Recovery Techniques

Database Recovery Management

- **Database recovery**: Restores database from a given state to a previously consistent state
- Recovery transactions are based on the atomic transaction property
  - **Atomic transaction property**: All portions of a transaction must be treated as a single logical unit of work
    - If transaction operation cannot be completed:
      - Transaction must be aborted
      - Changes to database must be rolled back

Recovery Concepts

- Recovery process restores database to most recent consistent state before time of failure
- Information kept in system log
- Typical recovery strategies
  - Restore backed-up copy of database
    - Best in cases of extensive damage
  - Identify any changes that may cause inconsistency
    - Best in cases of noncatastrophic failure
    - Some operations may require redo

Techniques Used in Transaction Recovery Procedures

Deferred-write technique or **deferred update**

- Only transaction log is updated
- Undo is not needed; redo may be needed

Write-through technique or **immediate update**

- Database is immediately updated by transaction operations during transaction's execution

Recovery Concepts (cont'd.)

- Undo and redo operations required to be idempotent
  - Executing operations multiple times equivalent to executing just once
  - Entire recovery process should be idempotent
- 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

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

Recovery Concepts (cont'd.)

- Main strategies
  - 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

Recovery Concepts (cont'd.)

- Write-ahead logging
  - Ensure the before-image (BFIM) is recorded
  - Appropriate log entry flushed to disk
  - Necessary for UNDO operation if needed
- UNDO-type log entries
- REDO-type log entries

Recovery Concepts (cont'd.)

Steal/no-steal and force/no-force

- 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

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
  - Advantage of "steal" is that it avoids need for very large buffer space to store all updated pages in memory
  - Advantage of "no-force" is that an updated page of a committed transaction may still be in the buffer when another transaction needs to update it, thus eliminating the I/O cost to write that page multiple times to disk and possible having to read it again from disk

Recovery Concepts (cont'd.)

- Write-Ahead Logging (WAL) 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

Checkpoints in the System Log and Fuzzy Checkpointing

- 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
  - It may be measured in time – say, every ‘m’ minutes – or in the number ‘t’ of committed transactions since the last checkpoint

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

- 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

Transaction Rollback

- Transaction failure after update but before commit
  - Necessary to roll back the transaction
  - Old data values restored using undo-type log entries
- Cascading rollback
  - If transaction T is rolled back, any transaction S that has read value of item written by T, similarly any transaction R that has read value of item written by S, and so on…, must also be rolled back
  - possible only when recovery protocol ensures *recoverable* schedules but does not ensure *strict* or *cascade less* schedule
  - Almost all recovery mechanisms designed to avoid this

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)