RECOVERY

CHAPTER 21,23 (6/E)

CHAPTER 17,19 (5/E)

LECTURE OUTLINE

- Failures
- Recoverable schedules
- Transaction logs
- Recovery procedure

PURPOSE OF DATABASE RECOVERY

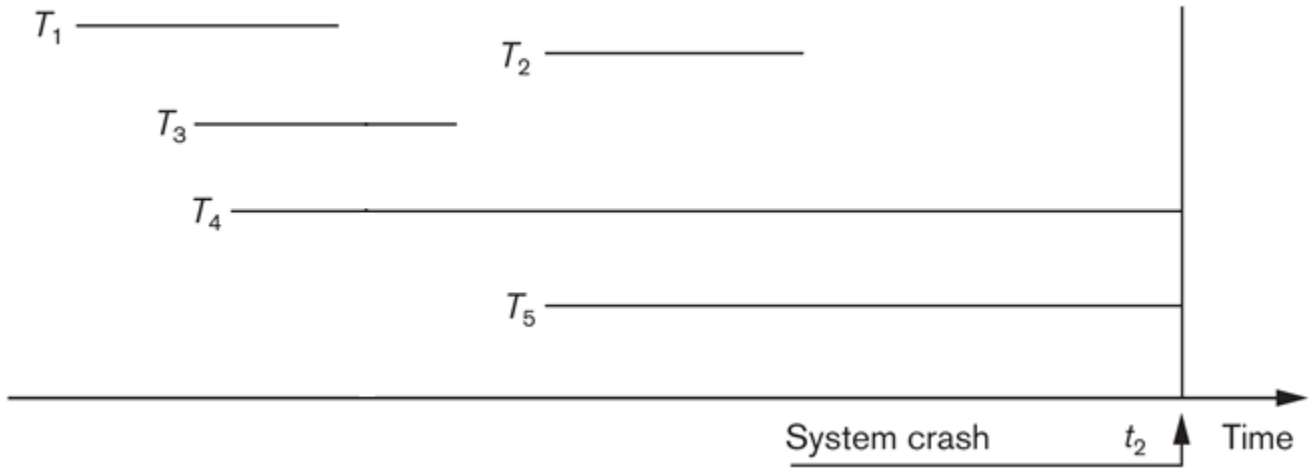
- To bring the database into the most recent consistent state prior to a failure
- Goal: preserve ACID properties

  Atomicity, Consistency, Isolation and *Durability*

  - abort (and restart) transactions active at time of failure
  - ensure changes made by committed transactions are not lost
- Example:
  - If the system crashes before a fund transfer transaction completes its execution, then either one or both accounts may have incorrect value. Thus, the database must be restored to the state before the transaction modified any of the accounts.
- Complication
  - Data items packed into I/O blocks (pages)
  - Updated data first stored in DB cache (at time of write)
  - Actually written to disk (flushed) sometime later

PROBLEM SITUATION

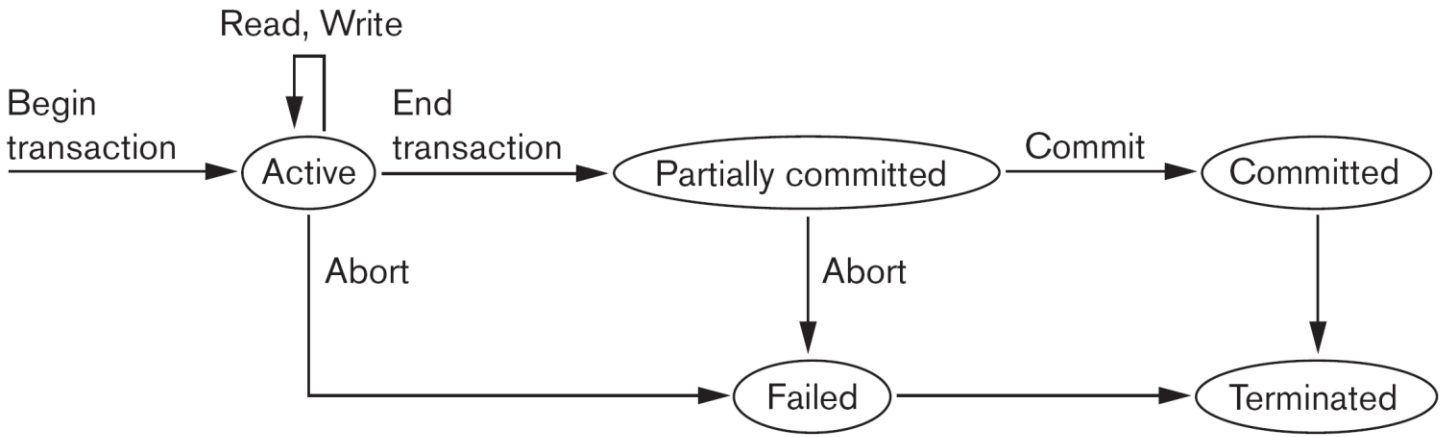
- How can we recover from a system crash?
  - DB files preserved but in-memory data lost
    - Contents of data buffers lost
    - Executing programs' states unknown
  - T1, T2, T3 have committed
  - T4, T5 still in progress
  - Any of the transactions might have written data
  - Some (unknown) subset of the writes have been flushed to disk



TRANSACTION STATES

- Active (executing read, write operations)
- Partially committed (ended but waiting for system checks to determine success or failure)
- Committed (transaction succeeded)
- Failed (transaction failed, must be rolled back)
- Terminated (transaction left system)

Figure 21.4
State transition diagram illustrating the states for transaction execution.



ASSUME RECOVERABLE SCHEDULES

- One where no *committed* transaction needs to be rolled back
  - Required for durability (ACID property)
    - Commit ⟹ not subsequently aborted
  - A schedule S is **recoverable** if no transaction T in S commits until all transactions T' that have written an item that T reads have committed.
- **Cascadeless schedule**: A schedule in which all transactions read only committed values.
  - The set of cascadeless schedules is a subset of the set of recoverable schedules.
- **Strict schedule**: A schedule in which all transactions read only committed values and overwrite only committed values.
  - The set of strict schedules is a subset of the set of cascadeless schedules.
  - Trivially, all serial schedules are strict.
  - Rigorous 2PL guarantees strict schedules.

CAUSES OF FAILURE

- Database may become unavailable for use due to
  - **Transaction failure**
    - Incorrect input, deadlock, incorrect synchronization
    - Result: transaction *abort*
  - **System failure**
    - Addressing error, application error, operating system fault, etc.
  - **Media failure**
    - RAM failure, disk head crash, power disruption, etc.
- We wish to recover from system failure.
  - The database server is halted abruptly.
  - Processing of in-progress SQL command(s) is halted abruptly.
  - Connections to application programs (clients) are broken.
  - Contents of memory buffers are lost.
  - Database files are *not* damaged.
    - Recovery from media failure similar, but may need to restore database files from **backup**

KEEP A SYSTEM LOG FILE

- Append-only file
  - Keep track of all operations of all transactions
  - In the order in which operations occurred
- Stored on disk
  - Persistent except for disk or catastrophic failure
  - Periodically backed up
    - Guard against disk and catastrophic failures
- Main memory buffer
  - Holds records being appended
  - Occasionally whole buffer appended to end of log on disk (flush)

SYSTEM LOG RECORDS

- [**start_transaction**, T]
  - Transaction T has started execution.
- [**write_item**, T, X, old_value, new_value]
  - T has changed the value of item X from old_value to new_value.
- [**commit**, T]
  - T has completed successfully and committed
  - T's effects (writes) must be durable
- [**abort**, T]
  - T has been aborted
  - T's effects (writes) must be ignored and undone

- *Note*: [**read_item**, T, X] not needed if schedules guaranteed to be recoverable (values read must have been committed)

TRANSACTION LOG

- Before Image (old_value) needed to **undo(X)**
  - Reverse effect of a write operation
- After Image (new_value) needed to **redo(X)**
  - Re-apply effect of a write operation

Log Seq #	TID	Prev LSN	Op	Item	Before Image	After Image
1	T1	0	B			
2	T1	1	W	X	100	200
3	T2	0	B			
4	T1	2	W	Y	50	100
5	T3	0	B			
6	T1	4	E			
7	T1	6	C			
8	T2	3	W	Y	100	300

COMMIT POINT OF A TRANSACTION

- Definition: A transaction T reaches its **commit point** when
  1. all its operations that access the database have been executed successfully, and
  2. the effect of all the transaction operations on the database has been recorded in the log file (on disk).

  The transaction is then said to be **committed**.
- ***Write-ahead log***
  - Before a transaction reaches its commit point, any main memory buffers of the log that have not been written to disk yet must be copied to disk.
    - **Force write** the log buffer (flush log)

RECOVERY PROCESS

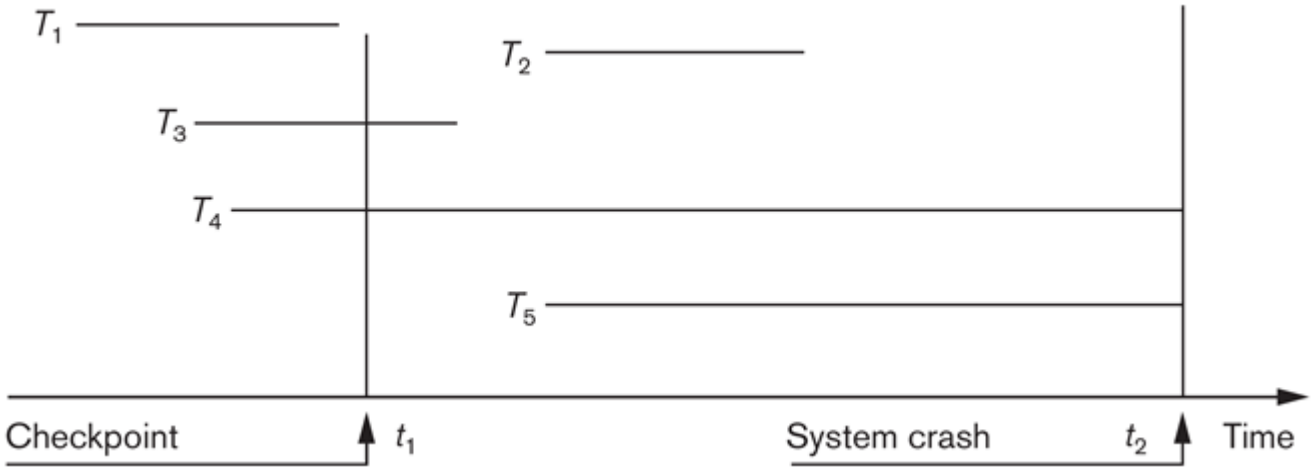
- Assumes *strict* schedules
  - Transactions read or overwrite *committed* values only

1. Roll-forward (redo)
   - Scan log from the start, re-executing all updates
     - Use *after image* for new values
       - Note: re-executing an operation is OK (idempotent)
2. Roll-back (undo)
   - For any uncommitted transaction:
     - Follow chain backwards and for each update:
       1. Restore *before image*
       2. Append [undo] record to log (in case of crash *during* recovery)
     - *Note*: only need to undo [write] for which ∄ corresponding [undo]
3. Restart executing all in-progress transactions

   (those neither committed nor aborted)

CHECKPOINTING

- To save redo effort, use **checkpoints**
  - Occasionally flush data buffers
    1. Suspend execution of transactions temporarily.
    2. Force-write modified (dirty) buffer data to disk.
    3. Append [checkpoint] record to log.
    4. Flush log to disk.
    5. Resume normal transaction execution.
  - During recovery, redo required only for log records appearing after [checkpoint] record



LECTURE SUMMARY

- Databases Recovery
  - Types of Failure
  - Transaction Log
  - Transaction Roll-back (Undo) and Roll-Forward (Redo)
  - Checkpointing