

Student name:

Student ID: 19520113

Class:

(Đáp án chỉ mang tính chất tham khảo)

Question 2

Hãy vẽ đồ thị trình tự gán nhãn cho các lịch thao tác sau và tìm xem có những lịch nào là view-serializable:

c) r1(A); r3(D); w1(B); r2(B); w3(B); r4(B); w2(c); r5(C); w4(E); r5(E); w5(B);

d) w1(A); r2(A); w3(A); r4(A); w5(A); r6(A);

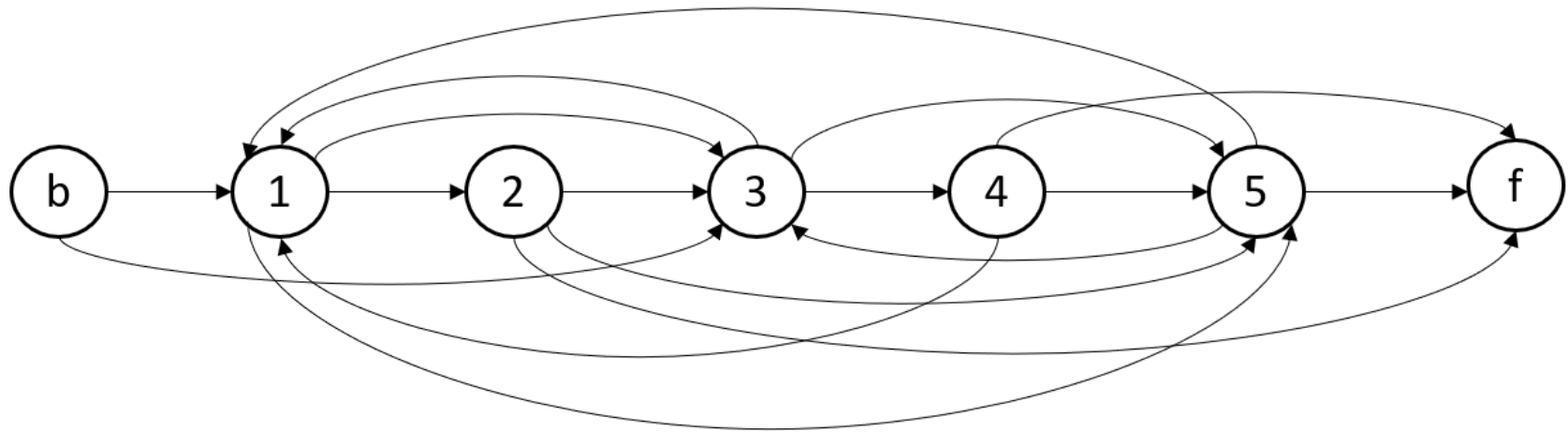
Answer:

c.

Tb	T1	T2	T3	T4	T5	Tf
W(A)						
W(B)						
W(C)						
W(D)						
W(E)						
	R(A)					

			R(D)			
	W(B)					
		R(B)				
			W(B)			
				R(B)		
		W(C)				
					R(C)	
				W(E)		
					R(E)	
					W(B)	
						R(A)
						R(B)
						R(C)
						R(D)
						R(E)

Poly graph for schedule S:



The poly graph for schedule S contains cycles.

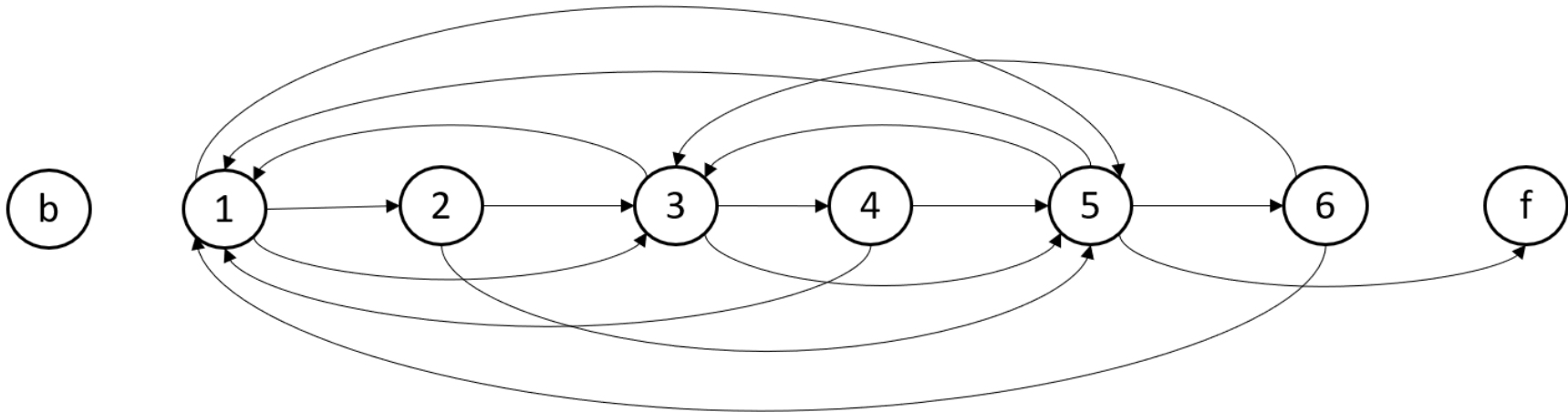
→ Schedule S is not view-serializable.

d.

Tb	T1	T2	T3	T4	T5	T6	Tf
W(A)							
	W(A)						
		R(A)					
			W(A)				
				R(A)			
					W(A)		

						R(A)	
							R(A)

Poly graph for schedule S:



The poly graph for schedule S contains cycles.

→ Schedule S is not view-serializable.

Question 3:

Cho lịch thao tác sau:

	T1	T2	T3	T4
1		Read(A)		
2			Read (A)	
3		Write(B)		
4			Write(A)	
5	Read(B)			
6				Read (B)
7	Read (A)			
8	Write(C)			
9				Write(A)

3.1 Hãy xét tính khả tuần tự của lịch thao tác này.

3.2 Dùng kỹ thuật timestamp từng phần để điều khiển truy xuất đồng thời của 4 giao tác trên, với timestamp của các giao tác T1, T2, T3, T4 lần lượt là:

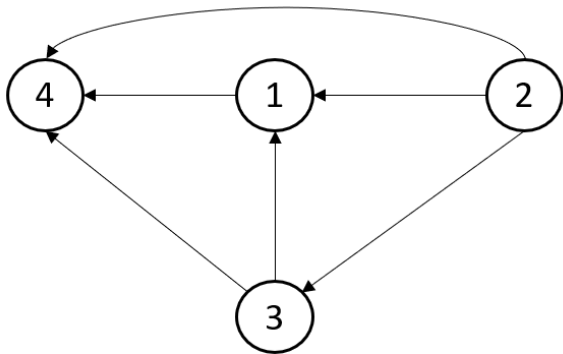
a) 300, 310, 320, 330

b) 250, 200, 210, 275

Trong mỗi trường hợp hãy cho biết RT và WT của 3 đơn vị dữ liệu chứa A, B, C.

Answer:

3.1. Precedence graph for schedule S:



The precedence graph for schedule S has no cycles.

→ Schedule S is conflict-serializable.

3.2.a. *Partial timestamp technique

	T1 TS = 300	T2 TS = 310	T3 TS = 320	T4 TS = 330	A RT = 0 WT = 0	B RT = 0 WT = 0	C RT = 0 WT = 0	
1		R(A)			RT = 310 WT = 0			WT(A) ≤ TS(T2) → T2 read A
2			R(A)		RT = 320 WT = 0			WT(A) ≤ TS(T3) → T3 read A
3		W(B)				RT = 0 WT = 310		RT(B) ≤ TS(T2) WT(B) ≤ TS(T2) → T2 write B
4			W(A)		RT = 320 WT = 320			RT(A) ≤ TS(T3) WT(A) ≤ TS(T3) → T3 write A
5	R(B)							WT(B) > TS(T1) → Roll back T1

Re-initialize T1 so that $TS(T1) > TS(T4)$. Assume $TS(T1) = 340$.

	T1 TS = 340	T2 TS = 310	T3 TS = 320	T4 TS = 330	A RT = 0 WT = 0	B RT = 0 WT = 0	C RT = 0 WT = 0	
1		R(A)			RT = 310 WT = 0			WT(A) ≤ TS(T2) → T2 read A
2			R(A)		RT = 320 WT = 0			WT(A) ≤ TS(T3) → T3 read A
3		W(B)				RT = 0 WT = 310		RT(B) ≤ TS(T2) WT(B) ≤ TS(T2) → T2 write B
4			W(A)		RT = 320 WT = 320			RT(A) ≤ TS(T3) WT(A) ≤ TS(T3) → T3 write A
5				R(B)		RT = 330 WT = 310		WT(B) ≤ TS(T4) → T4 read B
6				W(A)	RT = 320 WT = 330			RT(A) ≤ TS(T4) WT(A) ≤ TS(T4) → T4 write B
7	R(B)					RT = 340 WT = 310		WT(B) ≤ TS(T1) → T1 read B
8	R(A)				RT = 340 WT = 330			WT(A) ≤ TS(T1) → T1 read A
9	W(C)						RT = 0 WT = 340	RT(C) ≤ TS(T1) WT(C) ≤ TS(T1) → T1 write C

⇒ Schedule S is serializable in the order T2, T3, T4, T1.

3.2.b. *Total timestamp technique

	T1 TS = 250	T2 TS = 200	T3 TS = 210	T4 TS = 275	A TS = 0	B TS = 0	C TS = 0	
1		R(A)			TS = 200			TS(A) <= TS(T2) → T2 read A
2			R(A)		TS = 210			TS(A) <= TS(T3) → T3 read A
3		W(B)				TS = 200		TS(B) <= TS(T2) → T2 write B
4			W(A)		TS = 210			TS(A) <= TS(T3) → T3 write A
5	R(B)					TS = 250		TS(B) <= TS(T1) → T1 read B
6				R(B)		TS = 275		TS(B) <= TS(T4) → T4 read B
7	R(A)				TS = 250			TS(A) <= TS(T1) → T1 read A
8	W(C)						TS = 250	TS(C) <= TS(T1) → T1 write C
9				W(A)	TS = 275			TS(A) <= TS(T4) → T4 write A

⇒ Schedule S is serializable in the order T2, T3, T1, T4.

Question 4:

Cho lịch S như sau:

	T1	T2	T3	T4
1	Rlock A			
2		Rlock A		
3	Unlock A			
4			Wlock B	
5		Unlock A		
6				Wlock A
7			Unlock B	
8	Rlock B			
9				Wlock C
10	Unlock B			
11		Wlock B		
12				Unlock A
13			Wlock A	
14		Unlock B		
15			Unlock A	
16				Unlock C

- a) Trong các giao tác trong lịch trên giao tác nào viết đúng nghi thức khóa hai giai đoạn?
- b) Lịch S có khả tuần tự không? Nếu có thì tương đương với lịch tuần tự nào?
- c) Thay Rlock bởi Read, thay Wlock bởi Write, bỏ qua các thao tác Unlock. Biết các timestamp của các giao tác là $t(T1) = 100$, $t(T2) = 200$, $t(T3) = 300$, $t(T4) = 400$.

Hãy điều khiển việc truy xuất đồng thời của các giao tác dùng:

- Kỹ thuật timestamp toàn phần
- Kỹ thuật timestamp từng phần

Answer:

a.

Consider transaction T1: $RL_1(A) \dots UL_1(A) \dots RL_1(B) \dots UL_1(B) \dots$

→ Rlock(B) executes after Unlock(A) → T1 does not satisfy 2PL.

Consider transaction T2: ...RL₂(A)...UL₂(A)... WL₂(B)... UL₂(B)...

→ Wlock(B) executes after Unlock(A) → T2 does not satisfy 2PL.

Consider transaction T3: ...WL₃(B)...UL₃(B)... WL₃(A)... UL₃(A)...

→ Wlock(A) executes after Unlock(B) → T3 does not satisfy 2PL.

Consider transaction T4: ...WL₄(A)...WL₄(C)... UL₄(A)... UL₄(C)

← no unlock → no lock

→ T4 satisfy 2PL.

b.

- Consider transaction T1:

+ Pair(8, 11) \Rightarrow T1 \rightarrow T2.

+ Pair(1, 13) \Rightarrow T1 \rightarrow T3.

+ Pair(1, 6) \Rightarrow T1 \rightarrow T4.

- Consider transaction T2:

+ Pair(2, 13) \Rightarrow T2 \rightarrow T3.

+ Pair(2, 6) \Rightarrow T2 \rightarrow T4.

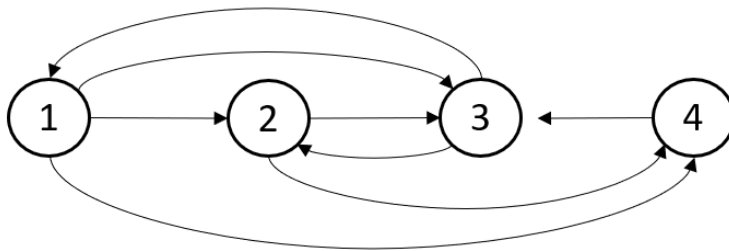
- Consider transaction T3:

+ Pair(4, 7, 8) \Rightarrow T3 \rightarrow T1.

+ Pair(4, 11) \Rightarrow T3 \rightarrow T2.

- Consider transaction T4:

+ Pair(6, 13) \Rightarrow T4 \rightarrow T3.



The graph for S has a cycle, so schedule S is not serializable

Sequential schedules equivalent to S1 is T1 \rightarrow T2 \rightarrow T3 \rightarrow T4 (COI LẠI)

b.

*Partial timestamp technique

*Total timestamp technique

	T1 TS = 100	T2 TS = 200	T3 TS = 300	T4 TS = 400	A TS = 0	B TS = 0	C TS = 0	
1	R(A)				TS = 100			TS(A) ≤ TS(T1) → T1 read A
2		R(A)			TS = 200			TS(A) ≤ TS(T2) → T2 read A
3			W(B)			TS = 300		TS(B) ≤ TS(T3) → T3 write B
4				W(A)	TS = 400			TS(A) ≤ TS(T4) → T4 write A
5	R(B)							TS(B) > TS(T1) → Abort(T1)

Reset TS(T1) so that TS(T1) > TS(T4). Assume TS(T1) = 500.

	T1 TS = 500	T2 TS = 200	T3 TS = 300	T4 TS = 400	A TS = 0	B TS = 0	C TS = 0	
1		R(A)			TS = 200			TS(A) ≤ TS(T2) → T2 read A
2			W(B)			TS = 300		TS(B) ≤ TS(T3) → T3 write B
3				W(A)	TS = 400			TS(A) ≤ TS(T4) → T4 write A
4				W(C)			TS = 400	TS(C) ≤ TS(T4) → T4 write C
5		W(B)						TS(B) > TS(T2) → Abort(T2)

Reset TS(T2) so that TS(T2) > TS(T1). Assume TS(T2) = 600.

	T1 TS = 500	T2 TS = 600	T3 TS = 300	T4 TS = 400	A TS = 0	B TS = 0	C TS = 0	
1			W(B)			TS = 300		TS(B) ≤ TS(T3) → T3 write B
2				W(A)	TS = 400			TS(A) ≤ TS(T4) → T4 write A
3				W(C)			TS = 400	TS(C) ≤ TS(T4) → T4 write C
4			W(A)					TS(C) > TS(T3) → Abort(T3)

Reset $TS(T3)$ so that $TS(T3) > TS(T2)$. Assume $TS(T3) = 700$.

	T1 TS = 500	T2 TS = 600	T3 TS = 700	T4 TS = 400	A TS = 0	B TS = 0	C TS = 0	
1				W(A)	TS = 400			$TS(A) \leq TS(T4) \rightarrow T4 \text{ write A}$
2				W(C)			TS = 400	$TS(C) \leq TS(T4) \rightarrow T4 \text{ write C}$
3	R(A)				TS = 500			$TS(A) \leq TS(T1) \rightarrow T1 \text{ read A}$
4	R(B)					TS = 500		$TS(B) \leq TS(T1) \rightarrow T1 \text{ read B}$
5		R(A)			TS = 600			$TS(A) \leq TS(T2) \rightarrow T2 \text{ read A}$
6		W(B)				TS = 600		$TS(B) \leq TS(T2) \rightarrow T2 \text{ write B}$
7			W(B)					$TS(B) \leq TS(T3) \rightarrow T3 \text{ write B}$
8			W(A)					$TS(A) \leq TS(T3) \rightarrow T3 \text{ write A}$

\Rightarrow Schedule S is serializable in the order T4, T1, T2, T3.

Question 5:

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:

<checkpoint>

<start T1>

<T1, A, 30, 40>

<T1, B, 20, 10>

<start T2>

<T2, C, 10, 15>

<start T3>

<T3, D, 10, 20>

<commit T3>

<T2, C, 15, 40>

<T2, D, 20, 40>

Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp Undo/Redo logging.

Answer:

<checkpoint>	↑ scan
<start T1>	
<T1, A, 30, 40>	
<T1, B, 20, 10>	
<start T2>	
<T2, C, 10, 15>	
<start T3>	
<T3, D, 10, 20>	
<commit T3>	
<T2, C, 15, 40>	
<T2, D, 20, 40>	

<T2, D, 20, 40>

T2 is incompleted → Recover D = 20.

<T2, C, 15, 40>

T2 is incompleted → Recover C = 15.

<commit T3>

T3 is completed

<T3, D, 10, 20>

Recover D = 20.

<T2, C, 10, 15>

T2 is incompleted → Recover C = 10.

<T1, B, 20, 10>

T1 is incompleted → Recover B = 20.

<T1, A, 30, 40>

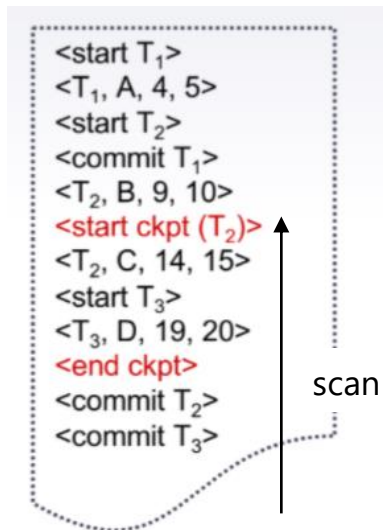
T1 is incompleted → Recover A = 30.

Reach <checkpoint>

Stop.

Question 5:

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:



Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp Undo/Redo logging.

Answer:

<commit T3>

T3 is completed.

<commit T2>

T2 is completed.

Found <end ckpt>

Do nothing with T1

<T3, D, 19, 20>

Recover D = 20

<T2, C, 14, 15>

Recover C = 15

Reach <start ckpt>

Stop.

FILE BÀI TẬP DẠNG 3

Ghi chú:

Trong các bài tập dưới đây, tính khả tuần tự được mặc định là conflict-serializable.

Bài tập 3.1

Cho lịch thao tác sau:

	T1	T2	T3	T4
1		Read(A)		
2			Read (A)	
3		Write(B)		
4			Write(A)	
5	Read(B)			
6				Read (B)
7	Read (A)			
8	Write(C)			
9				Write(A)

Hãy xét tính khả tuần tự của lịch thao tác này với dữ liệu B, C được lưu trên cùng một đơn vị dữ liệu, A được lưu trên đơn vị dữ liệu khác.

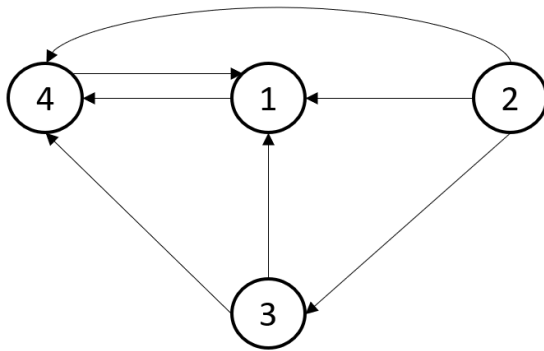
Answer:

Call the data unit B and C is X, we have:

T1	T2	T3	T4
	R(A)		
		R(A)	
	W(X)		
		W(A)	
R(X)			
			R(X)
R(A)			
W(X)			

			W(A)
--	--	--	------

Precedence graph for schedule S:



The precedence graph for schedule S contains a cycles.

→ Schedule S is not conflict-serializable.

Bài tập 3.3

Cho lịch S như sau

	T1	T2	T3	T4
1	RL(A)			
2			RL(B)	
3	RL(C)			
4	UL(C)			
5		WL(C)		
6		UL(C)		
7				WL(C)
8	UL(A)			
9				WL(A)
10				UL(A)
11			WL(A)	
12			UL(B)	
13		RL(B)		
14		UL(B)		
15			WL(B)	
16			UL(A)	
17			UL(B)	
18				UL(C)

- Các giao tác T1, T2, T3, T4 có thỏa nghi thức khóa hai giai đoạn không?
- S có khả tuần tự không? Nếu có thì S tương đương với lịch khả tuần tự nào?
- Giả sử trong lịch S trên bỏ các UL, thay các RL thành Read, các WL thành Write và lịch được thực hiện theo kỹ thuật timestamp. Hãy cho biết các bước thực hiện các giao tác trên nếu các timestamps của các giao tác như sau : T1=100, T2=300, T3=200, T4=400 (thực hiện với kỹ thuật timestamp riêng phần nhiều phiên bản).

Answer:

a.

Consider transaction T1: $RL_1(A) \dots RL_1(C) UL_1(C) \dots UL_4(A)$

$\xleftarrow{\text{no unlock}} \qquad \qquad \qquad \xrightarrow{\text{no lock}}$

→ T1 satisfy 2PL.

Consider transaction T2: $\dots WL_2(C) \dots UL_2(C) \dots RL_2(B) \dots UL_2(B) \dots$

→ Rlock(B) executes after Unlock(C) → T2 does not satisfy 2PL.

Consider transaction T3: $\dots RL_3(B) \dots WL_3(A) UL_3(B) \dots WL_3(B) UL_3(A) UL_3(B) \dots$

→ Wlock(B) executes after Unlock(B) → T3 does not satisfy 2PL.

Consider transaction T4: $\dots WL_4(C) \dots WL_4(A) UL_4(A) \dots UL_4(C)$

$\xleftarrow{\text{no unlock}} \qquad \qquad \qquad \xrightarrow{\text{no lock}}$

→ T4 satisfy 2PL.

b.

c.

T1	T2	T3	T4	A	B	C				
TS = 100	TS = 300	TS = 200	TS = 400	RT = 0 WT = 0	RT = 0 WT = 0	RT = 0 WT = 0	C1	C2	A1	
R(A)				RT = 100						WT(A0) < TS(T1) → T1 read A

				WT = 0						
		R(B)			RT = 200 WT = 0					WT(B0) < TS(T3) → T3 read B
R(C)						RT = 100 WT = 0				WT(C0) < TS(T1) → T1 read C0
	W(C)						RT = 0 WT = 300			WT(C0) < TS(T2) RT(C0) < TS(T2) → Create version C1 → T2 write C1
			W(C)					RT = 0 WT = 400		WT(C1) < TS(T2) RT(C1) < TS(T2) → Create version C2 → T4 write C2
			W(A)						RT = 0 WT = 400	WT(A0) < TS(T4) RT(A0) < TS(T4) → Create version A1 → T4 write A1
		W(A)							RT = 0 WT = 300	WT(A1) > TS(T3) → i = 0 WT(A0) < TS(T3) RT(A0) < TS(T3)

										→ Create version A1 → T3 write A1
	R(B)				RT = 300 WT = 0					WT(B0) < TS(T2) → T1 read C0
		W(B)								WT(B0) < TS(T3) RT(B0) > TS(T3) → Rollback T3.

Bài tập 3.5

Cho A=1, B=2, C=1, D=2, E=3

- Dùng đồ thị chờ để đánh giá có deadlock hay không?
- Nếu có deadlock, hãy đưa ra 1 giải pháp cụ thể để tránh và 1 giải pháp để giải quyết.
- Cho biết các giá trị của A, B, C, D, E ứng với các giải pháp này sau khi kết thúc các Ti.

	T1	T2	T3
1	Rlock A		
2	S1:= A		
3		Rlock C	
4		S2:= C+1	

5			Wlock E
6			E:=E-1
7	Wlock B		
8	B:= S1 + B		
9		Rlock B	
10		S2:= S2-B	
11			Rlock B
12			S3:=B+1
13	Wlock C		
14	C:= C+1		
15		Wlock E	
16		E:=S2	
17		Rlock D	
18		Print D	
19			Wlock C
20			C := S3
(...)	Unlock ...	Unlock ...	Unlock ...

Answer:

a.

- Considering the data unit A: There are no pairs.

- Considering the data unit B:

+ $\text{Pair}(7, 9) \Rightarrow T2 \rightarrow T1.$

+ $\text{Pair}(7, 11) \Rightarrow T3 \rightarrow T1.$

- Considering the data unit C:

+ $\text{Pair}(3, 13) \Rightarrow T1 \rightarrow T2.$

+ $\text{Pair}(3, 19) \Rightarrow T3 \rightarrow T2.$

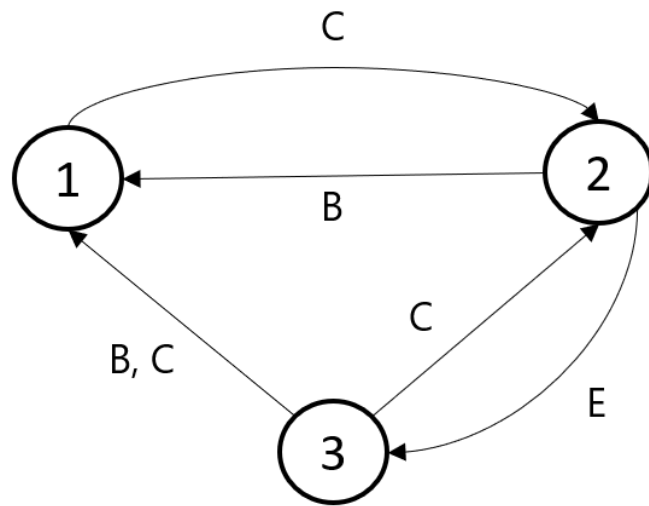
+ $\text{Pair}(13, 19) \Rightarrow T3 \rightarrow T1.$

- Considering the data unit D: There are no pairs.

- Considering the data unit E: There are no pairs.

+ $\text{Pair}(5, 15) \Rightarrow T2 \rightarrow T3.$

The wait graph for so schedule S:



The wait graph has cycles, so schedule S has a deadlock.

b.

*Method 1:

Choose T2 to rollback. T2 will release the lock on data unit C.

T1, T3 will get lock on C.

Suppose T1 receives a lock on C. T1 completes and terminates. T1 releases the lock on B and C.

T3 receives the lock on B and C, T3 completes and terminates. T3 releases the lock on B, C and E.

T2 will re-execute at some point and get the lock on B, C, and E when T1, T3 complete.

⇒ Schedule S is serializable in the order T1, T3, T2.

*Method 2: Avoid Deadlock – wait-die solution.

C.

After T1 ends B = 3, C = 2.

After T3 ends E = 2, C = 4.

After T2 ends E = 2.

Conclusion: A = 1, B = 3, C = 4, D = 2, E = 2.

BÀI TẬP LÝ THUYẾT DẠNG 4:

Bài tập 4.1

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:

```
<checkpoint>
<start T1>
<T1, A, 30, 40>
<T1, B, 20, 10>
<start T2>
<T2, C, 10, 15>
<start T3>
<T3, D, 10, 20>
<commit T3>
<T2, C, 15, 40>
<T2, D, 20, 40>
```

Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp Undo/Redo logging.

Answer:

<checkpoint>	↑ scan
<start T1>	
<T1, A, 30, 40>	
<T1, B, 20, 10>	
<start T2>	
<T2, C, 10, 15>	
<start T3>	
<T3, D, 10, 20>	
<commit T3>	
<T2, C, 15, 40>	
<T2, D, 20, 40>	

<T2, D, 20, 40>

T2 is incompleted → Recover D = 20.

<T2, C, 15, 40>

T2 is incompleted → Recover C = 15.

<commit T3>

T3 is completed.

<T3, D, 10, 20>

Recover D = 20.

<T2, C, 10, 15>

T2 is incompleted → Recover C = 10.

<T1, B, 20, 10>

T1 is incompleted → Recover B = 20.

<T1, A, 30, 40>

T1 is incompleted → Recover A = 30.


Reach <checkpoint>

Stop.

Bài tập 4.2

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:

```
<start T1>
<T1, A, 30, 40>
<start T2>
<T2, B, 40, 60>
<T1, C, 20, 30>
<commit T2>
<start T3>
<T3, B, 60, 50>
<commit T3>
```



scan

Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp Undo/Redo logging.

Answer:

<commit T3>

T3 is completed.

<T3, B, 60, 50>

Recover B = 50.

<commit T2>

T2 is completed.

<T1, C, 20, 30>

T1 is incompleted → Recover C = 20.

<T2, B, 40, 60>

Recover B = 60.

<T1, A, 30, 40>

T1 is incompleted → Recover A = 30.

<start T1>

Stop.

Bài tập 4.3

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:

<start ckpt (T1, T2, T3)>

<end ckpt>

<T1, A, 10>

<T2, B, 20>

<T3, C, 30>

<commit T2>

<T3, B, 40>

<T1, D, 50>

<abort T3>

Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp:

a. Undo logging


b. Redo logging

Answer:

```

<start ckpt (T1, T2, T3)>
<end ckpt>
<T1, A, 10>
<T2, B, 20>
<T3, C, 30>
<commit T2>
<T3, B, 40>
<T1, D, 50>
<abort T3>

```



scan

a.

```
<abort T3>
```

T3 is completed.

```
<T1, D, 50>
```

T1 is incompleted → Recover D = 30.

```
<commit T2>
```

T2 is completed.

```
<T1, A, 10>
```

T1 is incompleted → Recover A = 10.

Reach <start ckpt (T1, T2, T3)>

Stop.

b.

```
<abort T3>
```

T3 is completed.

```
<commit T2>
```

T2 is completed.

```
<T2, B, 20>
```

Recover B = 30.

Reach <start ckpt (T1, T2, T3)>

Stop.

Bài tập 4.4

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:

```
<start ckpt (T1, T2, T3)>
<T1, A, 10>
<commit T1>
<T2, B, 20>
<abort T2>
<T3, C, 30>
<end ckpt>
<commit T3>
```

Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp:

a. Undo logging

b. Redo logging

Answer:

a.

```
<commit T3>
```

T3 is completed

```
<abort T2>
```

T2 is completed

<commit T1>

T1 is completed

Reach <start ckpt (T1, T2, T3)>

Stop.

b.

<commit T3>

T3 is completed

<T3, C, 30>

Recover C = 30.

<abort T2>

T2 is completed

<commit T1>

T1 is completed

<T1, A, 10>

Recover A = 10.

Reach <start ckpt (T1, T2, T3)>

Stop.

Bài tập 4.5

Giả sử sau khi sự cố hệ thống xảy ra, DBMS được khởi động lại với tập tin nhật ký như sau:

<T1, A, 10>

<T1, B, 20>

<T2, D, 30>

<T3, C, 40>

<commit T3>

<T2, D, 50>

<T2, C, 60>

<abort T2>

Hãy mô tả tiến trình khôi phục của DBMS dựa trên tập tin nhật ký này theo phương pháp:

a. Undo logging

b. Redo logging

Answer:

a.

<abort T2>

T2 is completed.

<commit T3>

T3 is completed.

<T1, B, 20>

T1 is incompleted → Recover B = 20.

<T1, A, 10>

T1 is incompleted → Recover B = 20.

b.

<abort T2>

T2 is completed.

<commit T3>

T3 is completed.

<T3, C, 40>

Recover C = 40.

Bài tập 4.6

Cho tập tin nhật ký sau:

```
01) <start T1>
02) <T1, A, 60>
03) <commit T1>
04) <start T2>
05) <T2, A, 10>
06) <start T3>
07) <T3, B, 20>
08) <T2, C, 30>
09) <start T4>
10) <T3, D, 40>
11) <T4, F, 70>
12) <commit T3>
13) <T2, E, 50>
14) <commit T2>
15) <T4, B, 80>
16) <commit T4>
```

Giả sử đặt một điểm lưu trữ (nonquiescent checkpoint) sau các bước dưới đây thì mẫu tin <end ckpt> có thể được ghi xuống bộ nhớ khi nào?

- a. <T1, A, 60>
- b. <T2, A, 10>
- c. <T3, B, 20>
- d. <T3, D, 40>
- e. <T2, E, 50>

Giả sử sự cố hệ thống xảy ra ngay sau các bước trên thì tiến trình khôi phục của DBMS như thế nào khi dùng phương pháp:

a. Undo logging

b. Redo logging

Answer:

Đặt checkpoint sau bước <T1, A, 60> thì mẫu tin <end ckpt> có thể được ghi xuống bộ nhớ sau mẫu tin <commit T1>

```
01) <start T1>
02) <T1, A, 60>
    <start ckpt>
03) <commit T1>
    <end ckpt>
04) <start T2>
05) <T2, A, 10>
06) <start T3>
07) <T3, B, 20>
08) <T2, C, 30>
09) <start T4>
10) <T3, D, 40>
11) <T4, F, 70>
12) <commit T3>
13) <T2, E, 50>
14) <commit T2>
15) <T4, B, 80>
16) <commit T4>
```

Đặt checkpoint sau bước <T2, A, 10> thì mẫu tin <end ckpt> có thể được ghi xuống bộ nhớ sau khi <commit T2>

```
01) <start T1>
02) <T1, A, 60>
03) <commit T1>
04) <start T2>
05) <T2, A, 10>
<start ckpt>
06) <start T3>
07) <T3, B, 20>
08) <T2, C, 30>
09) <start T4>
10) <T3, D, 40>
11) <T4, F, 70>
12) <commit T3>
13) <T2, E, 50>
14) <commit T2>
<end ckpt>
15) <T4, B, 80>
16) <commit T4>
```

Đặt checkpoint sau bước <T3, B, 20> thì mẫu tin <end ckpt> có thể được ghi xuống bộ nhớ sau khi <commit T3>

```
01) <start T1>
02) <T1, A, 60>
03) <commit T1>
04) <start T2>
05) <T2, A, 10>
```

```
06) <start T3>
07) <T3, B, 20>
<start ckpt>
08) <T2, C, 30>
09) <start T4>
10) <T3, D, 40>
11) <T4, F, 70>
12) <commit T3>
<end ckpt>
13) <T2, E, 50>
14) <commit T2>
15) <T4, B, 80>
16) <commit T4>
```

Đặt checkpoint sau bước <T3, D, 40> thì mẫu tin <end ckpt> có thể được ghi xuống bộ nhớ sau khi <commit T3>

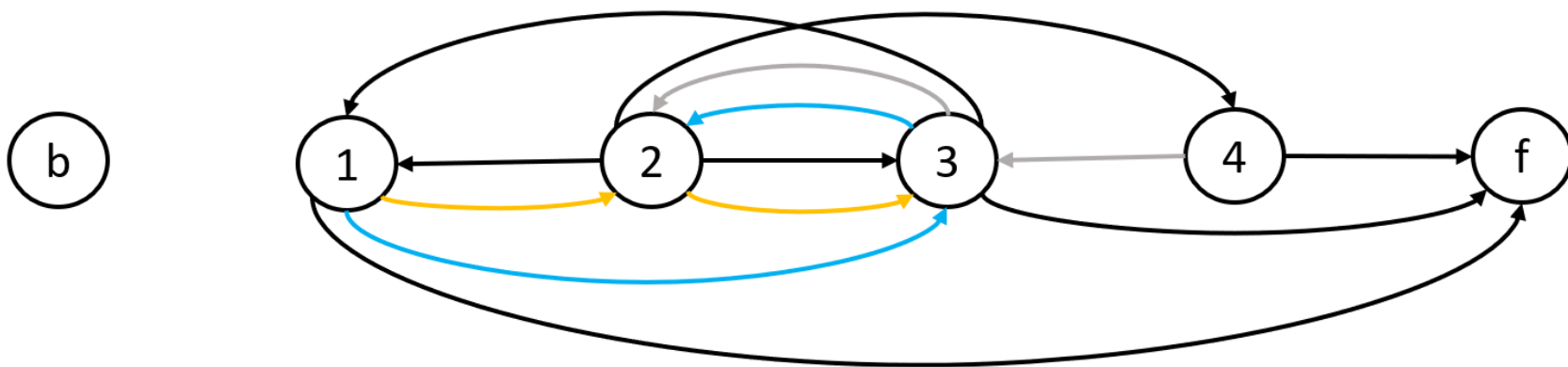
```
01) <start T1>
02) <T1, A, 60>
03) <commit T1>
04) <start T2>
05) <T2, A, 10>
06) <start T3>
07) <T3, B, 20>
08) <T2, C, 30>
09) <start T4>
10) <T3, D, 40>
<start ckpt>
11) <T4, F, 70>
12) <commit T3>
<end ckpt>
```

- 13) <T2, E, 50>
- 14) <commit T2>
- 15) <T4, B, 80>
- 16) <commit T4>

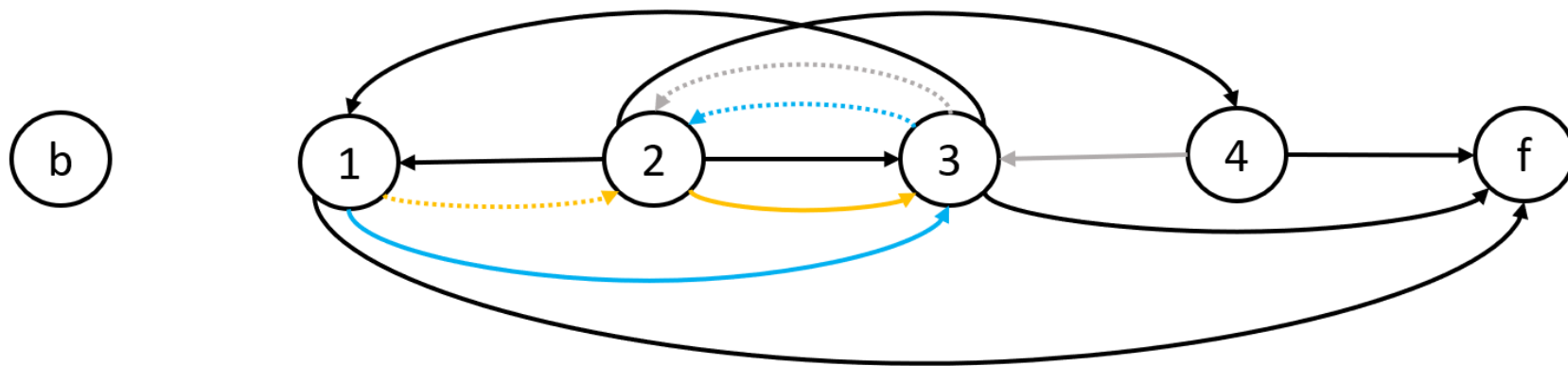
Đặt checkpoint sau bước <T2, E, 50> thì mẫu tin <end ckpt> có thể được ghi xuống bộ nhớ sau khi <commit T2>

- 01) <start T1>
- 02) <T1, A, 60>
- 03) <commit T1>
- 04) <start T2>
- 05) <T2, A, 10>
- 06) <start T3>
- 07) <T3, B, 20>
- 08) <T2, C, 30>
- 09) <start T4>
- 10) <T3, D, 40>
- 11) <T4, F, 70>
- 12) <commit T3>
- 13) <T2, E, 50>
- <start ckpt>
- 14) <commit T2>
- <end ckpt>
- 15) <T4, B, 80>
- 16) <commit T4>

Tb	T1	T2	T3	T4	Tf
W(A)					
W(B)					
W(C)					
W(D)					
			W(A)		
		W(C)			
	R(A)				
	W(B)				
	R(C)				
		W(A)			
			W(C)		
				R(A)	
				W(D)	
					R(A)
					R(B)
					R(C)
					R(D)



Loại bỏ 1 cung trong các cặp cung cùng màu sao cho không tạo chu trình.



Đồ thị cuối cùng có chu trình nên S KHÔNG view-serializable