Nguyên lý Hệ quản trị CSDL

Chương 3: Giao dịch (Transaction)

Mục tiêu

Giới thiệu các nguyên tắc xử lý giao dịch trong các Hệ quản trị CSDL bao gồm các khái niệm có liên quan đến xử lý giao dịch, điều khiển sự cạnh tranh giữa các giao dịch và một số tính chất của các lịch trình

Nội dung

- Khái niệm
- Các trạng thái của một giao dịch
- Thực thi tính nguyên tử và tính bền vững
- Các thực hiện cạnh tranh (concurrent executions)
- Tính khả tuần tự (serializability)
  - Khả tuần tự xung đột (conflict serializable)
  - Khả tuần tự view (view serializable)
- Tính khả phục hồi (recoverability)

Khái niệm (1)

- Giao dịch (GD): là một đơn vị thực hiện chương trình truy xuất và có thể cập nhật nhiều hạng mục dữ liệu (hay: tập các chỉ thị tạo thành một đơn vị công việc logic).
- Thông thường, để tạo một giao dịch, các ngôn ngữ/DBMS dùng các lệnh **BEGIN TRANSACTION** và **END TRANSACTION**.
- 4 tính chất của một GD:
  - Nguyên tử (**A**tomicity):
  - Tính nhất quán (**C**onsistency):
  - Tính cô lập (**I**solation):
  - Tính bền vững (**D**urability):

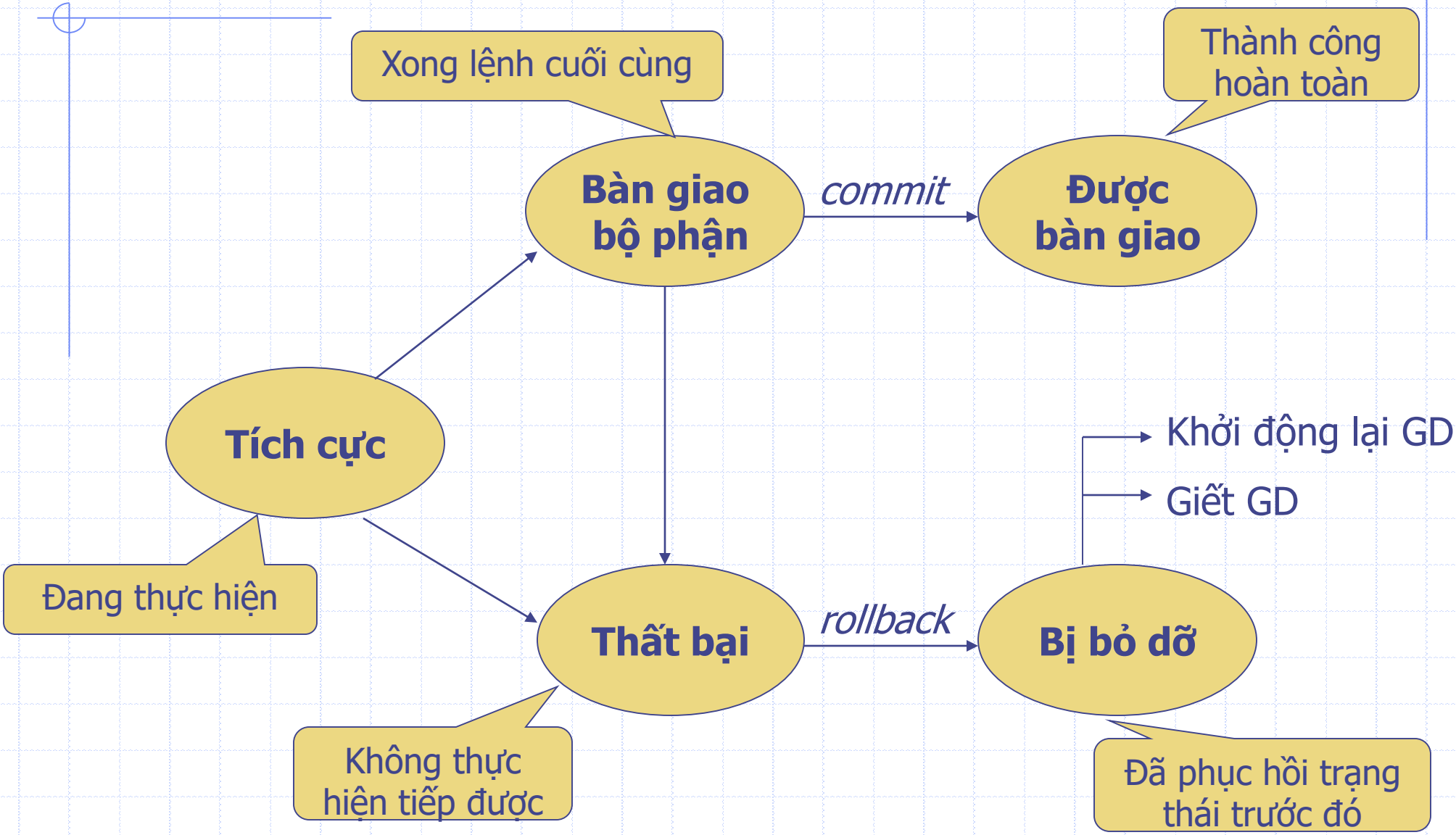
Khái niệm (2)

- Các thao tác trong GD:
  - READ(X): đọc một hạng mục X từ CSDL vào vùng nhớ của GD
  - WRITE(X): ghi hạng mục X từ vùng nhớ GD vào CSDL
- Ví dụ:

READ(A)
A = A - 50
WRITE(A)
READ(B)
B = B + 50
WRITE(B)

- Tính nguyên tử: hoặc tất cả các lệnh được thực hiện, hoặc không có lệnh nào được thực hiện.
- Tính nhất quán: tổng A và B không đổi bởi sự thực hiện của GD (*)
- Tính cô lập: các GD trong hệ thống phải được cô lập nhau, không được truy xuất các giá trị dữ liệu không nhất quán của GD khác.
- Tính bền vững: sau khi GD hoàn tất thì tất cả các cập nhật của GD phải được thể hiện trong CSDL, ngay cả trường hợp hệ thống có sự cố.

Các trạng thái của Giao dịch



Các thực hiện cạnh tranh (1)

- Trong các hệ thống xử lý giao dịch: thường cho phép các GD thực hiện đồng thời (cạnh tranh)
  => khó khăn trong việc đảm bảo tính nhất quán của DL.

T1
R(A)
A = A - 50
W(A)
R(B)
B = B + 50
W(B)

T2
R(A)
temp = A*0.1
A = A - temp
W(A)
R(B)
B = B + temp
W(B)

Các thực hiện cạnh tranh (2)

T1	T2	A	B
R(A)			
A = A - 50			
W(A)		950	
R(B)			
B = B + 50			
W(B)			2050
	R(A)		
	temp = A*0.1		
	A = A - temp		
	W(A)	**855**	
	R(B)		
	B = B + temp		
	W(B)		**2145**

Schedule 1

T1	T2	A	B
	R(A)		
	temp = A*0.1		
	A = A - temp		
	W(A)	900	
	R(B)		
	B = B + temp		
	W(B)		2100
R(A)			
A = A - 50			
W(A)		**850**	
R(B)			
B = B + 50			
W(B)			**2150**

Schedule 2

Các thực hiện cạnh tranh (3)

T1	T2	A	B
R(A)			
A = A - 50			
W(A)		950	
	R(A)		
	temp = A*0.1		
	A = A - temp		
	W(A)	**855**	
R(B)			
B = B + 50			
W(B)			2050
	R(B)		
	B = B + temp		
	W(B)		**2145**

Schedule 3

T1	T2	A	B
R(A)			
A = A – 50			
	R(A)		
	temp = A*0.1		
	A = A - temp		
	W(A)	900	
	R(B)		
W(A)		**950**	
R(B)			
B = B + 50			
W(B)			2050
	B = B + temp		
	W(B)		**2100**

Schedule 4

Các thực hiện cạnh tranh (4)

- Tuy nhiên, 2 lý do để thực hiện cạnh tranh:
  - Tăng hiệu suất sử dụng tài nguyên hệ thống.
  - Giảm thời gian đáp ứng trung bình của hệ thống.

- **Lịch trình (schedule):** dùng để mô tả trình tự thực hiện các chỉ thị của các GD trong hệ thống.
  - Bao gồm tất cả các chỉ thị của các GD.
  - Bảo tồn thứ tự của các chỉ thị trong cùng một GD.
  - Lịch trình tuần tự: là lịch trình trong đó các GD diễn ra tuần tự (n!).
  - Lịch trình cạnh tranh: là lịch trình trong đó các GD thực hiện cạnh tranh/đồng thời với nhau (> n!).

Tính khả tuần tự (Serializability)

- Để có thể hiểu được cách thức các HQTCSDL điều khiển sự cạnh tranh giữa các GD để đảm bảo tính nhất quán, trước tiên ta tìm hiểu cách xác định lịch trình nào nhất quán, lịch trình nào không.
- Rất khó để cho ta xác định tính nhất quán dựa tất cả các hoạt động của 1 GD => ta sẽ không giải thích kiểu hoạt động mà chỉ dựa vào 2 thao tác Read và Write.
- Lịch trình chỉ có 2 thao tác này được gọi là lịch trình biểu diễn dưới dạng quy ước.

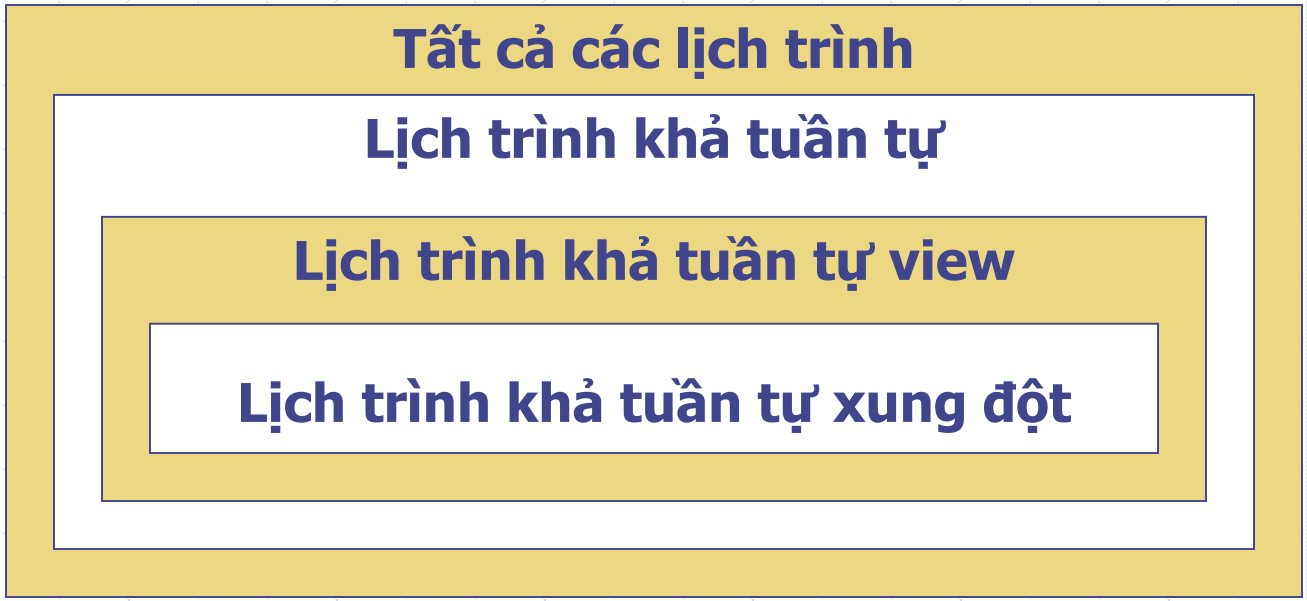
T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)

Schedule 3
(dạng quy ước)

$$S3 = r_1(A)\ w_1(A)\ r_2(A)\ w_2(A)\ r_1(B)\ w_1(B)\ r_2(B)\ w_2(B)$$

Tính khả tuần tự (2)

- Nếu các GD là nhất quán, các lịch trình tuần tự sẽ nhất quán
- Trong các hệ thống xử lý song song các GD, làm thế nào để xác định một lịch trình cạnh tranh là nhất quán (tương đương với một LT tuần tự)?
  - Khả tuần tự xung đột (conflict serializability)
  - Khả tuần tự view (view serializability)



Khả tuần tự xung đột (1)

- Một số khái niệm:
  - **Chỉ thị xung đột** (conflicting instructions): hai chỉ thị I_i và I_j của hai GD T_i và T_j được gọi là xung đột nếu như chúng truy xuất đến cùng hạng mục dữ liệu và có ít nhất 1 chỉ thị write

    (việc thể đảo chỗ hai chỉ thị có thể làm ảnh hưởng đến kết quả của lịch trình)
  - **Tương đương xung đột** (conflict equivalent): nếu một lịch trình S có thể biến đổi thành lịch trình S′ bằng cách đảo chỗ các chỉ thị không xung đột => S và S′ tương đương xung đột

    $\Rightarrow$ Hai LT tương đương xung đột sẽ cho kết quả giống nhau (cùng trạng thái cuối)
  - **Khả tuần tự xung đột** (conflict serializable): lịch trình S được gọi là khả tuần tự xung đột nếu S tương đương xung đột với một LT tuần tự.

Khả tuần tự xung đột (2)

- Ví dụ:

T1	T2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

Schedule 1 (dạng thỏa thuận)

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
W(B)	
	R(B)
	W(B)

Schedule 3 (dạng thỏa thuận)

T1	T2
R(A)	
	R(A)
	W(A)
	R(B)
W(A)	
R(B)	
W(B)	
	W(B)

Schedule 4 (dạng thỏa thuận)

Tương đương XĐ ?

Tương đương XĐ ?

Khả tuần tự xung đột (3)

- Làm thế nào để kiểm tra tính khả tuần tự XĐ?
  Dùng cách đảo chổ các chỉ thị không xung đột để kiểm tra tính khả tuần tự XĐ ➔ **Không khả thi**

- Giải pháp: Dùng Đồ thị trình tự P. Hai bước:

1. Xây dựng ĐTTT: P(S) = (V, E).

- V: tập các đỉnh, các GD tham gia vào lịch trình
- E: tập các cung, xác định bằng quy tắc: một cung $T_i \rightarrow T_j$ được thêm vào P(S) nếu tồn tại một cặp chỉ thị xung đột giữa T_i và T_j, trong đó chỉ thị xung đột của GD T_i đứng trước T_j.

2. Kiểm tra:

- Nếu P(S) không chứa chu trình => S khả tuần tự xung đột
- Ngược lại, S không khả tuần tự xung đột

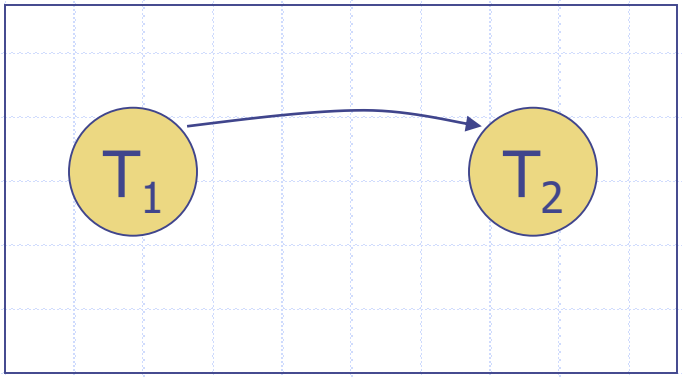
Khả tuần tự xung đột (4)

- Ví dụ: xét tính khả tuần tự xung đột của S3 và S4:

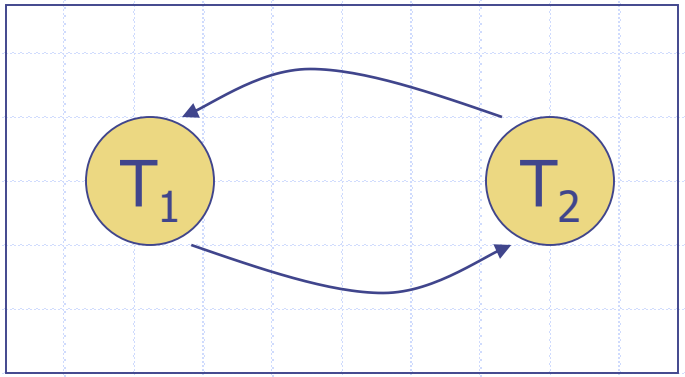
  S3 = $r_1(A)\ w_1(A)\ r_2(A)\ w_2(A)\ r_1(B)\ w_1(B)\ r_2(B)\ w_2(B)$

  S4 = $r_1(A)\ r_2(A)\ w_2(A)\ r_2(B)\ w_1(A)\ r_1(B)\ w_1(B)\ w_2(B)$

- Giải:



P(S3)



P(S4)

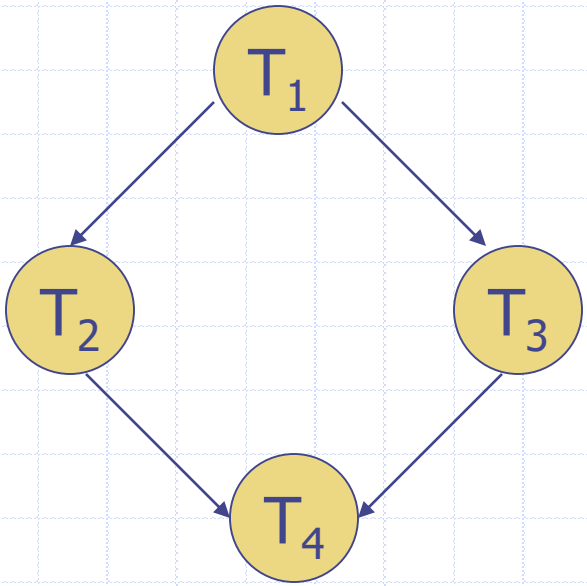
Khả tuần tự xung đột (5)

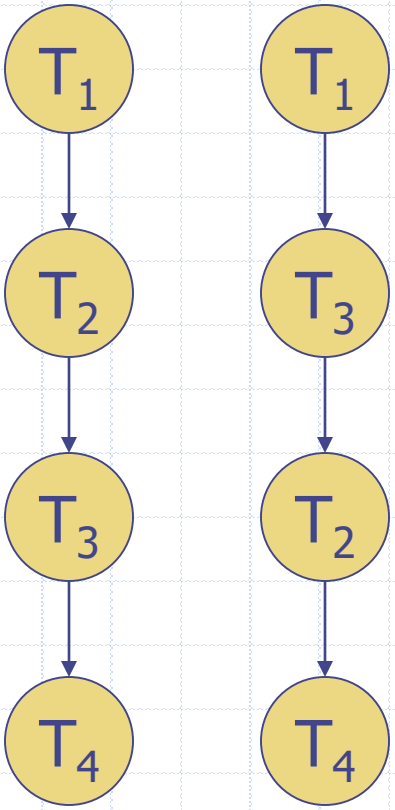
- **Bài tập**: Xét tính khả tuần tự xung đột của S5:

S5 = R3(Q) W4(Q) W3(Q) W6(Q)

Khả tuần tự xung đột (6)

- Làm thế nào để xác định lịch trình tuần tự tương đương xung đột với một lịch trình đã cho?



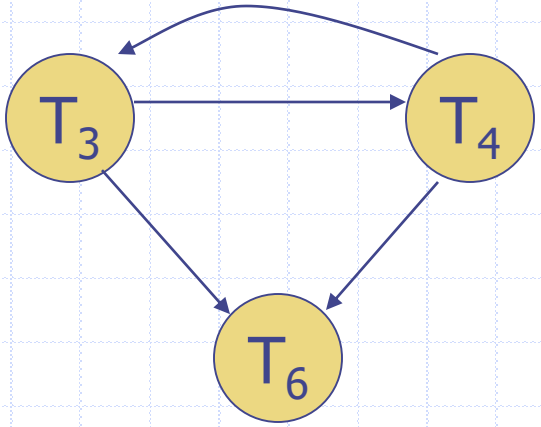


Sắp xếp topo

https://vi.wikipedia.org/wiki/Sắp_xếp_tô_pô

Khả tuần tự View (1)

- Xét một lịch trình sau: S9 = $r_3(Q)\ w_4(Q)\ w_3(Q)\ w_6(Q)$



➔ Không khả tuần tự XĐ
(chu trình $T_3 \rightleftarrows T_4$)

Tuy nhiên, LT này tương đương với LT: T_3, T_4, T_6.
Hoạt động Write của T_6 được gọi là hoạt động Write mù.

- Các lịch trình có hoạt động Write mù xuất hiện thì không khả tuần tự XĐ.

Khả tuần tự View (2)

- Một số khái niệm:
  - **Tương đương view** (view equivalent): hai lịch trình S1, S2 được gọi là tương đương view nếu

    (1) Nếu trong S1: $w_j(Q) \Rightarrow r_i(Q)$

    thì trong S2: $w_j(Q) \Rightarrow r_i(Q)$

    $\Rightarrow$ có nghĩa là "đọc giá trị được tạo ra"

    (2) Nếu trong S1: T_i đọc giá trị khởi đầu của Q,

    thì trong S2: T_i cũng đọc giá trị khởi đầu của Q

    (3) Nếu trong S1: T_i thực hiện việc ghi cuối cùng lên Q,

    thì trong S2: T_i cũng thực hiện việc ghi cuối cùng lên Q
  - **Khả tuần tự view** (view serializable): lịch trình S là khả tuần tự view nếu nó tương đương view với một LT tuần tự

Tuần tự xung đột $\Rightarrow$ Tuần tự view

Tuần tự view $\not\Rightarrow$ Tuần tự xung đột

Khả tuần tự View (3)

- Để kiểm tra tính khả tuần tự view của S, ta dùng đồ thị trình tự gán nhãn LP(S).

1. Thêm GD $\mathbf{T_b}$ để nó ghi ra tất cả các hạng mục dữ liệu:
   $S' = \underbrace{w_b(A)\ w_b(B)}_{\mathbf{T_b}}\ldots\mathbf{S}$
2. Thêm GD $\mathbf{T_f}$ để nó đọc tất cả các hạng mục dữ liệu:
   $S' = \mathbf{S}\ldots\ \underbrace{r_f(A)\ r_f(B)}_{\mathbf{T_f}}$
3. Tạo đồ thị trình tự gán nhãn (slide tiếp theo)
4. Kiểm tra tính khả tuần tự view của S: tách đồ thị LP(S) ra thành các Đồ thị thành phần, nếu tồn tại ít nhất một đồ thị thành phần không có chu trình thì S là khả tuần tự view.

Khả tuần tự View (4)

Giải thuật tạo Đồ thị trình tự gán nhãn:

a. Nếu ($w_i(A) \Rightarrow r_j(A)$ trong S') thì thêm cung $T_i \xrightarrow{0} T_j$

b. For each $w_i(A) \Rightarrow r_j(A)$ do
xem xét mọi $w_k(A)$: [Tk ≠Tb]

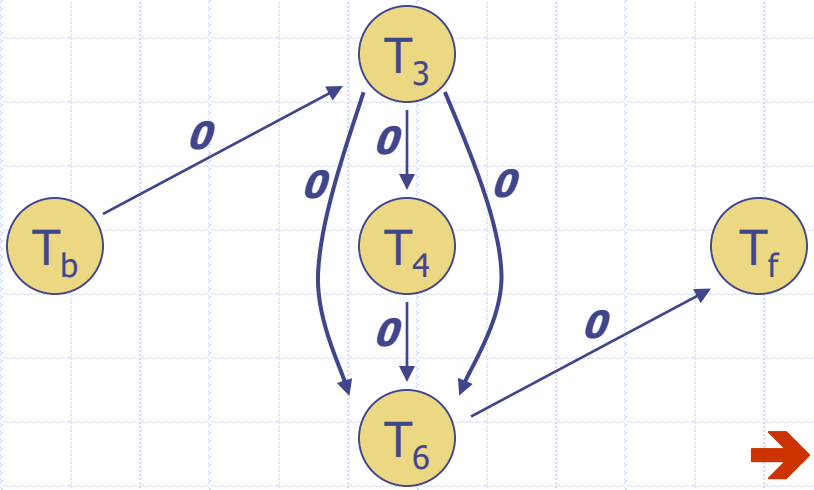
- Nếu $T_i \neq T_b \wedge T_j \neq T_f$ thì thêm cung:
  $T_k \xrightarrow{p} T_i$
  $T_j \xrightarrow{p} T_k$
- Nếu $T_i = T_b \wedge T_j \neq T_f$ thì thêm cung:
  $T_j \xrightarrow{0} T_k$
- Nếu $T_i \neq T_b \wedge T_j = T_f$ thì thêm cung:
  $T_k \xrightarrow{0} T_i$

Trong đó, p là số nguyên nhỏ nhất chưa sử dụng đến.

Khả tuần tự View (5)

- Xét tính khả tuần tự view của lịch trình sau:
  $S = r_3(Q)\ w_4(Q)\ w_3(Q)\ w_6(Q)$

1. $S' = w_b(Q)\ r_3(Q)\ w_4(Q)\ w_3(Q)\ w_6(Q)\ r_f(Q)$
2. Đồ thị trình tự gán nhãn.



- $T_i = T_b$, $T_j = T_3$
  - $T_k = T_4/T_6$
- $T_i = T_6$, $T_j = T_f$
  - $T_k = T_3/T_4$

➔ **Khả tuần tự view?**

Khả tuần tự View (6)

- Xét tính khả tuần tự view của lịch trình sau:

$$S = r_3(Q)\ w_4(Q)\ r_7(Q)\ w_3(Q)\ w_7(Q)$$

Bài giải

Lịch trình Khả phục hồi (Recoverable Schedules)

- Trong một hệ thống quản lý GD, nếu một GD T_i thất bại:
  1) Hủy bỏ tác động của GD này (nguyên tử)
  2) Hủy bỏ các GD phụ thuộc vào T_i

- Lịch trình khả phục hồi:
  - Một lịch trình khả phục hồi: là lịch trình có khả năng hủy bỏ bỏ GD bị thất bại và các GD phụ thuộc của nó.
  - Để một lịch trình khả phục hồi: đối với mỗi cặp T_i, T_j, nếu T_i đọc giá trị dữ liệu do T_j ghi thì T_i phải bàn giao sau T_j.

T_8	T_9
R(A) W(A)	
	R(A) Commit
R(B) ✗	
Schedule 10	

Khả phục hồi?

Cuộn lại hàng loạt (Cascading rollback)

- Là tình trạng một giao dịch hỏng hóc bị cuộn lại dẫn đến một loạt các giao dịch bị cuộn lại theo
- Trong lịch trình 11, nếu T10 hỏng hóc bị cuộn lại sẽ dẫn đến T11 và T12 phải cuộn lại theo

⇒ Mất rất nhiều chi phí cho việc cuộn lại

T_{10}	T_{11}	T_{12}
R(A)		
R(B)		
W(A)		
	R(A)	
	W(A)	
		R(A)

Schedule 11

Lịch trình Cascadeless (Cascadeless Schedules)

- Lịch trình cascadeless (không cuộn lại hàng loạt):
  - Là một lịch trình trong đó mỗi cặp giao dịch T_i, T_j: Nếu T_i đọc hạng mục dữ liệu do T_j ghi thì hoạt đông bàn giao của T_j phải xảy ra trước hoạt động Read của T_i.

T_{10}	T_{11}	T_{12}
R(A)		
W(A)		
	R(A)	
	W(A)	
		R(A)

Schedule 12

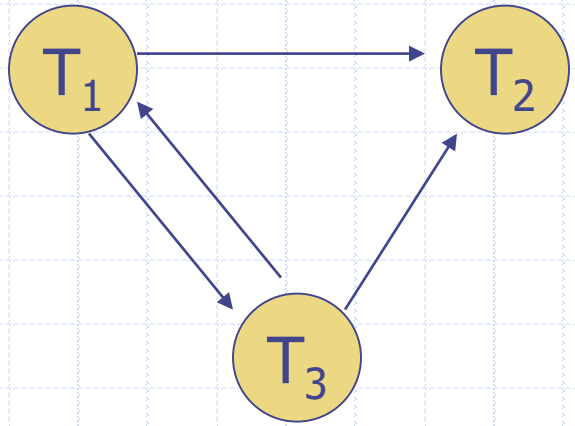
Cascadeless?

Cascadeless ⇒ Recoverable
Recoverable ⇒ Cascadeless

Lịch trình Cascadeless (2)

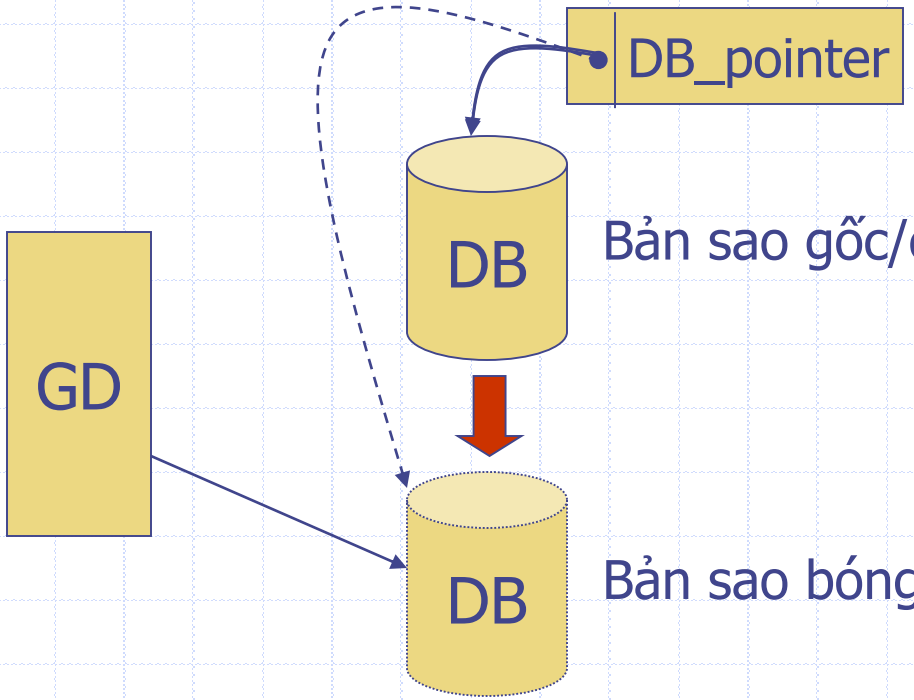
- S1 = $r_1(X)\ w_2(X)\ w_1(X)\ r_3(X)\ C_1\ C_2\ C_3$
- S2= $w_1(A)r_1(B)r_1(C)w_3(B)w_1(B)w_3(C)r_2(C)w_2(B)w_2(C)r_3(A)$
- Xác định tính Khả phục hồi và Không cuộn lại hàng loạt cho lịch trình S3:

  S3= $w_1(X)r_2(Y)r_1(Y)r_2(X)C_1C_2$



Thực thi tính nguyên tử và bền vững

- Trong các hệ QTCSDL, hệ thống phục hồi đảm bảo tính nguyên tử và bảo vệ cho các GD
- Một sơ đồ đơn giản để thực hiện, đó là sơ đồ Bản sao bóng (shadow copy)



Yêu cầu:

- Chỉ một GD tích cực tại 1 thời điểm.
- Sự cập nhật DB_pointer là nguyên tử

Không hiệu quả.

Các mức độ cô lập (Isolation levels)

- Các mức độ cô lập của SQL-92 (xếp theo độ giảm dần của mức độ cô lập):
  - Serializable: chế độ mặc định, các giao dịch phải thực hiện tuần tự khi truy xuất cùng hạng mục dữ liệu
  - Repeatable read: chỉ cho phép đọc các dữ liệu đã commit và giữa các chỉ thị đọc của cùng một giao dịch trên một hạng mục dữ liệu thì không cho phép giao dịch khác thực hiện cập nhật trên hạng mục dữ liệu đó
  - Read committed: chỉ cho phép đọc các dữ liệu đã commit, không có yêu cầu thêm về read khả lặp (phantom rows)
  - Read uncommitted: cho phép đọc cả các dữ liệu chưa commit (dirty read)

Questions?