Chapter 16  Recovery System

- Restart recovery
- Transaction rollback
- Physical undo
- Physical logging
- Transaction rollback
- Checkpoints
- Restart recovery
- Redo phase
- Undo phase
- Repeating history
- Buffer management
- Log-record buffering
- Write-ahead logging (WAL)
- Log force
- Database buffering
- Latches
- Operating system and buffer management
- Fuzzy checkpointing
- Early lock release
- Logical operations
- Logical logging
- Logical undo
- Loss of nonvolatile storage
- Archival dump
- Fuzzy dump
- ARIES
- Log sequence number (LSN)
- PageLSN
- Physiological redo
- Compensation log record (CLR)
- DirtyPageTable
- Checkpoint log record
- Analysis phase
- Redo phase
- Undo phase
- High availability
- Remote backup systems
  - Primary site
  - Remote backup site
  - Secondary site
- Detection of failure
- Transfer of control
- Time to recover
- Hot-spare configuration
- Time to commit
  - One-safe
  - Two-very-safe
  - Two-safe

Practice Exercises

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

16.2 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?
16.3 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 16.4?

16.4 Describe how to modify the recovery algorithm of Section 16.4 to implement savepoints, and to perform rollback to a savepoint. (Savepoints are described in Section 16.8.3.)

16.5 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?

16.6 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 and deletions.

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

16.7 Suppose we (incorrectly) modify the recovery algorithm of Section 16.4 to not 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.)

16.8 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.
Chapter 16  Recovery System

16.9 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.

16.10 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.)

16.11 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 16.4 to implement point-in-time recovery using database dumps.

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

Exercises

16.12 Explain the difference between the three storage types—volatile, non-volatile, and stable—in terms of I/O cost.

16.13 Stable storage cannot be implemented.

a. Explain why it cannot be.

b. Explain how database systems deal with this problem.
16.14 Explain how the database may become inconsistent if some log records pertaining to a block are not output to stable storage before the block is output to disk.

16.15 Outline the drawbacks of the no-steal and force buffer management policies.

16.16 Physiological redo logging can reduce logging overheads significantly, especially with a slotted page record organization. Explain why.

16.17 Explain why logical undo logging is used widely, whereas logical redo logging (other than physiological redo logging) is rarely used.

16.18 Consider the log in Figure 16.5. Suppose there is a crash just before the \(< T_0 \text{ abort}>\) log record is written out. Explain what would happen during recovery.

16.19 Suppose there is a transaction that has been running for a very long time, but has performed very few updates.
   a. What effect would the transaction have on recovery time with the recovery algorithm of Section 16.4, and with the ARIES recovery algorithm.
   b. What effect would the transaction have on deletion of old log records?

16.20 Consider the log in Figure 16.6. Suppose there is a crash during recovery, just after before the operation abort log record is written for operation \(O_1\). Explain what would happen when the system recovers again.

16.21 Compare log-based recovery with the shadow-copy scheme in terms of their overheads, for the case when data is being added to newly allocated disk pages (in other words, there is no old value to be restored in case the transaction aborts).

16.22 In the ARIES recovery algorithm:
   a. If at the beginning of the analysis pass, a page is not in the checkpoint dirty page table, will we need to apply any redo records to it? Why?
   b. What is RecLSN, and how is it used to minimize unnecessary redos?

16.23 Explain the difference between a system crash and a “disaster.”

16.24 For each of the following requirements, identify the best choice of degree of durability in a remote backup system:
   a. Data loss must be avoided but some loss of availability may be tolerated.
   b. Transaction commit must be accomplished quickly, even at the cost of loss of some committed transactions in a disaster.
   c. A high degree of availability and durability is required, but a longer running time for the transaction commit protocol is acceptable.
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.

### Bibliographical Notes

Gray and Reuter [1993] is an excellent textbook source of information about recovery, including interesting implementation and historical details. Bernstein and Goodman [1981] is an early textbook source of information on concurrency control and recovery.

An overview of the recovery scheme of System R is presented by Gray et al. [1981]. Tutorial and survey papers on various recovery techniques for database systems include Gray [1978], Lindsay et al. [1980], and Verhofstad [1978]. The concepts of fuzzy checkpointing and fuzzy dumps are described in Lindsay et al. [1980]. A comprehensive presentation of the principles of recovery is offered by Haerder and Reuter [1983].

The state-of-the-art in recovery methods is best illustrated by the ARIES recovery method, described in Mohan et al. [1992] and Mohan [1990b]. Mohan and Levine [1992] presents ARIES IM, an extension of ARIES to optimize B⁺-tree concurrency control and recovery using logical undo logging. ARIES and its variants are used in several database products, including IBM DB2 and Microsoft SQL Server. Recovery in Oracle is described in Lahiri et al. [2001].


A generalized version of the theory of serializability, with short duration lower-level locks during operations, combined with longer duration higher-level locks, is described by Weikum [1991]. In Section 16.7.3, we saw the requirement that an operation should acquire all lower-level locks that may be needed for the logical undo of the operation. This requirement can be relaxed by performing all physical undo operations first, before performing any logical undo operations. A generalized version of this idea, called multi-level recovery, presented in Weikum et al. [1990], allows multiple levels of logical operations, with level-by-level undo passes during recovery.

Remote backup algorithms for disaster recovery are presented in King et al. [1991] and Polyzois and Garcia-Molina [1994].