

Chương 4

ĐIỀU KHIỂN ĐỒNG THỜI

THỜI LƯỢNG: 12 TIẾT

NỘI DUNG CHI TIẾT

- Các vấn đề trong truy xuất đồng thời
 - Mất dữ liệu đã cập nhật (lost updated)
 - Không thể đọc lại (unrepeatable read)
 - “Bóng ma” (phantom)
 - Đọc dữ liệu chưa chính xác (dirty read)
- Kỹ thuật khóa (locking)
 - Giới thiệu
 - Khóa 2 giai đoạn (two-phase)
 - Khóa đọc viết
 - Khóa đa hạt (multiple granularity)
 - Nghi thức cây (tree protocol)

NỘI DUNG CHI TIẾT (TT)

- **Kỹ thuật nhãn thời gian (timestamps)**
 - Giới thiệu
 - Nhãn thời gian toàn phần
 - Nhãn thời gian riêng phần
 - Nhãn thời gian nhiều phiên bản (multiversion)
- **Kỹ thuật xác nhận hợp lệ (validation)**

VẤN ĐỀ MẤT DỮ LIỆU ĐÃ CẬP NHẬT (LOST UPDATED)

- Xét 2 giao tác

T_1	T_2
Read(A)	Read(A)
$A:=A+10$	$A:=A+20$
Write(A)	Write(A)

- Giả sử T_1 và T_2 được thực hiện đồng thời

A=50	T_1	T_2
t_1	Read(A)	
t_2		Read(A)
t_3	$A:=A+10$	
t_4	Write(A)	
t_5		$A:=A+20$
t_6		Write(A)

A=60

A=70

➤ Dữ liệu đã cập nhật tại t_4 của T_1 bị mất vì đã bị ghi chồng lên ở thời điểm t_6

VẤN ĐỀ KHÔNG THỂ ĐỌC LẠI (UNREPEATABLE READ)

○ Xét 2 giao tác

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

○ Giả sử T_1 và T_2 được thực hiện đồng thời

A=50	T_1	T_2	
t_1	Read(A)		
t_2		Read(A)	A=50
t_3	$A:=A+10$		
t_4		Print(A)	A=50
t_5	Write(A)		
t_6		Read(A)	A=60
t_7		Print(A)	A=60

➤ T_2 tiến hành đọc A hai lần thì cho hai kết quả khác nhau

VẤN ĐỀ “BÓNG MA” (PHANTOM)

- Xét 2 giao tác T_1 và T_2 được xử lý đồng thời
 - A và B là 2 tài khoản
 - T_1 rút 1 số tiền ở tài khoản A rồi đưa vào tài khoản B
 - T_2 kiểm tra đã nhận đủ tiền hay chưa?

A=70, B=50	T_1	T_2	
t_1	Read(A)		A=70
t_2	A:=A-50		
t_3	Write(A)		A=20
t_4		Read(A)	A=20
t_5		Read(B)	B=50
t_6		Print(A+B)	A+B=70
t_7	Read(B)		
t_8	B:=B+50		
t_9	Write(B)		

mất 50 ???

VẤN ĐỀ ĐỌC DỮ LIỆU CHƯA CHÍNH XÁC (DIRTY READ)

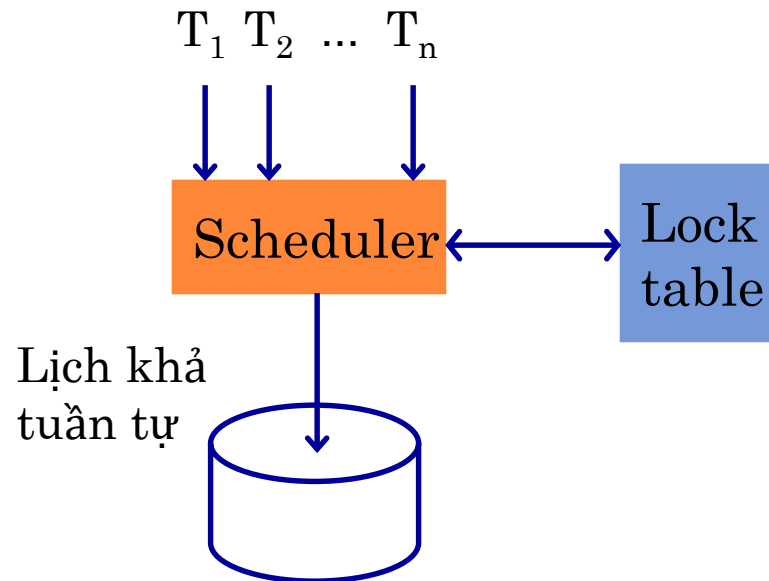
- Xét 2 giao tác T_1 và T_2 được xử lý đồng thời

	T_1	T_2
t_1	Read(A)	
t_2	$A:=A+10$	
t_3	Write(A)	
t_4		Read(A)
t_5		Print(A)
t_6	Abort	

- T_2 đã đọc dữ liệu được ghi bởi T_1 nhưng sau đó T_1 yêu cầu hủy việc ghi

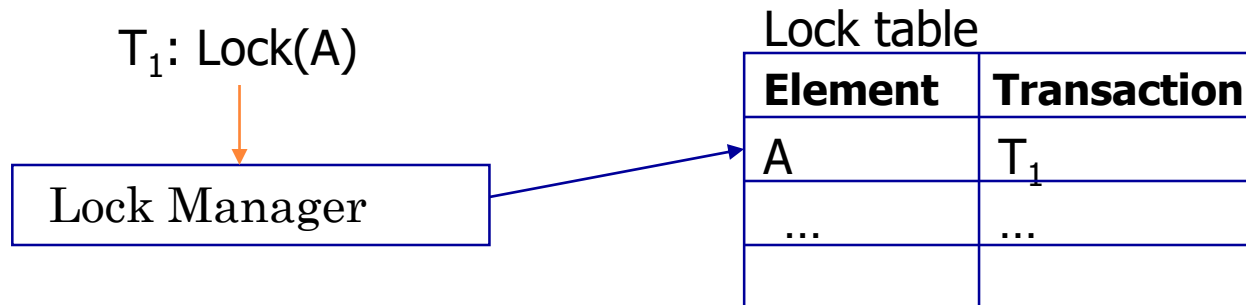
GIỚI THIỆU

- Làm thế nào để bộ lập lịch ép buộc 1 lịch phải khả tuần tự?
- Bộ lập lịch với cơ chế khóa (locking scheduler)
 - Có thêm 2 hành động
 - Lock
 - Unlock



KỸ THUẬT KHÓA

- Các giao tác trước khi muốn đọc/viết lên 1 đơn vị dữ liệu phải phát ra 1 yêu cầu xin khóa (lock) đơn vị dữ liệu đó
 - Lock(A) hay l(A)
- Yêu cầu này được bộ phận quản lý khóa xử lý
 - Nếu yêu cầu được chấp thuận thì giao tác mới được phép đọc/ghi lên đơn vị dữ liệu



- Sau khi thao tác xong thì giao tác phải phát ra lệnh giải phóng đơn vị dữ liệu (unlock)
 - Unlock(A) hay u(A)

KỸ THUẬT KHÓA (TT)

○ Quy tắc

- (1) Giao tác đúng đắn

$T_i : \quad \dots l(A) \dots r(A) / w(A) \dots u(A) \dots$

- (2) Lịch thao tác hợp lệ

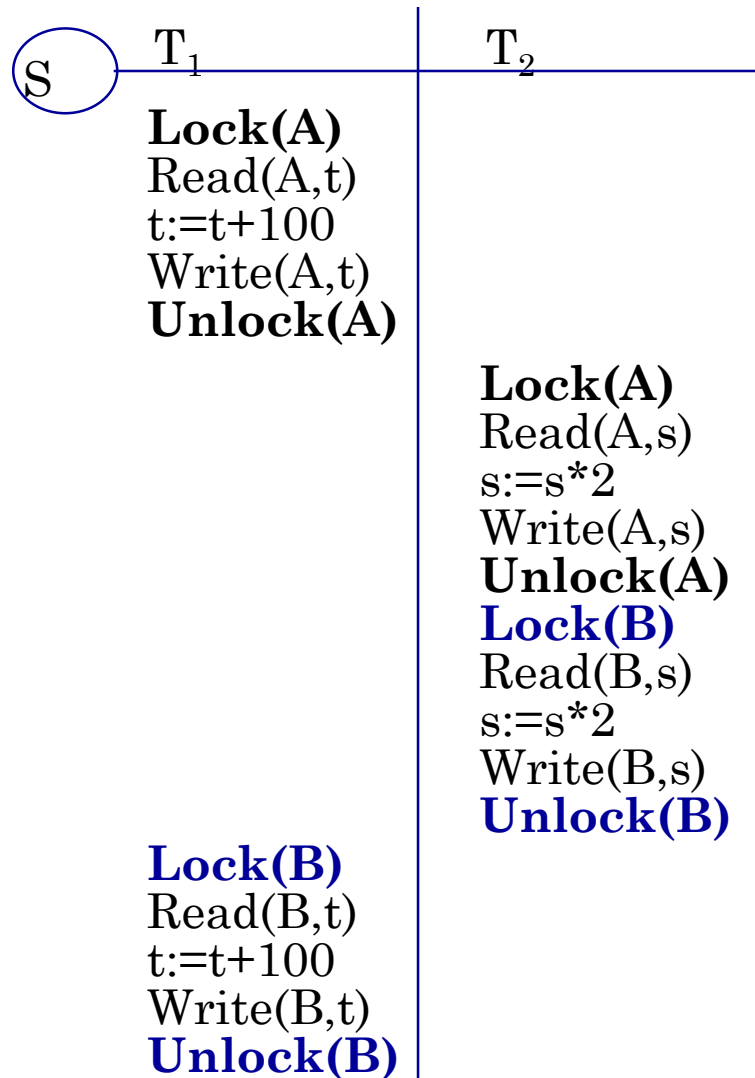
$S : \quad \dots l_i(A) \dots\dots\dots u_i(A) \dots$



không có $l_j(A)$

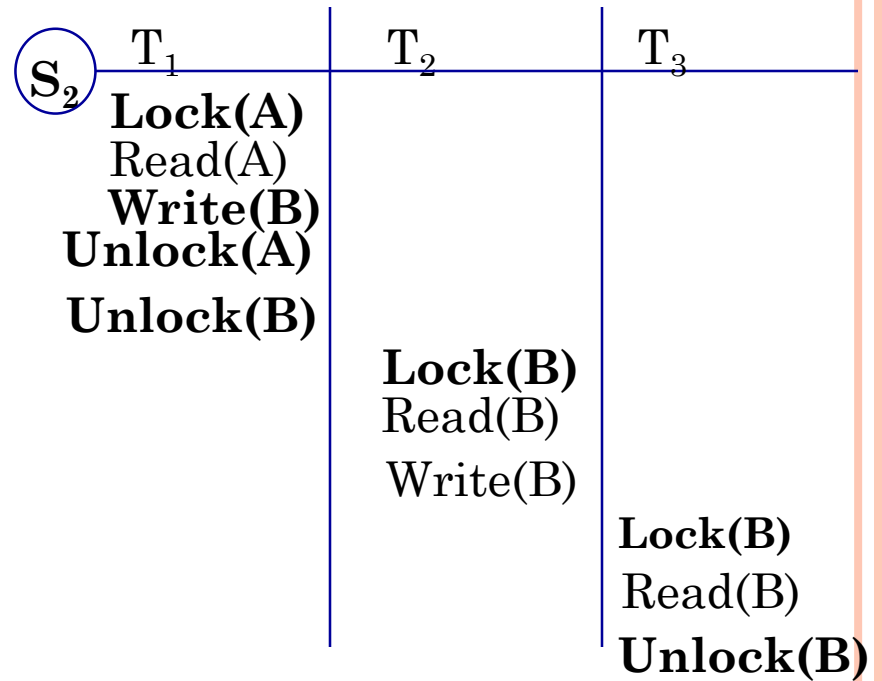
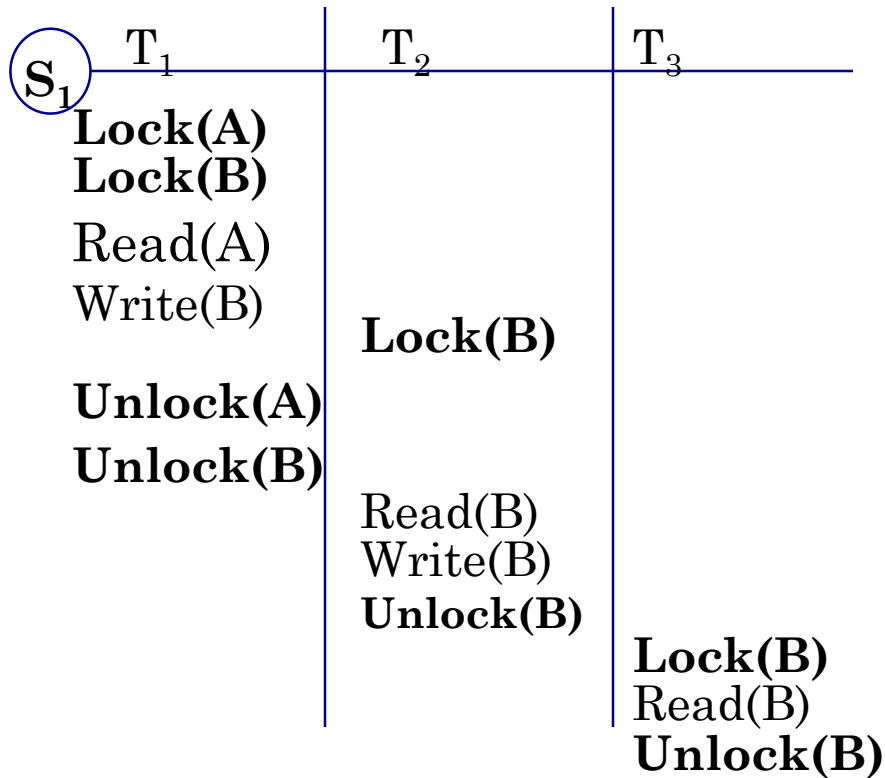


VÍ DỤ



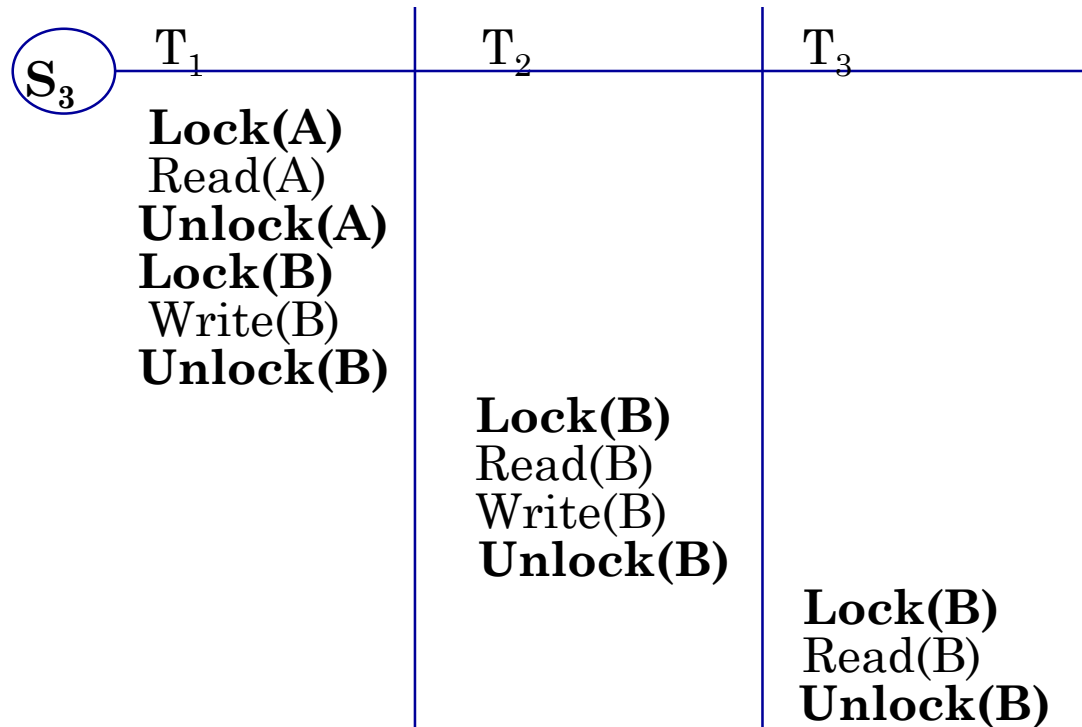
BÀI TẬP

- Cho biết lịch nào hợp lệ? Giao tác nào là đúng?



BÀI TẬP (TT)

- Cho biết lịch nào hợp lệ? Giao tác nào là đúng?



KỸ THUẬT KHÓA (TT)

- Nếu lịch S hợp lệ thì S có khả tuần tự không?

S	T ₁	T ₂	A	B
	Lock(A); Read(A,t) t:=t+100 Write(A,t); Unlock(A)	Lock(A); Read(A,s) s:=s*2 Write(A,s); Unlock(A) Lock(B); Read(B,s) s:=s*2 Write(B,s); Unlock(B)	25 125 250	25 50 150
	Lock(B); Read(B,t) t:=t+100 Write(B,t); Unlock(B)			

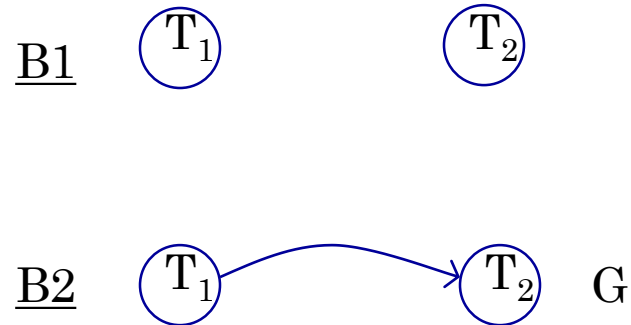
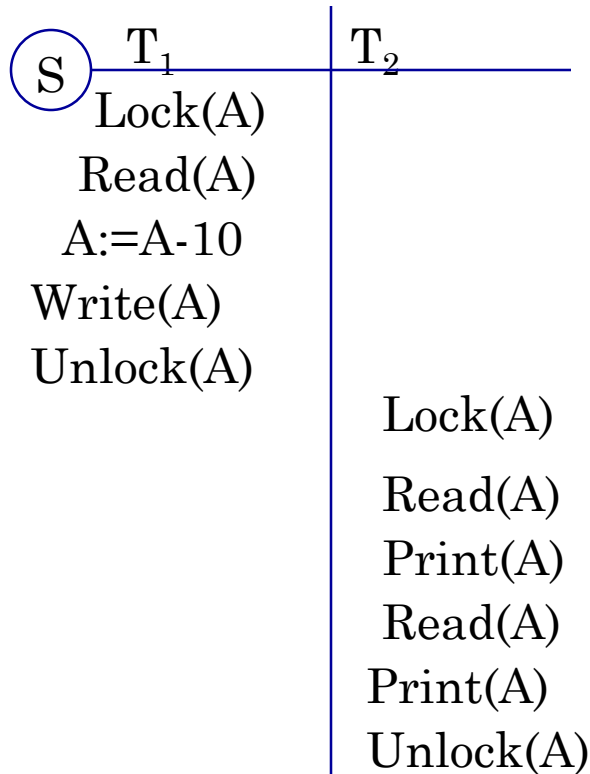


KỸ THUẬT KHÓA (TT)

- Kiểm tra tính khả tuần tự
 - Input : Lịch S được lập từ n giao tác xử lý đồng thời T_1, T_2, \dots, T_n theo kỹ thuật khóa đơn giản
 - Output : S khả tuần tự hay không?
- Xây dựng 1 đồ thị có hướng G
 - Mỗi giao tác T_i là 1 đỉnh của đồ thị
 - Nếu một giao tác T_j phát ra **Lock(A)** sau một giao tác T_i phát ra **Unlock(A)** thì sẽ vẽ cung từ T_i đến T_j , $i \neq j$
 - S khả tuần tự nếu G không có chu trình



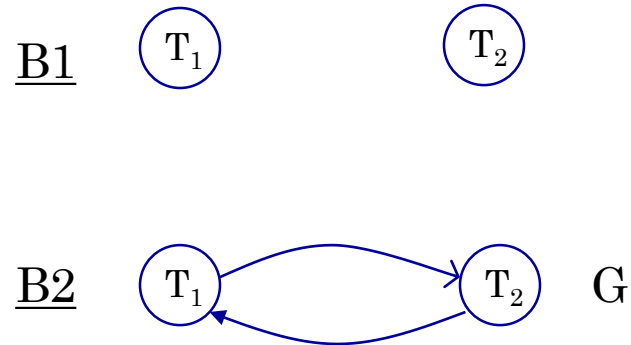
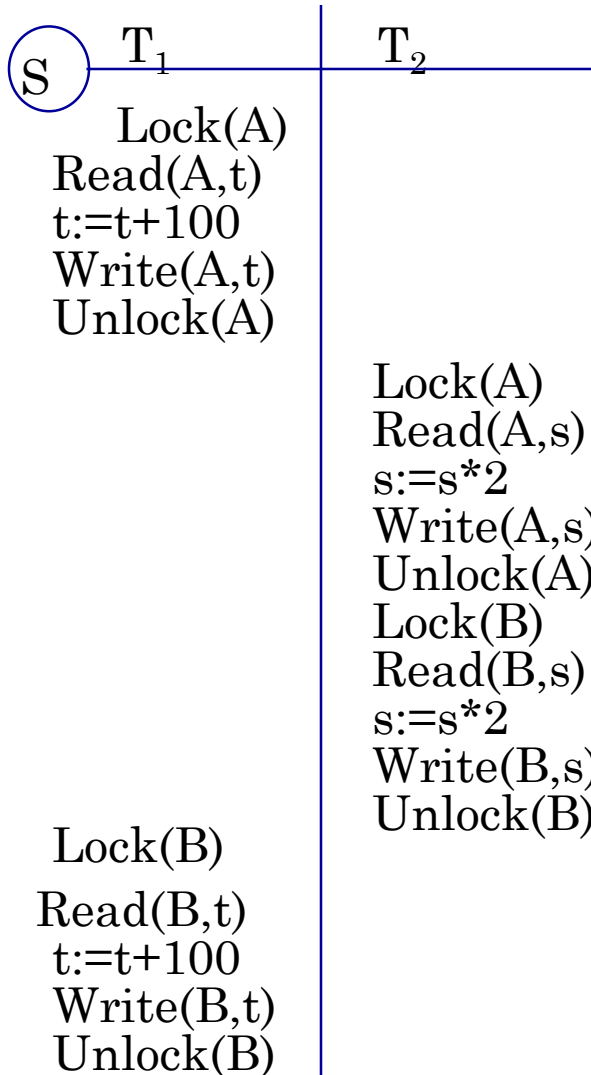
KỸ THUẬT KHÓA (TT)



B3 G không có chu trình \Rightarrow S khả
tuần tự
theo thứ tự T_1 thực hiện trước
rồi tới T_2



KỸ THUẬT KHÓA (TT)



B3 G có chu trình => S không khả tuần tự theo thứ tự T₁ thực hiện trước rồi tới T₂



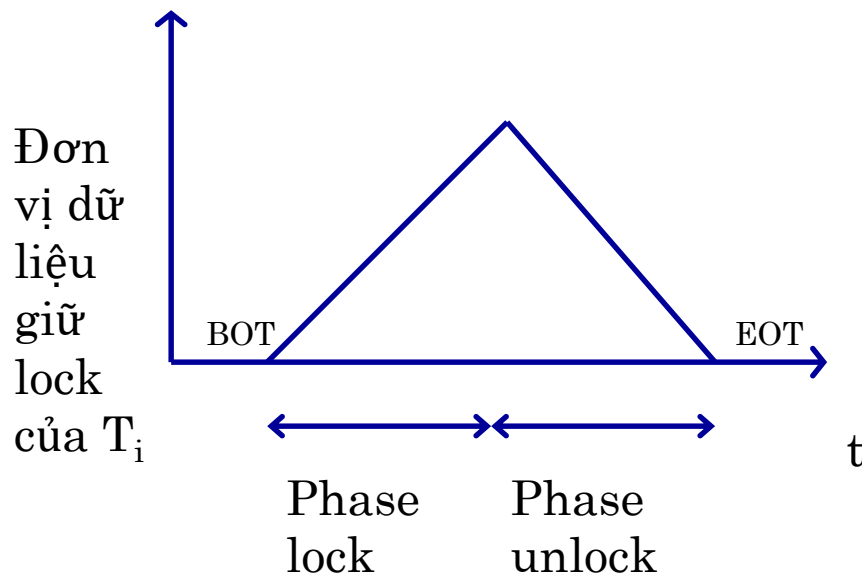
KỸ THUẬT KHÓA 2 GIAI ĐOẠN (2PL: PHASE LOCK)

- **Qui tắc** Một giao tác thỏa nghi thức khóa 2PL nếu kể từ 1 unlock đầu tiên của giao tác thì sẽ không có yêu cầu xin lock nào khác được phát ra (bất chấp đơn vị dữ liệu)
 - **(3) Giao tác 2PL**

S : $l_i(A)$ $u_i(A)$...

←
không có unlock

→
không có lock



Thực hiện xong hết tất cả các yêu cầu lock rồi mới tiến hành unlock



KỸ THUẬT KHÓA 2 GIAI ĐOẠN (TT)

T ₁
Lock(A)
Read(A)
Lock(B)
Read(B)
B:=B+A
Write(B)
Unlock(A)
Unlock(B)

T ₂
Lock(B)
Read(B)
Lock(A)
Read(A)
Unlock(B)
A:=A+B
Write(A)
Unlock(A)

T ₃
Lock(B)
Read(B)
B=B-50
Write(B)
Unlock(B)
Lock(A)
Read(A)
A=A+50
Write(A)
Unlock(A)

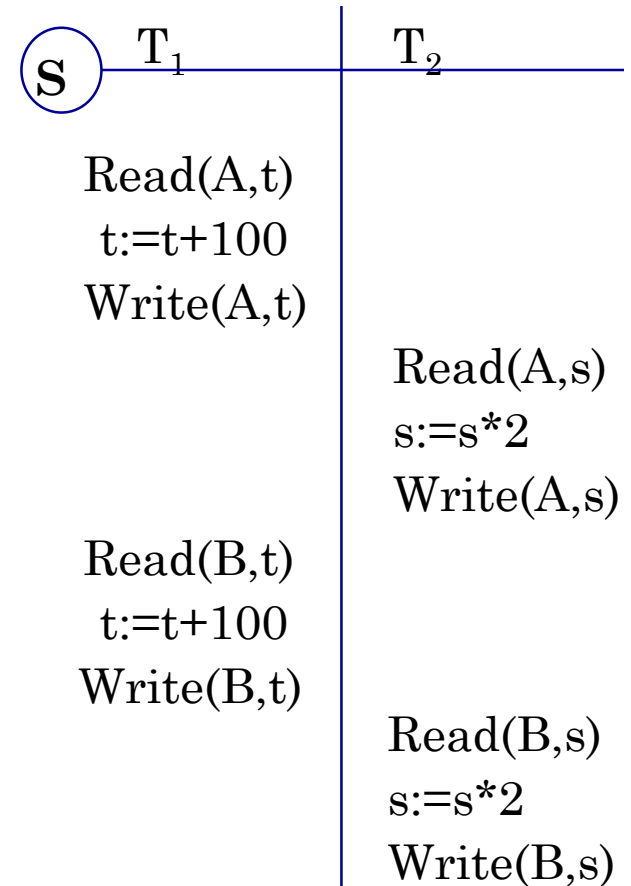
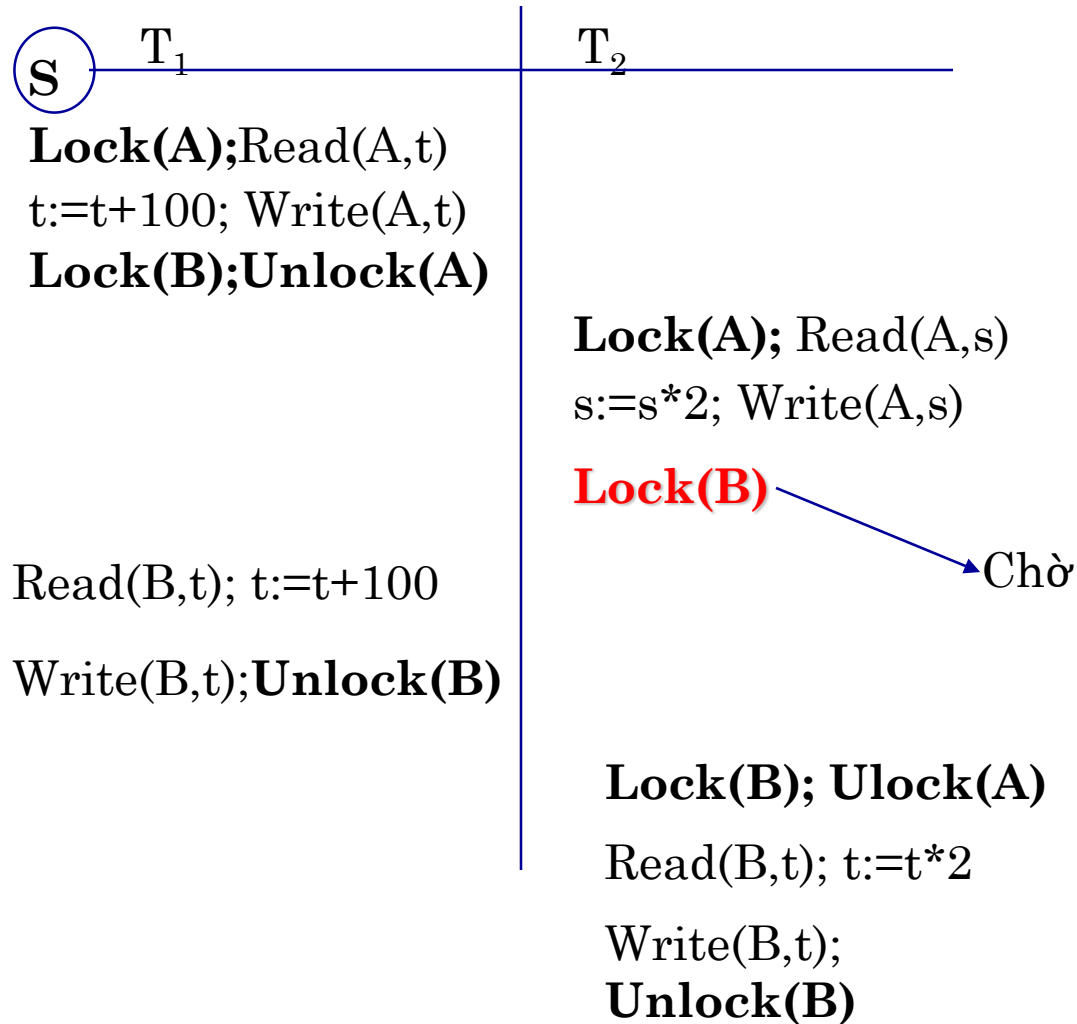
T ₄
Lock(A)
Read(A)
Unlock(A)
Lock(B)
Read(B)
Unlock(B)
Printn(A+B)

*Thỏa nghi thức khóa
2 giai đoạn*

*Không thỏa nghi thức
khóa 2 giai đoạn*



KỸ THUẬT KHÓA 2 GIAI ĐOẠN (TT)



KỸ THUẬT KHÓA 2 GIAI ĐOẠN (TT)

○ Định lý

S thỏa qui tắc (1), (2), (3) \Rightarrow S conflict-serializable

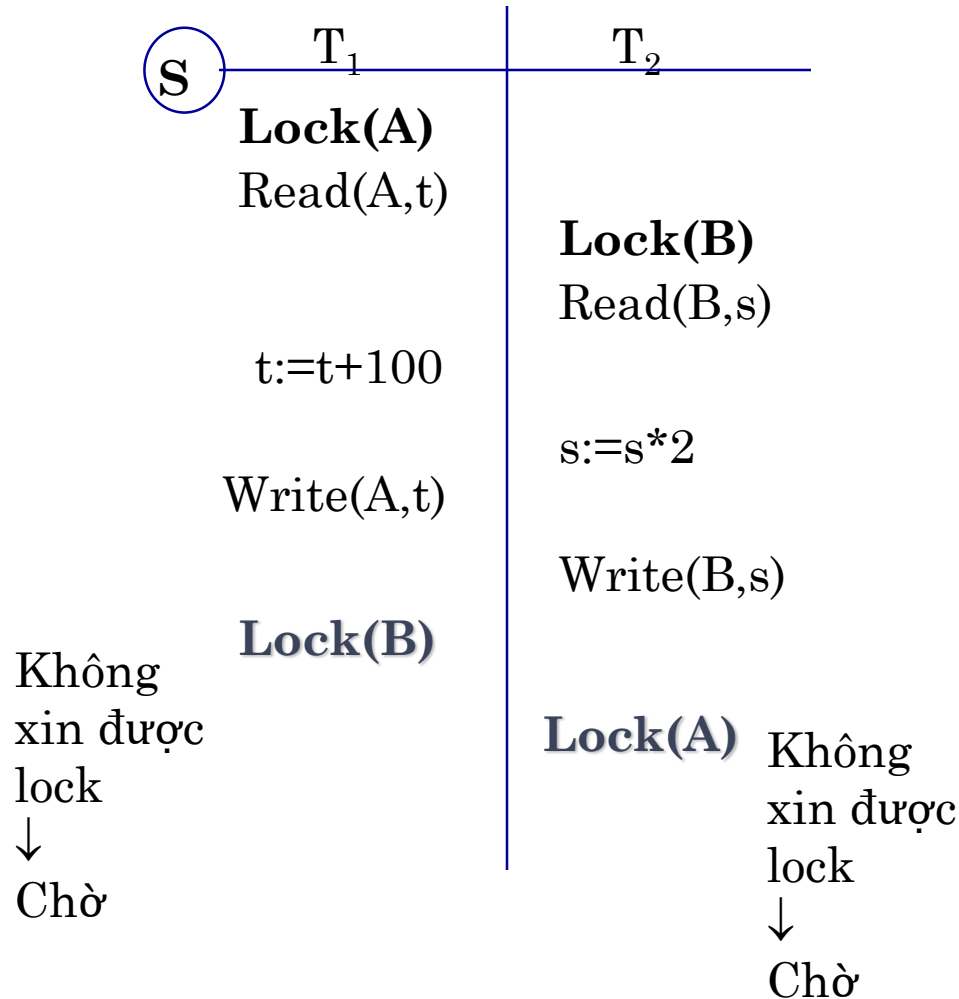
○ Chứng minh

- Giả sử $G(S)$ có chu trình
- $T_1 \rightarrow T_2 \rightarrow \dots T_n \rightarrow T_1$
- T_1 thực hiện lock những đơn vị dữ liệu được unlock bởi T_n
- T_1 có dạng ... lock ... unlock ... lock
- Điều này vô lý vì T_1 là giao tác thỏa 2PL
- $G(S)$ không thể có chu trình
- S conflict-serializable



KỸ THUẬT KHÓA 2 GIAI ĐOẠN (TT)

○ Chú ý



KỸ THUẬT KHÓA ĐỌC VIẾT

○ Vấn đề

T_i	T_j
Lock(A) Read(A) Unlock(A)	Lock(A) Read(A) Unlock(A)

○ Bộ lập lịch có các hành động

- Khóa đọc (Read lock, Shared lock)
 - RLock(A) hay rl(A)
- Khóa ghi (Write lock, Exclusive lock)
 - WLock(A) hay wl(A)
- Giải phóng khóa
 - Unlock(A) hay u(A)




KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

- Cho 1 đơn vị dữ liệu A bất kỳ
 - WLock(A)
 - Hoặc có 1 khóa ghi duy nhất lên A
 - Hoặc không có khóa ghi nào lên A
 - RLock(A)
 - Có thể có nhiều khóa đọc được thiết lập lên A



KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

- Giao tác muốn Write(A)
 - Yêu cầu WLock(A)
 - WLock(A) sẽ được chấp thuận nếu A tự do
 - Sẽ không có giao tác nào nhận được WLock(A) hay RLock(A)
 - Giao tác muốn Read(A)
 - Yêu cầu RLock(A) hoặc WLock(A)
 - RLock(A) sẽ được chấp thuận nếu A không đang giữ một WLock nào
 - Không ngăn chặn các thao tác khác cùng xin Rlock(A)
 - Các giao tác không cần phải chờ nhau khi đọc A
 - Sau khi thao tác xong thì giao tác phải giải phóng khóa trên đơn vị dữ liệu A
 - ULock(A)
- 

KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

○ Qui tắc

- (1) Giao tác đúng đắn

$T_i : \quad \dots rl(A) \dots r(A) \dots u(A) \dots$

$T_i : \quad \dots wl(A) \dots w(A) \dots u(A) \dots$



KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

○ Vấn đề

- Các giao tác đọc và ghi trên cùng 1 đơn vị dữ liệu

T_1

Read(B)

Write(B)?

○ Giải quyết

- Cách 1 - yêu cầu khóa độc quyền

T_i : ... wl(A) ... r(A) ... w(A) ... u(A) ...

- Cách 2 - nâng cấp khóa

T_i : ... rl(A) ... r(A) ... wl(A) ... w(A) ... u(A) ...



BÀI TẬP

- **Hãy suy nghĩ và cho biết cách nào là hợp lý**
 - **Xin khóa thứ 2 cho đơn vị dữ liệu muốn ghi?**
 - **Xin khóa độc quyền ngay từ đầu?**
- **Cho ví dụ và giải thích**




KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

○ Qui tắc

- **(2) - Lịch thao tác hợp lệ**

S : ... rl_i(A) u_i(A) ...
 ↔
 không có wl_j(A)

S : ... wl_i(A) u_i(A) ...



không có wl_j(A)

không có rl_j(A)



KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

- Ma trận tương thích (compatibility matrices)

		Yêu cầu lock	
		Share	eXclusive
Trạng thái hiện hành	Share	yes	no
	eXclusive	no	no



KỸ THUẬT KHÓA ĐỌC VIẾT (TT)


○ Qui tắc

- **(3) - Giao tác 2PL**

- Ngoại trừ trường hợp nâng cấp khóa, các trường hợp còn lại đều giống với nghi thức khóa
- Nâng cấp xin nhiều khóa hơn

[illegible]

- ## • Nâng cấp giải phóng khóa đọc

S : ... $rl_i(A)$... **$ul_i(A)$** ... $wl_i(A)$ $u_i(A)$...

 vẫn chấp nhận trong pha lock



KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

○ Định lý

- S thỏa qui tắc (1), (2), (3) \Rightarrow S conflict-serializable của khóa đọc viết

○ Chứng minh

- Bài tập về nhà



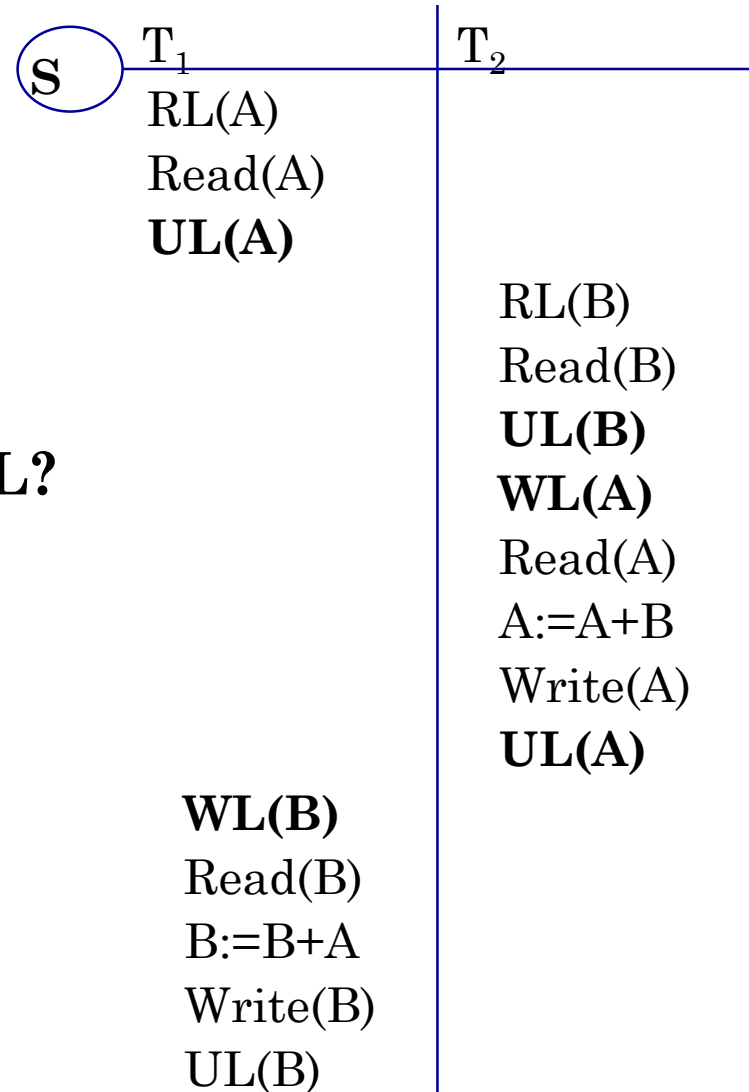
KỸ THUẬT KHÓA ĐỌC VIẾT (TT)

- Kiểm tra tính khả tuần tự
 - Input : Lịch S được lập từ n giao tác xử lý đồng thời T_1, T_2, \dots, T_n theo kỹ thuật khóa đọc viết
 - Output : S khả tuần tự hay không?
- Xây dựng 1 đồ thị có hướng G
 - Mỗi giao tác T_i là 1 đỉnh của đồ
 - Xác định cung như sau:
 - ❖ Nếu T_i thực hiện thao tác **Rlock(A)** hoặc **WLock(A)** và trong giao tác T_j thực hiện sau đó thao tác **Wlock(A)** thì vẽ cung từ $T_i \rightarrow T_j$ (tạm gọi là cung kiểu 1)
 - ❖ Nếu T_i thực hiện thao tác **Wlock(A)** , và trong T_j thực hiện thao tác **Rlock(A)** sau khi T_i thực hiện xong thao tác **Unlock(A)** nhưng trước khi các thao tác khác thực hiện thao tác **Wlock(A)** thì vẽ 1 cung từ $T_i \rightarrow T_j$. (tạm gọi là cung kiểu 2)
- Lịch thao tác khả tuần tự \Leftrightarrow đồ thị không có chu trình.



VÍ DỤ

- S có khả năng tuần tự không?
- Giao tác nào không thỏa 2PL?



VÍ DỤ (TT)

- S có khả năng tuần tự hay không?
- Giao tác nào không thỏa 2PL?

T ₁	T ₂	T ₃	T ₄
	RL(A)		
	WL(B)	RL(A)	
	UL(A)		
		WL(A)	
	UL(B)		
RL(B)		UL(A)	
			RL(B)
RL(A)			UL(B)
WL(C)			
UL(A)			
			WL(B)
			UL(B)
UL(B)			
UL(C)			



NỘI DUNG CHI TIẾT

- Các vấn đề truy xuất đồng thời
- Kỹ thuật khóa (locking)
- **Kỹ thuật nhãn thời gian (timestamps)**
 - Giới thiệu
 - Nhãn thời gian toàn phần
 - Nhãn thời gian riêng phần
 - Nhãn thời gian nhiều phiên bản (multiversion)
- Kỹ thuật xác nhận hợp lệ (validation)

GIỚI THIỆU

○ Ý tưởng

- Giả sử không có hành động nào vi phạm tính khả tuần tự
- Nếu có, hủy giao tác có hành động đó và thực hiện lại giao tác
- Chọn một thứ tự thực hiện nào đó cho các giao tác bằng cách gán nhãn thời gian (TimeStamping)
 - Mỗi giao tác T sẽ có 1 nhãn, ký hiệu TS(T)
 - Tại thời điểm giao tác bắt đầu
 - Thứ tự của các nhãn tăng dần
 - Giao tác bắt đầu trễ thì sẽ có nhãn thời gian lớn hơn các giao tác bắt đầu sớm

GIỚI THIỆU (TT)

- **Để gán nhãn**
 - Đồng hồ của máy tính
 - Bộ đếm (do bộ lập lịch quản lý)
- **Chiến lược cơ bản**
 - Nếu $ST(T_i) < ST(T_j)$ thì lịch thao tác được phát sinh phải tương đương với lịch biểu tuần tự $\{T_i, T_j\}$

NHÃN THỜI GIAN TOÀN PHẦN

- Mỗi giao tác T khi phát sinh sẽ được gán 1 nhãn $TS(T)$ ghi nhận lại thời điểm phát sinh của T
- Mỗi đơn vị dữ liệu X cũng có 1 nhãn thời $TS(X)$, nhãn này ghi lại $TS(T)$ của giao tác T đã thao tác read/write thành công sau cùng lên X
- Khi đến lượt giao tác T thao tác trên dữ liệu X , so sánh $TS(T)$ và $TS(X)$



NHÂN THỜI GIAN TOÀN PHẦN (TT)

Read(T, X)

```
If TS (X) <= TS (T)
    Read (X) ;
    //cho phép đọc X
    TS (X) := TS (T) ;
Else
    Abort {T} ;
    //hủy bỏ T
    //khởi tạo lại TS
```

Write(T, X)

```
If TS (X) <= TS (T)
    Write (X) ;
    //cho phép ghi X
    TS (X) := TS (T) ;
Else
    Abort {T} ;
    //hủy bỏ T
    //khởi tạo lại TS
```



VÍ DỤ

	T_1	T_2	A	B
	$TS(T_1)=100$	$TS(T_2)=200$	$TS(A)=0$	$TS(B)=0$
1	Read(A)		$TS(A)=100$	
2		Read(B)		$TS(B)=200$
	$A=A*2$			
3	Write(A)		$TS(A)=100$	
		$B=B+20$		
4		Write(B)		$TS(B)=200$
5	Read(B)			

↓
Abort

$TS(A) \leq TS(T_1)$: T_1 đọc được A

$TS(B) \leq TS(T_2)$: T_2 đọc được B

$TS(A) \leq TS(T_1)$: T_1 ghi lên A được

$TS(B) \leq TS(T_2)$: T_2 ghi lên B được

$TS(B) > TS(T_1)$: T_1 không đọc được B

Khởi tạo lại $TS(T_1) \rightarrow TS(T_2) < TS(T_1)$
 Lịch khả tuần tự theo thứ tự $\{T_2, T_1\}$



VÍ DỤ

	T_1	T_2	A	B
	$TS(T_1)=100$	$TS(T_2)=200$	$TS(A)=0$	$TS(B)=0$
1	Read(A)		$TS(A)=100$	
2		Read(B)		$TS(B)=200$
	$A=A*2$			
3	Write(A)		$TS(A)=100$	
		$B=B+20$		
4		Write(B)		$TS(B)=200$
5	Read(B)			

↓
Abort

$TS(A) \leq TS(T_1)$: T_1 đọc được A

$TS(B) \leq TS(T_2)$: T_2 đọc được B

$TS(A) \leq TS(T_1)$: T_1 ghi lên A được

$TS(B) \leq TS(T_2)$: T_2 ghi lên B được

$TS(B) > TS(T_1)$: T_1 không đọc được B

Khởi tạo lại $TS(T_1) \rightarrow TS(T_2) < TS(T_1)$
Lịch khả tuần tự theo thứ tự $\{T_2, T_1\}$



VÍ DỤ

	T_1 $TS(T_1)=300$	T_2 $TS(T_2)=200$	A $TS(A)=0$	B $TS(B)=0$
1	Read(A)		$TS(A)=300$	
2		Read(B)		$TS(B)=200$
	$A=A*2$			
3	Write(A)		$TS(A)=300$	
		$B=B+20$		
4		Write(B)		$TS(B)=200$
5	Read(B)			$TS(B)=300$

$TS(A) \leq TS(T_1)$: T_1 đọc được A

$TS(B) \leq TS(T_2)$: T_2 đọc được B

$TS(A) \leq TS(T_1)$: T_1 ghi lên A được

$TS(B) \leq TS(T_2)$: T_2 ghi lên B được

$TS(B) > TS(T_1)$: T_1 đọc được B

Khởi tạo lại $TS(T_1) \rightarrow TS(T_2) < TS(T_1)$
 Lịch khả tuần tự theo thứ tự $\{T_2, T_1\}$



NHÂN THỜI GIAN TOÀN PHẦN (TT)

○ Nhận xét

T_1	T_2	A	T_1	T_2	A
$TS(T_1)=100$	$TS(T_2)=120$	$TS(A)=0$	$TS(T_1)=100$	$TS(T_2)=120$	$TS(A)=0$
Read(A)		$TS(A)=100$	Read(A)		$TS(A)=100$
	Read(A)	$TS(A)=120$		Read(A)	$TS(A)=120$
	Write(A)	$TS(A)=120$		Read(A)	$TS(A)=120$
Write(A)			Read(A)		

T_1 bị hủy và bắt đầu thực hiện lại với timestamp mới

T_1 bị hủy và bắt đầu thực hiện lại với timestamp mới

Không phân biệt tính chất của thao tác là đọc hay viết
→ T_1 vẫn bị hủy và làm lại từ đầu với 1 timestamp mới



NHÃN THỜI GIAN RIÊNG PHẦN (TỪNG PHẦN)

- **Nhãn của đơn vị dữ liệu X được tách ra thành 2 loại:**
 - **RT(X) - read**
 - Ghi nhận TS(T) gần nhất đọc X thành công
 - **WT(X) - write**
 - Ghi nhận TS(T) gần nhất ghi X thành công



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

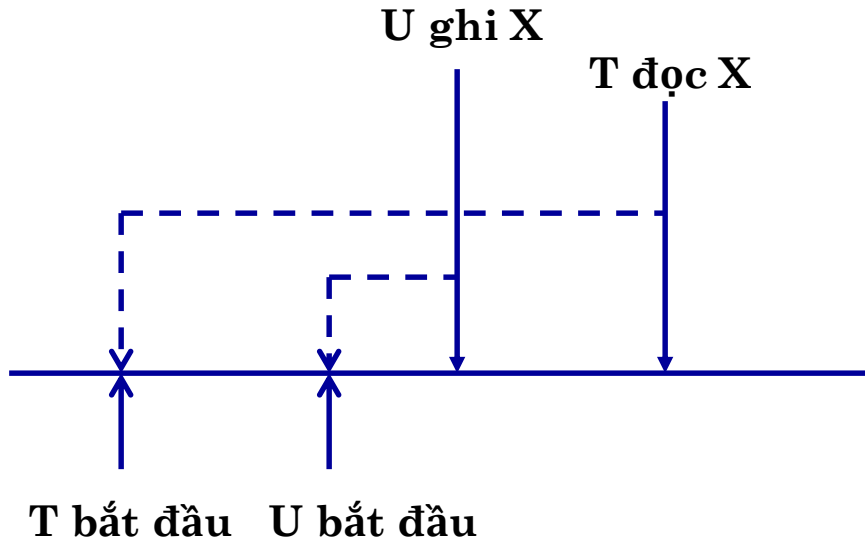
○ Công việc của bộ lập lịch

- Gán nhãn $RT(X)$, $WT(X)$ và $C(X)$
- Kiểm tra thao tác đọc/ghi xuất hiện vào lúc nào
- Có xảy ra tình huống
 - Đọc quá trễ
 - Ghi quá trễ
 - Đọc dữ liệu rác (dirty read)
 - Qui tắc ghi Thomas



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

◦ Đọc quá trễ

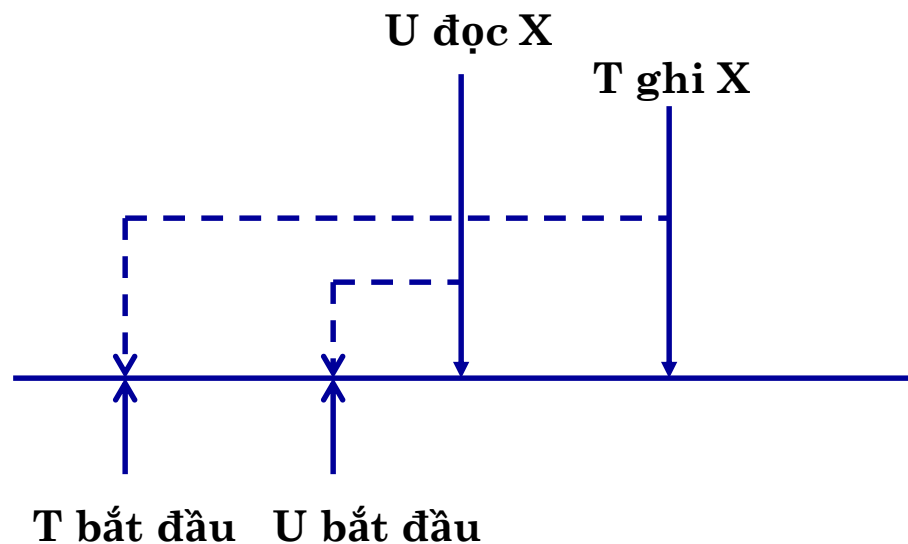


- $ST(T) < ST(U)$
- U ghi X trước, T đọc X sau
 - $ST(T) < WT(X)$
- T không thể đọc X sau U
→ Hủy T



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

○ Ghi quá trễ

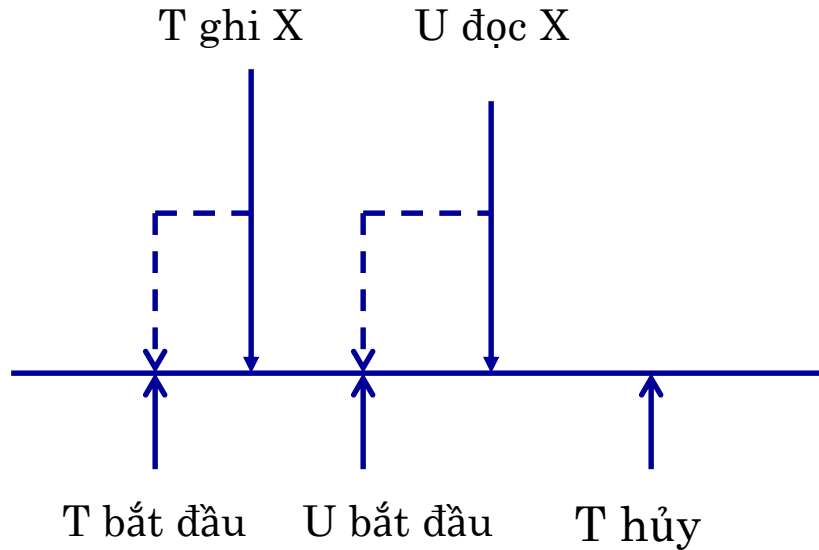


- $ST(T) < ST(U)$
- U đọc X trước, T ghi X sau
 - $WT(X) < ST(T) < RT(X)$
- U phải đọc được giá trị X được ghi bởi T
→ Hủy T



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

◦ Đọc dữ liệu rác

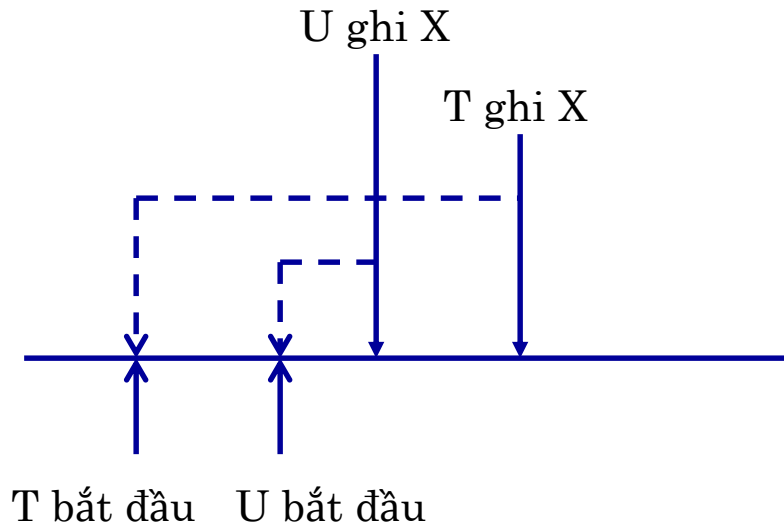


- $ST(T) < ST(U)$
- T ghi X trước, U đọc X sau
 - Thao tác bình thường
- Nhưng T hủy
 - U đọc X sau khi T commit
 - U đọc X sau khi T abort



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

○ Quy tắc ghi Thomas

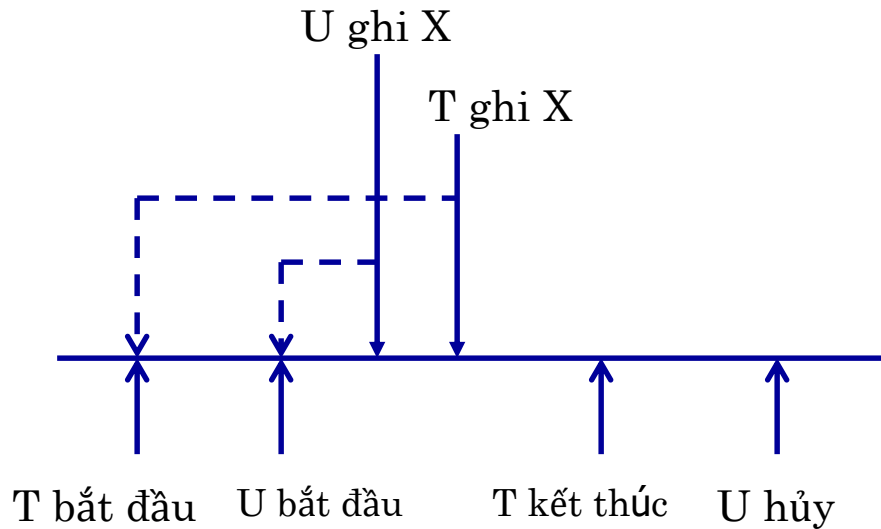


- $ST(T) < ST(U)$
- U ghi X trước, T ghi X sau
 - $ST(T) < WT(X)$
- T ghi X xong thì không làm được gì
 - Không có giao tác nào đọc được giá trị X của T (nếu có cũng bị hủy do đọc quá trễ)
 - Các giao tác đọc sau T và U thì mong muốn đọc giá trị X của U

→ Bỏ qua thao tác ghi của T

NHÂN THỜI GIAN RIÊNG PHẦN (TT)

○ Quy tắc ghi Thomas

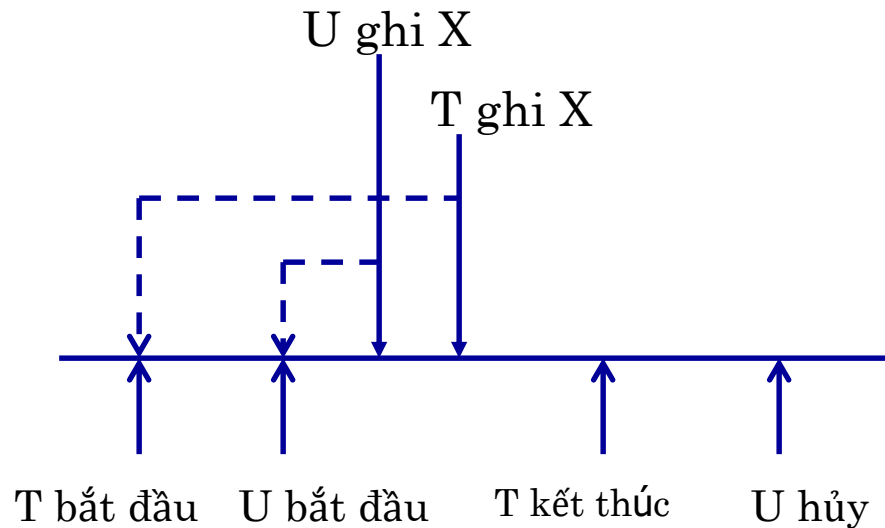


- Nhưng U hủy
 - Giá trị của X được ghi bởi U bị mất
 - Cần khôi phục lại giá trị X trước đó
- Và T đã kết thúc
 - X có thể khôi phục từ thao tác ghi của T
- Do quy tắc ghi Thomas
 - Thao tác ghi đã được bỏ qua
 - Quá trễ để khôi phục X



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

○ Quy tắc ghi Thomas



- Khi T muốn ghi
 - Cho T thử ghi và sẽ gỡ bỏ nếu T hủy
 - Sao lưu giá trị cũ của X và nhận $WT(X)$ trước đó



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

○ Tóm lại

- Khi có yêu cầu đọc và ghi từ giao tác T
- Bộ lập lịch sẽ
 - Đáp ứng yêu cầu
 - Hủy T và khởi tạo lại T với 1 timestamp mới
 - T rollback
 - Trì hoãn T, sau đó mới quyết định phải hủy T hoặc đáp ứng yêu cầu



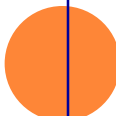
NHÂN THỜI GIAN RIÊNG PHẦN (TT)

Read(T, X)

```
If WT(X) <= TS(T)
    Read(X) ; //cho phép đọc X
    RT(X) := max(RT(X), TS(T)) ;
Else
    Rollback{T} ;
    //hủy T và khởi tạo lại TS mới
```

Write(T, X)

```
If RT(X) <= TS(T)
    If WT(X) <= TS(T)
        Write(X) ; //cho phép ghi X
        WT(X) := TS(T) ;
        //Else không làm gì cả
    Else
        Rollback{T} ;
        //hủy T và khởi tạo lại TS mới
```



VÍ DỤ

	T_1 TS(T_1)=100	T_2 TS(T_2)=200	A RT(A)=0 WT(A)=0	B RT(B)=0 WT(B)=0	C RT(C)=0 WT(C)=0	
1	Read(A)		RT(A)=100 WT(A)=0			WT(A) ≤ TS(T_1) T_1 đọc được A
2		Read(B)		RT(B)=200 WT(B)=0		WT(B) ≤ TS(T_2) T_2 đọc được B
3	Write(A)		RT(A)=100 WT(A)=100			RT(A) ≤ TS(T_1) T_1 ghi lên WT(A) = TS(T_1) A được
4		Write(B)		RT(B)=200 WT(B)=200		RT(B) ≤ TS(T_2) T_2 ghi lên WT(B) = TS(T_2) B được
5		Read(C)			RT(C)=200 WT(C)=0	WT(C) ≤ TS(T_2) T_2 đọc được C
6	Read(C)				RT(C)=200 WT(C)=0	WT(C) ≤ TS(T_1) T_1 đọc được C
7	Write(C)					RT(C) > TS(T_1) T_1 không ghi lên C được

↓
Abort

VÍ DỤ

	T_1 TS(T_1)=300	T_2 TS(T_2)=200	A RT(A)=0 WT(A)=0	B RT(B)=0 WT(B)=0	C RT(C)=0 WT(C)=0	
1	Read(A)		RT(A)=300 WT(A)=0			WT(A) ≤ TS(T_1) T_1 đọc được A
2		Read(B)		RT(B)=200 WT(B)=0		WT(B) ≤ TS(T_2) T_2 đọc được B
3	Write(A)		RT(A)=300 WT(A)=300			RT(A) ≤ TS(T_1) T_1 ghi lên WT(A) = TS(T_1) A được
4		Write(B)		RT(B)=200 WT(B)=200		RT(B) ≤ TS(T_2) T_2 ghi lên WT(B) = TS(T_2) B được
5		Read(C)			RT(C)=200 WT(C)=0	WT(C) ≤ TS(T_2) T_2 đọc được C
6	Read(C)				RT(C)=300 WT(C)=0	WT(C) ≤ TS(T_1) T_1 đọc được C
7	Write(C)				RT(C)=300 WT(C)=300	T_1 ghi lên C được

VÍ DỤ (TT)

T_1 TS=200	T_2 TS=150	T_3 TS=175	A RT=0 WT=0	B RT=0 WT=0	C RT=0 WT=0
Read(B)					
	Read(A)				
		Read(C)			
Write(B)					
Write(A)					
	Write(C)				
		Write(A)			



VÍ DỤ (TT)

T ₁ TS=200	T ₂ TS=150	T ₃ TS=175	A RT=0 WT=0	B RT=0 WT=0	C RT=0 WT=0
Read(B)				RT=200 WT=0	
	Read(A)		RT=150 WT=0		
		Read(C)			RT=175 WT=0
Write(B)				RT=200 WT=200	
Write(A)			RT=150 WT=200		
	Write(C)				
		Write(A)			

Rollback

Giá trị của A đã sao lưu bởi T1
→ T3 không bị rollback và không cần ghi A



VÍ DỤ (TT)

T ₁ TS=150	T ₂ TS=200	T ₃ TS=175	T ₄ TS=255	A RT=0 WT=0
Read(A)				
Write(A)				
	Read(A)			
	Write(A)			
		Read(A)		
			Read(A)	



VÍ DỤ (TT)

T ₁ TS=150	T ₂ TS=200	T ₃ TS=175	T ₄ TS=255	A RT=0 WT=0
Read(A)				RT=150 WT=0
Write(A)				RT=150 WT=150
	Read(A)			RT=200 WT=150
	Write(A)			RT=200 WT=200
		Read(A)		
			Read(A)	RT=255 WT=200

Rollback



NHÂN THỜI GIAN RIÊNG PHẦN (TT)

○ Nhận xét

- Thao tác $\text{read}_3(A)$ làm cho giao tác T_3 bị hủy
- T_3 đọc giá trị của A sẽ được ghi đè trong tương lai bởi T_2
- Giả sử nếu T_3 đọc được giá trị của A do T_1 ghi thì sẽ không bị hủy



NHÂN THỜI GIAN NHIỀU PHIÊN BẢN

○ Ý tưởng

- Cho phép thao tác $\text{read}_3(A)$ thực hiện
- Bên cạnh việc lưu trữ giá trị hiện hành của A, ta giữ lại các giá trị được sao lưu trước kia của A (phiên bản của A)
- Giao tác T sẽ đọc được giá trị của A ở 1 phiên bản thích hợp nào đó



NHÂN THỜI GIAN NHIỀU PHIÊN BẢN (TT)

- Mỗi phiên bản của 1 đơn vị dữ liệu X có
 - $RT(X)$
 - Ghi nhận lại giao tác sau cùng đọc X thành công
 - $WT(X)$
 - Ghi nhận lại giao tác sau cùng ghi X thành công
- Khi giao tác T phát ra yêu cầu thao tác lên X
 - Tìm 1 phiên bản thích hợp của X
 - Đảm bảo tính khả tuần tự
- Một phiên bản mới của X sẽ được tạo khi hành động ghi X thành công



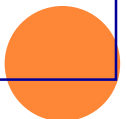
NHÂN THỜI GIAN NHIỀU PHIÊN BẢN (TT)

Read(T, X)

```
i="số thứ tự phiên bản sau cùng nhất của A"  
While  $WT(X_i) > TS(T)$   
     $i := i - 1$ ; //lùi lại  
Read( $X_i$ ) ;  
 $RT(X_i) := \max(RT(X_i), TS(T))$  ;
```

Write(T, X)

```
i="số thứ tự phiên bản sau cùng nhất của A"  
While  $WT(X_i) > TS(T)$   
     $i := i - 1$ ; //lùi lại  
If  $RT(X_i) > TS(T)$   
    Rollback T//Hủy và khởi tạo TS mới  
Else  
    Tạo và chèn thêm phiên bản  $A_{i+1}$  ;  
    Write( $X_{i+1}$ ) ;  
     $RT(X_{i+1}) = 0$ ; //chưa có ai đọc  
     $WT(X_{i+1}) = TS(T)$  ;
```



VÍ DỤ

T₁ TS=150	T₂ TS=200	T₃ TS=175	T₄ TS=255	A₀ RT=0 WT=0	A₁	A₂
Read(A)				RT=150 WT=0		
Write(A)					RT=0 WT=150	
	Read(A)				RT=200 WT=150	
	Write(A)					RT=0 WT=200
		Read(A)			RT=200 WT=150	
			Read(A)			RT=255 WT=200



VÍ DỤ (TT)

T ₁ TS=100	T ₂ TS=200	A ₀ RT=0 WT=0	B₀ RT=0 WT=0	A ₂	B ₁
Read(A)		RT=100 WT=0			
	Write(A)		RT=0 WT=200	RT=0 WT=200	
	Write(B)				RT=0 WT=200
Read(B)				RT=100 WT=0	
Write(A)			RT=0 WT=100		



NHÂN THỜI GIAN NHIỀU PHIÊN BẢN (TT)

○ Nhận xét

- Thao tác đọc
 - Giao tác T chỉ đọc giá trị của phiên bản do T hay những giao tác trước T cập nhật
 - T không đọc giá trị của các phiên bản do các giao tác sau T cập nhật
→ Thao tác đọc không bị rollback
- Thao tác ghi
 - Thực hiện được bằng cách chèn thêm phiên bản mới
 - Không thực hiện được thì rollback
- Tốn nhiều chi phí tìm kiếm, tốn bộ nhớ
- Nên giải phóng các phiên bản quá cũ không còn được các giao tác sử dụng



NỘI DUNG CHI TIẾT

- Các vấn đề truy xuất đồng thời
- Kỹ thuật khóa (locking)
- Kỹ thuật nhãn thời gian (timestamps)
- Kỹ thuật xác nhận hợp lệ (validation)

GIỚI THIỆU

○ Ý tưởng

- Cho phép các giao tác truy xuất dữ liệu một cách tự do
- Kiểm tra tính khả tuần tự của các giao tác
 - Trước khi ghi, tập hợp các đơn vị dữ liệu của 1 giao tác sẽ được so sánh với tập đơn vị dữ liệu của những giao tác khác
 - Nếu không hợp lệ, giao tác phải rollback

○ Trong khi nhận thời gian

- Lưu giữ lại các phiên bản của đơn vị dữ liệu

○ Thì xác nhận tính hợp lệ

- Quan tâm đến các giao tác đang làm gì

XÁC NHẬN HỢP LỆ

○ Một giao tác có 3 giai đoạn

- (1) Đọc - Read set - $RS(T)$
 - Đọc tất cả các đơn vị dữ liệu có trong giao tác
 - Tính toán rồi lưu trữ vào bộ nhớ phụ
 - Không sử dụng cơ chế khóa
- (2) Kiểm tra hợp lệ - Validate
 - Kiểm tra tính khả tuần tự
- (3) Ghi - Write set - $WS(T)$
 - Nếu (2) hợp lệ thì ghi xuống CSDL

○ Chiến lược cơ bản

- Nếu T_1, T_2, \dots, T_n là thứ tự hợp lệ thì kết quả sẽ conflict-serializable với lịch tuần tự $\{T_1, T_2, \dots, T_n\}$

XÁC NHẬN HỢP LỆ (TT)

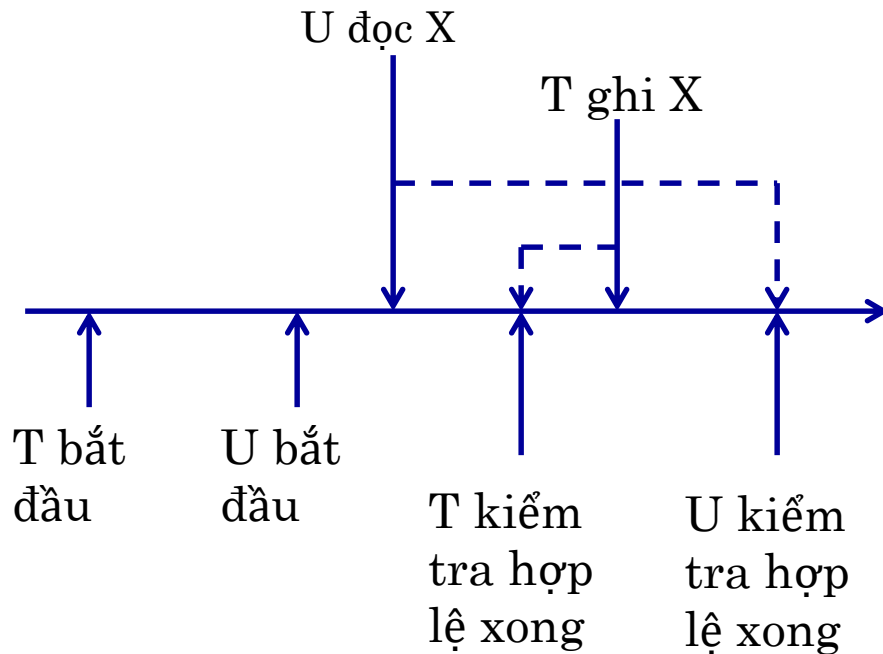
- Bộ lập lịch xem xét 3 tập hợp
 - START
 - Tập các giao tác đã bắt đầu nhưng chưa kiểm tra hợp lệ xong
 - $START(T)$ ghi nhận thời điểm bắt đầu của T
 - VAL
 - Tập các giao tác được kiểm tra hợp lệ nhưng chưa hoàn tất ghi
 - Các giao tác đã hoàn tất giai đoạn 2
 - $VAL(T)$ ghi nhận thời điểm T kiểm tra xong
 - FIN
 - Tập các giao tác đã hoàn tất việc ghi
 - Các giao tác đã hoàn tất giai đoạn 3
 - $FIN(T)$ ghi nhận thời điểm T hoàn tất

VÍ DỤ

T1	T2
Read(B)	
	Read(B)
	B:=B-50
	Read(A)
	A:=A+50
Read(A)	
<i>Xác nhận tính hợp lệ</i>	
Display(A+B)	
	<i>Xác nhận tính hợp lệ</i>
	Write(B)
	Write(A)

XÁC NHẬN HỢP LỆ (TT)

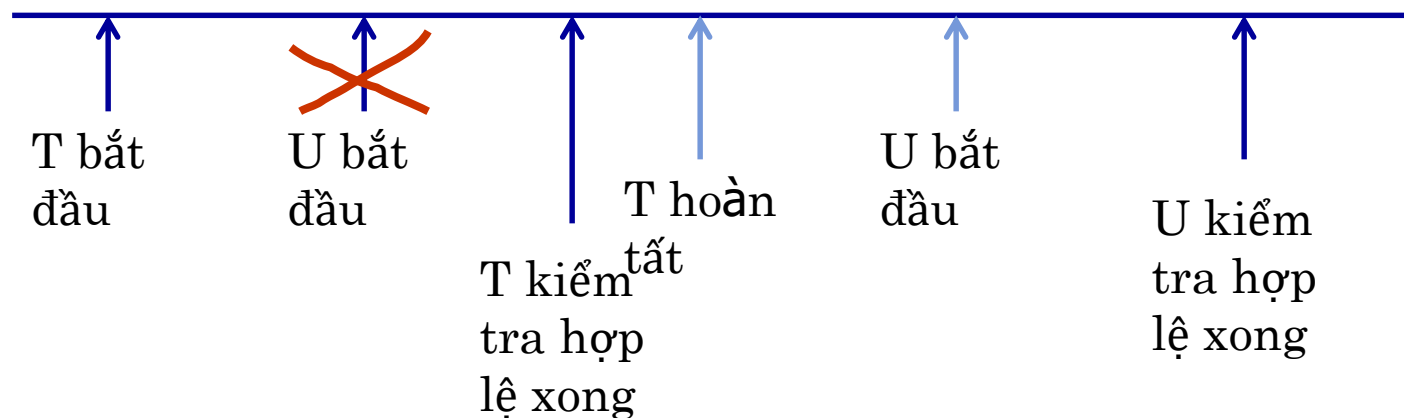
○ Vấn đề 1



- T đã kiểm tra hợp lệ xong
 - T chưa hoàn tất ghi thì U bắt đầu đọc
 - $RS(U) \cap WS(T) = \{X\}$
 - U có thể không đọc được giá trị X ghi bởi T
- Rollback U

XÁC NHẬN HỢP LỆ (TT)

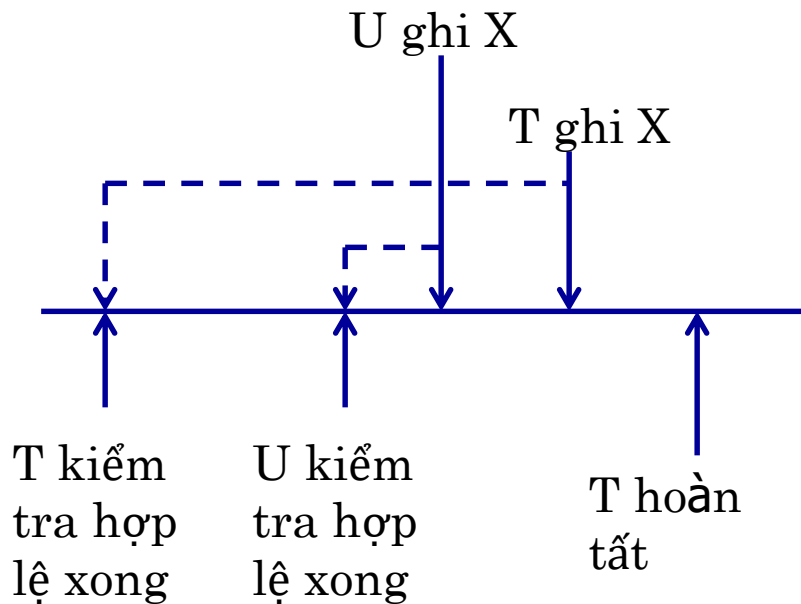
○ Vấn đề 1



Sau khi T hoàn tất thì U mới bắt đầu

XÁC NHẬN HỢP LỆ (TT)

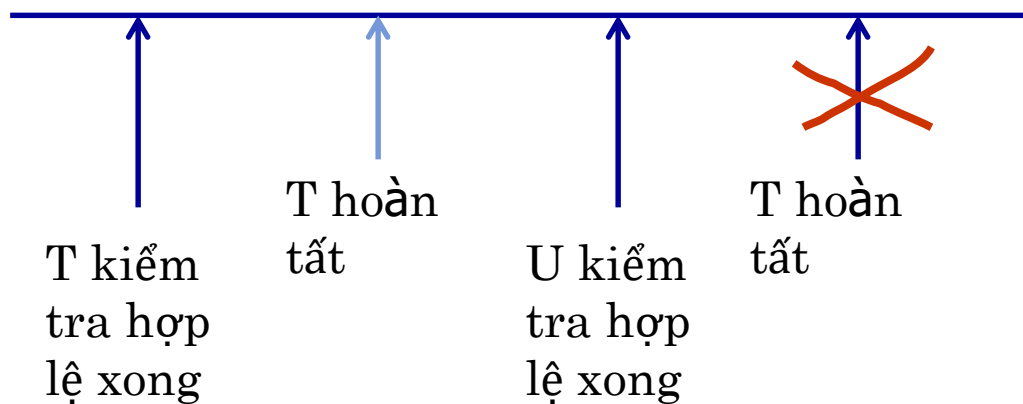
○ Vấn đề 2



- T đã kiểm tra hợp lệ xong
- T chưa hoàn tất ghi thì U kiểm tra hợp lệ
- $WS(U) \cap WS(T) = \{X\}$
- U có thể ghi X trước T
→ Rollback U

XÁC NHẬN HỢP LỆ (TT)

○ Vấn đề 2



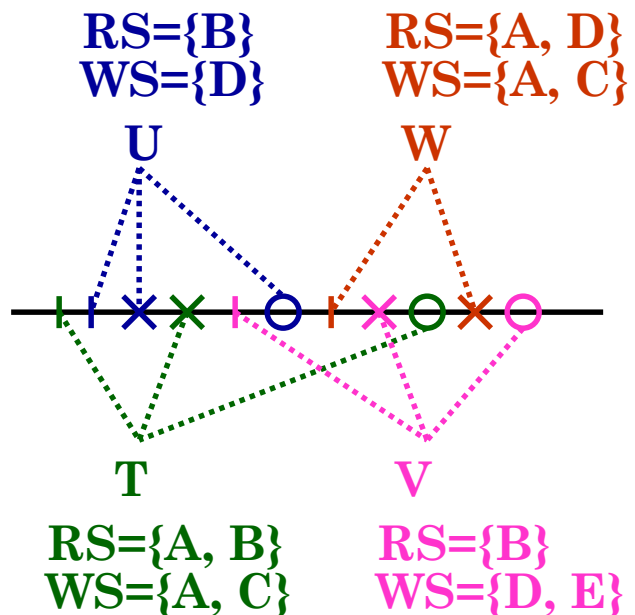
Sau khi T hoàn tất thì U mới được kiểm tra hợp lệ

XÁC NHẬN HỢP LỆ (TT)

○ Qui tắc

- (1) - Nếu có T chưa hoàn tất mà U bắt đầu
 - Kiểm tra $RS(U) \cap WS(T) = \emptyset$
- (2) - Nếu có T chưa hoàn tất mà U kiểm tra hợp lệ
 - Kiểm tra $WS(U) \cap WS(T) = \emptyset$

VÍ DỤ



| = bắt đầu
 X = kiểm tra hợp lệ
 O = hoàn tất

Khi U kiểm tra hợp lệ:

Không có giao tác nào kiểm tra hợp lệ xong trước đó

→ U kiểm tra hợp lệ thành công và ghi D

Khi T kiểm tra hợp lệ:

U đã kiểm tra hợp lệ xong nhưng chưa hoàn tất nên kiểm tra $WS(U) = \{D\}$

$\cap [RS(T) = \{A, B\}, WS(T) = \{A, C\}]$

→ T kiểm tra hợp lệ thành công và ghi A, C

Khi V kiểm tra hợp lệ:

Vì V bắt đầu trước khi U hoàn tất nên kiểm tra $RS(V) \cap WS(U)$

T kiểm tra hợp lệ xong nhưng chưa hoàn tất nên kiểm tra $WS(T) \cap [RS(V), WS(V)]$

→ V kiểm tra hợp lệ thành công và ghi D, E

Khi W kiểm tra hợp lệ:

U hoàn tất trước khi W bắt đầu → kg kiểm tra

Vì W bắt đầu trước khi T hoàn tất nên kiểm tra:

$RS(W) \cap WS(T) = \{A\}$

V kiểm tra hợp lệ xong nhưng chưa hoàn tất nên kiểm tra: $WS(V)$ và $[RS(W), WS(W)]$

* $WS(V) \cap RS(W) = \{D\}$

* $WS(V) \cap WS(W) = \emptyset$

→ W không hợp lệ và phải rollback

BÀI TẬP

Cho tập các giao tác sau:

- 1. $R_1(A,B); R_2(B,C); V_1; R_3(C,D); V_3; W_1(A); V_2; W_2(A); W_3(D);$**
- 2. $R_1(A,B); R_2(B,C); R_3(C); V_1; V_2; V_3; W_1(A); W_2(B); W_3(C);$**

Trong đó:

- $R_i(X)$: giao tác T_i bắt đầu, đọc đơn vị dữ liệu X .**
- V_i : T_i kiểm tra hợp lệ.**
- $W_i(X)$: giao tác T_i kết thúc, ghi đơn vị dữ liệu X .**

Áp dụng kỹ thuật xác nhận hợp lệ cho tập các giao tác trên.

NHẬN XÉT

- Kỹ thuật nào hiệu quả hơn???
 - Khóa (locking)
 - Nhãn thời gian (timestamps)
 - Xác nhận hợp lệ (validation)
- Dựa vào
 - Lưu trữ
 - Tỷ lệ với số lượng đơn vị dữ liệu
 - Khả năng thực hiện
 - Các giao tác ảnh hưởng với nhau như thế nào? Nhiều hay ít?

NHẬN XÉT (TT)

	Khóa	Nhãn thời gian	Xác nhận hợp lệ
Delay	Trì hoãn các giao tác, ít rollback	Không trì hoãn các giao tác, nhưng gây ra rollback	
Rollback		Xử lý rollback nhanh	Xử lý rollback chậm
Storage	Phụ thuộc vào số lượng đơn vị dữ liệu bị khóa	Phụ thuộc vào nhãn đọc và ghi của từng đơn vị dữ liệu	Phụ thuộc vào nhãn WS và RS của các giao tác hiện hành và 1 vài giao tác đã hoàn tất sau 1 giao tác bắt đầu nào đó
		Sử dụng nhiều bộ nhớ hơn	

ảnh hưởng
nhiều

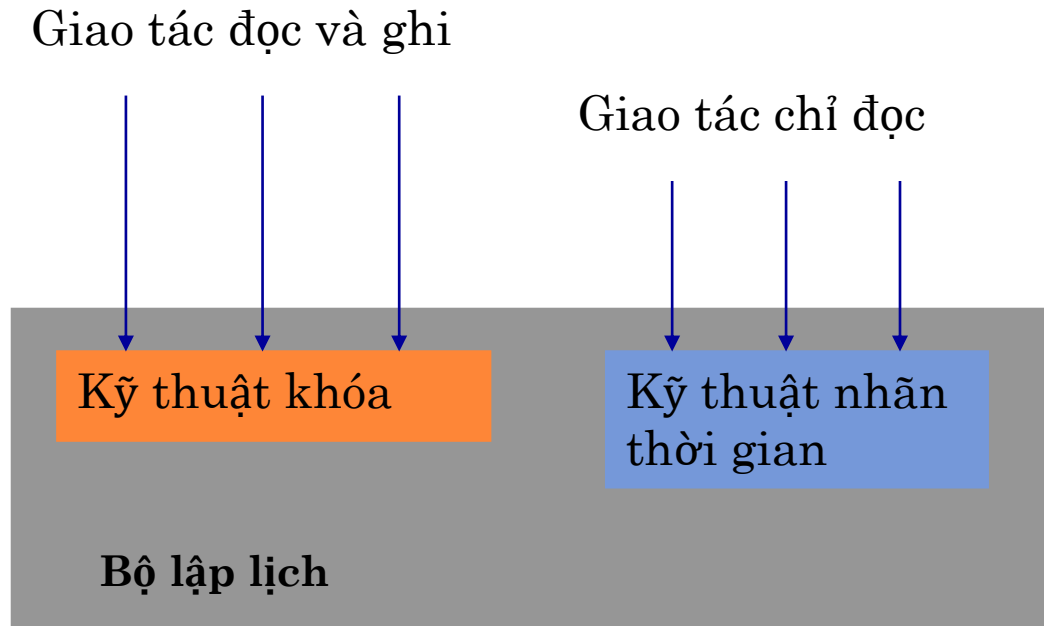
ảnh hưởng ít

NHẬN XÉT (TT)

○ Khóa & nhân thời gian

- Nếu các giao tác chỉ thực hiện đọc không thôi thì kỹ thuật nhân thời gian tốt hơn
 - Ít có tình huống các giao tác cố gắng đọc và ghi cùng 1 đơn vị dữ liệu
- Nhưng kỹ thuật khóa sẽ tốt hơn trong những tình huống xảy ra tranh chấp
 - Kỹ thuật khóa thường hay trì hoãn các giao tác để chờ xin được khóa
 - Dẫn đến deadlock
 - Nếu có các giao tác đọc và ghi cùng 1 đơn vị dữ liệu thì việc rollback là thường xuyên hơn

NHẬN XÉT (TT)



KẾT LUẬN

Mỗi kỹ thuật đều có ưu việt riêng