CS 348 - Homework 5

Concurrency Control
(100 Points)

Fall 2019

Due on: 11/22/19 11:59pm

Note: There will be a 10% penalty for each late calendar day. After five calendar days, the homework will not be accepted.

1. (25 points)
   A. (10 points)Prove that the basic two-phase locking protocol guarantees conflict serializability of schedules. (Hint: Show that, if a serializability graph for a schedule has a cycle, then at least one of the transactions participating in the schedule does not obey the two-phase locking protocol.)

If serializability graph of schedule s has a cycle
Then there exist [o1(X); ...; o2(X);] ...; [o2(Y); ...; o3(Y);] ... ; [on(Z); ...; o1(Z);]
within the schedule where each pair of operations [o,o] is either [w,w], or [w, r], or [r,w])
Then transaction T1 has the following sequence: T1: ...; o1(X); ... ; o1(Z); ...
Since T2 uses X after T1 and Tn uses Z before T1, T1 has to unlock X before locking Z.
Now T1 does not obey two-phase locking protocol because all locking should be done before unlocking

B. (10 points) Prove that cautious waiting avoids deadlock.

A deadlock happens when Ti is waiting on resources occupied by Tj while Tj is waiting on resources occupied by Ti are happening at the same time.
In cautious waiting, a transaction Ti can wait on a transaction Tj only if Tj is not waiting on other transaction.
Because if Ti is already waiting on Tj, if Tj wants to wait on Ti, it is not allowed and will abort according to the algorithm.
So deadlock will never happen.

C. (5 points) Prove that strict two-phase locking guarantees strict schedules.

A strict schedules means transactions can neither read nor write an item X until the last transaction that wrote X has committed or aborted.
Strict two-phase locking guarantees that transaction does not release exclusive locks until after it commits or aborts
For example, transaction Ti occupies item X and it will not release its lock until after Ti commits or aborts. Before this happens, since item X is locked by transaction Ti, no other transaction can read nor write it. This satisfy the definition of strict schedule.

2. (20 points) Consider the following actions taken by transaction T_1 on database objects X and Y: **R(X), W(X), R(Y), W(Y)**

   A. (10 points) Give an example of another transaction T_2 that, if run concurrently to transaction T_1 without some form of concurrency control, could interfere with T_1.

   T2: R(X),W(X)

   Without concurrency control, T2 may read item X before T1 writes X and hence the update is lost

   B. (10 points) Strict 2PL is a locking protocol where

      1. A transaction requests a shared/exclusive lock on the object before it reads/modifies the object
      2. All locks held by a transaction are released when the transaction is complete.

   Explain how the use of Strict 2PL would prevent interference between the two transactions in part 1.

   With the use of strict 2PL, before T1 reads item X, T1 locks X until T1 is complete
   So T2 can only read X when T1 is over and item X is updated completely.

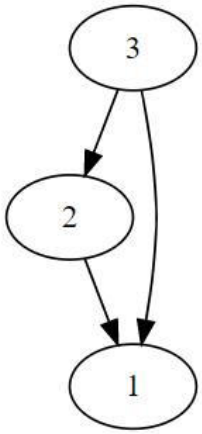
3. (25 points) Consider the following schedules with three transactions. The actions are listed in the order they are scheduled, and are prefixed with the transaction name.

S1: T1:R(Y), T3:R(X), T3:W(X), T2:R(Y), T2:W(X), T3:W(Z), T1:R(X), T2:R(Z)
S2: T2:R(Y), T2:W(Y), T1:W(X), T1:R(Y), T3:R(X), T1:W(Y), T2:W(X)

A. Is the schedule S1 serial? [2.5 points]

Yes, you can go from T3 to T2 and then to T1

B. Draw the dependency graph for S1. [5 points]



C. Is S1 conflict serializable? If so, what is the conflict equivalent serial schedule? [5 points]

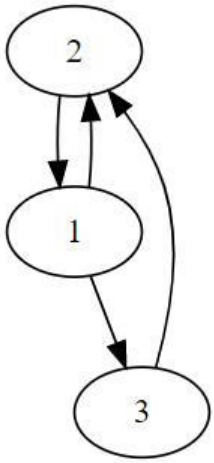
Yes, It is conflict serializable because dependency graph doesn't have any circle

A conflict equivalence schedule can be
T3:R(X), T3:W(X), T3:W(Z), T2:R(Y), T2:W(X), T2:R(Z), T1:R(Y), T1:R(X)

D. Is the schedule S2 serial? [2.5 points]

Yes, you can go from T2 to T1 and then to T3

E. Draw the dependency graph for S2. [5 points]



F. Is S2 conflict serializable? If so, what is the conflict equivalent serial schedule? [5 points]

No, because there is a cycle between T2 and T1.

4. Consider the following lock requests in Table 1. And note that

    1. S(.) and X(.) stand for 'shared lock' and 'exclusive lock', respectively.
    2. T1, T2, T3 and T4 represent three transactions.
    3. LM stands for Lock Manager.
    4. Transactions never release a granted lock.

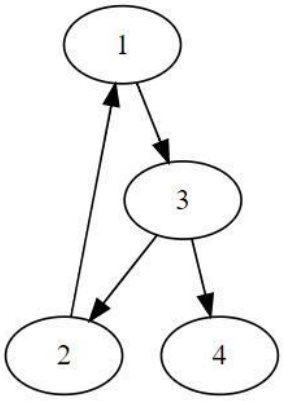
time	t1	t2	t3	t4	t5	t6	t7	t8
T1	S(A)		S(B)					
T2		X(B)						X(D)
T3				S(C)	X(D)		X(A)	
T4						X(C)		
LM	g							

Table 1: Transactions and lock manager.

1. For the lock requests in Table 1, determine which lock will be granted or blocked by the lock manager, if it has no deadlock prevention strategy. Please write 'g' in the LM row to indicate that the lock is granted and 'b' to indicate that the lock is blocked and 'a' for abort.

time	t1	t2	t3	t4	t5	t6	t7	t8
T1	S(A)		S(B)					
T2		X(B)						X(D)
T3				S(C)	X(D)		X(A)	
T4						X(C)		
LM	g	g	b	g	g	b	b	b

2. Determine where there exists a deadlock in the lock requests in Table 1 and briefly explain why.



If we draw the dependency graph, you can see there is a cycle among 1,2,3
So there exists a deadlock.

3. To prevent deadlock, we use a lock manager (LM) that adopts the Wait-Die policy. We assume that in terms of priority: T1 > T2 > T3 > T4. Here, T1 > T2 because T1 is older than T2. (i.e., older transactions have higher priority.) Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a') (as its transaction aborts). Follow the same format as the previous question.

time	t1	t2	t3	t4	t5	t6	t7	t8
T1	S(A)		S(B)					
T2		X(B)						X(D)
T3				S(C)	X(D)		X(A)	
T4						X(C)		
LM	g	g	b	g	g	a	a	g

4. Now, we use a lock manager (LM) that adopts the Wound-Wait policy. We assume that in terms of priority: T1 > T2 > T3 > T4. Here, T1 > T2 because T1 is older than T2. (i.e. older transactions have higher priority.) Determine which lock request will be granted ('g'), blocked ('b') or aborted ('a').

time	t1	t2	t3	t4	t5	t6	t7	t8
T1	S(A)		S(B)					
T2		X(B)						X(D)
T3				S(C)	X(D)		X(A)	
T4						X(C)		
LM	g	g	g	g	g	b	b	a