CSE 444 Homework 4

Yuanfeng Li

TOTAL POINTS

39 / 40

QUESTION 1

20 pts

1.1 a 5 / 5

✓ **- 0 pts Correct**

- 1 pts Write-read conflict incorrect

- 1 pts Read-write conflict incorrect

- 1 pts Write-write conflict incorrect

- 0.5 pts Actions within a transaction is incorrect (minor error)

- 1 pts Actions within a transaction is incorrect

The unrepeatable read should be the second R2(X).

1.2 b 5 / 5

✓ **- 0 pts Correct**

- 5 pts Incorrect

1.3 c 5 / 5

✓ **- 0 pts Correct**

- 1 pts Multiple commit statement

- 2 pts Need to unlock locks after transaction

- 1 pts Commit should be after unlock

- 2 pts 2 Transactions acquired lock at the same time

- 1 pts No Commits shown

- 2 pts Not shown if the lock is granted or denied

- 5 pts Insufficient solution

- 2 pts No Sample schedule

- 2 pts Need to show the whole transaction working

- 2 pts Cannot have an exclusive lock and shared lock at same time

1.4 d 4 / 5

- 0 pts Correct

- 1 pts Mention more precisely how it helps the overall schedule

✓ **- 1 pts Need more information how strict 2PL is better**

QUESTION 2

20 pts

2.1 a 10 / 10

✓ **- 0 pts Correct**

- 1 pts Ignoring actions that should instead be delayed

- 1 pts Timestamps do not match schedule

- 1 pts Mistaken application of Thomas write rule

- 1 pts Forgot to delay a read or write

- 0.5 pts Could have executed delayed actions earlier

- 0.5 pts Forgot to update timestamp caused by delayed action

- 2 pts Did not abort infeasible transaction ordering

- 1 pts Minor deviation from provided schedule

- 1 pts Set commit = True while modifying transaction was active

- 1 pts Delayed action unnecessarily

✓ **- 0 pts Note that 'commit' action does not execute after abort**

- 10 pts No submission

- 1 pts Forgot to execute delayed actions

2.2 b 10 / 10

✓ **- 0 pts Correct**

- 1 pts W2(X) shouldn't be allowed

- 1 pts T2 should be aborted

- 1 pts Missing/Wrong RT for X1

- 1 pts Missing/Wrong RT for X3

- **10 pts** unanswered
- **1 pts** Wrong format

1. Your schedule should contain a write-read conflict that causes one of the transactions to perform a dirty read

T1	T2
R(X)	
	R(X)
R(Y)	
	R(Y)
	W(Y)
W(X)	
	R(X) dirty read
COMMIT	
	R(Y)
	W(X)
	R(Z)
	W(Z)
	COMMIT

2. Your schedule should contain a read-write conflict that causes one of the transactions to encounter an unrepeatable read.

T1	T2
R(X)	
R(Y)	
	R(X) unrepeatable read
W(X)	
	R(Y)
COMMIT	
	W(Y)
	R(X)
	R(Y)
	W(X)
	R(Z)
	W(Z)
	COMMIT

3. Your schedule should contain a write-write conflict that causes a lost update.

T1	T2
R(X)	
	R(X)
R(Y)	
	R(Y)
	W(Y)
	R(X)
	R(Y)
W(X)	
	W(X) lost update
	R(Z)
	W(Z)
COMMIT	
	COMMIT

1.1 a **5 / 5**

✓ **- 0 pts** **Correct**

- 1 pts Write-read conflict incorrect

- 1 pts Read-write conflict incorrect

- 1 pts Write-write conflict incorrect

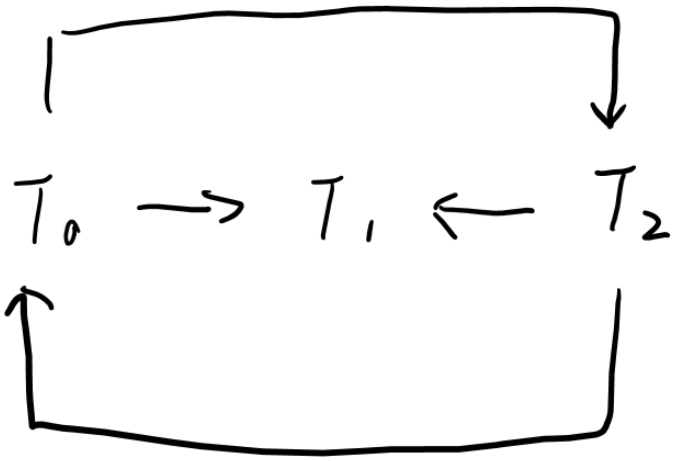
- 0.5 pts Actions within a transaction is incorrect (minor error)

- 1 pts Actions within a transaction is incorrect

The unrepeatable read should be the second R2(X).

Q1.2:

- It is not a conflict-serializable schedule.
- We can tell the reason when we draw a Precedence Graph for the schedule:



There is a cycle in it, so the schedule is not conflict serializable.

1.2 b **5 / 5**

✓ **- 0 pts** **Correct**

- 5 pts Incorrect

T0	T1	T2
L0(A), L0(B), R0(A)		
W0(A), U0(A)		
		L2(A), L2(B) Denied, R2(A)
		W2(A)
	L1(A), R1(A) Denied (#move to later)	
R0(B)		
W0(B), U0(B)		
		L2(B) Granted, R2(B), U2(A)
	L1(A) Granted, R1(A)	
		W2(B), U2(B)
	L1(B), R1(B), U1(A), U1(B)	
	Commit 1	
Commit 0		
		Commit 2

1.3 C **5 / 5**

✓ **- 0 pts** **Correct**

- 1 pts Multiple commit statement

- 2 pts Need to unlock locks after transaction

- 1 pts Commit should be after unlock

- 2 pts 2 Transactions acquired lock at the same time

- 1 pts No Commits shown

- 2 pts Not shown if the lock is granted or denied

- 5 pts Insufficient solution

- 2 pts No Sample schedule

- 2 pts Need to show the whole transaction working

- 2 pts Cannot have an exclusive lock and shared lock at same time

If 2PL ensures conflict-serializability, why do we need strict 2PL? Explain briefly

Answer:

1)

The main reason of why we need strict 2PL than 2PL is because strict 2PL is doing a better job on recoverability while guarantees the schedule conflict serializability at the same time.

The 2PL might run into some problem when there are rollbacks in some transactions while other transaction has already committed the operations based on the rolled-back data and values.

In order to resolve the problem, Strict 2PL make all unlocks done together with the COMMIT or ROLLBACK, which guarantees the recoverability for the database.

2)

Strict 2PL will not cause deadlock in it, which is also an advantage over 2PL. Which let Strict 2PL saves time from a lot of trouble such as figure out there is a deadlock, choose which one to kill, roll back all the changes it has made, and etc.

1.4 d **4 / 5**

- **0 pts** Correct
- **1 pts** Mention more precisely how it helps the overall schedule

✓ **- 1 pts** **Need more information how strict 2PL is better**

T1	T2	T3	T4	X	Y
1	2	3	4	Rt=0 Wt=0 C=True	Rt=0 WT=0 C=True
	R2(X)			rt=2 wt=0 c=true	
R1(X)				rt=2 wt=0 c=true	
	W2(X)			rt=2 wt=2 c=false	
			w4(x)	rt=2 wt=4 c=false	
w1(X) abort				rt=2 wt=4 c=false	
c1				rt=2 wt=4 c=false	
		w3(X) delay		rt=2 wt=4 c=false	
			A4	rt=2 wt=2 c=false	
		grant w3(x)		rt=2 wt=3 c=false	
	R2(Y)				rt=2 wt=0 c=true
	w2(y)				rt=2 wt=2 c=false
		r3(y) delay			rt=2 wt=2 c=false
	c2				rt=2 wt=2 c=true
		grant r3(y)			rt=3 wt=2 c=true
		w3(y)			rt=3 wt=3 c=false
		C3		rt=2 wt=2 c=true	rt=3 wt=3 c=true

2.1 a **10 / 10**

✓ **- 0 pts** **Correct**

- 1 pts Ignoring actions that should instead be delayed

- 1 pts Timestamps do not match schedule

- 1 pts Mistaken application of Thomas write rule

- 1 pts Forgot to delay a read or write

- 0.5 pts Could have executed delayed actions earlier

- 0.5 pts Forgot to update timestamp caused by delayed action

- 2 pts Did not abort infeasible transaction ordering

- 1 pts Minor deviation from provided schedule

- 1 pts Set commit = True while modifying transaction was active

- 1 pts Delayed action unnecessarily

✓ **- 0 pts** **Note that 'commit' action does not execute after abort**

- 10 pts No submission

- 1 pts Forgot to execute delayed actions

T1	T2	T3	T4	X0	X3	X4
1	2	3	4			
R1(X)				RT = 1		
		R3(X)		RT = 3		
		W3(X)			Create	
	R2(X)			Rt = 3		
			R4(X)		RT = 4	
	W2(X): abort					
			W4(X)			Create

2.2 b **10 / 10**

✓ **- 0 pts Correct**

- 1 pts W2(X) shouldn't be allowed

- 1 pts T2 should be aborted

- 1 pts Missing/Wrong RT for X1

- 1 pts Missing/Wrong RT for X3

- 10 pts unanswered

- 1 pts Wrong format