Kayvaun Khoshkhou

ID: 920357344

CSC675 Homework 4

Question1. (11 points, 1 point each)

Answer True or False to these questions:

a) Schedules using 2PL could have cascading aborts.

True

b) Strict 2PL accepts only serial schedules.

True

c) Every Strict schedule is conflict serializable.

True

d) Every serializable schedule is conflict serializable.

True

e) Every serializable schedule can be produced using 2PL.

False

f) 2PL allows only strict schedules.

False

g) Schedules under Strict 2PL may not lead to deadlocks.

True

h) Recovery manager ensures transaction atomicity and durability.

True

i) 2PL allows only shared locks to be released before the end of the transaction.

False

j) Several transactions can hold a shared lock on an object at the same time.

True

k) In both the policies of deadlock prevention, only the transaction with a lower priority (i.e. the younger transaction) might be aborted and restarted.

False

Question 2. (28 points, 14 points each schedule)

Consider schedules S1 and S2. Complete the following table and answer these questions for each schedule.

a. (8 points) Is the schedule conflict-serializable? Why?

S1 is conflict serializable. It contains the cycle in the precedence graph. S2 is not conflict serializable. It does not contain the cycle in the precedence graph.

b. (2 points) What is the equivalent serial schedule?

R1(X), W1(X), R1(Y), W1(Y), R2(X), R2(Y), W2(Y), W2(X)

c. (2 points) Could the schedule be produced using 2PL protocol?

Only S1 can be produced since it is conflict serializable. Therefore S2 cannot be produced.

d. (2 points) Could the schedule be produced using strict 2PL protocol?

No, it is not possible for T2 to read, T1 needs to release its exclusive lock and W2(X) needs a lock later on.

S1: R1(A), R2(A), W2(A), R3(B), R2(B), W3(B), W2(B), W1(A)

S2: R1(A), R2(A), R2(B), R3(A), R4(B), W1(A), W2(B)

Schedule	Conflict-serializable	Equivalent serial schedule	Produced by 2PL	Produced by strict 2PL
S1	No	No	No	No
S2	Yes	Yes	Yes	Yes

Question 3. (24 points, 8 points each)

Consider the following classes of schedules: serializable, conflict-serializable, avoids cascading-aborts, and strict. For each of the following schedules, state which of the preceding classes it belongs to. The actions are listed in the order they are scheduled and prefixed with the transaction name. If a commit or abort is not shown, the schedule is incomplete; assume that abort or commit must follow all the listed actions.

1. T1:R(X), T2:R(X), T2:W(X), T2:Commit, T1:W(X), T1:Commit

a) Conflict-serializable b) Avoid cascading abort

[✓ c) Serializable] d) Strict

2. T1:W(X), T2:W(X), T2:Commit, T1:Commit

a) Conflict-serializable [✓ b) Avoid cascading abort]

c) Serializable d) Strict

3. T2: R(X), T3:W(X), T3:Commit, T1:W(Y), T1:Commit, T2:R(Y), T2:W(Z), T2:Commit

a) Conflict-serializable b) Avoid cascading abort

[✓ c) Serializable] d) Strict

Question 4. (12 points)

The following table presents the lock requests by transactions T1, T2, and T3. S(.) and X(.) stand for `shared lock' and `exclusive lock', respectively. Assume we follow deadlock detection policy.

time	t1	t3	t4	t6
T1	S(X)			X(Y)
T2		X(X)		
T3			X(Y)	
Lock Manager				

a) (4 points) For the lock requests in the Table, determine which lock will be granted or blocked by the lock manager. Please write `grant' in the last row to indicate the lock is granted and `block' to indicate the lock is blocked.

Time	T1	T3	T4	T6
T1	S(X)			X(Y)
T2		X(X)		
T3			X(Y)	
Lock Manager	Grant	Grant	Grant	Block

b) (6 points) Give the wait-for graph for the lock requests in the above Table.

$$\begin{tikzcd} T_1 \arrow[dr] & & T_2 \arrow[ll] \\ & T_3 & \end{tikzcd}$$

c) (2 points) Determine whether there exists a deadlock in the lock requests in the above Table, and why.

No, the deadlock does not exist in the table because it does not contain any sort of cycle. The order executes in a one-way direction.

Question 5. (10 points) The following table presents the lock requests by transactions T1, T2, and T3. S(.) and X(.) stand for `shared lock' and `exclusive lock', respectively. Assume we follow deadlock prevention using Wound-Wait policy with these priorities: T1>T2>T3

Time	T1	T3	T4	T5	T6
T1	S(X)				X(Y)
T2		X(Y)	X(X)		
T3				X(Y)	
Lock Manager	GRANT	GRANT	WAIT	WAIT	ABORT T2

Question 6. (15 points) Consider the following sequence of actions, listed in the order in which they are submitted to the DBMS: Sequence S: T1:R(X), T2:W(X), T2:W(Y), T3:W(Y), T1:W(Y), T1: Commit, T2:Commit, T3:Commit

For each sequence and for each of the following concurrency control mechanisms, fill the table based on the concurrency control mechanisms. Assume that the timestamp of transaction Ti is i. For lock-based concurrency control mechanism, add lock and unlock requests to the above sequence of actions as per the locking protocol. If a transaction is blocked, assume that all of its actions are queued until it is resumed; the DBMS continues with the next action (according to the listed sequence) of an unblocked transaction.

A) (7 points) Strict 2PL with timestamps used for deadlock prevention using wait-die policy, priority: T1>T2>T

Time	T1	T2	T3	T4	T5	T6	T7
T1	R(X)				W(Y)	Commit	
T2		W(X)	W(Y)				Commit
T3				W(Y)			
Lock Manager	Grant	Grant	Grant	Wait T2	Abort T1	Grant	Grant

B) (8 points) Strict 2PL with deadlock detection. Show the waits-for graph if a deadlock cycle develops

Time	T1	T2	T3	T4	T5	T6	T7
T1	R(X)				W(Y)	Commit	
T2		W(X)	W(Y)				Commit
T3				W(Y)			
Lock Manager	Grant	Grant	Grant	Wait T2	Wait T3	Grant	Grant