COP 4722 – Survey of Database Systems

Assignment 2

Instructor: Nagarajan Prabakar

Student: Carlos Barrios

Florida International University

Spring 2019

==

Name: Carlos Barrios

Panther-ID: 6048821

Course: COP 4722

Assignment#: 2

Due: Wed, Feb 20, 2019

I hereby certify that this work is my own and none of it is the work of any other person.

Signature: CBarrios

===

The purpose of this assignment is to familiarize with **serializability**. Show all intermediate steps for each answer.

I. Consider the three transactions T_a, T_b, and T_c, and the schedules S_1, S_2, S_3 and S_4 given below. Which of the schedules is (conflict) serializable? The subscript for each database operation in a schedule denotes the transaction number for that operation. For each schedule, show all conflicts, draw the precedence graph, determine and write down if it is serializable or not, and the equivalent serial schedules if exist.

T_a: $r_a(x)$; $w_a(x)$;

T_b: $r_b(x)$;

T_c: $r_c(x)$; $w_c(x)$;

S_1: $r_a(x)$; $r_c(x)$; $w_a(x)$; $r_b(x)$; $w_c(x)$;

S_2: $r_a(x)$; $r_c(x)$; $w_c(x)$; $w_a(x)$; $r_b(x)$;

S_3: $r_c(x)$; $r_b(x)$; $w_c(x)$; $r_a(x)$; $w_a(x)$;

S_4: $r_c(x)$; $r_b(x)$; $r_a(x)$; $w_c(x)$; $w_a(x)$;

II. Consider the three transactions T_1, T_2, and T_3, and the schedules S_5 and S_6 given below. Show all conflicts and draw the serializability (precedence) graphs for S_5 and S_6, and state whether each schedule is serializable or not. If a schedule is serializable, write down the equivalent serial schedule(s).

T_1: $r_1(p)$; $r_1(r)$; $w_1(p)$;

T_2: $r_2(r)$; $r_2(q)$; $w_2(r)$; $w_2(q)$;

T_3: $r_3(p)$; $r_3(q)$; $w_3(q)$;

S_5: $r_1(p)$; $r_2(r)$; $r_1(r)$; $r_3(p)$; $r_3(q)$; $w_1(p)$; $w_3(q)$; $r_2(q)$; $w_2(r)$; $w_2(q)$;

S_6: $r_1(p)$; $r_2(r)$; $r_3(p)$; $r_1(r)$; $r_2(q)$; $r_3(q)$; $w_1(p)$; $w_2(r)$; $w_3(q)$; $w_2(q)$;

I. ANSWERS

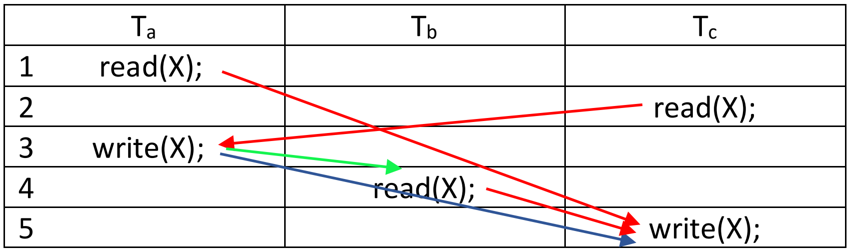
Legend

→ WAR (Write after Read)

→ RAW (Read after Write)

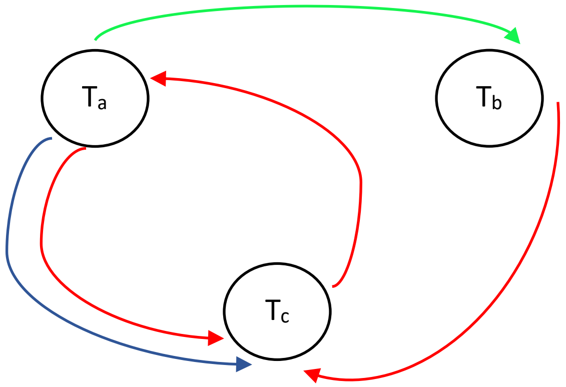
→ WAW (Write after Write)

S_1: $r_a(x)$; $r_c(x)$; $w_a(x)$; $r_b(x)$; $w_c(x)$;



T_a	T_b	T_c
1 read(X);		
2		read(X);
3 write(X);		
4	read(X);	
5		write(X);

Conflict Precedence Graph:



Since the precedence graph has the following two cycles:

Ta → Tc → Ta

Ta → Tb → Tc → Ta

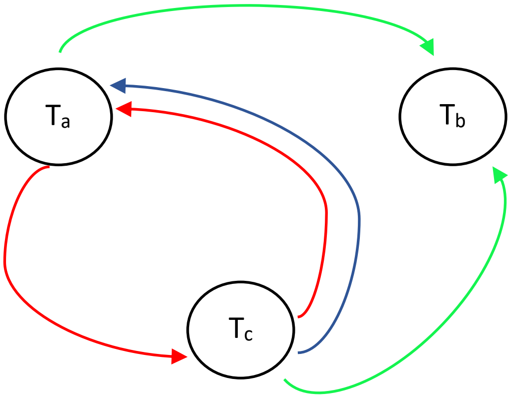
This schedule is <u>not conflict serializable</u>.

So, there is no_conflict_equivalent_serial_schedule for this schedule.

S_2: $r_a(x)$; $r_c(x)$; $w_c(x)$; $w_a(x)$; $r_b(x)$;

T_a	T_b	T_c
1 read(X);		
2		read(X);
3		write(X);
4 write(X);		
5	read(X);	

Conflict Precedence Graph:



Since the precedence graph has the following cycle:

Ta → Tc → Ta

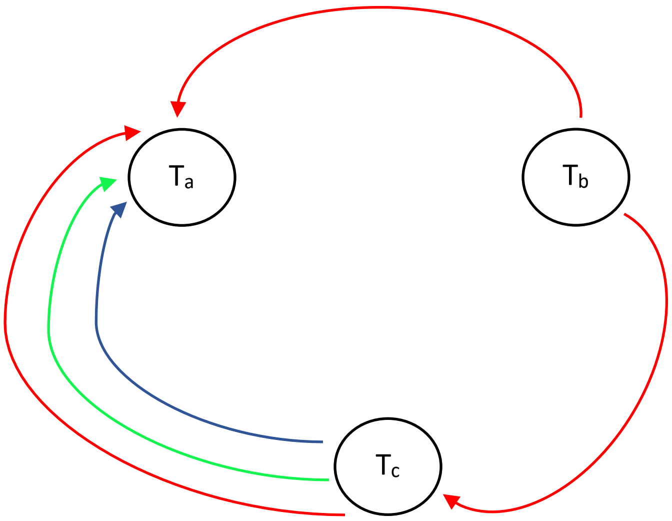
This schedule is <u>not conflict serializable</u>.

So, there is no_conflict_equivalent_serial_schedule for this schedule.

S_3: $r_c(x)$; $r_b(x)$; $w_c(x)$; $r_a(x)$; $w_a(x)$;

T_a	T_b	T_c
1		read(X);
2	read(X);	
3		write(X);
4 read(X);		
5 write(X);		

Conflict Precedence Graph:



Since the precedence graph has no cycle:

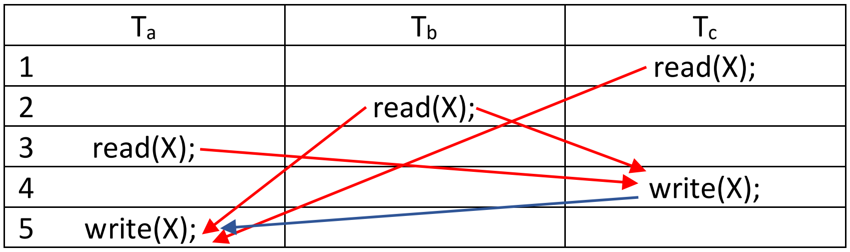
This schedule is <u>conflict serializable</u>.

So, the conflict_equivalent_serial_schedule for this schedule is:

Tb → Tc → Ta

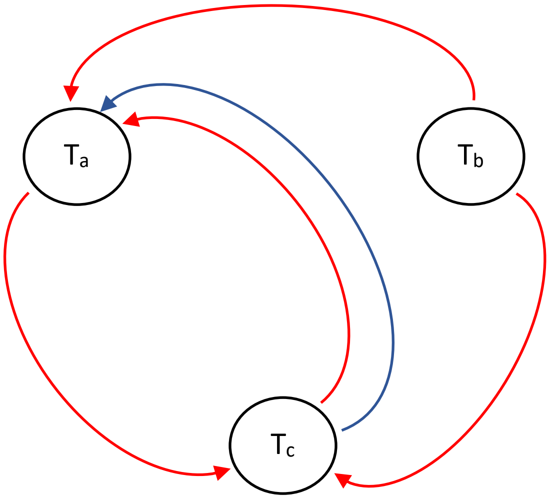
T_a	T_b	T_c
1	read(X);	
2		read(X);
3		write(X);
4 read(X);		
5 write(X);		

S_4: $r_c(x)$; $r_b(x)$; $r_a(x)$; $w_c(x)$; $w_a(x)$;



	T_a	T_b	T_c
1			read(X);
2		read(X);	
3	read(X);		
4			write(X);
5	write(X);		

Conflict Precedence Graph:



Since the precedence graph has the following cycle:

Ta → Tc → Ta

This schedule is <u>not conflict serializable</u>.

So, there is no_conflict_equivalent_serial_schedule for this schedule.

II. ANSWERS

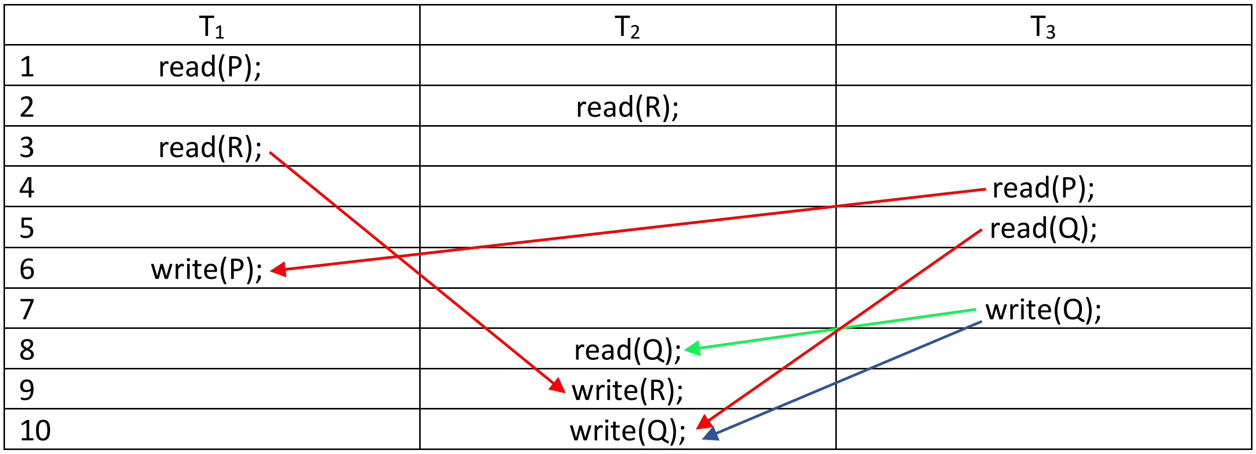
Legend

→ WAR (Write after Read)

→ RAW (Read after Write)

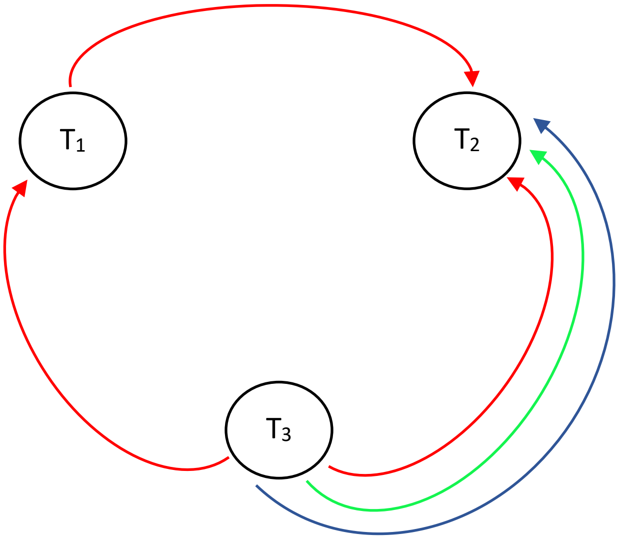
→ WAW (Write after Write)

S_5: $r_1(p)$; $r_2(r)$; $r_1(r)$; $r_3(p)$; $r_3(q)$; $w_1(p)$; $w_3(q)$; $r_2(q)$; $w_2(r)$; $w_2(q)$;



	T_1	T_2	T_3
1	read(P);		
2		read(R);	
3	read(R);		
4			read(P);
5			read(Q);
6	write(P);		
7			write(Q);
8		read(Q);	
9		write(R);	
10		write(Q);	

Conflict Precedence Graph:

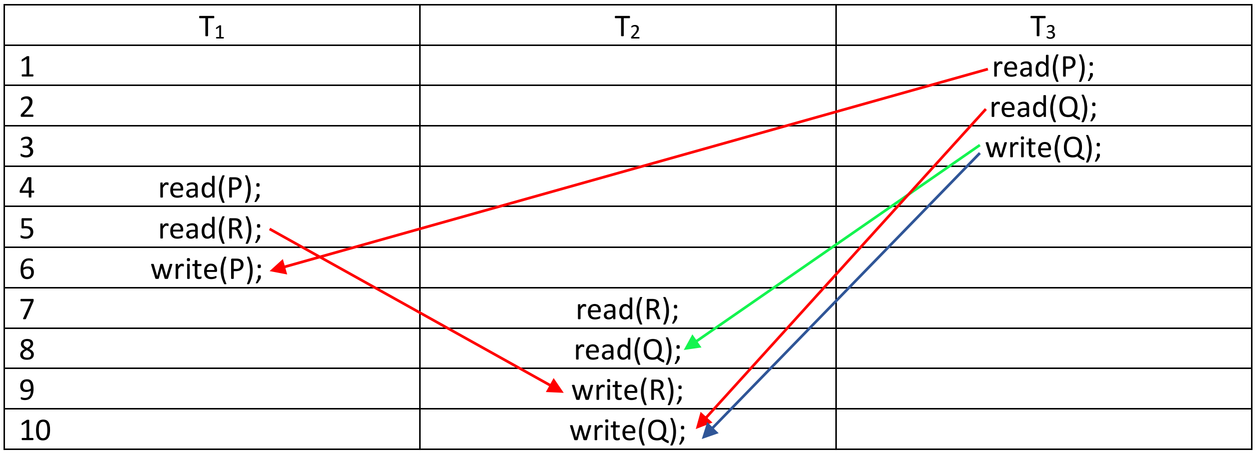


Since the precedence graph has no cycle:

This schedule is <u>conflict serializable</u>.

So, the conflict_equivalent_serial_schedule for this schedule is:

T3 → T1 → T2

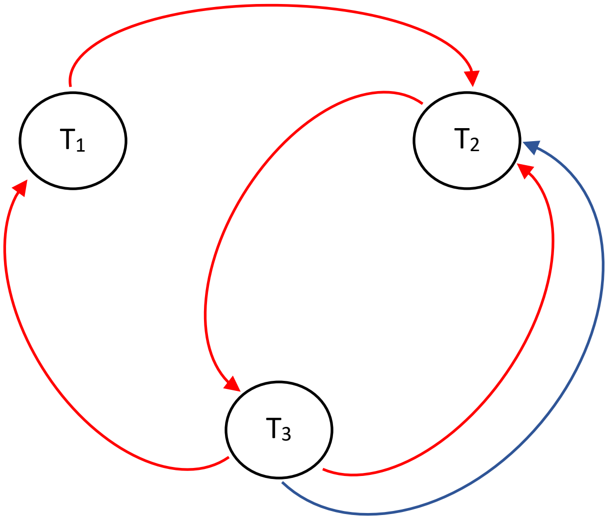


	T_1	T_2	T_3
1			read(P);
2			read(Q);
3			write(Q);
4	read(P);		
5	read(R);		
6	write(P);		
7		read(R);	
8		read(Q);	
9		write(R);	
10		write(Q);	

S_6: $r_1(p)$; $r_2(r)$; $r_3(p)$; $r_1(r)$; $r_2(q)$; $r_3(q)$; $w_1(p)$; $w_2(r)$; $w_3(q)$; $w_2(q)$;

T_1	T_2	T_3
1 read(P);		
2	read(R);	
3		read(P);
4 read(R);		
5	read(Q);	
6		read(Q);
7 write(P);		
8	write(R);	
9		write(Q);
10	write(Q);	

Conflict Precedence Graph:



Since the precedence graph has the following two cycles:

T2 → T3 → T2

T2 → T3 → T1→ T2

This schedule is <u>not conflict serializable</u>.

So, there is no_conflict_equivalent_serial_schedule for this schedule.