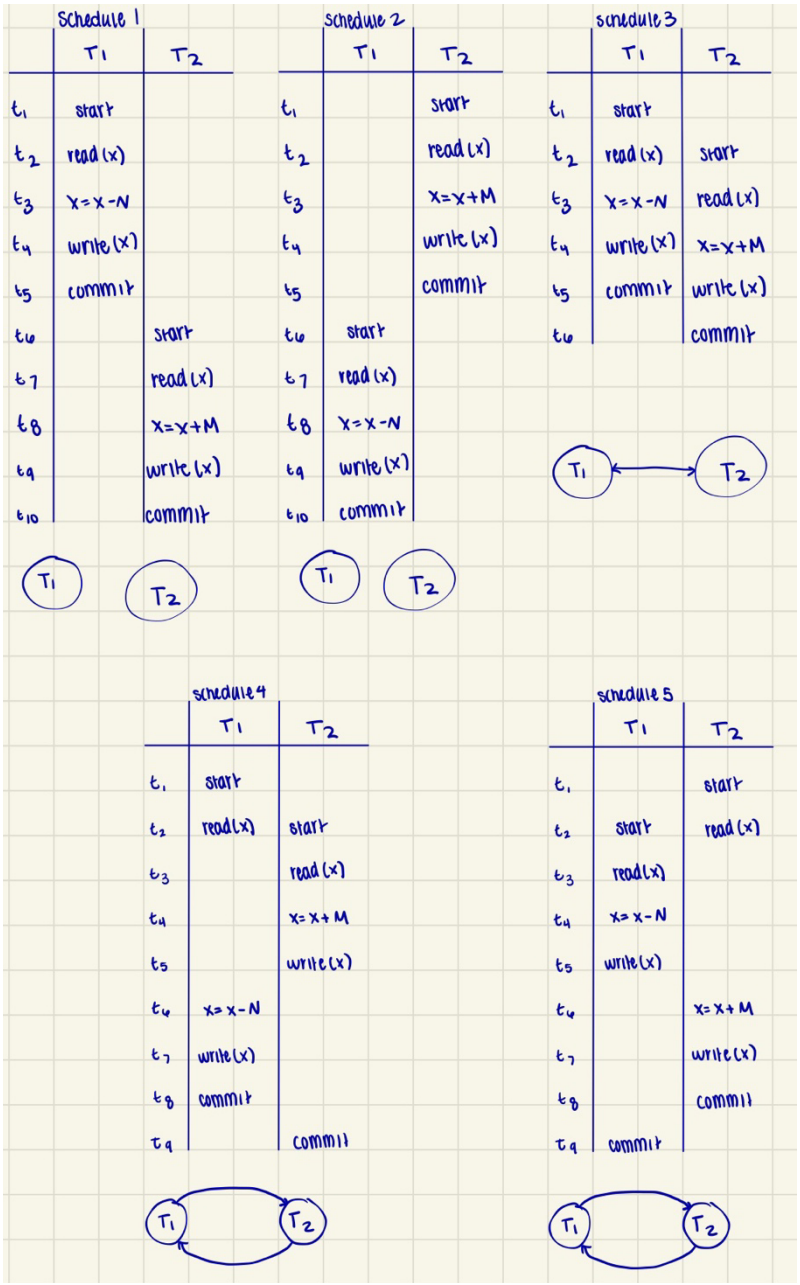
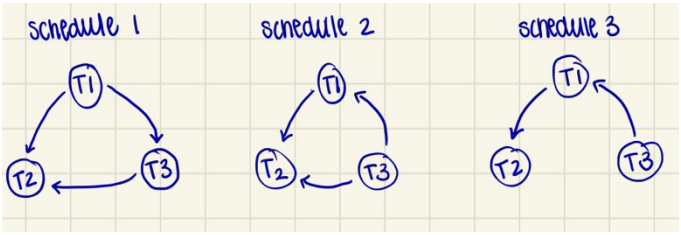
Kristen Lee
CS 5200 -Database Management Systems
Homework 10

1. **Given the following 2 different transactions. List all potential schedules for T1 and T2 and determine which schedules are conflict serializable and which are not.**

   Serial schedules 1 and 2 are always conflict serializable because there are no overlapping operations between the transactions. Schedule 3 is conflict serializable. Schedules 4 and 5 are not conflict serializable because there are overlapping operations, and we can see the precedence graphs are cyclic.



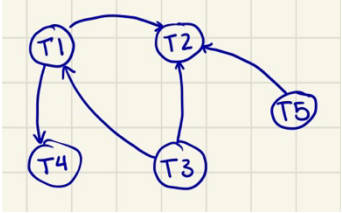
2. **Draw the precedence graph for the 3 schedules below**



3. **Which of the following schedules for problem 2 are conflict serializable? For each conflict serializable schedule determine the equivalent serial schedule.**

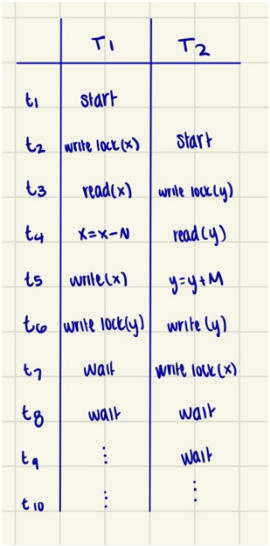
   All three of these schedules are conflict serializable.
   Equivalent serial schedules:
   Schedule 1: T1 → T3 → T2
   Schedule 2: T3 → T1 → T2
   Schedule 3: T3 → T1→T2

4. **Create a waits-for graph for the following lock table**



5. **Provide a schedule that exhibits the deadlock problem. Describe the issue.**

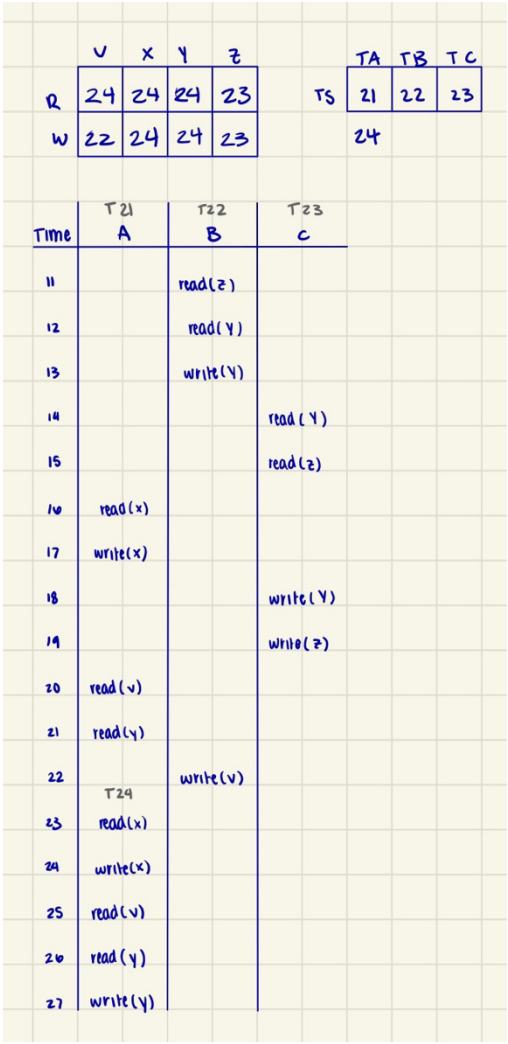
   T1 holds X which waits for Y, which is held by T2, and T2 holds Y which waits for X which is held by T1. So, T1 is waiting for T2, and T2 is waiting for T1, and this means the transactions cannot proceed and we have a dead lock.



	T_1	T_2
t_1	start	
t_2	write lock(x)	start
t_3	read(x)	write lock(y)
t_4	x=x-N	read(y)
t_5	write(x)	y=y+M
t_6	write lock(y)	write (y)
t_7	wait	write lock(x)
t_8	wait	wait
t_9	⋮	wait
t_{10}	⋮	⋮

6. **Apply the basic timestamping algorithm to the following schedule using the values of the read and write timestamps for the accessed data objects V, X, Y, and Z. Determine the read and write timestamps for the objects after the schedule is run. Also, provide the schedule that was run to complete these transactions. Assume that the transaction id issued to the latest transaction is 20 (transaction 20 is the latest transaction before transactions A, B, AND C are started). Determine if the schedule can be performed as is or if a transaction(s) will need to be restarted. If so, what transactions will need to be restarted given the basic timestamping ordering algorithm? Also, assume the same original timing for the steps for each transaction that is restarted.**

Transaction A must be restarted because at t21 when it tries to read Y, it is not allowed because it was already read and written by a younger transaction, which is Transaction C. So, we restart Transaction A as a younger transaction after B and C are completed.



	V	X	Y	Z
R	24	24	24	23
W	22	24	24	23

	TA	TB	TC
TS	21	22	23
	24		

Time	T21 A	T22 B	T23 C
11		read(z)	
12		read(Y)	
13		write(Y)	
14			read(Y)
15			read(z)
16	read(x)		
17	write(x)		
18			write(Y)
19			write(z)
20	read(v)		
21	read(y)		
22		write(v)	
	T24		
23	read(x)		
24	write(x)		
25	read(v)		
26	read(y)		
27	write(y)		

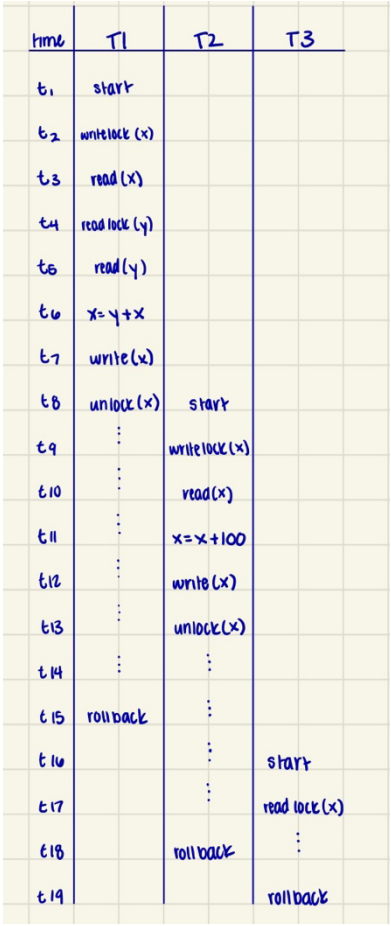
7. **What are the two-phase locking protocols, explain its types.**

According to the rules of the 2PL, a transaction can be divided into two phases, the growing phase and the shrinking phase. In the growing phase, all the locks are acquired but cannot release

the locks. In the shrinking phase, the locks are released, but new ones can't be acquired. A transaction follows 2PL if all locking operations precede first unlock operation in the transaction.

8. **Define cascading rollback. Provide an example of a schedule with a cascade rollback.**

   Cascading rollback is when a single transaction failure leads to a series of transactions being restarted or rolled back. Cascading rollbacks are undesirable because they can lead to the undoing of a significant amount of work.



9. **Given the schedule below classified the schedule as recoverable or not recoverable. If classified as recoverable, then specify if it is strict-recoverable, cascadeless, cascade recoverable. Please support your answer with a valid description.**

   This schedule is recoverable. A schedule is recoverable if a transaction commits only after the transactions it depends on commit. This schedule is cascade recoverable because it could allow roll backs. If T1aborts, then T3 would have to roll back because it has read the uncommitted value of X from T1. The schedule ensures recoverability by committing T3 only after T1 commits. It is not strict-recoverable because strict schedules require that no transaction reads or writes a value that is uncommitted, and this schedule does so at t3 where T3 reads X before T1 commits. This is not cascadeless because cascadeless schedules prevent rollback by making sure uncommitted values are not read by a transaction. As stated before, T3 reads the uncommitted value of X before T1 commits it.