Date _____ Day _____

"RECOVERY TECHNIQUE"

- Deffersed Update
  - ↳ Donot physically updates the DB until the transaction comitts.
  - ↳ DB mein kuch update hi nhi hoga to jab failure hoga to koi UNDO oper perform nhi hoga.
  - ↳ Lekin agr comitt sucessful hogaya to scari changes jo log file mein koi thi we DB par REDO krni parengi.
  - ↳ Also called NO-UNDO/REDO algo.

- Immediate Update:
  - ↳ The DB may be update by some operations of a transaction before the trans reaches comitt.
  - ↳ But the opesations also recorded in log file.
  - ↳ both UNDO & redo oper may require for recovery.
  - ↳ Also called UNDO/REDO Algo.

A variation of the algo where all changes are requ to be recorded in DB before comitt
  - ↳ In this case only UNDO is required.
  - ↳ Also called UNDO/NO-REDO Algo.

Date _____ Day _____

→ The UNDO & REDO are required to be idempotent.
↳ mtlab jitni baar bhi kare to same result aaye.

- Caching (Buffering) of Disk Blocks :
  ↳ DBMS cache : a collection of in-memory buffers
  ↳ phele buffer mein update hoga phir DB mein
  ↳ Cache directory keeps track of which DB items are in the buffers.
  ↳ When a request is made to do an action on an item first it is checked in cache directory if it is not there then the appropiate disk pages are copied into the cache.
  ↳ Some pages may need to be replace / flush to make space.

Dirty Bit : Associated with each buffer in the cache
↳ Indicates which buffer is modified
↳ Contents of the buffer needs to be written to disk if dirty bit = 1, before flush.

Pin-unpin Bit : A page in the cache is pinned (bit value 1) if it cannot be written back to disk as yet.

Day

Date

- Two main strategies can be employed when flushing a modified buffer back to disk.
  - ↳ in-place Updating:
    - ↳ Writes the buffer to the same to original location on disk
    - ↳ Overwrite the old value
    - ↳ So only a single copy is maintained.
  - ↳ Shadowing:
    - ↳ Writes the modified buffer to diff location on disk, to maintain multiple copies.
    - ↳ Not typically used.
    - Before-image (BFIM) old value of data item
    - After-image (AFIM) new value of data item

- Write-Ahead Logging:
  - ↳ Ensure that BFIM is recorded
  - ↳ Appropriate log entry flush to disk
  - ↳ Necessary for UNDO operation if needed.

UNDO-TYPE log entries: include the old value (BFIM) of the items ∴ it needs to be written back during UNDO.

REDO-TYPE log entries: includes the new value (AFIM) of the item which will be req during redo.

Date _____ Day _____

→ Rules that govern when a page from DB cache can be written to disk

- STEAL / NO-STEAL Approach:
  - ↳ No Steal Appr.: (UNDO will never be req)
    - ↳ Cache buffer page updated by a trans cannot be written to disk before the trans comitt. (Pin bit = 1)
      - The desired update follows this approach
  - ↳ Steal Appr.:
    - ↳ Recovery protocol allows writing an updated buffer before the trans comitts.
      - on to disk
- FORCE / NO-FORCE Approach: (REDO not req in recovery)
  - ↳ All pages updated by a trans are immediately written to disk before the trans comitt.
    - Otherwise, no force.

→ Usually steal/no-force approach is used
- ↓ as it saves buffer
- ↳ saves disk I/O

Day

Write-Ahead Logging (WAL) Protocol:

(1) The BFIM cannot be overwritten by its AFIM in DB until all UNDO-type log entries for the updating transaction - up to this point - have been written to the disk.

(2) The commit oper of a trans cannot be completed until all the REDO & UNDO type log records for that trans have been force written to the disk.

→ DBMS keeps track of the list of all active, committed and aborted transaction since the last checkpoint.

CHECKPOINT:

Taking a checkpoint consist of following actions

(1) Suspend exec of trans temporarily
(2) Force-write all main memory buffers that have been modified to disk
(3) Write a [checkpoint] record to the log, and force-write the log to disk
(4) Resume exe trans.

→

Date _____ Day _____

- **Fuzzy Checkpointing:**

→ In this technique the ongoing transaction need not to be stopped.

→ The system can resume trans processing after a [begin_checkpoint] record is written to the log without having to wait for all buffers to be written to disk.

→ When all record is written on [end_checkpoint, ...] record is written.

→ And the pointer to the checkpoint now points to the new checkpoint.

See page # 820 (Book).