AALBORG UNIVERSITET

1 Serializability

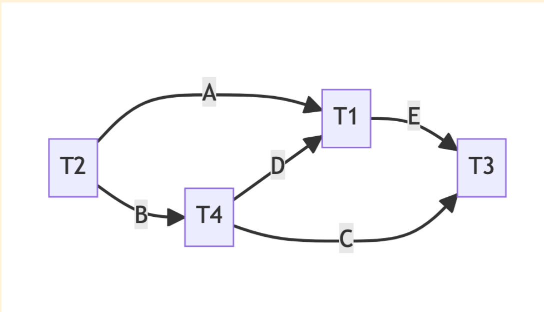
Consider the transactions T_1, T_2, T_3, T_4 below where R(·) and W(·) stand for "Read" and "Write", respectively.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
T_1			W(E)		R(D)				W(A)	
T_2		W(B)		R(A)						
T_3								R(E)		W(C)
T_4	W(D)					R(B)	R(C)			

Table 1: A schedule with 4 transactions

Draw the correct conflict dependency graph for the schedule.

Solution.



Answer all the following.

1. Is this schedule serial?
2. Is the schedule conflict serializable?
3. What is the minimum number of transactions that need to be removed to produce a conflict serializable schedule? (Zero if the original was already conflict serializable).
4. Is the schedule above conflict equivalent to the schedule T_2, T_4, T_1, T_3. Explain your answer.

Solution.

1. *The schedule is interleaved, not serial.*
2. *The schedule is conflict serializable because there are no directed cycles.*
3. *Zero*
4. *It is conflict equivalent to T_2, T_4, T_1, T_3, and only that, because following the dependency graph that is the schedule that do not invert order of conflicts.*

2 Locking

2.1 Exercise

Rewrite the schedule in Table 1 adding where needed Shared or Exclusive lock request and the corresponding unlock actions. Use 2-Phase locking, but not strict strong 2PL, so release locks as soon as possible. Remember to show the actions of the lock manager. How will the schedule evolve? Will it generate a deadlock? Will it generate dirty reads? Explain your answer.

Solution.

time	t_1	t_2	t_3	t_4		t_5	t_6	t_7			t_8	t_9			t_{10}	
T_1			*XL(E); W(E)*			*SL(D)*				*R(D)*		*XL(A); W(A)*	*UL(E); UL(D); UL(A)*			
T_2		*XL(B); W(B)*		*SL(A); R(A)*	*UL(B); UL(A)*											
T_3											*SL(E);*			*R(E)*	*XL(C); W(C)*	*UL(C); UL(E)*
T_4	*XL(D); W(D)*						*SL(B); R(B)*	*SL(C); R(C)*	*UL(D); UL(B); UL(C)*							
LM	$G(D,X,T_4)$	$G(B,X,T_2)$	$G(E,X,T_1)$	$G(A,S,T_2)$	*Rl(B); Rl(A)*	$D(D,T_1)$	$G(B,S,T_4)$	$G(C,S,T_4)$	*Rl(B); Rl(C)*	$G(D,S,T_1)$	*D(E)*	$G(A,X,T_1)$	*Rl(E); Rl(D); Rl(A)*	$G(E,S,T_3)$	$G(C,X,T_3)$	*Rl(E); Rl(C)*
Locks	(D,X,T_4)	(D,X,T_4) (B,X,T_2)	(D,X,T_4) (B,X,T_2) (E,X,T_1)	(D,X,T_4) (B,X,T_2) (E,X,T_1) (A,S,T_2)	(D,X,T_4) (E,X,T_1)	(D,X,T_4) (E,X,T_1)	(D,X,T_4) (E,X,T_1) (B,S,T_4)	(D,X,T_4) (E,X,T_1) (B,S,T_4) (C,S,T_4)	(E,X,T_1)	(E,X,T_1) (D,S,T_1)	(E,X,T_1) (D,S,T_1)	(E,X,T_1) (D,S,T_1) (A,X,T_1)		(E,S,T_3)	(E,S,T_3) (C,X,T_3)	

Here: G is "grant", D is "deny", UL is "unlock", and Rl is "release".

Solution.

1. *The execution will be the same as a strict strong 2PL anyway because there are no opportunities to release any lock earlier than the end of the transaction, all transaction require a lock as second-to-last operation.*
2. *There are no deadlocks.*
3. *The are no dirty reads because no transaction aborts.*

2.2 Exercise

Consider the the transactions T_1, T_2, T_3, T_4 below where R(·) and W(·) stand for "Read" and "Write", respectively.

time	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
T_1	W(E)				R(E)		R(D)				W(A)	
T_2				W(B)		R(A)						
T_3			W(B)							R(E)		W(C)
T_4		W(D)						R(B)	R(C)			

Table 2: A schedule with 4 transactions

Rewrite the schedule in Table 2 adding where needed Shared or Exclusive lock request and the corresponding unlock actions. Use strict strong 2-Phase locking. Remember to show the actions of the lock manager. How will the schedule evolve? Will it generate a deadlock? Will it generate dirty reads? Explain your answer.

Solution.

time	t_1	t_2	t_3	t_4	t_5	t_7	t_8	t_{10}	
T_1	*XL(E); W(E)*				*R(E)*	*SL(D)*			
T_2				*XL(B);*					
T_3			*XL(B); W(B)*					*SL(E)*	
T_4		*XL(D); W(D)*					*SL(B);*		
LM	*G(E,X,T_1)*	*G(D,X,T_4)*	*G(B,X,T_3)*	*D(B,X,T_2)*		*D(D,S,T_1)*	*D(B,S,T_4)*	*D(E,S,T_4)*	*DLOCK!*
Locks	*(E,X,T_1)*	*(E,X,T_1)* *(D,X,T_4)*	*(E,X,T_1)* *(D,X,T_4)* *(B,X,T_3)*	*(E,X,T_1)* *(D,X,T_4)* *(B,X,T_3)*	*(E,X,T_1)* *(D,X,T_4)* *(B,X,T_3)*	*(E,X,T_1)* *(D,X,T_4)* *(B,X,T_3)*	*(E,X,T_1)* *(D,X,T_4)* *(B,X,T_3)*	*(E,X,T_1)* *(D,X,T_4)* *(B,X,T_3)*	

Here: G is "grant", D is "deny", UL is "unlock", Rl is "release", and DLOCK is "deadlock detected".

Solution.

1. *The execution will end in the deadlock because, first T2 will wait on T3, then T1 will wait on T4, then T4 will wait on T3 as well, and finally T3 will wait on T1.*
2. *Thus there will be a deadlock. One transaction at least will have to abort.*
3. *Since we are executing strict strong 2PL there will never be dirty reads.*