Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.814/6.830 Database Systems: Spring 2021 Quiz II

There are 15 questions and 11 pages in this quiz booklet. To receive credit for a question, answer it according to the instructions given. *You can receive partial credit on questions.* You have **75 minutes** to answer the questions.

Write your name on this cover sheet AND at the bottom of each page of this booklet.
Some questions may be harder than others. Attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
YOU MAY USE A LAPTOP OR CALCULATOR.
PLEASE DO NOT USE THE INTERNET EXCEPT TO VIEW COURSE NOTES, VIDEOS, OR SLIDES.

Do not write in the boxes below

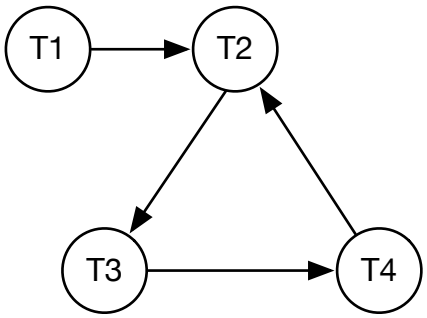
1-3 (xx/24)	4-6(xx/14)	7-9 (xx/21)	10-14 (xx/26)	15 (xx/15)	Total (xx/100)

Name:

I Transactions

Consider the following *waits for* graphs indicating transactions waiting for locks held by other transactions; here, an arrow from T1 to T2 indicates T1 is waiting for a lock T2 holds. For each graph, indicate which are possible serial equivalent schedules for the transactions, assuming the use of strict two-phase locking, and, in the case of deadlock, a random transaction is aborted to break the deadlock.

1. [8 points]:



Which are possible serial equivalent schedules? Assume transactions not listed were aborted because of deadlock and that aborted transactions don't restart.

(Choose yes/no for each choice.)

A. Yes / No T1, T2, T4

B. Yes / No T3, T2, T1

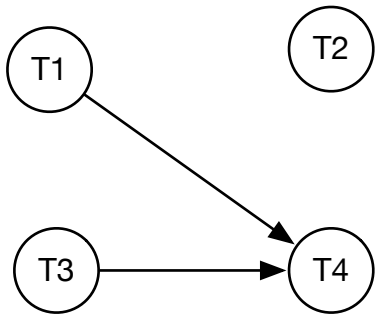
C. Yes / No T4, T1, T2

D. Yes / No T4, T2, T1

Because of two-phase locking, if Ti waited for Tj, then there is either a deadlock or Tj acquired all of its locks before Ti. One of T2, T3, or T4 must have aborted abort because of deadlock. The remaining transactions will serialize in reverse order. Disconnected transactions could serialize at any point. Of the options listed, the only possible choice is T3, T2, T1.

Name:

2. [8 points]:



Which are possible serial equivalent schedules?

(Choose yes/no for each choice.)

A. Yes / No T2, T4, T1, T3

B. Yes / No T1, T4, T3, T2

C. Yes / No T4, T3, T1, T2

D. Yes / No T4, T1, T3, T2

T2, T4, T1, T3 ; T4, T3, T1, T2 ; T4, T1, T3, T2

Name:

II Distributed Transactions

3. [8 points]: Suppose you have a distributed database system with 3 nodes, each responsible for storing and operating on a distinct subset of the data. You are running transactions that span multiple nodes and would like to use 2-phase commit to ensure ACID property for the whole database. Unfortunately, Murphy (of Murphy's Law) plans to cause various failures in your system, hoping your system will violate ACID semantics. It is your job to prove to Murphy that his efforts are futile. For each of the following scenarios, state whether the transaction commits or aborts, and why all nodes will eventually arrive at the same conclusion. Assume that the system is running the 2PC algorithm presented in lecture 17 without optimizations, and everything other than the specified failure execute normally.

(Indicate the outcome in each case and provide a brief explanation for how the system ensures all nodes come to eventual agreement.)

A. COMMIT / ABORT One worker crashes during execution of the transaction.

Abort. Worker does not finish transaction and PREPARE either gets a no or does not receive a response.

B. COMMIT / ABORT The coordinator's PREPARE message to one of the workers is always dropped.

Abort. The coordinator eventually times out and aborts the transaction.

C. COMMIT / ABORT One worker crashes after replying yes in the PREPARE phase, but before receiving the commit message. Every other worker replies yes without crashing.

Commit. The coordinator will continue resending commits until the worker recovers.

D. COMMIT / ABORT The coordinator crashes after sending out some COMMIT messages, but not all of them.

Commit. The coordinator has logged the commit and will recover to resend messages.

Name:

4. [5 points]: Murphy is unhappy about this. He decides to switch tactics and make your network unbearably slow instead. Machines no longer break, but 2PC messages are now up to 2x slower to arrive compared to before (if they arrive at all). Which of the following optimizations can maintain ACID semantics in all cases and reduce commit latency of the protocol by around 2x in at least some outcomes / for some transactions? Assume disk writes are at least as slow as network communication.

(Circle "yes" for optimizations that are both correct and improve speed.)

A. Yes / No Read-only workers vote "read-only" in the PREPARE phase and then do not participate in future phases.

B. Yes / No The coordinator only sends out a second round of messages if the transactions need to abort. All workers assume the transaction commits if they have not heard from the coordinator after some time threshold.

C. Yes / No Now that failures can no longer occur, the coordinator and workers stop writing log records before sending messages.

D. Yes / No The coordinator commits the transaction after receiving yes votes from a majority of workers.

E. Yes / No Workers vote yes preemptively after finishing their portion of the transaction.

Yes, No, Yes, No, Yes

5. [4 points]: Murphy is now exhausted. However, you are very nervous, because your users have been complaining about temporary outages while you battled with Murphy and his shenanigans. You want to optimize your 2PC implementation before he strikes again. Can you improve 2PC such that Murphy can never make your system appear unavailable while maintaining the same ACID guarantees? Why or why not?

(Write your answer, in one sentence or less.)

No. CAP theorem.

6. [5 points]: After some thought you finally noticed Murphy's weakness – Murphy only gets off his couch to muck with your system **after** you start executing user transactions. In other words, your system is now failure-free after receiving user requests, up until when a worker actually begins executing the transaction. Briefly explain how you can exploit this and implement distributed transactions **without** 2-phase commit.

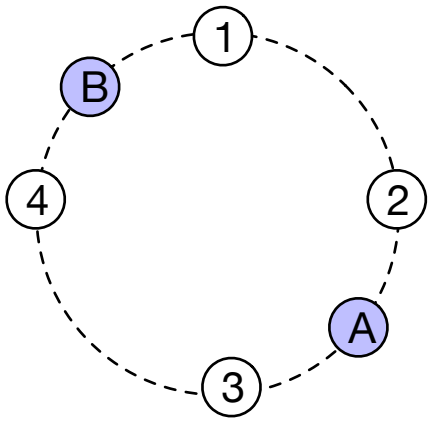
(Write your answer, in one sentence or less.)

We can use the failure-free period to order transactions and then deterministically execute them, like Calvin.

Name:

III Dynamo

Consider a dynamo table with two values, A, and B, replicated on three nodes (i.e., N = 3) in a 4 node cluster as shown below. Initially A=0 and B=0. Client 1 writes A=2, coordinated by node 3, client 2 writes B = 3, coordinated by node 1, and finally client 1 writes B = 4, also coordinated by node 1. Some time later, client 3 reads A and B, both coordinated by node 1.



7. [6 points]: Assume that Dynamo tells the client all writes completed (i.e., succeeded on at least W nodes), the use of strict (not sloppy) quorums, and R=2, W=2. In addition, assume that the coordinating node of the read/write is also one of the replicas doing the read/write. Which of the following are possible values the client 3 could read for A and B?

(Choose yes/no for each choice.)

A. Yes / No B = 0, A = 0

B. Yes / No B = 3, A = 0

C. Yes / No B = 4, A = 0

D. Yes / No B = 3, A = 2

E. Yes / No B = 4, A = 2

F. Yes / No B = 0, A = 2

Because of the use of strict quorums, client 3 will see all of the writes, so the only possible choice is B=4, A=2.

Name:

8. [6 points]: Assume that Dynamo tells the client all writes completed (i.e., succeeded on at least W nodes), the use of sloppy quorums, and R=2, W=2. In addition, assume that the coordinating node of the read/write is also one of the replicas doing the read/write. Which of the following are possible values the client 3 could read for A and B?

(Choose yes/no for each choice.)

A. Yes / No B = 0, A = 0

B. Yes / No B = 3, A = 0

C. Yes / No B = 4, A = 0

D. Yes / No B = 3, A = 2

E. Yes / No B = 4, A = 2

F. Yes / No B = 0, A = 2

Because node 1 coordinated the 2nd and 3rd write, it must have seen both of them. So it will see B=4. Client 1 wrote A=2, which definitely went to node 3. We don't know whether node 1 saw it or not. So A could be 0 or 2. So correct answers are B=4, A=0 and B=4, A=2.

IV MapReduce

9. [9 points]: Suppose you run a MapReduce job to count the number of words in a large document corpus. The map job produces a list containing (word, count) pairs for each word in each document, and the reducers compute the counts per word. You run your job on 1 mapper and 1 reducer and get a runtime of N, and then run it on 100 mappers and 100 reducers and get a runtime of much more than N/100. List three possible reasons for this, using one sentence for each.

(Write your answers in the spaces below.)

1.

2.

3.

We accepted several answers here, including:

1. Some words occur a lot more than others, so some reducers are slow.
2. Some documents are much larger than others so some mappers are slow.
3. Some hardware is slow, leading to stragglers.

Name:

4. Some nodes failed, requiring restarts.
5. Startup or synchronization overheads of various sorts.

Name:

V ARIES

10. [10 points]: Which of the following statements about ARIES recovery are true?

A. True / False If a CLR (Compensation Log Record) is found in the log, the system must have crashed during the REDO phase of the ARIES Algorithm.

B. True / False In theory, if the recovery algorithm keeps crashing during recovery forever, then due to the CLR logs being added the size of the log can keep on increasing forever.

C. True / False Dirty pages are flushed to the disk at checkpoints.

D. True / False We can always get rid of the log before the second last checkpoint.

E. True / False PrevLSN is used to determine where to start the REDO phase from.

All False

Below is a partial log of the system that crashed after LSN 110. On the side is the state at checkpoint (LSN=102) along with the final disk state. Answer the following questions using this information.

LSN	Tx ID	Prev LSN	Type	Page
101	1	95	UP	A
102	-	-	CHECKPOINT	-
103	2	-	SOT	-
104	1	101	UP	A
105	3	100	UP	C
106	2	103	UP	B
107	1	104	COMMIT	-
108	3	105	UP	D
109	2	106	UP	A
110	2	109	COMMIT	-

Checkpoint (LSN=102) State:

Last LSN	Tx ID
101	1
100	3

Dirty Page	recLSN
C	100

Final Disk State:

PageId	PageLSN
A	101
B	106
C	100
D	108

11. [4 points]: Why is A not in the dirty page table at checkpoint (LSN=102)?

12. [4 points]: At which LSN does the REDO phase start?

13. [4 points]: During REDO, which updates can be skipped? List the LSNs of these records.

14. [4 points]: After recovery, can the PageLSN for Page D on disk be greater than 108? Briefly explain why.

1. It must have been flushed before LSN=102.

2. 100

3. 100,101,106,108

4. Yes - because of CLRs.

Name:

VI Distributed Query Processing

Consider a distributed database with 3 tables A, B, and C. All 3 tables have 100 columns - `c1`, `c2` ... `c100`. Each column occupies 10 bytes. Tables A and B have 1 billion tuples and are 1 TB each in size. Table C has 10 million tuples and is 10 GB in size. `c1` is the primary key of all three tables. Assume that disk is about as fast as network, and CPU costs are negligible. Further, assume that data is not heavily skewed (e.g., data is mostly uniform, not huge numbers of duplicate values.)

Users of this database run only the following query:

```
SELECT A.c2, B.c1, B.c2, ... ,B.c100, C.c1, C.c2, ... ,C.c100
FROM A, B, C
WHERE A.c1 = B.c2 AND B.c3 = C.c3
ORDER BY C.c3
```

`A.c1 = B.c2` is a foreign key join i.e, `B.c2` is a foreign key referencing `A.c1`.

The distributed database runs on a cluster of 10 machines with 100 GB of RAM and 1 TB of disk each.

Your task is to identify the best way to layout data. You can choose between round robin, range, hash partitioning, and replication of both base tables and intermediate results. Assume that the initial partitioning/replication of data has already been done (i.e., does not incur any additional cost.)

15. [15 points]: If the cardinality of the output is 100 rows, what do you think is the best way to layout the data and execute the query? You should consider both local and distributed aspects of query planning, including how data is repartitioned (if needed), the type and order of joins, and how the final answer is generated. Provide the best initial data layout and the query plan.

The cardinality of the output is 100 tuples. The cardinality of B join C is 100 tuples as A join B is a foreign key join.The main idea here is to layout the data to reduce the cost of loading data from disk while incurring a small network cost for shuffling a few tuples.

Layout: Co-partition B and C on B.c3/C.c3 using hash partitioning. Hash partition A on A.c1.

Query plan: First join B and C using hash join. Build the hash table on C as it fits in memory. The output of B join C is wide (200 columns). So use semi-join reduction to join with A. Send the column B.c2 to the appropriate partition of A. Find the matching tuples using an index nested loop join with A. Send A.c1 and A.c2 back and compute the join output. Collect all data at one node and do an in-memory sort.

Grading details: We did not deduct any points for assuming no index on A.c1.

A major portion of the grade was assigned for copartitioning B and C. Partial points were allocated based on how well your layout and query plan worked together.

End of Quiz II

Name: