

Problem Set 3: Transactions and Recovery

Assigned 10/13

Due 10/27

1 Introduction

The purpose of this problem set is to get you thinking about some of the core concepts involved in transactions and recovery. This assignment requires you to be familiar with the following papers, (the last of which we will discuss on the day this assignment is due, so you will need to read ahead!):

1. “ARIES”, by C.Mohan et al.
2. “Principles of Transaction-Oriented Database Recovery” by Haerder and Reuter.
3. “On Optimistic Methods for Concurrency Control” by Kung and Robinson.

2 Questions

1. Suppose your database system never STOLE pages (see the paper from Haerder and Reuter) – e.g., that dirty pages were never written to disk. How would that affect the design of the recovery manager of your database system?
2. In this problem, you are given three different workloads, each of which contains of a set of concurrent transactions (consisting of READ and WRITE statements on objects). For each workload, several different interleavings of execution are given. Your job is to indicate whether, for each of the five interleavings:
 - The given interleaving has an equivalent serial ordering. If so, indicate what the serial ordering is.
 - Whether the given interleaving would be valid using lock-based concurrency control. Assume that locks, when needed, are acquired (with the appropriate lock mode) on an object just before the statement that reads or writes the object and that all locks are released during the COMMIT statement (and no sooner.) If the interleaving is not valid, indicate whether or not it would simply never occur or would result in deadlock, and the time when the deadlock would occur.
 - Whether the given ordering would be valid using optimistic concurrency control. If not, indicate which transaction will be aborted. Assume the use of the Parallel Validation scheme described in Section 5 of the Optimistic Concurrency Control paper by Kung and Robinson, and that the validation and the write phases of optimistic concurrency control happen during the COMMIT statement (and no sooner.)

Assume in all cases that written values can depend on previously read values.(The workloads are shown on the next page.)

Workload 1

<u>Transaction 1</u>	<u>Transaction 2</u>
READ A	READ A
READ B	WRITE B
WRITE C	WRITE A

Interleaving 1:

```

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

```

Interleaving 2:

```

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

```

Workload 2

<u>Transaction 1</u>	<u>Transaction 2</u>	<u>Transaction 3</u>
READ A	READ A	READ A
WRITE A	WRITE B	WRITE A

Interleaving 3:

```

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

```

Workload 3

<u>Transaction 1</u>	<u>Transaction 2</u>
WRITE A	WRITE B
READ B	READ C

Interleaving 4:

```

1 T1:  Write A
2 T2:  Write B
3 T1:  Read B
4 T2:  Read C
5 T1:  Commit
6 T2:  Commit

```

Interleaving 5:

```

1 T1:  Write A
2 T2:  Write B
3 T2:  Read C
4 T2:  Commit
5 T1:  Read B
6 T1:  Commit

```

3. Suppose you are told that the following transactions are run concurrently on a (locking-based, degree 3 consistency) database system that has just been restarted and is fully recovered. Suppose the system crashes while executing the statement marked by an “***” in Transaction 1. Suppose that Transaction 2 has committed, and the state of Transactions 3 and 4 are unknown (e.g., they may or may not have committed.) Assume that each object (e.g., X, Y, etc.) occupies exactly one page of memory.

<u>Trans 1:</u>	<u>Trans 2:</u>	<u>Trans 3:</u>	<u>Trans 4:</u>
x1 = READ X	WRITE Y, 0	z3 = READ X	a4 = READ A
WRITE X, x1 + 1	WRITE B, 0	a3 = READ A	z4 = READ Z
*** y1 = READ Y	x2 = READ X	WRITE A, a3 + 10	WRITE B, (a4-z4)
WRITE Y, y1 + x	WRITE Z, x2	z3 = READ Z	
	y2 = READ Y	WRITE Z, z3 - 10	
	WRITE A, x2 + x1		

- (a) Show an equivalent serial order that could have resulted from these statements, given what you know about what statement was executing when the system crashed. In addition, show an interleaving of the statements from these transactions that is equivalent to your serial order; make sure this serial order could result from a locking-based concurrency control protocol (again assuming that locks are acquired immediately before an item is accessed and released just before the commit statement.)
- (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 you showed above, for each of the 5 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.
4. Suppose you know that the machine your database will run on has a substantial amount of fast, non-volatile memory – that is, memory that would survive a power failure or restart. How would you exploit this memory to improve the performance of the transaction management and recovery mechanisms in your database system? You should consider issues such as the design of the logging system and the need for various phases of recovery. In what ways would this system be faster or simpler than ARIES-style recovery?