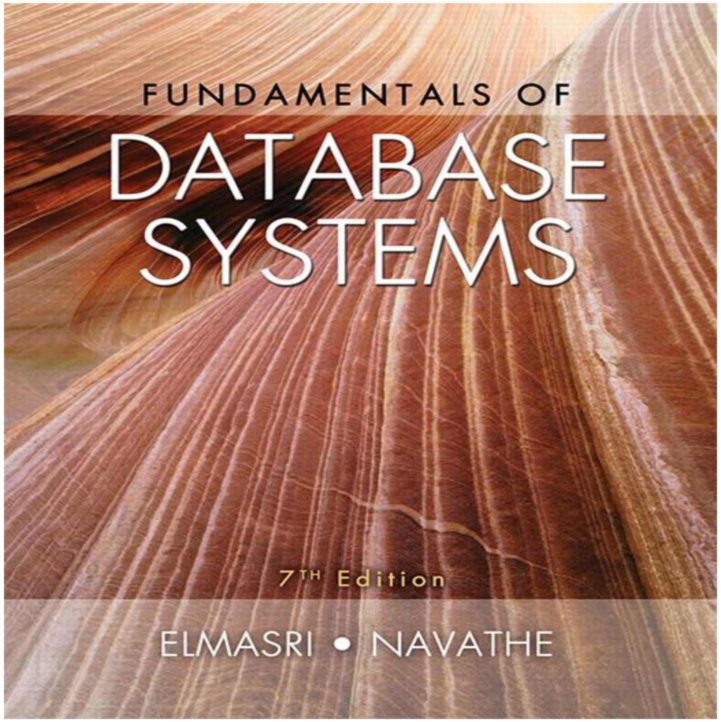
Comp-4150: Advanced and Practical Database Systems

•Ramez Elmasri , Shamkant B. Navathe(2016) Fundamentals of Database Systems (7th Edition), Pearson, isbn 10: 0-13-397077-9; isbn-13:978-0-13-397077-7.

CHAPTER 22

Database Recovery Techniques



CHAPTER 22: Database Recovery Techniques Outline

- 1. Recovery Concepts
- 2. No-Undo/Redo Recovery Based on Deferred Update
- 3. Recovery Techniques Based on Immediate Update
- 4. Shadow Paging
- 5. The Aries Recovery Algorithm
- 6. Recovery in Multidatabase Systems

Chapter 22: Introduction

- Recovery algorithms when there is system failure are discussed after discussing these relevant concepts.
- Recovery concepts
  - 1. Write-ahead logging
  - 2. In-place versus shadow updates
  - 3. Rollback
  - 4. Deferred update
  - 5. Immediate update
- Certain recovery techniques are best used with specific concurrency control methods, but we discuss recovery techniques independent of concurrency control method.

22.1 Recovery Concepts

- Recovery process restores database to most recent consistent state before time of failure
- Information about changes applied to data items by various transactions are kept in the system log and used for recovery.
- Typical recovery strategies
  - 1. Restore backed-up copy of database by re-doing the operations of committed transactions from the back-up log up to the time of failure.
    - This method is best in cases of extensive damage
  - 2. Identify any changes that may cause inconsistency from online system log on disk
    - This method is best in cases of noncatastrophic failure
    - Some operations may require redo but not all.

22.1 Recovery Concepts (cont'd.)

- 3. Deferred update techniques
  - Do not physically update the database on disk until after transaction commits and is recorded in the database say in the main memory buffer.
  - Undo is not needed; redo may be needed
- 4. Immediate update techniques
  - Database may be updated by some operations of a transaction before it reaches commit point
  - These Operations are also recorded in log meaning that if the transaction fails before it commits, the update already done in the database must be rolled back.
  - Recovery (undo and Redo) may still be possible here.

22.1 Recovery Concepts (cont'd.)

- 5. Undo and redo operations required to be idempotent
  - That is, executing recovery operations multiple times is equivalent to executing it just once
  - Entire recovery process should be idempotent
- 6. Caching (buffering) of disk blocks
  - DBMS cache: a collection of in-memory buffers
  - Cache directory keeps track of which database items are in the buffers

22.1 Recovery Concepts (cont'd.)

- Cache buffers replaced (flushed) to make space for new items
- Dirty bit associated with each buffer in the cache
  - Indicates whether the buffer has been modified
- Contents written back to disk before flush if dirty bit equals one
- Pin-unpin bit
  - Page is pinned if it cannot be written back to disk yet

22.1 Recovery Concepts (cont'd.)

- 7. Main strategies for flushing a modified buffer back to disk
  - In-place updating
    - Writes the buffer to the same original disk location
    - Overwrites old values of any changed data items
  - Shadowing
    - Writes an updated buffer at a different disk location, to maintain multiple versions of data items
    - Not typically used in practice
- Before-image: old value of data item
- After-image: new value of data item

22.1 Recovery Concepts (cont'd.)

- 8. Write-ahead logging consists of the following process
  - Ensure the before-image (BFIM) of the data item is recorded in the appropriate log entry
  - And the Appropriate log entry is flushed to disk before the BFIM is overwritten with the AFIM in the database on disk.
  - Necessary for UNDO operation if needed
- UNDO-type log entries include the old value (BFIM) of the data item
- REDO-type log entries include the new value (AFIM) of the data item

22.1 Recovery Concepts (cont'd.)

- Steal/no-steal and force/no-force
  - Specify rules that govern when a page from the database cache can be written to disk
- No-steal approach
  - Cache buffer page updated by a transaction cannot be written to disk before the transaction commits
- Steal approach
  - Recovery protocol allows writing an updated buffer before the transaction commits

22.1 Recovery Concepts (cont'd.)

- Force approach
  - All pages updated by a transaction are immediately written to disk before the transaction commits
  - Otherwise, no-force
- Typical database systems employ a steal/no-force strategy
  - Avoids need for very large buffer space
  - Reduces disk I/O operations for heavily updated pages

22.1 Recovery Concepts (cont'd.)

- 9. Write-ahead logging protocol for recovery algorithm requiring both UNDO and REDO
  - BFIM of an item cannot be overwritten by its after image until all UNDO-type log entries have been force-written to disk
  - Commit operation of a transaction cannot be completed until all REDO-type and UNDO-type log records for that transaction have been force-written to disk

22.1 Checkpoints in the System Log and Fuzzy Checkpointing

- 10. Taking a checkpoint
  - Suspend execution of all transactions temporarily
  - Force-write all main memory buffers that have been modified to disk
  - Write a checkpoint record to the log, and force-write the log to the disk
  - Resume executing transactions
- DBMS recovery manager decides on checkpoint interval

22.1 Checkpoints in the System Log and Fuzzy Checkpointing (cont'd.)

- 11. Fuzzy checkpointing
  - System can resume transaction processing after a begin_checkpoint record is written to the log
  - Previous checkpoint record maintained until end_checkpoint record is written

22.1 Transaction Rollback

- 12. Transaction failure after update but before commit
  - Necessary to roll back the transaction
  - Old data values restored using undo-type log entries
- 13. Cascading rollback
  - If transaction T is rolled back, any transaction S that has read value of item written by T must also be rolled back
  - Almost all recovery mechanisms designed to avoid this

Figure 22.1 Illustrating cascading rollback (a process that never occurs in strict or cascadeless schedules) (a) The read and write operations of three transactions (b) System log at point of crash (c) Operations before the crash

(a)

T_1
read_item(*A*)
read_item(*D*)
write_item(*D*)

T_2
read_item(*B*)
write_item(*B*)
read_item(*D*)
write_item(*D*)

T_3
read_item(*C*)
write_item(*B*)
read_item(*A*)
write_item(*A*)

(b)

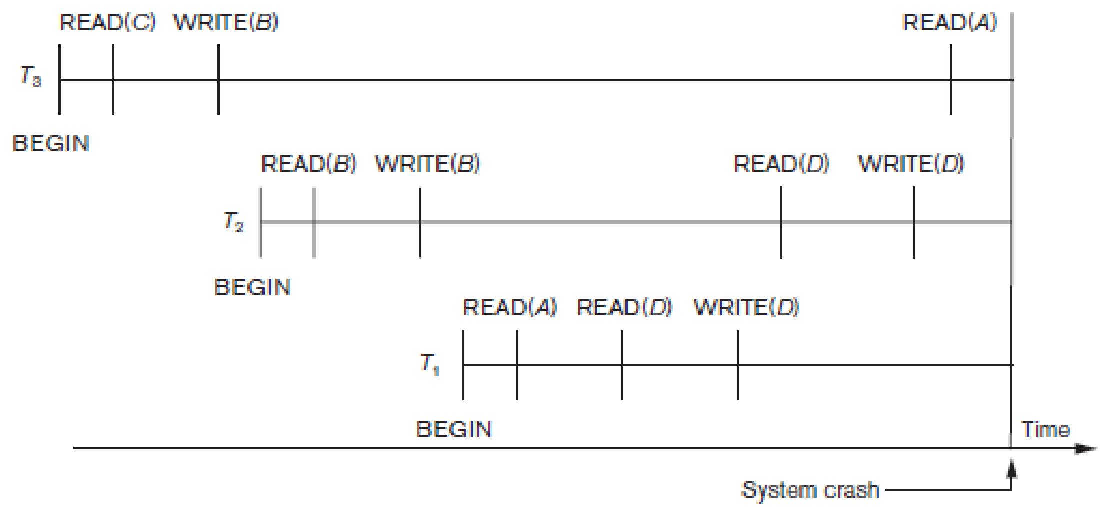
		A	*B*	*C*	*D*
		30	15	40	20
	[start_transaction,T_3]				
	[read_item,T_3,*C*]				
*	[write_item,T_3,*B*,15,12]		12		
	[start_transaction,T_2]				
	[read_item,T_2,*B*]				
**	[write_item,T_2,*B*,12,18]		18		
	[start_transaction,T_1]				
	[read_item,T_1,*A*]				
	[read_item,T_1,*D*]				
	[write_item,T_1,*D*,20,25]				25
	[read_item,T_2,*D*]				
**	[write_item,T_2,*D*,25,26]				26
	[read_item,T_3,*A*]				

← System crash

* T_3 is rolled back because it did not reach its commit point.

** T_2 is rolled back because it reads the value of item *B* written by T_3.

(c)



22.1 Transactions that Do Not Affect the Database

- Example actions: generating and printing messages and reports
- If transaction fails before completion, may not want user to get these reports
  - Reports should be generated only after transaction reaches commit point
- Commands that generate reports issued as batch jobs executed only after transaction reaches commit point
  - Batch jobs canceled if transaction fails

22.2 NO-UNDO/REDO Recovery Based on Deferred Update

- Deferred update concept
  - Postpone updates to the database on disk until the transaction completes successfully and reaches its commit point
  - Redo-type log entries are needed
  - Undo-type log entries not necessary
  - Can only be used for short transactions and transactions that change few items
    - Buffer space an issue with longer transactions

22.2 NO-UNDO/REDO Recovery Based on Deferred Update (cont'd.)

- Deferred update protocol
  - Transaction cannot change the database on disk until it reaches its commit point
    - All buffers changed by the transaction must be pinned until the transaction commits (no-steal policy)
  - Transaction does not reach its commit point until all its REDO-type log entries are recorded in log and log buffer is force-written to disk

22.2 NO-UNDO/REDO Recovery Based on Deferred Update (cont'd.)

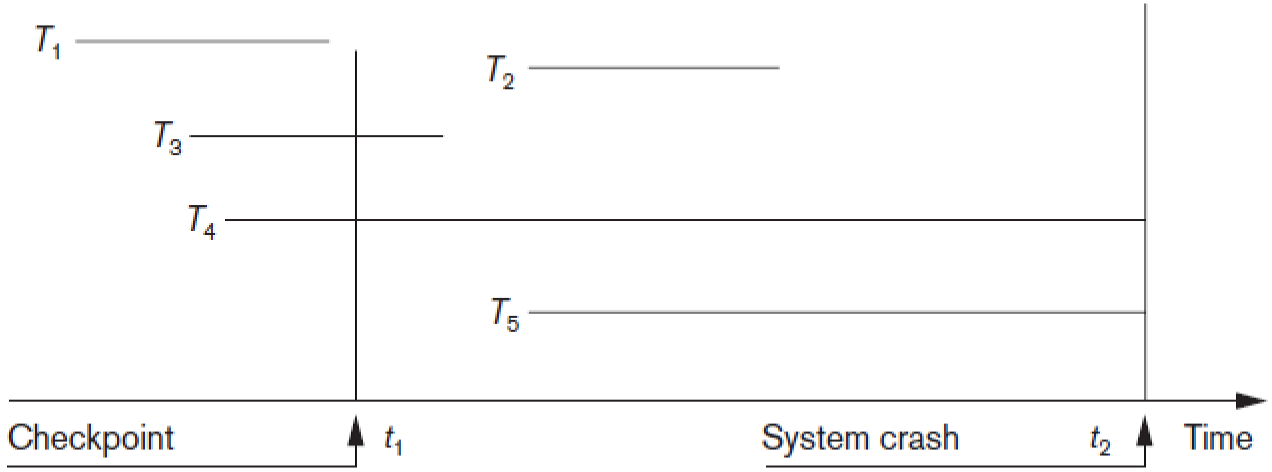


Figure 22.2 An example of a recovery timeline to illustrate the effect of checkpointing

22.3 Recovery Techniques Based on Immediate Update

- Database can be updated immediately
  - No need to wait for transaction to reach commit point
  - Not a requirement that every update be immediate
- UNDO-type log entries must be stored
- Recovery algorithms
  - UNDO/NO-REDO (steal/force strategy)
  - UNDO/REDO (steal/no-force strategy)

T_1
read_item(A)
read_item(D)
write_item(D)

T_2
read_item(B)
write_item(B)
read_item(D)
write_item(D)

T_3
read_item(A)
write_item(A)
read_item(C)
write_item(C)

T_4
read_item(B)
write_item(B)
read_item(A)
write_item(A)

Figure 22.3 An example of recovery using deferred update with concurrent transactions (a) The READ and WRITE operations of four transactions (b) System log at the point of crash

[start_transaction, T_1]
[write_item, T_1, D, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[write_item, T_4, A, 20]
[commit, T_4]
[start_transaction, T_2]
[write_item, T_2, B, 12]
[start_transaction, T_3]
[write_item, T_3, A, 30]
[write_item, T_2, D, 25] ← System crash

T_2 and T_3 are ignored because they did not reach their commit points.

T_4 is redone because its commit point is after the last system checkpoint.

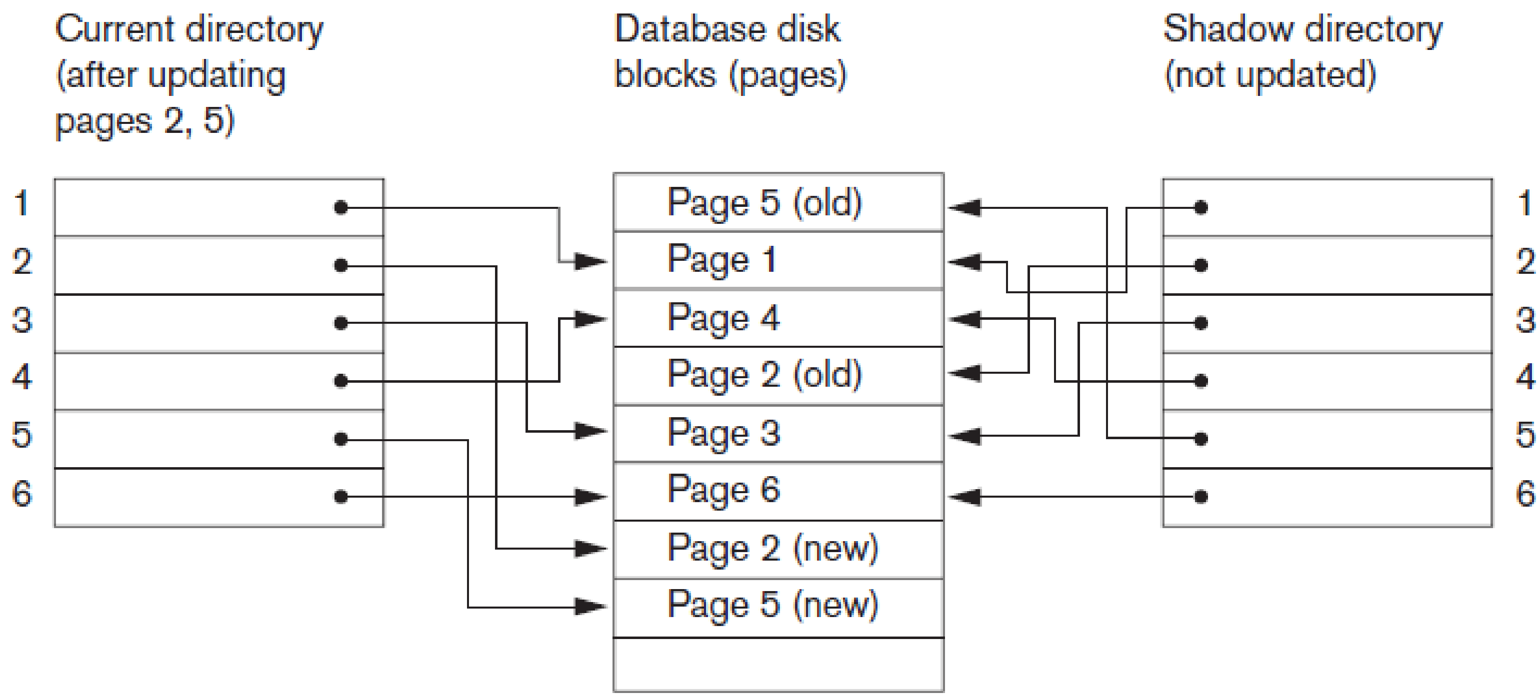
22.4 Shadow Paging

- No log required in a single-user environment
  - Log may be needed in a multiuser environment for the concurrency control method
- Shadow paging considers disk to be made of *n* fixed-size disk pages
  - Directory with *n* entries is constructed
  - When transaction begins executing, directory copied into shadow directory to save while current directory is being used
  - Shadow directory is never modified

22.4 Shadow Paging (cont'd.)

- New copy of the modified page created and stored elsewhere
  - Current directory modified to point to new disk block
  - Shadow directory still points to old disk block
- Failure recovery
  - Discard current directory
  - Free modified database pages
  - NO-UNDO/NO-REDO technique

22.4 Shadow Paging (cont'd.)



[5]The directory is similar to the page table maintained by the operating system for each process.

Figure 22.4 An example of shadow paging

22.5 The ARIES Recovery Algorithm

- Used in many IBM relational database products
- Uses a steal/no-force approach for writing
- Concepts
  - Write-ahead logging
  - Repeating history during redo
    - Retrace all database system actions prior to crash to reconstruct database state when crash occurred
  - Logging changes during undo
    - Prevents ARIES from repeating completed undo operations if failure occurs during recovery

22.5 The ARIES Recovery Algorithm (cont'd.)

- Analysis step
  - Identifies dirty (updated) pages in the buffer and set of transactions active at the time of crash
  - Determines appropriate start point in the log for the REDO operation
- REDO
  - Reapplies updates from the log to the database
  - Only necessary REDO operations are applied

22.5 The ARIES Recovery Algorithm (cont'd.)

- UNDO
  - Log is scanned backward
  - Operations of transactions that were active at the time of the crash are undone in reverse order
- Every log record has associated log sequence number (LSN)
  - Indicates address of log record on disk
  - Corresponds to a specific change of some transaction

22.5 ARIES Recovery Example

Figure 22.5 An example of recovery in ARIES (a) The log at point of crash (b) The Transaction and Dirty Page Tables at time of checkpoint (c) The Transaction and Dirty Page Tables after the analysis phase

(a)

Lsn	Last_lsn	Tran_id	Type	Page_id	Other_information
1	0	T_1	update	*C*	. . .
2	0	T_2	update	*B*	. . .
3	1	T_1	commit		. . .
4	begin checkpoint				
5	end checkpoint				
6	0	T_3	update	*A*	. . .
7	2	T_2	update	*C*	. . .
8	7	T_2	commit		. . .

(b)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	2	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	1
B	2

(c)

TRANSACTION TABLE

Transaction_id	Last_lsn	Status
T_1	3	commit
T_2	8	commit
T_3	6	in progress

DIRTY PAGE TABLE

Page_id	Lsn
C	7
B	2
A	6

22.6 Recovery in Multidatabase Systems

- Two-level recovery mechanism
- Global recovery manager (coordinator) needed to maintain recovery information
- Coordinator follows two-phase commit protocol
  - Phase 1: Prepare for commit message
    - Ready to commit or cannot commit signal returned
  - Phase 2: Issue commit signal
- Either all participating databases commit the effect of the transaction or none of them do

22.6 Recovery in Multidatabase Systems (cont'd.)

- Always possible to recover to a state where either the transaction is committed or it is rolled back
- Failure during phase 1 requires rollback
- Failure during phase 2 means successful transaction can recover and commit

22.7 Database Backup and Recovery from Catastrophic Failures

- Database backup
  - Entire database and log periodically copied onto inexpensive storage medium
  - Latest backup copy can be reloaded from disk in case of catastrophic failure
- Backups often moved to physically separate locations
  - Subterranean storage vaults

22.7 Database Backup and Recovery from Catastrophic Failures (cont'd.)

- Backup system log at more frequent intervals and copy to magnetic tape
  - System log smaller than database
    - Can be backed up more frequently
  - Benefit: users do not lose all transactions since last database backup

22.8 Summary

- Main goal of recovery
  - Ensure atomicity property of a transaction
- Caching
- In-place updating versus shadowing
- Before and after images of data items
- UNDO and REDO operations
- Deferred versus immediate update
- Shadow paging
- Catastrophic failure recovery