WEEK 10

TRANSACTION – RECOVERY MANAGEMENT

STUDENT OBJECTIVES

- Upon completion of this video, you should be able to:
  - Explain the difference between full backups and differential backups
  - List at least 3 types of failures that could cause us to require to perform a backup
  - Look at a checkpoint diagram to determine what should happen to the various transactions
  - Show what happens with deferred updates (no undo/redo)
  - Show what happens with immediate updates (undo/redo)

RECOVERY MANAGEMENT

- Recovery restores a database from an inconsistent state to a consistent state → the MOST CURRENT consistent state
- Each transaction is treated as an atomic unit and is rolled back using the log (recovery undoes all transactions that could not be completed) or redone (recommitted, rolled forward) using the log.

BACKUPS

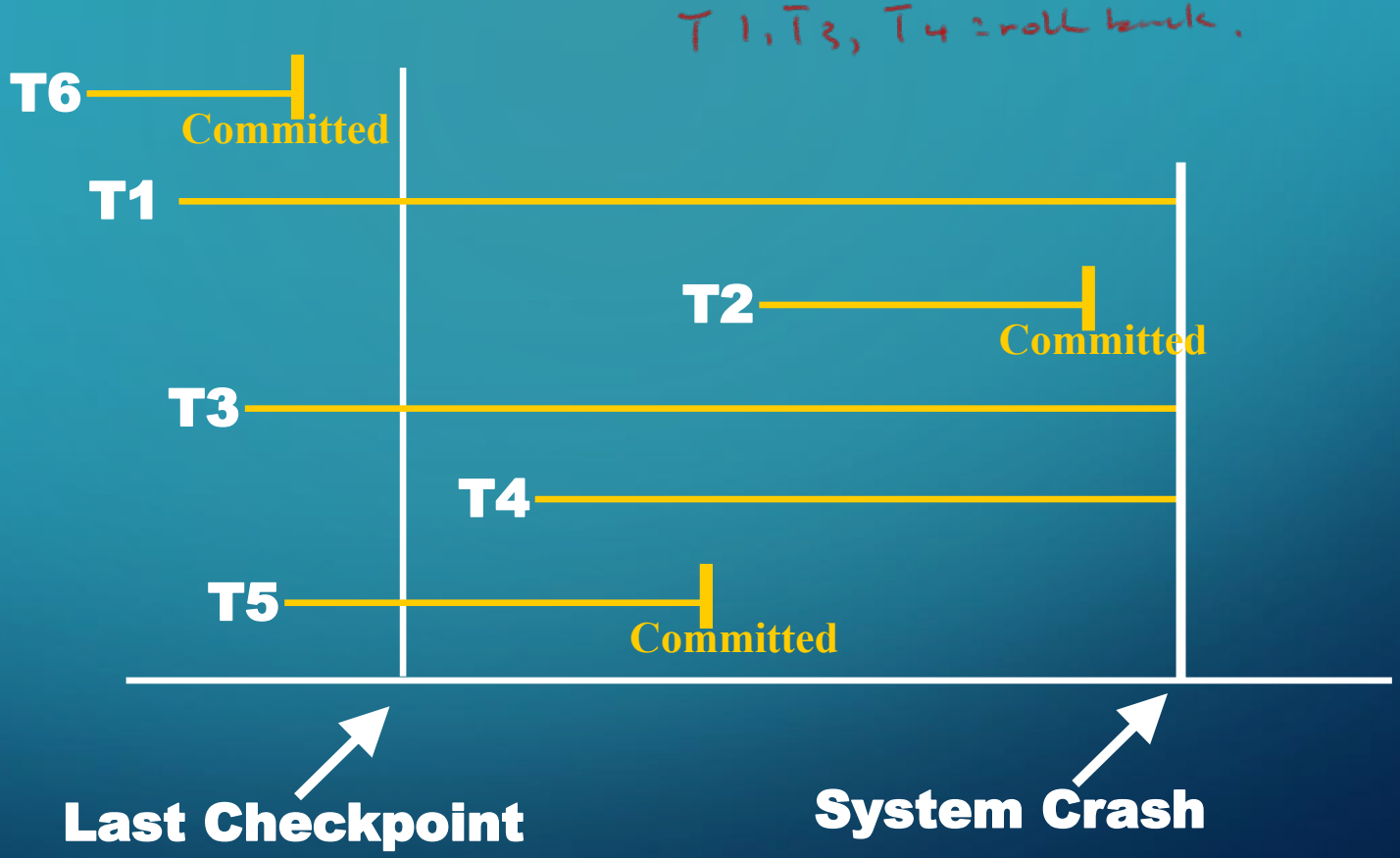
- Need Backups: (store backups in safe place)
  - Full Backup (needed for catastrophic failures, old backup is used and then the log is used to bring it back to its current state)
    - Maybe you do this once a day – too expensive to do this every minute
  - Differential Backup (only last changes are recorded, you can recreate database using original full backup and differential backups)
  - Backup of transaction log only – every single insert, update, delete are recorded.
    - As soon as the database finishes writing the redo log, it performs a redo log switch and moves it to a safe location so it becomes a backup redo log and we start a new redo log.
    - Eg For Oracle you need to start the database in ARCHIVELOG mode to make sure it creates the log files.

- Failures caused by:
  - Human Errors
    - Oops – someone accidently dropped a critical table
    - Programming Errors
  - Hardware (disk crash, …)
  - Transactions (deadlock so system aborts),
  - External (fire, flood, …)
- Depending on the severity might just backup to before uncommitted transactions or might use a backup from the night before?

TRANSACTION RECOVERY

- **Write ahead log protocol:** Log file is always written to BEFORE the database is actually changed so that we can always recover (I.e. move back to a consistent state) what was done using the log file.
- **Checkpoint in Logs:** periodically records which transactions have committed and which haven't at a given time →WRITTEN TO THE LOG. Actions consist of:
  - 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 disk
  - Resume executing transactions

CHECKPOINT:


T1, T3, T4 : roll back.
T6
Committed
T1
T2
Committed
T3
T4
T5
Committed
Last Checkpoint
System Crash

DEFERRED UPDATE

- Database not actually changed until after the commit, only the transaction log is updated.
- Changes are stored in a buffer and written to disk when the commit occurs.
- During commit, the log file is written FIRST, then the actual changes.
- If a transaction is aborted we do NOT have to UNDO anything because no changes occurred, we may have to REDO committed transactions in the log though.
- This a **NO-UNDO/REDO**

DEFERRED UPDATE STEPS:

- Find last checkpoint in transaction log (last time data was physically changed on disk)
- For transactions that started and committed before checkpoint, do nothing since that data is already saved to disk
- For committed transactions AFTER last checkpoint, use the transaction log to redo transaction and update database (use the NEW values) (REDO)
- For transactions that were rolled back after last checkpoint or never got to commit, we don't need to do anything as the database was never updated. (NO UNDO)

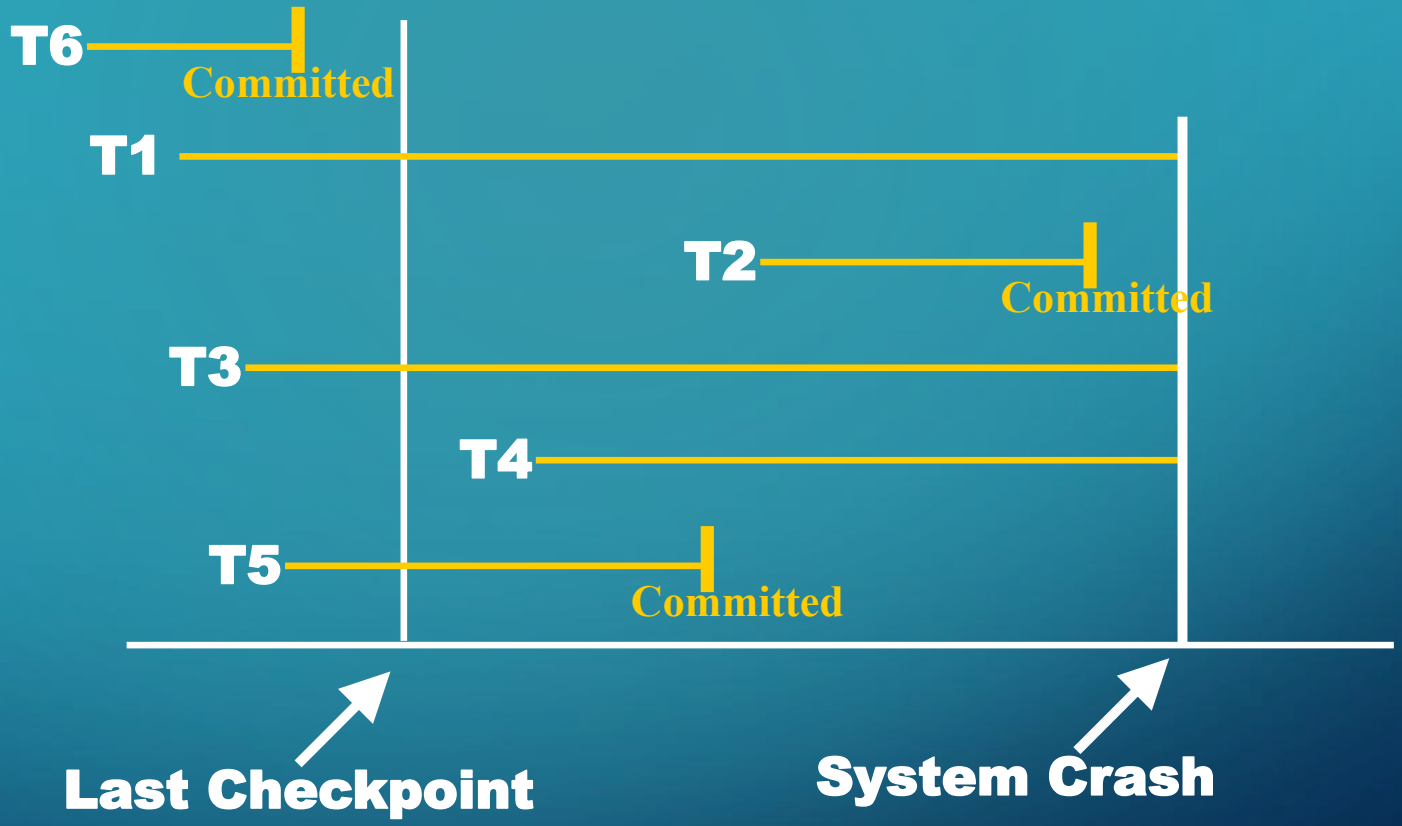
Immediate Update

- Changes are written to the database as they occur before the commit.
- The changes are first written to the log and then to the database.
- If a transaction aborts before the commit point, the previous operations done in the database must be rolled back and some may have to be redone
- This is **UNDO/REDO.**

IMMEDIATE UPDATE STEPS:

- Find last checkpoint in transaction log (last time data was physically changed on disk)
- For transactions that started and committed before checkpoint, do nothing since that data is already saved to disk
- For committed transactions AFTER last checkpoint, use the transaction log to redo transaction using NEW values and update database (REDO)
- For transactions that were rolled back after last checkpoint or never got to commit, the transaction log is used to find the OLD values to undo the operations (from newest to oldest) (UNDO)

CONSIDER THIS DIAGRAM AGAIN:



QUESTION: A REDO log would have to keep track of the New value of the data, whereas an UNDO log would have to keep track of the old value of the data (OLD, NEW).

Commited: REDO
in transaction: UNDO / NO-UNDO

- **For Deferred Update**:
  - T2 and T5 need a REDO (use **new** values in log file)
  - T6 was already written to database before checkpoint
  - T1, T3, T4 were never written to the database so NO UNDO

- **For Immediate Update**:
  - T2 and T5 need a REDO (use **new** values in log file)
  - T6 was already written to database before checkpoint
  - T1, T3, T4 were written to the database so the **old** values must be restored so UNDO