CARNEGIE MELLON UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE
15-445/645 – DATABASE SYSTEMS (FALL 2018)
PROF. ANDY PAVLO

Homework 5 (by Tupac Shakur) – Solutions

Due: **Monday Dec 3, 2018 @ 11:59pm**

IMPORTANT:

- **Upload this PDF** with your answers to **Gradescope by 11:59pm on Monday Dec 3, 2018**.
- **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

Revision : 2018/12/10 14:04

Question	Points	Score
Two-Phase Commit	40	
Distributed Joins	25	
Replication	35	
Total:	100	

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:

time	message
1	N_0 to N_1: "`Phase1:PREPARE`"
2	N_0 to N_2: "`Phase1:PREPARE`"
3	N_0 to N_3: "`Phase1:PREPARE`"
4	N_2 to N_0: "`OK`"
5	N_1 to N_0: "`OK`"

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_3 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_3 has to send a response to N_0

(c) **[10 points]** Suppose that N_0 never received the "`OK`" response from N_1 at time 5 in Figure 1 (the message got dropped due to a hardware failure). Instead, N_0 "times out" after waiting a certain amount of time. What should happen under the two-phase commit protocol in this scenario?

- ☐ N_0 resends "`Phase1:PREPARE`" to N_1
- ☐ N_0 resends "`Phase1:PREPARE`" to all of the participant nodes
- ☐ N_0 sends "`ABORT`" to N_1
- ■ **N_0 sends "`ABORT`" all of the participant nodes**
- ☐ N_0 sends "`Phase2:COMMIT`" all of the participant nodes
- ☐ N_1 resends "`OK`" to N_0
- ☐ It is not possible to determine

Question 1 continues. . .

Solution: After a timeout, the coordinator (N_0) will assume that N_1 has failed and it 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.

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.

A	B	C
a1	b2	c3
a4	b5	c6
a1	b2	c4
a5	b3	c2
a8	b9	c7

(a) `R(A,B,C)`

C	D	E
c1	d4	e1
c2	d3	e2
c3	d4	e5
c1	d2	e3
c3	d6	e8

(b) `S(C,D,E)`

Table 1: Sample database

(a) Consider the relations `R(A,B,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 $R \ltimes S$?

- ☐ { (a4,b5,c3), (a4,b5,c3), (a3,b2,c3) }
- ☐ { (a5,b3,c2), (c2,d3,e2), (a1,b2,c3), (c3,d4,e5), (c3,d6,e8) }
- ☐ { (c2,b3,a5), (c2,d3,e2), (c3,d4,e5), (c3,d6,e8) }
- ☐ { (a1,b2), (a4,b5) }
- ■ { **(a5,b3,c2), (a1,b2,c3)** }
- ☐ { (a5,b2,c3), (a1,b2,c3), (a5,b3,c2) }
- ☐ None of the above

ii. **[10 points]** What is the output of $S \ltimes R$?

- ■ { **(c3,d6,e8), (c3,d4,e5), (c2,d3,e2)** }
- ☐ { (c2,d3,e2), (a1,b2,c3), (c3,d6,e8), (c3,d4,e5), (a5,b3,c2) }
- ☐ { (c1,d4,e1), (c2,d3,e2), (c1,d2,e3) }
- ☐ { (c2,d3,e2), (c1,d4,e1), (c3,d6,e8), (c1,d2,e3), (c3,d4,e5) }
- ☐ { (d3,e2), (d6,e8) }
- ☐ { (c2,d3,e2), (c3,d6,e8) }
- ☐ None of the above

(b) **[5 points]** In general, is the semijoin operation symmetric for every posssible database? That is, is the following equation always true for any possible relations $R1$ and $R2$?

$$R1 \ltimes R2 =? R2 \ltimes R1 \tag{1}$$

☐ Yes
■ **No**
☐ It is not possible to determine

Solution: To be written...

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:

id	val
1	aaa
2	bbb

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).

time	operation
①	`BEGIN;`
②	`UPDATE foo SET val = 'xxx';`
③	`UPDATE foo SET val = 'yyy' WHERE id = 1;`
④	`UPDATE foo SET val = 'zzz' WHERE id = 2;`
⑤	`COMMIT;`

(a) Transaction #1 – NODE A

time	operation
②	`BEGIN READ ONLY;`
③	`SELECT val FROM foo WHERE id = 1;`
④	`SELECT val FROM foo WHERE id = 2;`
⑤	`SELECT val FROM foo WHERE id = 2;`
⑥	`COMMIT;`

(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 ④ before it executes the third `UPDATE` 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**
- □ xxx
- □ yyy
- □ None of the above

Solution: SNAPSHOT ISOLATION means that the transaction will only see the versions that were committed before it started. That means at ②, 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 ④? Select all that are possible.

■ **bbb**
■ **xxx**
□ zzz
□ 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., "bbb") or after Transaction #1 executed the UPDATE query at ② (i.e., "xxx"). It cannot be "zzz" because that query never got executed before NODE A crashed.

(b) **[15 points]** Assume that the DBMS is using *semi-synchronous* replication with *continuous* log streaming. Suppose that both NODE A and NODE B crash at exactly the same time at timestamp ⑥ 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,xxx), (2,bbb) }
□ { (1,xxx), (2,xxx) }
□ { (1,yyy), (2,bbb) }
□ { (1,yyy), (2,xxx) }
■ { **(1,yyy), (2,zzz)** }
□ None of the above

Solution: Semi-synchronous means that the replica only received the log records from the master but it did not write them to disk. The master sent the notification to the client that the txn committed but it is only guaranteed to be durable on disk on the master and not the replica. When the system come back on-line, we don't know whether the txn was also flushed to disk on the replica.

Thus, the only two correct states of the database are if Transaction #1 never executed or if it did execute. 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.