CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (FALL 2021)
PROF. LIN MA

Homework #5 (by Preetansh Goyal and Joseph Koshakow) – Solutions
Due: **Thursday Dec 2, 2021 @ 11:59pm**

IMPORTANT:
- **Upload this PDF** with your answers to **Gradescope by 11:59pm on Thursday Dec 2, 2021**.
- **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 **120** points; **4** questions total

Revision : 2021/11/26 16:04

Question	Points	Score
Write-Ahead Logging	35	
Replication	33	
Two-Phase Commit	40	
Miscellaneous	12	
Total:	120	

Question 1: Write-Ahead Logging...........................[35 points]

Consider a DBMS using write-ahead logging with physical log records with the STEAL and NO-FORCE buffer pool management policy. Assume the DBMS executes a non-fuzzy checkpoint where all dirty pages are written to disk.

Its transaction recovery log contains log records of the following form:

```
<txnId, objectId, beforeValue, afterValue>
```

The log also contains checkpoint, transaction begin, and transaction commit records.

The database contains three objects (i.e., A, B, and C).

The DBMS sees records as in Figure 1 in the WAL on disk after a crash.

Assume the DBMS uses ARIES as described in class to recover from failures.

LSN	WAL Record
1	<T1 BEGIN>
2	<T1, A, 6, 7>
3	<T1, B, 42, 43>
4	<T2 BEGIN>
5	<T2, C, 33, 71>
6	<T1 COMMIT>
7	<T2, B, 43, 100>
8	<T3 BEGIN>
9	<T3, A, 7, 20>
10	<T2, B, 100, 67>
11	<**CHECKPOINT**>
12	<T3, A, 20, 42>
13	<T2, C, 71, 13>
14	<T2 COMMIT>
15	<T3, A, 42, 66>

Figure 1: WAL

(a) **[10 points]** What are the values of A, B, and C in the database stored on disk before the DBMS recovers the state of the database?

- ☐ A=6, B=100, C=71
- ☐ A=66, B=67, C=13
- ☐ A=7, B:Not possible to determine, C=43
- ☐ A=42, B=42, C=71
- ☐ A=20, B:43, C=Not possible to determine
- ☐ A=20, B:Not possible to determine, C=43
- ☐ A=20, B,C:Not possible to determine
- ☐ A:Not possible to determine, B=42 C=71
- ■ **A:Not possible to determine, B=67, C:Not possible to determine**

☐ A,B,C:Not possible to determine

Solution: The checkpoint flushed everything to disk, but then the data objects A,C were modified by transactions after the checkpoint.

Since we are using NO-FORCE, any dirty page could be written to disk, so therefore we don't know the contents of the database on disk at the crash.

(b) **[5 points]** What should be the correct action on T1 when recovering the database from WAL?

- ■ **do nothing to T1**
- □ redo all of T1's changes
- □ undo all of T1's changes

Solution: T1 committed before the checkpoint. All of its changes were written to disk. There is nothing to redo or undo.

(c) **[5 points]** What should be the correct action on T2 when recovering the database from WAL?

- □ do nothing to T2
- ■ **redo all of T2's changes**
- □ undo all of T2's changes

Solution: T2 committed after the checkpont, so that means the DBMS has to redo all of its changes.

(d) **[5 points]** What should be the correct action on T3 when recovering the database from WAL?

- □ do nothing to T3
- □ redo all of T3's changes
- ■ **undo all of T3's changes**

Solution: T3 never committed. All of its changes should only be undone.

(e) **[10 points]** Assume that the DBMS flushes all dirty pages when the recovery process finishes. What are the values of A, B, and C after the DBMS recovers the state of the database from the WAL in Figure 1?

- □ A=6, B=42, C=33
- □ A=66, B=67, C=13
- □ A=6, B=100, C=13
- ■ **A=7, B=67, C=13**
- □ A=20, B=42, C=71
- □ A=42, B=100, C=33
- □ A=7, B=100, C=71
- □ A=42, B=67, C=13
- □ A=20, B=43, C=33
- □ A=66, B=43, C=71

☐ A=42, B=42, C=13

☐ Not possible to determine

Solution: A = 7 (committed by T1)
B = 67 (rollback to the afterValue made by T2)
C = 13 (rollback to the afterValue made by T2)

Question 2: Replication .... [33 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.

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

id	val
1	x
2	y
3	z

Table 1: **`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.

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

(a) Transaction #1 – NODE A

time	operation
②	`BEGIN READ ONLY;`
③	`SELECT val FROM foo WHERE id = 3;`
④	`SELECT val FROM foo WHERE id = 1;`
⑤	`SELECT val FROM foo WHERE id = 1;`
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 ⑤ <u>before</u> it executes the `COMMIT` operation.

i. **[10 points]** If Transaction #2 is running under `READ COMMITTED`, what is the return result of the `val` attribute for its `SELECT` query at timestamp ⑤? Select all that are possible.

☐ zz
■ **x**
☐ y
☐ yyy
☐ z

☐ xx
☐ None of the above

Solution: READ COMMITTED means that the transaction will only see the versions that were committed. That means at ⑤, 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 ⑤? Select all that are possible.

■ **zz**
■ **x**
☐ y
☐ yyy
☐ z
■ **xx**
☐ 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., "x") or after Transaction #1 executed the UPDATE query at ② (i.e., "xx"), or after Transaction #1 executed the UPDATE query at ④ (i.e., "zz").

(b) **[13 points]** Assume that the DBMS is using *synchronous* replication with *on commit* propagation. Suppose that both NODE A and NODE B crash at exactly the same time at timestamp ⑥ <u>after</u> 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,x), (2,y), (3,z) }
☐ { (1,xx), (2,xx), (3,xx) }
☐ { (1,xx), (2,xx), (3,yyy) }
■ **{ (1,zz), (2,xx), (3,yyy) }**
☐ { (1,x), (2,xx), (3,z) }
☐ { (1,x), (2,xx), (3,xx) }
☐ None of the above

Solution: Synchronous replication with On Commit propagation means that the replica received the log records from the master when Transaction #1 committed. The master sent the notification to the client that the txn committed only when it was guaranteed to be durable on disk on the master and the replica. When the system come back on-line, we know that the txn was also flushed to disk on the replica. Thus, the only correct state of the database is if Transaction #1 did execute. There cannot be any partial updates to the database.

Question 3: 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_1 to N_0: "**OK**"
3	N_0 to N_2: "**Phase1:PREPARE**"
4	N_0 to N_3: "**Phase1:PREPARE**"

Figure 3: Two-Phase Commit messages for transaction T

(a) **[10 points]** Who should send a message next at time 5 in Figure 3? 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 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_2 has to send a response to N_0 N_3 has to send a response to N_0

(c) **[10 points]** Suppose that N_0 received the "**ABORT**" response from N_2 at time 5 in Figure 3. What should happen under the two-phase commit protocol in this scenario?

- ☐ N_0 resends "**Phase1:PREPARE**" to N_2
- ☐ N_2 resends "**OK**" to N_0
- ☐ N_0 sends "**Phase2:COMMIT**" all of the participant nodes
- ■ **N_0 sends "ABORT" all of the participant nodes**
- ☐ N_0 resends "**Phase1:PREPARE**" to all of the participant nodes
- ☐ It is not possible to determine

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. It then sends the "**Phase2:COMMIT**" message to all of the participants but N_1 and N_3 crash before they receives this message. What is the status of the transaction T when N_1 comes back on-line?

☐ T's status is *aborted*

■ **T's status is *committed***

☐ 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_1 and N_3 would have restore T when it comes back on-line.

Question 4: Miscellaneous . . . . [12 points]

(a) **[4 points]** With consistent hashing, if a node fails then all keys must be reshuffled among the remaining nodes.

☐ True

■ **False**

(b) **[4 points]** For a DBMS that uses ARIES, all updated pages must be flushed to disk for a transaction to commit.

☐ True

■ **False**

(c) **[4 points]** During the undo phase of ARIES, all transactions that committed after the last checkpoint are undone.

☐ True

■ **False**