6.814/6.830 Database Systems: Fall 2014 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 80 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.
YOU MAY NOT ACCESS THE INTERNET.

Do not write in the boxes below

<table>
<thead>
<tr>
<th>1-4 (xx/20)</th>
<th>5-7 (xx/26)</th>
<th>8-10 (xx/22)</th>
<th>11-15 (xx/32)</th>
<th>Total (xx/100)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name:
I ARIES Recovery

Consider a DBMS running the ARIES recovery algorithm as described in the paper by C. Mohan. The table below shows the log on disk prior to a crash. The system crashes just after T3 commits / log record 19 is written. Two checkpoints are taken at the indicated times. No flushes occur during the execution of these transactions. At the time of checkpoint 1, the dirty page table and the transaction table are both empty.

<table>
<thead>
<tr>
<th>LSN</th>
<th>Xaction ID</th>
<th>Type</th>
<th>PageID/Object name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checkpoint 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>T1</td>
<td>SOT</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>T1</td>
<td>UP</td>
<td>P1/A</td>
</tr>
<tr>
<td>13</td>
<td>T2</td>
<td>SOT</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>T3</td>
<td>SOT</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>T2</td>
<td>UP</td>
<td>P5/B</td>
</tr>
<tr>
<td>16</td>
<td>T2</td>
<td>COMMIT</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>T3</td>
<td>UP</td>
<td>P3/C</td>
</tr>
<tr>
<td>Checkpoint 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>T3</td>
<td>UP</td>
<td>P1/A</td>
</tr>
<tr>
<td>19</td>
<td>T3</td>
<td>COMMIT</td>
<td></td>
</tr>
</tbody>
</table>

1. **[4 points]**: At what LSN does the analysis phase begin?

   *LSN 18, right after the last checkpoint.*

2. **[4 points]**: At what LSN does the REDO phase begin?

   *LSN 12, the smallest recLSN in dirty page table.*

3. **[4 points]**: What is the first LSN that is undone?

   *LSN 12, the largest lastLSN(loser) in transaction table.*

Name:
4. [8 points]:

Fill in the blanks in the tables below to reflect their status in memory at the time of the crash.

<table>
<thead>
<tr>
<th>PageID</th>
<th>recLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>12</td>
</tr>
<tr>
<td>P3</td>
<td>17</td>
</tr>
<tr>
<td>P5</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>lastLSN</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>12</td>
</tr>
</tbody>
</table>

II Locking

Ben Bitdiddle is running an online bookstore, BensBooks.com. He has a transaction, buy_book(customer, book_title), which allows a customer to buy any book as long as it is in stock.

At the moment, Ben is running a promotion in which customers ordering for the first time get two copies of the book when they place an order.

```
L1: order_cnt = SELECT count(*) FROM orders WHERE cust = customer
L2: qty = (order_cnt == 0 ? 2 : 1)
L3: book_cnt = SELECT available_copies FROM inventory WHERE title = book_title
L4: IF book_cnt >= qty
L6: INSERT INTO orders VALUES (customer, book_title, qty)
```

The line numbers L1...L6 are labels for use in describing schedules; they are not part of the code. The expression \((\text{order\_cnt} == 0 ? 2 : 1)\) has the value 2 if \(\text{order\_cnt}\) is 0, and 1 otherwise. Also note that only statements containing SQL queries actually read or modify records in the database.

One day, Alice, Bob, and Steve (all regular customers of BensBooks.com with previous orders) attempt to buy To Kill a Mockingbird from BensBooks.com at the same time, causing the database to run the following three transactions, T1, T2, and T3 concurrently:

T1. buy_book(Alice, To Kill a Mockingbird)
T2. buy_book(Bob, To Kill a Mockingbird)
T3. buy_book(Steve, To Kill a Mockingbird)

Name:
5. [10 points]: Ben has two copies of *To Kill a Mockingbird* in stock. To his surprise, after the above three transactions complete, each customer thinks he or she has successfully purchased the book. Furthermore, the inventory table shows that there is still one copy of *To Kill a Mockingbird* available. Show an interleaving of the above commands that could result in this outcome. Indicate your answer by filling in the 18 lines below, using the L1...L6 labels above, prepended with the transaction IDs (e.g., T1L1, T2L2, etc). Assume each of the lines is executed atomically.

(Write your answer in the space below.)

There are many possible interleavings, but one valid interleaving is shown below. The important feature is that all transactions complete L1-L3 before any transactions complete L5-L6.

1. T1L1
2. T2L1
3. T3L1
4. T1L2
5. T2L2
6. T3L2
7. T1L3
8. T2L3
9. T3L3
10. T1L4
11. T2L4
12. T3L4
13. T1L5
14. T2L5
15. T3L5
16. T1L6
17. T2L6
18. T3L6

Below we reproduce L1–L6 for your reference:


L1: order_cnt = SELECT count(*) FROM orders WHERE cust = customer
L2: qty = (order_cnt == 0 ? 2 : 1)
L3: book_cnt = SELECT available_copies FROM inventory WHERE title = book_title
L4: IF book_cnt >= qty
L6: INSERT INTO orders VALUES (customer, book_title, qty)

Name:
6. [6 points]: Ben realizes that this error probably happened because his database does not have a locking protocol. To fix it, he implements support for strict two-phase locking with record-level shared and exclusive locks. He also modifies the code of buy_book to issue a \texttt{BEGIN TRANSACTION} before L1, and a \texttt{COMMIT} after L6. Explain briefly why this fixes the problem.

(Write your answer in the space below.)

Two-phase locking (2PL) ensures that a transaction cannot release any locks before all needed locks have been acquired. The above schedule is not valid with 2PL because none of the transactions will be able to upgrade their locks at L5. Specifically, after L3, all transactions have a read lock on \textit{To Kill A Mockingbird} in the inventory table. When the transactions try to gain a write lock at L5, a deadlock will occur. If deadlock detection has been implemented correctly, all but one transaction will abort, and when the others restart, they will see the updated value in the inventory table. Put another way, 2PL effectively enforces serializability.

7. [10 points]: A new customer, Annie Hacker, creates a BensBooks.com account and logs into it on 10 different friends’ laptops. On Annie’s cue, her friends simultaneously order a copy of \textit{Harry Potter and the Chamber of Secrets}. A few days later, 20 copies of \textit{Harry Potter and the Chamber of Secrets} show up at Annie’s door. How did this happen? What can Ben do to prevent this sort of scam from happening again in the future?

(Write your answer in the space below.)

This is a classic example of the phantom problem. All 10 friends execute L1 before any of them execute L6. Because the locking is done at the record-level, nothing prevents any of the transactions from inserting a record into the orders table at L6, and all 10 friends successfully purchase two copies of the book for the price of one.

Ben could fix this by implementing next-key locking with a B-tree on the cust field in the orders table. There are several other solutions that could also work. We accepted answers that specified intention locking, predicate locking, and creating a customer table with order count as a separate field, to name a few.
III Two-phase commit

Consider the Two-phase commit protocol, as described in the paper “Transaction Management in the R* Distributed Database Management System” by Mohan et al. Suppose you are running the protocol with three nodes, a coordinator C, and two workers, W1 and W2. Suppose the coordinator stores data and participates in query processing as well (i.e., reads and writes its own data items.)

You are listening on the network connection between C and W1 and C and W2, and you see the following messages. Note that the network is lossy, so there may be additional messages that were sent that you did not see (in which case the recipient also would not receive them), and some messages that you see that the recipient does not receive. This means that some of the messages shown below may not reach their destination.

Unless otherwise stated, assume you are running the standard (neither presumed abort nor presumed commit) variant of the two-phase commit protocol.

At the end of this set of messages, the coordinator crashes.

C <----- W1
---> PREPARE T1
<--- VOTE YES FOR T1
---> PREPARE T2
<--- VOTE YES FOR T2
---> PREPARE T3
<--- VOTE READ ONLY FOR T3
----> COMMIT T2
<---- ACK COMMIT T2

C <----- W2
-----> PREPARE T1
-----> PREPARE T2
<----- VOTE YES FOR T2
<----- VOTE NO FOR T1
-----> PREPARE T3
<----- VOTE READ ONLY FOR T3
----> COMMIT T2

To understand the notation here, the first --> PREPARE T1 indicates that C sends a PREPARE T1 message to W1.

8. [6 points]: For transactions T1/T2/T3 indicate whether they will abort or commit when the coordinator recovers, or whether their outcome is unknown (cannot be determined given the messages above). (Circle ABORT, COMMIT, or UNKNOWN for T1–T3)

T1: ABORT COMMIT UNKNOWN
T2: ABORT COMMIT UNKNOWN
T3: ABORT COMMIT UNKNOWN

Name:
9. [8 points]: Which of the following statements about the operation of the two-phase commit protocol in the above example is true?

(Circle True or False for each item below.)

A. True / False C definitely will send additional messages about transaction T3
   F – it may not, since both voted read only

B. True / False If running the presumed abort variant of the protocol, C definitely will not send any messages regarding transaction T1
   F – no evidence it heard the no vote

C. True / False C definitely will not send any additional messages about transaction T2 to W1
   F, the ACK from T2 to C may have been lost, and C will re-send per the commit, as per page 383 of the paper (“If the recovery process finds a transaction in the committing (respectively, aborting) state, it periodically tries to send the COMMIT (ABORT) to all the subordinates that have not acknowledged and awaits their ACKs. “)

D. True / False C definitely will send additional messages about transaction T2 to W2
   T – W2 has not ack’d COMMIT; C will have to re-send commit.
IV Spark and RDDs

10. [8 points]: Which of the following statements about the paper “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing” by Zaharia et al are true:

(Circle True or False for each item below.)

A. True / False For fault tolerance, the authors propose logging and replaying previous operations on datasets stored persistently on disk.

True. Spark uses logs to store lineage information and replays them to recover datasets that are no longer in memory. If a node has crashed, it will have to load checkpoints of datasets from disk.

B. True / False Like C-Store, Spark employs vertical partitioning of datasets to improve performance.

False. Spark doesn’t impose any particular structure on the datasets it operates on – they could be vertically partitioned but aren’t necessarily.

C. True / False When running on datasets that exceed the size of memory, Spark generally performs about the same as Hadoop.

False. The paper makes a big deal about how it is faster than Hadoop, even when Hadoop is in memory or it is reading 100% of data from disk. The experimental results show that several of their benchmarks are 3–4x faster than Hadoop even when 0% of the data is in memory.

D. True / False The primary goal of the Spark paper is to provide relational algebra-like abstractions (join, group by, etc.) on top of HDFS.

False. The system does provide relational algebra-like operations, but the primary goal of the paper is to discuss the RDD abstraction.

Name:
V Optimistic Concurrency Control

Suppose you are running the serial validation variant of optimistic concurrency control. Below we give example read and write sets for pairs of transactions $T_1$ and $T_2$, where $T_1$ is assigned its transaction ID before $T_2$, and $T_2$ starts its read phase before $T_1$ completes its write phase.

For each pair of read/write sets, indicate whether $T_2$ will commit or abort, or whether it is not possible to tell (“unknown”).

11. [4 points]: $T_1$: Read Set: [A,B], Write Set: [A] $T_2$: Read Set: [A,B], Write Set: [B]
   (Circle ABORT, COMMIT, or UNKNOWN for $T_2$ outcome)

   $T_2$ Outcome: ABORT COMMIT UNKNOWN

   ABORT – read and write sets intersect.

12. [6 points]: $T_1$: Read Set: [A,C], Write Set: [A] $T_2$: Read Set: [B,C], Write Set: [A]
   (Circle ABORT, COMMIT, or UNKNOWN for $T_2$ outcome)

   $T_2$ Outcome: ABORT COMMIT UNKNOWN

   COMMIT – By rule 2 of OCC, write set of $T_1$ does not intersect read set of $T_2$, and $T_1$ completes its write phase before $T_2$ starts its write phase, due to the use of serial validation.

13. [6 points]: $T_1$: Read Set: [A,C], Write Set: [A] $T_2$: Read Set: [B,C], Write Set: [A,B]
   (Circle ABORT, COMMIT, or UNKNOWN for $T_2$ outcome)

   $T_2$ Outcome: ABORT COMMIT UNKNOWN

   COMMIT – for the same reason as in the previous problem; adding $B$ to the write set of $T_2$ doesn’t change anything.

Name:
VI  Dynamo

You are given a Dynamo instance with nodes A-L, as shown below.

This graph has no virtual nodes, a sloppy quorum, and uses an N/R/W configuration of 3/1/3.

14. [8 points]:
If a write for key x arrives and node K is off-line, which of the following is a valid set of nodes to participate in the write:

(Circle “Yes” if the set of write participants is valid, “No” otherwise.)

A. Yes / No  J,L
   No (we accepted answers that circled both this and choice C, due to the ambiguity of whether we wanted the full set or just a valid subset.)

B. Yes / No  J,K,L
   No, K has failed.

C. Yes / No  J,L,A
   Yes.

D. Yes / No  I,J,K
   No, K has failed.

Name:
Now suppose you are using the same Dynamo instance, with an N/R/W configuration of 4/2/2.

15. [8 points]:
   If a read for key y arrives and no nodes in the system have ever been off-line, which of the following is a valid set of nodes to participate in the read?
   (Circle “Yes” if the set of read participants is valid, “No” otherwise.)

   A. Yes / No  C
      No (we accepted answers that circled both this and choice B, due to the ambiguity of whether we wanted the full set or just a valid subset.)

   B. Yes / No  C,F
      Yes.

   C. Yes / No  D,G
      No, without failures, G is not in the read quorum of y.

   D. Yes / No  B,C,D,E
      No, without failures, B is not in the read quorum of y, and the read set is only of size 2.

End of Quiz II