1. Indexing

For CSEE 4121, the instruction staff is maintaining a database that will contain a table with following attributes: Columbia ID, student name, grade.

Part A (multiple choice) 1 point
The staff is designing a simple web page that allows students to login to view their grade. Assuming the database has sufficient DRAM to store the index in its entirety, which of the following index designs would provide the fastest read time for this web page:

1. Multi-level index with grade as a search key
2. Secondary index with grade as search key
3. Dense index with student name as search key
4. Dense index with Columbia ID as search key
5. Sparse index with Columbia ID as search key
6. Multi-level index with Columbia ID as search key

Answer:
4. A dense index is generally the fastest, because it can be implemented with just one access to the index, after which it points directly to the correct entry. The search key that makes most sense is the Columbia ID because that's how students would login (student name / grade are not unique attributes).

Part B (multiple choice) 1 point
Now the staff wants to build a dashboard that will allow them to run aggregate statistics on the grades of the class (e.g., average grade, minimum grade, median grade, etc). Once again, assuming the database has sufficient DRAM to store the index in its entirety, which of the following index designs would provide the best performance for computing such statistics:

1. Multi-level index with grade as a search key
2. Secondary index with grade as search key
3. Dense index with student name as search key
4. Dense index with Columbia ID as search key
5. Sparse index with Columbia ID as search key
6. Multi-level index with Columbia ID as search key

Answer:

2. If we want to answer queries about grades, it makes sense to use it as a search key, rather than using another search key such as student name or Columbia ID, which would require us to read in all the data entries. A secondary index would typically be faster than a multi-level index, since it directly points to different ranges of grades, while with a multi-level index we would need two (or more) index lookups to retrieve the appropriate entries.

Part C (multiple choice) 1 point
It turns out that enrollment to CSEE 4121 has been much higher than usual, and unfortunately the server that hosts the web site doesn't contain enough DRAM to store the entire index. Therefore, a hard disk (i.e., spinning disk) needs to be used to both store part of the index, as well as all the actual entries the index is pointing to. In this case, which of the following index designs would provide ideal performance for supporting the grade viewing application in Part A. Assume the search key in this question is the same one you picked for Part A.

1. Multi-level index, where the outer index is a sparse index (stored on DRAM), and the inner index is a dense index (stored on disk). The inner index is stored on the same storage block (i.e., in the same physical storage region) with the entry it is pointing to.
2. Single-level dense index, where part of it is stored on disk and part of it is stored on DRAM.
3. Single-level sparse index (stored on DRAM), where the entries the index points to (stored on disk) contain a pointer to the next subsequent entry. In order to locate the data entry, you have to traverse the entries until you find the one you are looking for.

Answer:
1. A multi-level index is the most efficient, because it would only require one on-disk read (to fetch both the inner index and the data which is stored in the same block). In option 2. at least some of the requests would require two on-disk lookups (for the index and the data). In the case of 3. while the index does not require an on-disk lookup, entries would have to be fetched from disk one by one until the desired entry is found, which could lead to many on-disk lookups.

Part D (check integer) 2 point
For the index that provides the fastest average lookup in Part C, how many disk seeks would be needed to read a single data entry? Assume that a single entry is always stored contiguously on disk, and that the disk is solely serving this database (no other

applications are using it). If your answer is not round, please round it to the closest integer.

Answer:
1. As written above, the multi-level index requires just one disk seek (to fetch both the inner index and the actual data).

Grading:
No partial credit here.

2. Bloom filters - 3 points

In class we learned that the basic Bloom filter design doesn't support deletions. But a friend of yours came up with an (ingenious?) idea for supporting deletes: what if your database maintains two Bloom filters: a normal one (that tracks inserted objects) and another special one, "delete Bloom filter" that tracks deletes. The delete Bloom filter works exactly the same as a normal Bloom filter, except that it marks the Bloom filter's array with 1's only for delete operations (and not for insert operations). E.g., delete("cat") would mark '1' in the array entries that are the outputs of the hash functions applied to "cat".

Then, before the database looks up an item from disk, it first checks both the normal and delete Bloom filters. If the corresponding entries for that item in the delete Bloom filter are all '1's, the database can return: "object deleted", and it is assumed not to be stored in the database, removing the need to look it up from disk.

Would this idea work? Please explain.

Answer:
No. This solution has all kinds of problems. For example, an item that gets inserted, then deleted, and then inserted. Or an item that gets inserted, and then a bunch of other items get deleted that "collide" with its bits, causing a false positive in the deleted Bloom filter.

Grading:
1 point for answering yes/no correctly + 2 points for clear explanation. If the explanation is not clear enough then 1 point.

3. ACID - multiple choice, each part is 1 point

Please choose which ACID property guarantees each of the following statements (Part A-E).

Part A

Your database needs to ensure that the “student ID” field is not NULL.

- Atomicity
- Consistency
- Isolation
- Durability

Answer:
Consistency. A database constraint (e.g., primary key constraint) is a type of consistency guarantee.

Part B

If there’s a power outage, after rebooting the server, the database’s state reflects all committed transactions.

- Atomicity
- Consistency
- Isolation
- Durability

Answer:
Durability. After power goes down, anything in memory is lost, so we need to make sure that we can recover committed transactions correctly (and roll back in-progress transactions) using the WAL.

Part C

All bank transfers in your database need to appear as if they were executed one after the other serially, even if they are overlapping in time.

- Atomicity
- Consistency

- Isolation
- Durability

Answer:
Isolation. Serializability is an isolation guarantee.

Part D

If a bug causes your entire server to crash, the database’s state reflects all committed transactions.

- Atomicity
- Consistency
- Isolation
- Durability

Answer:
Durability. A bug that causes the server to crash also leads all data in memory to be erased, which means we need to rely on flushed log entries to recover correctly.

Part E

Discounts applied to the “price” attribute in the Products table either get applied entirely, or not at all.

- Atomicity
- Consistency
- Isolation
- Durability

Answer:
Atomicity. Transactions need to be committed all or nothing.

4. Transactions 2 points

Do any of the ACID properties guarantee that a particular transaction will be executed? If so, please say which and why, and if not, explain why.

Answer:

No. ACID ensures that transactions commit all or nothing, but it does not guarantee they will ever be committed. Deadlocks are possible with transactions, and there is no formal guarantee that the system will not get “stuck”.

Grading: 1 point for yes/no, 1 point for explanation.

5. Conflict serializability

Part A (text box) 1 point

Please explain why the following schedule is not serializable.

R1(A), W2(B), R3(C), W3(A), W2(A), R1(B), R2(B)

Answer:
The conflict graph is cyclical:
T1→T2
T1→T3
T2→T1
T3→T2

Part B (text box) 3 points

In order to try to make the schedule serializable you are allowed to reorder one operation, but you must: 1. keep the order of all other operations, 2. you cannot reorder the operations within the same transaction (e.g. for transaction 1: R1(A) must appear before R1(B)).

If such a reordering is possible, please write it below. If not, please explain why.

Answer:
It is possible. We can move R1(B) to the second position:
R1(A), R1(B) W2(B), R3(C), W3(A), W2(A), R2(B)

Now, the conflict graph is acyclical:
T1→T2
T1→T3
T3→T2

Grading: 2 points for right ordering. 1 extra point for explanation.

6. Deadlock prevention

Part A 2 points

You decide to use conservative 2 phase locking to try and prevent deadlocks. You use the following protocol: each transaction first tries to acquire all locks. If it is able to, it goes ahead and executes the transaction. If not, it unlocks the locks, waits for a second, and tries again.

You need to process the following 2 concurrent transactions:
T1: W1(A), R1(B)
T2: W2(B), R2(A)

You try to process these transactions using a very slow and deterministic database, where each operation (acquiring a lock, releasing a lock, writing data, reading data) always takes exactly 1 second. Assume both transactions start concurrently at the exact same time on this database.

Does this protocol enable the two transactions to make progress? Please explain.

Answer:
No. Unfortunately because of our weird database, the two transactions get into a loop where they can never make progress.
T1 locks A and T2 locks B in the first second, then they can't acquire the shared locks on B and A, and have to try again and unlock in the second second. Then they wait for exactly one second, and again try to exclusively lock A and B. So they are stuck in an infinite loop.

Grading: 1 point for yes/no, 1 point for explanation.

Part B 1 point

You decide to tweak your conservative 2 phase locking protocol to use a different waiting time: if the transaction is unable to acquire the locks, it waits for a random time duration between [1, 10] seconds before trying again.

Does this protocol enable the two transactions to make progress? Please explain.

Answer:

Yes. Because the number of seconds is drawn randomly from 1-10, eventually one of the transactions will back off for a long period, and the other one will be able to lock both values and finish its transaction.

Grading: 1 point for both correct answer and explanation, no partial credit.

7. Locks

Part A 1 points

Is the following schedule serializable? Please explain.

W1(A), R2(B), W1(B), W3(C), R3(A), W2(A)

Answer:
No. We have a cycle in the conflict graph:
T1→T2
T2→T1

Part B 3 points

Your database tries to execute the operations from Part A using strict 2 phase locking. Can strict 2 phase locking complete this schedule?
If yes, please detail the serializable schedule produced by strict 2 phase locking using the standard schedule format.
If no, explain why, and give an example of a transaction that should be aborted to “unfreeze” the schedule.

Answer:
No. We get a cycle in the waits-for graph (T1→T2, and T2→T1)
W1(B) needs to wait for the shared lock obtained by R2(B), but W2(A) has to wait for the lock obtained by W1(A).
If we abort either T1 or T2 we can unfreeze the schedule and allow the other transaction to commit (and release its locks) before retrying the aborted transaction.

Grading: 1 point for yes/no. 1 point for explanation. 1 point for explaining which to abort (we can accept **either** T1 or T2 as the answer for which transaction to abort).

8. Amdahl's law

Suppose we have two systems running a batch of jobs. System A runs 5 jobs in 2 seconds, system B runs the same 5 jobs in 1 seconds. If the answer to one of the questions is not a whole number, please provide the answer rounded up to the first decimal place.

Part A (1 point) text box

What is the throughput of system A? (in jobs/second)

Answer:
5/2=2.5

Part B (1 point) multiple choice

What is the average latency of the jobs processed by system B?

1. 1 seconds
2. 0.5 seconds
3. 0.4 seconds
4. 0.2 seconds
5. We do not have enough information to determine the average latency of system B

5.
We do not have enough information to determine the average latency. For example, for System B maybe 4 of the jobs completed after 0.9 seconds, and the last job completed after 1 second, so the average latency would be close to 1 second. In contrast, we could also have a scenario where 4 jobs completed immediately (in almost 0 seconds), but the last job completed after 1 seconds, so the average latency is about 0.2 seconds.

Part C (1 point) multiple choice

System A's first 3 jobs finish after 1 second, the fourth job returns after 1.5 seconds, and the fifth job returns after 2 seconds.

What is the median latency of the jobs processed by system A?

1. 0.5 seconds
2. 1 seconds
3. 1.3 seconds

4. 1.5 seconds
5. 2 seconds
6. We do not have enough information to determine the median latency of system A

2. (1 second)
There are 5 requests. The median request in terms of latency is the third job when you order all jobs by latency, which finishes after 1 second .

9. WAL

Part A 2 points, multiple choice
Your server crashes. After you reboot it, you read the WAL from disk, and you find the entries below, with the following format: (transaction id, key, old value, new value)

T100, A, 30, 40
T101, B, 1, 3
T100, C, 2, 4
T100, COMMIT
T101, A, 30, 35

You check the value of A in the actual database table. What are possible values for A?
- 30
- 35
- 40
- 30 or 35
- 35 or 40
- 30, 35 or 40

Answer:
30, 35 or 40. Any of these values may be possible in the database, since the only requirement is that log entries will get flushed before the data entries.

Grading: this should be graded all or nothing (need to check if CourseWorks does this automatically).

Part B, 1 point multiple choice
After the crash, before servicing new requests, your database wants to clear the records in the WAL. To do so, it needs to ensure the data on disk is up to date with transactions that have completed. Which transactions should be persisted in the database's tables, and which need to be rolled back?

- Both T100 and T101 can be persisted
- T100 can be persisted and T101 should be rolled back
- T100 should be rolled back and T101 should be persisted
- Both T100 and T101 should be rolled back

Answer:
T100 needs to be persisted (needs to make sure it is up to date on disk) since it has been committed to the log, while T101 should be rolled back, since there is no commit record.

Part B 1 point, text box

What is the value of A in the table on disk after the transactions have been persisted/rolled back?

Answer:
40. T100 has been applied to the table, but T101 has been rolled back.

Question 10 Storage
Part A 2 points

Which one of the following sentences is true (more than one answer can be possible).

1. Flash requires large sequential reads to get optimal read performance
2. Magnetic disk requires large sequential reads to get optimal read performance
3. DRAM requires large sequential reads to get optimal read performance
4. Flash requires large sequential writes to get optimal write performance
5. Magnetic disk requires large sequential writes to get optimal write performance
6. DRAM requires large sequential writes to get optimal write performance

2, 4, 5.
Magnetic disk requires large sequential reads and writes to avoid seek and rotational latencies. Flash has good random read performance, but requires large sequential writes due to its erase/write granularity mismatch, and due to multi-channel interleaving.

Grading: this should be graded all or nothing, need to check whether Courseworks does that.

Part B 2 points

List one advantage and one disadvantage of magnetic disk over flash.

Advantages of magnetic disks: cheaper, don't have write durability problems of flash, no wear leveling.
Advantages of flash: lower latency and higher throughput, no moving parts (more durable against physical movement), supports fast random reads, more predictable performance (no seeks/rotational latencies).

Grading: 1 point for advantage of magnetic disk, 1 point for advantage of flash.

Question 11, JOINs, multiple correct answers, 2 points, no partial grading

Consider two tables: T1 and T2 and four queries: Q1,... Q4. The four queries all perform a simple join between an attribute on T1 and T2, except Q1 performs an inner join, Q2 performs a left join, Q3 performs a right join, Q4 performs a full outer join.

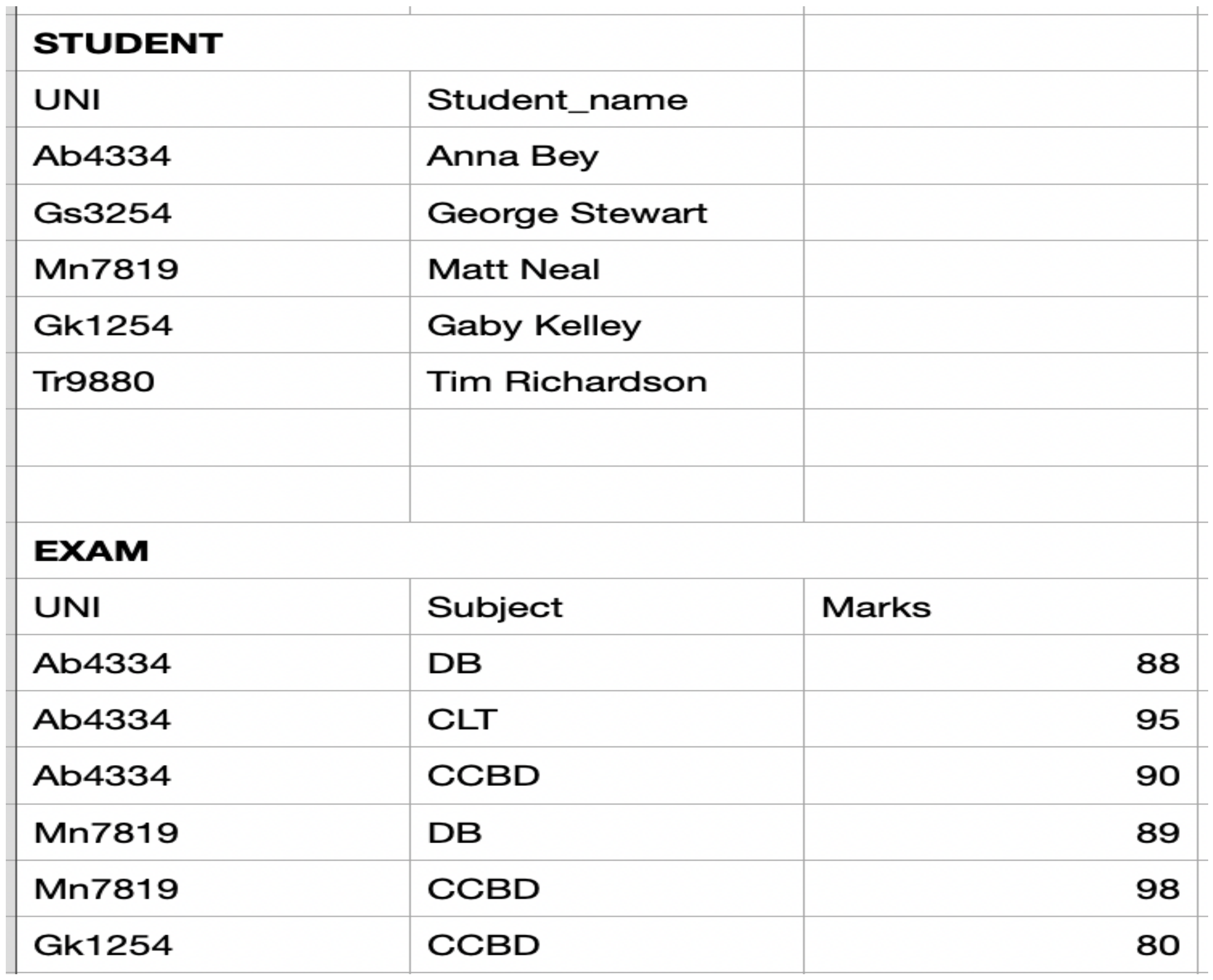
Which one of the following sentences is always correct (more than one sentence can be correct):

1. The result of Q3 is a superset of the result of Q1
2. The result of Q2 is a superset of the result of Q3
3. The result of Q4 is a superset of the result of Q1
4. The result of Q4 is a superset of the result of Q2
5. The result of Q1 is a superset of the result of Q4

1, 3, 4 are correct
Q2 and Q3 are supersets of Q1, and Q4 is a superset of Q2 and Q3.

Grading: This should be graded all or nothing, need to check whether that's the case in CourseWorks.

Question 12, SQL 3 points, text box (accepts integer)



STUDENT		
UNI	Student_name	
Ab4334	Anna Bey	
Gs3254	George Stewart	
Mn7819	Matt Neal	
Gk1254	Gaby Kelley	
Tr9880	Tim Richardson	
EXAM		
UNI	Subject	Marks
Ab4334	DB	88
Ab4334	CLT	95
Ab4334	CCBD	90
Mn7819	DB	89
Mn7819	CCBD	98
Gk1254	CCBD	80

Consider the SQL query given below:

```
SELECT S.UNI, SUM (E.Marks)
     FROM STUDENT S, EXAM E
     WHERE E.UNI = S.UNI AND E.Marks > 84
     GROUP BY S.UNI
```

What is the number of rows returned by the SQL query above?

2.
The last row in the Exam table is filtered (since its grade is lower than 84). Then we inner join group on the ID attribute and group by on the remaining two UNI's and sum their Marks.