CARNEGIE MELLON UNIVERSITY
COMPUTER SCIENCE DEPARTMENT
15-445/645 – DATABASE SYSTEMS (SPRING 2025)
PROF. JIGNESH PATEL

Homework #6 (by Alexis and Hyoungjoo) – Solutions
Due: **Wednesday April 20, 2025 @ 11:59pm**

IMPORTANT:

- Enter all of your answers into **Gradescope by 11:59pm on Wednesday April 20, 2025**.
- **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; **4** questions total

Revision : 2025/04/22 21:47

Question	Points	Score
ARIES	28	
Two-Phase Commit	24	
Distributed Query Plan	18	
Miscellaneous	30	
Total:	100	

Question 1: ARIES . . . . [28 points]

RusTub uses ARIES recovery with fuzzy checkpoints. It also has a background thread that may arbitrarily flush a dirty bufferpool page to disk at any time.

For this question, assume objects `X`, `Y`, `Z` reside in three different pages `X`, `Y`, `Z`, respectively.

LSN	WAL Record
1	`<T1, BEGIN>`
2	`<T2, BEGIN>`
3	`<T3, BEGIN>`
4	`<T1, UPDATE, prev=1, X, 5→15>`
5	`<T3, UPDATE, prev=3, Z, 700→800>`
6	`<T3, COMMIT, prev=5>`
7	`<CHECKPOINT BEGIN>`
8	`<T2, UPDATE, prev=2, Y, 30→50>`
9	`<T1, UPDATE, prev=4, Z, 800→900>`
10	`<CHECKPOINT END, ATT={T1, T2, T3}, DPT={Z}>`
11	`<T2, UPDATE, prev=8, Y, 50→60>`
12	`<T3, TXN-END>`
13	`<CHECKPOINT BEGIN>`
14	`<T1, COMMIT, prev=9>`
15	`<CHECKPOINT END, ATT={T1, T2}, DPT={?}>`
16	`<T2, UPDATE, prev=11, X, 15→25>`
17	`<T2, UPDATE, prev=16, Y, 60→70>`
18	`<T2, ABORT, prev=19>`
19	`<T2, CLR, prev=18, Y, 70→60, undoNext=16>`

Figure 1: WAL

(a) Suppose the system crashes and, when it recovers, the WAL contains the first 10 records (up to `<CHECKPOINT END, ATT={T1, T2, T3}, DPT={Z}>`). Of the object states below, which states are possibly stored on disk before recovery starts? Select all that apply.

i. **[4 points]** □ `X=5` ■ **`X=15`** □ `X=25` □ `X=35` □ Cannot be determined

ii. **[4 points]** ■ **`Y=30`** ■ **`Y=50`** □ `Y=60` □ Cannot be determined

iii. **[4 points]** ■ **`Z=700`** ■ **`Z=800`** ■ **`Z=900`** □ Cannot be determined

Solution: Page X was not in the DPT at LSN=10. Hence A=15. Page Y was not in the DPT at LSN=10, but there has been a change after the checkpoint start. Hence Y=30 or Y=50. Page Z was in the DPT at LSN=10. Hence it can be any value.

(b) **[4 points]** Select all possible values of DPT in record 15.

□ `X` ■ **`Y`** ■ **`Z`** □ `X, Y` □ `X, Z` ■ **`Y, Z`** □ `X, Y, Z` ■ **Empty**

Solution: Dirty pages at LSN=15: Y and Z (we know X was clean). Since the system can flush dirty pages at any time, any subset of Y, Z is possible.

(c) **[4 points]** For the next 3 questions, assume that the database restarts and finds all log records up to LSN 19 in the WAL. Also assume the DPT is {`Z (recLSN=5)`} at LSN 15.

According to the ARIES recovery protocol, which **log records** need to be redone during the **redo phase** of recovery? Select all that apply.

□ `[LSN 4] <T1, UPDATE, X, 5→15>` ■ **`[LSN 5] <T3, UPDATE, Z, 700→800>`**
□ `[LSN 8] <T2, UPDATE, Y, 30→50>` ■ **`[LSN 9] <T1, UPDATE, Z, 800→900>`**
□ `[LSN 11] <T2, UPDATE, Y, 50→60>` ■ **`[LSN 16] <T2, UPDATE, X, 15→25>`**
■ **`[LSN 17] <T2, UPDATE, Y, 60→70>`** ■ **`[LSN 19] <T2, CLR, Y, 70→60>`**

Solution: After building the DPT, we get the following state: {`X (recLSN=16)`}, {`Y (recLSN=17)`}, {`Z (recLSN=5)`} We redo all LSNs $\geq$ 16 for X, LSNs $\geq$ 17 for Y, and LSNs $\geq$ 5 for Z.

(d) **[4 points]** Select all transactions that should be undone during recovery.
□ T1 ■ **T2** □ T3 □ None of them

Solution: T2 was aborted but not ended, so it will be undone during recovery.

(e) **[4 points]** How many new CLR records will be appended to the WAL after the database fully recovers?
□ 0 □ 1 □ 2 ■ **3** □ 4 □ 5 □ 6

Solution: LSNs 16, 11, and 8 still need to be undone.

Question 2: Two-Phase Commit .... [24 points]

Consider a distributed transaction T operating under the two-phase commit protocol with the ***early acknowledgement optimization***. Let N_0 be the *coordinator* node, and N_1, N_2, N_3 be the *participant* nodes. Let the C be the client application issuing the transaction request. Assume that the client communicates directly with the coordinator node. Assume that the coordinator will treat a node that has failed to respond after ***three*** "`Phase1:PREPARE`" as dead.

The following messages have been sent:

time	message
1	C to N_0: "`REQUEST:COMMIT`"
2	N_0 to N_1: "`Phase1:PREPARE`"
3	N_0 to N_3: "`Phase1:PREPARE`"
4	N_0 to N_2: "`Phase1:PREPARE`"
5	N_3 to N_0: "`OK`"
6	N_0 to N_1: "`Phase1:PREPARE`"
7	N_1 to N_0: "`OK`"
8	N_0 to N_2: "`Phase1:PREPARE`"
9	N_0 to N_2: "`Phase1:PREPARE`"

Figure 2: Two-Phase Commit messages for transaction T

(a) **[6 points]** Who should send message(s) next at time 10 in Figure 2? Select *all* the possible answers.

□ C
■ N_0
□ N_1
■ N_2
□ N_3
□ It is not possible to determine

Solution: There are two possible options. If N_2 is alive, N_2 needs to respond to N_0 with a "`OK`" or "`ABORT`" message. Alternatively, N_0 will decide T is dead and proceed to the "`ABORT`" procedure.

(b) **[6 points]** Assume N_2 responds "`OK`" at time 10, who does N_0 send messages to at time 11? Select *all* the possible answers.

■ C
□ N_0
■ N_1
■ N_2
■ N_3
□ It is not possible to determine

Solution: At the time when all participant nodes have responded with "`OK`", T is considered to be committed. N_0 must notify the participant nodes N_1, N_2, and N_3 of the decision to commit. At this time N_0 should also notify the client application since the protocol is running with the early acknowledgement optimization.

(c) **[6 points]** Suppose that N_0 decides to abort the transaction at time 10 in Figure 2. What should happen under the two-phase commit protocol in this scenario? Select *all* that applies.

□ N_0 resends "**Phase1:PREPARE**" to all of the participant nodes
□ N_0 sends "**ABORT**" to only N_1 and N_3
■ **N_0 sends "ABORT" to the client**
■ **N_0 sends "ABORT" to all of the participant nodes**
□ N_0 resends "**Phase1:PREPARE**" to N_2
□ N_1 sends "**ABORT**" to N_0
□ It is not possible to determine

Solution: The coordinator (N_0) will mark the transaction as aborted and send "**ABORT**" message to the client and all of the participant nodes.

(d) **[6 points]** Again, suppose that N_0 decides to abort the transaction at time 10, but N_1 crashes before any further messages are delivered. What would N_1 decide about the status of the transaction T when it 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) decides to abort the transaction, then the transaction is considered to be aborted even though a node may crash during the second phase. In this example, N_1 would consult the coordinator and finds out it should abort T. NOTE: Another unintended interpretation of the question is that N_1 needs to wait until N_0 comes back online to determine the status of T (hence, the answer would be "It is not possible to determine").

Question 3: Distributed Query Plan .... [18 points]

The CMU-DB team is optimizing distributed databases for aggregations and joins in their new project, AutoDist. This project aims to enable efficient computation of distributed joins followed by aggregations.

Given the following schema:

```
CREATE TABLE artist(PRIMARY KEY a_id INT, a_name VARCHAR,
                    a_nation INT);
CREATE TABLE release(PRIMARY KEY r_id INT, r_name VARCHAR,
                    r_artist INT, r_year INT, r_qty INT);
CREATE TABLE nation(PRIMARY KEY n_id INT, n_name VARCHAR);
```

AutoDist partitions these tables across nodes based on the partition key. Consider the following query:

```
SELECT nation.n_name, SUM(release.r_qty)
FROM artist
JOIN release ON artist.a_id = release.r_artist
JOIN nation ON artist.a_nation = nation.n_id
WHERE release.r_year = 2025
GROUP BY nation.n_name;
```

You can make the following assumptions:

1. There are 8 nodes in the system. Each node can store up to 10,000 rows.
2. The `artist` table contains 20,000 rows.
3. The `release` table contains 40,000 rows, of which 8,000 are released in year 2025.
4. The `nation` table contains 100 rows.

(a) **[5 points]** Which data distribution strategy minimizes the total network data transfer for the given query? Assume that if we choose to partition a table, its rows are distributed evenly across the nodes.

- □ Partition `artist` table by `a_id`, `release` table by `r_id`, and `nation` table by `n_id`.
- □ Partition `artist` table by `a_id`, `release` table by `r_artist`, and `nation` table by `n_id`.
- □ Partition `artist` table by `a_id` and `release` table by `r_id`, and replicate `nation` table across all nodes.
- ■ **Partition `artist` table by `a_id` and `release` table by `r_artist`, and replicate `nation` table across all nodes.**
- □ Partition `release` table by `r_id`, and replicate `artist` and `nation` tables across all nodes.

Question 3 continues…

Solution: This strategy minimizes cross-node data transfer by localizing joins involving `artist` and `release` to individual nodes and avoiding the need to transfer `nation` rows across nodes by replicating them everywhere. Note that replicating the `artist` table is impossible as it exceeds the capacity of each node. NOTE: This question was glitched on Gradescope, so we gave full credit to everyone.

(b) **[5 points]** Assuming the selected strategy from question (a) is implemented, what is the estimated total data transferred over the network for the given query? Assume that *only* the central node can perform aggregations.

- □ Less than 5,000 rows
- ■ **Between 5,001 to 25,000 rows**
- □ Between 25,001 to 100,000 rows
- □ More than 100,000 rows

Solution: Given that `nation` data are replicated across all nodes and joins are localized, the data transfer will be the result of all local joins. Worst case scenario is the max cardinality of the join output: 8,000 rows. If the local join results are distributed evenly, the number becomes 7,000 rows.

(c) **[5 points]** If AutoDist introduces a feature that allows each node to compute a partial aggregation before sending it to the central node for final aggregation, what is the new estimated total network data transfer?

- ■ **Less than 5,000 rows**
- □ Between 5,001 to 25,000 rows
- □ Between 25,001 to 100,000 rows
- □ More than 100,000 rows

Solution: Here, the only data transfer will be the result of local aggregations needing to be combined at a central node for final computation. Since we have 8 nodes and 100 rows in `nation`, this translates into an upper bound of 800 row transfers.

(d) **[3 points]** What are the primary drawbacks of implementing a feature that allows for intermediate aggregation results to be computed on each node before sending these results to a central node for final aggregation? Consider the impact on system resources. Select *all* that apply.

- ■ **It increases the computational load on each node.**
- ■ **It increases the memory usage on each node for storing the intermediate results.**
- □ It increases the total disk usage for storing the persistent tables.
- □ It increases the amount of data transferred over the network.
- □ It decreases the overall system performance.

Solution: Performing intermediate aggregations locally increases each node's computational load and memory usage, as they must handle and store aggregation results. This method reduces network traffic but demands more computational and memory resources.

The third option is incorrect since the amount of persistent data is determined by the partitioning strategy, not by the intermediate aggregation feature. The fourth option is incorrect since the goal is to decrease data transfer. The last option is too general; local aggregations often enhance performance by easing network demands.

Question 4: Miscellaneous .... [30 points]

(a) **[3 points]** A distributed DBMS can immediately commit a transaction under network partitioning without any loss of data consistency.

□ True

■ **False**

Solution: False. This is a violation of the CAP theorem.

(b) **[3 points]** ARIES employs two passes of the log during the recovery process to handle both redo and undo operations.

□ True

■ **False**

Solution: ARIES is a three phase algorithm and requires at most three passes over the log.

(c) **[3 points]** The CAP theorem implies that a distributed system cannot simultaneously guarantee consistency, availability, and partition tolerance.

■ **True**

□ False

Solution: By definition.

(d) **[3 points]** In ARIES, only transactions that commit will have an associated "TXN-END" record in the log.

□ True

■ **False**

Solution: Transactions will commit when the COMMIT log is written, TXN-END is an optimization in the ARIES protocol that indicates it can be removed from the transaction table.

(e) **[3 points]** In the context of distributed DBMS, data replication increases availability but can lead to challenges in maintaining data consistency across nodes.

■ **True**

□ False

Solution: CAP limitations.

(f) **[3 points]** Both PAXOS and Two-Phase Commit protocols can be used to implement distributed transactions.

Question 4 continues. . .

■ **True**

□ False

Solution: 2PC is a degenerate case of Paxos, which is a consensus protocol for distributed transactions.

(g) **[3 points]** In reference to recovery algorithms that use a write-ahead log (WAL). Under NO-STEAL + FORCE policy, a DBMS will have to undo the changes of an aborted transaction during recovery.

□ True

■ **False**

Solution: The FORCE policy guarantees that all changes from committed transactions are immediately written to disk, ensuring durability, while it is the NO-STEAL policy that prevents changes by uncommitted transactions from being written to disk, avoiding the need for undo operations on disk for those transactions.

(h) **[3 points]** Fuzzy checkpoints need to block the execution of all transactions while a consistent snapshot is written to disk.

□ True

■ **False**

Solution: See motivation of checkpointing.

(i) **[3 points]** With consistent hashing, if a node fails, then only a subset and not all keys will be reshuffled among the remaining nodes.

■ **True**

□ False

Solution: See motivation of consistent hashing.

(j) **[3 points]** In a system with strong consistency requirements, it is best for the DBMS to implement active-passive replication with synchronous replication and continuous log streaming.

■ **True**

□ False

Solution: Synchronous replication can ensure strong consistency.