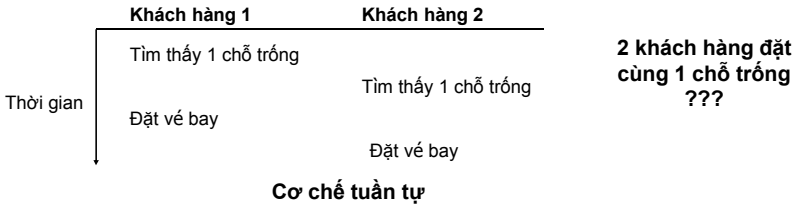
Chương 3: Quản lý giao tác

Nội dung chi tiết

- Giới thiệu
- Khái niệm giao tác (transaction)
  - Định nghĩa
  - Tính chất ACID của giao tác
  - Các thao tác của giao tác
  - Trạng thái của giao tác
- Lịch thao tác (schedule)
  - Giới thiệu
  - Định nghĩa
  - Lịch tuần tự (Serial schedule)
  - Lịch khả tuần tự (Serializable schedule)
    - Conflict-Serializable
    - View-Serializable

Giới thiệu

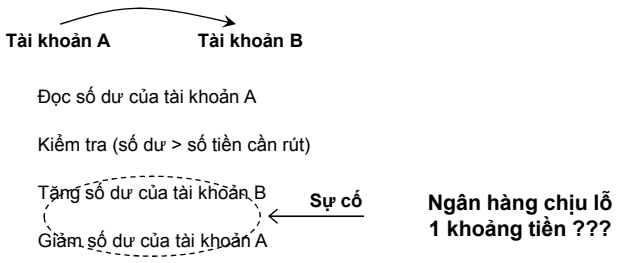
- Ví dụ
  - Hệ thống giao dịch ngân hàng
  - Hệ thống đặt vé bay
- DBMS là môi trường đa người dùng
  - Nhiều thao tác truy xuất lên cùng một đơn vị dữ liệu
  - Nhiều thao tác thi hành đồng thời



Cơ chế tuần tự

Giới thiệu (tt)

- Khi DBMS gặp sự cố
  - Các thao tác có thể làm cho trạng thái CSDL không chính xác



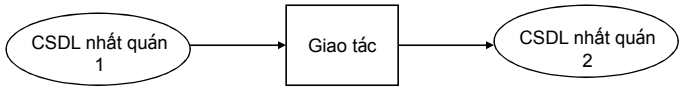
Nguyên tố

Nội dung chi tiết

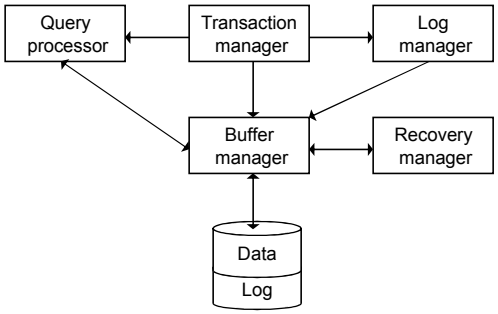
- Giới thiệu
- Khái niệm giao tác (transaction)
  - Định nghĩa
  - Tính chất ACID của giao tác
  - Các thao tác của giao tác
  - Trạng thái của giao tác
- Lịch thao tác (schedule)

Giao tác (Transaction)

- Giao tác là 1 đơn vị xử lý nguyên tố gồm 1 chuỗi các hành động tương tác lên CSDL
  - Nguyên tố: không thể phân chia được nữa



Giao tác (tt)



Tính chất ACID của giao tác

- Nguyên tố (**A**tomicity)
  - Hoặc là toàn bộ hoạt động của giao dịch được phản ánh đúng đắn trong CSDL hoặc không có hoạt động nào cả
- Nhất quán (**C**onsistency)
  - Một giao tác được thực hiện độc lập với các giao tác khác xử lý đồng thời với nó để bảo đảm tính nhất quán cho CSDL
- Cô lập (**I**solation)
  - Một giao tác không quan tâm đến các giao tác khác xử lý đồng thời với nó
- Bền vững (**D**urability)
  - Mọi thay đổi mà giao tác thực hiện trên CSDL phải được ghi nhận bền vững

Ví dụ

```
T: Read(A,t);
   t:=t-50;
   Write(A,t);
   Read(B,t);
   t:=t+50;
   Write(B,t);
```

- Consistency
  - Tổng A+B là không đổi
  - Nếu CSDL nhất quán trước khi T được thực hiện thì sau khi T hoàn tất CSDL vẫn còn nhất quán

Ví dụ (tt)

```
T: Read(A,t);
   t:=t-50;
   Write(A,t);
   Read(B,t);
   t:=t+50;
   Write(B,t);
```

- Atomicity
  - A=100, B=200 (A+B=300)
  - Tại thời điểm sau khi write(A,t)
    - A=50, B=200 (A+B=250) - CSDL không nhất quán
  - Tại thời điểm sau khi write(B,t)
    - A=50, B=250 (A+B=300) - CSDL nhất quán
  - Nếu T không bao giờ bắt đầu thực hiện hoặc T được đảm bảo phải hoàn tất thì trạng thái không nhất quán sẽ không xuất hiện

Ví dụ (tt)

```
T: Read(A,t);
   t:=t-50;
   Write(A,t);
   Read(B,t);
   t:=t+50;
   Write(B,t);
```

- Durability
  - Khi T kết thúc thành công
  - Dữ liệu sẽ không thể nào bị mất bất chấp có sự cố hệ thống xảy ra

Ví dụ (tt)

```
T: Read(A,t);
   t:=t-50;
T' Write(A,t);
   Read(B,t);
   t:=t+50;
   Write(B,t);
```

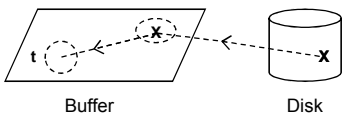
- Isolation
  - Giả sử có 1 giao tác T' thực hiện phép toán A+B và chen vào giữa thời gian thực hiện của T
  - T' kết thúc: A+B=50+200=250
  - T kết thúc: A+B=50+250=300
  - Hệ thống của các giao tác thực hiện đồng thời có trạng thái tương đương với trạng thái hệ thống của các giao tác thực hiện tuần tự theo 1 thứ tự nào đó

Các thao tác của giao tác

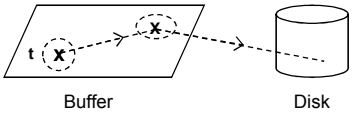
- Giả sử CSDL gồm nhiều đơn vị dữ liệu
- Một đơn vị dữ liệu (element)
  - Có một giá trị
  - Được truy xuất và sửa đổi bởi các giao tác
    - Quan hệ (relation) - Lớp (class)
    - Khối dữ liệu trên đĩa (block) / trang (page)
    - Bộ (tuple) - Đối tượng (object)

Các thao tác của giao tác (tt)

- Input(X)
- Read(X, t)



- Write(X, t)
- Output(X)



- Bufffer manager
  - Input
  - Output
- Transaction
  - Read
  - Write

Ví dụ

- Giả sử CSDL có 2 đơn vị dữ liệu A và B với ràng buộc A=B trong mọi trạng thái nhất quán
- Giao tác T thực hiện 2 bước
  - A:=A*2
  - B:=B*2

- Biểu diễn T
  - Read(A,t) ; t=t*2; Write(A,t);
  - Read(B,t) ; t=t*2; Write(B,t);

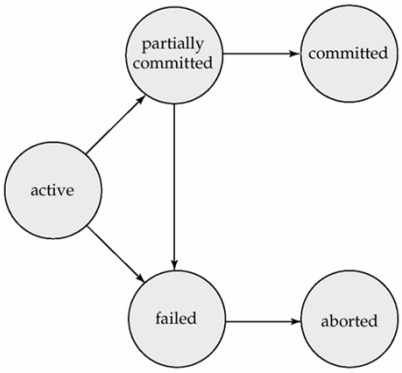
Ví dụ (tt)

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

Trạng thái của giao tác

- Active
  - Ngay khi bắt đầu thực hiện thao tác đọc/ghi
- Partially committed
  - Sau khi lệnh thi hành cuối cùng thực hiện
- Failed
  - Sau khi nhận ra không thể thực hiện các hành động được nữa
- Aborted
  - Sau khi giao tác được quay lui và CSDL được phục hồi về trạng thái trước trạng thái bắt đầu giao dịch
    - Bắt đầu lại giao tác (nếu có thể)
    - Hủy giao tác
- Committed
  - Sau khi mọi hành động hoàn tất thành công

Sơ đồ trạng thái của giao tác



Nội dung chi tiết

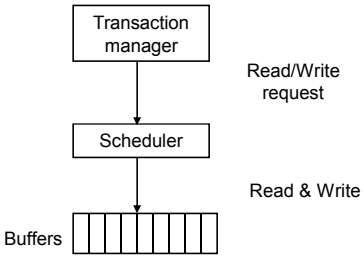
- Giới thiệu
- Khái niệm giao tác (transaction)
- Lịch thao tác (schedule)
  - Giới thiệu
  - Định nghĩa
  - Lịch tuần tự (Serial schedule)
  - Lịch khả tuần tự (Serializable schedule)
    - Conflict-Serializable
    - View-Serializable

Giới thiệu

- Thực hiện tuần tự
  - Tại một thời điểm, một giao tác chỉ có thể bắt đầu khi giao tác trước nó hoàn tất
- Thực hiện đồng thời
  - Cho phép nhiều giao tác cùng truy xuất dữ liệu
  - Gây ra nhiều phức tạp về nhất quán dữ liệu
  - Tuy nhiên
    - Tận dụng tài nguyên và thông lượng (throughput)
      - Trong khi 1 giao tác đang thực hiện đọc/ghi trên đĩa, 1 giao tác khác đang xử lý tính toán trên CPU
    - Giảm thời gian chờ
      - Các giao tác ngắn phải chờ đợi các giao tác dài
      - Chia sẻ chu kỳ CPU và truy cập đĩa để làm giảm sự trì hoãn trong khi các giao tác thực thi

Bộ lập lịch (Scheduler)

- Là một thành phần của DBMS có nhiệm vụ lập 1 lịch để thực hiện *n* giao tác xử lý đồng thời



Lịch thao tác (Schedule)

- Một lịch thao tác *S* được lập từ *n* giao tác $T_1, T_2, ..., T_n$ được xử lý đồng thời là 1 **thứ tự thực hiện 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 trong giao tác
- Gồm có
  - Lịch tuần tự (Serial)
  - Lịch khả tuần tự (Serializable)
    - Confict-Serializability
    - View-Serializability

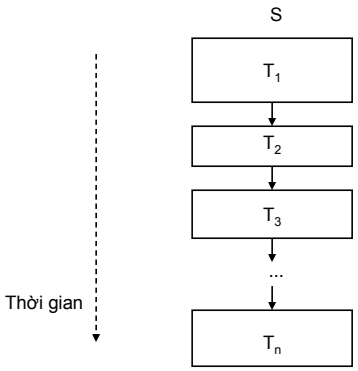
Ví dụ

T_1	T_2
Read(A,t)	Read(A,s)
t:=t+100	s:=s*2
Write(A,t)	Write(A,s)
Read(B,t)	Read(B,s)
t:=t+100	s:=s*2
Write(B,t)	Write(B,s)

- Giả sử ràng buộc nhất quán trên CSDL là A=B
- Từng giao tác thực hiện riêng lẽ thì tính nhất quán sẽ được bảo toàn

Lịch tuần tự (Serial schedule)

- Một lịch *S* được gọi là tuần tự nếu các hành động của các giao tác T_i *(i=1..n)* được thực hiện liên tiếp nhau



Lịch tuần tự (tt)

S_1

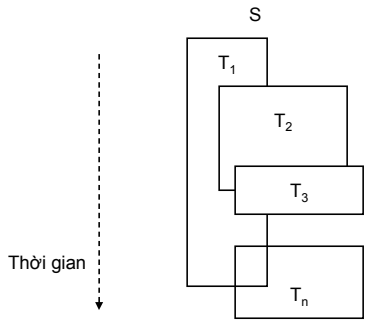
T_1	T_2	A	B
		25	25
Read(A,t)			
t:=t+100			
Write(A,t)		125	
Read(B,t)			
t:=t+100			
Write(B,t)			125
	Read(A,s)		
	s:=s*2		
	Write(A,s)	**250**	
	Read(B,s)		
	s:=s*2		
	Write(B,s)		**250**

S_2

T_1	T_2	A	B
		25	25
	Read(A,s)		
	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 khả tuần tự (Serializable schedule)

- Một lịch S được lập từ *n* giao tác $T_1, T_2, ..., T_n$ xử lý đồng thời được gọi là khả tuần tự nếu nó cho cùng kết quả với 1 lịch tuần tự nào đó được lập từ *n* giao tác này



Lịch khả tuần tự (tt)

S_3

T_1	T_2	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,t)			
t:=t+100			
Write(B,t)			125
	Read(B,s)		
	s:=s*2		
	Write(B,s)		**250**

- Trước S_3 khi thực hiện
  - A=B=c
  - với c là hằng số
- Sau khi S_3 kết thúc
  - A=2*(c+100)
  - B=2*(c+100)
- Trạng thái CSDL nhất quán
- S_3 là khả tuần tự

Lịch khả tuần tự (tt)

S_4

T_1	T_2	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**

- 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 không nhất quán
- S_4 không khả tuần tự

Lịch khả tuần tự (tt)

S_5	T_1	T_2	A	B
			25	25
	Read(A,t)			
	t:=t+100			
	Write(A,t)		125	
		Read(A,s)		
		s:=s*1		
		Write(A,s)	**125**	
		Read(B,s)		
		s:=s*1		
		Write(B,s)		25
	Read(B,t)			
	t:=t+100			
	Write(B,t)			**125**

- Khi S_5 kết thúc
  - A và B bằng nhau
  - Trạng thái cuối cùng nhất quán
- S_5 khả tuần tự, có kết quả giống với lịch tuần tự
  - T_1, T_2
  - T_2, T_1

Lịch khả tuần tự (tt)

- Để xác định 1 lịch thao tác có <u>khả tuần tự</u> hay không
  - Xem xét chi tiết các hành động của các giao tác???
- Tuy nhiên
  - Bộ lập lịch khó biết được "Giao tác này có nhân A với hằng số khác 1 hay không?"
- Nhưng
  - Bộ lập lịch phải biết các thao tác đọc/ghi của giao tác
    - Những đơn vị dữ liệu nào được giao tác đọc
    - Những đơn vị dữ liệu nào có thể bị thay đổi
- Để đơn giản công việc cho bộ lập lịch
  - Nếu có hành động nào tác động lên đơn vị dữ liệu A làm cho trạng thái CSDL không nhất quán thì giao tác vẫn thực hiện hành động đó
  - Thao tác đọc và ghi – Read(X) / Write(X)
    - Qui ước: $r_i(X)$ và $w_i(X)$

Conflict-Serializability

- Ý tưởng
  - Xét 2 hành động liên tiếp nhau trong 1 lịch thao tác
    - Nếu thứ tự của chúng được đổi cho nhau
    - Thì hoạt động của ít nhất 1 giao tác có thể thay đổi

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 đơn vị dữ liệu X, Y
  - $r_i(X)$; $w_j(Y)$
    - Không xung đột khi X≠Y
    - T_j ghi Y sau khi T_i đọc X, giá trị của X không bị thay đổi
    - T_i đọc X không ảnh hưởng gì đến T_j ghi giá trị của Y
  - $w_i(X)$; $r_j(Y)$
    - Không xung đột khi X≠Y
  - $w_i(X)$; $w_j(Y)$
    - Không xung đột khi X≠Y

Conflict-Serializability (tt)

- Hai hành động xung đột nếu
  - Thuộc 2 giao tác khác nhau
  - Truy xuất đến cùng 1 đơn vị dữ liệu
  - Có ít nhất một hành động ghi (write)

  → không thể hoán vị thứ tự

T_i	T_j
Write(A)	
	Write(A)

Loại bỏ sự trùng hợp ngẫu nhiên

T_i	T_j
Read(A)	
	Write(A)

T_i	T_j
Write(A)	
	Read(A)

Conflict-Serializability (tt)

- Ví dụ

S	T_1	T_2
	Read(A)	
	Write(A)	
		Read(A)
		Write(A)
	Read(B)	
	Write(B)	
		Read(B)
		Write(B)

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

S′	T_1	T_2
	Read(A)	
	Write(A)	
	Read(B)	
	Write(B)	
		Read(A)
		Write(A)
		Read(B)
		Write(B)

Conflict-Serializability (tt)

- Định nghĩa
  - S, S' là những lịch thao tác conflict-equivalent
    - Nếu S có thể được chuyển thành S' bằng một chuỗi những hoán vị các 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 conflict-serializable → S khả tuần tự
- S conflict-serializable ← S khả tuần tự ???

Conflict-Serializability (tt)

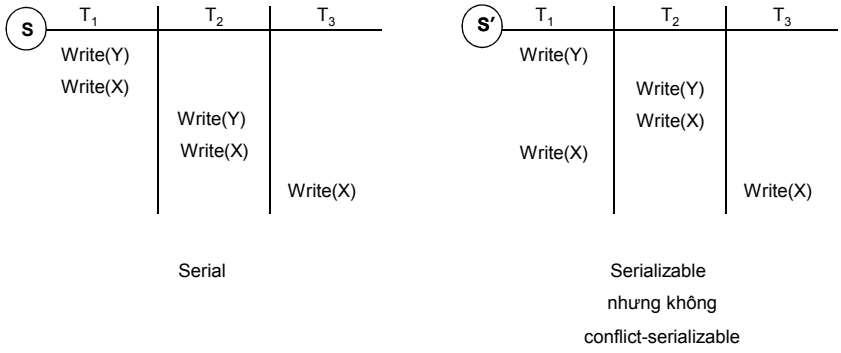
❑ Xét lại lịch S_5

S_5	T_1	T_2	A	B
			25	25
	Read(A,t)			
	t:=t+100			
	Write(A,t)		125	
		Read(A,s)		
		s:=s*1		
		Write(A,s)	**125**	
		Read(B,s)		
		s:=s*1		
		Write(B,s)		25
	Read(B,t)			
	t:=t+100			
	Write(B,t)			**125**

Serializable nhưng không conflict-serializable

Conflict-Serializability (tt)

- Xét trường hợp

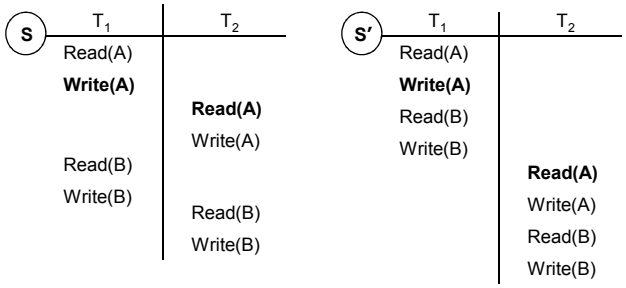


Serial

Serializable nhưng không conflict-serializable

Kiểm tra Conflict-Serializability

- Cho lịch S
  - S có conflict-serializable 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' sẽ cũng ở thứ tự đó



Kiểm tra Conflict-Serializability (tt)

- Cho lịch S có 2 giao tác T_1, T_2
  - T_1 thực hiện hành động A_1
  - T_2 thực hiện hành động A_2
  - Ta nói T_1 thực hiện trước T_2, ký kiệu $T_1 <_S T_2$, khi
    - A_1 được thực hiện trước A_2 trong S
      - A_1 không nhất thiết phải liên tiếp A_2
    - A_1 và A_2 cùng thao tác lên 1 đơn vị dữ liệu
    - Có ít nhất 1 hành động ghi trong A_1 và A_2

Precedence graph

- Cho lịch S gồm các giao tác T_1, T_2, …, T_n
- Đồ thị trình tự của S, ký hiệu P(S), có
  - Đỉnh là các giao tác T_i
    - Ta có thể đặt nhãn cho đỉnh là i
  - Cung đi từ T_i đến T_j nếu $T_i <_S T_j$
- Nếu P(S) không có chu trình thì S conflict-serializable
- Thứ tự hình học (topological order) của các đỉnh là thứ tự của các giao tác trong lịch tuần tự

Precedence graph (tt)

- Bổ đề
  - S_1, S_2 conflict-equivalent $\Rightarrow P(S_1) = P(S_2)$
- Chứng minh
  - Giả sử $P(S_1) \neq P(S_2)$
  - $\Rightarrow \exists T_i$ sao cho $T_i \rightarrow T_j$ có trong S_1 và không có trong S_2
  - $\Rightarrow S_1 = \ldots p_i(A) \ldots q_j(A) \ldots$
    $S_2 = \ldots q_j(A) \ldots p_i(A) \ldots$
  - Và $p_i(A)$ và $q_j(A)$ là xung đột
  - $\Rightarrow S_1$, S_2 không conflict-equivalent

Precedence graph (tt)

- Chú ý
  - $P(S_1) = P(S_2) \Rightarrow S_1$, S_2 conflict-equivalent
- Xét 2 trường hợp

Precedence graph (tt)

- Định lý
  - $P(S_1)$ không có chu trình $\Leftrightarrow$ S_1 conflict-serializable
- Chứng minh ($\Leftarrow$)
  - Giả sử S_1 conflict-serializable
  - $\Rightarrow \exists S_2$ sao cho: S_1 và S_2 conflict-equivalent
  - $\Rightarrow P(S_2) = P(S_1)$
  - S_2 là lịch tuần tự
  - $P(S_1)$ không có chu trình vì $P(S_2)$ không có chu trình

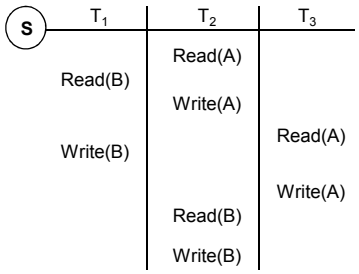
Precedence graph (tt)

- Định lý
  - $P(S_1)$ không có chu trình $\Leftrightarrow$ S_1 conflict-serializable
- Chứng minh ($\Rightarrow$)
  - Giả sử $P(S_1)$ không có chu trình
  - Ta biến đổi S_1 như sau
    - Chọn ra 1 giao tác T_1 không có cung nào đi đến nó

      $S_1 = \ldots q_j(A) \ldots p_1(A) \ldots$
    - Đem T_1 lên vị trí đầu

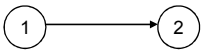
      $S_1 =$ < hành động của T_1 >< … phần còn lại ...>
    - Lập lại quá trình này để tuần tự hoá cho phần còn lại
  - S_1 tuần tự

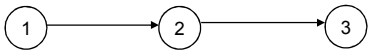
Ví dụ



- $T_2 <_S T_3$

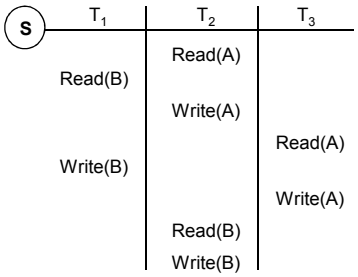
- $T_1 <_S T_2$





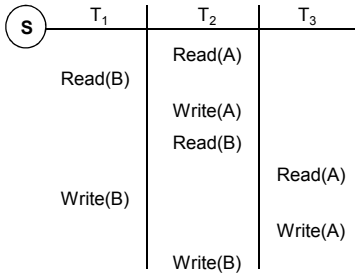
- P(S) không có chu trình
- S conflict-serializable theo thứ tự T_1, T_2, T_3

Ví dụ (tt)



- S conflict-serializable theo thứ tự T_1, T_2, T_3

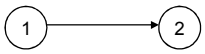
Ví dụ (tt)

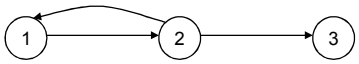


- $T_2 <_S T_3$

- $T_2 <_S T_1$

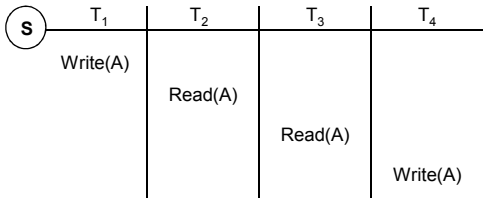
- $T_1 <_S T_2$





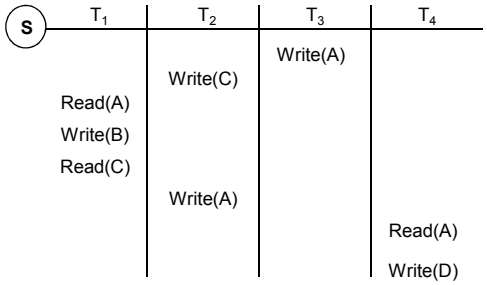
- P(S) có chu trình
- S không conflict-serializable

Bài tập



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

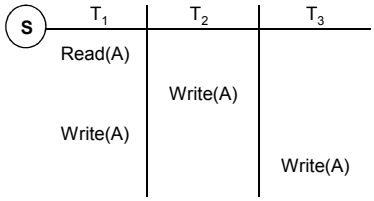
Bài tập (tt)

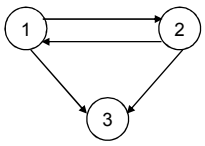


- Vẽ P(S)
- S có conflict-serializable không?

View-Serializability

- Xét lịch S





- P(S) có chu trình
- S không conflict-serializable

View-Serializability (tt)

❑ So sánh lịch S và 1 lịch tuần tự S'

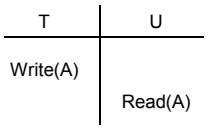
Không conflict-serializable

Serial

- Trong S và S' đều có T_1 thực hiện read(A)
- T_2 và T_3 không đọc A
- Kết quả của S và S' giống nhau

View-Serializability (tt)

- Ý tưởng
  - Xét trường hợp



  - Nhận xét
    - Sau khi T ghi A xong mà không có giao tác nào đọc giá trị của A
    - Khi đó, hành động $w_T(A)$ có thể chuyển đến 1 vị trí khác trong lịch thao tác mà ở đó cũng không có giao tác nào đọc A
  - Ta nói
    - Hành động $r_U(A)$ có gốc là giao tác T

View-Serializability (tt)

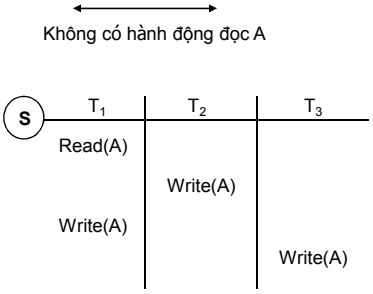
- Định nghĩa
  - S, S' là những lịch thao tác view-equivalent
    - 1- Nếu trong S có $w_j(A)$ … $r_j(A)$ thì trong S' cũng có $w_j(A)$ … $r_j(A)$
    - 2- Nếu trong S có $r_i(A)$ là thao tác đọc giá trị ban đầu của A thì trong S' cũng $r_i(A)$ đọc giá trị ban đầu của A
    - 3- Nếu trong S có $w_i(A)$ là thao tác ghi giá trị sau cùng lên A thì trong S' cũng có $w_i(A)$ ghi giá trị sau cùng lên A
  - Một lịch thao tác S là view-serializable
    - Nếu S là view-equivalent với một lịch thao tác tuần tự nào đó
- S view-serializable → S conflict-serializable
- S view-serializable ← S conflict-serializable???

View-Serializability (tt)

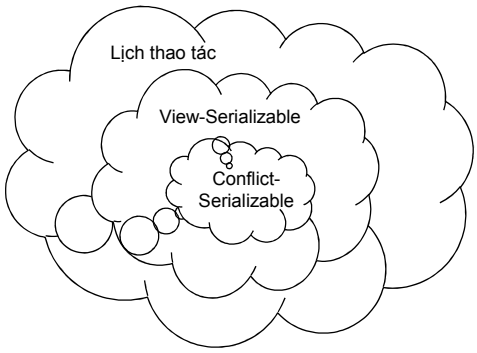
- S conflict-serializable ⇒ S view-serializable
- Chứng minh
  - Hoán vị các hành động không xung đột
    - Không làm ảnh hưởng đến những thao tác đọc
    - Cũng không làm ảnh hưởng đến trạng thái CSDL

View-Serializability (tt)

- S view-serializable ⇒ S/conflict-serializable
  - Trong S có những hành động ghi không có tác dụng (useless)
    - S = … $w_2(A)$ …… $w_3(A)$ …



View-Serializability (tt)



Kiểm tra View-Serializability (tt)

- Cho 1 lịch thao tác S
- Thêm 1 giao tác cuối T_f vào trong S sao cho T_f thực hiện việc đọc hết tất cả đơn vị dữ liệu ở trong S
  - (bỏ qua điều kiện thứ 3 của định nghĩa view-equivalent)
  - S = … $w_1(A)$……$w_2(A)$ $\mathbf{r_f(A)}$

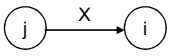
Ghi A cuối cùng

- Thêm 1 giao tác đầu tiên T_b vào trong S sao cho Tb thực hiện việc ghi các giá trị ban đầu cho các đơn vị dữ liệu
  - (bỏ qua điều kiện thứ 2 của định nghĩa view-equivalent)
  - S = $\mathbf{w_b(A)}$… $w_1(A)$……$w_2(A)$…

Kiểm tra View-Serializability (tt)

- Vẽ đồ thị trình tự gán nhãn cho S, ký hiệu G(S), (PolyGraph)
  - Đỉnh là các giao tác T_i (bao gồm T_b và T_f)
  - Cung
    - (1) Nếu có $r_i(X)$ với gốc là T_j thì vẽ cung đi từ T_j đến T_i

      $w_j(X)$ … $r_i(X)$



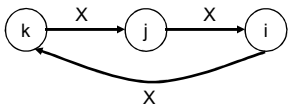
    - (2) Với mỗi $w_j(X)$ … $r_i(X)$, xét $w_k(X)$ sao cho T_k không chèn vào giữa T_j và T_i

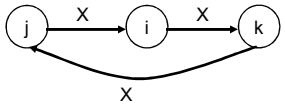
Kiểm tra View-Serializability (tt)

- (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	T_j	T_i
Write(X)		
	Write(X)	
		Read(X)

T_j	T_i	T_k
Write(X)		
	Read(X)	
		Write(X)





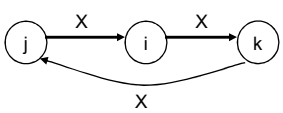
Chọn 1 cung vừa tạo sao cho đồ thị không có chu trình

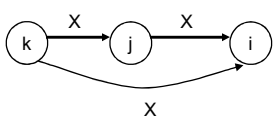
Kiểm tra View-Serializability (tt)

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

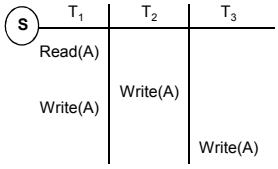
T_k	$T_j = T_b$	T_i	T_k
Write(X)			
	Write(X)		
		Read(X)	
			Write(X)

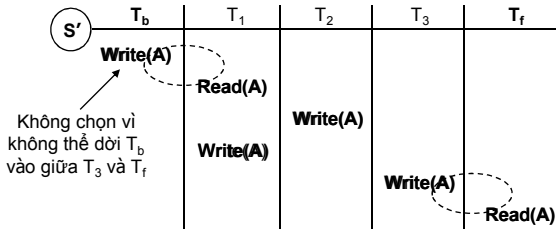
T_k	T_j	$T_i = T_f$	T_k
Write(X)			
	Write(X)		
		Read(X)	
			Write(X)



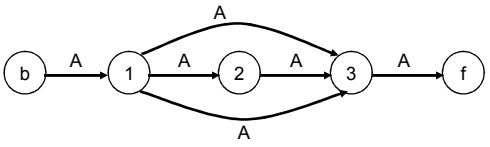


Ví dụ

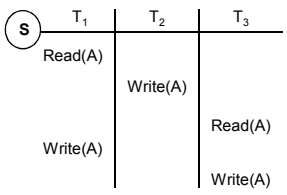


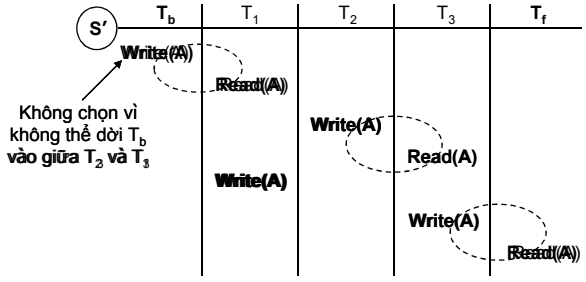


- G(S) không có chu trình
- S view-serializable theo thứ tự T_b, T_1, T_2, T_3, T_f

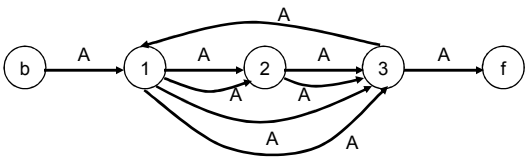


Ví dụ (tt)

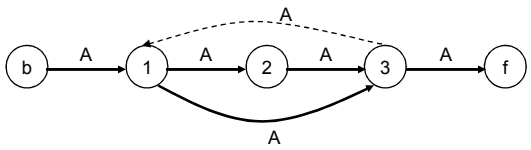




- G(S) có chu trình
- S không view-serializable



Ví dụ (tt)



- Bỏ cung từ T_3 sang T_1
- G(S) không chu trình
- S view-serializable theo thứ tự T_b, T_1, T_2, T_3, T_f

Bài tập

S	T_1	T_2	T_3
		Read(B)	
		Write(A)	
	Read(A)		
			Read(A)
	Write(B)		
		Write(B)	
			Write(B)

- Vẽ G(S)
- S có view-serializable?

Bài tập (tt)

S

T_1	T_2	T_3	T_4
	Read(A)		
Read(A)			
Write(C)			
		Read(C)	
Write(B)			
			Read(B)
		Write(A)	
			Read(C)
	Write(D)		
	Read(B)		
			Write(A)
			Write(B)

- Vẽ G(S)
- S có view-serializable?