Recoverability

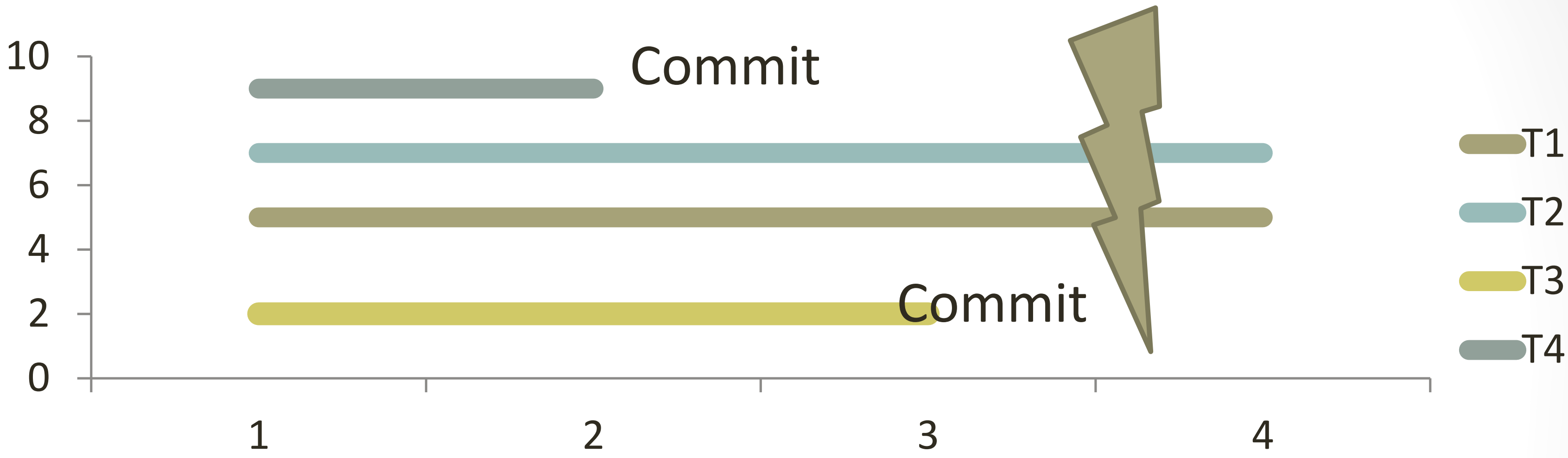
Kathleen Durant PhD

CS3200

Recovery Manager

- Recovery manager ensures the ACID principles of atomicity and durability
  - Atomicity: either all actions in a transaction are done or none are done
  - Durability: if a transaction is committed, changes persist within the database
- Desired behavior
  - keep actions of committed transactions
  - discard actions of uncommitted transactions

Keep the committed transactions



Throw away the active transactions work

- T3 and T4 actions should appear in the database
- T1 and T2 actions should not appear in the database

Database Recovery

Process of restoring database to a correct state in the event of a failure.

- **Need for Recovery Control**
  - **Two types of storage: volatile (main memory) and nonvolatile.**
  - **Volatile storage does not survive system crashes.**
  - **Stable storage represents information that has been replicated in several nonvolatile storage media with independent failure modes.**

Types of Failures

- System crashes, resulting in loss of main memory.
- Media failures, resulting in loss of parts of secondary storage.
- Application software errors.
- Natural physical disasters.
- Carelessness or unintentional destruction of data or facilities.
- Sabotage.

Transactions and Recovery

- **Transactions represent basic unit of recovery.**
- **Recovery manager responsible for atomicity and durability.**
- **If failure occurs between commit and database buffers being flushed to secondary storage then, to ensure durability, recovery manager has to *redo* (*rollforward*) transaction's updates.**

Transactions and Recovery

- **If transaction had not committed at failure time, recovery manager has to *undo* (*rollback*) any effects of that transaction for atomicity.**
- **Partial undo - only one transaction has to be undone.**
- **Global undo - all transactions have to be undone.**

Buffer pool management

- FORCE – every write to disk?
  - Poor performance (many writes clustered on same page)
  - At least this guarantees the persistence of the data
- STEAL – allow dirty pages to be written to disk?
  - If so, reading data from uncommitted transactions violates atomicity
  - If not, poor performance

	Force - every write to disk	No Force – write when optimal
Steal – use internal DB buffer for read		Desired but complicated
No Steal - always read only committed data	Easy but slow	

Complications from NO FORCE and STEAL

- NO FORCE
  - What if the system crashes before a modified page can be written to disk?
  - Write as little as possible to a convenient place at commit time to support **REDO**ing the data update
- STEAL
  - Current updated data can be flushed to disk but still locked by a transaction T1
    - What if T1 aborts?
    - Need to **UNDO** the data update done by T1

Recovery Facilities

- DBMS should provide following facilities to assist with recovery:
  - Backup mechanism
    - Makes periodic backup copies of database.
  - Logging facilities
    - Keeps track of current state of transactions and database changes.
  - Checkpoint facility
    - Enables updates to database in progress to be made permanent.
  - Recovery manager
    - Allows DBMS to restore database to consistent state following a failure.

Log File

- **Collection of records that represent the history of actions executed by the DBMS**
- **Contains information about all updates to database:**
  - **Transaction records.**
  - **Checkpoint records.**
- **Often used for other purposes (for example, auditing).**

Log File Data

- **Transaction records contain:**
  - **Transaction identifier.**
  - **Type of log record, (transaction start, insert, update, delete, abort, commit).**
  - **Identifier of data item affected by database action (insert, delete, and update operations).**
  - **Before-image of data item.**
  - **After-image of data item.**
  - **Log management information (Transaction operation links)**

Write-ahead Logging

- The Write-Ahead Logging Protocol:
  1. Must force the log record to permanent storage *before* the corresponding data page gets written to disk.
  2. Must write all log records for a transaction *before commit*.
  - #1 guarantees Atomicity.
  - #2 guarantees Durability.

Sample Log File

Tid	Time	Operation	Object	Before image	After image	pPtr	nPtr
T1	10:12	START				0	2
T1	10:13	UPDATE	STAFF SL21	(old value)	(new value)	1	8
T2	10:14	START				0	4
T2	10:16	INSERT	STAFF SG37		(new value)	3	5
T2	10:17	DELETE	STAFF SA9	(old value)		4	6
T2	10:17	UPDATE	PROPERTY PG16	(old value)	(new value)	5	9
T3	10:18	START				0	11
T1	10:18	COMMIT				2	0
	10:19	CHECKPOINT	T2, T3				
T2	10:19	COMMIT				6	0
T3	10:20	INSERT	PROPERTY PG4		(new value)	7	12
T3	10:21	COMMIT				11	0

14

Log File

- **Log file may be duplexed or triplexed.**
- **Log file sometimes split into two separate random-access files.**
- **Potential bottleneck; critical in determining overall performance.**

15

Checkpointing

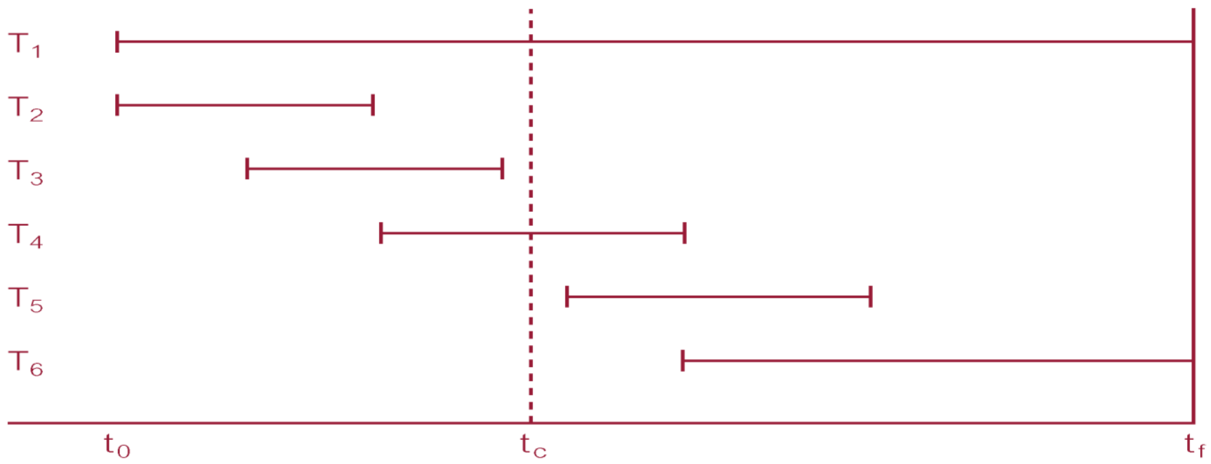
<u>Checkpoint</u>

Point of synchronization between database and log file. All buffers are force-written to secondary storage.

- Checkpoint record is created containing identifiers of all active transactions.
- When failure occurs:
  - Redo all transactions that committed since the checkpoint and
  - Undo all transactions active at time of crash.

16

Checkpoint Example



- **DBMS starts at time t_0, but fails at time t_f. Assume data for transactions T_2 and T_3 have been written to secondary storage.**
- **T_1 and T_6 have to be undone. In absence of any other information, recovery manager has to redo T_2, T_3, T_4, and T_5.**

17

Checkpointing

- **In previous example, with checkpoint at time t_c, changes made by T_2 and T_3 have been written to secondary storage.**
- **Thus:**
  - **only redo T_4 and T_5,**
  - **undo transactions T_1 and T_6.**

Recovery Techniques

- **If database has been damaged:**
  - **Need to restore last backup copy of database and reapply updates of committed transactions using log file.**
- **If database is only inconsistent:**
  - **Need to undo changes that caused inconsistency. May also need to redo some transactions to ensure updates reach secondary storage.**
  - **Do not need backup version of the database, but can restore database using before- and after-images in the log file.**

Main Recovery Techniques

- Three main recovery techniques:
  - Deferred Update
  - Immediate Update
  - Shadow Paging

20

Deferred Update

- Updates are not written to the database until after a transaction has reached its commit point.
- If transaction fails before commit, it will not have modified database and so no undoing of changes required.
- May be necessary to redo updates of committed transactions as their effect may not have reached database.

Immediate Update

- **Updates are applied to database as they occur.**
- **Need to redo updates of committed transactions following a failure.**
- **May need to undo effects of transactions that had not committed at time of failure.**
- **Essential that log records are written before write to database. *Write-ahead log protocol*.**

Immediate Update

- **If no "*transaction commit*" record in log, then that transaction was active at failure and must be undone.**
- **Undo operations are performed *in reverse order in which they were written to log*.**

Shadow Paging

- Maintain two page tables during life of a transaction: *current* page and *shadow* page table.
- When transaction starts, two pages are the same.
- Shadow page table is never changed thereafter and is used to restore database in event of failure.
- During transaction, current page table records all updates to database.
- When transaction completes, current page table becomes shadow page table.

Summary

- Recovery Manager guarantees Atomicity and Durability.
- Different recovery techniques available
- The recovery of a database is dependent on the type of failure the database encountered
- If the current version of the database is not recoverable use the log and a backup version of the database to get the database to a consistent state
- If the current version of the database is recoverable and in an inconsistent state then use the Log with the current version of the database to recover from the failure.
- Checkpointing: A quick way to limit the amount of log to scan on recovery