1 Short Answer

1. What does a commit record guarantee for that transaction?
2. If a system can never crash, is recovery system necessary?
3. Explain what each of the four log entries does: start, update, commit, and checkpoint.
4. In immediate database modification, what happens when a write command is issued? How does this explain why it is necessary to undo or redo writes when there is a system failure?

1.1 Solution

1. What does a commit record guarantee for that transaction?

That all of the transaction's data has been written to non-volatile storage.

2. If a system can never crash, is recovery system necessary?

It is still necessary. The recovery system does not only recover from system failures, it also rolls back transactions that are aborted due to a number of reasons, like errors in the transaction, or deadlock.

3. Explain what each of the four log entries does: start, update, commit, and checkpoint.

A start log indicates a start of a transaction. An update indicates a transaction changing a value. A commit indicates when updates are flushed to disk. A checkpoint indicates that previous transaction logs were deleted, since the database is in a consistent state.

4. In immediate database modification, what happens when a write command is issued? How does this explain why it is necessary to undo or redo writes when there is a system failure?

The update is written to the disk buffer in memory, which is a part of memory holding data that will be eventually written back to disk, or it is written to disk.

Since the update might not have been written to non-volatile memory, it can be lost if the system fails, so to guarantee atomicity and durability, instructions after a checkpoint must be either undone or redone (depending on whether the transaction has committed yet).

2 Question 1

Consider a database with the following initial values, and the attached command log:

```
A = 41, B = 66, C = -2, D = 104, E = 23

LOG:
< T0, start >
< T1, start >
< T1, B, 66, 102 >
< T2, start >
< T2, C, -2, 99 >
< T1, B, 102, 142 >
```

```
< checkpoint : T0, T1, T2 >
< T0, A, 41, 100 >
< T3, start >
< T2, commit >
< T3, D, 104, -40 >
< T3, commit >
< T4, start >
< T4, E, 23, 24 >
```

Consider the situation where the system crashes before the remaining transactions can commit. Use the recovery protocol for concurrent transactions (which persists all in-memory dirty pages and transaction log entries at each checkpoint) to answer the following questions. Assume immediate database modification.

1. List any transactions that will need to be undone or redone in the recovery process.
2. List, in order, the set of logged operations to be performed to undo or redo the transactions, using the recovery protocol for concurrent transactions. (i.e. "Set A to 7", "Set B to 39", etc.)
3. Give the final values for A, B, C, D, and E after the recovery.

2.1 Solution

1. List any transactions that will need to be undone or redone in the recovery process.

T0, T1 and T4 will need to be undone. T2 and T3 will need to be redone.

2. List, in order, the set of logged operations to be performed to undo or redo the transactions, using the recovery protocol for concurrent transactions. (i.e. "Set A to 7", "Set B to 39", etc.)

```
// UNDO
Set E to 23
Set A to 41
Set B to 102
Set B to 66

// REDO
Set D to -40
```

3. Give the final values for A, B, C, D, and E after the recovery.

A = 41, B = 66, C = 99, D = -40, and E = 23

3 Question 2

Assume the following transactions: T1, T2 and T3:

time	T1	T2	T3
1			R(A)
2	R(A)		
3			W(C)
4		R(B)	
5		R(C)	
6	W(B)		

Assume that the transactions begin at the following timestamps, T1: 2, T2: 4, T3: 1. All transactions will commit at the same time. Will any transactions in this schedule abort if we are using the time-stamp protocol? Which one(s)?

3.1 Solution

Answer: T1 will abort and restart. The reason is that, at time 6, T1 write(B) will not proceed because T1's start time is earlier than B's read timestamp, which is T2's start time at 4.

Feedback

As this is a new course, we appreciate any feedback you have for us! If you enjoyed this assignment, hated this assignment, or have other thoughts to share, please do so here!