i chỉ có giá trị 0 hoặc 1.

Định lý sau đây cho thấy ý nghĩa của ma trận đặc trưng.

Định lý 1. *Tại mỗi thời điểm xây dựng lịch S, đồ thị của S là phi chu trình khi và chỉ khi mọi phần tử trên đường chéo chính của ma trận M bằng 0.*

Chứng minh

Điều kiện đủ: Giả sử M thỏa mãn điều kiện $M(i,j) = 0$, $i = 1, ..., n$. Ta sẽ chỉ ra là đồ thị của S là phi chu trình. Thật vậy, giả sử đồ thị có chu trình, khi đó sẽ tồn tại hai giao tác k và z tạo thành chu trình $k > ... > z > ... > k$ (quan hệ $x > y$ có ngữ nghĩa là giao tác x phải đứng trước giao tác y trong đồ thị của S). Theo định nghĩa của M sẽ có $M(k,k) = 1$, điều này trái với giả thiết. Vậy đồ thị của S là phi chu trình.

Điều kiện cần: Giả sử đồ thị của S là phi chu trình, chúng ta phải chỉ ra các phần tử trên đường chéo chính của ma trận M bằng 0. Điều này là hiển nhiên vì trong đồ thị không tồn tại chu trình $i > ... > i$ $(\forall i = 1, ..., n)$.

Định nghĩa 3. Cho hai danh sách có n phần tử có giá trị 0 hoặc 1 là $x = (x_1, x_2, ..., x_n)$ và $y = (y_1, y_2, ..., y_n)$. Phép hợp nhất hai danh sách x và y, cho kết quả là một danh sách z gồm n phần tử $z = (z_1, z_2, ..., z_n)$, trong đó $z_i = x_i \vee y_i$ và $\vee$ là phép toán hội của đại số Bool. Chúng ta dùng ký hiệu $z = x \vee y$ để biểu diễn phép hợp nhất hai danh sách x và y.

Từ Định nghĩa 3, chúng ta có khái niệm *phép hợp nhất n danh sách* $x^1, x^2, ..., x^n$ như sau $x^1 \vee x^2 \vee ... \vee x^n = (x^1 \vee x^2 ... \vee x^{n-1}) \vee x^n$.

Ngoài ra chúng ta sử dụng một số ký hiệu sau:

Ar: tập các giao tác đã đọc giá trị hiện tại của ĐVDL A.

Aw: giao tác đã đọc–ghi giá trị hiện tại của ĐVDL A. Trị *Aw* được thay đổi sau mỗi lần thao tác $T : W(A)$, được chấp nhận xếp vào lịch S, cụ thể *Aw* sẽ mang giá trị T.

2.4. Thuật toán lập lịch biểu LLBII

Vào: Một tập các giao tác $\mathcal{T} = T_1, T_2, ..., T_n$.

Ra: Một lịch biểu khả tuần tự S của $\mathcal{T}$.

Thuật toán xây dựng lịch biểu được tiến hành theo các bước sau:

Bước 1. Khởi động:

S là danh sách rỗng $S = ( )$.

M là ma trận đặc trưng cấp $n \times n$, $M(i,j) = 0$ $(\forall i = 1, ..., n,\ j = 1, ...n)$.

Đối với mỗi ĐVDL A khởi trị $Ar = \{\}$ và giá trị Aw = null. Chú ý *Ar* là một tập trong khi đó *Aw* là một giá trị đơn.

Bước 2. Chọn thao tác:

Lấy thao tác t đầu tiên còn lại của một giao tác T_i bất kỳ. Có các khả năng sau:

(1) Nếu t là thao tác $R(A)$:

(1.1) Nếu Aw = null hoặc $Aw = T_i$ thì thao tác được chấp nhận.

(1.2) Nếu $Aw = T_k$ và $T_k \neq T_i$ có nghĩa là T_i bắt buộc phải đi sau T_k trong đồ thị đợi, do đó nó phải đi sau tất cả các giao tác đi trước T_k và thông tin về các giao tác đó được thể hiện trong cột thứ k của ma trận M. Tính $z = c^i \vee c^k$ (với c^i là cột thứ i của M và c^k là cột thứ k của M), nếu $z_i = 0$ thì thao tác được chấp nhận và các phần tử của cột i được thay bởi các giá trị tương ứng của z' tạo từ z theo nguyên tắc sau:

$$z'_j = 1 \text{ với } j = k,$$

$$z'_j = z_j \text{ với } j \neq k,$$

đồng thời cho $Ar = Ar \cup \{T_i\}$. Trong trường hợp ngược lại toàn bộ các thao tác của T_i (hoặc T_k) đã xếp trong S sẽ phải được hủy bỏ để khởi động lại và các phần tử trên hàng và cột i (hoặc k) của M được gán bằng 0.

(2) Nếu t là thao tác $W(A)$:

(2.1) Nếu tập $Ar \cup \{Aw\} = \{\}$ thì t được chấp nhận.

(2.2) Ngược lại, với mỗi cột $c^{k_1}, c^{k_2}, ..., c^{k_h}$ ứng với các giao tác thuộc tập $Ar \cup \{Aw\}$ mà khác với T_i, tính $z = c^{k_1} \vee c^{k_2} \vee ... \vee c^{k_h} \vee c^i$. Nếu $z_i = 0$ thì thao tác được chấp nhận và các phần tử của cột c^i được thay bởi các giá trị tương ứng của z' được tạo từ z theo nguyên tắc:

$$z'_j = 1 \text{ với } j \text{ là một trong các giá trị } k_1, k_2, ..., k_h,$$

$$z'_j = z_j \text{ trong các trường hợp còn lại,}$$

đồng thời gán $Aw = T_i$ và $Ar = \{\}$. Trong trường hợp ngược lại toàn bộ các thao tác của giao tác T_i (hoặc những thao tác trong các giao tác $T_{k_1}, T_{k_2}, ..., T_{k_h}$ mà làm cho $z_i = 1$) đã xếp trong S sẽ phải được hủy bỏ để khởi động lại và các phần tử trên hàng và cột của ma trận M tương ứng với giao tác này được gán về 0.

Bước 3. Tiếp tục:
Lặp lại bước 2 cho tới khi toàn bộ các thao tác đã được xếp.

Bước 4. Kết thúc:
Danh sách thứ tự thực hiện của S chính là lịch biểu khả tuần tự.

Theo thuật toán ở bước 2, nếu giao tác T_i được hủy bỏ và bắt đầu lại, có khả năng thuật toán sẽ không kết thúc. Để tránh trường hợp này, chúng ta có thể dàn xếp việc lựa chọn thao thác đầu tiên của một trong các giao tác theo nguyên tắc nào đó như là chỉ bắt đầu lại giao tác đã bị hủy bỏ khi các giao tác mà trực tiếp làm cho nó bị hủy bỏ đã kết thúc hoặc theo nguyên tắc sẽ loại bỏ hẳn thao tác sau khi cho phép nó khởi động lại một số lần nhất định. Nếu không hủy bỏ T_i mà hủy bỏ những thao tác khác có liên quan đến khả năng làm mất tính khả tuần tự của S thì thuật toán sẽ luôn kết thúc. Trong trường hợp giao tác T_i có nhiều đụng độ dữ liệu với các giao tác khác thì lựa chọn này sẽ không tốt vì có nhiều giao tác phải hủy bỏ trong khi đó chỉ cần hủy bỏ T_i.

Tính đúng đắn của thuật toán trên được chỉ ra bởi mệnh đề sau:

Mệnh đề 1. *Thuật toán LLBII là đúng đắn, nghĩa là luôn lập được một lịch biểu S khả tuần tự từ các giao tác* $T_1, T_2, ..., T_n$ *(với* $n \geq 1$*).*

Chứng minh. Khi thuật toán kết thúc, rõ ràng là các phần tử trên đường chéo chính của ma trận M đều bằng 0 ($M(i,i) = 0 \; \forall i = 1, ..., n$). Theo Định lý 1, nếu M là ma trận đặc trưng của S ta có thể suy ra S là khả tuần tự. Ta sẽ chứng minh rằng, khi thuật toán kết thúc, ma trận M đúng là ma trận đặc trưng của lịch biểu S thu được. Muốn vậy ta cần chỉ ra rằng khi lịch biểu chuyển từ trạng thái S^x sang trạng thái S^{x+1} thì dữ liệu trong ma trận M được cập nhật lại theo bước 2 là đúng. Thực vậy, khi thao tác t được chấp nhận đưa vào lịch, nếu rơi vào trường hợp (1.1) thì trong đồ thị của S^{x+1}, T_i không bắt buộc phải đi sau bất cứ giao tác nào, do vậy trạng thái của M không thay đổi khi bổ sung thêm thao tác này. Trong trường hợp (1.2), do ĐVDL A được ghi bởi T_k nào đó, do vậy T_k phải đứng trước T_i trong đồ thị của S^{x+1}, đương nhiên những giao tác đứng trước T_k trong đồ thị cũng sẽ đứng trước T_i. Còn trong trường hợp (2.1), do chưa có giao tác nào tác động lên ĐVDL A nên không có ràng buộc thêm về việc T_k bắt buộc đi sau giao tác nào, do vậy

ma trận M không thay đổi khi bổ sung thao tác này. Trong trường hợp (2.2), do thao tác của T_i ghi vào ĐVDL A đã được đọc hay đọc-ghi bởi một số giao tác nào đó khác T_i, nên T_i buộc phải đi sau các giao tác này và cả những giao tác đi trước các giao tác này. Như vậy cách làm ở bước 2 để tạo ra giá trị mới cho ma trận M khi bổ sung thao tác t là đúng đắn. □

Chúng ta xét ví dụ sau:

Ví dụ. Giả sử các giao tác T_1, T_2, T_3, T_4 như sau:

$T_1 : R(A), R(C)$; $T_2 : R(A), R(B)$; $T_3 : W(A), W(B)$; $T_4 : R(B), W(A)$.

Chúng ta có thể lựa chọn thao tác đầu tiên của các giao tác theo phương pháp lần lượt từ T_1, T_2, T_3, T_4, T_1... Các bước thực hiện được thể hiện trong bảng 1.

Bảng 1. Giá trị của ma trận M được xây dựng theo các bước

TT	Chọn thao tác	S	M	Tập đọc (Xr)	Tập ghi (Xw)
0			0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	A: B:	A: B:
1	$T_1 : R(A)$	$T_1 : R(A)$	Không đổi	$A : T_1$ B:	Không đổi
2	$T_2 : R(A)$	$T_1 : R(A)$; $T_2 : R(A)$	Không đổi	$A : T_1, T_2$ B:	Không đổi
3	$T_3 : W(A)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$	0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0	A: B:	$A : T_3$ B:
4	$T_4 : R(B)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$; $T_4 : R(B)$	Không đổi	A : $B : \{T_4\}$	Không đổi
5	$T_1 : W(B)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$; $T_4 : R(B)$ $T_1 : W(B)$	0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0	A: B:	$A : T_3$ $B : T_1$
6	$T_2 : R(B)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$; $T_4 : R(B)$ $T_1 : W(B)$; $T_2 : R(B)$	0 1 1 0 0 0 1 0 0 0 0 0 1 1 0 0	A: $B : T_2$	Không đổi
7	$T_3 : W(B)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$; $T_4 : R(B)$ $T_1 : W(B)$; $T_2 : R(B)$ $T_3 : W(B)$	0 1 1 0 0 0 1 0 0 0 0 0 1 1 1 0	A: B:	$A : T_3$ $B : T_3$
8	$T_4 : W(A)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A), T_1 : W(B)$ $T_2 : R(B)$; $T_3 : W(B)$ T_4 bị loại bỏ vì xung đột với T_3	0 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0	A: B:	$A : T_3$ $B : T_3$

9	$T_4 : R(B)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$; $T_1 : W(B)$ $T_2 : R(B)$; $T_3 : W(B)$ $T_4 : R(B)$	0 0 0 0	1 0 0 0	1 1 0 0	0 0 1 0	A: $B : \{T_4\}$	Không đổi
10	$T_4 : W(A)$	$T_1 : R(A)$; $T_2 : R(A)$ $T_3 : W(A)$; $T_1 : W(B)$ $T_2 : R(B)$; $T_3 : W(B)$ $T_4 : R(B)$; $T_4 : W(A)$	0 0 0 0	1 0 0 0	1 1 0 0	0 0 1 0		$A : T_4$ $B : T_3$

Khi lịch biểu ở trạng thái S^7, thao tác $T_4 : W(A)$ được xét. Vì $Aw = T_3$ và $Ar = \{\}$ nên T_4 phải đứng sau T_3 do đó $z = c^4 \vee c^3 = (1, 1, 0, 1)$ và như vậy giao tác T_4 cần phải được hủy bỏ và bắt đầu lại. Tiếp tục quá trình theo thuật toán, lịch biểu S thu được như sau:

$\{T_1 : R(A);\ T_2 : R(A);\ T_3 : W(A);\ T_1 : W(B);\ T_2 : R(B);\ T_3 : W(B);\ T_4 : R(B);\ T_4 : W(A)\}$

4. ĐÁNH GIÁ ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

Mệnh đề 2. *Trong trường hợp xấu nhất thuật toán LLBII đòi hỏi thời gian $a.k.n^2$. Trong đó k là tổng số các thao tác của các giao tác tham gia xếp lịch và a là số lần tối đa hệ thống cho phép một giao tác được khởi động lại (giá trị của a là tùy chọn).*

Chứng minh. Để đánh giá độ phức tạp của thuật toán (ký hiệu là ĐPT), chúng ta xem xét độ phức tạp của thuật toán ở bước 2 (ký hiệu là ĐPTb2). Xét hai trường hợp như sau:

Trường hợp 1: Thao tác cần xét đưa vào lịch biểu là thao tác $T_i : R(A)$. Khi đó nếu $Aw = \text{null}$ thì không phải làm gì. Nếu $Aw = T_j$, phải tính $z = c^i \vee c^j$, do vậy ĐPTb2 $= O(n)$.

Trường hợp 2: Thao tác cần xét đưa vào lịch biểu là thao tác $T_i : R(A)$. Trong trường hợp xấu nhất tập $Ar \cup \{Aw\}$ có $n-1$ giao tác còn lại và chúng ta phải tính $z = c^1 \vee c^2 \vee ... \vee c^n$, do vậy ĐPTb2 $= O(n^2)$.

Như vậy trong trường hợp xấu nhất ĐPT $= a.k.n^2$. Trong đó $k = \sum h_i$ với $i = 1, ..., n$, a là số tối đa hệ thống cho phép giao tác được khởi động lại.

Giá trị của a là tùy chọn. Việc lựa chọn a phụ thuộc vào mức độ yêu cầu về thời gian xếp lịch, số giao tác cùng xảy ra đồng thời trung bình và khả năng đụng độ dữ liệu của các giao tác. Trong trường hợp có các giao tác nào đó đã khởi động lại a lần mà vẫn chưa xếp được, thuật toán có thể xếp lần lượt các giao tác đó vào phía cuối của lịch S.

Nhận xét: Qua đánh giá độ phức tạp tính toán, chúng ta nhận thấy thuật toán đơn giản, đòi hỏi về bộ nhớ không lớn (ngoại trừ các vùng để lưu giữ các giao tác và lịch biểu được lập cần phải sử dụng thêm số lượng bộ nhớ như sau: nếu n là số 1 byte, đối với mỗi đơn vị dữ liệu cần n byte cho Ar và 1 byte cho Aw. Để lưu giữ ma trận M cần $n \times n$ byte). Độ phức tạp tính toán phụ thuộc bậc 2 vào số các giao tác và bậc nhất vào tổng số các thao tác của các giao tác. Thuật toán sẽ hoạt động tốt trong trường hợp các thao tác đọc dữ liệu chiếm ưu thế.

TÀI LIỆU THAM KHẢO

[1] Ceri S., Pelagatti G., *Distributed Databases Principles and Systems*, Mc Graw-Hill, 1984.
[2] Ullman J., *Principles of Database and Knowledge-Base System*, Vol. 1, Prentice-Hall, 1987.

Nhận bài ngày 1 - 9 - 1999
Nhận lại sau khi sửa ngày 4 - 10 - 1999

Viện Công nghệ thông tin