Problem Set 3

Release Date: Wednesday, October 30, 2024

Due Date: Monday, November 25, 2024

Submit to Gradescope: `https://www.gradescope.com/courses/846234/assignments/5248694`

You may work in pairs on this problem set. Only one of you needs to submit on Gradescope, but the other member must be added as a group member on the submission.

The purpose of this problem set is to give you some practice with concepts related to recovery, parallel query processing, two-phase commit, and other papers we read during the second half of the course.

1 Transactions

Consider the following transactions, with initial values of X=1 and Y=2 and Z=3. For writes we also indicate the values that are written (i.e., `WY ;Y=tx*3` means that a write to Y is done that sets its value to `tx*2`.)

```
T1             T2             T3
----------     ----------     --------
tz = RZ        ty = RY        tz = RZ
WY ;Y=tz*5     WZ ;Z=ty-3     WX ;X=tz*2
```

Suppose you know that the first operation that runs is RZ in T1.

1. [2 points]: Assuming there are no deadlocks, list all possible values for X, Y, and Z that could result from an execution of these transactions with rigorous two-phase locking at the serialization level SERIALIZABLE?

2. [2 points]: Now suppose you run these transactions at the serialization level READ COMMITTED. List all possible values for X, Y, and Z that could result from an execution of these transactions? Assume READ COMMITTED is implemented as described in lecture 14, i.e., transactions only hold short read locks (continuing to hold write locks for the duration of the transaction).

3. [2 points]: Now suppose you run these transactions at the serialization level READ UNCOMMITTED. List all possible values for X, Y, and Z that could result from an execution of these transactions? Assume READ UNCOMMITTED is implemented as described in lecture 14, i.e., transactions don't acquire read locks at all (continuing to hold write locks for the duration of the transaction).

2 Serializability

4. [8 points]: Below, we provide 4 interleavings of several concurrent transactions (consisting of READ and WRITE statements on objects).

For each of the interleavings indicate whether:

- The interleaving has an equivalent serial ordering. If so, indicate what the serial ordering is.
- Whether the interleaving would be permitted/could arise under strict two-phase locking.
- Whether the interleaving would be permitted/could arise under rigorous two-phase locking.

Assume in all cases that written values depend on previously read values.

```
Interleaving 1:
1 T1: READ B
2             T2: READ C
3                         T3: READ C
4             T2: READ A
5             T2: WRITE A
6             T2: COMMIT
7                         T3: WRITE A
8                         T3: COMMIT
9 T1: WRITE A
10 T1: COMMIT

Interleaving 2:
1                   T2: READ A
2 T1: READ A
3                   T2: WRITE A
4                   T2: COMMIT
5 T1: WRITE A
6 T1: COMMIT

Interleaving 3:
1                   T2: READ A
2                   T2: READ B
3                   T2: WRITE B
4 T1: READ A
5                   T2: COMMIT
6 T1: WRITE A
7 T1: COMMIT

Interleaving 4:
1 T1: READ A
2               T2: READ A
3               T2: READ B
4               T2: WRITE C
5               T2: COMMIT
6 T1: READ C
7 T1: WRITE A
8 T1: COMMIT

Interleaving 5:
1 T1: READ A
2               T2: WRITE A
3               T2: COMMIT
4 T1: WRITE A
5                         T3: WRITE A
6                         T3: COMMIT
```

```
7 T1: READ B
8 T1: WRITE B
9 T1: COMMIT
```

3 Recovery

5. [6 points]: Suppose you are told that the following transactions are run concurrently on a database system that has just been restarted and is fully recovered, and is running Rigorous 2PL.

```
T1      T2      T3      T4
--      --      --      --
RX
        RY
                RW
                        RZ
WX
RV
        WY
        ABORT
                RY
                        WZ
                        COMMIT
WZ
                WW
```

Suppose the system crashes after executing WW in T3. Assume that each object (e.g., X, Y, etc.) occupies exactly one page of memory.

A. What is a serial order these transactions are equivalent to?

B. Show all of the records that should be in the log at the time of the crash (given your serial order), assuming that there have been no checkpoints and that you are using an ARIES-style logging and recovery protocol. Your records should include all of the relevant fields described in Section 4.1 of the ARIES paper. Also show the status of the transaction table (as described in Section 4.3 of the ARIES paper) after the analysis phase of recovery has run.

C. Suppose you have 2 pages of memory, and are using a STEAL/NO-FORCE buffer management policy as in ARIES. Given the interleaving shown above, for each of the 4 pages used in these transactions, show one possible assignment of LSN values for those pages as they are on disk before recovery runs. You should use the value "?" if the LSN is unchanged from the prior state of the page before these transactions began. Finally, indicate which pages will be modified during the UNDO pass, and which will be modified during the REDO pass. Assume an eviction policy of Most Recently Used.

4 Parallel Query Processing

The standard way to execute a query in a shared-nothing parallel database is to horizontally partition all tables across all nodes in the database. Under such a setting, distributed joins can be computed by repartitioning tables over the join attributes.

Suppose these tables are partitioned across a 5 node distributed database, where each node has 10 TB of disk storage and 2 GB of RAM. Also, assume that:

- The disk can read at 200 MB/sec (for the purposes of this problem, you may ignore differences between sequential and random I/O),
- The network on each node can transmit data at 150 MB/sec, regardless of the number or rate at which other nodes are simultaneously transmitting,
- A computer cannot send over the network and read or write from its disk at the same time,
- CPU costs are negligible relative to disk or network I/O time.
- The network costs to compute the final aggregate are negligible.

For each of the following queries, indicate what partitioning strategy you would recommend. This is the only query run by the system, and your job is to choose the best up-front partitioning for this particular query / setting. Choose between replicating each table on all nodes, or hash, range, or round-robin partitioning it. For hash partitioning, specify the attribute you would use; for range partitioning specify the ranges you would use. Assume there is no index available.

6. [3 points]:

```
SELECT COUNT(*)
FROM R
```

Here, R is 2 TBs and several billion records.

Indicate the best partitioning / replication strategy (and partitioning attributes) for R, as well as a brief explanation.

7. [3 points]:

```
SELECT COUNT(*)
FROM R JOIN S
ON R.a = S.b //S.b is a primary key
WHERE S.c > 100 //selectivity of .1
```

Here, S.b is a primary key, R is 2 TBs and several billion records, and S is 100 MB and several million records.

Indicate the best partitioning / replication strategy (and partitioning attributes) for R and S, as well as a brief explanation.

8. [3 points]:

```
SELECT COUNT(*)
FROM R JOIN S
ON R.a = S.b // S.b is a primary key
JOIN T on R.c = T.d //T.d is a primary key
WHERE T.e = 1 //T.e is uniformly distribute in [1..100]
AND S.c > 100 //selectivity of .1
```

Here, S.b and T.d are primary keys, R is 2 TBs and several billion records, and S is 1 TB and several hundred million records, and T is 1 GB and several million records.

Indicate the best partitioning / replication strategy (and partitioning attributes) for R, S, and T as well as a brief explanation.

5 Two phase commit

Recall that the two-phase commit protocol is used to process transactions over a distributed database on several nodes. In this set of questions, suppose you are running the following three transactions over four values, A, B, C, and D. Assume the use of the basic (e.g., not presumed commit or presumed abort version) of the protocol described in the paper "Transaction Management in the R* Distributed Database Management Systems" by C. Mohan et al.

```
T1:
WA(1)
WB(1)
COMMIT

T2:
WC(2)
COMMIT

T3:
WD(2)
COMMIT
```

Here, the notation WA(1) means "write value 1 to A".

You run these three transactions concurrently on two workers, W1 and W2, as well as a coordinator. Data item A and C are on W1 and B and D are on W2. The system crashes during the execution, and you know that each transaction has begun but do not know how far into its execution it has proceeded. Before recovery begins, you observe the a small piece of the logs on one or more of the workers and the coordinator. Based on just the information in each set of partial logs shown below, indicate whether each transaction has already or definitely will commit or abort, or whether its outcome is unknown.

9. [3 points]:

W1 Log:

```
...
T2 UP C
PREPARE T1 -- VOTE YES
PREPARE T3 -- VOTE YES
COMMIT T3
PREPARE T2 -- VOTE YES
...
```

W2 Log:

```
...
PREPARE T1  -- VOTE YES
....
```

There may be other log entries before and after those shown here. The notation "T2 UP C" means that T2 updated value C.

Indicate whether each of T1, T2, or T3 will commit, abort, or its outcome is unknown.

10. [3 points]:

Coordinator Log:

```
...
COMMIT T2
...
```

W2 Log:

```
...
PREPARE T1  -- VOTE NO
ACK T3
....
```

Indicate whether each of T1, T2, or T3 will commit, abort, or its outcome is unknown.