Database System Implementation

Solutions for Chapter 8

Solutions for Section 8.1

Solutions for Section 8.2

Solutions for Section 8.3

Solutions for Section 8.4

Solutions for Section 8.5

Return to Top

Solutions for Section 8.1

Exercise 8.1.1(a)

Let *a* and *b* be the initial values of *A* and *B*, respectively. The values of *A* and *B* at the end of these two steps are *a+b* and *a+2b*, respectively. Since we may assume *0 <= a <= b*, it is easy to show *0 <= a+b <= a+2b*. Thus, consistency is preserved.

Return to Top

Solutions for Section 8.2

Exercise 8.2.2(a)

There are five events of interest, which we shall call:

1. *A*: database element *A* is written to disk.
2. *B*: database element *B* is written to disk.
3. *LA*: the log record for *A* is written to disk.
4. *LB*: the log record for *B* is written to disk.
5. *C*: the commit record is written to disk.

The undo rules imply the following constraints:

1. *C* must be last.
2. *LA* is before *A*.
3. *LB* is before *B*.
4. *LA* is before *LB*.

Note that (4) comes from the fact that we assume log records are written to disk in the order created.

We can conclude that *LA* is first; every other event must be preceded by something. Either *A* or *LB* can be next. In the first case, the only possible order of events is *LA,A,LB,B,C*. In the second case, *A*

and *B* can then be written in either order, giving us the following two sequences: *LA,LB,A,B,C* and *LA,LB,B,A,C*, or a total of 3 possible sequences.

Exercise 8.2.4(b)

U is identified as committed, while *T* is not. Thus, *T* must be undone. Scanning the log backwards, we set *C* to 30 and *A* to 10. Then, we write an `<ABORT>` *T* record on the log.

Exercise 8.2.6

One problem is that if, as in Exercise 8.2.4(b), the crash occurs after *U*'s commit record was preserved on the log, but before *T*'s was, then as in Exercise 8.2.4(b), we set the value of *A* to 10. However, it appears as if *T* never ran, but *U* did. Thus, the value of *A* after crash recovery should be whatever *U* wrote.

As hinted at by the statement of the exercise, the reason things didn't go right can not be blamed directly on the undo logging rules. Rather, it was that *T* and *U* were both writing *A* at about the same time. Even had there been no crash, it is quite possible that the value of *A* after both transactions wrote would not reflect the actions of both.

Exercise 8.2.7(b)

T is the only active transaction, so the log record is `<START CKPT(T)>`. We can write the `<END CKPT>` record after the `<COMMIT T>` record. If the crash occurs after that time, we have to search back only to the `<START CKPT(T)>`. However, for crashes prior to the `<COMMIT T>` record, the search must continue back as far as the `<START T>` record, since that is the (lone) transaction that was active at the start of the checkpoint.

Return to Top

Solutions for Section 8.3

Exercise 8.3.2(a)

With redo logging, the writing of the elements *A* and *B* must occur after all log records are written. Thus, the only option is which element gets written first. Using the notation from Exercise 8.2.2(a), the legal sequences of events are *LA,LB,C,A,B* and *LA,LB,C,B,A*.

Exercise 8.3.3(b)

Since *T* is not complete, we can be sure that none of its changes reached disk, and we can ignore log records about *T*. On the other hand, *U* was committed, and its log record reached disk, so some of its changes *may* have reached disk. To be sure, we must redo all of the actions of *U*. Thus, we first set *B* to 20 and then set *D* to 40.

Return to Top

Solutions for Section 8.4

Exercise 8.4.2(a)

We shall use the notation developed for Exercise 8.2.2(a). The only constraints that undo/redo

logging requires are that *LA* appears before *A*, and *LB* appears before *B*. Of course the log records must be written to disk in the order in which they appear in the log: *LA,LB,C*. The eight orders consistent with these constraints are: *LA,A,LB,B,C*, *LA,A,LB,C,B*, *LA,LB,A,B,C*, *LA,LB,A,C,B*, *LA,LB,B,A,C*, *LA,LB,C,A,B*, *LA,LB,B,C,A*, and *LA,LB,C,B,A*.

To check that we have all possible orders, start with the fixed sequence *LA,LB,C*. The action *B* can be placed either before or after the *C* (two choices). Whichever choice we make for *B*, we can place *A* in any of four positions, from immediately after *LA* to the right end. Thus, there are 2*4 = 8 legal orders.

Exercise 8.4.3(b)

Since *U* is committed, we redo its actions, setting *B* to 21 and *D* to 41. Then, since *T* is uncommitted, we undo its actions from the end moving backwards; we set *C* to 30 and *A* to 10.

Exercise 8.4.5(b)

i. The `END CKPT` record could appear anywhere after the record `<T,A,61,62>`. The reason is that the writing of dirty blocks can be performed independent of whatever actions the transactions are performing in the interrum.

ii. The only active transaction when the checkpoint began was *T*. If the crash occurs after the `END CKPT`, then we need only go back as far as the start of *T*. However, if the crash occurs before the checkpoint ended, then any transaction that was active when the *previous* checkpoint ended may have written some but not all of its data to disk. In this case, the only other transaction that could qualify is *S*, so we must look back to the beginning of *S*, i.e., to the beginning of the log in this simple example.

Return to Top

Solutions for Section 8.5

Exercise 8.5.1(b)

Since the log is a redo log, and the dump completed before *T_1* did, we know that no changes by that transaction can have reached disk. Therefore, no changes by *T_1* could be in the archive. As for any recovery with a redo log, we ignore incomplete transactions, so *T_1* is made to abort and there is no need to perform any changes to the database concerned with *T_1*.

Return to Top