CHƯƠNG 6:

ĐIỀU KHIỂN TƯƠNG TRANH

CONTENTS

- 1. Một số vấn đề điều khiển đồng thời
- 2. Một số tính chất khi thao tác trên đơn vị dữ liệu
- 3. Lịch tuần tự và lịch khả tuần tự
- 4. Sắp xếp các giao tác bằng nhãn thời gian
- 5. Điều khiển tương tranh bằng cơ chế khóa

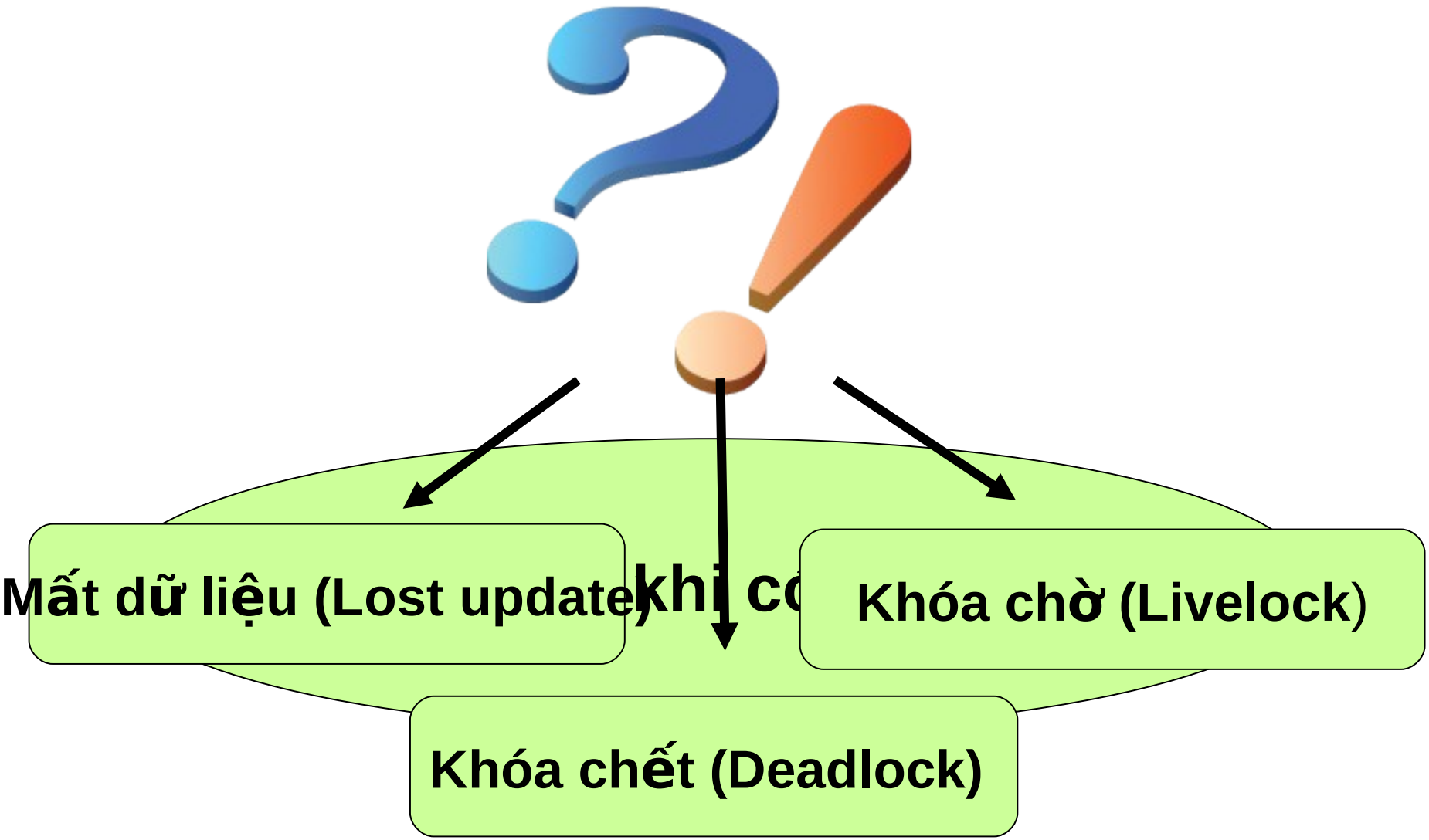
MỤC ĐÍCH

Giới thiệu

- Giao tác là một tập hợp các thao tác có thứ tự truy xuất dữ liệu trên CSDL thành một đơn vị công việc logic (xem là một thao tác nguyên tố), chuyển CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác.

- Nguyên lý điều khiển tương tranh: Là quá trình điều khiển giúp cho nhiều giao tác diễn ra đồng thời mà không xảy ra tranh chấp.

Giới thiệu


Mất dữ liệu (Lost update)
Khóa chờ (Livelock)
Khóa chết (Deadlock)

Giới thiệu

➢Để ghi nhận sự hoàn tất hay không của một thao tác người ta sử dụng các lệnh sau:

BEGIN TRANSACTION	*Bắt đầu giao tác*
COMMIT TRANSACTION	*Kết thúc một giao tác thành công*
ROLLBACK TRANSACTION	*Kết thúc một giao tác không thành công, những thao tác làm ảnh hưởng CSDL trước đó được undo, CSDL được trả về tình trạng trước khi thực hiện giao tác.*
END TRANSACTION	*Chỉ mạng ý nghĩa hình thức, thường không sử dụng*

1. Một số vấn đề điều khiển đồng thời

1. 1. Mất dữ liệu cập nhật (lost update)

- Ví dụ: Cho quan hệ HANGHOA (MaHH, Tên HH, ĐVT, SLTon)
- Giả sử: SLTon =20

Transaction		Gía trị		
T_1 (bán hàng)	T_2 (bán hàng)	x_1	x_2	SLTon
Begin Transaction	Begin Transaction			20
Read(SLTon) $\rightarrow x_1$		**20**		20
	Read(SLTon) $\rightarrow x_2$	20	**20**	20
x_1 - 10 $\rightarrow x_1$		**10**	20	20
	x_2 - 3 $\rightarrow x_2$	10	**17**	20
Write $x_1 \rightarrow$ SLTon		10	17	
	Write $x_2 \rightarrow$ SLTon	10	17	
Commit T_1	Commit T_2			

Lost Update: Thao tác cập nhật của T_1 xem như bị mất, không được ghi nhận (do những thay đổi của các giao tác khác ghi đè lên).

1. Một số vấn đề điều khiển đồng thời

1.2. Đọc dữ liệu chưa commit (uncommitted data)

Ví dụ: Cho quan hệ HANGHOA (MaHH,TenHH, ĐVT, SLTon)

SLton = 20

Transaction		Giá trị		
T_1(nhập hàng)	T_2 (bán hàng)	x_1	x_2	SLton
Begin transaction				20
Read (SLton) → x_1		**20**		20
x_1 + 100 → x_1		**120**		20
Write x_1 → SLton		120		120
	Begin transaction	120		
	Read (SLton) → x_2	120	**120**	
	x_2 – 5 → x_2	120		
	Write x_2 → SLton	120		**115**
	Commit T_2	120		
Rollback T_1				**20**

Transaction T_1 sửa đổi dòng X nhưng chưa commit, transaction T_2 đọc dòng X. Transaction T_1 rollback những gì thay đổi trên dòng X → dữ liệu mà Transaction T_2 đang đọc chưa hề tồn tại.

1. Một số vấn đề điều khiển đồng thời

1.3. Thao tác đọc không thể lập lại (unrepeatable data)

Ví dụ: Xét 2 giao tác sau:

T_1	T_2
Read(A)	
	Read(A)
A = A + 10	Print (A)
Write (A)	Read (A)
	Print (A)

➢Giả sử A = 20 và 2 giao tác này thực hiện đồng thời theo thứ tự sau:

1. Một số vấn đề điều khiển đồng thời

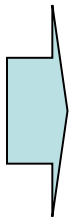
1.3. Thao tác đọc không thể lập lại (unrepeatable data)

Transaction		Giá trị		
T_1	T_2	x_1	x_2	A
Begin transaction				20
Read (A) → x_1		20		20
	Begin transaction			
	Read (A) → x_2	20		20
x_1 +10 → x_1		30		20
	Print (x_2)	30		20
Write x_1 → A		30		
	Read (A) → x_2	30	30	
	Commit T_2			
Commit T_1				

1. Một số vấn đề điều khiển đồng thời

1.4. Vấn đề bóng ma (phantom)

Ví dụ: T_1 thực hiện việc chuyển tiền từ tài khoản A sang tài khoản B. Khi T_1 mới chỉ thực hiện thao tác trừ số tiền ở tài khoản A (*chưa cộng số tiền vào tài khoản B*) thì T_2 muốn xem tổng số tiền ở 2 tài khoản → Tổng số tiền không chính xác.



Transaction	
T_1	T_2
Begin transaction Read (A) A = A – 50 Write (A)	
	Begin transaction Read (A) Read (B) Print (A+B) *Commit* T_2
Read (B) B = B+50 Write (B) *Commit* T_1	

1. Một số vấn đề điều khiển đồng thời

➢ **Như vậy:**

➢ **Một tiêu chuẩn để lập lịch các giao tác là việc thực hiện đồng thời các giao tác cho kết quả như khi thực hiện tuần tự các giao tác.**

➢ **Để giải quyết đụng độ thì phải có "Cơ chế điều khiển tương".**

2. Một số tính chất khi thao tác trên đơn vị dữ liệu

2.1. Hai thao tác tương thích

➢Hai thao tác O_i và O_j (O_i € T_i, O_j € T_j) gọi là tương thích nếu và chỉ nếu kết quả của việc thực hiện đồng thời O_i và O_j giống như kết quả của việc thực hiện tuần tự O_i rồi đến O_j hoặc O_j rồi đến O_i.

	T_1	T_2	
O_{11}	Read A→ a_1 a_1 + 1 → a_1 Print a_1	Read A→ a_2 a_2*2 → a_2 Print a_2	O_{21}
O_{12}	Read A→ a_1 a_1 + 5 → a_1 Write a_1→ A	Read A→ a_2 a_2*2 → a_2 Write a_2→ A	O_{22}
O_{13}	Read A→ a_1 a_1 + 2 → a_1 Write a_1→ A	Read A→ a_2 a_2 + 7 → a_2 Write a_2→ A	O_{23}
O_{14}	Read B→ b_1 b_1 + 1 → b_1		

O_{11} và O_{21} là tương thích

O_{12} và O_{22} không tương thích

O_{13} và O_{23} không tương thích

O_{14} tương thích với các thao tác còn lại

2. Một số tính chất khi thao tác trên đơn vị dữ liệu

2.2. Hai thao tác khả hoán vị

➢ Hai thao tác O_i và O_j (O_i thuộc T_i, O_j thuộc T_j) là khả hoán vị nếu kết quả thực hiện O_i ,O_j hay O_j, O_i là như nhau.

	T_1	T_2	
O_{11}	Read A→ a_1 $a_1 + 1 \rightarrow a_1$ Print a_1	Read A→ a_2 $a_2 * 2 \rightarrow a_2$ Print a_2	O_{21}
O_{12}	Read A→ a_1 $a_1 + 5 \rightarrow a_1$ Write a_1→ A	Read A→ a_2 $a_2 * 2 \rightarrow a_2$ Write a_2→ A	O_{22}
O_{13}	Read A→ a_1 $a_1 + 2 \rightarrow a_1$ Write a_1→ A	Read A→ a_2 $a_2 + 7 \rightarrow a_2$ Write a_2→ A	O_{23}
O_{14}	Read B→ b_1 $b_1 + 1 \rightarrow b_1$ Write b_1→ B		

➢ O_{11} và O_{21} là khả hoán vị

➢ O_{12} và O_{22} không khả hoán vị

➢ O_{13} và O_{23} là khả hoán vị

➢ O_{14} khả hoán vị với các thao tác còn lại

2. Một số tính chất khi thao tác trên đơn vị dữ liệu

Nhận xét:

ao tác truy xuất các đơn vị dữ liệu khác nhau là tương thích và khả

Các thao tác truy xuất trên cùng đơn vị dữ liệu:

- Nếu có liên quan đến phép cộng, trừ thì khả hoán vị
  - Read – Read → khả hoán vị
  - Write – Write → không có tính khả hoán vị
  - Read – Write → không có tính khả hoán vị
  - Write – Read → không có tính khả hoán vị

Chú ý: Hai thao tác không khả hoán vị thì gọi là xung đột

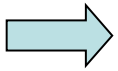
3. Lịch tuần tự và lịch khả tuần tự

3.1. Lịch tuần tự (serial schedule)

➢ Một lịch S được lập từ n giao tác T_1, T_2,….,T_n xử lí đồng thời gọi là lịch tuần tự nếu các thao tác của từng giao tác được thực hiện liên tiếp nhau.

T_i	T_j	T_k
R(x)		
W(x)		
R(y)		
	R(x)	
	W(y)	
		R(y)
		W(x)

Tuần tự



T_i	T_j	T_k
R(x)		
	R(x)	
W(y)		
		R(y)
	W(x)	

Không tuần tự

3. Lịch tuần tự và lịch khả tuần tự

3.2. Lịch khả tuần tự (serializable schedule)

➢ Một lịch S được lập từ n giao tác T_1, T_2,….,T_n xử lí đồng thời gọi là lịch khả tuần tự nếu nó cho cùng kết quả với một lịch tuần tự được lập từ n giao tác trên.

3. Lịch tuần tự và lịch khả tuần tự

3.2. Lịch khả tuần tự (serializable schedule)

➢ Ví dụ 1:

Lịch 1 (A = 1, B = 2)		Lịch 2 tuần tự (A = 1, B = 2)	
T_1	T_2	T_1	T_2
Read A → a_1 $a_1 + 1 \rightarrow a_1$ Write a_1 → A	Read A → a_2 $a_2 * 2 \rightarrow a_2$ Write a_2 → A	Read A → a_1 $a_1 + 1 \rightarrow a_1$ Write a_1 → A	Read A → a_2 $a_2 * 2 \rightarrow a_2$ Write a_2 → A
Read B → b_1 $b_1 + 1 \rightarrow b_1$ Write b_1 → B	Read B → b_2 $b_2 * 2 \rightarrow b_2$ Write b_2 → B	Read B → b_1 $b_1 + 1 \rightarrow b_1$ Write b_1 → B	Read B → b_2 $b_2 * 2 \rightarrow b_2$ Write b_2 → B

Kết quả Lị[illegible] **Lịch 1 có tính khả tuần tự** [illegible]ch 2 (A = 4, B = 6)

3. Lịch tuần tự và lịch khả tuần tự

3.2. Lịch khả tuần tự (serializable schedule)

- Ví dụ 2:

Lịch 1 (A = 1, B = 2)		Lịch 2 (A = 1, B = 2)	
T_1	T_2	T_1	T_2
Read A → a_1 $a_1 + 1 \rightarrow a_1$ Write a_1 → A Read B → b_1 $b_1 + 1 \rightarrow b_1$ Write b_1 → B	Read A → a_2 $a_2 * 2 \rightarrow a_2$ Write a_2 → A Read B → b_2 $b_2 * 2 \rightarrow b_2$ Write b_2 → B	Read A → a_1 $a_1 + 1 \rightarrow a1$ Write a_1 → A Read B → b_1 $b_1 + 1 \rightarrow b_1$ Write b_1 → B	Read A → a_2 $a_2 * 2 \rightarrow a_2$ Write a_2 → A Read B → b_2 $b_2 * 2 \rightarrow b_2$ Write b_2 → B

Kết quả: **Lịch 2 không có tính khả tuần tự** (A = 2, B = 4)

3. Lịch tuần tự và lịch khả tuần tự

Như vậy

➢Tính khả tuần tự của các giao tác là điều kiện đủ để tránh đụng độ trong việc truy xuất đồng thời (Một lịch nếu khả tuần tự thì không đụng độ, nếu không có khả tuần tự thì chưa chắc đụng độ)

➢Bộ lập lịch (schedule): Là một bộ phận của DBMS chịu trách nhiệm lập lịch khả tuần tự từ **n** giao tác xử lí đồng thời, sẽ tiến hành lập lịch các thao tác (thao tác nào sẽ được thực hiện trước, thao tác nào sẽ được thực hiện sau).

4. Sắp xếp các giao tác bằng nhãn thời gian

4.1. Khái niệm nhãn thời gian (timestamp)

➢Là 1 con số được phát sinh bởi bộ lập lịch, được gán cho mỗi giao tác để chỉ định thời điểm bắt đầu thực hiện giao tác. **Nhãn thời gian có tính chất duy nhất và tăng dần** ($T_i < T_j \leftrightarrow t_{Ti} < t_{Tj}$)

➢Nhãn thời gian của đơn vị dữ liệu: là nhãn thời gian của giao tác cuối cùng có truy cập đến đơn vị dữ liệu đó thành công, hay là nhãn thời gian cao nhất trong số các giao tác có truy cập thành công đến đơn vị dữ liệu đó.

T_1	T_2	T_3	t_A
Read A	**Read A**	**Read A**	$t_A = t_{T1}$ $t_A = t_{T2}$ $t_A = t_{T3}$

4. Sắp xếp các giao tác bằng nhãn thời gian

4.2. Thuật toán sắp xếp toàn phần

Procedure Read (T_i, A)

Begin

If $t_A \le t_{Ti}$ then

Thực hiện thao tác đọc

$t_A := t_{Ti}$

If $t_A \le t_{Ti}$ then

Thực hiện thao tác ghi

$t_A := t_{Ti}$

Else

Ghi chú:

- t_A: nhãn thời gian của đơn vị dữ liệu A
- t_{Ti}: nhãn thời gian của giao tác T_i
- Ban đầu $t_A = 0$ khi chưa có giao tác truy cập

Nhận xét

- ***Thuật toán chỉ sắp xếp thứ tự các giao tác (thời gian bắt đầu) không nhằm sắp xếp trình tự thực hiện các thao tác trong mỗi giao tác.***

- Ví dụ 1: $t_{T1} = 100$, $t_{T2} = 120$, $t_A = 0$

T_1	T_2	t_A
Read A		$t_A = 100$
	Read A	$t_A = 120$
A = A + 1		
	A = A + 1	
	Write A	
Write A		$t_A = 120$
		$t_A > t_{T1}$ nên T_1 phải rollback và bắt đầu lại với timestamp mới

4. Sắp xếp các giao tác bằng nhãn thời gian

4.2. Thuật toán sắp xếp toàn phần

➢**Ví dụ 2: $t_{T1} = 100$, $t_{T2} = 120$, $t_A = 0$**

T_1	T_2	t_A
Read A		**$t_A = 100$**
	Read A	**$t_A = 120$**
	Read A	**$t_A = 120$**
Read A		**$t_A > t_{T1}$ nên T_1 phải rollback và bắt đầu lại với timestamp mới**

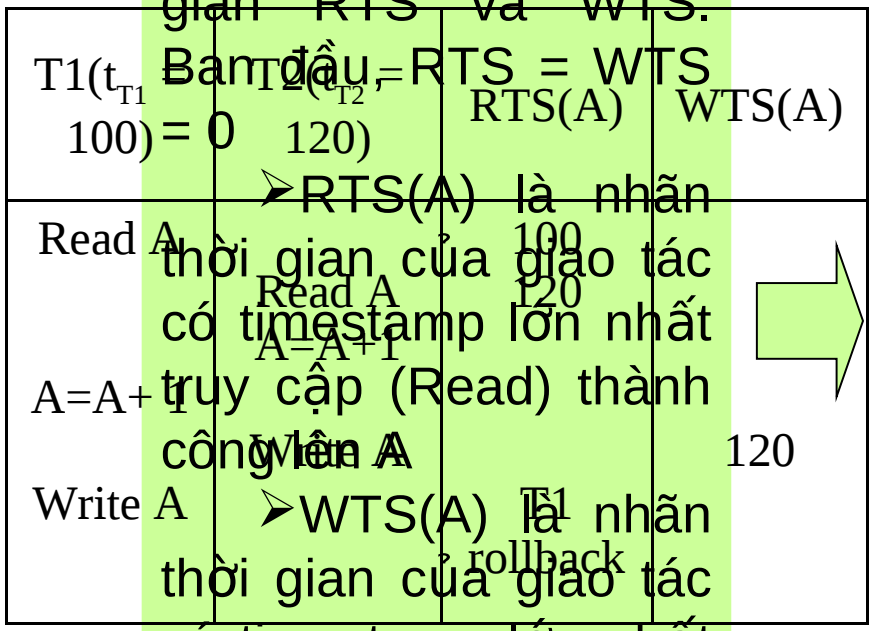
Như vậy: Thuật toán sắp xếp toàn phần : không quan tâm đến tính chất của thao tác dữ liệu (Read/Write) nên chỉ có 1 nhãn thời gian duy nhất cho 1 đơn vị dữ liệu. Nếu quan tâm đến tính chất của thao tác dữ liệu thì cần 2 nhãn thời gian cho 1 đơn vị dữ liệu tương ứng với thao tác đọc và ghi trên đơn vị dữ liệu đó.

➢Trong trường hợp này rõ ràng không xảy ra đụng độ T_1 không cần phải rollback và bắt đầu lại với timestamp mới. Tuy nhiên do thuật toán sắp xếp toàn phần không phân biệt tính chất của thao tác dữ liệu là Read hay Write nên

4. Sắp xếp các giao tác bằng nhãn thời gian

4.3. Thuật toán sắp xếp

- Mỗi đơn vị dữ liệu A có 2 nhãn thời gian RTS và WTS. Ban đầu RTS = WTS = 0
- RTS(A) là nhãn thời gian của giao tác có timestamp lớn nhất truy cập (Read) thành công lên A
- WTS(A) là nhãn thời gian của giao tác có timestamp lớn nhất truy cập (Write) thành công lên A



Procedure Read (T_i, A)

Begin

If WTS(A) <= t_{Ti} then

Thực hiện thao tác đọc

RTS(A) :=Max(RTS(A),t_{Ti})

Else

Rollback T_i và bắt đầu lại với nhãn thời gian mới

Procedure Write (T_i, A)

Begin

If (RTS(A) <= t_{Ti}) and (WTS(A) <= t_{Ti}) then

Thực hiện thao tác ghi

WTS(A) :=t_{Ti}

Else

4. Sắp xếp các giao tác bằng nhãn thời gian

4.3. Thuật toán sắp xếp từng phần

➢Nhận xét : Trong thuật toán sắp xếp từng phần, số lượng giao tác bị rollback lại ít hơn trong thuật toán sắp xếp toàn phần (do nếu 2 giao tác chỉ thực hiện «Đọc» thì không gây ra đụng độ)

T_1	T_2	Nhận xét
(1) Read A (3) Read A	(2) Read A	Thuật toán sắp xếp toàn phần : T_1 bị rollback ở (3) Thuật toán sắp xếp từng phần : không có rollback

T_1	T_2	Nhận xét
(1) Read A (3) Read A	(2) Write A	Thuật toán sắp xếp toàn phần : T_1 bị rollback ở (3) Thuật toán sắp xếp từng phần : T_1 bị rollback ở (3)

5. Điều khiển tương tranh bằng cơ chế khóa

5.1. Khái niệm khóa (lock)

- Phương pháp thông dụng nhất để điều khiển việc truy xuất các đơn vị dữ liệu là sử dụng khóa (lock).

- Lock là một đặc quyền truy xuất (access priveleg) lên các đơn vị dữ liêu của các giao tác mà bộ quản lý khóa có thể trao cho một giao tác hay thu hồi lại. Khi 1 giao tác đã khóa (lock) trên 1 đơn vị dữ liệu nào đó thì các giao tác khác không được phép truy cập đến đơn vị dữ liệu đó cho đến khi nó nhả khóa (unlock).

- Khi một giao tác T thực hiện được việc lock đơn vị dữ liệu A, ta nói, T đang giữ lock A

- Tại mỗi thời điểm, chỉ có một tập con các đơn vị dữ liệu bị khóa, vì vậy bộ quản lý khóa có thể lưu các khóa hiện hành trong một bảng khóa (lock table) với các mẫu tin có dạng sau: (A, L, T) (giao tác T có một khóa kiểu L trên đơn vị dữ liệu A).

5. Điều khiển tương tranh bằng cơ chế khóa

5.2. Kỹ thuật khóa đơn giản

➢Một giao tác khi có yêu cầu truy xuất đến đơn vị dữ liệu thì phải phát ra yêu cầu xin khóa (lock) trên đơn vị dữ liệu đó, nếu yêu cầu này được chấp thuận thì được quyền thao tác và như vậy các giao tác khác sẽ không được phép truy cập đến đơn vị dữ liệu đó cho đến khi giao tác giữ khóa được unlock

Không khóa

T_1	T_2	A (5)
Read A		5
	Read A	5
A=A+1		5
	A=A+1	5
	Write A	6
Write A		6

T_1	T_2	Nhận xét
Lock A		
Read A		
	Lock A	T_2 chờ T_1
	Read A	Unlock
A=A+1		
	A=A+1	T_1 sẽ
	Write A	hoàn tất
Write A		trước khi
Unlock A		T_2 bắt
	Unlock A	đầu. Khi

Kỹ thuật khóa

5. Điều khiển tương tranh bằng cơ chế khóa

5.2. Kỹ thuật khóa đơn giản

➢Ví dụ

T_1	T_2
Lock A Read A A=A+1 Write A Lock B Read B B=B+1 Write B Unlock B Unlock A	 Lock A Read A Unlock A

➢Nhận xét

✓Nếu không phân biệt khóa cho thao tác đọc hay viết thì rõ ràng sẽ có nhiều giao tác phải chờ để được quyền khóa trên 1 đơn vị dữ liệu.
vị dữ liệu nhưng không thay đổi giá trị đó.

✓Vì vậy để giảm bớt tình huống phải chờ khi các giao tác cùng đọc dữ liệu, người ta đề nghị tách yêu cầu khóa thành 2 yêu cầu khóa riêng biệt

5. Điều khiển tương tranh bằng cơ chế khóa

5.3. Kỹ thuật khóa đọc/viết (Readlock/Writelock)

* Khóa để đọc (hay Shared lock): một giao tác T chỉ muốn đọc 1 đơn vị dữ liệu A sẽ thực hiện lệnh RLOCK(A), ngăn không cho bất kì giao tác khác ghi giá trị mới của A trong khi T đã khóa A. Tuy nhiên các giao tác khác vẫn có thể giữ 1 khóa đọc trên A cùng lúc với T.

* Khóa để ghi (Exclusive lock): một giao tác T muốn thay đổi giá trị của 1 đơn vị dữ liệu A đầu tiên sẽ lấy khóa ghi bằng cách thực hiện lệnh WLOCK(A). Khi 1 giao tác đang giữ 1 khóa ghi trên 1 đơn vị dữ liệu, các giao tác khác không thể lấy được khóa đọc hay khóa ghi trên A cùng

- Cả hai khóa đọc và khóa ghi đều được loại bỏ bằng lệnh UNLOCK

5. Điều khiển tương tranh bằng cơ chế khóa

5.3. Kỹ thuật khóa đọc/viết (Readlock/Writelock)

Điều kiện để xin khóa đọc/viết

- Một yêu cầu xin RLOCK(A) chỉ được chấp thuận nếu A chưa bị khóa bởi 1 WLOCK trước đó.

- Một yêu cầu xin WLOCK(A) chỉ được chấp thuận nếu A được tự do.

Bảng tương thích các loại khóa

		Lock đang được giữ	
		R-lock	W-lock
Giao tác yêu cầu lock	R-lock	Yes	No
	W-lock	No	No

5. Điều khiển tương tranh bằng cơ chế khóa

5.3. Kỹ thuật khóa đọc/viết (Readlock/Writelock)

Ví dụ (Với đơn vị dữ liệu B=2)

T_1	T_2	Ghi chú
Rlock B		
Read B → a_1		a_1=B=2
Unlock B		
	Rlock B	
	Read B → a_2	a_2=B=2
	Unlock B	
	a_2 + 1 → a_2	a_2=3
	Wlock B	
	Write a_2 → B	B=a_2=3
	Unlock B	
a_1 + 1 → a_1		a_1=3
Wlock B		
Write a_1 → B		B=a_1=3
Unlock B		

Nhận xét

- Lịch thao tác không khả tuần tự nên xảy ra lost update.
- Việc sử dụng cơ chế lock không đủ để bảo đảm tính khả tuần tự cho
- Để giải quyết tính không khả tuần tự. Dùng chiến lược lock 2 pha, hoặc yêu cầu các giao tác lock các đơn vị dữ liệu theo 1 thứ tự cố định nào đó.

5. Điều khiển tương tranh bằng cơ chế khóa

➢Giả sử khi T_1 giải phóng khóa trên A, khóa này được trao lại cho T_2. Điều gì sẽ xảy ra nếu như trong khi T_2 đang đợi nhận khóa, một giao tác T_3 khác cũng xin một khóa trên A, và T_3 lại được trao khóa này trước T_2. Rồi sau khi T_3 được trao khóa trên A thì lại có 1 giao tác T_4 xin khóa trên A,…Và rất có thể T_2 phải đợi mãi và chẳng bao giờ nhận được khóa trong khi luôn có 1 giao tác khác giữ

Như vậy, ***Livelock là trường hợp 1 giao tác chờ được cấp quyền lock trên 1 đơn vị dữ liệu nào đó mà không xác định được thời điểm được đáp ứng yêu***

Yêu cầu hệ thống khi trao khóa phải ghi nhận tất cả các thỉnh cầu chưa được đáp ứng, và khi đơn vị dữ liệu A được mở khóa thì trao cho các giao tác đã xin đầu tiên trong số những giao tác đang đợi khóa A. Và khi đó bộ lập lịch (schedulers) sẽ tiến hành thực hiện cơ chế: giao tác nào yêu cầu trước thì được đáp ứng trước (FIFO)

5. Điều khiển tương tranh bằng cơ chế khóa

5.4. Các vấn đề trong kỹ thuật khóa

Deadlock

T_1	T_2	Nhận xét
Lock A		- T_1 chờ T_2 unlock B
	Lock B	- T_2 chờ T_1 unlock A

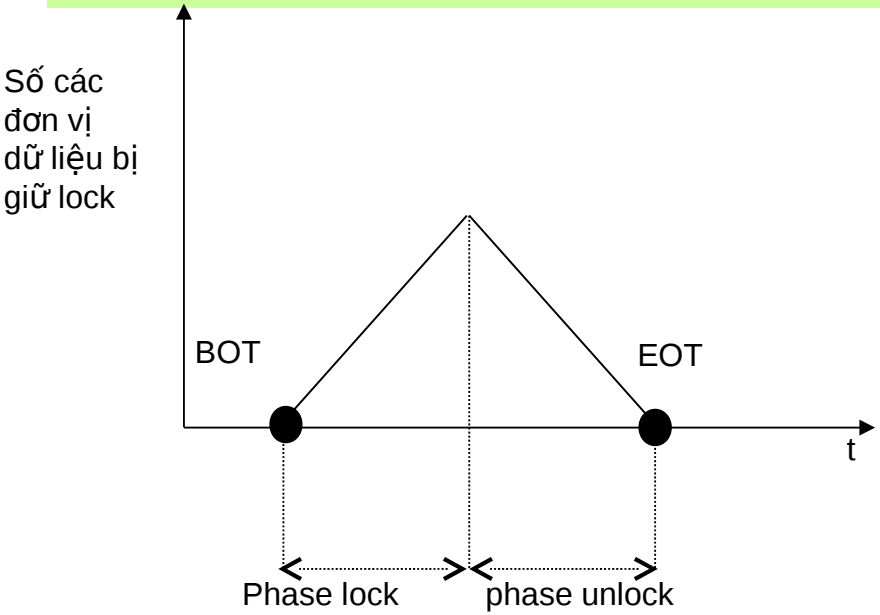
- T_1 yêu cầu và được trao khóa trên A, còn T_2 yêu cầu và được trao khóa trên B. Do đó khi T_1 yêu cầu khóa trên B, nó sẽ phải đợi vì T_2 đã khóa B. Tương tự khi T_2 yêu cầu khóa trên A, nó cũng buộc phải đợi vì T_1 đã khóa A. Kết quả là không một giao tác nào tiếp tục hoạt động được: mỗi giao

Deadlock là tình trạng trong đó những giao tác có liên quan không thể thực hiện tiếp các thao tác của nó mà phải chờ nhau mãi

ỏ qua (Hủy, làm lại) | Ngăn ngừa (Chờ theo chiều nhất định) | Phát hiện (Đồ thị chờ

5. Điều khiển tương tranh bằng cơ chế khóa

➢**Một giao tác thực hiện cơ chế lock 2 phase là một giao tác không thực hiện một lock nào nữa sau khi đã unlock (*thực hiện xong hết tất cả các yêu cầu lock rồi mới thực hiện unlock*).**



Thời gian	T_1	T_2	T_3
		Lock(A)	
		Unlock(A)	
			Lock(A)
			Unlock(A)
	Lock(B)		
	Unlock(B)		
		Lock(B)	

Trong các giao tác trên, T_1 và T_3 là các giao tác 2 pha, còn T_2 thì không

5. Điều khiển tương tranh bằng cơ chế khóa

5.4. Nghi thức lock 2 giai đoạn (2 phase)

Nhận xét

- Một lịch S được lập từ n giao tác thỏa nghi thức
- Sử dụng nghi thức lock 2 giai đoạn chỉ có thể đảm bảo tính khả tuần tự cho lịch thao tác nhưng không thể bảo đảm không xảy ra vấn đề deadlock.

T_1	T_2	Ghi chú
Lock A		
	Lock B	
Lock B		
	Lock A	T_1 phải chờ T_2 unlock B T_2 phải chờ T_1 unlock A → deadlock
Unlock A		
	Unlock B	
Unlock B		
	Unlock A	

HẾT CHƯƠNG 6