

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Chương 2:
GIAO TÁC VÀ LỊCH
GIAO TÁC

GVLT: *Nguyễn Trường Sơn*



Nội dung trình bày

- **Giới thiệu**
- **Giao tác**
 - Khái niệm
 - Tính ACID của giao tác
 - Các thao tác của giao tác
 - Các trạng thái của giao tác
- **Lịch thao tác**
 - Giới thiệu
 - Khái niệm
 - Lịch tuần tự
 - Lịch khả tuần tự



Giới thiệu

- Hai yêu cầu cơ bản của ứng dụng khai thác CSDL trong thực tế:
 - Cho phép nhiều người dùng *đồng thời* khai thác CSDL nhưng phải giải quyết được các *tranh chấp*.
 - Sự cố kỹ thuật có thể luôn luôn xảy ra nhưng phải giải quyết được vấn đề về *nhất quán dữ liệu*.
- Một số ví dụ về ứng dụng có sử dụng CSDL :
 - Hệ thống giao dịch ở ngân hàng
 - Hệ thống đặt vé máy bay
 - Hệ thống quản lý học sinh
 - ...



Giới thiệu – Một số tình huống

■ *Hệ thống đặt vé máy bay:*

- “Khi hành khách mua vé”
- “Khi hai hành khách cùng đặt một ghế trống”
- ...

GHẾ (Mã ghế, Mã CB, *Trạng thái*)
CHUYỂN BAY(Mã CB, Ngày giờ, Số ghế còn)

■ *Hệ thống ngân hàng:*

- “Khi chuyển tiền từ tài khoản A sang tài khoản B”
- “Khi rút tiền của một tài khoản”
- “Nhiều người cùng rút tiền trên một tài khoản”
- ...

TÀI KHOẢN(Mã TK, Số dư)
GIAO DỊCH(Mã GD, Loại, Số tiền)

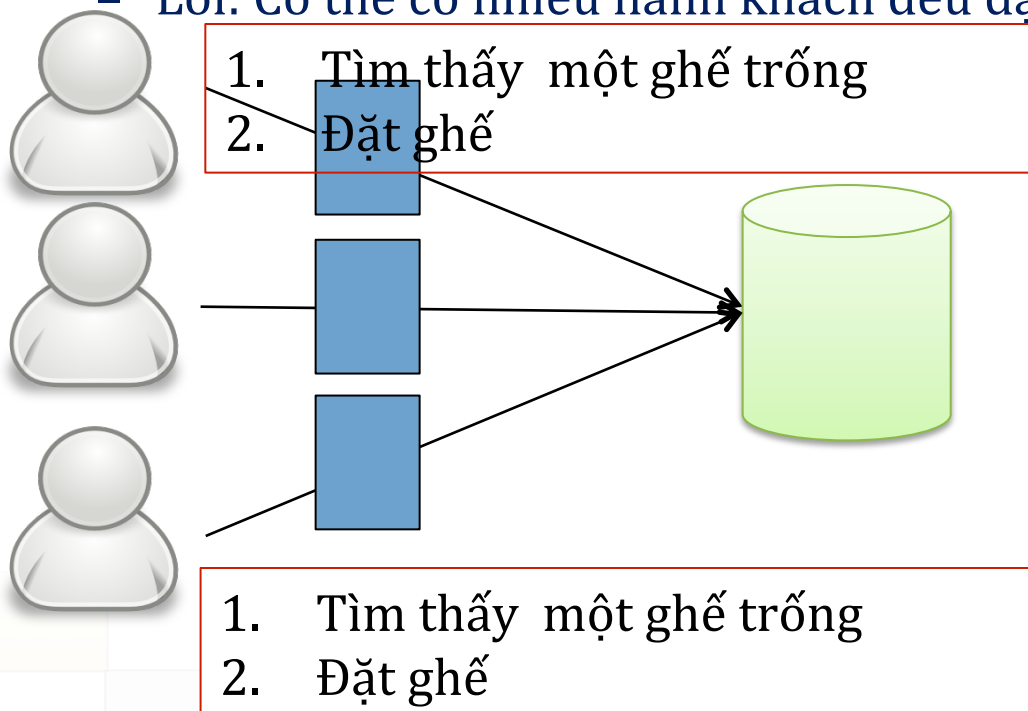
■ *Hệ thống quản lý học sinh:*

- Thêm một học sinh mới
- Chuyển lớp
- ...

Lớp học(Mã lớp, Tên, Sĩ số)
Học sinh (Mã HS, Họ tên, Mã lớp)

Giới thiệu – Một số tình huống

- “Hai (nhiều) hành khách cùng đặt một ghế trống”
 - Lỗi: Có thể có nhiều hành khách đều đặt được dù chỉ còn 1 ghế



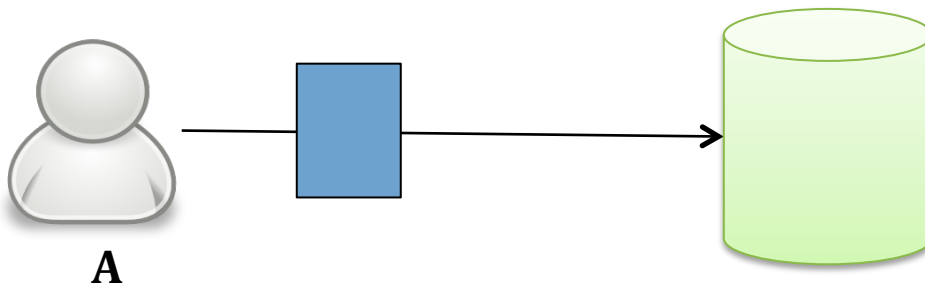
GHẾ (Mã ghế, Mã CB, *Trạng thái*)

Mã ghế	Mã CB	Trạng thái
1001	100	No
1002	100	No
1003	100	Yes
...	...	No
1050	100	No

→ Phải giải quyết được tranh chấp để đảm

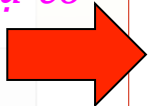
Giới thiệu – Một số tình huống

- “Chuyển tiền từ tài khoản A sang tài khoản B”
 - Lỗi: Có thể đã rút tiền từ A nhưng chưa cập nhật số dư của B



1. update TAIKHOAN set SoDu=SoDu-50
where MATK=A
2. update TAIKHOAN set SoDu=SoDu+50
where MATK=B

Sự cố



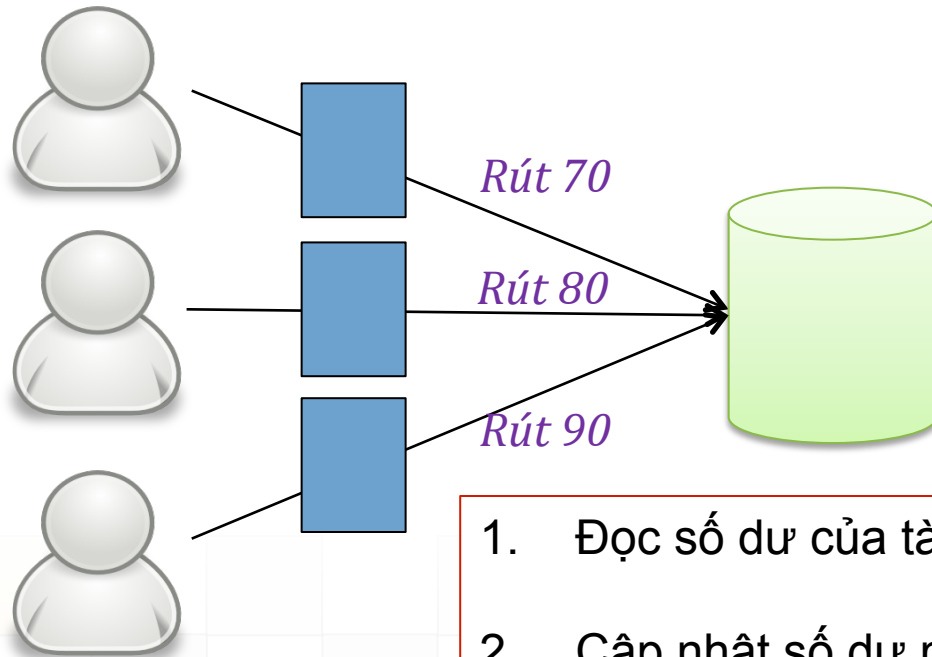
TÀI KHOẢN(Mã TK, Số dư)

Mã TK	Số dư
A	50
B	100
C	60
...	...
N	90

→ Phải đảm bảo được nhất quán dữ liệu khi có sự cố.

Giới thiệu – Một số tình huống

- “Nhiều người cùng rút tiền từ một tài khoản”
 - Lỗi: Có thể rút nhiều hơn số tiền thực có



TÀI KHOẢN(Mã TK, Số dư)

Mã TK	Số dư
A	100

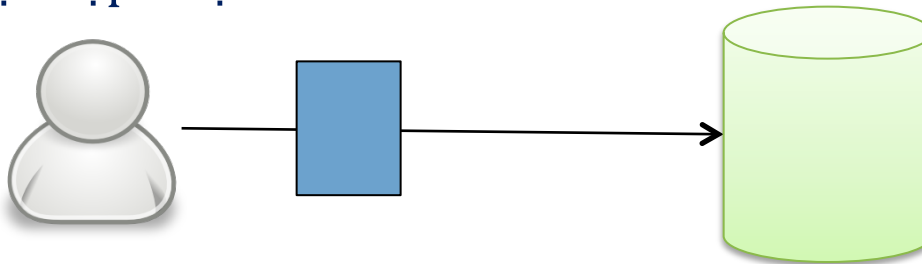
1. Đọc số dư của tài khoản A vào **X**
2. Cập nhật số dư mới của tài khoản A bằng **X – Số tiền**

→ Phải giải quyết được tranh chấp để đảm bảo được nhất quán dữ liệu.

Giới thiệu – Một số tình huống

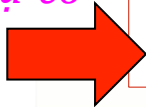
■ “Thêm một học sinh mới”

- Lỗi: Có thể xảy ra trường hợp học sinh đã được thêm nhưng sĩ số không được cập nhật.



1. Thêm vào một học sinh của lớp

Sự cố



2. Cập nhật sĩ số lớp tăng lên 1

→ Phải đảm bảo được nhất quán dữ liệu khi có

Lớp học (Mã lớp, Tên, Sĩ số)

Mã lớp	Tên	Sĩ số
1	10A	3

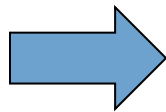
Học sinh (Mã HS, Họ tên, Mã lớp)

Mã HS	Họ tên	Mã lớp
1	An	1
2	Thảo	1
3	Bình	1



Giới thiệu

- Nhận xét:
 - Thường xuyên xảy ra vấn đề nhất quán dữ liệu nếu một xử lý gặp sự cố hoặc khi các xử lý được gọi truy xuất đồng thời.
- Cần 1 khái niệm biểu diễn một đơn vị xử lý với các tính chất:
 - Nguyên tố
 - Cô lập
 - Nhất quán
 - Bền vững



Giao tác

Là một khái niệm nền tảng của điều khiển truy xuất đồng thời và khôi phục khi có sự cố.



Nội dung trình bày

- Giới thiệu
- **Giao tác**
 - Khái niệm
 - Tính ACID của giao tác
 - Các thao tác của giao tác
 - Các trạng thái của giao tác
- Lịch thao tác
 - Giới thiệu
 - Khái niệm
 - Lịch tuần tự
 - Lịch khả tuần tự



Giao tác là gì ?

- **Giao tác (Transaction)** là một đơn vị xử lý **nguyên tố** gồm một chuỗi các hành động đọc / ghi trên các *đối tượng CSDL*
 - **Nguyên tố**: Không thể phân chia được nữa. Các hành động trong một giao tác hoặc là thực hiện được tất cả hoặc là không thực hiện được bất cứ hành động nào.



T

```
-- statement 1  
-- statement 2  
-- statement 3  
--  
-- statement n
```

- Trong kiến trúc hệ quản trị CSDL:
 - Bộ phận **Điều khiển đồng thời** đóng vai trò quản lý giao tác.



Tính chất ACID của giao tác

- Tính nguyên tử (**A**tomicity)
 - Hoặc là toàn bộ hoạt động được phản ánh đúng đắn trong CSDL, hoặc không có hoạt động nào cả.
- Tính nhất quán (**C**onsistency)
 - Khi một giao tác kết thúc (thành công hay thất bại), CSDL phải ở trạng thái nhất quán (Đảm bảo mọi RBTV). Một giao tác đưa CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác.
- Cô lập (**I**solation)
 - Một giao tác khi thực hiện sẽ không bị ảnh hưởng bởi các giao tác khác thực hiện đồng thời với nó.
- Bền vững (**D**urability)
 - Mọi thay đổi trên CSDL được ghi nhận bền vững vào thiết bị lưu trữ dù có sự cố có thể xảy ra.

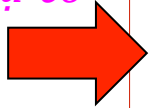
Ví dụ về tính chất **ACID**

Chuyển khoản tiền từ tài khoản A sang tài khoản B

Giao tác Chuyển khoản

1. update TAIKHOAN set SoDu=SoDu-50
where MATK=A

Sự cố



2. update TAIKHOAN set
SoDu=SoDu+50 where MATK=B

Cuối giao tác

Atomicity: Hoặc cả 2 bước trên đều thực hiện hoặc không bước nào được thực hiện. Nếu có sự cố bước 2 thì HQT CSDL có cơ chế khôi phục lại dữ liệu như lúc ban đầu.

Mã TK	Số dư
A	50
B	100
C	60
...	...
N	90

Consistency: Với giao tác chuyển tiền, tổng số dư của A và B luôn luôn không đổi.

Ví dụ về tính chất **ACID**

Thêm học sinh mới vào một lớp

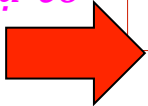
Giao tác Thêm học sinh mới

1. Thêm một học sinh vào bảng học sinh

2. Cập nhật số của lớp tăng lên 1

Cuối giao tác

Sự cố



Atomicity: Hoặc cả 2 bước trên đều thực hiện hoặc không bước nào được thực hiện. Nếu có sự cố bước 2 thì HQT CSDL có cơ chế khôi phục lại dữ liệu như lúc ban đầu.

Lớp học(Mã lớp, Tên, Sĩ số)

Mã lớp	Tên	Sĩ số
--------	-----	-------

1	10A	3
---	-----	---

Học sinh (Mã HS, Họ tên, Mã lớp)

Mã HS	Họ tên	Mã lớp
-------	--------	--------

1	An	1
---	----	---

2	Thảo	1
---	------	---

3	Bình	1
---	------	---

Consistency: Sĩ số của lớp phải luôn bằng số học sinh thực sự và không quá 3.

Ví dụ về tính chất **ACID**

Rút tiền (TK1, 80)

T1

Đọc số dư: t

Cập nhật số dư ($=t-80$)

Gửi tiền (TK1, 50)

T2

Đọc số dư: t

Cập nhật số dư ($=t+50$)

Thời gian

Tài khoản (Mã TK, Số dư)

Mã TK	Số dư
1	100
2	500
3	200

Isolation: Tính chất cô lập đảm bảo mặc dù các giao tác có thể đan xen nhau nhưng kết quả của chúng tương tự với một kết quả tuần tự nào đó → Các giao tác không bị ảnh hưởng bởi các giao tác khác khi thực thi.



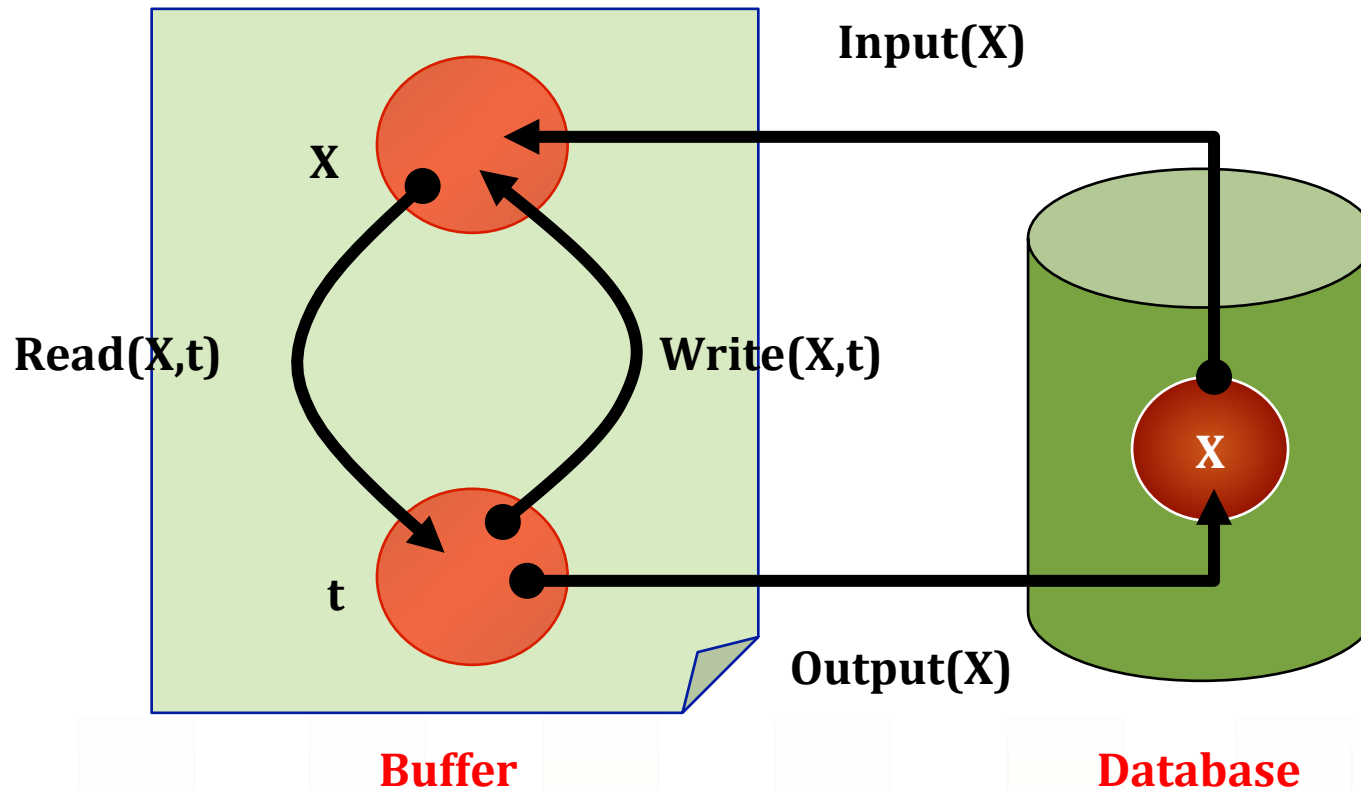
Đơn vị dữ liệu

- Đối tượng CSDL mà giao tác thực hiện các xử lý đọc /ghi còn được gọi là **đơn vị dữ liệu**.



- Một đơn vị dữ liệu (element) có thể là các thành phần :
 - Quan hệ (Relations)
 - Khối dữ liệu trên đĩa (Blocks)
 - Bộ (Tuples)
- Một CSDL bao gồm nhiều đơn vị dữ liệu.

Các thao tác của giao tác



Read (A, t) : Đọc đơn vị dữ liệu A vào t

Write (A, t) : Ghi t vào đơn vị dữ liệu A



Ví dụ về biểu diễn giao tác

- Giả sử có 2 đơn vị dữ liệu A và B với ràng buộc $A = B$ (nếu có một trạng thái nào đó mà $A \neq B$ thì sẽ mất tính nhất quán)
- Giao tác T thực hiện 2 bước:
 - $A = A * 2$
 - $B = B * 2$

- Biểu diễn T:

T
<i>Read(A, t);</i> <i>t = t * 2;</i> <i>Write(A, t)</i> <i>Read(B, t);</i> <i>t = t * 2;</i> <i>Write(B, t)</i>

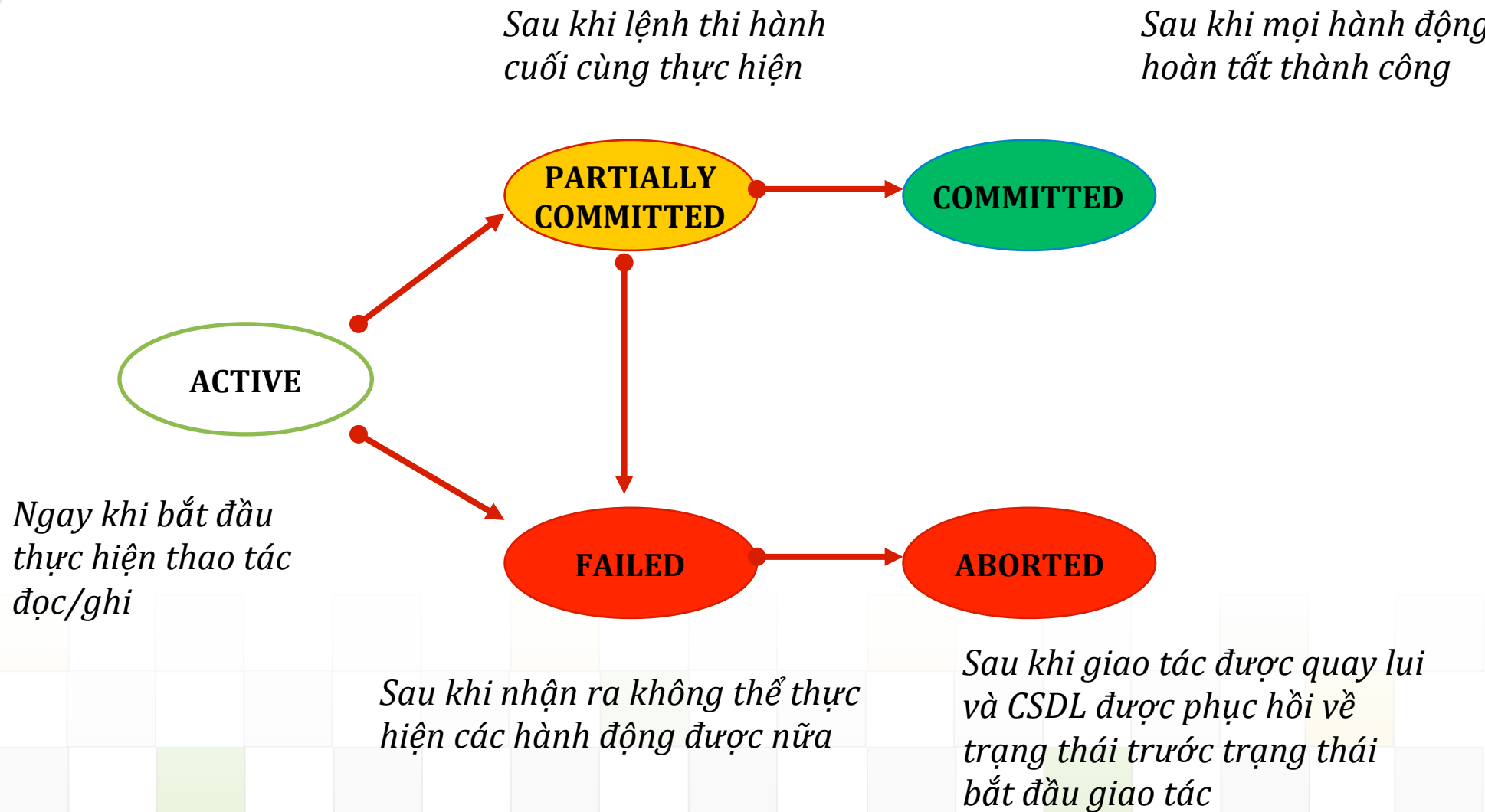
- Hoặc:

T: $\text{Read}(A, t); t = t * 2; \text{Write}(A, t); \text{Read}(B, t); t = t * 2; \text{Write}(B, t)$

Giao tác: Ví dụ (tt)

Hành động	t	Mem A	Mem B	Disk A	Disk B
Input (A)		8		8	8
Read (A, t)	8	8		8	8
$t := t * 2$	16	8		8	8
Write (A, t)	16	16		8	8
Input (B)	16	16	8	8	8
Read (B, t)	8	16	8	8	8
$t := t * 2$	16	16	8	8	8
Write (B, t)	16	16	16	8	8
Output (A)	16	16	16	16	8
Output (B)	16	16	16	16	16

Các trạng thái của giao tác





Khai báo giao tác trong T-SQL

BEGIN TRANSACTION	Bắt đầu giao tác
COMMIT TRANSACTION	Kết thúc giao tác thành công
ROLLBACK TRANSACTION	Kết thúc giao tác không thành công. CSDL được đưa về tình trạng trước khi thực hiện giao tác.



Nội dung trình bày

- Giới thiệu
- Giao tác
 - Khái niệm
 - Tính ACID của giao tác
 - Các thao tác của giao tác
 - Các trạng thái của giao tác
- **Lịch thao tác**
 - Giới thiệu
 - Khái niệm
 - Lịch tuần tự
 - Lịch khả tuần tự




Các cách thực hiện của các giao tác

- Thực hiện tuần tự: Các thao tác khi thực hiện mà không giao nhau về mặt thời gian.
 - ↑ **Ưu**: Nếu thao tác đúng đắn thì luôn luôn đảm bảo nhất quán dữ liệu.
 - ↓ **Khuyết**: Không tối ưu về việc sử dụng tài nguyên và tốc độ.
- Thực hiện đồng thời: Các lệnh của các giao tác khác nhau xen kẽ nhau trên trục thời gian.
 - ↓ **Khuyết**: Gây ra nhiều phức tạp về nhất quán dữ liệu
 - ↑ **Ưu**:
 - **Tận dụng tài nguyên và thông lượng (throughput)**. Ví dụ: Trong khi một giao tác đang thực hiện việc đọc / ghi trên đĩa, một giao tác khác đang xử lý tính toán trên CPU.
 - **Giảm thời gian chờ**. Ví dụ: Chia sẻ chu kỳ CPU và truy cập đĩa để làm giảm sự trì hoãn trong các giao tác thực thi.

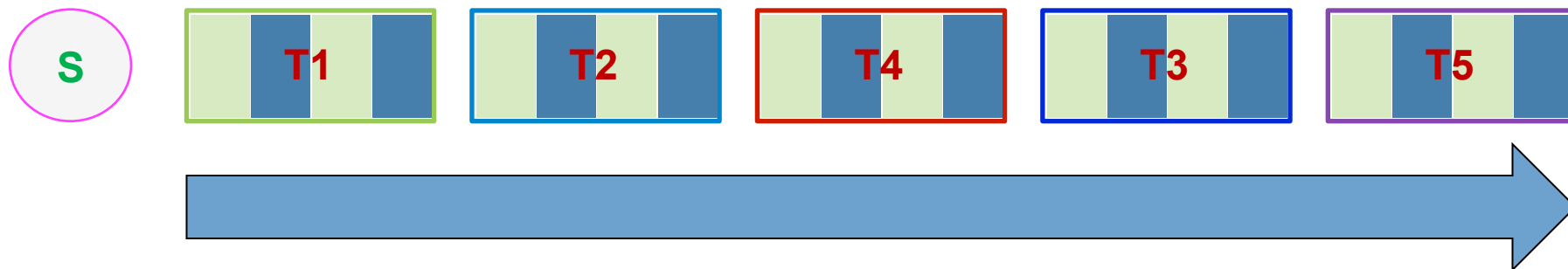


Lịch thao tác là gì ?

- 
- Định nghĩa: Một lịch thao tác **S** được lập từ **n** giao tác T_1, T_2, \dots, T_n được xử lý đồng thời là một thứ tự thực hiện xen kẽ các hành động của **n** giao tác này.
 - Thứ tự xuất hiện của các thao tác trong lịch phải **giống** với thứ tự xuất hiện của chúng trong giao tác.
 - Bộ lập lịch (Scheduler): Là một thành phần của DBMS có nhiệm vụ lập một lịch để thực hiện n giao tác xử lý đồng thời.
 - *Các loại lịch thao tác*:
 - **Lịch tuần tự** (Serial)
 - **Lịch khả tuần tự** (Serializable):
 - Conflict – Serializability
 - View – Serializability

Lịch tuần tự

- Một lịch S được gọi là tuần tự nếu các hành động của các giao tác T_i được thực hiện liên tiếp nhau, không có sự giao nhau về mặt thời gian.



Lịch tuần tự

Ví dụ:

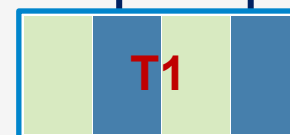
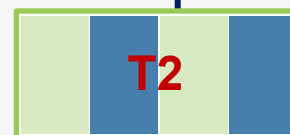
S1

T1	T2	A	B
Read(A, t)		25	25
t:=t+100			
Write(A, t)		125	
Read(B, t)			
t:=t+100			
Write(B, t)			125
	Read(A, s)		
	s:=s*2		
	Write(A, s)	250	
	Read(B, s)		
	s:=s*2		
	Write(B, s)		250



S2

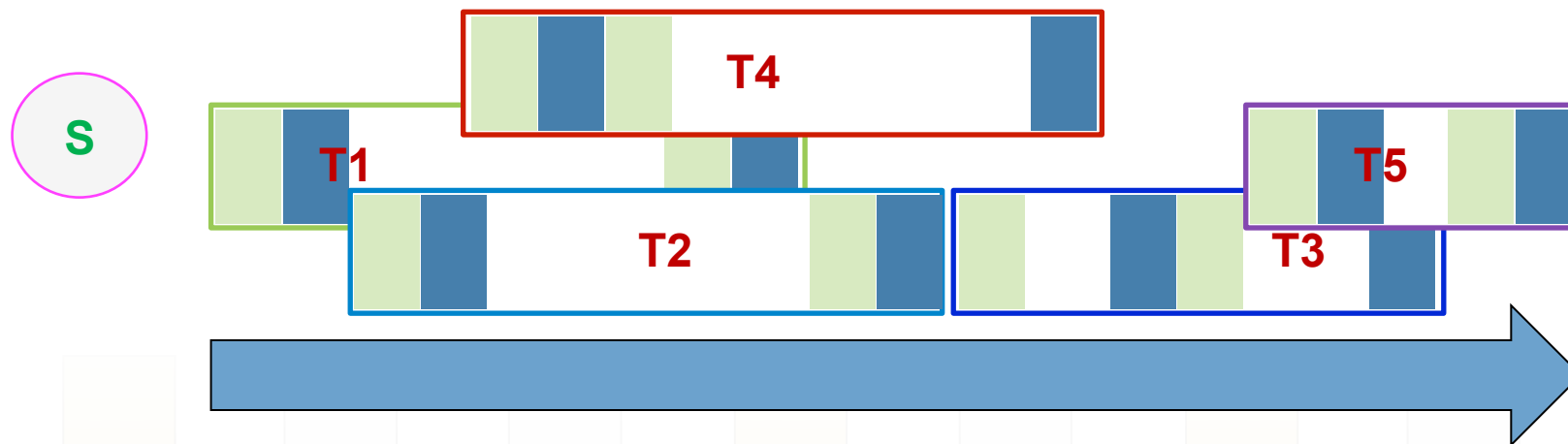
T1	T2	A	B
	Read(A, s)	25	25
	s:=s*2		
	Write(A, s)	50	
	Read(B, s)		
	s:=s*2		
	Write(B, s)		50
Read(A, t)			
t:=t+100			
Write(A, t)		150	
Read(B, t)			
t:=t+100			
Write(B, t)			150



Lịch tuần tự luôn luôn đảm bảo được tính nhất quán của CSDL

Lịch xử lý đồng thời

- Lịch xử lý đồng thời là lịch mà các giao tác trong đó giao nhau về mặt thời gian



Lịch xử lý đồng thời

Lịch đồng thời luôn luôn tiềm ẩn khả năng làm cho CSDL mất tính nhất quán

Lịch đồng thời

S3

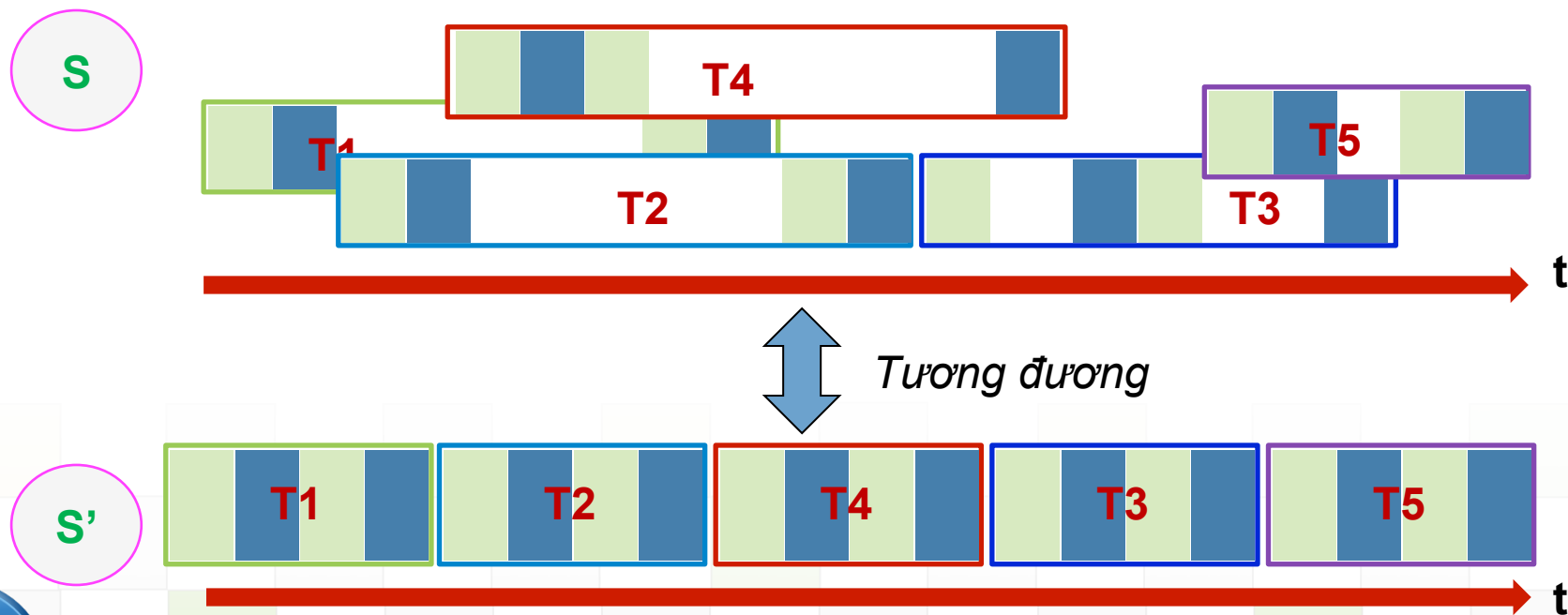
T1	T2	A	B
Read(A, t) t:=t+100 Write(A,t)		25	25
	Read(A, s) s:=s*2 Write(A,s)	125	
	Read(B, s) s:=s*2 Write(B, s)	250	50
Read(B, t) t:=t+100 Write(B, t)			150

Ví dụ:

- S3 là một lịch xử lý đồng thời vì các giao tác giao thoa với nhau
- Lịch xử lý đồng thời S3 gây ra sự mất nhất quán dữ liệu
 - Trước S khi thực hiện
 - $A=B$
 - Sau khi S kết thúc
 - $A \neq B$

Lịch khả tuần tự

- Một lịch S được lập ra từ n giao tác T1, T2, ..., Tn *xử lý đồng thời* được gọi là **lịch khả tuần tự** nếu nó cho cùng kết quả với một lịch tuần tự nào đó được lập ra từ n giao tác này.



Lịch khả tuần tự cũng không gây nên tình trạng mất nhất quán dữ liệu

Lịch khả tuần tự

Ví dụ:

- Trước S_4 khi thực hiện
 - $A=B=c$
 - với c là hằng số
- Sau khi S_4 kết thúc
 - $A=2*(c+100)$
 - $B=2*(c+100)$
- Trạng thái CSDL nhất quán
- S_4 là **khả tuần tự**

(Kết quả của lịch S_4 tương tự với kết quả của lịch tuần tự $S < T1, T2 >$)

S_4	T1	T2	A	B
	Read(A, t) $t:=t+100$ Write(A, t)	Read(A, s) $s:=s*2$ Write(A, s)	25 125 250	25 125 250
	Read(B, t) $t:=t+100$ Write(B, t)	Read(B, s) $s:=s*2$ Write(B, s)		

Lịch khả tuần tự

Ví dụ:

■ Trước S_5 khi thực hiện

- $A=B=c$
- với c là hằng số

■ Sau khi S_5 kết thúc

- $A = 2*(c+100)$
- $B = 2*c + 100$

→ Trạng thái CSDL không nhất quán

■ S_5 **không** khả tuần tự

S_5	T1	T2	A	B
			25	25
	Read(A, t) $t:=t+100$ Write(A, t)		125	
		Read(A, s) $s:=s*2$ Write(A, s)	250	
		Read(B, s) $s:=s*2$ Write(B, s)		50
	Read(B, t) $t:=t+100$ Write(B, t)			150

Lịch khả tuần tự

Ví dụ:

■ Trước S_{5b} khi thực hiện

- $A=B=c$
- với c là hằng số

■ Sau khi S_{5b} kết thúc

- $A = 1 * (c + 100)$
- $B = 1 * c + 100$

➔ Trạng thái CSDL vẫn nhất quán

S_{5b}	T1	T2	A	B
	Read(A, t) $t := t + 100$ Write(A, t)		25	25
		Read(A, s) $s := s * 1$ Write(A, s)	125	
		Read(B, s) $s := s * 1$ Write(B, s)	125	
	Read(B, t) $t := t + 100$ Write(B, t)			25
				125

- ❑ Để xem xét tính nhất quán một cách kỹ lưỡng → phải hiểu rõ ngữ nghĩa của từng giao tác → không khả thi
- ❑ Thực tế chỉ xem xét các lệnh của giao tác là đọc hay ghi

Biểu diễn lịch thao tác

Cách 1

S	T1	T2
	Read(A, t) t:=t+100 Write(A,t)	Read(A, s) s:=s*2 Write(A,s)
	Read(B, t) t:=t+100 Write(B, t)	Read(B, s) s:=s*2 Write(B, s)

Cách 2

S	T1	T2
	Read(A) Write(A)	Read(A) Write(A)
	Read(B) Write(B)	Read(B) Write(B)

Cách 3

S: R1(A); W1(A); R2(A); W2(A); R1(B); W1(B); R2(B); W2 (B)



Lịch khả tuần tự

- Có 2 loại lịch khả tuần tự:

Conflict Serializable

- Dựa trên ý tưởng hoán vị các hành động không xung đột để chuyển một lịch đồng thời S về một lịch tuần tự S' . Nếu có một cách biến đổi như vậy thì S là một lịch conflict serializable.

View Serializable

- Dựa trên ý tưởng lịch đồng thời S và lịch tuần tự S' đọc và ghi những giá trị dữ liệu giống nhau. Nếu có một lịch S' như vậy thì S là một lịch view serializable.

Conflict Serializability

- **Ý tưởng:** Xét 2 hành động liên tiếp nhau của 2 giao tác khác nhau trong một lịch thao tác, khi 2 hành động đó được đảo thứ tự có thể dẫn đến 1 trong 2 hệ quả:
 - Hoạt động của cả hai giao tác chứa 2 hành động ấy **không** bị ảnh hưởng gì → **2 hành động đó không xung đột với nhau**
 - Hoạt động của ít nhất một trong 2 giao tác chứa 2 hành động ấy bị ảnh hưởng → **2 hành động xung đột**

T	T'
Hành động 1	
Hành động 2	Hành động 1'
	Hành động 2'
Hành động 3	
Hành động 4	Hành động 3'
	Hành động 4'



Conflict Serializability (tt)

- Cho lịch S có 2 giao tác T_i và T_j , xét các trường hợp:
 - $r_i(X) ; r_j(Y)$
 - **Không bao giờ có xung đột**, ngay cả khi $X=Y$
 - Cả 2 thao tác không làm thay đổi giá trị của X và Y
 - $r_i(X) ; w_j(Y)$
 - **Không xung đột khi $X \neq Y$**
 - T_j không thay đổi dữ liệu đọc của T_i
 - T_i không sử dụng dữ liệu ghi của T_j
 - **Xung đột khi $X = Y$**
 - $w_i(X) ; r_j(Y)$
 - **Không xung đột khi $X \neq Y$, Xung đột khi $X=Y$**
 - $w_i(X) ; w_j(Y)$
 - **Không xung đột khi $X \neq Y$, Xung đột khi $X=Y$**



Conflict Serializability (tt)

- Tóm lại, hai hành động xung đột nếu chúng:

- ❖ Thuộc 2 giao tác khác nhau
- ❖ Truy xuất đến 1 đơn vị dữ liệu
- ❖ Trong chúng có ít nhất một hành động ghi (write)

- Hai hành động xung đột thì không thể nào đảo thứ tự của chúng trong một lịch thao tác.

Conflict Serializability (tt)

■ Ví dụ:


S_6	T_1	T_2		S_6'	T_1	T_2
	Read(A)				Read(A)	
	Write(A)				Write(A)	
		Read(A)			Read(B)	
		Write(A)			Write(B)	
	Read(B)					Read(A)
	Write(B)					Write(A)
		Read(B)				Read(B)
		Write(B)				Write(B)

S_6 có khả năng chuyển đổi thành S_6' bằng cách hoán vị các cặp hành động không xung đột hay không ?



Conflict Serializability (tt)

■ Định nghĩa:

- 
- S và S' là những lịch thao tác *conflict equivalent* nếu S có thể chuyển được thành S' thông qua một chuỗi các hoán vị những thao tác không xung đột.
 - Một lịch thao tác S là conflict serializable nếu S là *conflict equivalent* với một lịch thao tác tuần tự nào đó.

■ S là conflict serializable thì S khả tuần tự.

■ S là khả tuần tự thì không chắc S conflict serializable

■ Lịch S ở slide trước có phải conflict serializable hay không ?

Conflict Serializability (tt)

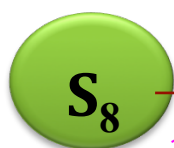
- Xét lịch **S7** như sau:

	T₁	T₂
S₇		
1	Read(A)	
2		Read(A)
3	Write(A)	
4		Write(A)

- Lịch **S7** trên có thỏa Conflict Serializability hay không ?

Conflict Serializability (tt)

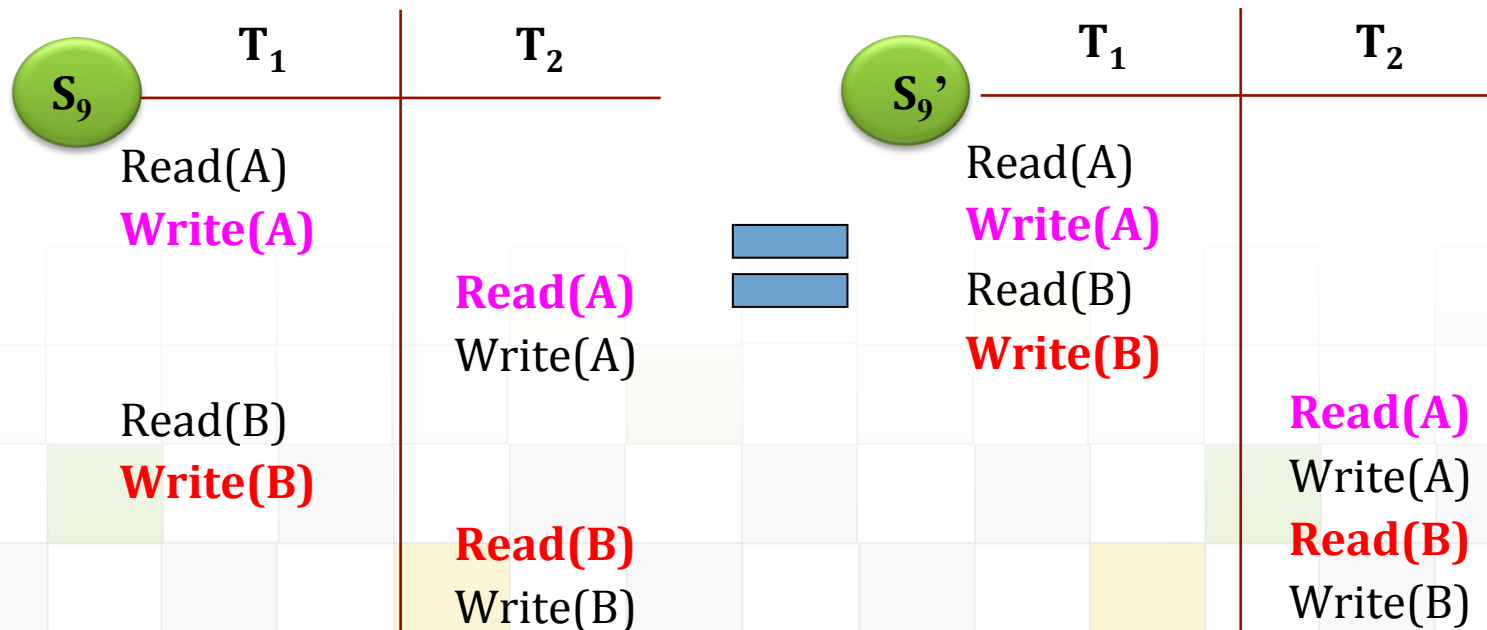
- Xét lịch S_8 như sau:

 S_8	T_1	T_2
1	Read(A)	
2	Write(A)	
3		Read(A)
4		Write(A)
5		Read(B)
6		Write(B)
7	Read(B)	
8	Write(B)	

- Lịch S_8 trên có thỏa Conflict Serializability hay không ?

Kiểm tra Conflict Serializability

- Cho lịch S_9 :
 - S_9 có conflict serializability hay không ?
- Ý tưởng:
 - Các hành động xung đột trong lịch S được thực hiện theo thứ tự nào thì các giao tác thực hiện chúng trong S' (kết quả sau hoán vị) cũng theo thứ tự đó.





Kiểm tra Conflict Serializability

- Cho lịch S có 2 giao tác T1 và T2:
 - T1 thực hiện hành động A1
 - T2 thực hiện hành động A2
 - Ta nói T1 thực hiện trước hành động **T2** trong S, ký hiệu $T1 <_s T2$, khi:
 - **A1** được thực hiện trước **A2** trong S
 - A1 không nhất thiết phải liên tiếp A2
 - **A1** và **A2** là 2 hành động xung đột (A1 và A2 cùng thao tác lên 1 đơn vị dữ liệu và có ít nhất 1 hành động là ghi trong A1 và A2)



Kiểm tra Conflict Serializability

Phương pháp Precedence Graph:

- Cho lịch S bao gồm các thao tác T_1, T_2, \dots, T_n
- Đồ thị trình tự của S (Precedence graph) của S ký hiệu là $P(S)$ có:
 - Đỉnh là các giao tác T_i
 - Cung đi từ T_i đến T_j nếu $T_i <_S T_j$
- S Conflict Serializable khi và chỉ khi $P(S)$ không có chu trình
- Thứ tự hình học các đỉnh là thứ tự của các giao tác trong lịch tuần tự tương đương với S.
- Với 2 lịch S và S' được lập từ cùng các giao tác, S và S' conflict equivalent khi và chỉ khi $P(S) = P(S')$

Kiểm tra Conflict Serializability

- Ví dụ 1: S_{10} có Conflict Serializability hay không ?

S_{10}	T_1	T_2	T_3
		Read(A)	
Read(B)		Write(A)	
			Read(A)
Write(B)			Write(A)
		Read(B)	
		Write(B)	



* $P(S_{10})$ không có chu trình

* S_{10} conflict-serializable theo thứ tự T_1, T_2, T_3

Kiểm tra Conflict Serializability

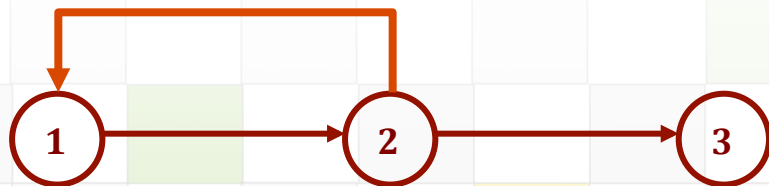
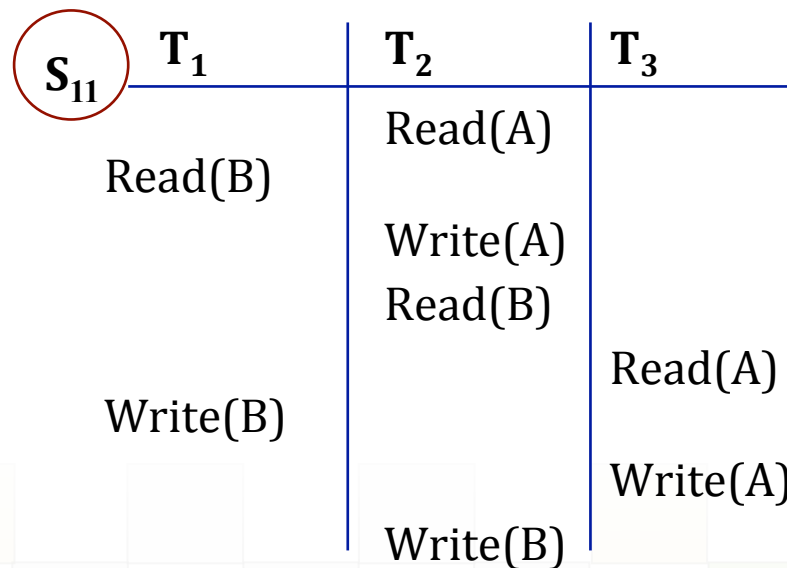
- Ví dụ 1: S_{10} có Conflict Serializability hay không ?

S_{10}	T_1	T_2	T_3		S	T_1	T_2	T_3
	Read(B)	Read(A)				Read(B)		
		Write(A)				Write(B)		
	Write(B)		Read(A)				Read(A)	
			Write(A)				Write(A)	
		Read(B)					Read(B)	
		Write(B)					Write(B)	
								Read(A)
								Write(A)

Kiểm tra Conflict Serializability

- Ví dụ 2: S_{11} có Conflict Serializability hay không ?

S_{11} : $r_2(A)$ $r_1(B)$ $w_2(A)$ $r_2(B)$ $r_3(A)$ $w_1(B)$ $w_3(A)$ $w_2(B)$



* $P(S_{11})$ có chu trình

* S_{11} không conflict-serializable



Bài tập Conflict-Serializability

■ Cho các lịch S sau:

1. S: $w_1(A)$ $r_2(A)$ $r_3(A)$ $w_4(A)$
2. S: $w_3(A)$ $w_2(C)$ $r_1(A)$ $w_1(B)$ $r_1(C)$ $w_2(A)$ $r_4(A)$ $w_4(D)$
3. **S: $r_1(A)$ $w_2(A)$ $w_1(A)$ $w_3(A)$**
4. **S: $r_2(B)$ $w_2(A)$ $r_1(A)$ $r_3(A)$ $w_1(B)$ $w_2(B)$ $w_3(B)$**
5. S: $w_1(X)$ $w_2(Y)$ $w_2(X)$ $w_1(X)$ $w_3(X)$
6. S: $r_2(A)$ $r_1(B)$ $w_2(A)$ $r_3(A)$ $w_1(B)$ $r_2(B)$ $w_2(B)$
7. S: $r_2(A)$ $r_1(B)$ $w_2(A)$ $r_2(B)$ $r_3(A)$ $w_1(B)$ $w_3(A)$ $w_2(B)$

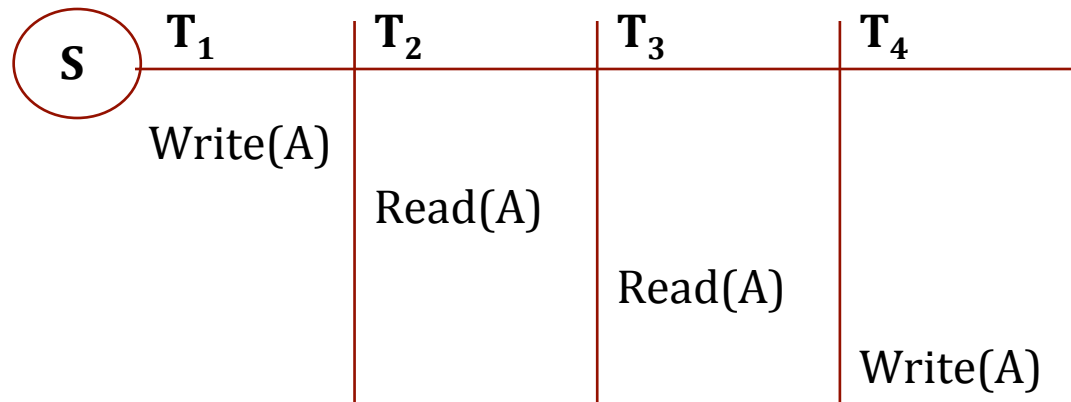
■ Vẽ P(S)

- S có conflict-serializable không? Nếu có thì S tương đương với lịch tuần tự nào ?

Bài tập 1

- Cho lịch S:

S: $w1(A)$ $r2(A)$ $r3(A)$ $w4(A)$

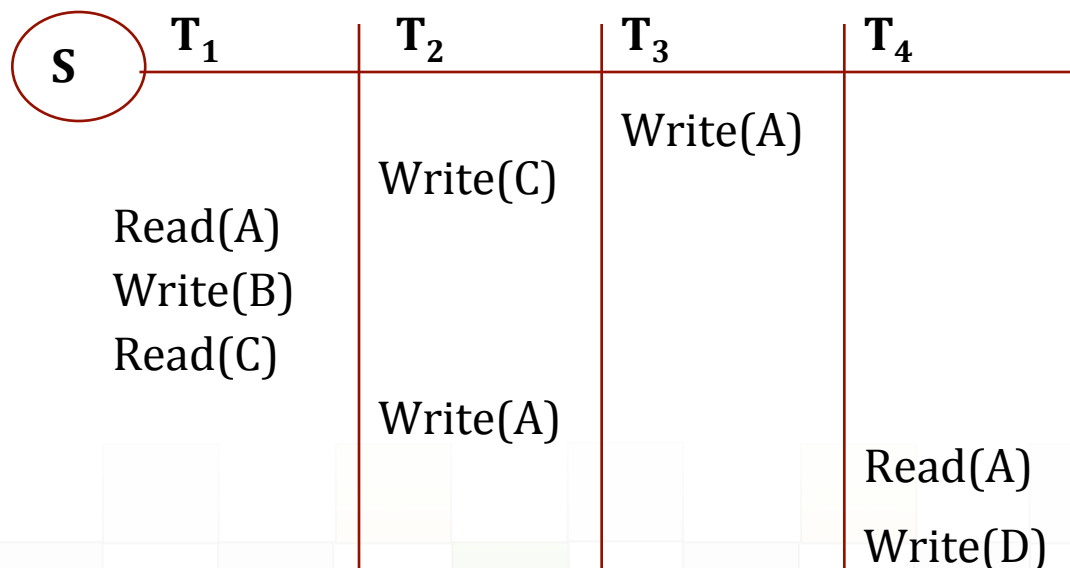


- Vẽ P(S)
- S có conflict-serializable không?

Bài tập 2

- Cho lịch S:

S: w3(A) w2(C) r1(A) w1(B) r1(C) w2(A) r4(A) w4(D)

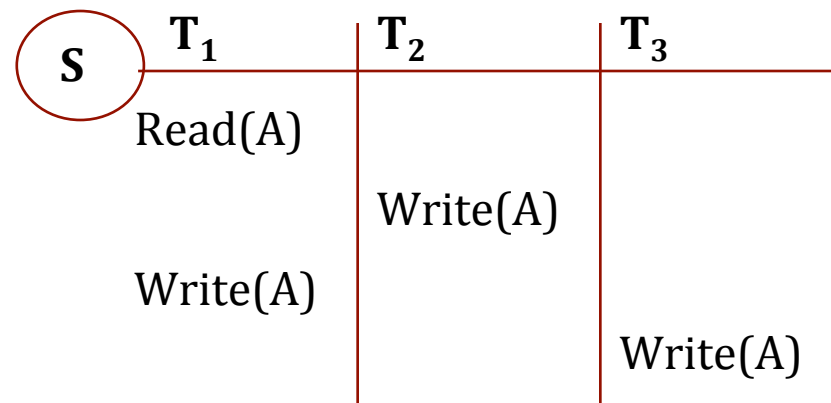


- Vẽ P(S)
- S có conflict-serializable không?

Bài tập 3

- Cho lịch S:

S: r1(A) w2(A) w1(A) w3(A)



- Vẽ P(S)
- S có conflict-serializable không?



Bài tập 4

- Cho lịch S:

S: $r_2(B)$ $w_2(A)$ $r_1(A)$ $r_3(A)$ $w_1(B)$ $w_2(B)$ $w_3(B)$

- Vẽ P(S)
- S có conflict-serializable không ?

Bài tập 5

- Cho lịch S:

S: w1(X) w2(Y) w2(X) w1(X) w3(X)

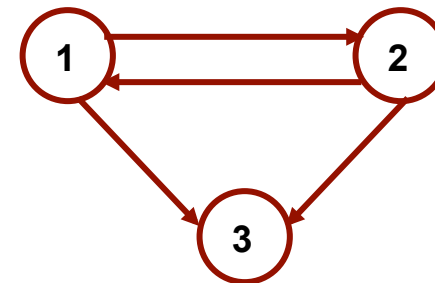
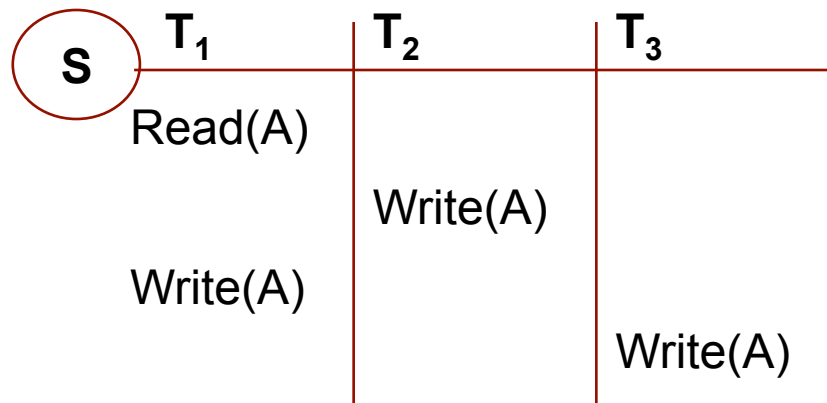
S

T1	T2	T3
Write(Y)		
	Write(Y)	
	Write(X)	
Write(X)		
		Write(X)

- Vẽ đồ thị P(S)
- S có conflict serializable hay không ?

View-Serializability

■ Xét lịch S



* $P(S)$ có chu trình

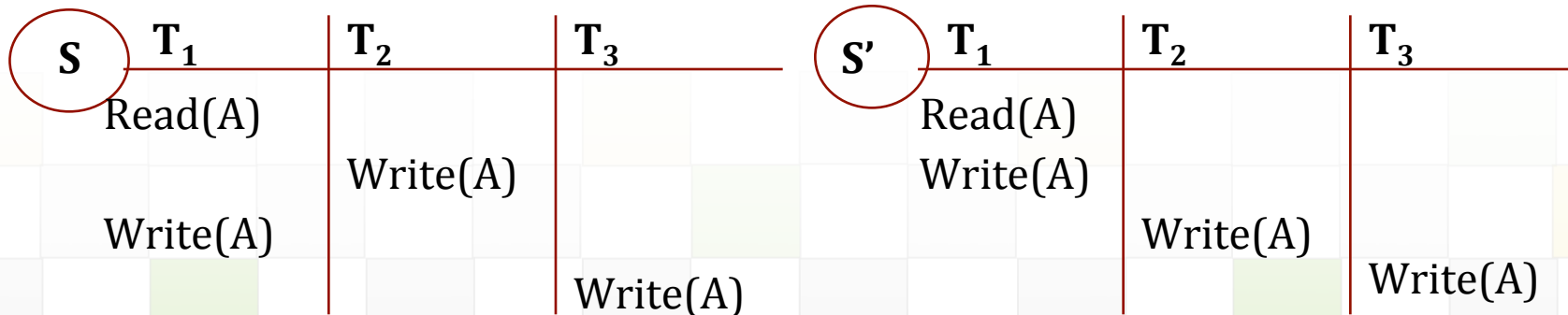
* S không conflict-serializable

* S khả tuần tự hay không ?

View-Serializability (tt)

■ So sánh lịch S và 1 lịch tuần tự S'

- Giả sử trước khi lịch S thực hiện, có giao tác Tb thực hiện việc ghi A và sau khi S thực hiện có giao tác Tf thực hiện việc đọc A.
 - Nhận xét lịch S và S':
 - Đều có T1 thực hiện read(A) từ giao tác Tb → Kết quả đọc luôn giống nhau
 - Đều có T3 thực hiện việc ghi cuối cùng lên A. T2, T3 không có lệnh đọc A → Dù S hay S' được thực hiện thì kết quả đọc A của Tf luôn giống nhau
- Kết quả của S và S' giống nhau → S vẫn khả tuần tự




Không conflict-serializable

Serial



View-Serializability (tt)

■ Khả tuần tự View (View-serializability):

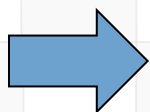
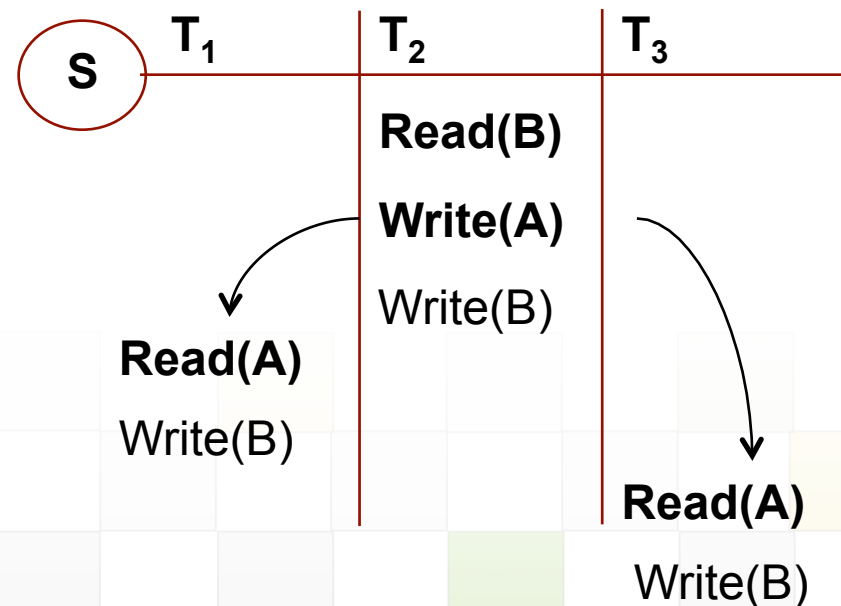
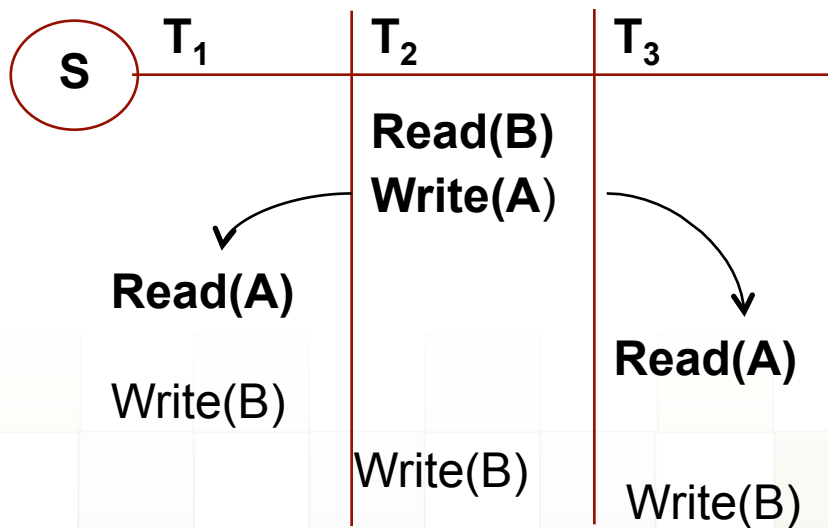
- 
- Một lịch S được gọi là khả tuần tự view nếu tồn tại một lịch tuần tự S' được tạo từ các giao tác của S sao cho S và S' đọc và ghi những giá trị giống nhau.
 - Lịch S được gọi là khả tuần tự view khi và chỉ khi nó tương đương view (*view-equivalent*) với một lịch tuần tự S'

View-Serializability (tt)

- Ví dụ: Cho lịch S và S' như sau:

S : r2(B) w2(A) r1(A) r3(A) w1(B) w2(B) w3(B)

S' : r2(B) w2(A) w2(B) r1(A) w1(B) r3(A) w3(B)



S khả tuần tự view



View-Serializability (tt)

- **View-equivalent:** S và S' là những lịch view-equivalent nếu thỏa các điều kiện sau:
 - Nếu trong S có T_i đọc giá trị ban đầu của A thì nó cũng đọc giá trị ban đầu của A trong S' .
 - Nếu T_i đọc giá trị của A được ghi bởi T_j trong S thì T_i cũng phải đọc giá trị của A được ghi bởi T_j trong S' .
 - Với mỗi dvtl A , giao tác thực hiện lệnh ghi cuối cùng lên A (nếu có) trong S thì giao tác đó cũng phải thực hiện lệnh ghi cuối cùng lên A trong S' .
- Một lịch giao tác S là view-serializable:
 - Nếu S là view-equivalent với một Lịch giao tác tuần tự S' nào đó
- Nếu S là conflict-serializable $\rightarrow S$ view-serializable.



Chứng minh (*)



View Serializability (tt)

Lịch thao tác

View-Serializable

**Conflict-
Serializable**

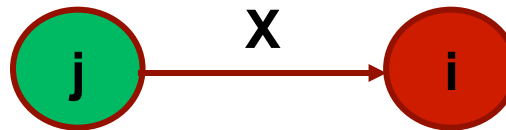


Kiểm tra View Serializability

- Cho 1 Lịch giao tác S
- Thêm 1 giao tác cuối **Tf** vào trong S sao cho **Tf** thực hiện việc đọc hết tất cả đơn vị dữ liệu ở trong S
 - $S = \dots w1(A) \dots w2(A) \text{ rf}(A)$
- Thêm 1 giao tác đầu tiên **Tb** vào trong S sao cho **Tb** thực hiện việc ghi các giá trị ban đầu cho tất cả đơn vị dữ liệu
 - $S = \text{wb}(A) \dots w1(A) \dots w2(A) \dots$

Kiểm tra View-Serializability (tt)

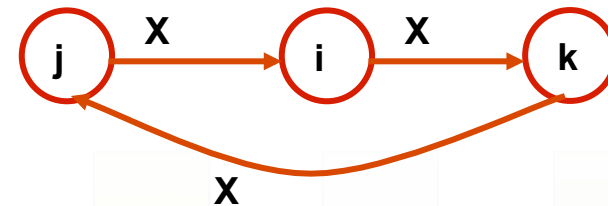
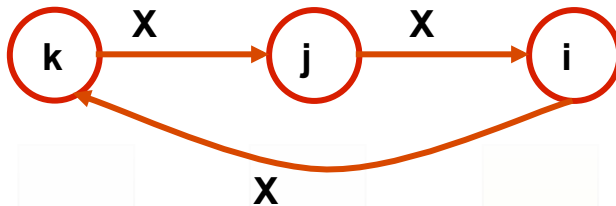
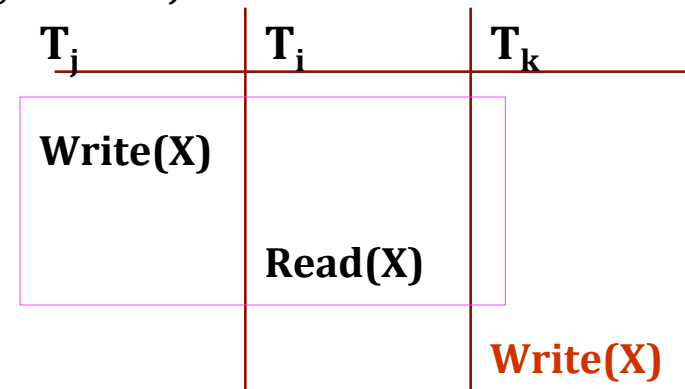
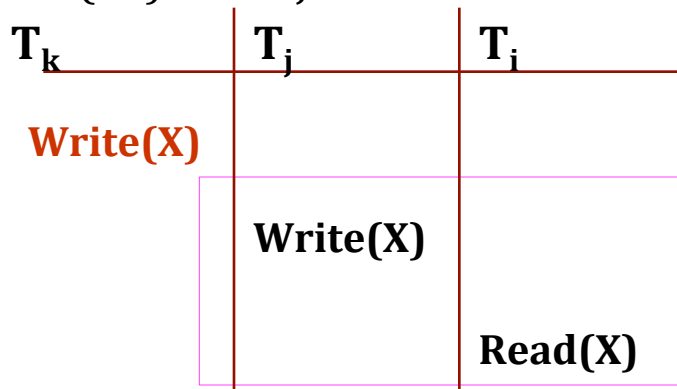
- Vẽ đồ thị phức (*PolyGraph*) cho S , ký hiệu $G(S)$ với
 - Đỉnh là các giao tác T_i (bao gồm cả T_b và T_f)
 - Cung:
 - (1) Nếu giá trị mà $ri(X)$ đọc được là do T_j ghi ($ri(X)$ có gốc là T_j) thì vẽ cung đi từ T_j đến T_i



- (2) Với mỗi $w_j(X) \dots ri(X)$, xét $w_k(X)$ khác T_b sao cho T_k không chen vào giữa T_j và T_i

Kiểm tra View-Serializability (tt)

- (2a) Nếu $T_j \neq T_b$ và $T_i \neq T_f$ thì vẽ cung $T_k \rightarrow T_j$ và $T_i \rightarrow T_k$



T_k có thể nằm trước T_j hoặc sau T_i

Kiểm tra View-Serializability (tt)

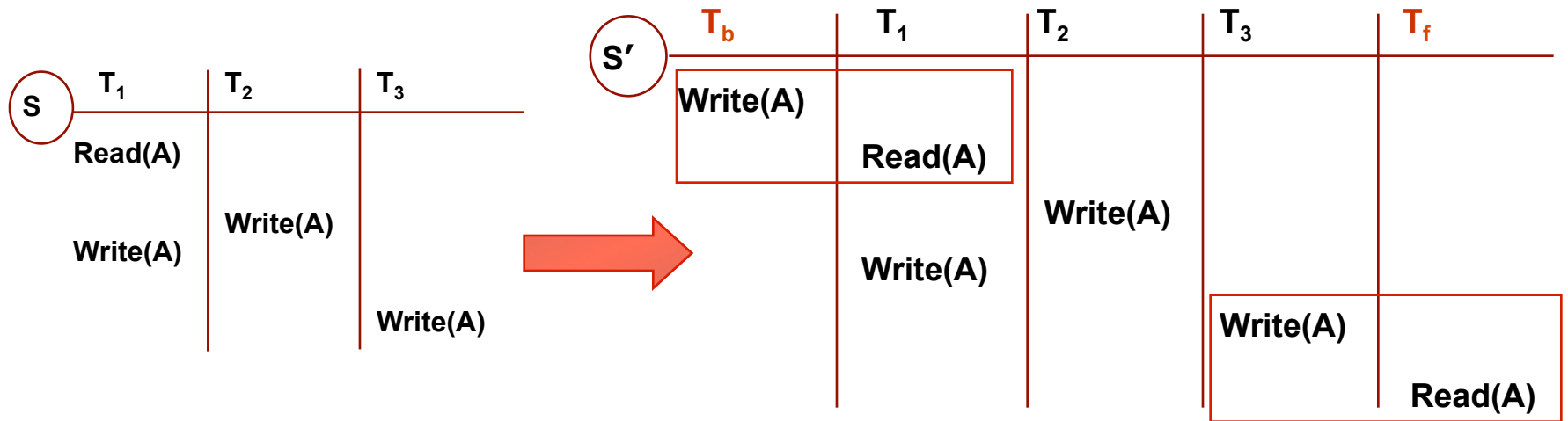
- (2b) Nếu $T_j = T_b$ thì vẽ cung $T_i \rightarrow T_k$
- (2c) Nếu $T_i = T_f$ thì vẽ cung $T_k \rightarrow T_j$

$T_j = T_b$	T_i	T_k
Write(X)	Read(X)	Write(X)

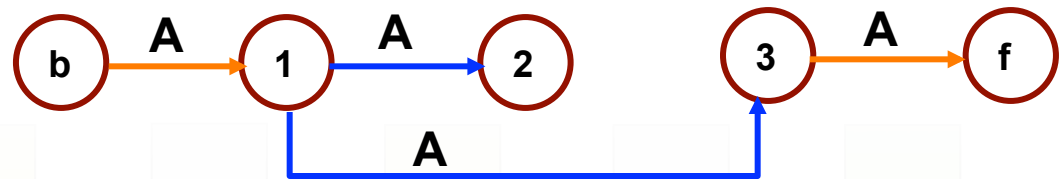
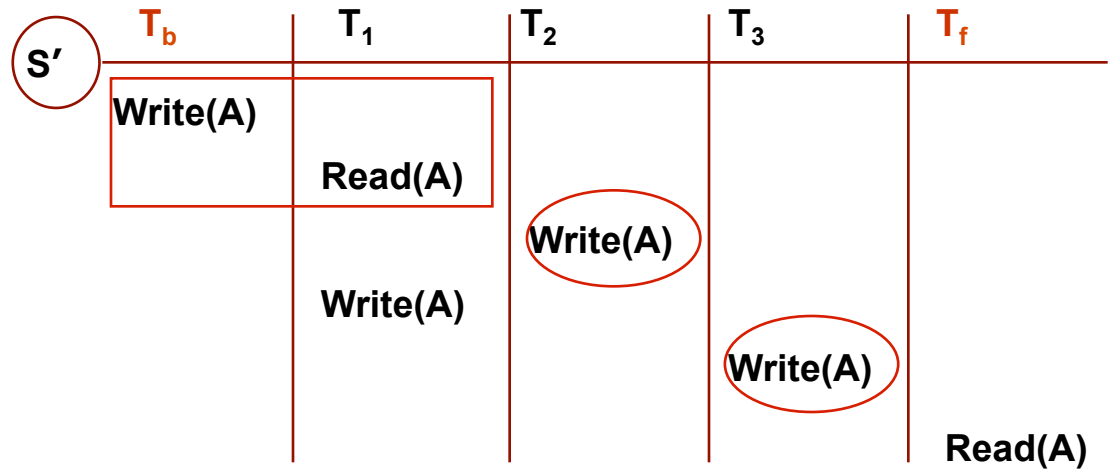
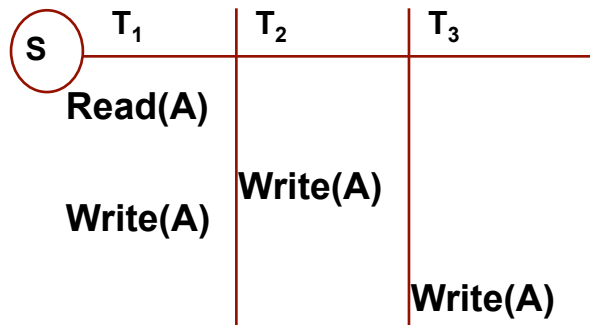
T_k	T_j	$T_i = T_f$
Write(X)	Write(X)	Read(X)



Ví dụ



Ví dụ

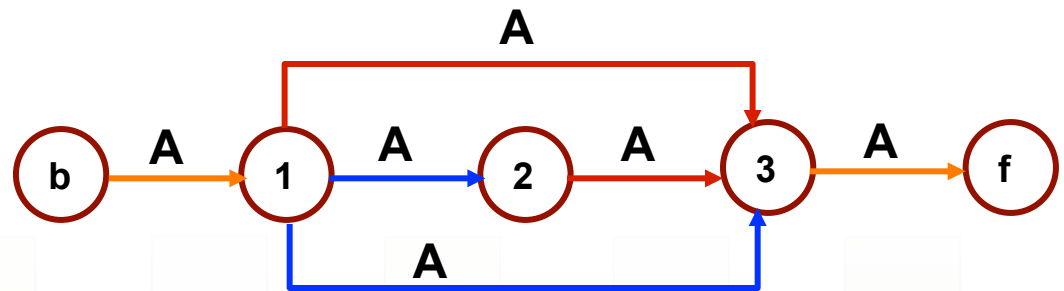


Ví dụ (tt)

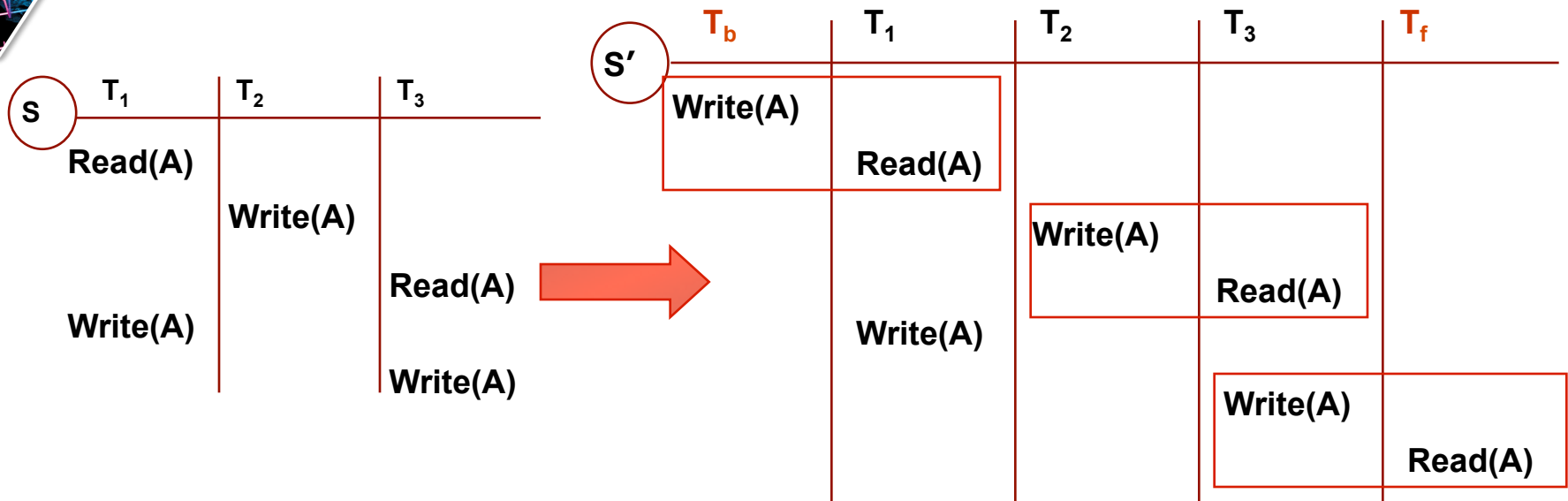
S	T ₁	T ₂	T ₃
	Read(A)		
	Write(A)	Write(A)	
			Write(A)

S'	T _b	T ₁	T ₂	T ₃	T _f
	Write(A)				
		Read(A)			
		Write(A)	Write(A)		
				Write(A)	
					Read(A)

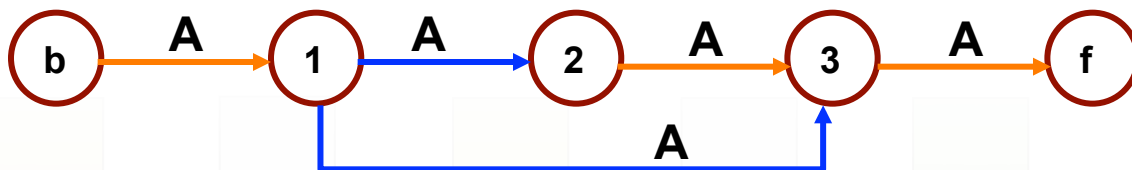
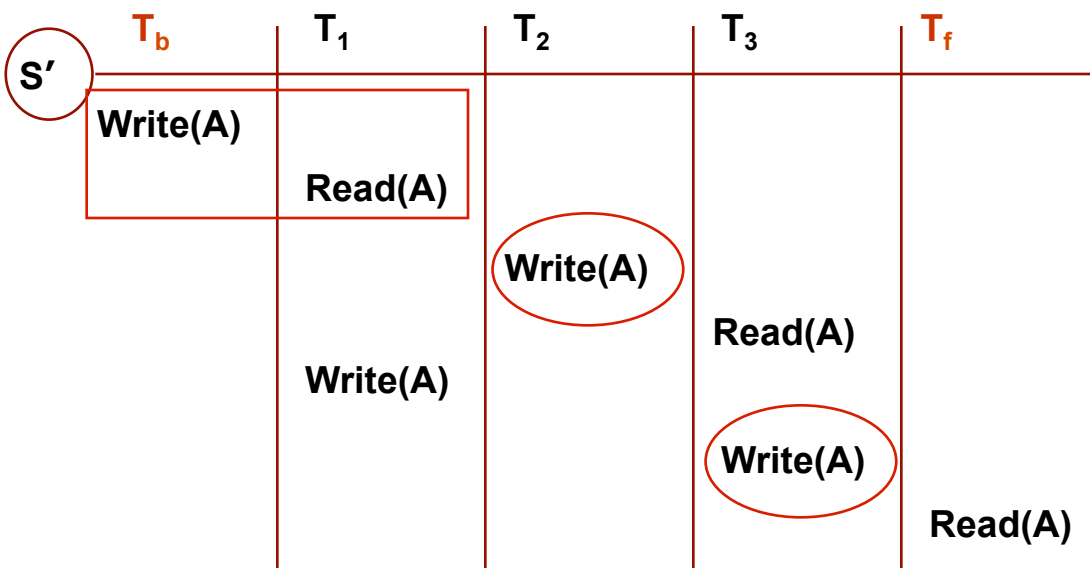
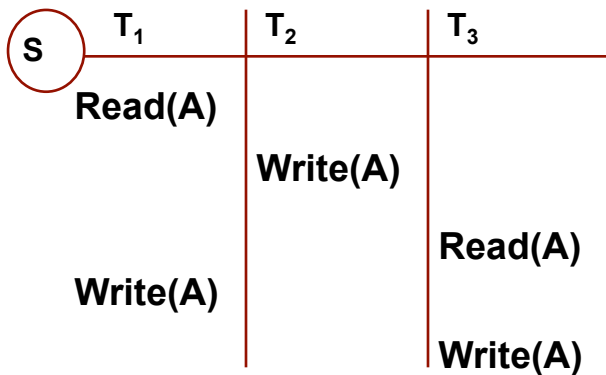
- * $G(S)$ không có chu trình
- * S view-serializable theo thứ tự T_b, T_1, T_2, T_3, T_f



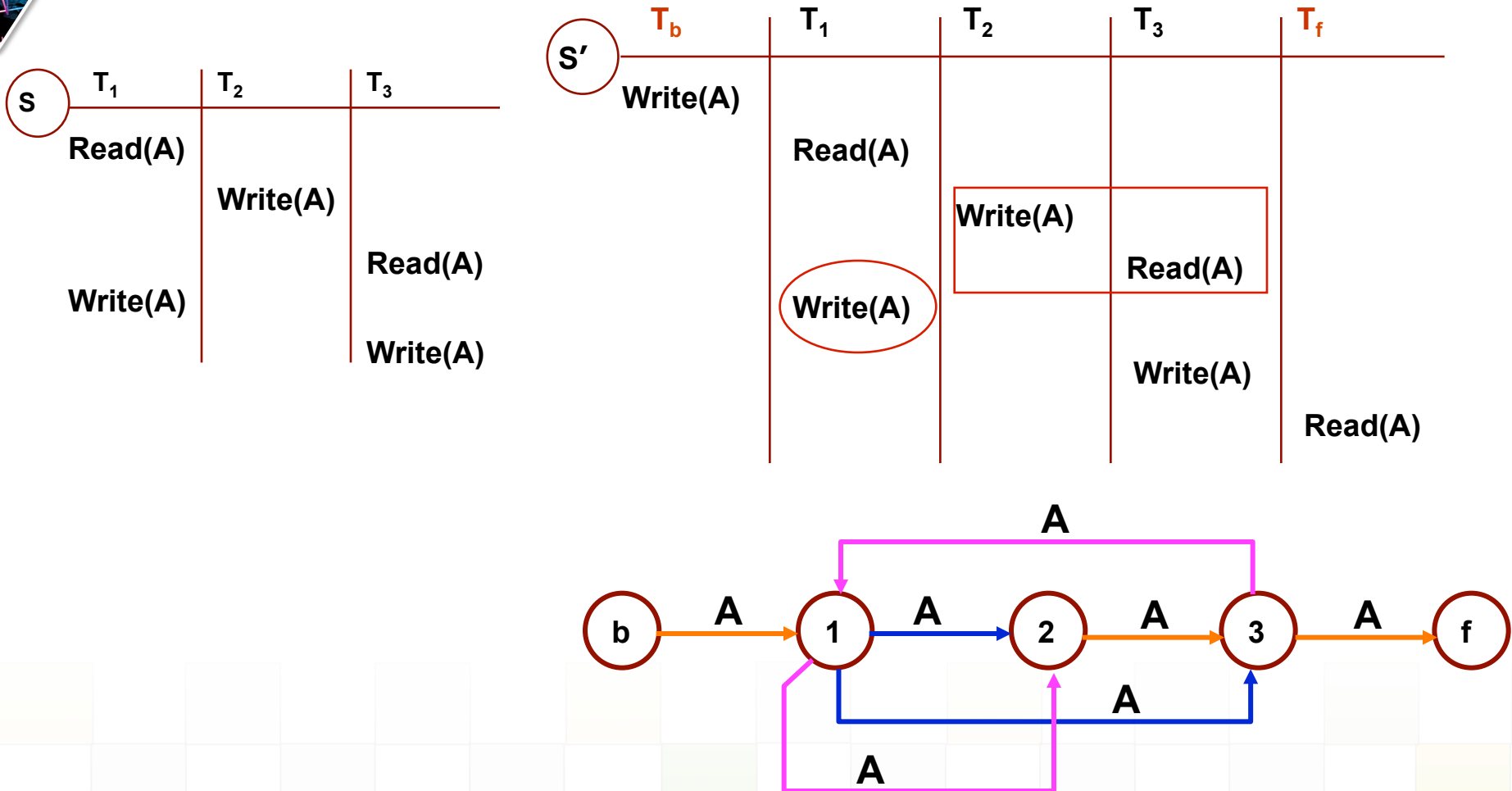
Ví dụ (tt)



Ví dụ (tt)



Ví dụ (tt)

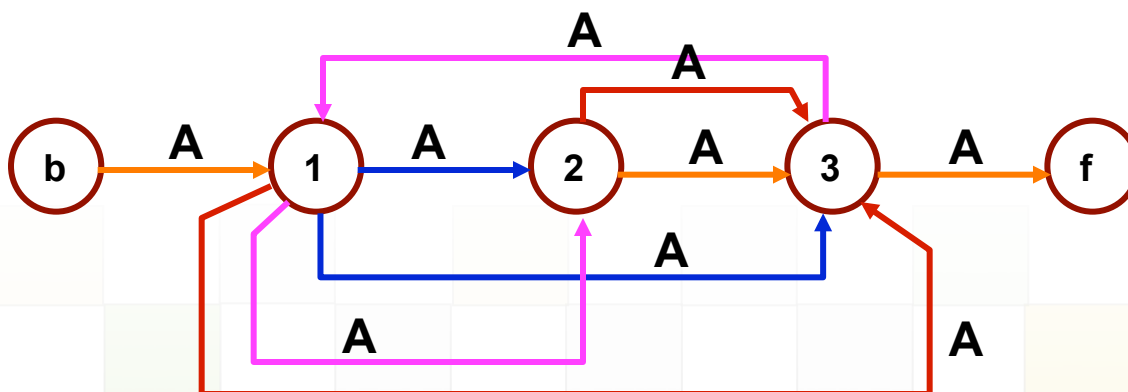


Chọn 1 trong 2 cung sao cho không có chu trình

Ví dụ (tt)

S	T ₁	T ₂	T ₃
	Read(A)		
		Write(A)	
	Write(A)		Read(A)
			Write(A)

S'	T _b	T ₁	T ₂	T ₃	T _f
	Write(A)				
		Read(A)			
		Write(A)	Write(A)	Read(A)	
				Write(A)	Read(A)

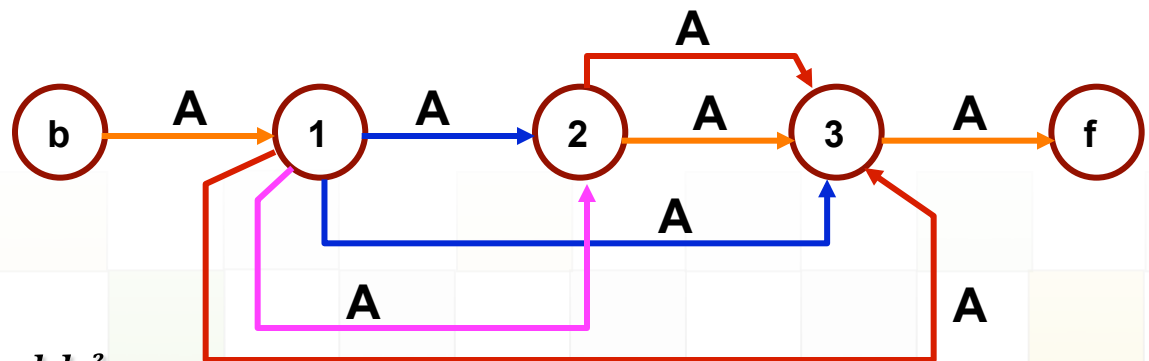


* $G(S)$ có chu trình

Ví dụ (tt)

S	T ₁	T ₂	T ₃
	Read(A)		
		Write(A)	
	Write(A)		Read(A)
			Write(A)

S'	T _b	T ₁	T ₂	T ₃	T _f
	Write(A)				
		Read(A)			
		Write(A)	Write(A)	Read(A)	
				Write(A)	Read(A)



* $G(S)$ không có chu trình sau khi bỏ cung

* S view-serializable



Bài tập View-Serializability

■ Cho lịch S:

1. S: $r_2(B)$ $w_2(A)$ $r_1(A)$ $r_3(A)$ $w_1(B)$ $w_2(B)$ $w_3(B)$
2. S: $w_1(A)$ $r_3(A)$ $r_2(A)$ $w_2(A)$ $r_1(A)$ $w_3(A)$
3. S: $r_2(A)$ $r_1(A)$ $w_1(C)$ $r_3(C)$ $w_1(B)$ $r_4(B)$ $w_3(A)$ $r_4(C)$ $w_2(D)$ $r_2(B)$ $w_4(A)$ $w_4(B)$
4. S: $w_1(A)$ $r_2(A)$ $w_2(A)$ $r_1(A)$
5. S: $r_1(A)$ $r_3(D)$ $w_1(B)$ $r_2(B)$ $w_3(B)$ $r_4(B)$ $w_2(C)$ $r_5(C)$ $w_4(E)$ $r_5(E)$ $w_5(B)$
6. S: $w_1(A)$ $r_2(A)$ $w_3(A)$ $r_4(A)$ $w_5(A)$ $r_6(A)$
7. S: $r_1(X)$ $r_2(X)$ $w_1(X)$ $w_2(X)$

■ Yêu cầu:

- Vẽ $G(S)$
- S có view-serializable hay không ?



TÓM TẮT CHƯƠNG 2

- Một số tình huống về tranh chấp
- Khái niệm giao tác
- Tính chất ACID của giao tác
- Lịch thao tác:
 - Lịch tuần tự
 - Lịch đồng thời
 - Lịch Khả tuần tự
 - Lịch khả tuần tự xung đột (Conflict Serializability)
 - Kiểm tra khả tuần tự xung đột bằng đồ thị trình tự (Precedence graph)
 - Lịch khả tuần tự view (View Serializability)
 - Kiểm tra khả tuần tự view bằng đồ thị phức (Poly graph)

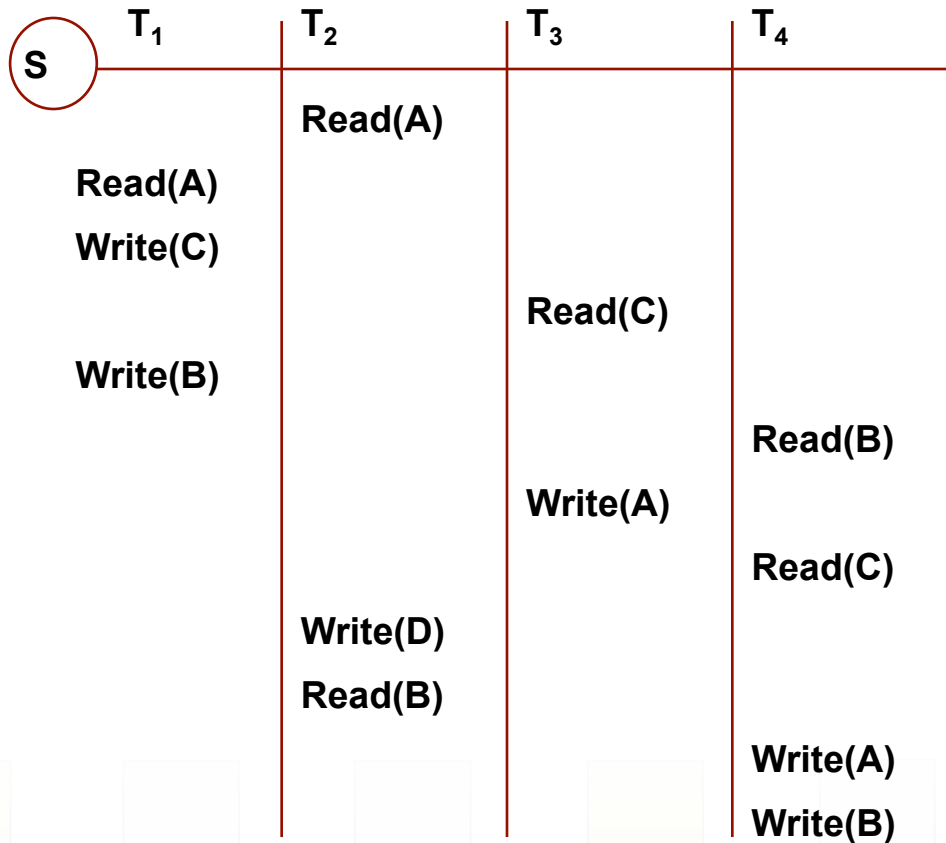
Bài tập

S	T ₁	T ₂	T ₃
		Read(B) Write(A)	
Read(A)			
Write(B)			Read(A)
		Write(B)	
			Write(B)

* *Vẽ $G(S)$*

* *S có view-serializable?*

Bài tập (tt)



* *Vẽ $G(S)$*

* *S có view-serializable?*



TÀI LIỆU THAM KHẢO

- [1] *Database Management Systems*, 3rd Edition, Raghu Ramakrishnan and Johannes Gehrke
- [2] *Fundamentals of Database Systems*, 4th Edition, Elmasri Navathe
- [3] *Database System Concepts*, 4th Edition, Silberschatz–Korth–Sudarshan
- [4] *Database Systems Implementation*, Hector Garcia-Molina, D. Ullman, Jennifer D. Widom
- [5] *Database systems: the complete book*, Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom, Pearson Prentice Hall, 2009



TÀI LIỆU THAM KHẢO

- <http://infolab.stanford.edu/~ullman/dscb/vs-old.pdf>
- http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed-english/Ch17_CC-95.pdf
- <http://djitz.com/neu-mscs/how-to-check-for-view-serializable-and-conflict-serializable/>
- <http://inst.eecs.berkeley.edu/~cs186/fa05/lecs/18cc-6up.pdf>
- http://folk.uio.no/inf212/handouts/uke19_2.pdf