CSC 370

Activity Worksheet: Logging

Dr. Sean Chester

Fall 2022

Notes

In this worksheet, you will practice recovering atomicity and consistency in a database using logs. Each question provides a log, a current state on disk, and a logging method, and you should indicate what would be the state on disk after recovery. The first question has been answered already as an exemplar.

Questions

1. Assume that your database loses power and when you inspect the data on disk, you see that A=10, B=25, C=13, and the undo log file reads as given below. To which state should the database be recovered?

   <START T1>
   <T1, C, 0>
   <START T2>
   <T1, A, 13>
   <T2, B, 13>
   <START T3>
   <COMMIT T2>
   <T3, B, 6>

   Solution:

   We read the log backwards, tracking which transactions have completed:

   - We encounter $< T3, B, 6 >$ but we have not seen any commit tags for T3. Therefore, we overwrite B from 25 to 6.
   - We observe a commit tag on T2, so we add it to our set of completed transactions
   - We observe a start tag for T3. No action is required.
   - We observe $< T2, B, 13 >$, but T2 is in our set of completed transactions; so, we ignore this record
   - We observe $< T1, A, 13 >$ but we have not seen any commit tags for T1. Therefore, we overwrite A from 10 to 13.
   - We observe a start tag for T2. No action is required for this, but for convenience we remove it from our set of completed transactions
   - We observe $< T1, C, 0 >$ but we have not seen any commit tags for T1. Therefore, we overwrite C from 6 to 0.
   - We observe a start tag for t1. No action is required.

   Having exhausted the log, we now have the following database state, which corresponds to rolling back transactions T1 and T3:

   - A = 13
   - B = 6
   - C = 0

2. Assume that your database loses power and when you inspect the data on disk, you see that A=10, B=25, C=13, and the redo log file reads as given below. To which state should the database be recovered?

<START T1>
<T1, A, 0>
<T1, B, 5>
<T2, C, 8>
<COMMIT T1>
<CKPT>
<START T2>
<T1, A, 13>
<START T3>
<T2, B, 13>
<COMMIT T2>
<T3, B, 6>

Solution:

3. Assume that your database loses power and when you inspect the data on disk, you see that A=10, B=25, C=13, and the undo log file reads as given below. To which state should the database be recovered?

<START T1>
<START T2>
<T1, A, 5>
<START CKPT(T1,T2)>
<T2, B, 8>
<START T3>
<T3, C, 17>
<COMMIT T3>
<COMMIT T1>

Solution:

4. Assume that your database loses power and when you inspect the data on disk, you see that A=10, B=25, C=13, and the undo/redo log file reads as given below. To which state should the database be recovered?

<START T1>
<START T2>
<T1, A, 5, 10>
<T2, B, 8, 25>
<COMMIT T1>
<START T3>
<T3, C, 17, 13>

Solution:

Solutions

Question 1

We read the log backwards, tracking which transactions have completed:

- We encounter $< T3, B, 6 >$but we have not seen any commit tags for T3. Therefore, we overwrite B from 25 to 6.
- We observe a commit tag on T2, so we add it to our set of completed transactions
- We observe a start tag for T3. No action is required.
- We observe $< T2, B, 13 >$, but T2 is in our set of completed transactions; so, we ignore this record
- We observe $< T1, A, 13 >$ but we have not seen any commit tags for T1. Therefore, we overwrite A from 10 to 13.
- We observe a start tag for T2. No action is required for this, but for convenience we remove it from our set of completed transactions
- We observe $< T1, C, 0 >$ but we have not seen any commit tags for T1. Therefore, we overwrite C from 6 to 0.
- We observe a start tag for t1. No action is required.

Having exhausted the log, we now have the following database state, which corresponds to rolling back transactions T1 and T3:

- A = 13
- B = 6
- C = 0

Question 2

To begin, we scan the log backwards to identify which transactions have committed. We keep scanning until either we hit a checkpoint (the most recent one) or the start of the file. In this case, we identify that transaction T2 has committed and stop the scan at the sixth line, where we encounter a $< CKPT >$. Thereafter, we scan the log forwards:

- We encounter a start tag for T2 but no action is required.
- We observe $< T1, A, 13 >$, but T1 is in not in our set of committed transactions; so, we ignore this record
- We encounter a start tag for T3 but no action is required.
- We observe $< T2, B, 13 >$. Since T2 is in our list of committed transactions, we overwrite B from 25 to 13.
- We observe a commit tag for T2. No action is required for this, but for convenience we remove it from our set of completed transactions
- We observe $< T3, B, 6 >$, but T3 is in not in our set of committed transactions; so, we ignore this record

Having exhausted the log, we now have the following database state, which corresponds to redoing transaction T2 and ignoring the others:

- A = 10
- B = 13
- C = 13

Question 3

We read the log backwards, tracking which transactions have completed:

- We observe a commit tag on T1, so we add it to our set of completed transactions
- We observe a commit tag on T3, so we add it to our set of completed transactions
- We observe $< T3, C, 17 >$, but T3 is in in our set of committed transactions; so, we ignore this record
- We encounter a start tag for T3. No action is required for this, but for convenience we remove it from our set of completed transactions
- We observe $< T2, B, 8 >$. Since T2 is in our list of committed transactions, we overwrite B from 25 to 8.
- We observe a nonquiescent checkpoint on T1 and T2; so, we only need to continue scanning the log until the start point of those two transactions. Moreover, because T1 committed and we are ignoring it, we only need to pay attention to T2 transactions henceforth.
- We observe $< T1, A, 5 >$, but we are ignoring everything except T2
- We observe a start tag for T2; so, we halt the recovery routine.

Having completed the recovery, the database state is restored to:

- A = 10
- B = 8
- C = 13

Question 4

We have an undo/redo log; so, we scan the log backwards, tracking which transactions have completed:

- We observe $< T3, C, 17, 13 >$, but T3 is in in our set of committed transactions; so, we undo the change and set C = 17
- We encounter a start tag for T3. No action is required for this.
- We observe a commit tag for T1, which we add to our list of completed transactions
- We observe $< T2, B, 8, 25 >$. Since T2 is in our list of committed transactions, we overwrite B from 25 to 8.
- We observe $< T1, A, 5, 10 >$. Since T1 is in our list of committed transactions, we ignore this for now
- We observe two start tags and hit the start of the file

Next, we scan the log forwards to replay any committed transactions:

- We observe a start tag for T1. No action is necessary
- We observe a start tag for T2. No action is necessary
- We observe $< T1, A, 5, 10 >$. Since T1 is in our list of committed transactions, we overwrite A from 10 to 10.
- We observe $< T2, B, 8, 25 >$. Since T2 is not in our list of committed transactions, we ignore this record
- We observe a commit tag for T1. Since this was the only remaining committed transaction that we were tracking, we can halt the recovery.

Having completed the recovery, the database state is restored to:

- A = 10
- B = 8
- C = 17