CHAPTER 19

Recovery System

Practice Exercises

19.1 Explain why log records for transactions on the undo-list must be processed in reverse order, whereas redo is performed in a forward direction.

19.2 Explain the purpose of the checkpoint mechanism. How often should checkpoints be performed? How does the frequency of checkpoints affect:

- System performance when no failure occurs?
- The time it takes to recover from a system crash?
- The time it takes to recover from a media (disk) failure?

19.3 Some database systems allow the administrator to choose between two forms of logging: *normal logging*, used to recover from system crashes, and *archival logging*, used to recover from media (disk) failure. When can a log record be deleted, in each of these cases, using the recovery algorithm of Section 19.4?

19.4 Describe how to modify the recovery algorithm of Section 19.4 to implement savepoints and to perform rollback to a savepoint. (Savepoints are described in Section 19.9.3.)

19.5 Suppose the deferred modification technique is used in a database.

a. Is the old value part of an update log record required any more? Why or why not?

b. If old values are not stored in update log records, transaction undo is clearly not feasible. How would the redo phase of recovery have to be modified as a result?

c. Deferred modification can be implemented by keeping updated data items in local memory of transactions and reading data items that have not been updated directly from the database buffer. Suggest how to efficiently implement a data item read, ensuring that a transaction sees its own updates.

d. What problem would arise with the above technique if transactions perform a large number of updates?

19.6 The shadow-paging scheme requires the page table to be copied. Suppose the page table is represented as a B^+-tree.

a. Suggest how to share as many nodes as possible between the new copy and the shadow copy of the B^+-tree, assuming that updates are made only to leaf entries, with no insertions or deletions.

b. Even with the above optimization, logging is much cheaper than a shadow copy scheme, for transactions that perform small updates. Explain why.

19.7 Suppose we (incorrectly) modify the recovery algorithm of Section 19.4 to note log actions taken during transaction rollback. When recovering from a system crash, transactions that were rolled back earlier would then be included in undo-list and rolled back again. Give an example to show how actions taken during the undo phase of recovery could result in an incorrect database state. (Hint: Consider a data item updated by an aborted transaction and then updated by a transaction that commits.)

19.8 Disk space allocated to a file as a result of a transaction should not be released even if the transaction is rolled back. Explain why, and explain how ARIES ensures that such actions are not rolled back.

19.9 Suppose a transaction deletes a record, and the free space generated thus is allocated to a record inserted by another transaction, even before the first transaction commits.

a. What problem can occur if the first transaction needs to be rolled back?

b. Would this problem be an issue if page-level locking is used instead of tuple-level locking?

c. Suggest how to solve this problem while supporting tuple-level locking, by logging post-commit actions in special log records, and executing them after commit. Make sure your scheme ensures that such actions are performed exactly once.

19.10 Explain the reasons why recovery of interactive transactions is more difficult to deal with than is recovery of batch transactions. Is there a simple way to deal with this difficulty? (Hint: Consider an automatic teller machine transaction in which cash is withdrawn.)

19.11 Sometimes a transaction has to be undone after it has committed because it was erroneously executed—for example, because of erroneous input by a bank teller.

a. Give an example to show that using the normal transaction undo mechanism to undo such a transaction could lead to an inconsistent state.

b. One way to handle this situation is to bring the whole database to a state prior to the commit of the erroneous transaction (called *point-in-time* recovery). Transactions that committed later have their effects rolled back with this scheme.

   Suggest a modification to the recovery algorithm of Section 19.4 to implement point-in-time recovery using database dumps.

c. Later nonerroneous transactions can be reexecuted logically, if the updates are available in the form of SQL but cannot be reexecuted using their log records. Why?

19.12 The recovery techniques that we described assume that blocks are written atomically to disk. However, a block may be partially written when power fails, with some sectors written, and others not yet written.

a. What problems can partial block writes cause?

b. Partial block writes can be detected using techniques similar to those used to validate sector reads. Explain how.

c. Explain how RAID 1 can be used to recover from a partially written block, restoring the block to either its old value or to its new value.

19.13 The Oracle database system uses undo log records to provide a snapshot view of the database under snapshot isolation. The snapshot view seen by transaction T_i reflects updates of all transactions that had committed when T_i started and the updates of T_i; updates of all other transactions are not visible to T_i.

Describe a scheme for buffer handling whereby transactions are given a snapshot view of pages in the buffer. Include details of how to use the log to generate the snapshot view. You can assume that operations as well as their undo actions affect only one page.