CS 245 Final Exam
Winter 2020

- Please read all instructions (including these) carefully. In the case of any ambiguity, state any assumptions you made alongside your answer.
- There are six problems, some with multiple parts, for a total of 100 points. You may use the entire exam window (until 11:59 Pacific on March 20th) to work on the exam.
- The exam is open-book, but you may not communicate with other people for it. You may also use the Internet during the exam, but keep in mind that many online resources might use terms differently from our course, and that directly copying an answer found online is considered plagiarism under the Stanford honor code and will not be allowed. We believe that the course materials should be all you need to do the exam.
- You may complete this exam digitally (e.g. using the Annotate tools in Preview on Mac), or print it, handwrite your answers legibly, and scan it using a scanner or a mobile app such as GeniusScan. In either case, ensure that the file you upload to Gradescope exactly matches the format of this document and is sharp and aligned.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. Write as legibly as possible, if you choose to handwrite your solutions.
- Solutions will be graded on correctness and clarity. For the long-answer problems, please show your intermediate work. Each problem has a relatively simple to explain solution, and we may deduct points if your solution is much more complex than necessary. Partial solutions will be graded for partial credit.

NAME: _____

SUID: _____

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination. A typed signature is fine.

SIGNATURE: _____

Grading: P1 = 20pt, P2 = 15pt, P3 = 15pt, P4 = 22pt, P5 = 14pt, P6 = 14pt, total = 100pt.

Problem 1: Short Answer Questions (20 points)

1. **(2 points)** In terms of the CAP theorem, which CAP property do each of the following types of systems **give up**? *(Circle exactly one item per row.)*

   i. Systems with serializable transactions: CONSISTENCY AVAILABILITY

   ii. Systems that use 2PC: CONSISTENCY AVAILABILITY

   iii. Eventually consistent systems: CONSISTENCY AVAILABILITY

   iv. Systems with BASE semantics: CONSISTENCY AVAILABILITY

2. **(2 points)** You are running validation on transaction T_2. The only other transaction is T_1, which has already validated but has not finished. T_1 has a read set of {A, B} and write set {B, C}. Which read and write sets of T_2 will allow it to validate? *(Check all that apply.)*

   - ❑ RS = {D}, WS = {A}
   - ❑ RS = {E}, WS = {D, F}
   - ❑ RS = {B, A}, WS = {D}
   - ❑ RS = {A}, WS = {C}

3. **(2 points)** What conditions **must** be met for a system using undo logging only to safely commit a transaction?

   All modified database items are persisted to disk.

4. **(2 points)** Object stores such as Amazon S3 often don't make a lot of guarantees to their clients. Which of the following guarantees **does** Amazon S3 make? *(Circle your answer.)*

   - ❑ Running a LIST operation after a PUT for a new key will show the new object.
   - ❑ Running a GET operation after a PUT will retrieve the new object contents.
   - ❑ PUT operations are atomic.
   - ❑ GET operations from two clients return the same result if called at the same time.

5. **(2 points)** In AWS Aurora, the primary node only sends redo log records to the storage servers to reduce communication costs. However, each storage server has to maintain a buffer to remember the redo log records it received for a short time before applying them to its in-memory data pages. Why can't each server just apply every log record the moment it receives it?

   The primary node only writes each record to a 4/6 quorum, so storage servers may not receive all the log records in order. They need to fill the gaps in their logs before applying the log records.

6. **(2 points)** What does the following CQL (Continuous Query Language) query mean, assuming we are running it against a relation called `users`? *(Circle your answer.)*

   ```
   SELECT DSTREAM(*) FROM users WHERE account_type="professional"
   ```

   - ❑ Compute a table containing all users whose account type is "professional".
   - ❑ Compute a stream containing all users whose account type is "professional".
   - ❑ Compute a stream containing a record each time a "professional" user changes their account type or deletes their account.
   - ❑ Compute a stream containing a record each time a user's account type changes to "professional".

7. **(2 points)** Suppose that we are running a database system with configurable isolation levels, which starts out with two items: A=1 and B=2. Clients then execute the following transactions on the system (possibly concurrently), and both transactions succeed:

   T_1: set A = A × B T_2: set B = B + A

   What final database states could we see after these two transactions run with **serializability** compared to **snapshot isolation**? For each of the states below, check the corresponding if it is possible under each isolation level:

Final state after T_1 & T_2:	A=2, B=4	A=3, B=4	A=3, B=3	A=2, B=3
Possible under serializability?	X		X	
Possible under snapshot isolation?	X		X	X

8. **(2 points)** Which of the following techniques would be useful to mitigate straggler nodes when running a parallel query on a distributed execution engine? *(Check all that apply.)*

   - ❑ 2-phase commits
   - ❑ Launching backup copies of slow tasks
   - ❑ Balancing the data equally between nodes
   - ❑ Organizing the data so that all records with the same key are on the same node

9. **(4 points)** For each of the following threats that a database system might face, circle which **one** of the security mechanisms below would be **most likely** to address the threat:

   (i) A forensic scientist recovers the identity of an individual from an anonymized dataset.

   ACCESS CONTROL ENCRYPTION

   DIFFERENTIAL PRIVACY AUDIT LOGS

   (ii) A network provider snoops on sensitive data being read from the database.

   ACCESS CONTROL ENCRYPTION

   DIFFERENTIAL PRIVACY AUDIT LOGS

   (iii) An unauthorized employee modifies the SALARIES table to give himself a raise.

   ACCESS CONTROL ENCRYPTION

   DIFFERENTIAL PRIVACY AUDIT LOGS

   (iv) A manager at the company accidentally sets the wrong salary for one employee.

   ACCESS CONTROL ENCRYPTION

   DIFFERENTIAL PRIVACY AUDIT LOGS

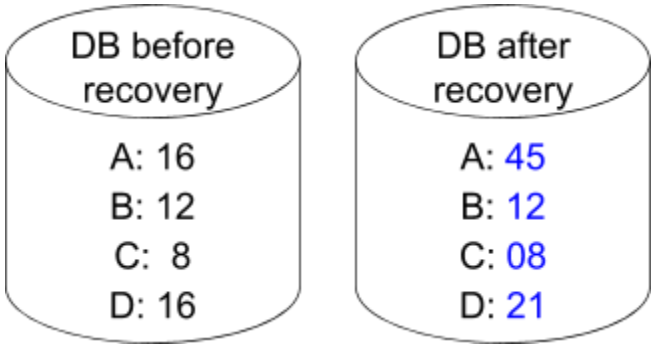
Problem 2: Logging and Failure Recovery (15 points)

Your key-value database has just crashed (fail-stop failure) and you're trying to recover its state from the on-disk copy of the data and log. You're using undo-redo logging with checkpoints.

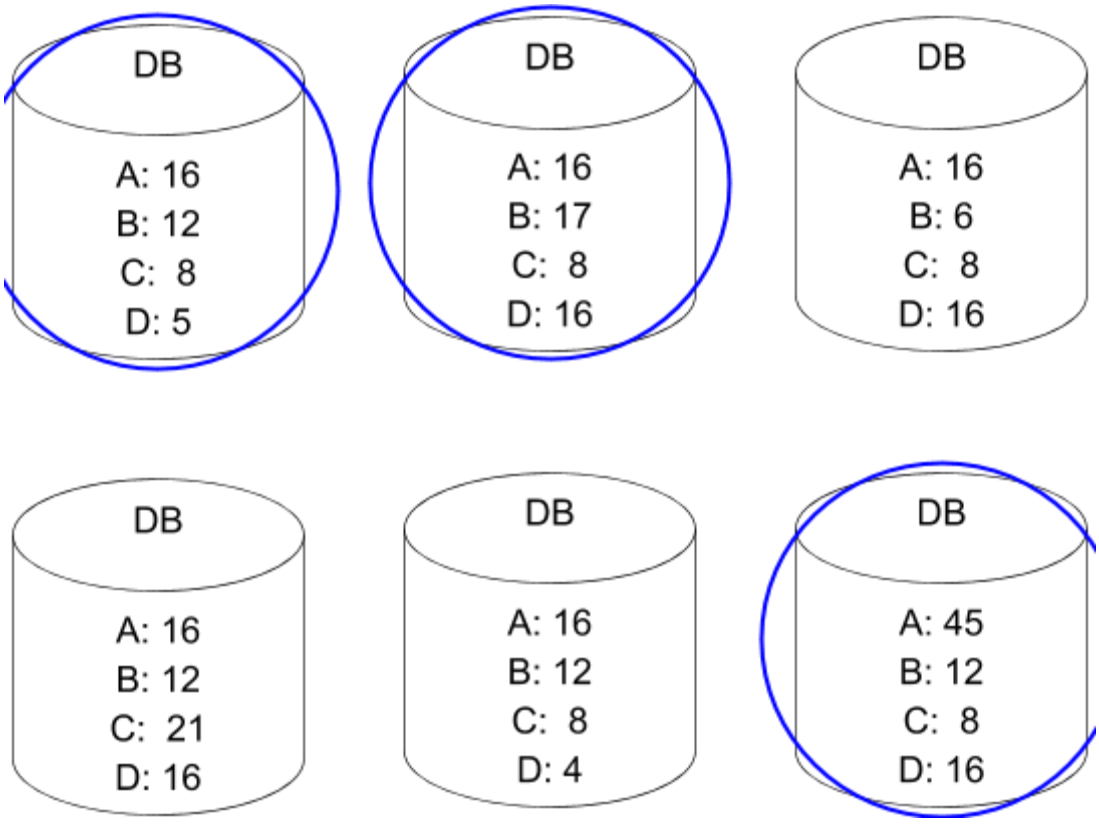
(a) (4 points) Put a check mark next to each record in the log which you have to read in order to recover. Assume each record in a transaction has a pointer to the previous record in the transaction, and the checkpoint start record has pointers to the last record for each active transaction. The format of a log record for a write is *<TransactionID, Key, OldValue, NewValue>*.

Log Entry	Need to Read It?
<T1, start>	
<T2, start>	X
<T1, B, 6, 12>	
<T3, start>	
<T2, D, 4, 16>	X
<T1, commit>	
<T3, C, 21, 8>	
<checkpoint 1 start, T2 and T3 active>	X
<T4, start>	X
<T3, commit>	X
<T4, B, 12, 17>	X
<T5, start>	X
<T2, A, 45, 16>	X
<T4, abort>	X
<checkpoint 1 end>	X
<T5, D, 16, 21>	X
<T5, commit>	X
<checkpoint 2 start, T2 active>	X
<T2, D, 21, 5>	X

(b) (4 points) What should the state of the database be after recovery?



(c) (3 points) Recall that writes to disk are asynchronous, and may be persisted out of order. Given that the log is as above when we boot the machine back up after the crash, which states of the on-disk DB are possible at the time of the crash? Circle the ones which are possible.



(d) (4 points) We now have a new database that uses pure redo logging. It processes the following series of writes and commits. How much memory is the database using, at minimum, after each action? Assume that each value to be written takes 1 unit of memory. Count only memory used for buffered write operations, and not for any other bookkeeping overhead.

Action	Memory Use
<T1, start>	0
<T1, write A>	1
<T2 start>	1
<T2, write B>	2
<T1, write A>	2
<T1, commit>	1
<T2, write D>	2
<T2, commit>	0

Problem 3: Distributed Databases (15 Points)

Lem E. Tweakit is optimizing the two-phase commit (2PC) protocol. He modifies it so that participants do not write transactions to their logs until after *Commit* is received. In all other ways Lem's protocol is identical to standard 2PC. Lem calls his optimized protocol L2PC.

In all parts of this question, consider the L2PC protocol. Assume we have a system where the only failures involve hosts halting with their disks and logs intact, followed by reboots, with no network message loss (if a server sends a message and the target crashes before responding, the server will resend the message after a reboot). Suppose there is a coordinator C and two participants P1 and P2.

(a) (3 points) Assume the following sequence of events happens:

C sends *Prepare Transaction T1* to P1, P2.
P1 sends *Prepared* to C.
P2 sends *Prepared* to C.

What message will be sent after this sequence in L2PC?

C sends *Commit* to P1 and P2.

(b) (3 points) Assume that before that message is sent, P1 crashes and recovers. The sequence of events now looks like this:

C sends *Prepare Transaction T1* to P1, P2.
P1 sends *Prepared* to C.
P2 sends *Prepared* to C.
P1 *crashes*.
P1 *recovers*.

What message will be sent after this sequence in L2PC?

C sends *Commit* to P1 and P2.

(c) (3 points) Assume that after the events of part b, the remainder of the L2PC protocol executes for transaction T1 without further incident. Does T1 commit successfully on P1? On P2? Why?

P1--No, T1 is not in the log.
P2--Yes, everything has gone normally.

(d) (3 points) Let us say 2PC was used instead of L2PC. Would T1 commit successfully on P1? On P2? Why?

P1--Yes, T1 was successfully recovered from the log.
P2--Yes, everything has gone normally.

(e) (3 points) If there is a difference, explain what 2PC guarantees (if any) this violates. If there is no difference, explain how the optimized protocol can provide the same guarantees as the original.

Atomic commitment. Either all nodes or no nodes should commit successfully, but here a transaction can happen on some nodes but not others.

Problem 4: Transaction Schedules (22 points)

(a) (5 points) For each statement below, indicate whether it's true or false.

i. Using finer locks can always be more efficient than coarser locks. TRUE / FALSE

ii. Two-phase locking is guaranteed to produce a schedule whose precedence graph is acyclic. TRUE / FALSE

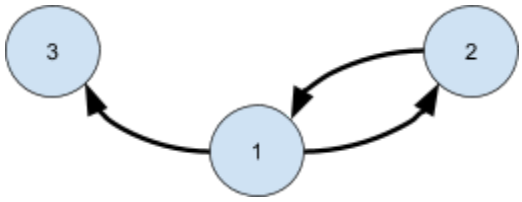
iii. Two-phase locking ensures recoverability. TRUE / FALSE

iv. Strict two-phase locking is guaranteed to produce a conflict serializable schedule. TRUE / FALSE

v. A system providing exclusive access on each object (locking before accessing and unlocking afterwards) guarantees serializability. TRUE / FALSE

(b) (1 point) Draw the precedence graph for the following schedule:

$r_1(A); r_2(A); r_3(B); w_2(A); r_1(B); w_3(B); w_1(A)$



(c) (4 points) Given the following two transactions, provide a schedule that is serializable but not conflict serializable. Explain why your schedule is serializable but not conflict serializable.

T_1: $r_1(A); w_1(B); w_1(B)$ T_2: $r_2(A); w_2(A); w_2(B)$

T_1: $r_1(A); w_1(B);$ $w_1(B)$
T_2: $r_2(A); w_2(A);$ $w_2(B)$
This is equivalent to T_2, T_1. But $w_2(A)r_1(A)$ and $w_1(B)w_2(B)$ are two conflicting actions.

(d) (3 points) Consider the following two transactions,

T_1: $r_1(X)$; $w_1(Y)$; c_1 $\qquad\qquad$ T_2: $w_2(Y)$; $r_2(X)$; c_2

For each of the following schedules, indicate whether it can be generated by 2PL, strict-2PL, both, or neither by circling your answer from the choices below. Note: $L\text{-}S_i(X)$ indicates that T_i acquires a **shared** lock on X, $L\text{-}X_i(X)$ indicates that T_i acquires an **exclusive** lock on X, and $U_i(X)$ indicates that T_i releases its locks on X.

$L\text{-}S_1(X)$; $r_1(X)$; $L\text{-}X_1(Y)$; $U_1(X)$; $L\text{-}S_2(X)$; $w_1(Y)$; $U_1(Y)$; c_1; $L\text{-}X_2(Y)$; $w_2(Y)$; $r_2(X)$; $U_2(X)$; $U_2(Y)$; c_2

2PL / strict 2PL / both / neither (both)

$L\text{-}X_2(Y)$; $L\text{-}S_2(X)$; $w_2(Y)$; $r_2(X)$; $U_2(Y)$; $U_2(X)$; c_2; $L\text{-}S_1(X)$; $r_1(X)$; $U_1(X)$; $L\text{-}X_1(Y)$; $w_1(Y)$; $U_1(Y)$; c_1

2PL / strict 2PL / both / neither (neither)

$L\text{-}S_1(X)$; $r_1(X)$; $L\text{-}X_1(Y)$; $w_1(Y)$; $U_1(Y)$; $L\text{-}X_2(Y)$; $U_1(X)$; c_1; $w_2(Y)$; $L\text{-}S_2(X)$; $U_2(Y)$; $r_2(X)$; $U_2(X)$; c_2

2PL / strict 2PL / both / neither (2PL)

(e) (3 points) "Conservative 2PL" is a variation of 2PL where transactions are required to lock all the items before they start to access (both read and write) them. If any of the items cannot be locked, the transaction does not lock any items and waits.

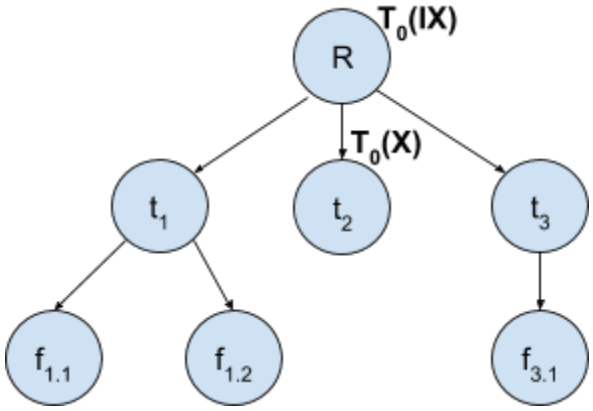
Does Conservative 2PL provide ACR schedules? If yes, explain the reason; otherwise, provide a schedule that is Conservative 2PL but not ACR.
No. Conservative 2PL only requires all locks to be acquired altogether at the beginning but not when to release the locks. So if T_1 releases locks before commit, T_2 can still get the lock and start to read the same object before T_1 commits. A conservative 2PL but not ACR schedule can be (in fact this is not recoverable):
$l_1(A)$ $l_1(B)$ $w_1(A)$ $u_1(A)$ $l_2(A)$ $r_2(A)$ $u_2(A)$ c_2 $w_1(B)$ $u_1(B)$ c_1

Of the properties below, check which ones are held by Conservative 2PL but **not** by 2PL:

- ❑ No conflicting actions in the schedule.
- ❑ No transaction aborts because of deadlock.
- ❑ Transactions that T_i reads from commit earlier than T_i.

- ❑ None of the above.

(f) (6 points) Consider a database system holding the records shown below that implements hierarchical locking:



Suppose that this database is currently running only one transaction T_0. **This transaction has already acquired the locks it needs as shown in the figure.** For each of the other transactions below, indicate whether it can successfully acquire the locks it needs and run. If yes, provide the locks it acquires; otherwise explain the reason. *Note: Consider each transaction below individually, all based on the locks T_0 is holding.*

T_1: Read $f_{3.1}$

Yes. IS(R), IS(t_3), S($f_{3.1}$)

T_2: Read t_1 and all child nodes under t_1, then update $f_{1.2}$

Yes. IX(R), SIX(t_1), X($f_{1.2}$)

T_3: Insert a t_4 under R

No.

Problem 5: Differential Privacy (14 Points)

Alice wants to help scientists better understand the benefits of exercise. In order to do so, she has collected exercise data from the residents of her town, and she wants to build an interface to allow scientists to query the data in a differentially-private way.

Recall that an algorithm *M* is differentially private if it satisfies the following definition: for any two datasets X and Y that differ in *one element*, $Pr[M(X) \in S] \leq e^{\varepsilon} Pr[M(Y) \in S]$.

(a) (2 points) Alice first wants to enable scientists to query the *number* of people in the town who exercise. What is the *sensitivity* of this query?

1

(b) (2 points) Alice decides that a total privacy loss budget of ε = 1 is appropriate for her data. If she wants to allow 5 scientists to make this query once each with independent noise, what value of ε should be allotted to each scientist? Assume that Alice wants the query accuracy to be equal for all 5 queries, and that the scientists can collaborate and share information.

0.2

(c) (2 points) If the number of scientists increases to 10, but the total budget ε remains fixed at 1, will the accuracy of each query increase, decrease, or stay the same?

Decrease

Alice's database has two tables: Ages, which contains the name and age of each resident, and Activities, which contains a tuple <*name, activity*> for each activity a person does. Alice thinks there may be a correlation between the age of the residents and the types of exercise they do, so she wants to allow join queries between the Ages and Activities tables, with Name as the join key. Assume no residents have the same name. The maximum age of any resident is 150.

(d) (2 points) PINQ implements a specialized join operator that groups rows together by key before performing a join. Why is this special join operator required in order for PINQ to provide differential privacy?

A normal join has unbounded sensitivity because one row in the first table could be joined with an arbitrary number of rows in the second table.

(e) (3 points) What is the *stability* of the join between Ages and Activities? Assume that this join is executed using PINQ's specialized join operator.

The stability is 2 because each resident can participate in multiple activities. (We accepted 1 as an answer if the student assumed each resident participates in exactly one activity.)

(f) (3 points) A scientist makes the following query:

```
SELECT AVG(age)
FROM ages JOIN activities
ON ages.name == activities.name
GROUP BY activities.activity;
```

What is the *sensitivity* of this query?

Note: there is an error in the question; PINQ cannot support this query as the join first groups records by name, making the group by activities impossible. Hence, we accepted answers corresponding to several reasonable interpretations of the query:

1. 150 (assuming each person participates in one *unique* activity)
2. 300 (assuming each person participates in exactly one activity)
3. 600 (counting the stability of JOIN and GROUP BY separately)
4. N*150, where N is a fixed, public number of possible activities.

Problem 6: System Design (14 Points)

You have been tasked to implement a high-speed stream processing system that keeps track of the portfolios of users making trades at a stock exchange. Each user's portfolio consists of an integer number of shares in various company stocks (e.g. 1×ACME, 3×WONKA, 2×CHOAM). Your system receives a *trade record* each time that two users make a trade. These records state the numbers and types of shares that the users exchanged, which could cover multiple stocks. For example, one trade record might represent the following trade:

<User 5 gave 2×ACME to User 7, and received 1×WONKA and 1×CHOAM in return.>

We expect that there are several millions of users at the stock exchange, several thousands of different stocks, and a very large number of trades per second.

Your system's job is to maintain a Portfolios table that users can query to **view their current holdings of each stock**. This table should have the schema (UserId, Stock, Quantity), with an entry for each stock that the user owns. For example, if user 1 owns 2×ACME and 1×CHOAM stock, that user should have two records in the table (one for ACME and one for CHOAM). Your system needs to maintain this information **durably**; you'll use disks for this purpose. However, the system also needs to run fast, updating balances as quickly as possible after a trade.

In this problem, we'll ask a few questions about how to design this system. Circle and explain your choice for each one. *Multiple designs might make sense for some of the questions, so you should justify your overall design and the assumptions you made that went into each choice*.

(a) (2 points) What **storage format** would you choose for the Portfolios table?

COLUMN-ORIENTED ROW-ORIENTED

Explain: Operations will generally update whole rows at once, or read both the stock and quantity fields for a user, so there will be fewer I/Os with a row store.

(b) (2 points) Would you attempt to maintain any **order** for the records in Portfolios? If so, specify which attribute(s) you would order the table by.

YES, ORDER BY __UserId, then Stock_____ NO

Explain: Both writes and reads only impact a few user IDs; also sorting by stock within a user will make it easier to look up their holdings for a specific stock.

(c) (2 points) Would you maintain any kind of **index** over the raw table? If so, specify the index type and the key(s) you will index on.

YES, __B-tree____ INDEX BY __UserId, then Stock_____ NO

Explain: A B-tree is easy to maintain and fast for range queries such as reading all of a user's holdings. There may be some good arguments to justify a hash index on this question too if one is worried about the number of I/Os, or even to justify no index.

(d) (2 points) Because each trade can require you to update multiple entries in Portfolios, you want to implement fault tolerance for your system using a log. Which type of log is likely to **minimize the latency** for trade records to be reliably accepted by the system?

UNDO LOG REDO LOG UNDO-REDO LOG

Explain: Undo logs have the highest latency because they require writing to the table on disk before we can commit a transaction (i.e. accept an incoming trade record). Redo logs and undo-redo logs both avoid this, but undo-redo logs are larger than redo ones so may not be as good for minimizing latency. They would help us flush pages from memory while a transaction is running, but our transactions don't touch too many pages, as there are only a few thousand stocks and we chose to cluster the records for each user together.

(e) (4 points) The trade volume becomes so high that you want to make your system parallel, by having multiple threads that can receive incoming trade records and update the Portfolios table. What approach (e.g. locks of some type, validation, etc) would you use for **concurrency control** to manage access to the table? What about concurrency control to the log?

There are multiple valid answers here. One good approach with locking would be to use increment locks, since we are only adding and subtracting values in most transactions. Hierarchical locking at the user level may also be good if read queries access all their stock holdings. Validation is probably not a good idea if some users trade very often, but some students made reasonable assumptions about the workload that would make it work. For access to the log, some kind of locking is needed, but it can happen in memory as threads add records into a buffer to later be written to the log.

(f) **(2 points)** Now that your system is working and running well in parallel, your boss also wants you to start computing a new table, TradeVolumes, which tracks the number of shares of each stock that were traded during each 1-second window of processing time (e.g. 10:00:00 to 10:00:01, 10:00:01 to 10:00:02, etc). The data in this table needs to be accurate for each window (including the current 1-second interval), but it is okay if the table is updated with a short delay after each trade. How can you compute this table with **minimal impact on the latency** of keeping the Portfolios table up-to-date?

The easiest way to compute this TradeVolumes table without having an impact on the latency of updating Portfolios is to have a separate thread that reads the redo log and updates TradeVolumes as write transactions are committed. We could use increment locks to update this table efficiently, or we could try to aggregate multiple updates together in a staging area before applying them to TradeVolumes in order to reduce the contention for entries in TradeVolumes (TradeVolumes only contains a few thousand entries for each 1-second time interval, because there are only a few thousand stocks).