CHAPTER 19: DATABASE RECOVERY TECHNIQUES

Answers to Selected Exercises

19.21 Suppose that the system crashes before the [read_item,T3,A] entry is written to the log in Figure 19.1(b); will that make any difference in the recovery process?

(a)

T_1	T_2	T_3
read_item(A)	read_item(B)	read_item(C)
read_item(D)	write_item(B)	write_item(B)
write_item(D)	read_item(D)	read_item(A)
	write_item(D)	write_item(A)

(b)

		A	B	C	D
		30	15	40	20
	[start_transaction,T_3]				
	[read_item,T_3,C]				
*	[write_item,T_3,B,15,12]		12		
	[start_transaction,T_2]				
	[read_item,T_2,B]				
**	[write_item,T_2,B,12,18]		18		
	[start_transaction,T_1]				
	[read_item,T_1,A]				
	[read_item,T_1,D]				
	[write_item,T_1,D,20,25]				25
	[read_item,T_2,D]				
**	[write_item,T_2,D,25,26]				26
	[read_item,T_3,A]				

← System crash

* T_3 is rolled back because it did not reach its commit point.

** T_2 is rolled back because it reads the value of item B written by T_3.

(c)

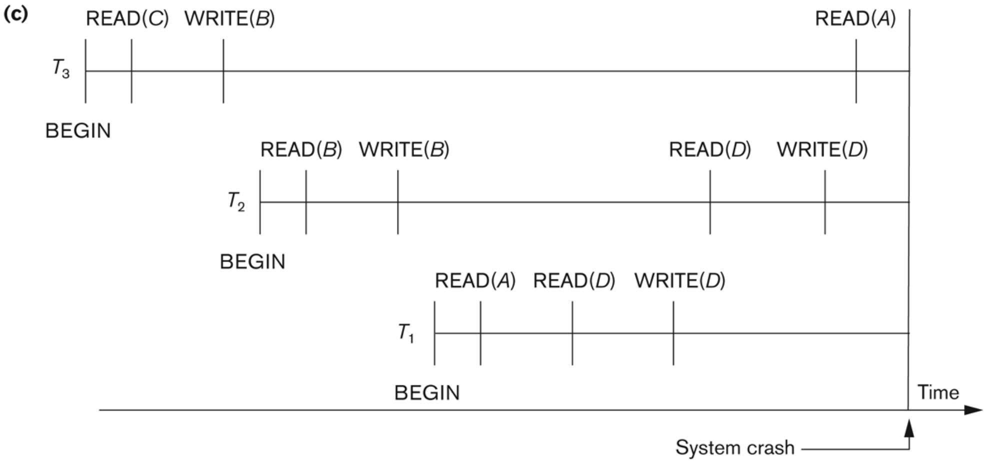


Figure 19.1
Illustrating cascading rollback (a process that never occurs in strict or cascadeless schedules). (a) The read and write operations of three transactions. (b) System log at point of crash. (c) Operations before the crash.

Answer:

There will be no difference in the recovery process, because read_item operations are needed only for determining if cascading rollback of additional transactions is necessary.

19.22 Suppose that the system crashes before the [write_item,T2,D,25,26] entry is written to the log in Figure 19.1(b); will that make any difference in the recovery process?

Answer:

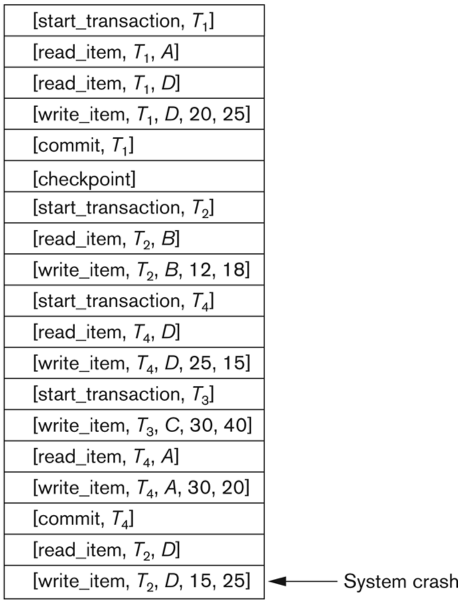
Since both transactions T2 and T3 are not yet committed, they have to be rolled back during the recovery process.

19.23 Figure 19.7 shows the log corresponding to a particular schedule at the point of a system crash for the four transactions T1, T2, T3, and T4 of Figure 19.4. Suppose that we use the immediate update protocol with checkpointing. Describe the recovery process from the system crash. Specify which transactions are rolled back, which operations in the log are redone and which (if any) are undone, and whether any cascading rollback takes place.

Answer:

First, we note that this schedule is not recoverable, since transaction T4 has read the value of B written by T2, and then T4 committed before T2 committed. Similarly, transaction T4 has read the value of A written by T3, and then T4 committed before T3 committed. The [commit, T4] should not be allowed in the schedule if a recoverable protocol is used, but should be postponed till after T2 and T3 commit. For this problem, let us assume that we can roll back a committed transaction in a non-recoverable schedule, such as the one shown in Figure 21.7.
By using the procedure RIU_M (recovery using immediate updates for a multiuser environment), the following result is obtained:
From Step 1 of procedure RIU_M, T2 and T3 are the active transactions. T1 was committed before the checkpoint and hence is not involved in the recovery.
From Step 2, the operations that are to be undone are:
[write_item,T2,D,25]
[write_item,T3,A,30]
[write_item,T2,B,12]
Note that the operations should be undone in the reverse of the order in which they were written into the log. Now since T4 read item B that as written by T2 and read item A that as written by T3, and since T2 and T3 will be rolled back, by the cascading rollback rule, T4 must be also rolled back. Hence, the following T4 operations must also be undone:
[write_item,T4,A,20]
[write_item,T4,B,15]
(Note that if the schedule was recoverable and T4 was committed, then from Step 3, the operations that are to be redone would have been:
[write_item,T4,B,15]
[write_item,T4,A,20]
In our case of non-recoverable schedule, no operations need to be redone in this example.)
At the point of system crash, transactions T2 and T3 are not committed yet. Hence, when T2 is rolled back, transaction T4 should also be rolled back as T4 reads the values of items B and A that were written by transactions T2 and T3. The write operations of T4 have to be undone in their correct order. Hence, the operations are undone in the following order:
[write_item,T2,D,25]
[write_item,T4,A,20]

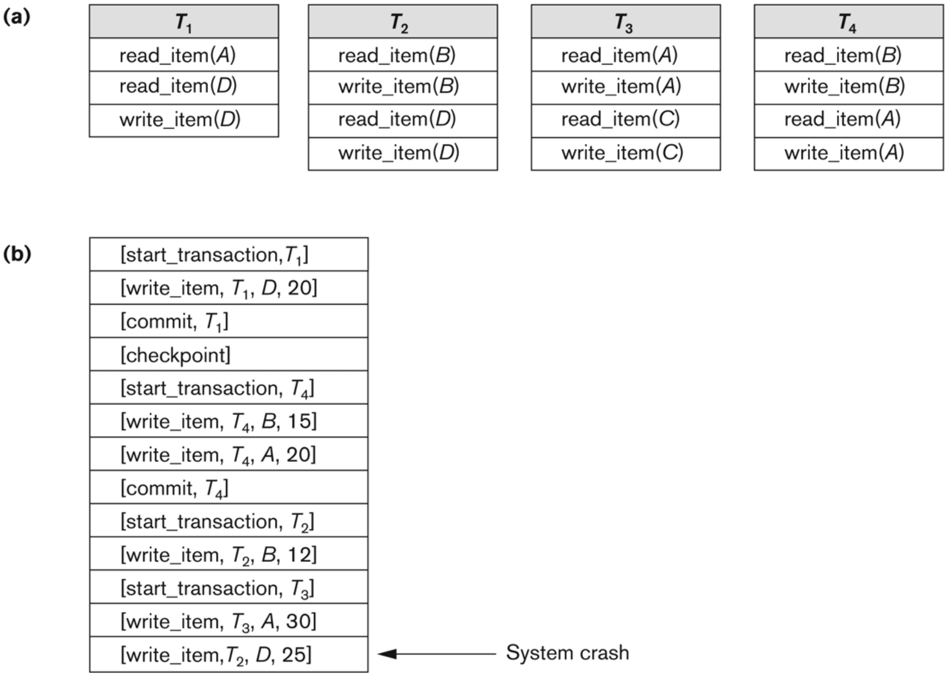
[write_item,T3,A,30]
[write_item,T4,B,15]
[write_item,T2,B,12]



[start_transaction, T_1]
[read_item, T_1, A]
[read_item, T_1, D]
[write_item, T_1, D, 20, 25]
[commit, T_1]
[checkpoint]
[start_transaction, T_2]
[read_item, T_2, B]
[write_item, T_2, B, 12, 18]
[start_transaction, T_4]
[read_item, T_4, D]
[write_item, T_4, D, 25, 15]
[start_transaction, T_3]
[write_item, T_3, C, 30, 40]
[read_item, T_4, A]
[write_item, T_4, A, 30, 20]
[commit, T_4]
[read_item, T_2, D]
[write_item, T_2, D, 15, 25]

System crash

Figure 19.7
An example schedule and its corresponding log.



(a)

T_1	T_2	T_3	T_4
read_item(A)	read_item(B)	read_item(A)	read_item(B)
read_item(D)	write_item(B)	write_item(A)	write_item(B)
write_item(D)	read_item(D)	read_item(C)	read_item(A)
	write_item(D)	write_item(C)	write_item(A)

(b)

[start_transaction,T_1]
[write_item, T_1, D, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[write_item, T_4, A, 20]
[commit, T_4]
[start_transaction, T_2]
[write_item, T_2, B, 12]
[start_transaction, T_3]
[write_item, T_3, A, 30]
[write_item,T_2, D, 25]

T_2 and T_3 are ignored because they did not reach their commit points.

T_4 is redone because its commit point is after the last system checkpoint.

Figure 19.4
An example of recovery using deferred update with concurrent transactions.
(a) The READ and WRITE operations of four transactions. (b) System log at the point of crash.

19.24 Suppose that we use the deferred update protocol for the example in Figure 19.7. Show how the log would be different in the case of deferred update by removing the unnecessary log entries; then describe the recovery process, using your modified log. Assume that only redo operations are applied, and specify which operations in the log are redone and which are ignored.

Answer:

In the case of deferred update, the write operations of uncommitted transactions are not recorded in the database until the transactions commit. Hence, the write operations of T2 and T3 would not have been applied to the database and so T4 would have read the previous (committed) values of items A and B, thus leading to a recoverable schedule.
By using the procedure RDU_M (deferred update with concurrent execution in a multiuser environment), the following result is obtained:
The list of committed transactions T since the last checkpoint contains only transaction T4. The list of active transactions T' contains transactions T2 and T3.
Only the WRITE operations of the committed transactions are to be redone. Hence, REDO is applied to:
[write_item,T4,B,15]
[write_item,T4,A,20]
The transactions that are active and did not commit i.e., transactions T2 and T3 are canceled and must be resubmitted. Their operations do not have to be undone since they were never applied to the database.

19.25 How does checkpointing in ARIES differ from checkpointing as described in Section 19.1.4?

Answer:

The main difference is that with ARIES, main memory buffers that have been modified are not flushed to disk. ARIES, however writes additional information to the LOG in the form of a Transaction Table and a Dirty Page Table when a checkpoint occurs.