ACS theory assignment 2

mnp553 ndp689

December 2021

1 Concurrency Control Concepts

1.1 Show a schedule that is conflict-serializable, but could not have been generated by a conservative two-phase locking (C2PL) scheduler.

T1	T2
W(A)	
	R(B)
W(B)	
	Commit
Commit	

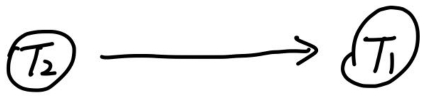


Figure 1: Precedence graph

From the precedence graph of this schedule, we know that there is no cycle in the precedence graph so the schedule is conflict-serializable. When C2PL is used, T1 acquires X locks both for A and B in the beginning, and T2 tries to get a S lock on B while the X lock that has been added by T1 has not been released yet, making this schedule impossible.

1.2 Show a schedule that could be generated by a C2PL scheduler, but not by a strict two-phase locking (S2PL) scheduler.

T1	T2
W(A)	
	R(A)
W(B)	
Commit	
	R(B)
	Commit

When using the C2PL, T1 acquires X locks both for A and B in the beginning, and the X lock for A will be released after executing W(A). Then, T2 can successfully add an S lock on A. But it cannot be generated by the S2PL since the X lock on A cannot be released until T1 commits, making R(A) in T2 impossible to execute.

1.3 Show a schedule that is serializable, but not view-serializable.

1.4 Show a schedule of two transactions that is not conflict-serializable and consists of the minimal possible number of read/write actions to have this property. Explain, in particular also why any schedule with fewer actions is confict-serializable.

The schedule and its precedence graph are shown below. The precedence shows the schedule is not conflict serializable.

T1	T2
W(A)	
	R(A)
W(A)	
Commit	
	Commit

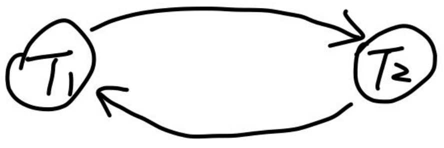
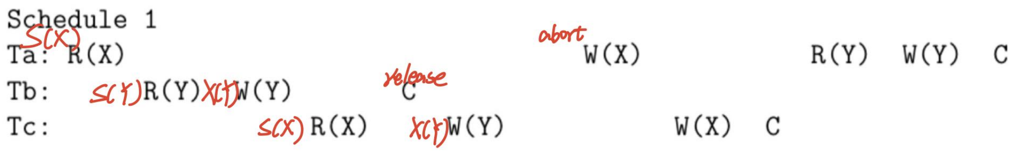


Figure 2: Precedence graph

2 More Concurrency Control

2.1 Could Schedule 1 have been generated by a strict two-phase locking (S2PL) scheduler? Explain why or why not.

No. Because for Ta, it need to update the lock for X from a shared lock to an exclusive lock before W(X), which is impossible since there already is a shared lock on X in Tc.



```
Schedule 1
    S(X)                                          abort
Ta: R(X)                                          W(X)             R(Y)  W(Y)  C
                                 release
Tb:   S(Y)R(Y)X(Y)W(Y)           C
Tc:              S(X) R(X)  X(Y)W(Y)                   W(X)  C
```

Figure 3: Schedule 1 with S2PL

2.2 Could Schedule 1 have been generated by a two-phase locking (2PL) scheduler? Explain why or why not.

No. Because for Ta, it need to update the lock for X from a shared lock to an exclusive lock before W(X), which is impossible since there already is a shared lock on X in Tc.

```
Schedule 1
Ta: S(X) R(X)                                   abort W(X)            R(Y)   W(Y)   C
Tb:       S(Y)R(Y)X(Y)W(Y)  release C
Tc:                  S(X) R(X)  X(Y)W(Y) release(Y)      W(X)   C
```

Figure 4: Schedule 1 with 2PL

2.3 Could Schedule 2 have been generated by a conservative two-phase locking (C2PL) scheduler? Explain why or why not.

No. Because for Tb, it need to acquire a exclusive lock for Y before Tb execution, which is impossible since there already is a exclusive lock on Y in Ta.

```
Schedule 2
    X(X) X(Y)
Ta: R(X)                      R(Y)   W(X)             W(Y)   C
Tb: abort R(Y)   W(Y)         C
Tc:                    R(X)                   W(X)   C
```

Figure 5: Schedule 2 with C2PL

2.4 Could Schedule 2 have been generated by a two-phase locking (2PL) scheduler? Explain why or why not.

No. Because for Ta, it need to update the lock for X from a shared lock to an exclusive lock before W(X), which is impossible since there already is a shared lock on X in Tc.

```
Schedule 2
Ta: S(X) R(X)                      S(Y)R(Y)  abort W(X)             W(Y)   C
Tb:       S(Y)R(Y) X(Y)W(Y) release(Y)C
Tc:                 S(X) R(X)                      W(X)   C
```

Figure 6: Schedule 2 with 2PL

3 Recovery Concepts

3.1 In a system implementing no force and no steal, is it necessary to implement a scheme for undo? What about a scheme for redo? Explain why.

It is not necessary to implement a scheme for undo, because the data of aborted transactions have not been written to disk.

It is necessary to implement a scheme for redo, because if the system crashes before a modified page is written to disk, the modified data will lose. Then, implementing a scheme for redo is necessary.

3.2 What is the difference between non-volatile and stable storage? What types of failures are survived by each type of storage?

(a) Non-volatile storage: access time on it is faster than on stable storage. Not lost on crash, but lost on media failure.

(b) Stable storage: implemented by maintaining multiple copies of information on non-volatile storage. Never lost.

3.3 In a system that implements Write-Ahead Logging, which are the two situations in which the log tail must be forced to stable storage? Explain why log forces are necessary in these situations and argue why they are sufficient for durability.

The log tail must be forced to stable storage in the 2 situations:

(a) When a transaction is committed. Because when a transaction is committed it has been ensured that the record of changes have been written in the log before it's written to disk. This way we know what changes have been done even after a crash.

(b) After modifying pages. Because when a page gets modified we have to know that there has been a change without committing so that we can undo the changes after a crash.

Both situations are sufficient for durability, because we can undo modifications and ensure the committed transactions survive a crash.

4 ARIES

4.1 the state of the transaction and dirty page tables after the analysis phase of recovery;

Applying the ARIES recovery algorithm to the scenario, the transaction table and the dirty page table need to be initialized at the END-CHECKPOINT.

T2 is ended, so it need to be removed from the transaction table, and T3 is committed so it can be labeled 'C'. So, we ended up on the following 2 tables:

Transaction table

XID	Status	LastLSN
1	U	9
3	C	7

Dirty page table

PID	recLSN
2	3
1	4
5	6

4.2 the sets of winner and loser transactions;

The set of winner transactions: {T2,T3}, because T2 has an end log record, so it is removed from the transaction table and T3 is labeled 'C'.

The set of loser transactions: {T1}, because it is in the transaction table and is labeled 'U'.

4.3 the values for the LSNs where the redo phase starts and where the undo phase ends;

The redo phase start with log record with the smallest recLSN in the dirty page table, in this case it would be LSN 3.

The undo phase goes through all loser transactions in ToUndo list and ends at the oldest LSN, in this case it would be LSN 3.

4.4 the set of log records that may cause pages to be rewritten during the redo phase;

The set is {3,4,5,6,9}. Because the redo phase starts by taking the smallest LSN from the dirty page table. Starting from the LSN, Redo scans forward until the end of the log. For each redoable log record encountered, Redo checks whether the logged action must be redone. The action must be redone unless all changes to this page have written to disk.

4.5 the set of log records undone during the undo phase;

The set is {9,4,3}. First, we need to choose the largest LSN among ToUndo, if the LSN is an update, we need to undo the update and add prevLSN to ToUndo. Then, repeat the operation until ToUndo is empty.

4.6 the contents of the log after the recovery procedure completes.

T1 needs to abort because it is the only transaction active. T3 has committed and T2 has ended, thus there is no need to UNDO them. Here I will show the contents of log after the crash.

LSN	PREV_LSN	XACT_ID	TYPE	PAGE_ID	UNDONEXTLSN
11	9	T1	abort	-	-
12	11	T1	CLR:UNDO P2	P2	4
13	12	T1	CLR:UNDO P1	P1	3
14	13	T1	CLR:UNDO P2	P2	NULL
15	14	T1	end	-	-

5 More ARIES

5.1 In the log, A-D are placeholders for the undonextLSN values of the CLRs. For each letter, state the specific value and explain why it has the given value.

A = 5, because the schedule is undoing LSN 6 and the previous LSN is 5.

B = 4, because the schedule is undoing LSN 8 and the previous LSN is 4.

C = NULL, because the schedule is undoing LSN 5 and the previous LSN is NULL.

D = 3, because the schedule is undoing LSN 4 and the previous LSN is 3.

5.2 What are the states of the transaction table and of the dirty page table after the analysis phase of recovery? Explain why.

Applying the ARIES recovery algorithm to the scenario, the transaction table and the dirty page table need to be initialized at the END-CHECKPOINT.

T3 is ended, so it need to be removed from the transaction table. So, we ended up on the following 2 tables:

Transaction table

XID	Status	LastLSN
1	abort	16
2	abort	15

Dirty page table

PID	recLSN
3	3
1	4
2	6
4	8

5.3 Show the additional log contents after the recovery procedure completes. Provide a brief explanation for why any new log records shown need to be added.

First, we need to abort loser transactions. Next, we put the LSN 15 16 to the ToUndo list. And then, we choose the largest LSN among the ToUndo to do the undo operation. Following the undo phase, we get the following answer:

LSN	PREV_LSN	XACT_ID	TYPE	PAGE_ID	UNDONEXTLSN
17	16	T1	abort	-	-
18	15	T2	abort	-	-
19	18	T2	end	-	-
20	17	T1	CLR: UNDO P3	P3	NULL
21	20	T1	end	-	-

5.4 Let us assume that the database is memory-resident, i.e., upon start-up all contents of non-volatile storage are loaded into volatile storage (which is large enough). Thus, no page replacements are ever necessary during normal operation. Furthermore, assume plenty of log buffer space is available and no background page writes are performed. Under these conditions, answer the following (1 paragraph each):

5.4.1 Which log records could trigger writes to stable storage during normal operation in the scenario above? Why?

The commit log record could trigger writes to stable storage. Because in the undo phase, the transactions labeled 'C' in the transaction table are winner transactions and they would not to be undone, which means their modifications were written to stable storage.

5.4.2 Following the ARIES log record structure introduced in the course for update records, is there any information that would not need to be written to stable storage in such a scenario? If so, explain which information and why. Otherwise, justify why all information in these records needs to reach stable storage.

All information in these records needs to reach stable storage. Because when doing redo or undo operations we need to use all these information to recover the system.