Chương 3

QUẢN LÝ GIAO TÁC

(TRANSACTION MANAGEMENT)

Thời lượng: 9 tiết

GIẢNG VIÊN: THS. THÁI BẢO TRÂN

1

TÀI LIỆU THAM KHẢO

- **"Database System the complete book" – Hector Garcia, Jeffrey D. Ullman and Jennifer Widom**

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
- Bộ lập lịch (schedule)
  - Giới thiệu
  - Định nghĩa
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)
    - Conflict-Serializable
    - View-Serializable

3

GIỚI THIỆU TRANSACTION

- Một transaction là một đơn vị thực hiện chương trình truy xuất và có thể cập nhật nhiều mục dữ liệu.
- Một transaction thường là kết quả của việc thực hiện một chương trình người dùng được viết trong một ngôn ngữ cấp cao hay ngôn ngữ SQL và được phân cách bởi các câu lệnh có dạng:

  begin transaction

  …

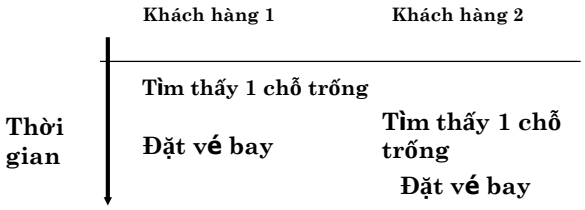
  end transaction.

- Giao dịch bao gồm tất cả các hoạt động được thực hiện giữa begin và end transaction.

4

GIỚI THIỆU TRANSACTION (TT)

- **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



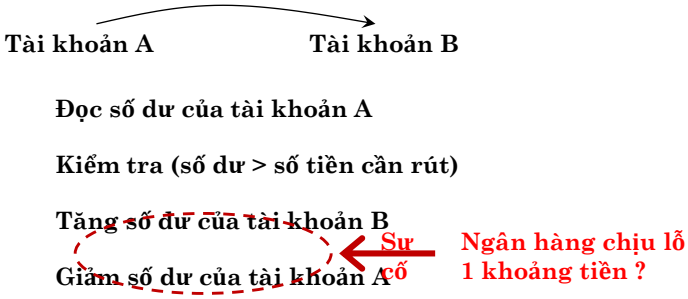
2 khách hàng đặt cùng 1 chỗ trống ?

5

Cơ chế tuần tự

GIỚI THIỆU TRANSACTION (TT)

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



Ngân hàng chịu lỗ 1 khoảng tiền ?

6

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)
  - Giới thiệu
  - Định nghĩa
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serializable schedule)
    - Conflict-Serializable
    - View-Serializable

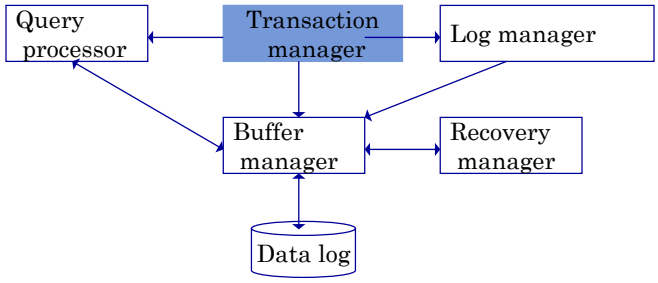
7

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

8

GIAO TÁC (TT)



9

ĐỂ ĐẢM BẢO TÍNH TOÀN VẸN CỦA DỮ LIỆU, TA YÊU CẦU HỆ CSDL DUY TRÌ CÁC TÍNH CHẤT SAU CỦA GIAO TÁC:

- Nguyên tố (Atomicity)
- Nhất quán (Consistency)
- Cô lập (Isolation)
- Bền vững (Durability)

TÍNH CHẤT ACID CỦA GIAO TÁC

10

TÍNH CHẤT ACID CỦA GIAO TÁC

- **Nguyên tố (Atomicity)**
  - 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 (Consistency)**
  - 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

11

TÍNH CHẤT ACID CỦA GIAO TÁC (TT)

- **Tính cô lập (Isolation)**
  - Cho dù nhiều giao dịch có thể thực hiện đồng thời, hệ thống phải đảm bảo rằng đối với mỗi cặp giao dịch T_i, T_j , hoặc T_j kết thúc thực hiện trước khi T_i khởi động hoặc T_j bắt đầu sự thực hiện sau khi T_i kết thúc.
  - Như vậy, mỗi giao dịch không cần biết đến các giao dịch khác đang thực hiện đồng thời trong hệ thống.
- **Tính bền vững (Durability)**. Sau một giao dịch hoàn thành, các thay đổi đã được tạo ra đối với CSDL vẫn còn ngay cả khi xảy ra sự cố hệ thống.

KÝ HIỆU

- **READ(X):** chuyển mục dữ liệu X từ CSDL đến buffer của giao dịch thực hiện hoạt động READ này.
  Thứ tự thực hiện:
  - Tìm địa chỉ bộ nhớ ngoài chứa X
  - Chép X vào bộ nhớ chính
  - Chuyển X vào biến X
- **WRITE(X):** chuyển mục dữ liệu X từ buffer của giao dịch thực hiện WRITE đến CSDL. Thứ tự thực hiện:
  - Tìm địa chỉ ô nhớ trong chứa X
  - Chép X vào biến X
  - Ghi dữ liệu lên bộ nhớ ngoài tại X
- Qui ước: r(X) và w(X)

VÍ DỤ

T là một giao dịch chuyển 50$ từ tài khoản A sang tài khoản B.
Giao dịch này có thể được xác định như sau:

```
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

14

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

15

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

16

VÍ DỤ (TT)

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

17

SỰ CẦN THIẾT ĐIỀU KHIỂN GIAO TÁC ĐỒNG THỜI

- **Xét ví dụ:**

  TaiKhoan(MATK,TenKhachHang,SoTien)

  RutTien(MaTK,NgayRut,SoTien)

  Giả sử có 2 giao tác: T1 và T2

T1	T2
Read(X) X = X-10 Write(X)	Read(X) X = X-5 Write(X)

VỚI SOTIEN=100

T1	T2
Read(X)	
X=100	Read(X)
X = X-10	X = 100
X = 90	X = X-5
Write(X)	X = 95
X=90	Write(X)
	X = 95

Dữ liệu bị sai!

Việc điều khiển giao tác đồng thời là quan trọng vì nếu không sẽ dẫn đến 2 sai phạm:

- **Đọc sai dữ liệu**
- **Ghi đè dữ liệu**

NHẬN XÉT

- **Khi giao tác chuyển tới HQTCSDL thì:**
  - **Hoặc là tất cả**
  - **Hoặc là giao tác không làm dữ liệu mâu thuẫn**
- **Cho dù nhiều giao dịch có thể thực hiện đồng thời, hệ thống phải đảm bảo rằng đối với mỗi cặp giao dịch T^i, T^j , hoặc T^j kết thúc thực hiện trước khi T^i khởi động hoặc T^j bắt đầu sự thực hiện sau khi T^i kết thúc.**
- **Như vậy, mỗi giao dịch không cần biết đến các giao dịch khác đang thực hiện đồng thời trong hệ thống.**

CÁC THAO TÁC CỦA GIAO TÁC

- Giả sử CSDL gồm nhiều đơn vị dữ liệu (element)
- Một đơn vị dữ liệu:
  - 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)

21

CÁC THAO TÁC CỦA GIAO TÁC (TT)

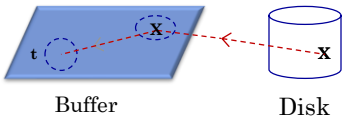
Các truy xuất CSDL được thực hiện bởi hai hoạt động sau:

- READ(X). chuyển hạng mục dữ liệu X từ CSDL đến buffer của giao dịch thực hiện hoạt động READ này.
- WRITE(X). chuyển hạng mục dữ liệu X từ buffer của giao dịch thực hiện WRITE đến CSDL

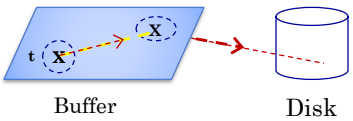
22

CÁC THAO TÁC CỦA GIAO TÁC (TT)

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



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



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

23

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);**

24

VÍ DỤ (TT)

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

25

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

26

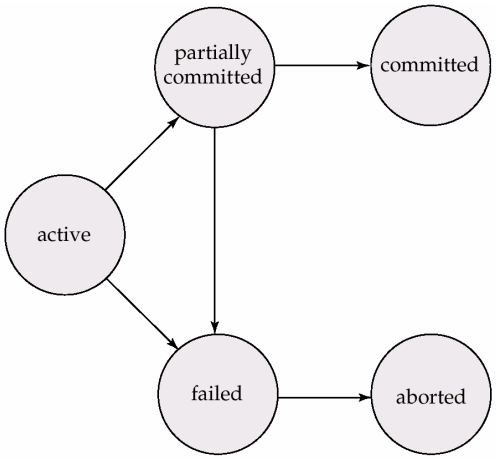
CÁC TRẠNG THÁI CỦA GIAO TÁC

Một giao dịch phải ở trong một trong các trạng thái sau:

- Hoạt động (Active)
  - Ngay khi bắt đầu thực hiện thao tác đọc/ghi
- Được bàn giao bộ phận (Partially committed)
  - Sau khi lệnh thi hành cuối cùng được thực hiện
- Thất bại (Failed)
  - Sau khi phát hiện ra sự thực hiện không thể tiếp tục được nữa
- Bỏ dở (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
- Được bàn giao (Committed)
  - Sau khi mọi hành động hoàn tất thành công

27

SƠ ĐỒ TRẠNG THÁI CỦA MỘT GIAO TÁC



28

NỘI DUNG CHI TIẾT

- Giới thiệu
- Khái niệm giao tác (transaction)
- Bộ lập lịch (schedule)
  - Giới thiệu
  - Định nghĩa
  - Lịch tuần tự (serial schedule)
  - Lịch khả tuần tự (serilizable schedule)
    - Conflict-Serializable
    - View-Serializable

29

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
  - Có 2 lý do để thực hiện đồng thời:
    - 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

30

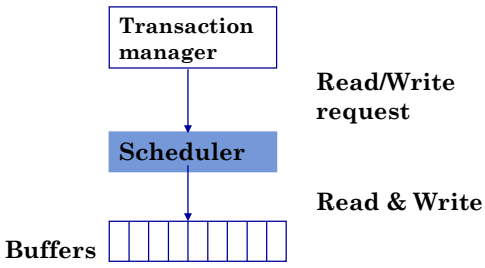
GIỚI THIỆU (TT)

- Khi có nhiều giao tác thực hiện đồng thời, tính nhất quán CSDL có thể bị phá vỡ mặc dù cá nhân mỗi giao tác vẫn thực hiện đúng đắn.
- Vì vậy, cần có khái niệm Lịch thao tác (schedule) để xác định những thực hiện nào đảm bảo tính nhất quán.
- Bộ phận quản lý các lịch thao tác này gọi là Bộ lập lịch (scheduler).

31

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



32

LỊCH THAO TÁC (SCHEDULE)

- Lịch S của n giao tác T_1, T_2, ..., T_n là dãy có thứ tự các thao tác trong n giao tác này
- Thứ tự xuất hiện của các giao tác trong lịch phải giống với thứ tự xuất hiện trong giao tác
- Gồm có:
  - Lịch tuần tự (Serial)
  - Lịch khả tuần tự (Serializable)
    - Conflict-Serializability
    - View-Serializability

33

VÍ DỤ

Cho lịch S của 2 giao tác T1 và T2 như sau:

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

Lịch S có thể được viết lại:
S: R1(X), R2(X), W1(X), R1(Y),W2(X), W1(Y)
Trong S chỉ quan tâm 2 hoạt động cơ sở là R và W.

Ví dụ (TT)

- Giả sử T_1 và T_2 là hai giao dịch chuyển khoản từ một tài khoản sang một tài khoản khác. Giao dịch T_1 chuyển 50$ từ tài khoản A sang tài khoản B. Giao dịch T_2 chuyển 10% số dư từ tài khoản A sang tài khoản B, và được xác định như sau:

T1:	T2:
Read(A); A:=A-50; Write(A); Read(B); B:=B+50; Write(B);	Read(A); Temp:=A*0.1; A:=A-temp; Write(A); Read(B); B:=B+temp; Write(B);

Giả sử giá trị hiện tại của A và B tương ứng là 1000$ và 2000$. Giả sử rằng hai giao dịch này được thực hiện theo trình tự:

Trường hợp 1: thực hiện xong giao dịch T_1 rồi đến giao dịch T_2

T1	T2
Read(A); A:=A-50; Write(A); Read(B); B:=B+50; Write(B);	
	Read(A); Temp:=A*0.1; A:=A-temp; Write(A); Read(B); B:=B+temp; Write(B);

S1: Giá trị sau cùng của A là 855, B là 2145, tổng 2 tài khoản (A+B) là không đổi

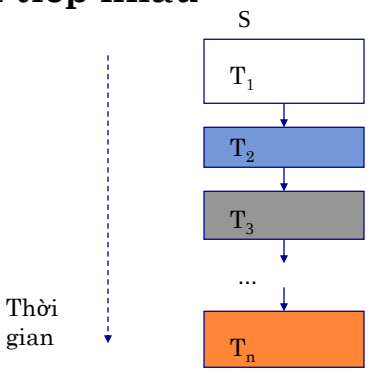
TRƯỜNG HỢP 2: THỰC HIỆN XONG GIAO DỊCH T_2 RỒI ĐẾN GIAO DỊCH T_1

T1	T2
	Read(A); Temp:=A*0.1; A:=A-temp; Write(A); Read(B); B:=B+temp; Write(B);
Read(A); A:=A-50; Write(A); Read(B); B:=B+50; Write(B);	

S2: Giá trị sau cùng của A là 850, B là 2150, tổng 2 tài khoản (A+B) là không đổi

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**



38

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 lẻ thì tính nhất quán sẽ được bảo toàn

39

Lịch tuần tự (TT)

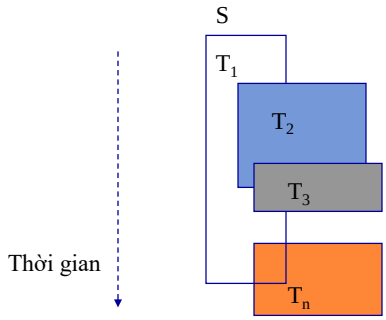
S_1 T_1	T_2	A	B
		25	25
Read(A,t)			
t:=t+100			
Write(A,t		125	
Read(B,t)			
t:=t+100			
Write(B,t)			125
	Read(A,s)		
	s:=s*2		
	Write(A,s	**250**	
	Read(B,s)		
	s:=s*2		
	Write(B,s)		**250**

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**

40

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**



41

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

42

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

43

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**

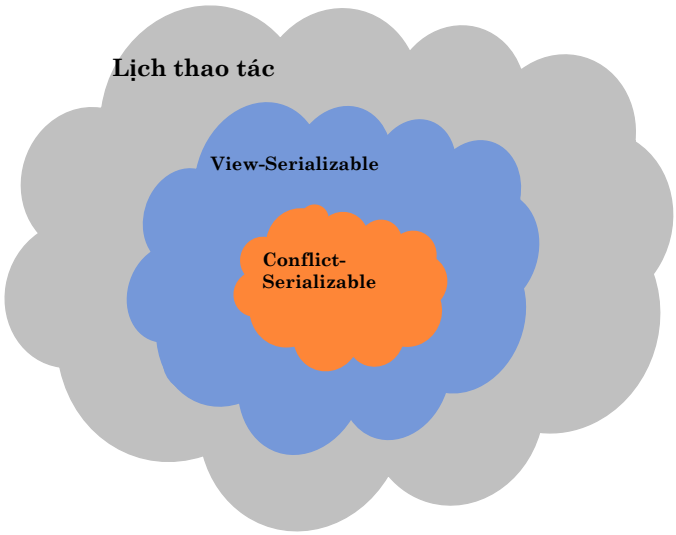
- Khi S_5 kết thúc
  - A và B bằng nhau
  - Trạng thái cuối cùng nhất quán
- S_5 khả tuần tự, ko có kết quả giống với lịch tuần tự
  - T_1, T_2
  - T_2, T_1

44

LỊCH KHẢ TUẦN TỰ (TT)

- Để xác định 1 lịch thao tác có khả tuần tự 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)

45



46

CONFLICT-SERIALIZABLE

- **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′

47

CONFLICT-SERIALIZABLE (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**

48

THAO TÁC XUNG ĐỘT

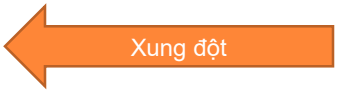
- Hai thao tác trong 1 lịch S gọi là xung đột nếu thỏa 3 điều kiện:
  - Thuộc 2 giao tác khác nhau
  - Truy xuất đến cùng một đơn vị dữ liệu
  - Ít nhất 1 trong 2 thao tác là WRITE

→ **không thể hoán vị thứ tự**

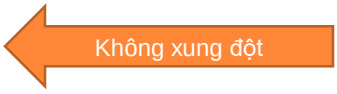
THAO TÁC XUNG ĐỘT (TT)

Ví dụ:

R1(X),W2(X)
R2(X),W1(X)
W1(X),W2(X)



R1(X), R2(X)
R1(X), W1(X)



CONFLICT-SERIALIZABLE (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 ???

51

CONFLICT-SERIALIZABLE (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)

52

LỊCH HOÀN HẢO (COMPLETE SCHEDULE)

- Lịch S của n giao tác T1,T2,…Tn được gọi là lịch hoàn hảo nếu thỏa mãn:
  - S bao gồm các thao tác trong T1,T2,… Tn và một thao tác commit hay abort ở cuối mỗi giao tác trong lịch trình.
  - Với mỗi cặp thao tác trong Ti, thứ tự xuất hiện của cũng trong S phải giống trong Ti
  - Với mỗi cặp thao tác xung đột, thì 1 trong 2 giao tác phải được thực hiện trước trong lịch

LỊCH TƯƠNG ĐƯƠNG

- Hai lịch trình của n giao tác gọi là tương đương (equivalent) nếu bất kỳ 2 thao tác xung đột nào cùng xuất hiện trong 2 lịch thì đều có cùng thứ tự.

Ví dụ:

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

T1	T2
R1(X)	
X = X-10	
W1(X)	
	R2(X)
	X = X+5
	W2(X)
R1(Y)	
Y=Y-15	
W1(Y)	

S1: R1(X), R2(X),W1(X),R1(Y),W2(X),W1(Y)
S2: R1(X),W1(X),R2(X),W2(X),R1(Y),W1(Y)

Cặp xung đột S1: R1(X)-W2(X); R2(X)-W1(X); W1(X)-W2(X)
Cặp xung đột S2: R1(X)-W2(X); W1(X)-R2(X); W1(X)-W2(X)

➔S1 và S2 là không tương đương

Ví dụ:

S1

T1	T2
R(X)	
X=X-10	
W(X)	
	R(X)
	X = X+5
R(Y)	
	W(X)
Y = Y+20	
W(Y)	

S2

T1	T2
R(X)	
X=X-10	
W(X)	
	R(X)
	X = X+5
	W(X)
R(Y)	
Y = Y+20	
W(Y)	

S1: R1(X),W1(X),R2(X),R1(Y),W2(X),W1(Y)

S1: R1(X),W1(X),R2(X), W2(X), R1(Y),W1(Y)
Cặp xung đột S1: R1(X)-W2(X); W1(X)-R2(X); W1(X)-W2(X)
Cặp xung đột S2: R1(X)-W2(X); W1(X)-R2(X),W1(X)-W2(X)
➔S1 và S2 là tương đương

NHẬN XÉT

- Lịch tuần tự đơn giản và dễ liệt kê
- Lịch khả tuần tự khó liệt kê
- Lịch tuần tự đơn giản dễ thiết lập. Nếu 1 giao tác T trong 1 lịch thực hiện nhiều hành động, nếu sử dụng lịch tuần tự thì các giao tác còn lại phải ở trạng thái chờ đợi lâu. Điều này hoàn toàn không phù hợp trong thực tế.

CÂU HỎI 1: XÉT 2 LỊCH SAU CÓ TƯƠNG ĐƯƠNG VỀ MẶT KẾT QUẢ KHÔNG?

- Giả sử x = 2, y=3

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

T1	T2
R1(X)	
X = X-10	
W1(X)	
	R2(X)
	X = X+5
	W2(X)
R1(Y)	
Y=Y-15	
W1(Y)	

CÂU HỎI 2: LỊCH SAU KHẢ TUẦN TỰ KHÔNG ?

- **Lịch S gọi là khả tuần tự nếu có kết quả như 1 lịch tuần tự. Lịch khả tuần tự là 1 lịch đúng**

T1	T2
R1(X)	
X = X-10	
W1(X)	
R1(Y)	
Y=Y-15	
W1(Y)	
	R2(X)
	X = X+5
	W2(X)

T1	T2
R1(X)	
X = X-10	
W1(X)	
	R2(X)
	X = X+5
	W2(X)
R1(Y)	
Y=Y-15	
W1(Y)	

CÂU HỎI 3: LỊCH SAU KHẢ TUẦN TỰ KHÔNG ?

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

CONFLICT-SERIALIZABLE (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

61

CONFLICT-SERIALIZABLE(TT)

- Xét trường hợp

S	T_1	T_2	T_3
	Write(Y)		
	Write(X)		
		Write(Y)	
		Write(X)	
			Write(X)

Serial

S'	T_1	T_2	T_3
	Write(Y)		
	Write(X)		
		Write(Y)	
		Write(X)	
			Write(X)

Serializable

62

KIỂM TRA CONFLICT-SERIALIZABLE

- 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

S

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)

63

KIỂM TRA CONFLICT-SERIALIZABLE (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**

64

ĐỒ THỊ TRÌNH TỰ (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

65

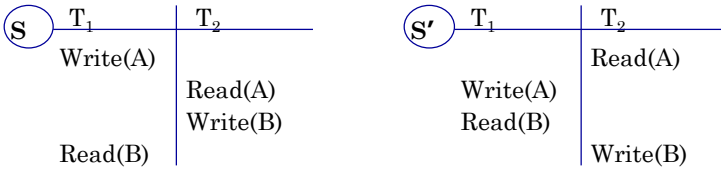
PRECEDENCE GRAPH (TT)

- 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

66

PRECEDENCE GRAPH (TT)

- Chú ý
  - $P(S_1) = P(S_2) \not\Rightarrow S_1, S_2$ conflict-equivalent
- Xét 2 trường hợp



67

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

68

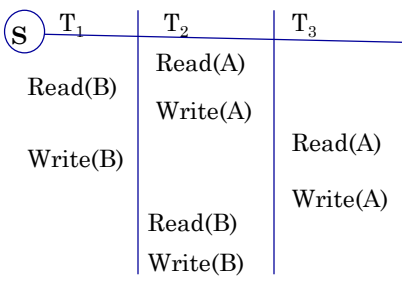
PRECEDENCE GRAPH (TT)

- Định lý
  - P(S_1) không có chu trình ⇔ S_1 conflict-serializable
- Chứng minh (⇒)
  - 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 = … $q_j(A)$ … $p_1(A)$ …
    - Đem T_1 lên vị trí đầu

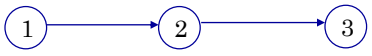
      S_1 = < hành động của T_1 ><… phần còn lại ...>
    - Lập lại quá trình này để tuần tự hoá cho phần còn lại
  - S_1 tuần tự

VÍ DỤ



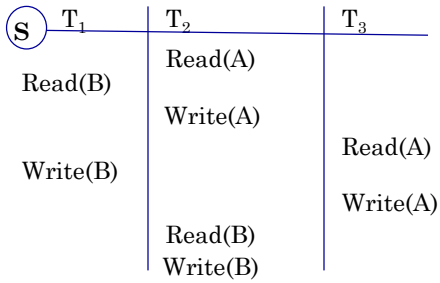
- $T_2 <_S T_3$

- $T_1 <_S T_2$



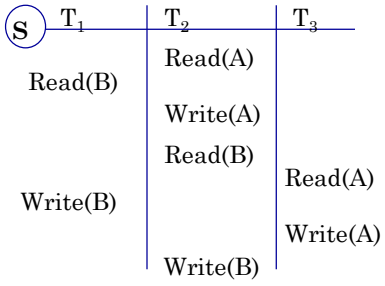
- P(S) không có chu trình
- S conflict-serializable theo thứ tự T_1, T_2, T_3

Ví dụ (TT)



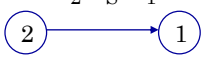
- S conflict-serializable theo thứ tự T_1, T_2, T_3

Ví dụ (TT)

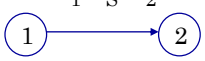


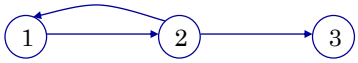
- $T_2 <_S T_3$

- $T_2 <_S T_1$



- $T_1 <_S T_2$





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

THUẬT TOÁN KIỂM TRA LỊCH S CÓ KHẢ TUẦN TỰ

- Input: Lịch S của n giao tác T1,T2,…Tn
- Output: S có khả tuần tự hay không

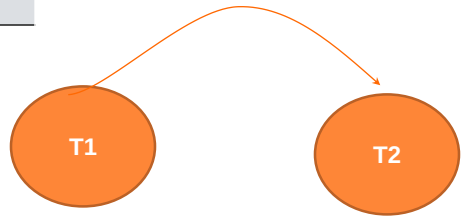
1. Với mỗi giao tác Ti tham gia vào lịch trình S, tạo 1 nút có nhãn là Ti
2. Với mỗi trường hợp S có giao tác Ti thực hiện Read (X) trước một Write(X) sau một giao tác Tj (i # j), tạo một cung (Ti➔Tj)
3. Với mỗi trường hợp S có giao tác Ti thực hiện write(X) trước một Read(X) thuộc giao tác Tj (với i#j),tạo 1 cung(Ti➔Tj)
4. Với mỗi trường hợp S có giao tác Ti thực hiện write(X) trước một write(X) thuộc giao tác Tj (với i#j), tạo 1 cung (Ti➔Tj)
5. Lịch S là khả tuần tự nếu và chỉ nếu đồ thị không có chu trình

CÂU HỎI 4:

T1	T2
R1(X)	
X = X-10	
W1(X)	
R1(Y)	
Y=Y-15	
W1(Y)	
	R2(X)
	X = X+5
	W2(X)

ĐÁP ÁN CÂU HỎI 4:

T1	T2
R1(X)	
X = X-10	
W1(X)	
R1(Y)	
Y=Y-15	
W1(Y)	
	R2(X)
	X = X+5
	W2(X)

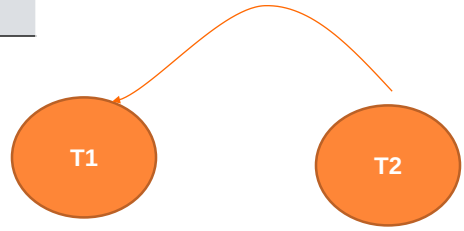


CÂU HỎI 5:

T1	T2
	R2(X)
	X = X+5
	W2(X)
R1(X)	
X = X-10	
W1(X)	
R1(Y)	
Y=Y-15	
W1(Y)	

ĐÁP ÁN CÂU HỎI 5:

T1	T2
	R2(X)
	X = X+5
	W2(X)
R1(X)	
X = X-10	
W1(X)	
R1(Y)	
Y=Y-15	
W1(Y)	

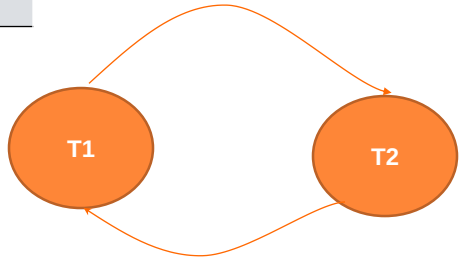


CÂU HỎI 6:

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

ĐÁP ÁN CÂU HỎI 6:

T1	T2
R1(X)	
X = X-10	
	R2(X)
	X = X+5
W1(X)	
R1(Y)	
	W2(X)
Y=Y-15	
W1(Y)	

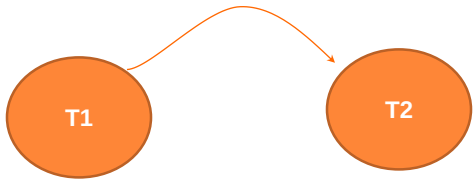


CÂU HỎI 7:

T1	T2
R1(X)	
X = X-10	
W1(X)	
	R2(X)
	X = X+5
	W2(X)
R1(Y)	
Y=Y-15	
W1(Y)	

ĐÁP ÁN CÂU HỎI 7:

T1	T2
R1(X) X = X-10 W1(X)	
	R2(X) X = X+5 W2(X)
R1(Y) Y=Y-15 W1(Y)	



VẬN DỤNG TÍNH KHẢ TUẦN TỰ TRONG THỰC TẾ:

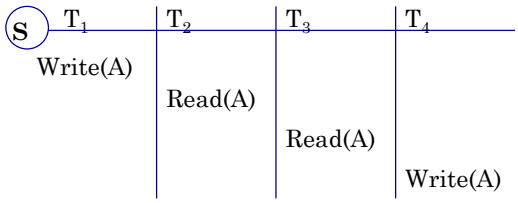
- Các lịch trình tuần tự kém hiệu quả do không cho phép giao tác thi hành xen kẻ
- Các lịch trình không khả tuần tự tiềm tàng khả năng gây các vấn đề bất thường
- Các lịch trình khả tuần tự cho phép giao tác thi hành xen kẻ và cho kết quả đúng

Tuy nhiên, việc kiểm tra tính khả tuần tự của một lịch trình thực tế rất khó khăn, do các nguyên nhân:

- Các giao tác, khi thi hành thường thể hiện dưới dạng những tiến trình của HĐH, thường được bản thân HĐH điều phối. HQTCSDL thực tế không điều phối được các thao tác trong giao tác.
- Các giao tác được gửi tới HQTCSDL liên tục, khó xác định điểm bắt đầu và kết thúc giao tác.

➔Các HQTCSDL thường sử dụng những quy tắc để đảm bảo tính khả tuần tự của lịch trình.

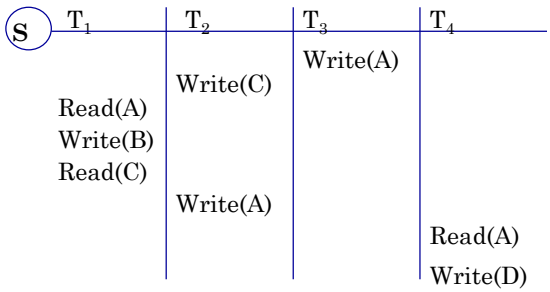
CÂU HỎI 8



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

83

CÂU HỎI 9

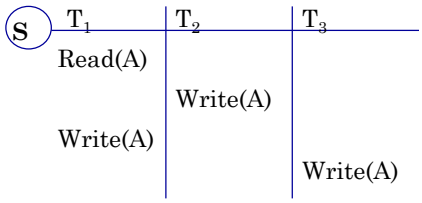


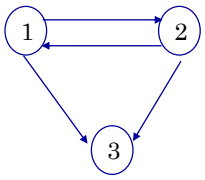
- Vẽ P(S)
- S có conflict-serializable không?

84

VIEW-SERIALIZABILITY

- Xét lịch S



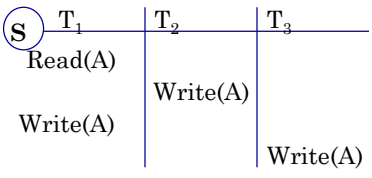


- P(S) có chu trình
- S không conflict-serializable

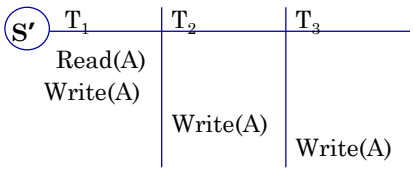
85

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

Giải thích như thế nào đây?

86

VIEW-SERIALIZABILITY (TT)

- Ý tưởng
  - Xét trường hợp

T	U
Write(A)	
	Read(A)

  - 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

87

VIEW-SERIALIZABILITY (TT)

- Định nghĩa
  - S, S' là những lịch thao tác view-equivalent
    - 1- Nếu trong S có $w_j(A)...r_i(A)$ thì trong S' cũng có $w_j(A)... r_i(A)$
    - 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 conflict-serializable → S view-serializable
- S conflict-serializable ↚ S view-serializable???

88

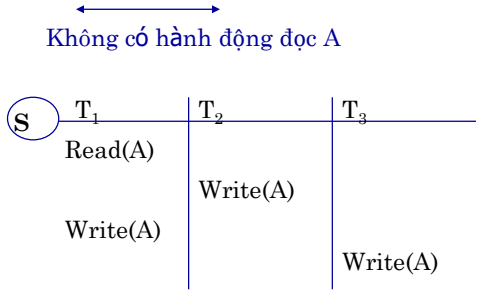
VIEW-SERIALIZABILITY (TT)

- **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**

89

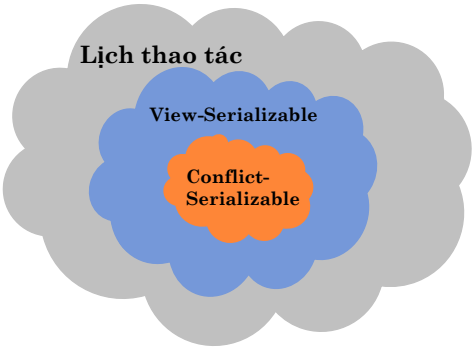
VIEW-SERIALIZABILITY (TT)

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



90

VIEW-SERIALIZABILITY (TT)



91

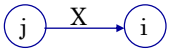
KIỂM TRA VIEW-SERIALIZABILITY (TT)

- 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 = … $w_1(A)$……$w_2(A)$ **$r_f(A)$**
    ↑ 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 = **$w_b(A)$**… $w_1(A)$……$w_2(A)$…

92

KIỂM TRA VIEW-SERIALIZABILITY (TT)

- **Vẽ đồ thị trình tự gán nhãn cho S, ký hiệu LP(S), (Labeled Prececence Graph)**
  - **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**
      $w_j(X)$ ... $r_i(X)$

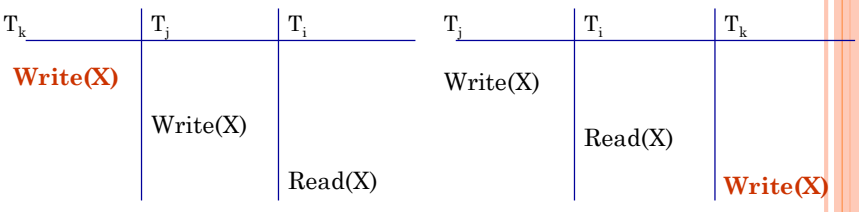


    - **(2) Với mỗi $w_j(X)$ ... $r_i(X)$, xét $w_k(X)$ sao cho T_k không chèn vào giữa T_j và T_i**

93

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 $T_i \rightarrow T_k$

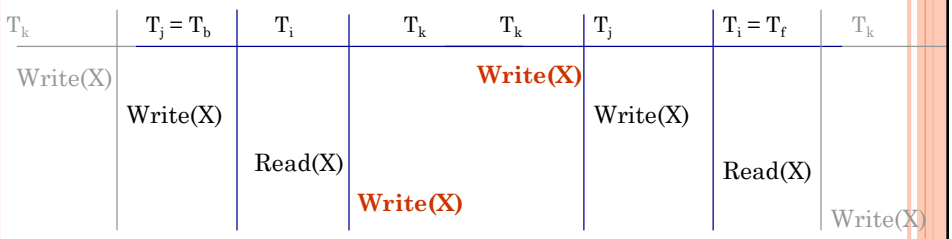


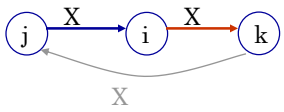
Chọn 1 cung vừa tạo sao cho đồ thị không có chu trình

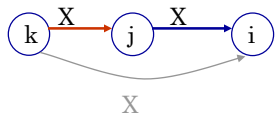
94

KIỂM TRA VIEW-SERIALIZABILITY (TT)

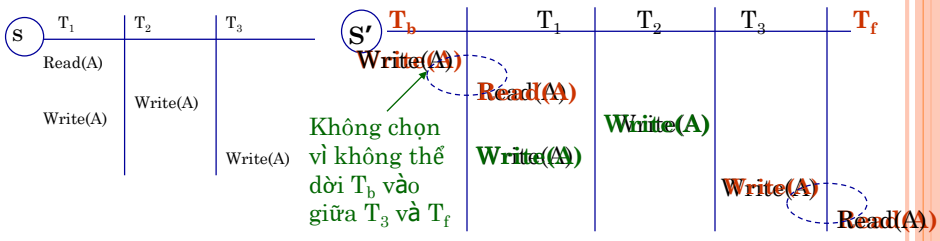
- (2b) Nếu $T_j = T_b$ thì vẽ cung $T_i \rightarrow T_k$
- (2c) Nếu Ti = Tf thì vẽ cung $T_k \rightarrow T_j$



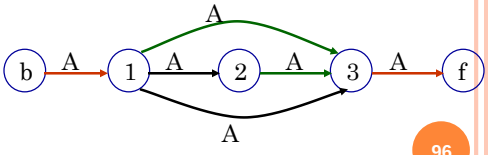




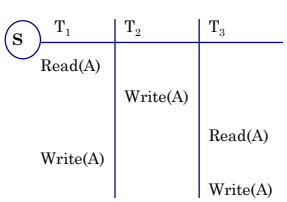
VÍ DỤ

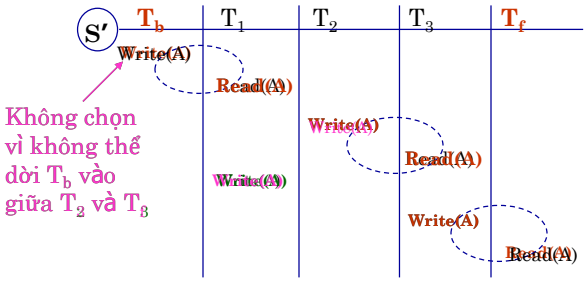


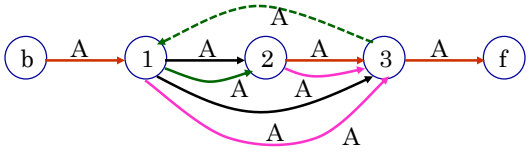
- LP(S) không có chu trình
- S view-serializable theo thứ tự T_b, T_1, T_2, T_3, T_f



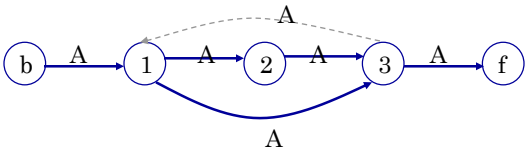
Ví dụ (TT)







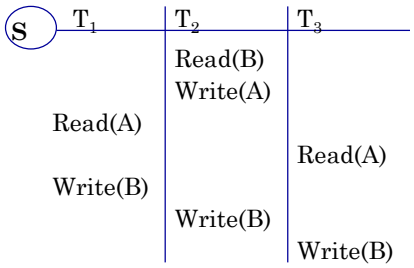
Ví dụ (TT)



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

98

CÂU HỎI 10



- Vẽ LP(S)
- S có view-serializable?

99

ĐÁP ÁN CÂU HỎI 10

Tb	T1	T2	T3	Tf
W(A)				
W(B)				
		R(B)		
		W(A)		
	R(A)			
			R(A)	
	W(B)			
		W(B)		
			W(B)	
				R(A)
				R(B)

100