Chương 3.

Quản lý các giao dịch phân tán.

Chương này gồm các nội dung liên quan đến việc quản lý các giao dịch phân tán như sau:

3.1 Các khái niệm cơ bản trong giao dịch phân tán.

- 3.1.1 Khái niêm về giao dịch trong CSDL phân tán
- 3.1.2 Khái niệm về sự đặt khóa
- 3.1.3 Các lịch biểu trong các giao dịch phân tán

3.2 Quản lý giao dịch phân tán bằng đặt khóa

- 3.2.1 Mô hình khóa tổng quát.
- 3.2.2 Một số tình huống xung đột khóa
- 3.2.3 Mô hình khóa có phân biệt đọc / ghi

3.3 Quản lý giao dịch phân tán bằng thời dấu.

- 3.3.1 Các khái niệm.
- 3.3.2 Điều khiển tương tranh bằng thời dấu

3.4 Hợp thức hóa các giao dịch phân tán.

- 3.4.1 Mục đích của hợp thức hoá
- 3.4.2 Giao thức hợp thức hóa hai giai đoạn

Bài tập chương 3

3.1 CÁC KHÁI NIỆM CƠ BẢN

Mở đầu: Trong CSDL phân tán, có nhiều ứng dụng đòi hỏi phải thực hiện nhiều giao dịch cùng một lúc. Chẳng hạn: hệ thống bán vé của một hãng hang không, tại một thời điểm có thể nhiều đại lý cùng bán vé cho một chuyến bay, do đó danh sách hành khách và số ghế đã bán có thể thay đổi liên tục. Nếu có hai giao dịch bán vé/đặt chỗ cùng thực hiện thì có thể dẫn tới cùng một chỗ ngồi được bán/đặt chỗ 2 lần, sẽ dẫn đến nhiều xung đột, trục trặc cho hệ thống.

Như vậy đặt ra vấn đề: cần phải quản lý các giao dịch trên CSDL phân tán (còn gọi là các giao dịch phân tán | Distributed transactions). Trong chương này ta chỉ nghiên cứu 2 vấn đề quan trọng của việc quản lý các giao dịch phân tán:

- Điều khiển tương tranh các giao dịch phân tán.
- Xử lý hợp thức hóa các giao dịch phân tán.

Trước hết, ta xét một số khái niệm cơ bản trong quản lý giao dịch phân tán.

3.1.1 Khái niệm về giao dịch trong CSDL phân tán

Khái niệm giao dịch trong CSDL

- Một *giao dịch* được tạo bởi một dãy các *thao tác* đọc và ghi trên CSDL cùng với các bước tính toán cần thiết. (có thể hiểu giao dịch là thực hiện 1 chương trình, còn thao tác là thực hiện câu lệnh trong chương trình đó)
- Định nghĩa khác của Ullman [1988]: Giao dịch là sự thực hiện của một chương trình có các câu vấn tin truy xuất đến CSDL.

Hoạt động của một giao dịch trong CSDL

Một giao dịch sẽ đọc/ghi một đơn vị dữ liệu (*mục dữ liệu* | *item*) của CSDL thông qua một bộ đệm của giao dịch (vùng làm việc riêng - *private work space*). Mỗi giao dịch có thể thực hiện các thao tác tính toán dẫn đến thay đổi giá trị của dữ liệu, nhưng những thay đổi này sẽ không tác động ngay vào CSDL cho đến khi các giá trị mới được ghi vào CSDL nhờ một thủ tục hợp thức (*commited*) khi kết thúc giao dịch. Giao dịch chỉ đọc sẽ không làm thay đổi CSDL.

Các giao dịch phân tán

Các giao dịch được thực hiện trên một CSDL phân tán gọi là các giao dịch phân tán. Một giao dịch phân tán bao gồm nhiều *giao dịch con* thực hiện trên các trạm.

Tính nguyên tố của một giao dịch phân tán: được hiểu là giao dịch đó đã được thực hiện trọn vẹn, khi tất cả các giao dịch con của nó đều đã được thực hiện, hoặc trái lại, nếu có dù chỉ một giao dịch con trên một trạm chưa được thực hiện thì toàn bộ giao dịch phân tán là chưa được thực hiện (chưa commited, chưa được ghi các thay đổi vào CSDL).

Như vậy giao dịch phân tán chỉ có hai trạng thái kết quả: đã thực hiện hoặc không thực hiện. Các thủ tục hợp thức hóa sẽ đảm bảo tính nguyên tố của các giao dịch phân tán.

- Một số thí dụ về giao dịch phân tán:
  - Truy vấn tài khoản/chuyển tiền qua dịch vụ Internet banking, rút tiền ở máy ATM...
  - Mua vé/đặt chỗ máy bay (qua mạng) ...

Giao dịch và sự nhất quán của CSDL

Mọi CSDL tập trung hay phân tán luôn luôn là nhất quán (thỏa mãn các ràng buộc toàn vẹn) trước và sau khi thực hiện một giao dịch. Trong khi giao dịch đang thực hiện, CSDL có thể tạm thời không nhất quán, nhưng sau khi thực hiện xong giao dịch, CSDL được cập nhật và phải ở trạng thái nhất quán, hệ thống sẽ hủy bỏ giao dịch nếu kết quả cập nhật của giao dịch làm cho CSDL trở nên không nhất quán.

Chẳng hạn: nếu nhập tuổi của cha ít hơn tuổi của con, hay chuyển khoản số tiền lớn hơn số dư thì sẽ làm cho CSDL trở nên không nhất quán, trong trường hợp này giao dịch bị hủy.

3.1.2 Khái niệm về sự đặt khóa (Locking)

Khái niệm mục dữ liệu.

Để quản lý các hoạt động tương tranh (các hoạt động đồng thời) trên một CSDL, người ta chia CSDL thành các đơn vị dữ liệu nhỏ nhất, cần được truy xuất có điều khiển, gọi là các *mục dữ liệu* (*items*) hay các hạt dữ liệu.

Bản chất và kích thước của các mục dữ liệu do người thiết kế hệ thống quyết định. Chẳng hạn: đối với mô hình quan hệ, ta có thể chọn các mục dữ liệu lớn như các quan hệ, hay chọn các mục dữ liệu nhỏ hơn như các bộ hay các thành phần của các bộ. Một hệ thống hạt mịn (*fine- grained)* sẽ sử dụng các mục dữ liệu nhỏ, còn hệ thống hạt thô (*coarse grained*) sẽ sử dụng các mục dữ lệu có kích thước lớn.

Việc chọn kích thước mục dữ liệu là rất quan trọng trong việc quản lý CSDLPT: nếu chọn hạt thô sẽ giảm được chi phí quản lý các mục dữ liệu, trái lại nếu chọn hạt mịn sẽ cho phép nhiều giao dịch thực hiện song song. Xác suất để các giao dịch có yêu cầu truy xuất cùng một mục dữ liệu với hệ thống hạt mịn sẽ nhỏ hơn khi chọn hệ thống hạt thô.

- Việc lựa chọn kích thước mục dữ liệu phụ thuộc vào các thao tác điển hình của các giao dịch trong CSDL:
  - Nếu thao tác điển hình là đọc ghi các bộ của một quan hệ qua một chỉ mục: ta chọn mỗi bộ làm một mục dữ liệu.
  - Nếu thao tác điển hình là kết nối các quan hệ, ta chọn mỗi quan hệ làm một mục dữ liệu. (vì phải truy xuất tất cả các bộ của mỗi quan hệ cho một phép kết nối)

Khái niệm sự đặt khóa (locking).

- Sự đặt khóa (còn gọi là khóa chốt, hay locking) trên một mục dữ liệu là việc hệ thống trao quyền truy xuất mục dữ liệu đó cho một giao dịch, hoặc thu hồi lại quyền này.
  - Ký hiệu LOCK A là chỉ mục dữ liệu A đã được trao quyền truy xuất cho một giao dịch.
  - Ký hiệu UNLOCK A là chỉ mục dữ liệu A đã thu hồi quyền truy xuất của mọi giao dịch.
- Có thể phân chia sự đặt khóa thành hai loại:
  - Khóa đọc (Read Lock): RLOCK A: Chỉ cho phép giao dịch đọc mục dữ liệu A, mà không được phép ghi trên A.
  - Khóa ghi (Write Lock): WLOCK A: Cho phép giao dịch đọc và ghi mục dữ liệu A.
- Tại mỗi thời điểm, chỉ có một tập con các mục dữ liệu bị khóa. Bộ quản lý khóa sẽ lưu trữ các khóa hiện hành trong "bảng khóa", là tập các bộ sau: Lock table = {(I , L , T)}

Trong đó, mỗi bộ của bảng gồm 3 thành phần I, L, T với ý nghĩa như sau: Giao dịch T được đặt một khóa L (với L là khóa RLOCK hoặc WLOCK) trên mục dữ liệu I. Như vậy khóa L không chỉ gắn với mỗi giao dịch T, mà còn gắn với mục dữ liệu I.

3.1.3 Các lịch biểu trong các giao dịch phân tán

Các khái niệm về Lịch biểu trong giao dich phân tán (Schedule).

- Lịch biểu cho một tập các giao dịch là một thứ tự (các bước) thực hiện các thao tác cơ bản (như đặt khóa, đọc, ghi...) của các giao dịch trên một CSDL phân tán. Kết quả của một lịch biểu là kết quả được cập nhật vào CSDL sau khi thực hiện các giao dịch theo lịch biểu này.
- Lịch biểu gọi là *tuần tự* nếu tất cả các bước thao tác của cùng một giao dịch xảy ra liền nhau. Lịch biểu tuần tự luôn cho kết quả như mong muốn (kết quả đúng)
- Lịch biểu gọi là *khả tuần tự* nếu các bước của các giao dịch khác nhau có thể thực hiện xen kẽ nhau, nhưng cho kết quả như kết quả của một lịch biểu tuần tự (kết quả đúng).
- Lịch biểu gọi là *bất khả tuần tự* nếu các bước của các giao dịch khác nhau có thể thực hiện xen kẽ nhau, nhưng cho kết quả khác với kết quả của một lịch biểu tuần tự (kết quả sai).

Thí dụ 3.1: Giả sử có 2 giao dịch T1 và T2, thực hiện các thao tác theo chương trình P như sau::

P: { T1: Read A; A := A - 10; Write A; Read B; B := B +10; Write B;
T2: Read B; B := B -20; Write B; Read C; C := C +20; Write C;

Xét 3 lịch biểu của 2 giao dịch T1 và T2:

(a)

T1	T2
READ A	
A := A-10	
WRITE A	
READ B	
B := B+10	
WRITE B	
	READ B
	B := B-20
	WRITE B
	READ C
	C := C+20
	WRITE C

(b)

T1	T2
READ A	
	READ B
A := A-10	
	B := B-20
WRITE A	
	WRITE B
READ B	
	READ C
B := B+10	
	C := C+20
WRITE B	
	WRITE C

(c)

T1	T2
READ A	
A := A-10	
	READ B
WRITE A	
	B := B-20
READ B	
	WRITE B
B := B+10	
	READ C
WRITE B	
	C := C+20
	WRITE C

Hình 3.1. Các lịch biểu (a), (b) và (c) của hai giao dịch T1, T2.

Các giao dịch T1 và T2 có thể hiểu là dịch vụ chuyển tiền giữa các tài khoản A, B và C, cụ thể A chuyển 10 (đơn vị tiền), B chuyển 10 còn C nhận 20.

Với mọi lịch biểu tuần tự thì tổng A + B + C không đổi trước và sau khi thực hiện các giao dịch. Chẳng hạn, nếu ban đầu A = B = C =100, thì sau khi thực hiện lịch biểu tuần tự cho 2 giao dịch, kết quả trong CSDL sẽ là: A= 90, B = 90, C = 120, tổng A + B + C trước và sau khi

thực hiện các giao dịch đều là 300, đây là nguyên tắc cốt lõi của dịch vụ chuyển tiền giữa các tài khoản trong hệ thống ngân hàng.

Trong các lịch biểu trên:

(a) Là lịch biểu tuần tự, kết quả đúng là: A:= A – 10 ; B: = B – 10; C := C + 20.
(tổng A + B + C = 300)

(b) Là lịch biểu khả tuần tự , cho kết quả như lịch biểu tuần tự (a), (tổng A + B + C = 300)

(c) Là lịch biểu bất khả tuần tự tuần tự: cho kết quả A:= A – 10 ; B: = B + 10; C := C + 20, khác với kết quả của lịch biểu tuần tự (a), tổng A + B + C = 320, tăng thêm 20, điều này thật nguy hiểm cho ngân hàng!! (do T1 ghi giá trị B := B+10 = 110, ghi đè lên kết quả của B:= B – 20 = 80 mà T2 đã ghi trước đó, cuối cùng có: A = 90, B = 110, C = 120!)

Kết luận: Điều khiển tương tranh các giao dịch phân tán là tìm một lịch biểu khả tuần tự cho một tập các giao dịch, nhằm tăng khả năng xử lý song song cho hệ thống, và ngăn chặn việc phát sinh các lịch biểu bất khả tuần tự.

3.2 Điều khiển tương tranh bằng đặt khóa

Mở đầu: Điều khiển tương tranh bằng đặt khóa là việc sử dụng một bảng khóa cho các mục dữ liệu đối với một tập giao dịch để ngăn chặn việc phát sinh các lịch biểu bất khả tuần tự. Chúng ta phân biệt các mô hình khóa tổng quát và mô hình khóa phân biệt đọc / ghi (RLOCK/WLOCK).

3.2.1 Mô hình khóa tổng quát.

3.2.1.1 Thí dụ về sự phát sinh lịch biểu bất khả tuần tự.

Thí dụ 3.2. Xét 2 giao dịch T1 và T2 cần truy xuất mục dữ liệu A (A có giá trị nguyên), rồi cộng thêm 1 vào A. Hai giao dịch này là các bước thực hiện của cùng một chương trình P dưới đây:

P: READ A; A := A+1 ; WRITE A.

Khi một giao dịch T1 hay T2 thực hiện, nó sẽ đọc A vào vùng đệm của nó, cộng thêm 1 vào giá trị của A, rồi ghi kết quả vào mục dữ liệu A trong CSDL khi kết thúc giao dịch. Giả sử giá trị ban đầu của A là 5, hoạt động của T1 và T2 theo chương trình P được mô tả trong hình sau:

Thứ tự (steps)	T1	T2	A trong CSDL	A trong vùng đệm của T1	A trong vùng đệm của T2
1	READ A		5	5	NULL
2		READ A	5	5	5
3	A:= A+1		6	6	5
4		A:= A+1	6	6	6
5	WRITE A		6	6	6
6		WRITE A	6	NULL	6

Hình 3.2. Hoạt động của 2 giao dịch T1 và T2

Nếu 2 giao dịch T1 và T2 được thực hiện theo một lịch biểu tuần tự, mỗi giao dịch đều cộng thêm 1 vào A, thì A phải tăng thêm 2, nhưng theo các bước của lịch biểu trên thì kết quả A

chỉ tăng thêm 1, đây là kết quả không mong muốn (kết quả sai) . Hãy tưởng tượng A là số vé máy bay đã bán, thì kết quả sai như trên thật là nguy hiểm!.

Lịch biểu tự phát sinh trên đây của 2 giao dịch T1 và T2 là do các hoạt động tương tranh của 2 giao dịch T1 và T2 trên cùng mục dữ liệu A đã không được kiểm soát, đây là một lịch biểu *bất khả tuần tự.*

3.2.1.2 Điều khiển tương tranh bằng khóa.

Điều khiển tương tranh bằng khóa là việc tuần tự hóa các hoạt động tương tranh bằng việc đặt khóa, được mô tả như sau: Mỗi giao dịch trước khi truy xuất mục dữ liệu A phải làm thủ tục đặt khóa trên A (LOCK A), sau khi hoàn thành các thao tác thì hủy bỏ việc đặt khóa trên A (UNLOCK A), mục dữ liệu A được giải phóng và giao dịch khác lại có thể đặt khóa trên A...

Trở lại thí dụ trên, cung cấp một khóa (tổng quát) cho A, trước khi truy xuất A, một giao dịch phải khóa A (LOCK A), sau khi thực hiện xong các thao tác, giao dịch này mở khóa cho A (UNLOCK A). Trong khi một giao dịch LOCK A, không giao dịch nào có thể truy xuất A cho nên không thể phát sinh các lịch biểu không tuần tự.

Thí dụ 3.3. Áp dụng việc đặt khóa cho các giao dịch trong thí dụ 1, chương trình thực hiện cho T1 và T2 ở thí dụ trên được sửa đổi như sau:

P*: LOCKA ; READ A; A := A+1 ; WRITE A ; UNLOCK A.

Sơ đồ hoạt động của T1 và T2 theo chương trình P* được mô tả trong hình sau:

Thứ tự (steps)	T1	T2	A trong CSDL	A trong vùng đệm của T1	A trong vùng đệm của T2
1	LOCK A		5	5	NULL
2	READ A		5	5	#
3	A:= A+1		5	6	#
4	WRITE A		6	6	#
5	UNLOCK A		6	6	#
6		LOCK A	6	NULL	6
7		READ A	6	#	6
8		A:= A+1	6	#	7
9		WRITE A	7	#	7
10		UNLOCK A	7	#	7

Hình 3.3. Hoạt động của 2 giao dịch T1 và T2 có đặt khóa

Do có sự đặt khóa, các giao dịch không thể tiến triển tự do, mà giao dịch T2 phải chờ giao dịch T1 thực hiện xong UNLOCK A thì T2 mới truy cập A được. Như vậy nhờ việc đặt khóa tổng quát, cấc giao dịch đã tự động phát sinh một lịch biểu tuần tự, sau khi T1 và T2 thực hiện xong lịch biểu này, mục dữ liệu A trong CSDL có giá trị là 7 (kết quả đúng).

3.2.2. Một số tình huống xung đột khóa

3.2.2.1 Tình huống khóa sống (live lock)

Một tình huống không mong muốn có thể xảy ra trong việc đặt khóa là một giao dịch T có thể không bao giờ nhận được khóa mà nó yêu cầu đối với một mục dữ liệu.

Chẳng hạn, với thí dụ 2, giả sử khi T1 giải phóng mục dữ liệu A (UNLOCK A), thì khóa này được trao cho T2, tuy nhiên có thể xảy ra tình huống là trong khi T2 chờ nhận khóa trên A, thì một giao dịch khác T3 đã được đặt khóa trên A, sau T3 lại là T4 và v.v...như vậy rất có thể giao dịch T2 không bao giờ nhận được khóa trên A, dù rằng nó có cơ hội để nhận. Tình huống như vậy được gọi là "khóa sống" (live-lock)

Một chiến lược để tránh ":khóa sống" là hệ thống phải ghi nhận tất cả các yêu cầu đặt khóa đối với một mục dũ liệu A, khi A đã UNLOCK thì phải trao khóa cho giao dịch đầu tiên trong số các giao dịch có yêu cầu đặt khóa trên A. Đó là nguyên tắc "xếp hàng": đến trước thì được phục vụ trước.

3.2.2.2 Tình huống khóa chết (hay tắc nghẽn | dead- lock).

Một tình huống nghiêm trọng hơn có thể xảy ra: Hai giao dịch có yêu cầu truy xuất một số mục dữ liệu, trong đó giao dịch này phải chờ giao dịch kia UNLOCK mục dữ liệu cần truy cập để thực hiện tiếp các thao tác, và không giao dịch nào tiến triển được.

Thí dụ 3.4. Giả sử có 2 giao dịch T1 và T2 đồng thời thực hiện theo các chương trình sau:

T1: LOCK A ; LOCK B ; UNLOCK A ; UNLOCK B.

T2: LOCK B ; LOCK A ; UNLOCK B ; UNLOCK A.

(Chú ý rằng T1 và T2 có thể thực hiện các tác vụ khác trên A và B)

Giả sử là T1 và T2 cùng thực hiện tại 1 thời điểm mà A và B đều đang không bị đặt khóa. T1 yêu cầu và được trao khóa trên A, T2 yêu cầu và được trao khóa trên B, sau đó T1 yêu cầu trao khóa trên B, nó phải chờ T2 giải phóng khóa này, nhưng T2 đang yêu cầu LOCK A, cũng phải chờ T1 giải phóng khóa này, giao dịch này phải chờ giao dịch kia giải phóng khóa trên mục dữ liệu cần truy cập, kết quả là cả hai giao dịch không thể tiến triển được. Đã xảy ra tình trạng tắc nghẽn (hay dead-lock) giữa 2 giao dịch.

Để khắc phục dead lock, một giao dịch phải bị hủy bỏ, và khởi động lại sau khi giao dịch kia đã thực hiện xong, tức là thực hiện một lịch biểu tuần tự cho 2 giao dịch.

3.2.2.3 Tình huống tắc nghẽn toàn cục

Tình trạng tắc nghẽn sẽ trở nên trầm trọng nếu nó xảy ra với một tập các giao dịch trên một tập các mục dữ liệu.

- Kí hiệu: Ti $\rightarrow$ Tj biểu diễn quan hệ chờ đợi : "giao dịch Ti chờ sự giải phóng một khóa được đặt bởi Tj".

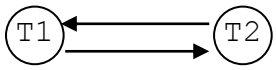
Chẳng hạn, tình huống trong thí dụ 3 được biểu diễn bởi tập các quan hệ chờ đợi:

$$\{ T2 \rightarrow T1 ; T1 \rightarrow T2 \}$$

- Đồ thị chờ đợi toàn cục cho một tập giao dịch: là một đồ thị G có hướng, trong đó các đỉnh được gán nhãn là các giao dịch, các cung được xác định như sau: từ Ti có cung đến Tj nếu giao dịch Ti chờ sự giải phóng 1 khóa được đặt bởi Tj, tức là có quan hệ chờ đợi: Ti → Tj.
- Nếu đồ thị đợi có chu trình: có tắc nghẽn toàn cục
- Khắc phục tắc nghẽn toàn cục: hủy bỏ một trong các giao dịch tham gia vào chu trình, và khởi động lại giao dịch này sau khi các giao dịch trong chu trình cũ đã được thực hiện.

Thí dụ 3.5

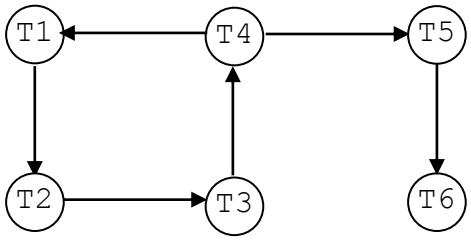
a). Chẳng hạn, với các giao dịch trong thí dụ 3, ta có đồ thị đợi:



Hình 3.4. Đồ thị đợi cho 2 giao dịch đặt khóa tổng quát

Đồ thị này có chu trình, đã xảy ra tắc nghẽn, để khắc phục tắ nghẽn, ta hủy bỏ một giao dịch và khởi động lại sau khi giao dịch kia đã hoàn thành.

b). Dưới đây là đồ thị đợi toàn cục cho 6 giao dịch:



Hình 3.5. Đồ thị đợi toàn cục cho 6 giao dịch

Các giao dịch T1, T2, T3, T4 làm thành một chu trình, xảy ra tắc nghẽn toàn cục. Để khắc phục tắc nghẽn, ta có thể hủy bỏ bất kỳ giao dịch nào trong số các giao dịch T1, T2, T3, T4, và khởi động lại sau khi các giao dịch còn lại của chu trình cũ đã được thực hiện.

3.2.3. Mô hình khóa có phân biệt đọc / ghi

Ở các phần trên, ta đã giả sử rằng khi một giao dịch khóa một mục dữ liệu A (LOCK A) thì nó sẽ toàn quyền đọc hoặc ghi trên. Tuy nhiên, thực tế có những giao dịch chỉ truy xuất A để đọc giá trị trên A, và không cần ghi trên A. Nếu phân biệt một giao dịch chỉ đọc (Read only) và một giao dịch đọc và ghi (read/write) thì việc sử dụng các khóa có phân biệt đoc/ghi sẽ cho một điều khiển hiệu quả hơn so với mô hình khóa tổng quát.

Trong mô hình khóa có phân biệt đọc/ ghi, sẽ có 3 loại khóa: RLOCK (khóa chỉ đọc), WLOCK (khóa có thể đọc/ghi) và UNLOCK (mở khóa cho mục dũ liệu). Các giao dịch sẽ đặt khóa như sau :

- Khóa đọc (Read Lock: RLOCK): khi giao dịch T muốn chỉ đọc một mục dữ liệu A, T sẽ đặt khóa RLOCK A, khi đó giao dịch T sẽ được đọc dữ liệu trên A, các giao dịch khác không được ghi vào A, nhưng vẫn được phép đọc trên A.

- Khóa ghi (Write Lock: WLOCK): khóa này có ý nghĩa như khóa tổng quát trên đây. Khi giao dịch T đã đặt khóa WLOCK A, thì không một giao dịch nào có thể thực hiện bất kỳ thao tác gì trên A, tức là mọi giao dịch khác không thể đặt bất kỳ khóa nào trên A (RLOCK hoặc WLOCK). Chỉ khi A được UNLOCK bởi giao dịch T, thì các giao dịch khác mới được phép đặt khóa (truy cập) trên A.
- UNLOCK: khi giao dịch thực hiện xong các thao tác trên A, đặt khóa UNLOCK để giải phóng mục dữ liệu này, cho phép các giao dịch khác có thể truy cập.

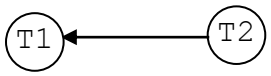
Rõ ràng với mô hình khóa đọc/ghi thì sẽ tăng hiệu quả xử lý tương tranh (đồng thời) giữa các giao dịch và giảm tắc nghẽn. Ta sẽ xem xét thí dụ sau :

Thí dụ 3.6. Trong thí dụ 3, giả sử các giao dịch T1 và T2 đọc và ghi dữ liệu trên A, nhưng cả 2 giao dịch chỉ đọc dữ liệu trên B. Áp dụng mô hình khóa đọc/ ghi cho 2 giao dịch này, ta có các chương trình cho mỗi giao dịch:

T1: WLOCK A ; RLOCK B ; UNLOCK A ; UNLOCK B.

T2: RLOCK B ; WLOCK A ; UNLOCK B ; UNLOCK A.

Đồ thị đợi toàn cục sẽ là:



Hình 3.6. Đồ thị đợi cho 2 giao dịch đã được đặt khóa đọc/ ghi

Đồ thị này không có chu trình, không có tình trạng khóa chết, không xảy ra tắc nghẽn, trong đó T1 tiến triển bình thường, T2 thực hiện RLOCK B, chờ T1 thực hiện UNLOCK A, sau đó cả hai giao dịch tiến triển cho đến khi kết thúc. Lịch biểu thực hiện các giao dịch là khả tuần tự.

3.3 ĐIỀU KHIỂN TƯƠNG TRANH BẰNG THỜI DẤU

Mở đầu: Một cách tiếp cận khác cho việc điều khiển tương tranh là đặt cho các giao dịch các *'nhãn thời gian'* (*thời dấu: Time-Stamps*). Tùy theo thời dấu của các giao dịch mà hệ thống cho phép giao dịch đó được thực hiện hay hủy bỏ.

Cũng như với mô hình điều khiển bằng đặt khóa, ta cũng có hai mô hình điều khiển bằng thời dấu: mô hình tổng quát và mô hình phân biệt đọc/ghi.

3.3.1 Các khái niệm.

3.3.1.1 Thời dấu của các giao dịch

Khi một giao dịch được khởi động, nó sẽ được gán một số hiệu duy nhất trong hệ thống gọi là *thời dấu* (hay nhãn thời gian: *Time Stamps*). Thời dấu của giao dịch sẽ quyết định việc cho phép giao dịch thực hiện các thao tác truy cập dữ liệu (read/write), hay hủy bỏ giao dịch đó. Như vậy, có thể sử dụng thời dấu để điều khiển các hoạt động tương tranh của các giao dịch trong CSDLPT.

Việc tạo các thời dấu duy nhất cho các giao dịch trong một hệ thống phân tán là việc rất khó khăn. Thường có hai cách tiếp cận:

- Cách tiếp cận tập trung: tạo thời dấu từ một bộ đệm toàn cục, được quản lý bởi một trạm duy nhất. Cách này tuy đơn giản nhưng lại phải mất một truy cập phân tán để khởi đầu việc thực hiện mỗi giao dịch.
- Cách tiếp cận phân tán: mỗi trạm đặt thời dấu một cách tự trị theo đồng hồ địa phương (bộ đếm cục bộ) và số hiệu của trạm. Một giao dịch khởi động tại trạm thứ i, tại thời điểm t theo đồng hồ tại trạm đó, sẽ được gán thời dấu <t, i>. Một số bít cuối cùng của nhãn thời gian sẽ được dùng để xác định duy nhất số hiệu trạm, chẳng hạn, nếu có không quá 256 trạm, ta chỉ cần 8 bit là đủ để xac định số hiệu mọi trạm.

3.3.1.2 Vấn đề đồng bộ hóa đồng hồ

Với cách tiếp cận phân tán, cần phải đồng bộ hóa đồng hồ (bộ đếm) của các trạm. Vấn đề này đã được giải quyết bằng cách: khi một trạm j nhận được thông báo về một giao dịch từ một trạm i với thời dấu của trạm gửi <t_i , i>, nếu thông báo mang thời dấu t_i lớn hơn giá trị đồng hồ của trạm nhận t_j , thì trạm nhận sẽ điều chỉnh đồng hồ của nó bằng thời dấu nhận được cộng thêm 1, ($t_j := t_i + 1$) và chấp nhận thực hiện giao dịch đó. Nếu giá trị đồng hồ của trạm j lớn hơn thời dấu của trạm gửi, thì trạm j sẽ giữ nguyên giá trị đồng hồ của nó, hủy bỏ giao dich từ trạm i và thông báo cho trạm i thời gian đúng hiện tại là $t_j + 1$.

Như vậy, bằng cách gửi thông báo giữa các trạm, đồng hồ các trạm sẽ được đồng bộ hóa. Vì vậy, trong các phần sau, ta luôn giả sử các thời dấu là có giá trị toàn cục và duy nhất trong hệ thống.

3.3.1.3 Thời dấu của các mục dữ liệu

Mỗi mục dữ liệu cũng được gán một thời dấu. Có thể phân biệt hai loại thời dấu cho các mục dữ liệu: thời dấu tổng quát và thời dấu phân biệt đọc ghi.

Thời dấu tổng quát: Mỗi mục dữ liệu được gán một thời dấu bằng thời dấu cao nhất của một giao dịch đã truy cập nó, không phân biệt thao tác của giao dịch là gì (đọc hay ghi)

Thời dấu phân biệt đọc ghi: Thời dấu của mục dữ liệu cũng có thể phân loại theo thao tác của giao dịch đã truy cập và gán thời dấu cho nó.

- Nếu giao dịch T có thời dấu *t*, chỉ đọc dữ liệu trên A thì A có thời dấu đọc là $RT = t$.
- Nếu giao dịch T có thời dấu *t*, đọc và ghi trên A thì ta nói A có thời dấu ghi $WT = t$.

3.3.2 Điều khiển tương tranh bằng thời dấu

Có hai mô hình điều khiển tương tranh bằng thời dấu, tùy thuộc loại thời dấu của mục dữ liệu là thời dấu tổng quát hay thời dấu có phân biệt đọc/ghi.

3.3.2.1 Mô hình thời dấu tổng quát.

Giả sử giao dịch T có thời dấu t_1, truy cập mục dữ liệu A có thời dấu t_2, khi đó quy tắc điều khiển tương tranh như sau:

- Nếu $t \geq t2$: giao dịch T được thực hiện (đọc hoặc ghi) trên A.
- Nếu $t_1 < t_2$: giao dịch T bị hủy bỏ, và có thể được khôi phục với thời dấu mới lớn hơn t_2.

Thí dụ 3.7. Xét 2 giao dịch T1 và T2 truy cập mục dữ liệu A để thực hiện các thao tác theo lịch biểu dưới đây, thời dấu của của các giao dịch và của mục dữ liệu được cho trong bảng sau:

Các giao dịch	T1	T2	Mục dữ liệu A
Thời dấu ban đầu	160	150	0
Bước (1)	READ A		160
(2)		READ A	T2 bị hủy
(3)	A := A+1		160
(4)	WRITE A		160

Hình 3.7. Lịch biểu của 2 giao dịch trong mô hình thời dấu tổng quát

Trong thí dụ trên, T2 bị hủy vì có thời dấu nhỏ hơn thời dấu mục dũ liệu A, T2 có thể được khởi động lại với thòi dấu mới lớn hơn 160.

Mô hình điều thời dấu có phân biệt đoc/ ghi sẽ cho một điều khiển "mịn" hơn và hiệu quả hơn.

3.3.2.2 Mô hình thời dấu phân biệt đọc/ghi.

Với mô hình này, mỗi mục dữ liệu A được gán hai giá trị thời dấu đọc (*RT*) và thời dấu ghi (*WT*). Khởi đầu, các mục dữ liệu gán thời dấu đọc và ghi đều bảng không, để đảm bảo rằng mọi giao dịch đầu tiên truy cập nó đều thực hiện được thao tác bất kỳ, sau đó nó sẽ được gán giá trị thời dấu đọc *RT* và/ hoặc thời dấu ghi *WT* bằng với giá trị thời dấu cao nhất của giao dịch gần nhất đọc hay ghi lên nó.

Quy tắc điều khiển tương tranh trong mô hình thời dấu đọc/ghi phát biểu như sau:

Giả sử giao dịch T có thời dấu t muốn thực hiện thao tác X (*X = Read* hoặc *Write*) trên mục dữ liệu A có thời dấu đọc là $RT = t_R$ và thời dấu ghi $WT = t_W$, khi đó:

1. Giao dịch T được thực hiện nếu $X = Read$ và $t \geq t_W$ hoặc $t \leq t_R$.
2. Giao dịch T được thực hiện nếu $X = Write$ và $t \geq t_R$, $t \geq t_W$.
3. Giao dịch T bị hủy bỏ nếu $X = Read$ và $t < t_W$, hoặc $X = Write$ và $t < t_R$.
4. Giao dịch T được được tiến triển nhưng không thực hiện gì nếu $X = Write$ và $t_R < t < t_W$. (T không làm thay đổi mục dữ liệu)

Khi một giao dịch bị hủy bỏ, nó có thể được khởi động lại với thời dấu mới lớn hơn các thời dấu của mục dữ liệu nhờ một thông báo từ trạm chứa mục dữ liệu tới trạm yêu cầu giao dịch.

Thí dụ 3.8. Xét lại thí dụ trên, với mô hình thời dấu đọc/ghi. Giả sử mục dữ dữ liệu A có các thời dấu ban đầu RT = WR = 0, các giao dịch T1 và T2 có thời dấu lần lượt là 160 và 150, thực hiện các thao tác theo lịch biểu sau:

Các giao dịch	T1	T2	Mục dữ liệu A
Thời dấu ban đầu	160	150	RT = WT = 0
Bước (1)	READ A		RT = 160
(2)		READ A	RT = 160
(3)	A := A+1		WT = 160
(4)	WRITE A		WT = 160

Hình 3.8. Lịch biểu của 2 giao dịch trong mô hình thời dấu đọc/ ghi

Trong thí dụ này, không giao dịch nào bị hủy bỏ, như vậy điều khiển tương tranh bằng thời dấu có phân biệt đọc ghi là hiệu quả hơn mô hình thời dấu tổng quát.

Thí dụ 3.9. Xét các giao dịch trong lịch biểu dưới đây. Giao dịch T1 có thời dấu 200, T2 có thời dấu 150, T3 có thời dấu 175 thực hiện truy xuất các mục dữ liệu A, B và C có cácthời dấu ban đầu RT = WR = 0,

Các giao dịch	T1	T2	T3	A	B	C
Thời dấu ban đầu	$t_1 = 200$	$t_2 = 150$	$t_3 = 175$	RT=WT=0	RT=WT=0	RT=WT=0
Bước (1)	READ B				RT = 200	
(2)		READ A		RT = 150		
(3)			READ C			RT = 175
(4)	WRITE B				WT = 200	
(5)	WRITE A			WT = 200		
(6)		WRITE C				T2 không ghi được trên C
(7)			WRITE A	T3 được ghi A		

Hình 3.9. Lịch biểu của 3 giao dịch trong mô hình thời dấu đọc/ ghi

Trong thí dụ này:

- giao dịch T1 được thực hiện.
- Giao dịch T2 bị hủy bỏ do thao tác (6): WRITE C không ghi được trên C (do : $t_2 <$ *RT(C),* vi phạm quy tắc thời dấu)
- Giao dịch T3 không bị hủy bỏ nhưng không ghi gì lên A do : $RT(A) < t_3 < WT(A)$

Kết luận: Điều khiển tương tranh bằng thời dấu cho phép 1 giao dịch được tiến triển tự do, chừng nào nó không vi phạm thứ tự các giao dịch về thời dấu. Những xung đột dẫn đến các thực thi bất khả tuần tự sẽ được phát hiện và điều khiển đơn giản bằng việc hủy bỏ các giao dịch vi phạm, và có thể được khởi động lại với thời dấu lớn hơn. Nhược điểm của cách tiếp cận này là số các giao dịch hủy bỏ và tái khởi động có thể khá lớn khi có mức tương tranh cao, điều này làm tăng cho phí quản lý các giao dịch phân tán.

3.4 HỢP THỨC HÓA CÁC GIAO DỊCH PHÂN TÁN

3.4.1 Mục đích của hợp thức hoá.

Một giao dịch phân tán (toàn cục) được thực hiện bằng tập hợp các giao dịch con (các giao dịch thành phần, giao dịch cục bộ) trên các trạm.

Mục đích của hợp thức hóa là đảm bảo tính nguyên tố của các giao dịch toàn cục: giao dịch hoặc được thực hiện ở mức toàn cục (khi tất cả các giao dịch cục bộ cùng được hợp thức), có nghĩa là tích hợp thực sự các cập nhật của nó vào CSDL, hoặc giao dịch bị huỷ bỏ (khi có một trong các giao dịch cục bộ chưa được hợp thức),

Một cách tiếp cận trực tiếp và ngây thơ để hợp thức một giao dịch phân tán, là xem các giao dịch con thành phần như một giao dịch cục bộ trên các trạm và có thể được hợp thức hóa độc lập với các giao dịch cục bộ khác. Tuy nhiên theo cách này một số giao dịch cục bộ có thể có thể được hợp thức hóa trong khi những giao dịch khác lại bị hủy bỏ. Như vậy cách tiếp cận này vi phạm điều kiện cơ bản là tính nguyên tố của các giao dịch phân tán, do không có sự đồng bộ giữa các giao dịch cục bộ tham gia vào giao dịch toàn cục.

Người ta đưa ra một giải pháp là sử dụng một giao thức hợp thức hóa hai giai đoạn hay giao thức hợp thức hóa hai phase, được thực hiện bởi các trạm tham gia vào giao dịch toàn cục dưới sự điều khiển của một trạm được phân công làm trạm điều phối.

3.4.2 Giao thức hợp thức hóa hai giai đoạn

Giao thức hợp thức hóa hai gia đoạn (hay hợp thức hóa hai pha: two-phases commit protocol) gồm hai giai đoạn sau:

1. Phase 1. Giai đoạn chuẩn bị: trạm điều phối yêu cầu mỗi trạm tham gia chuẩn bị cho sự hợp thức hóa.
2. Phase 2. Giai đoạn hợp thức : Trạm điều phối sẽ ra lệnh cho tất cả các trạm tham gia hợp thức hoá các cập nhật vào CSDL, nếu như tất cả các trạm đã hoàn thành phase 1 hoặc huỷ bỏ tất cả các giao dịch con sau một thời gian trễ.

Mỗi trạm tại đó thực hiện một giao dịch con gọi là trạm tham gia. Trạm điều phối có thể là một trong những trạm tham gia, và như vậy nó đóng cả hai vai trò.

Giao thức hợp thức hoá 2 phase sẽ đảm bảo tất cả các trạm tham gia sẽ hợp thức hoá hoặc huỷ bỏ tất cả. Vì vậy nó đảm bảo tính nguyên tố của các giao dịch toàn cục. Quyết định toàn cục là hợp tác hoá giao dịch T hoặc huỷ bỏ thì do trạm điều phối đưa ra sau giai đoạn 1.

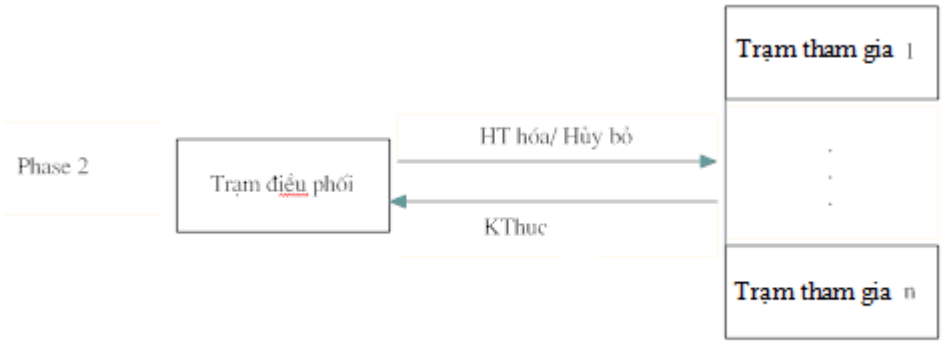
Mô tả quá trình hợp thức hóa hai giai đoạn:

Trong khi thực hiện giao thức, các trạng thái khác nhau của trạm điều phối và trạm tham gia sẽ được ghi vào nhật ký để khi có sự cố, thủ tục phục hồi có thể tiếp tục các phase cần thiết của giao thức.

- Đầu tiên, trạm điều phối khởi phát giai đoạn "chuẩn bị" bằng việc gửi thông báo "chuẩn bị" cho mọi trạm tham gia. Nhận được thông báo này, mỗi trạm tham gia thực hiện các thao tác đảm bảo cho nó có thể hợp thức hay hủy bỏ các giao dịch con khi xảy ra bất kỳ sự cố nào. Nếu trạm tham gia có khả năng hợp thức, nó sẽ gửi lại thong báo "Sẵn sàng" cho trạm điều phối, trường hợp ngược lại nó gửi thông báo ":*không sẵn sàng*".
- Sau 1 khỏng thời gian quy định, nếu trạm điều phối nhận được thông báo sẵn sàng từ tất cả các trạm tham gia, nó sẽ ra thông báo *"quyết định hợp thức hóa"* cho tất cả các tram. Trường hợp trái lại khi có ít nhất 1 trạm gửi thông báo không sẵn sàng, hoặc quá thời hạn quy định (time-out) mà không gửi thông báo, thì trạm điều phối sẽ ra quyết định toàn cục tới tất cả các trạm *"Hủy bỏ hợp thức hóa"*. Nhận được quyết định toàn cục, các trạm tham gia sẽ đồng loạt hợp thức hóa hay hủy bỏ hợp thức hóa, và thông báo cho trạm điều phối là đã kết thúc công việc.

Sơ đồ hợp thức hoá 2 phase:

Hình 3.10. Phase 1 của giao thức hợp thức hóa 2 phase

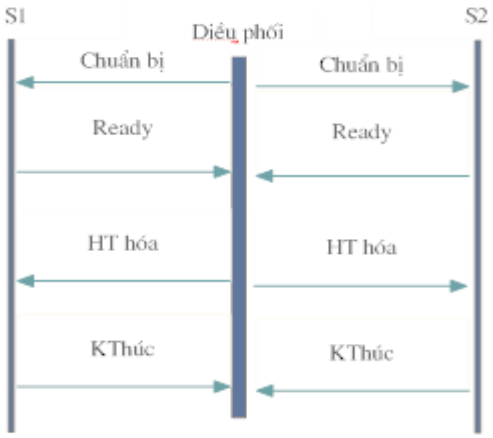


Hình 3.11. Phase 2 của giao thức hợp thức hóa 2 phase

3.4.3 Một số kịch bản hợp thức hóa

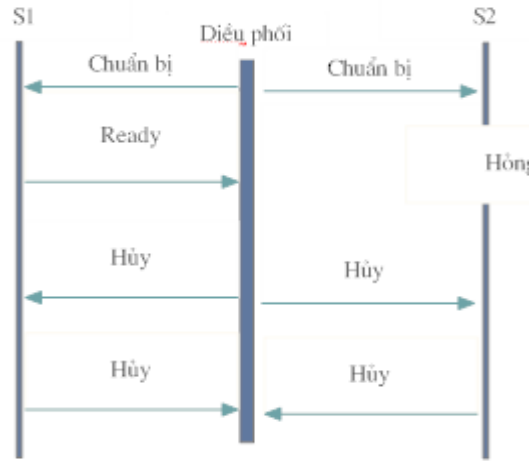
Ta sẽ trình bày một số tình huống có thể xảy ra trong quá trình hợp thức hóa của 1 hệ thống phân tán gồm 1 trạm điều phối, 2 trạm tham gia S1, S2.

Kịch bản 1: Giao dịch bình thường →hợp thức hóa được



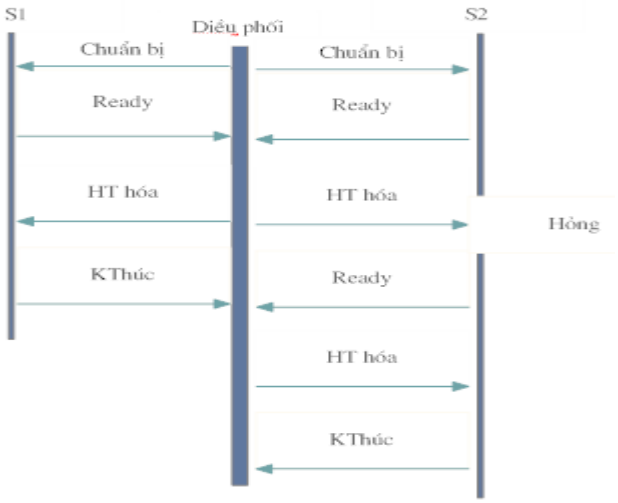
Hình 3.12. Hợp thức hóa bình thường

Kịch bản 2: Hỏng hóc tại S2 trước khi sẵn sàng → huỷ giao dịch



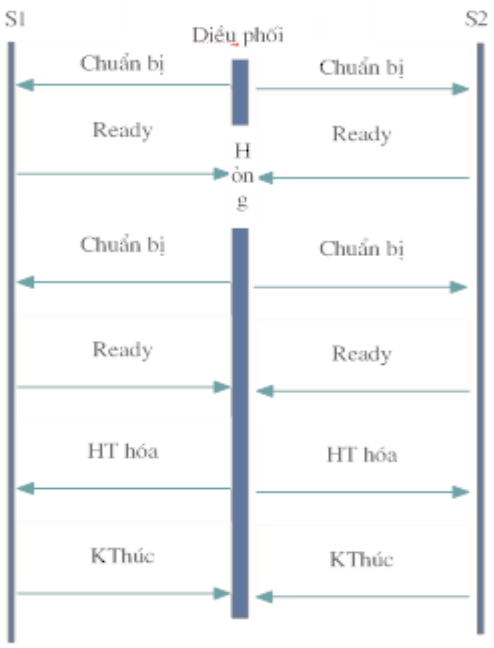
Hình 3.13. Giao dịch bị hủy bỏ

Kịch bản 3: Hỏng hóc của S2 sau khi sẵn sàng và khôi phục được. Hợp thức hoá được sau khi S2 hoạt động



Hình 3.14. Hỏng hóc được khôi phục, hợp thức hóa được

Kịch bản 4: Hỏng hóc tại trạm điều phối khi nhận được tín hiệu sẵn sàng. Quá trình hợp thức hóa sau khi điều phối hoạt động.



Hình 3.14. Hỏng hóc tại trạm điều phối được khôi phục, hợp thức hóa được

Kết luận:

Giao thức hợp thức hoá 2 phase sẽ đảm bảo tính nguyên tố của giao dịch toàn cục chống lại một số kiểu hỏng hóc, trừ hỏng hóc do "thảm họa". Đây là một mô hình hợp thức đơn giản nhất. Với những hệ thống phức tạp và yêu cầu có độ an toàn cao thì ta có giao thức hợp thức hoá 3 phase [5].

BÀI TẬP CHƯƠNG 3

1. Trình bày khái niệm về giao dịch, giao dịch phân tán, tính nguyên tố của giao dịch phân tán.

2. Thế nào là mục dữ liệu. Trình bày các khái niệm về một lịch biểu tuần tự, khả tuần tự và bất khả tuần tự?

3. Trình bày quy tắc điều khiển tương tranh bằng việc đặt khoá (locking): mô hình khóa tổng quát và mô hình phân biệt khoá đọc, khoá ghi. Thế nào là tình trạng khoá chết (dead lock) đối với một tập các giao dịch phân tán, cách khắc phục?

4. Trình bày quy tắc điều khiển tương tranh bằng thời dấu, trong trường hợp tổng quát và trường hợp phân biệt các thao tác đọc / ghi của các giao dịch phân tán.

5. Trình bày giao thức hợp thức hoá 2 phase cho một giao dịch phân tán. Giao thức hợp thức hóa 2 phase cho một giao dịch phân tán nhằm mục đích gì?.

6. Có hai giao dịch T_1 , T_2 truy xuất mục dữ liệu A theo lịch biểu dưới đây. Các thời dấu của các giao dịch và thời dấu đọc/ghi ban đầu của các mục dữ liệu cho trong bảng dưới đây. Các giao dịch đều cộng thêm giá trị vào mục dữ liệu khi thực hiện được thao tác WRITE.

	T_1	T_2	A = 5
Thời dấu	150	160	RT = WT = 0
Thứ tự : (1)	READ A		
(2)		READ A	
(3)	A := A+1		
(4)		A := A+2	
(5)		WRITE A	
(6)	WRITE A		

a/. Hãy xác định thời dấu mà mục dữ liệu A được gán sau mỗi bước (ghi vào cột cuối cùng, sau mỗi bước).

b/. Giả sử giá trị ban đầu của A = 5, sau bước (6) thì giá trị của A bằng bao nhiêu?.

c/. Trong các giao dịch trên, có giao dịch nào bị hủy bỏ không?, tại sao?

d/. Viết 1 lịch biểu tuần tự cho các giao dịch, tính giá trị lưu trữ trong các mục dữ liệu khi kết thúc lịch biểu tuần tự. Lịch biểu đã cho ban đầu có phải lịch biểu khả tuần tự không?

7. Cũng hỏi như bài tập 6, với giả thiết thời dấu ban đầu của giao dịch T1 là 180, còn thời dấu của T2 là 160.

8. Có ba giao dịch T_1 , T_2 , T_3 truy xuất các mục dữ liệu A, B và C theo lịch biểu dưới đây. Các thời dấu của các giao dịch và thời dấu đọc/ghi ban đầu của các mục dữ liệu cho trong bảng. Giả sử các mục dữ liệu có giá trị ban đầu bằng 5, các giao dịch đều cộng thêm 5 vào mục dữ liệu khi thực hiện được thao tác WRITE.

	T_1	T_2	T_3	A = 5	B = 5	C = 5
Thời dấu	40	30	50	RT=WT=0	RT=WT=0	RT=WT=0
Thứ tự : (1)		READ A				
(2)	WRITE A					
(3)			READ C			
(4)		WRITE C				
(5)		WRITE A				
(6)	READ B					
(7)			WRITEB			

a/. Trong lịch biểu trên, thao tác nào không thực hiện đượoc và giao dịch nào bị huỷ bỏ? Tại sao?

b/. Viết 1 lịch biểu tuần tự cho các giao dịch, tính giá trị lưu trữ trong các mục dữ liệu khi kết thúc lịch biểu.

c/. Viết 1 lịch biểu khả tuần tự cho các giao dịch, tính giá trị lưu trữ trong các mục dữ liệu khi kết thúc lịch biểu.

9. Cũng hỏi như bài tập 8, với giả thiết ba giao dịch T_1 , T_2 và T_3 được gán các thời dấu lần lượt là 200, 250 và 300.

10. Có ba giao dịch T_1 , T_2 , T_3 truy xuất các mục dữ liệu A, B và C theo lịch biểu dưới đây. Các thời dấu của các giao dịch và thời dấu đọc/ghi ban đầu của các mục dữ liệu cho trong bảng. Giả sử các mục dữ liệu có giá trị ban đầu bằng 10, các giao dịch đều cộng thêm 5 vào mục dữ liệu khi thực hiện được thao tác WRITE.

Thời dấu	T_1	T_2	T_3	A = 10	B =10	C = 10
	70	50	60	RT = WT= 0	RT = WT= 0	RT = WT= 0
Thứ tự: (1)			READ A			
(2)		WRITE A				
(3)	READ C					
(4)			WRITE C			
(5)			WRITE A			
(6)		READ B				
(7)	WRITE B					

a/. Trong lịch biểu trên, thao tác nào không thực hiện đượoc và giao dịch nào bị huỷ bỏ? Tại sao?

b/. Viết 1 lịch biểu tuần tự cho các giao dịch, tính giá trị lưu trữ trong các mục dữ liệu khi kết thúc lịch biểu.

c/. Viết 1 lịch biểu khả tuần tự cho các giao dịch, tính giá trị lưu trữ trong các mục dữ liệu khi kết thúc lịch biểu.

11. Bài tập trắc nghiệm:

Trong các câu hỏi dưới đây, hãy chọn chỉ một pương án trả lời đúng nhất (tô đầy hình tròn ứng với mỗi phương án đã chọn):

11.1 Quản lý các giao dịch phân tán bằng khóa phân biệt đọc/ghi (RLOCK / WLOCK) nhằm mục đích:

- ○ (A) Xắp xếp các giao dịch theo một lịch biểu khả tuần tự.
- ○ (B) Làm tăng khả năng xử lý song song cho CSDL phân tán.
- ○ (C) Tất cả các câu trả lời A và B đều đúng.
- ○ (D) Tất cả các câu trả lời A và B đều sai.

11.2 Quản lý các giao dịch phân tán bằng thời dấu tổng quát (không phân biệt đọc/ghi) nhằm mục đích:

- ○ (A) Xắp xếp các giao dịch theo một lịch biểu tuần tự.
- ○ (B) Làm tăng khả năng xử lý song song cho hệ thống phân tán.
- ○ (C) Tất cả các câu trả lời A và B đều đúng.
- ○ (D) Tất cả các câu trả lời A và B đều sai.

11.3 Một giao dịch T có thời dấu t_1 truy cập mục dữ liệu A có thời dấu t_2, khi đó:

- ○ (A) T không thể đọc A, nếu t_2 là thời dấu ghi, và $t_2 > t_1$
- ○ (B) T không thể đọc A, nếu t_2 là thời dấu đọc, và $t_2 > t_1$
- ○ (C) Tất cả các câu trả lời A và B đều sai.
- ○ (D) Tất cả các câu trả lời A và B đều đúng.

11.4 Một giao dịch T có thời dấu t_1 truy cập mục dữ liệu A có thời dấu t_2, khi đó:

- ○ (A) T không thể ghi A, nếu t_2 là thời dấu ghi, và $t_2 > t_1$
- ○ (B) T không thể ghi A, nếu t_2 là thời dấu đọc, và $t_2 < t_1$
- ○ (C) Tất cả các câu trả lời A và B đều sai.
- ○ (D) Tất cả các câu trả lời A và B đều đúng.

11.5 Có 3 giao dịch T_1, T_2, T_3 có thời dấu lần lượt là 10, 30, 20. Các giao dịch này tuần tự xuất mục dữ liệu A để thực hiện tập thao tác: {READ A; A:= A+1; WRITE A}. Giả sử mục dữ liệu A có thời dấu đọc/ghi ban đầu là RT= WT= 0. Khi đó:

- ○ (A) Giao dịch T_1 bị huỷ bỏ.
- ○ (B) Giao dịch T_2 bị huỷ bỏ.
- ○ (C) Giao dịch T_3 bị huỷ bỏ.
- ○ (D) Không có giao dịch nào bị huỷ bỏ.

11.6 Có 3 giao dịch T_1, T_2, T_3 cùng truy xuất mục dữ liệu A để thực hiện các thao tác: {LOCK A; READ A; A:= A+1; WRITE A; UNLOCK A}. Giả sử mục dữ liệu A có giá trị bằng 3, sau khi các giao dịch kết thúc, A sẽ có giá trị là:

- ○ (A) A = 4
- ○ (B) A = 5
- ○ (C) A = 6
- ○ (D) Tất cả các câu trả lời trên đều sai

11.7 Có 3 giao dịch T_1, T_2, T_3 có thời dấu lần lượt là 10, 20, 30. Các giao dịch này tuần tự truy xuất mục dữ liệu A để thực hiện tập thao tác: {READ A; A:= A+1; WRITE A}. Giả sử mục dữ liệu A có thời dấu đọc/ghi ban đầu là RT= WT = 40, và có giá trị ban đầu là 2. Khi kết thúc các giao dịch, A sẽ có giá trị:

- ○ (A) A = 3
- ○ (B) A = 4
- ○ (C) A = 5
- ○ (D) Tất cả các câu trả lời trên đều sai.

11.8 Có 3 giao dịch T_1, T_2, T_3 có thời dấu lần lượt là 20, 40, 30. Các giao dịch này tuần tự truy xuất mục dữ liệu A để thực hiện tập thao tác: {READ A; A:= A+1; WRITE A}. Giả sử mục dữ liệu A có thời dấu đọc/ghi ban đầu là RT = WT = 40, và có giá trị ban đầu là 6. Khi kết thúc các giao dịch, A sẽ có giá trị:

- ○ (A) A = 7
- ○ (B) A = 8
- ○ (C) A = 9
- ○ (D) Cả 3 phương án đều sai

11.9 Giao thức hợp thức hoá hai pha trong cập nhật các giao dịch phân tán nhằm mục đích:

- ○ (A) Đảm bảo cho tất cả các giao dịch cục bộ cùng hợp thức hóa hoặc cùng hủy bỏ.
- ○ (B) Đảm bảo tính nguyên tố cho các giao dịch phân tán
- ○ (C) Tất cả các câu trả lời A và B đều sai
- ○ (D) Tất cả các câu trả lời A và B đều đúng.

11.10 Giao thức hợp thức hoá hai pha trong cập nhật các giao dịch phân tán nhằm mục đích gì:

- ○ (A) Đảm bảo cho các giao dịch phân tán luôn luôn được hợp thức thành công
- ○ (B) Đảm bảo tính nguyên tố cho các giao dịch phân tán
- ○ (C) Tất cả các câu trả lời A và B đều sai
- ○ (D) Tất cả các câu trả lời A và B đều đúng.