Middle East Technical University Department of Computer Engineering

CENG 352 - Database Management Systems
Written Assignment 3

Alper KOCAMAN
2169589

24.05.2020

1

a) Is conflict serializability guaranteed? Why or why not?

In this case, the schedule is conflict serializable. When we consider the the two transaction given in the question, all Exclusive(X) locks held by the transaction be released until after the relevant transaction commits. Thus, this is a strict 2 phase locking. Strict 2-PL ensures that schedule is both recoverable and cascadeless.

b) Is deadlock possible? If so, then assuming that T1 starts first, which transaction(s) would be rolled back (aborted) under the wait-die deadlock prevention scheme?

In strict 2 phase locking, we are free to cascade(basic 2PL) and furthermore, it guarantees strict schedules but still deadlocks are possible.In Wait-Die scheme, if a transaction request for a resource that is locked by other transaction, the DBMS simply checks the timestamp of both transactions and allows the older transaction to wait until the resource is available for execution. Thus, in this case, T2 will be rolled back.

c) Is cascading rollback possible? If not, explain why not. If so, show a scenario that results in cascading rollback.

Two transactions given in the question are strict two phase locking. In this case, all exclusive locks are released with commit. Cascading rollback is not possible.

d) Is conflict serializability guaranteed? Why or why not?

In this case, given two transactions have been changed. When we investigate them, we'll see that all exclusive locks(X) are followed by unlock(U) operation. So, this case is conflict serializable.

e) Is deadlock possible? If so, then assuming that T1 starts first, which transaction(s) would be rolled back (aborted) under the wait-die deadlock prevention scheme?

If T1 transaction starts the execution, then T2 will wait T1 until U(A). If T2 begins the execution then T1 will wait until U(A) in T2. However, deadlock is not possible in this system when we analyze the locks and unlocks order.

f) Is cascading rollback possible? If not, explain why not. If so, show a scenario that results in cascading rollback.

Cascading rollback is possible in this case. In this case, transactions reads the uncommitted data. For example, T1 reads and writes A and that value is read and written by T2. But later on, assume that T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be rollbacked. As it has not committed, we can rollback T2 as well. So it is recoverable with cascading rollback. This is an example scenario that we can apply cascading rollback:
X1(A) R1(A) W1(A) X1(B) U1(A) X2(A) R1(B) W1(B) U1(B) X2(B) R2(B) R2(A) W2(A) U2(A) W2(B) U2(B)

2 Timestamp-based Scheduling

a) Justify whether the operation is accepted or rejected and the RTS, WTS and C(commit bit) of the data items after the operation is executed (or rejected).

Operation	A			B			C		
	RTS	WTS	C	RTS	WTS	C	RTS	WTS	C
R1(A)	1	0	True	0	0	True	0	0	True
R2(B)	1	0	True	2	0	True	0	0	True
R3(A)	3	0	True	2	0	True	0	0	True
W1(A) Rollback	3	0	True	2	0	True	0	0	True
R2(C)	3	0	True	2	0	True	2	0	True
W3(B)	3	0	True	2	3	False	2	0	True
W2(C)	3	0	True	2	3	False	2	2	False
C1	3	0	True	2	3	False	2	2	False
R2(A)	3	0	True	2	3	False	2	2	False
W3(C)	3	0	True	2	3	False	2	3	False
C3	3	0	True	2	3	True	2	3	False
W2(B) Ignore	3	0	True	2	3	True	2	3	True
C2	3	0	True	2	3	True	2	3	True

Figure 1: TS(T1)=1, TS(T2)=2, TS(T3)=3

Operation	A			B			C		
	RTS	WTS	C	RTS	WTS	C	RTS	WTS	C
R1(A)	2	0	True	0	0	True	0	0	True
R2(B)	2	0	True	3	0	True	0	0	True
R3(A)	2	0	True	3	0	True	0	0	True
W1(A)	2	2	False	3	0	True	0	0	True
R2(C)	2	2	False	3	0	True	3	0	True
W3(B) ROLLBACK	2	2	False	3	0	True	3	0	True
W2(C)	2	2	False	3	0	True	3	3	False
C1	2	2	False	3	0	True	3	3	False
R2(A)	3	2	True	3	0	True	3	3	False
W3(C) rejected	3	2	True	3	0	True	3	3	False
C3 rejected	3	2	True	3	0	True	3	3	False
W2(B)	3	2	True	3	3	False	3	3	False
C2	3	2	True	3	3	True	3	3	True

Figure 2: TS(T1)=2, TS(T2)=3, TS(T3)=1

b) Explain the importance of the commit bit, what could have changed if we don't use the commit bit.

Commit bit usage is important in DBMS. DBMS can follow up commit process of the transaction which has written to data lastly. By using commit bit, schedules are recoverable and avoid from cascading abort.

3

a) Write down the contents of the write-ahead-log (WAL), the transaction table and the dirty page table just before the crash. In addition, write down all pages that are in memory, showing their contents.

LSN	Transaction Id	Previous LSN	Type	page ID	Log entry
0	T1	-	Update	P1	Write(A->A1)
10	T2	-	Update	P1	Write(B->B1)
20	T2	10	Update	P2	Write(C->C1)
30	T2	20	Abort	-	-
35	-	-	-	-	Start Checkpoint
40	-	-	-	-	
50	T2	-	CLR	-	Undo T2 LSN 20
60	T2	-	CLR	-	Undo T2 LSN 10
70	T2	60	End	-	-
80	T3	-	Update	P2	Write(D->D1)
90	T1	0	Commit	-	-
100	T1	90	End	-	-
110	T4	-	Update	P1	Write(A1->A2)
120	T3	80	Update	P1	Write(B->B1)
130	T4	110	Commit	-	-

Figure 3: Write Ahead Log Table

Transaction Table

Transaction ID	TLSN	Status
T3	120	Running

T4	130	Committed

Dirty Page Table

Page ID	LSN
P1	10
P2	20

Memory Pages
P1: PageLSN is 120
P2: PageLSN is 80.

b) Write down the contents of the transaction table and dirty page table right after the Analysis phase.

Transaction Table

Transaction ID	TLSN	Status
T3	120	Running
T4	130	Committed

Dirty Page Table

Page ID	LSN
P1	10
P2	20

c) Indicate clearly at which LSN the Redo phase starts. Explain which changes are redone during the Redo phase. Write down the contents of pages in memory at the end of the Redo phase.

d) Write down the contents of the log at the end of the Undo phase. Also state clearly the contents of the pages in the memory, together with their pageLSNs.