CS145 Transaction Homework

Instructions

- ANSWERS are included below the question, in white text.
    - E.g. Highlight this text.
- Try to solve each problem yourself before looking at its solution.

1. Disk v main memory

You are working as the administrator for a database, and you are responsible for performance tuning. Your database server has a disk which can perform sequential I/O at 170MB/s, but when writing random 4KB chunks it only operates at 1MB/s.

Assume your database receives transactions that modify 4KB of data (using the convention 1000KB = 1MB)

a) How many transactions can you process per second if you don't use a transaction log? (assume transactions modify data in random locations on disk)

b) How many transactions can you process per second if you set your database to only write to the sequential log without ever updating the tables disk?

c) A typical memory access takes approximately 100ns, with read/write speeds of 25GB/s. How many transactions can you process per second operating only in memory? (assume each transaction requires 1 memory access + the time to write 4KB to memory)

d) Imagine you can perfectly scale your database workload across servers (so 100 servers will process transactions 100x faster -- in reality they would be somewhat slower). If you had a cluster of servers durably processing transactions as in part b), how many would it take to match the performance of a single server processing transactions in non-durable memory as in part c)?

Solution:

---

2. The problem with crashes/aborts

Examine the following transaction:

1. Read AccountA
2. Read AccountB
3. AccountB = AccountB + AccountA
4. AccountA = 0

For every step, explain how the output of the program would change if the computer crashes right before that step is executed.

Solution:

---

3. Atomicity

What is NOT a possible result from an atomic transaction?

a. All changes are made
b. Some changes are made

c. No changes are made

Solution:

---

4. Consistency

Which of the following example is related to consistency in transactions?

A. A teller looking up a balance at one time should not be allowed to see a concurrent transaction involving a deposit or withdraw from the same account. Only when the withdraw transaction commits successfully and the teller looks at the balance again will the new balance be reported.
B. When processing medical tests, there’s a whole set of rules that medical professionals have to follow. Doctors and their staff have to fill in a requisition properly. The specimen collection center has do verify that information, take samples and pass everything on to a lab. The lab that performs the tests must ensure that everything is valid before performing the test.
C. To transfer funds from one account to another involves making a withdraw operation from the first account and a deposit operation on the second. If the deposit operation failed, you don’t want the withdraw operation to happen either.
D. A data warehouse can still keep the contents of its database even facing a system crash or other failure.

Solution:

---

5. Isolation

Which of the following statements about isolation is correct?

A. To maintain the isolation of transaction, the transactions must be executed serially.
B. Isolation guarantees that once a transaction has committed, its effects remain in the database.

C. Isolation can be achieved by wisely scheduling the transactions as if they are executed serially.
D. Isolation of the transactions can be achieved by Write-Ahead Logging

Solution:

---

6. Durability

Given the following transaction T:
A = 1 and B = 2 at the start of the execution

T	R(A)	R(B)	A := A+B	B := A+B	COMMIT

We run this transaction on a database that does not guarantee **Durability**. If our computer crashes after committing T, what are the possible values of A and B after it restarts?

Solution:

---

7. Write-Ahead Logging (WAL)

Transaction T performs the following operations:

1. R(A)
2. R(B)
3. B := B+1, e.g. W(B)
4. A := A+1, e.g. W(A)

Questions:

A) Initially, A = 0, B = 1, and each write operation increases the record's value by 1. Write out the step-by-step changes to both the data and the log on main memory and disk, before T, during each step of T, and for each of the 2 steps in which WAL writes the log and data to disk (7 steps total).

B) Imagine we alter the WAL protocol so that we first commit T, then write the data and log records to disk. Describe which ACID property we lose and give an example why.

C) Now, say we alter our WAL protocol so instead it writes the data to disk before the corresponding log record. Describe what ACID property we lose, and give an example why.

Solution:
(A)

step	MM data	MM log	Disk data	Disk log
0 (before T)				
1				
2				
3				
4				
5 (Write log)				
6 (Write data)				

8. Scheduling

How can transaction scheduling affect the ACID properties of a DBMS? Give an example why.

Solution:

9. Serializability

Consider the following two schedules for transactions T_1 and T_2:

Schedule 1

T_1:		R(B)	W(C)			R(C)
T_2:	W(B)			R(C)	W(A)	
timestamp:	0	1	2	3	4	5

Schedule 2

T_1:	R(B)	W(C)				R(C)
T_2:			W(B)	R(C)	W(A)	
timestamp:	0	1	2	3	4	5

For each schedule, specify whether or not the schedule is conflict serializable and why. If the schedule is conflict serializable, give the equivalent serial schedule.

Solution:

10. Conflict Types

Let's say you are buying a $2.00 cup of coffee at Coupa Cafe. But *just* after you've paid and your debit card transaction is being processed, Stanford decides to levy a draconian tax of 2% of both your account and Coupa Cafe's account. Consider the following simplified interleaving of transactions:

T1:			[Your acct -= $2.00]		[Coupa's acct += $2.00]			
T2:	[Coupa's acct*= .98]						[Your acct*= .98]	
timestamp:	0	1	2	3	4	5	6	7

You can assume that each transaction above consists of a read and a write, in that order, with each read and each write taking up a unit of time.

What are all the pairs of actions that conflict and what type of conflict do they form? Identify them by their timestamp. (Hint: the read in “Coupa’s acct*= .98” occurs at timestamp 0, and the write occurs at timestamp 1).

Solution:

---

11. Conflict Serializability

Are the following schedules conflict serializable? If not, explain why. If yes, provide the serial schedule.

Schedule 1:

T1	**R(A)**		**R(B)**			**W(B)**		
T2		**W(A)**		**R(C)**	**W(C)**		**R(B)**	**W(B)**

Solution:

Schedule 2:

T1	**R(A)**		**R(B)**			**W(A)**		
T2		**R(A)**		**R(C)**	**W(C)**		**R(B)**	**W(B)**

Solution:

---

12. Two-Phase Locking (Strict 2PL)

Which of the following schedules is conflict serializable, but would not be possible under strict 2PL? Remember that the 2PL protocol grabs an X (exclusive) lock for writing and an S (shared) lock for reading.

Schedule 1

T_1:	R(A)	W(A)		R(B)	
T_2:			W(A)		W(B)
timestamp:	0	1	2	3	4

Schedule 2

T_1:	R(A)		R(B)	W(B)	
T_2:		R(A)			R(A)
timestamp:	0	1	2	3	4

Schedule 3

T_1:	R(A)		R(A)		
T_2:		W(A)		W(B)	R(B)
timestamp:	0	1	2	3	4

Solution:

13. Deadlocks

In lecture, we saw that 2PL is susceptible to deadlocks. Provide a series of lock requests made by transactions T1, T2, and T3 over shared resources A, B and C such that a deadlock occurs. Draw the corresponding waits-for graph for your solution.

Solution: