CANTHO UNIVERSITY

TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
BỘ MÔN CÔNG NGHỆ THÔNG TIN

QUẢN TRỊ DỮ LIỆU - CT467

Chương 3: GIAO DỊCH (Transaction)

Biên soạn:

Ths. Nguyễn Thị Kim Yến

Ntkyen@ctu.edu.vn

CANTHO UNIVERSITY

5.2 Khả tuần tự view (View Serializable)

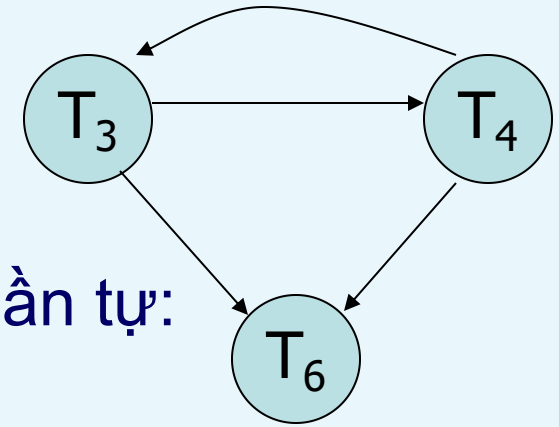
- Một lịch trình S được gọi là khả tuần tự view nếu tồn tại một lịch trình tuần tự S' được tạo ra từ các GD của S sao cho S và S' đọc và ghi những giá trị giống nhau
- Một lịch trình S gọi là khả tuần tự view nếu nó tương đương view (view equivalent) với lịch trình tuần tự
- **Tuần tự xung đột** ⟹ **Tuần tự view**

  ⟸ (Điều ngược lại không đúng)

5.2 Khả tuần tự view (tt)

- LT khả tuần tự xung đột là khả tuần tự view
- LT khả tuần tự view có thể không khả tuần tự xung đột ➔ do các write mù (blind write)
- Write mù: GD chỉ đọc không ghi hoặc ghi không đọc
- Xét LT $S_6 = R_3(Q)\ W_4(Q)\ W_3(Q)\ W_6(Q)$

  ==> Không khả tuần tự xung đột



- Tuy nhiên, LT này tương đương 2 LT tuần tự:

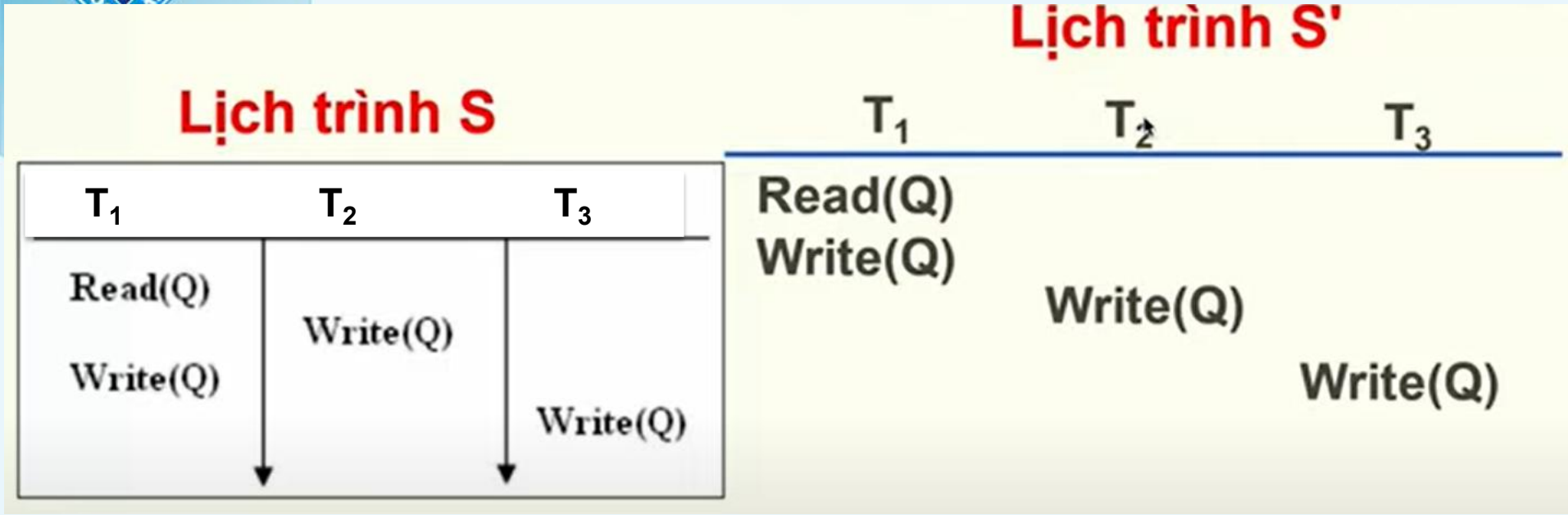
$T_4 \rightarrow T_3 \rightarrow T_6$ và $T_3 \rightarrow T_4 \rightarrow T_6$

5.2 Khả tuần tự view (tt)

❖ S và S' tương đương view nếu thỏa 3 điều kiện sau:

- **Đọc khởi đầu:** Nếu GD T_i đọc giá trị khởi đầu của Q trong S thì trong S', T_i cũng được đọc giá trị khởi đầu của Q.
- **Đọc được cập nhật:** Nếu GD T_i thực hiện *Read(Q)* trong LT S và giá trị đó được ghi bởi GD T_j thì trong S', GD T_i cũng đọc giá trị của Q do GD T_j ghi.
- **Ghi cuối cùng:** Nếu T_i thực hiện việc ghi cuối cùng trong S thì trong S', T_i cũng thực hiện ghi Q cuối cùng

✓ S và S' tương đương view

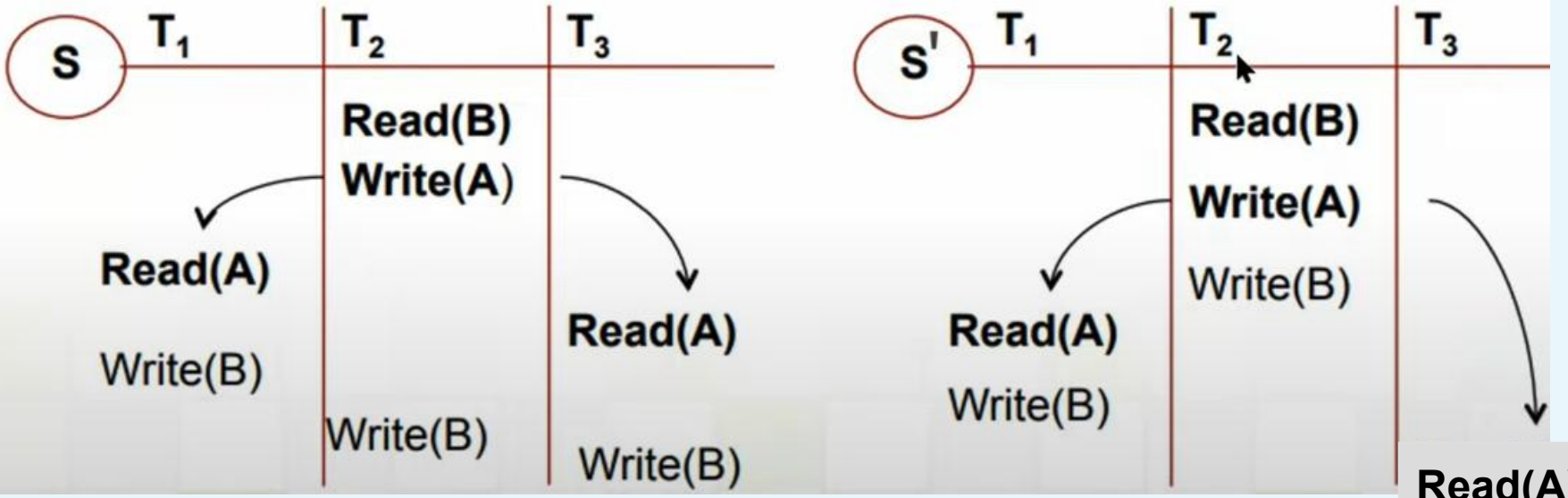


1. **Đọc khởi đầu:** Nếu $Read_1(Q)$ trong S, thì trong S' $Read_1(Q)$
2. **Đọc được cập nhật:** Nếu $Write_2(Q) => Read_1(Q)$ trong S, thì trong S' $Write_2(Q) => Read_1(Q)$
3. **Ghi cuối cùng:** Nếu $Write_3(Q)$ trong S, thì trong S' $Write_3(Q)$

Ví dụ: Tính khả tuần tự view

- 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

➢ Kiểm tra tính khả tuần tự view

Đồ thị trình tự gán nhãn: 1 đồ thị có hướng, trong đó mỗi cung sẽ có một nhãn, là một con số nguyên. Cho một LT S:

1. Xây dựng LT S' từ S bằng cách thêm vào LT S hai GD giả:

   – Thêm 1 chỉ thị đầu T_b vào S sao cho T_b thực hiện **ghi tất cả** các đơn vị dữ liệu trong S, tức là $\{T_b: Write(Q)\}$

   S= $w_b(Q).....w_1(Q).....w_2(Q)....$

   – Thêm 1 chỉ thị cuối T_f vào S sao cho T_f thực hiện **đọc tất cả** các đơn vị dữ liệu S, tức là $\{T_f: Read(Q)\}$

   S= $w_b(Q).....w_1(Q).....w_2(Q).....r_f(Q)$

➢ Kiểm tra tính khả tuần tự view (tt)

Sau đó, vẽ đồ thị tự gán nhãn cho S', ký hiệu **LP(S')**

2. Đỉnh là các giao tác T_i bao gồm cả T_b và T_f

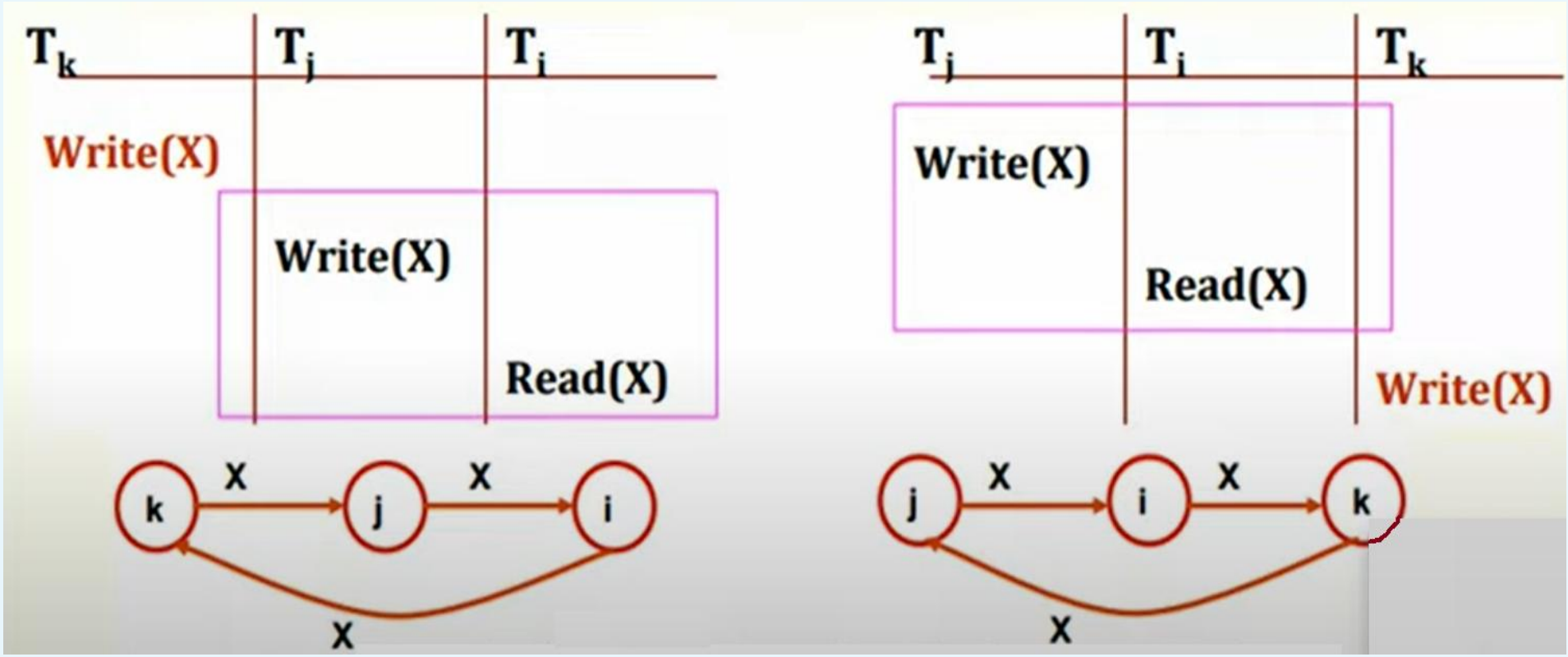
3. Cung:

– (1) $r_i(Q)$ đọc Q ghi bởi T_j ($r_i(Q)$ có gốc là T_j) thì vẽ cung đi từ T_j đến T_i

– (2) Với mỗi $w_j(Q)....r_i(Q)$, xét $w_k(Q)$ khác T_b sau cho T_k không chèn vào giữa T_j và T_i

➢ Kiểm tra tính khả tuần tự view (tt)

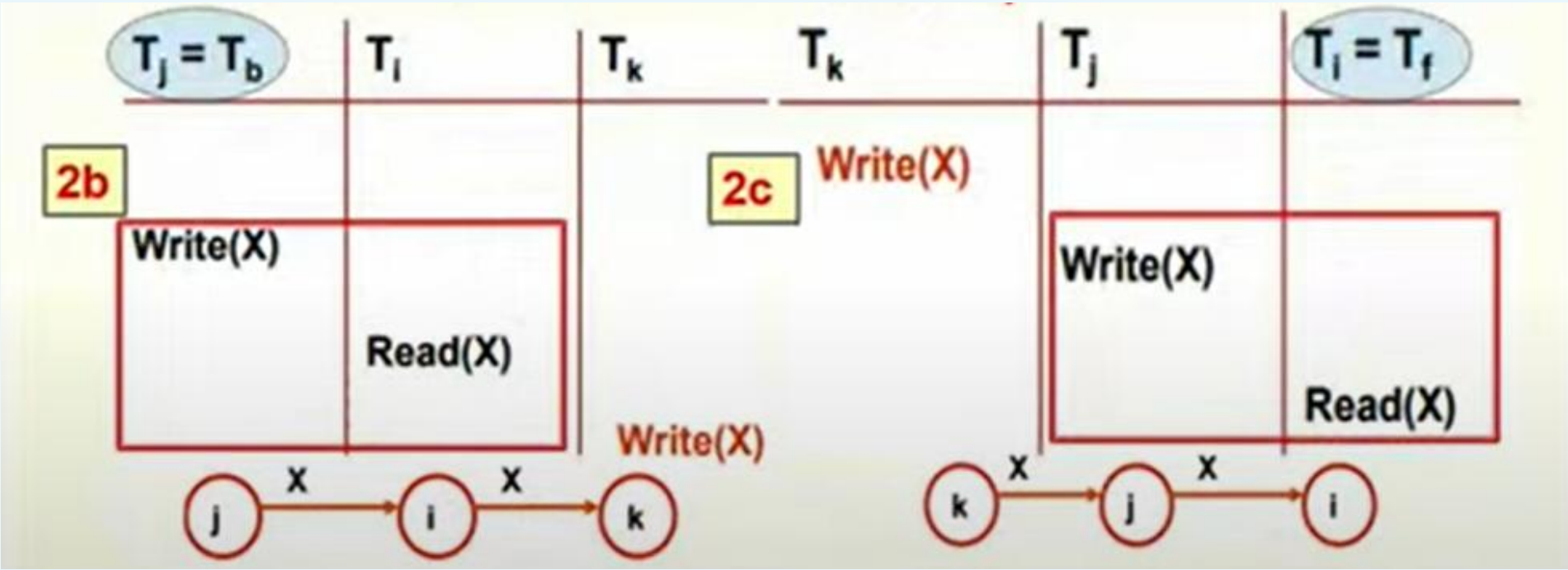
- (2a) Nếu $T_j \neq T_b$ và $T_i \neq T_f$ thì vẽ cung $T_k \rightarrow T_j$ và $T_i \rightarrow T_k$



T_k có thể nằm trước T_j hoặc sau T_i

➢ Kiểm tra tính khả tuần tự view (tt)

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



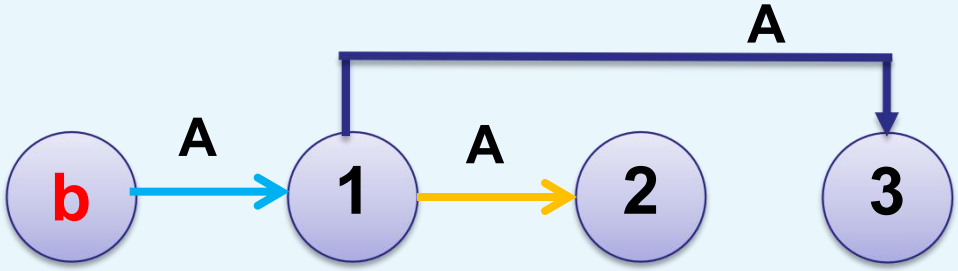
➢ Ví dụ 1: LT S có khả tuần tự view?

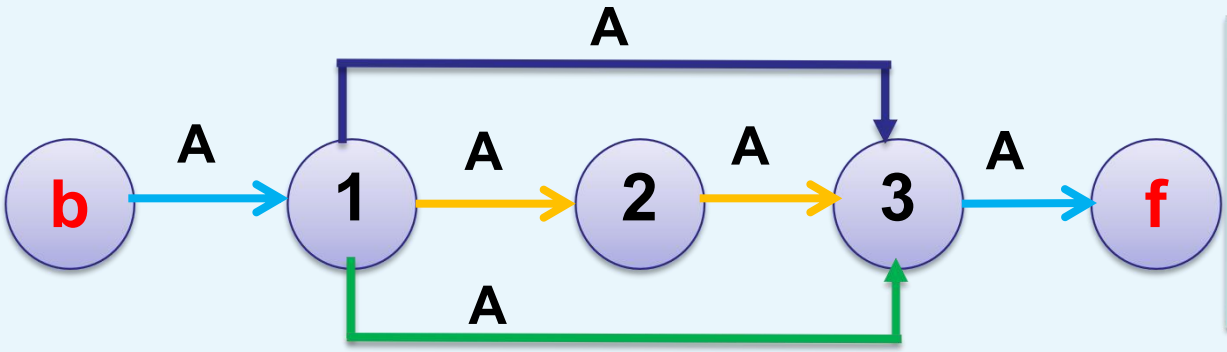
S

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)





- LP(S) không có chu trình
- S tuần tự view theo thứ tự T_b, T_1, T_2, T_3, T_f

VÍ DỤ 2

- Cho lịch trình **S = $W_1(A)$ $R_3(A)$ $R_2(A)$ $W_2(A)$ $R_1(A)$ $W_3(A)$**
- Vẽ đồ thị G(S)
- S có khả tuần tự view không?

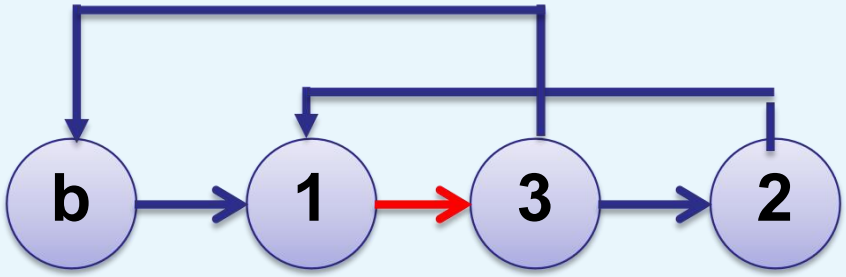
S' =

T_b	T_1	T_2	T_3	T_f
W(A)				
	W(A)			
			R(A)	
		R(A)		
		W(A)		
	R(A)			
			W(A)	
				R(A)

Ví dụ 2 (tt)

T_b	T_1	T_2	T_3	T_f
W(A)				
	W(A)			
			R(A)	
		R(A)		
		W(A)		
	R(A)			
			W(A)	
				R(A)

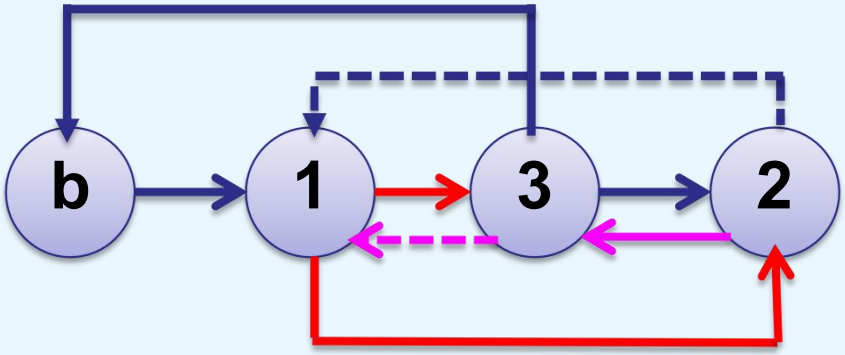
- Xét cặp: $W_1(A), R_3(A)$
- Với Write bên ngoài:
  - $W_1(A)$
  - $W_2(A)$



Ví dụ 2 (tt)

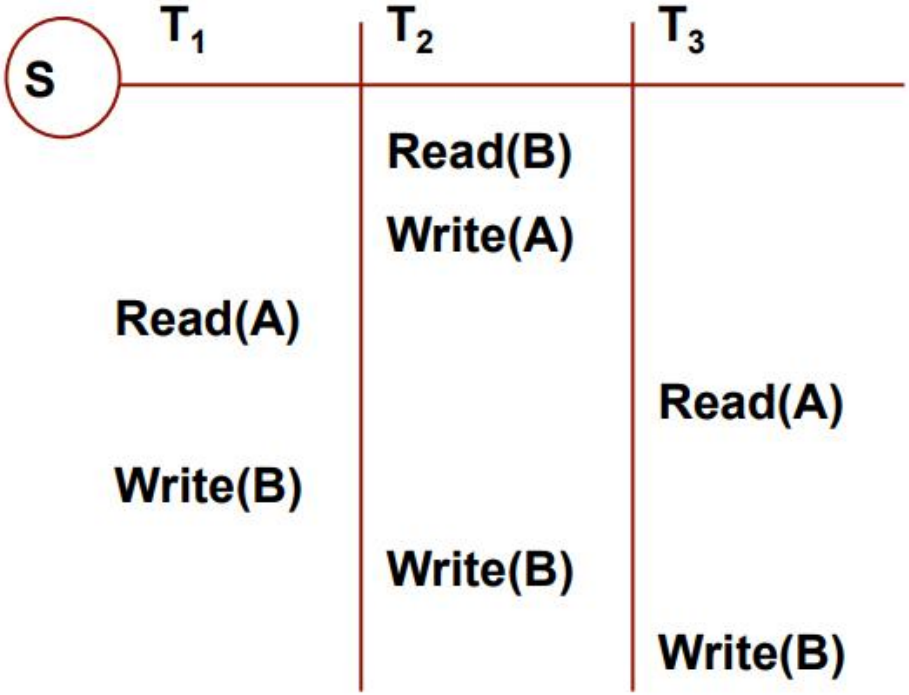
T_b	T_1	T_2	T_3	T_f
W(A)				
	W(A)			
			R(A)	
		R(A)		
		W(A)		
	R(A)			
			W(A)	
				R(A)

Chọn 1 trong 2 cung sau cho không có chu trình



G(S) có chu trình ==> S không khả tuần tự view

VÍ DỤ 3



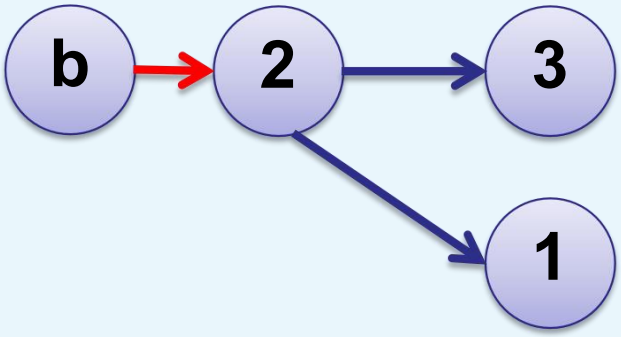
- *Vẽ G(S)*
- *S có view-serializable?*

Ví dụ 3 (tt)

T_b	T_1	T_2	T_3	T_f
W(A)				
W(B)				
		R(B)		
		W(A)		
	R(A)			
			R(A)	
	W(B)			
		W(B)		
			W(B)	
				R(B)
				R(A)

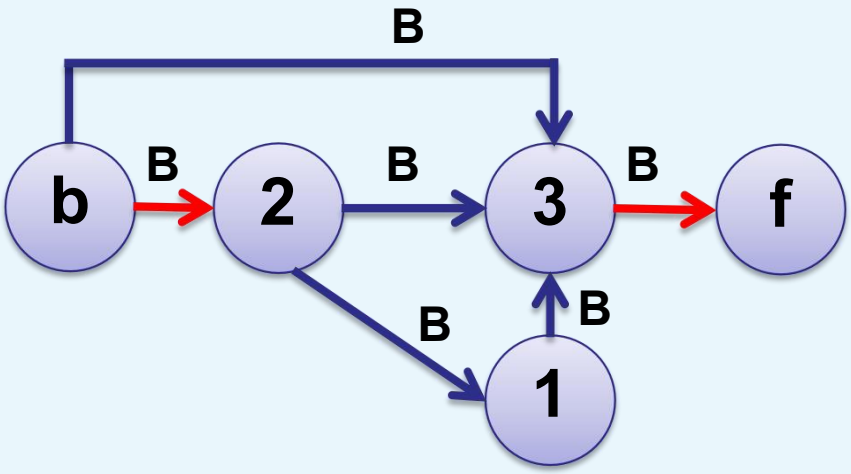
Ví dụ 3 (dữ liệu B)

T_b	T_1	T_2	T_3	T_f
W(A)				
W(B)				
		R(B)		
		W(A)		
	R(A)			
			R(A)	
	W(B)			
		W(B)		
			W(B)	
				R(B)
				R(A)



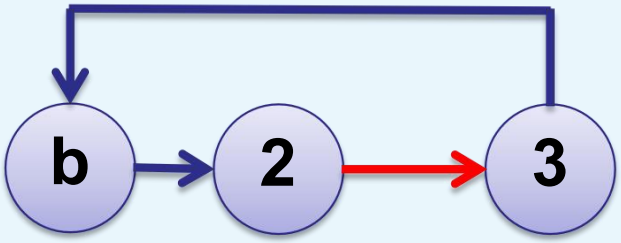
Ví dụ 3 (dữ liệu B)

T_b	T_1	T_2	T_3	T_f
W(A)				
W(B)				
		R(B)		
		W(A)		
	R(A)			
			R(A)	
	W(B)			
		W(B)		
			W(B)	
				R(B)
				R(A)



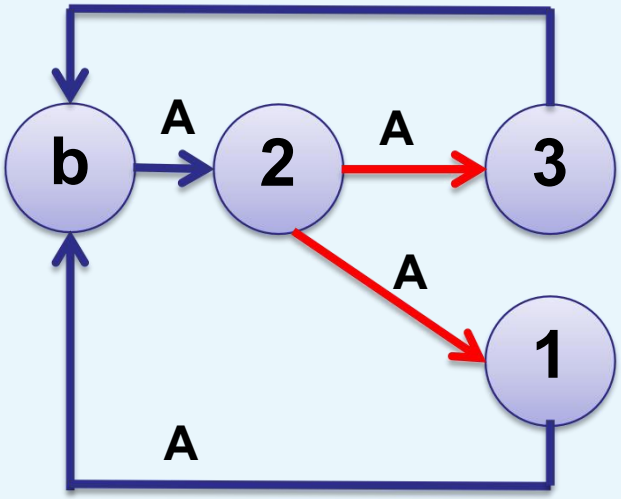
Ví dụ 3 (dữ liệu A)

T_b	T_1	T_2	T_3	T_f
W(A)				
W(B)				
		R(B)		
		W(A)		
	R(A)			
			R(A)	
	W(B)			
		W(B)		
			W(B)	
				R(B)
				R(A)

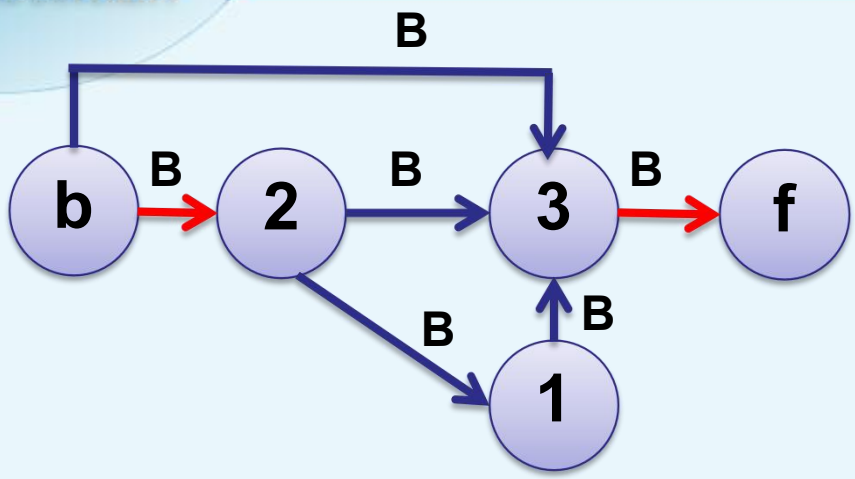


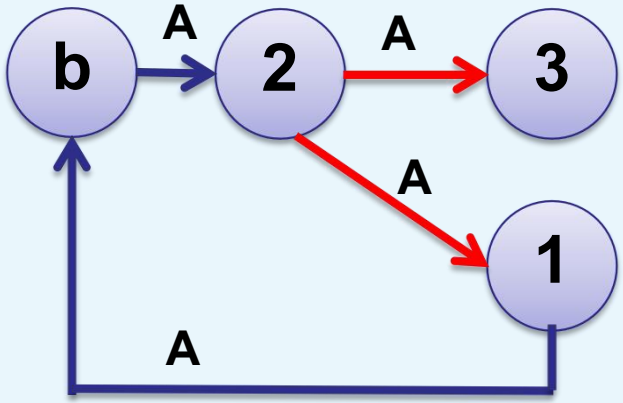
Ví dụ 3 (dữ liệu A)

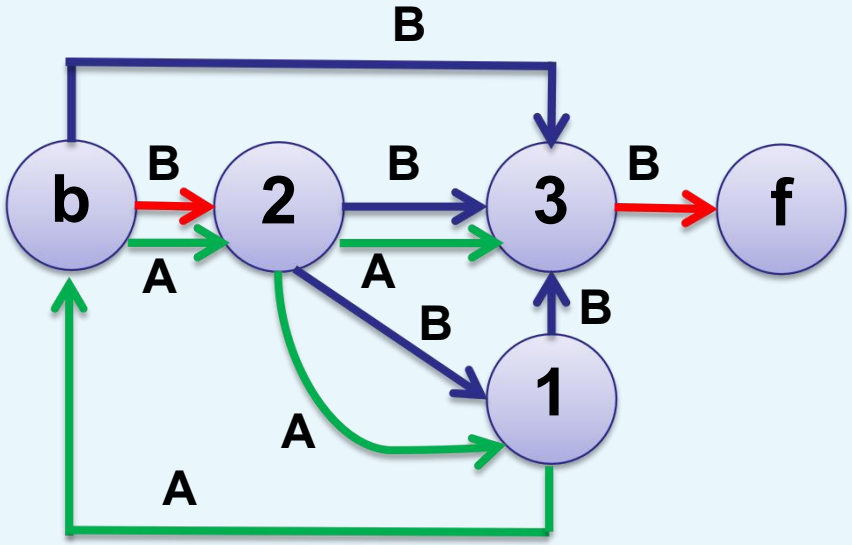
T_b	T_1	T_2	T_3	T_f
W(A)				
W(B)				
		R(B)		
		W(A)		
	R(A)			
			R(A)	
	W(B)			
		W(B)		
			W(B)	
				R(B)
				R(A)



Ví dụ 3 (tt)
Kết hợp A&B







G(S) không có chu trình

=> S khả tuần tự view

6. Tính khả phục hồi (Recoverability)

- Trong một hệ thống quản lý GD, nếu một GD T_i thất bại:
  - (1) Hủy bỏ GD này để đảm bảo tính nguyên tử của GD
  - (2) Hủy bỏ tất cả các GD phụ thuộc vào T_i
- Một số lịch trình có thể dễ dàng phục hồi, tuy nhiên một số khác thì không thể phục hồi.

CANTHO UNIVERSITY

❖ Lịch trình không khả phục hồi (nonrecoverable schedule)

- Là lịch trình mà có GD đã được bàn giao (commit) thì khi đó không thể bị cuộn lại (rollback)

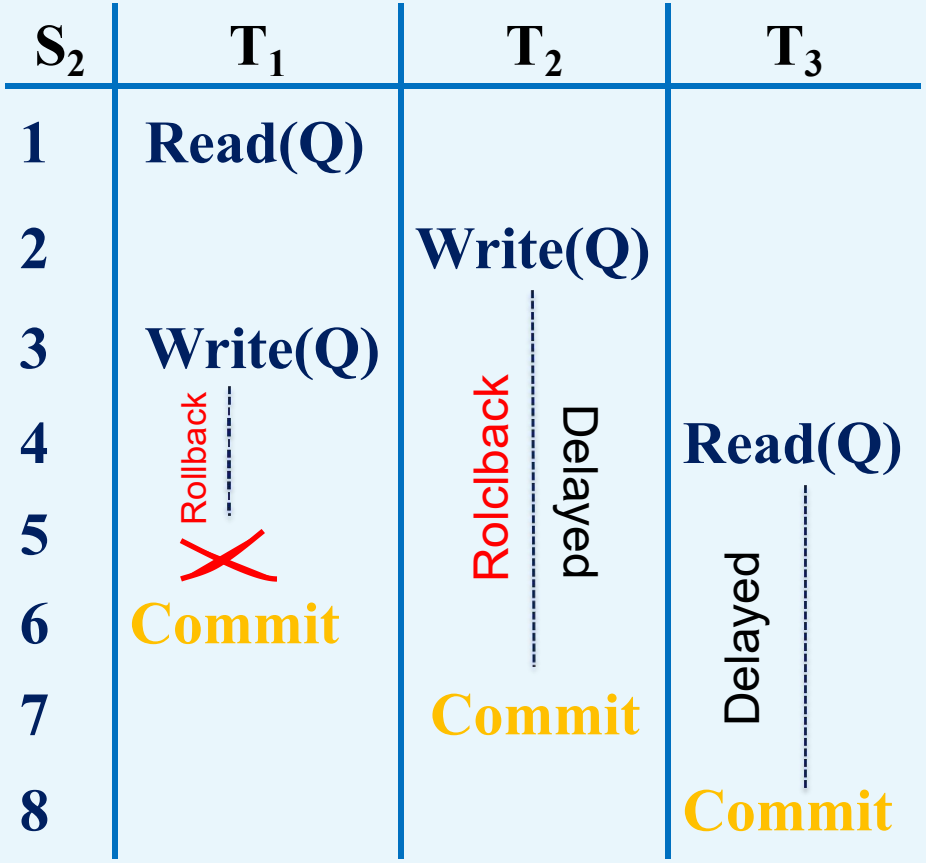
S_1	T_1	T_2
1	Read(A)	
2	Write(A)	
3	Rollback	Read(A)
4		Commit
5	Read(B)	

- Giả sử trường hợp T_1 gặp sự cố Read(B) và phải rollback T_1
- T_2 Read(A) do T_1 Write(A) (T_2 phụ thuộc T_1) ➔ phải rollback T_2
- Tuy nhiên, tại thời điểm T_1 thực hiện Read(B) thì T_2 đã bàn giao ➔ không thể rollback T_2 được

=> Lịch trình không thể phục hồi và không được phép

❖ Lịch trình khả phục hồi (recoverable schedule)

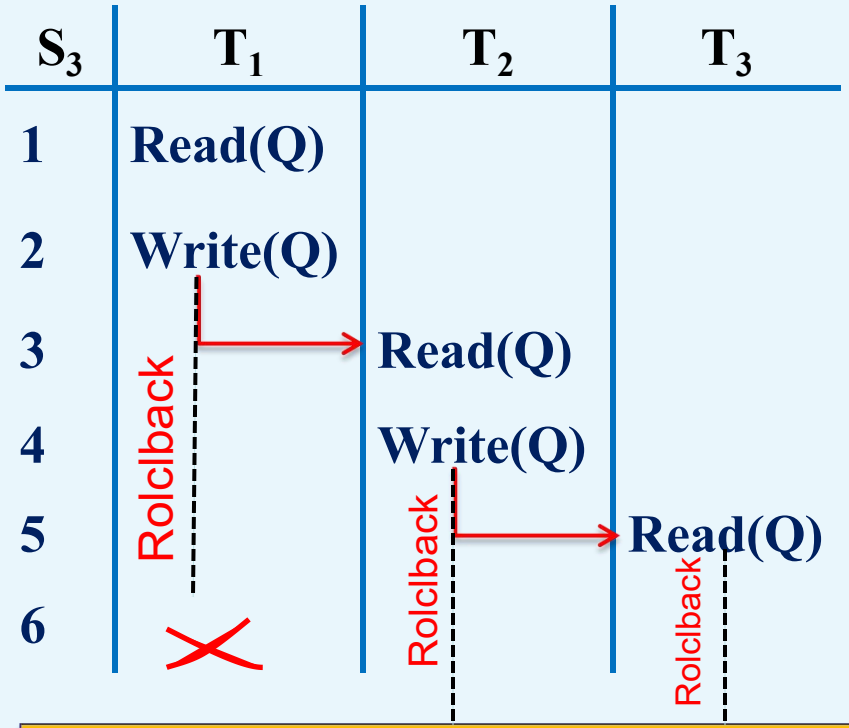
- Một lịch trình trong đó khi có một GD bị rollback thì ta có thể rollback các lịch trình phụ thuộc vào lịch trình bị cuộn lại.



Điều kiện để 1 lịch trình khả phục hồi: đối với mỗi cặp GD T_i, T_j, nếu T_j đọc giá trị dữ liệu do T_i ghi (T_j phụ thuộc T_i), thì lệnh commit của T_j phải diễn ra **sau** lệnh commit của T_i

➢ Cuộn lại hàng loạt (cascading rollback) - *cuộn ngược tầng - xếp tầng*

- **LT cuộn lại hàng loạt:** Khi một GD thất bại và bị cuộn lại dẫn đến hàng loạt các GD khác bị cuộn lại theo



- T_2 đọc do T_1 ghi và T_3 đọc do T_2 ghi ($T_3 \in T_2 \in T_1$)
- Nếu T_1 bị sự cố không thể tiếp tục, thì hệ thống phải cuộn lại T_1 sẽ dẫn đến T_2 và T_3 phải cuộn lại theo

=> Làm hao phí tài nguyên tính toán

Không cuộn lại hàng loạt (cascadeless schedule) - *không phân tầng*

- **Lịch trình cascadeless** (không phân tầng): Là một lịch trình trong đó mỗi cặp giao dịch T_i, T_j:

S_4	T_1	T_2	T_3
1	Read(A)		
2	Read(B)		
3	Write(A)		
4	Commit		
5		Read(A)	
6		Write(A)	
7		Commit	
8			Read(A)

Rolclback

- Nếu T_j đọc một hạng mục dữ liệu do T_i ghi, thì hoạt động **đọc** của T_j phải thực hiện **sau** lệnh **commit** của T_i

=> Tiết kiệm thời gian của CPU

CANTHO UNIVERSITY

✓ Lịch trình cascadeless (lịch trình không phân tầng)

❖ Ghi chú:

- Chỉ cho phép các hoạt động đọc có lệnh commit trước đó
- Tuy nhiên, nó cho phép các hoạt động ghi không commit

S_5	T_1	T_2
1	Read(A)	
2	Write(A)	
3	Delayed	Write(A)
4		✗
5	Commit	
		Read(A)
		Write(A)
		Commit

Cascadeless ⇒ Recoverable

✓ Lịch trình Strict (nghiêm ngặt)

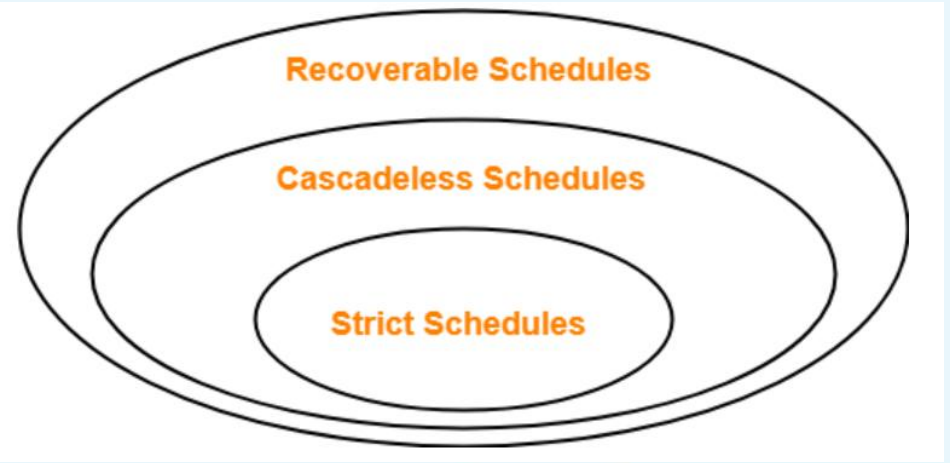
S_6	T_1	T_2	T_3
1	Read(X)		
2		Read(Y)	
3	Read(Z)		
4			Read(X)
5			Read(Y)
6	Write(X)		
7	Commit		
8			Write(Y)
9			Commit
10		Read(Y)	
11		Write(Z)	
12		Write(Y)	
13		Commit	

Lịch trình nghiêm ngặt:

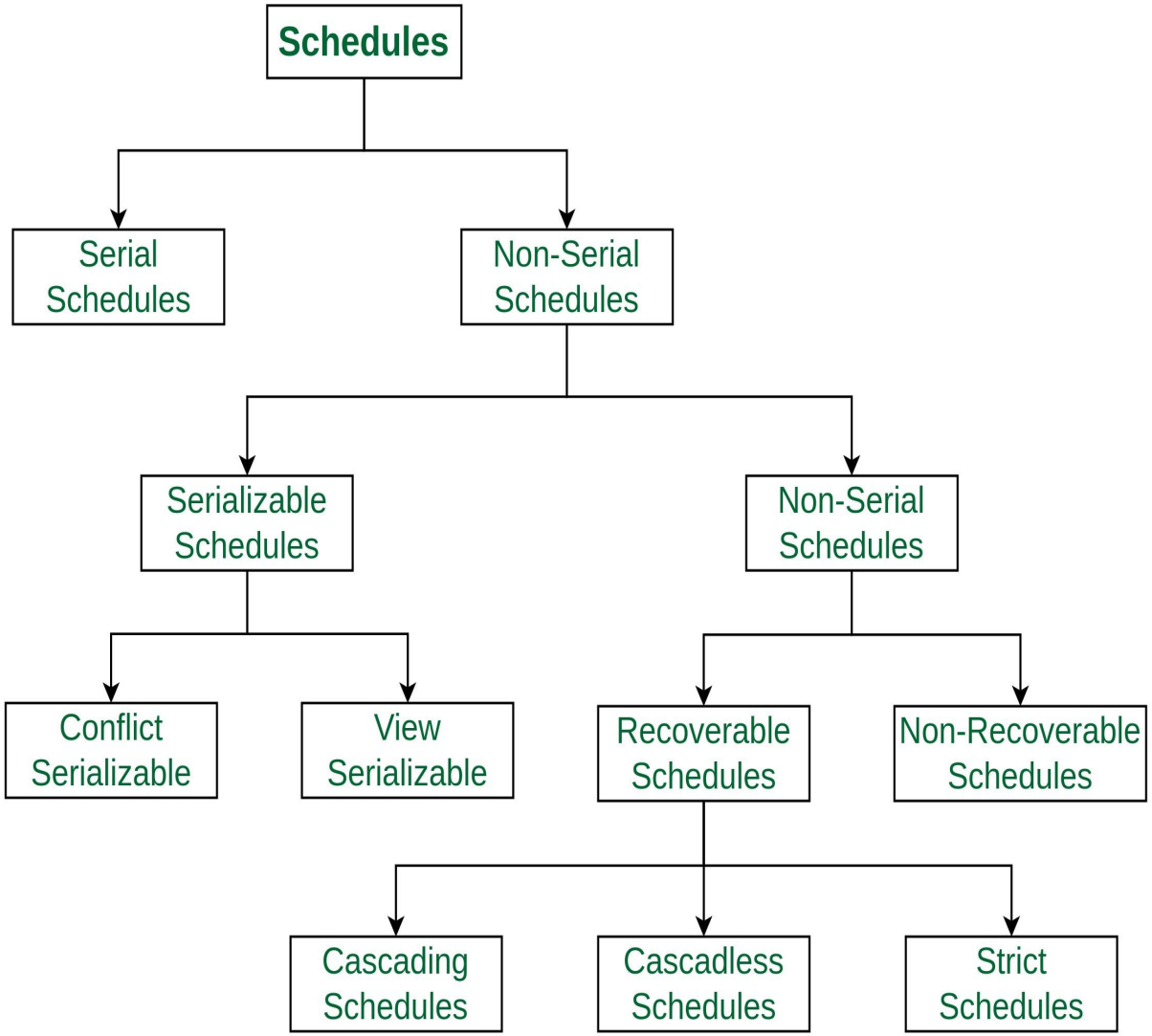
không có xung đột đọc - ghi

hoặc ghi - ghi nào phát sinh

trước lệnh commit



Types of schedules in DBMS



BÀI TẬP 1

- Xét lịch trình S (**chưa đầy đủ**) sau :

T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y)

- Với mỗi yêu cầu dưới đây, hãy chỉnh sửa S để tạo một lịch trình đầy đủ thỏa mãn các điều kiện đã cho. Nếu nó có thể hãy dùng số lượng hành động có thể nhỏ nhất (**Read, Write, Commit hay Abort**). Bạn có thể tùy ý thêm hành động ở bất kỳ chỗ nào trong lịch trình S.

1) Lịch trình cho kết quả không phân tầng (cascadeless)
2) Lịch trình cho kết quả khả phục hồi (recoverable)

1) Lịch trình cho kết quả không phân tầng (cascadeless)

2) Lịch trình cho kết quả khả phục hồi (recoverable)

T1:R(X), T1:R(Y), T1:W(X), T2:R(Y), T3:W(Y), T1:W(X), T2:R(Y)

T_1	T_2	T_3
R(X)		
R(Y)		
W(X)		
	R(Y)	
		W(Y)
W(X)		
	Commit	
	R(Y)	

T_1	T_2	T_3
R(X)		
R(Y)		
W(X)		
	R(Y)	
		W(Y)
		Commit
W(X)		
	R(Y)	
	Commit	

❖ Các mức độ cô lập GD (Isolation Levels)

- Các mức độ cô lập của SQL-92 (xếp theo độ tăng dần của mức độ cô lập):
  - Read uncommitted: đây là mức thấp nhất, cho phép đọc cả các dữ liệu chưa commit - đọc bẩn (dirty read)
  - Read commited: chỉ cho phép đọc các dữ liệu đã commit, nhưng không tránh được các mẫu tin ma (phantom row)

CANTHO UNIVERSITY

✓ Mẫu tin ma (bóng ma)

❖ Xét 2 giao dịch T_1 và T_2 được xử lý đồng thời:

- T_1 xử lý trên toàn bộ tập dữ liệu A
- Khi T_1 đang xử lý T_2 thêm hay xóa 1 số phần tử trong tập A

	T1	T2
t1	Read(A)	
t2	Xử lý 1 trên A	
t3		Thêm ai vào A
t4	Xử lý 2 trên A	
t5		Xoá aj khỏi A
t6	Xử lý 3 trên A	

❖ Các mức độ cô lập GD (tt)

- Repeatable read: chỉ cho phép đọc các dữ liệu đã commit và giữa các chỉ thị đọc của cùng một GD trên một hạng mục dữ liệu thì không cho phép GD khác thực hiện cập nhật trên hạng mục dữ liệu đó.
- Serializable: chế độ mặc định, các GD phải thực hiện tuần tự khi truy xuất cùng hạng mục dữ liệu. Đây là mức cao nhất, vì thế hiện tượng các mẫu tin ma không còn.

==> Tất cả mức độ cô lập trên đều không cho phép thao tác ghi bẩn (ghi bởi GD khác chưa commit hoặc hủy bỏ)

❖ Các tính năng của từng mức isolation

- Khi càng lên mức cao, đòi hỏi về tính toàn vẹn và nhất quán dữ liệu càng cao. Do đó nó càng tăng tình trạng locking và blocking trong database ➔ hiệu năng của hệ thống bị giảm đi.

Isolation level	Dirty read	Nonrepeatable read	Phantom
Read uncommitted	yes	yes	yes
Read committed	no	yes	yes
Repeatable read	no	no	yes
Serializable	no	no	no

HẾT CHƯƠNG 3