Homework #5 (by Zhaozhe Song) – Solutions
Due: Tuesday Dec 3, 2019 @ 11:59pm

IMPORTANT:
• Upload this PDF with your answers to Gradescope by 11:59pm on Tuesday Dec 3, 2019.
• Plagiarism: Homework may be discussed with other students, but all homework is to be completed individually.
• You have to use this PDF for all of your answers.

For your information:
• Graded out of 100 points; 3 questions total

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Phase Commit</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Distributed Joins</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>Replication</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>Total:</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Revision: 2019/12/08 14:18

Number of Days this Assignment is Late:

Number of Late Day You Have Left:
Question 1: Two-Phase Commit................................. [40 points]

Consider a distributed transaction $T$ operating under the two-phase commit protocol. Let $N_0$ be the coordinator node, and $N_1, N_2, N_3$ be the participant nodes.

The following messages have been sent:

<table>
<thead>
<tr>
<th>time</th>
<th>message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$N_0$ to $N_1$: “Phase1:PREPARE”</td>
</tr>
<tr>
<td>2</td>
<td>$N_0$ to $N_2$: “Phase1:PREPARE”</td>
</tr>
<tr>
<td>3</td>
<td>$N_1$ to $N_0$: “OK”</td>
</tr>
<tr>
<td>4</td>
<td>$N_0$ to $N_3$: “Phase1:PREPARE”</td>
</tr>
<tr>
<td>5</td>
<td>$N_3$ to $N_0$: “OK”</td>
</tr>
</tbody>
</table>

Figure 1: Two-Phase Commit messages for transaction $T$

(a) [10 points] Who should send a message next at time 6 in Figure 1? Select all the possible answers.

- $N_0$
- $N_1$
- $N_2$
- $N_3$
- It is not possible to determine

**Solution:** $N_2$ has to send a response to $N_0$

(b) [10 points] To whom? Again, select all the possible answers.

- $N_0$
- $N_1$
- $N_2$
- $N_3$
- It is not possible to determine

**Solution:** $N_2$ has to send a response to $N_0$

(c) [10 points] Suppose that $N_0$ received the “ABORT” response from $N_3$ at time 5 in Figure 1. What should happen under the two-phase commit protocol in this scenario?

- $N_0$ resends “Phase1:PREPARE” to $N_3$
- $N_0$ resends “Phase1:PREPARE” to all of the participant nodes
- $N_0$ sends “ABORT” all of the participant nodes
- $N_0$ sends “Phase2:COMMIT” all of the participant nodes
- $N_3$ resends “OK” to $N_0$
- It is not possible to determine

Question 1 continues...
Solution: The coordinator ($N_0$) will mark the transaction as aborted. 2PC requires that all participants respond with “OK”.

(d) [10 points] Suppose that $N_0$ successfully receives all of the “OK” messages from the participants from the first phase (i.e., after time 6 in Figure 1). It then sends the “Phase2: COMMIT” message to all of the participants at time 7 but $N_2$ crashes before it receives this message. What is the status of the transaction $T$ when $N_2$ comes back on-line?

- $T$’s status is committed
- $T$’s status is aborted
- It is not possible to determine

Solution: Once the coordinator ($N_0$) gets a “OK” message from all participants, then the transaction is considered to be committed even though a node may crash during the second phase. In this example, $N_2$ would have restore $T$ when it comes back on-line.

Homework #5 continues...
Question 2: Distributed Joins ........................................... [25 points]

Answer the following questions about performing joins in a distributed database. You can assume that the DBMS uses a shared-nothing architecture.

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a2</td>
<td>c3</td>
</tr>
<tr>
<td>a5</td>
<td>c6</td>
</tr>
<tr>
<td>a3</td>
<td>c4</td>
</tr>
<tr>
<td>a5</td>
<td>c2</td>
</tr>
<tr>
<td>a1</td>
<td>c7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>d1</td>
<td>e1</td>
</tr>
<tr>
<td>c2</td>
<td>d5</td>
<td>e2</td>
</tr>
<tr>
<td>c3</td>
<td>d3</td>
<td>e5</td>
</tr>
<tr>
<td>c1</td>
<td>d3</td>
<td>e3</td>
</tr>
<tr>
<td>c3</td>
<td>d6</td>
<td>e8</td>
</tr>
</tbody>
</table>

(a) **R(A,C)**
(b) **S(C,D,E)**

Table 1: Sample database

(a) Consider the relations \( R(A,C) \) and \( S(C,D,E) \) shown in Table 1, where attribute \( S.C \) is a foreign key of attribute \( R.C \).

i. **[10 points]** What is the output of \( R \bowtie S \)?
   - \( \{ (a2,c3,d4,e5), (a2,c3,d6,e8), (a5,c2,d3,e2) \} \)
   - \( \{ (a2,c3,d4,e5), (a5,c2,d3,e2) \} \)
   - \( \{ (c2,b3,a5), (c2,d3,e2), (c3,d4,e5), (c3,d6,e8) \} \)
   - \( \{ (a2), (a5) \} \)
   - \( \{ (a5,c2), (a2,c3) \} \)
   - \( \{ (a5,c2), (a2,c3), (a5,c3) \} \)
   - None of the above

ii. **[10 points]** What is the output of \( S \bowtie R \)?
   - \( \{ (c3,d6,e8), (c3,d3,e5), (c2,d5,e2) \} \)
   - \( \{ (c2,d3,e2), (a1,b2,c3), (c3,d6,e8), (c3,d4,e5), (a5,b3,c2) \} \)
   - \( \{ (c2,d5,e2), (c3,d3,e5) \} \)
   - \( \{ (c2,d3,e2), (c1,d4,e1), (c3,d6,e8), (c1,d2,e3), (c3,d4,e5) \} \)
   - \( \{ (d3,e2), (d6,e8) \} \)
   - \( \{ (c2,d5,e2), (c3,d6,e8) \} \)
   - None of the above

Question 2 continues...
(b) [5 points] In general, is the semijoin operation symmetric for every possible database? That is, is the following equation always true for any possible relations $R_1$ and $R_2$?

\[
R_1 \Join R_2 = \neq R_2 \Join R_1
\]  

\(1\)

- □ Yes
- ■ No
- □ It is not possible to determine

**Solution:** Consider a database where $R_1(A,B) = \{ (a_1,b_1) \}$, $R_2(A,C) = \{ (a_1,c_1) \}$

Then, $(R_1 \Join R_2 = R_1) \neq (R_2 \Join R_1 = R_2)$
Question 3: Replication ................................................. [35 points]

Consider a DBMS using active-passive, master-replica replication with multi-versioned concurrency control. All read-write transactions go to the master node (NODE A), while read-only transactions are routed to the replica (NODE B). You can assume that the DBMS has “instant” fail-over and master elections. That is, there is no time gap between when the master goes down and when the replica gets promoted as the new master. For example, if NODE A goes down at timestamp ① then NODE B will be elected the new master at ②. Note that this is not a realistic assumption but we’re using it to simplify the problem setup.

The database has a single table foo(id, val) with the following tuples:

<table>
<thead>
<tr>
<th>id</th>
<th>val</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yyy</td>
</tr>
<tr>
<td>2</td>
<td>xxx</td>
</tr>
</tbody>
</table>

Table 2: foo(id, val)

For each questions listed below, assume that the following transactions shown in Figure 2 are executing in the DBMS: (1) Transaction #1 on NODE A and (2) Transaction #2 on NODE B. You can assume that the timestamps for each operation is the real physical time of when it was invoked at the DBMS and that the clocks on both nodes are perfectly synchronized (again, this is not a realistic assumption).

<table>
<thead>
<tr>
<th>time</th>
<th>operation</th>
<th>time</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>①</td>
<td>BEGIN;</td>
<td>②</td>
<td>BEGIN READ ONLY;</td>
</tr>
<tr>
<td>②</td>
<td>UPDATE foo SET val = 'aaa';</td>
<td>③</td>
<td>SELECT val FROM foo WHERE id = 1;</td>
</tr>
<tr>
<td>③</td>
<td>UPDATE foo SET val = 'bbb' WHERE id = 2;</td>
<td>④</td>
<td>SELECT val FROM foo WHERE id = 2;</td>
</tr>
<tr>
<td>④</td>
<td>UPDATE foo SET val = 'ccc' WHERE id = 1;</td>
<td>⑤</td>
<td>SELECT val FROM foo WHERE id = 2;</td>
</tr>
<tr>
<td>⑤</td>
<td>COMMIT;</td>
<td>⑥</td>
<td>COMMIT;</td>
</tr>
</tbody>
</table>

(a) Transaction #1 – NODE A

(b) Transaction #2 – NODE B

Figure 2: Transactions executing in the DBMS.

(a) Assume that the DBMS is using asynchronous replication with continuous log streaming (i.e., the master node sends log records to the replica in the background after the transaction executes them). Suppose that NODE A crashes at timestamp ⑤ before it executes the COMMIT operation.

i. **[10 points]** If Transaction #2 is running under SNAPSHOT ISOLATION, what is the return result of the val attribute for its SELECT query at timestamp ③? Select all that are possible.

- □ aaa
- □ bbb
- □ ccc
- ■ xxx
- □ None of the above

Question 3 continues...
Solution: SNAPSHOT ISOLATION means that the transaction will only see the versions that were committed before it started. That means at 4⃝, Transaction #1 has not committed yet so therefore Transaction #2 cannot see any of its versions.

ii. [10 points] If Transaction #2 is running under the READ UNCOMMITTED isolation level, what is the return result of the val attribute for its SELECT query at timestamp 4⃝? Select all that are possible.

- aaa
- xxx
- bbb
- ccc
- yyy
- None of the above

Solution: READ UNCOMMITTED means that it will read any version of the tuple that exists in the database. But what version of tuple 1 that the transaction will read depends on whether the master node shipped the log record over before the query is executed. Since we are doing continuous log shipping, we have no idea. So it could read the version of the tuple that existed before Transaction #1 started (i.e., “xxx”) or after Transaction #1 executed the UPDATE query at 2⃝ (i.e., “aaa”), or after Transaction #1 executed the UPDATE query at 3⃝ (i.e., “bbb”).

(b) [15 points] Assume that the DBMS is using synchronous replication with continuous log streaming. Suppose that both NODE A and NODE B crash at exactly the same time at timestamp 6⃝ after executing Transaction #1’s COMMIT operation. You can assume that the application was notified that the Transaction #1 was committed successfully.

After the crash, you find that NODE A had a major hardware failure and cannot boot. NODE B is able to recover and is elected the new master.

What are the values of the tuples in the database when the system comes back online? Select all that are possible.

- { (1,aaa), (2,bbb) }
- { (1,ccc), (2,bbb) }
- { (1,xxx), (2,xxx) }
- { (1,yyy), (2,bbb) }
- { (1,yyy), (2,xxx) }
- { (1,yyy), (2,zzz) }
- None of the above

Solution: Synchronous means that the replica had fully applied the changes before acknowledging the master. Then the master sent the notification to the client that the txn committed. It is guaranteed that updates were both durable on disk on the master and the replica. The fact that we are doing continuous log shipping doesn’t matter here because the transaction’s changes are either committed or aborted. There cannot be any partial updates to the database.