Normalization: (also known as Schema Refinement)

→ Organizing data into multiple related tables

→ Minimize Data Redundancy (Not elimination)

→ eliminates Anamolies like Insertion, Deletion, Updation.

1NF

→ each cell must have single value

→ Column should have values of same data type

→ Each column should have unique name.

→ Order of data doesn't matter.

2NF

→ must be in 1NF

→ NO partial dependency

→ A column depending only on one field

→ Can be eliminated by moving the field to another table.

3NF

→ must be in 2NF

→ NO Transitive dependency

A → B → C

Transitive

→ Create New table.

Decompositions: Decomposition is done when table is not in appropriate normal form.

→ Breaking into multiple tables.

→ If no proper decomposition is done, it leads to loss of info.

→ used to eliminate problems like bad design - anomalies, inconsistencies & redundancy.

Types:

1. **Lossless decomposition:**
   → After decomposition, information is not lost.
   → guarantees that join of relation results in same relation before decomposition.

2. **Lossy:** if joined after decomposition, then data may be lost.

Properties:

1. **Dependency preserving:**
   → functional dependencies (columns) b/w attributes in a relation (table) are preserved when transforming the table from one NF to another.
2. Decomposition must be lossless.
3. Lack of Data redundancy.

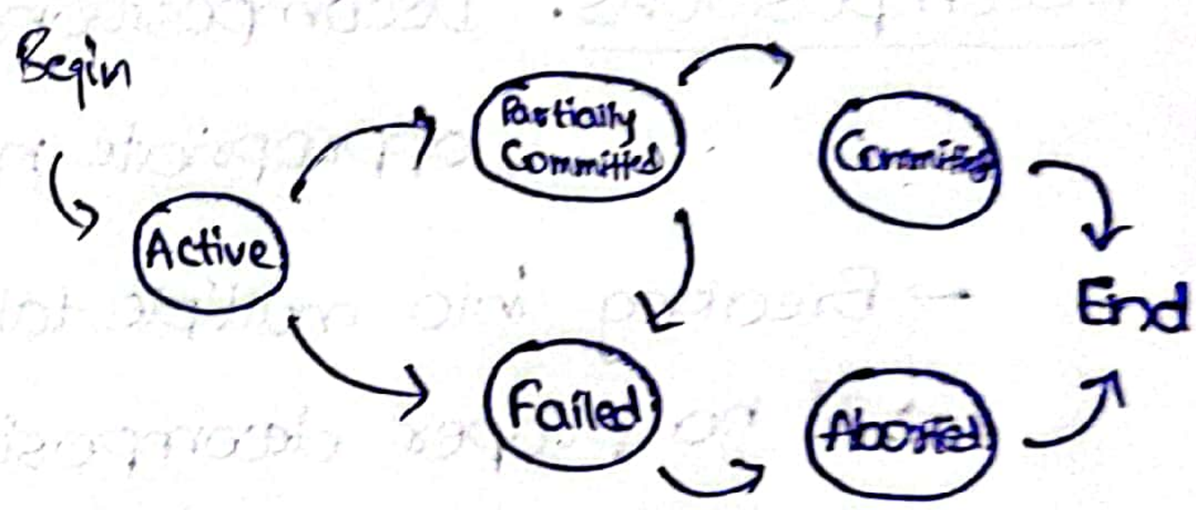
Candidate key: It is the minimal set of attributes whose attribute closure is set of all attributes of relation is called the candidate keys of relation.

Primary key	foreign key
→ identity each record in DB	→ links two tables together. refers to Primary key of 2nd table.
→ NO NULL values	→ Can have NULL values
→ Only one P.K.	→ >1 can be acceptable.
→ Indexed automatically	→ Indexed Manually (can)
→ Cannot be deleted	→ Can be deleted
→ Cannot Create parent-child relationship	→ Create parent-child relationship

---

Module-4

Phases of Transaction



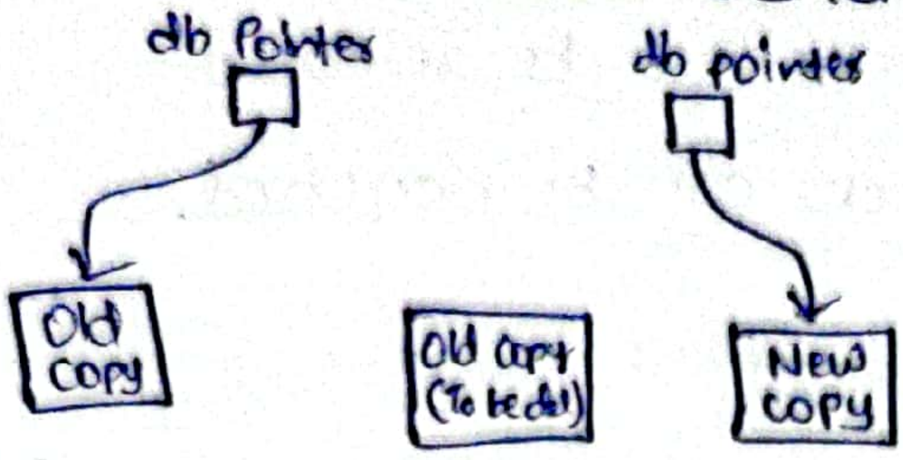
ACID Properties:

1. **Atomicity** :- (also "all or nothing rule")
   Either the transaction takes place or doesn't happen at all.
   There's no midway. like partial transaction.
   Two operations : Abort & Commit.
2. **Consistency**:- DB must be consistent before & after the transaction
   Refers to correctness of DB.
3. **Isolation**:- Multiple transactions occur independently with no interference.
   , Changes occuring in particular transaction will not be visible to any other transaction until it occured.
4. **Durability**:- After transaction, updates & modifications to DB are stored in & written to disk & they persist even if System failure occurs. Thus, effect of transactions are never lost.

Implementation of Atomicity & Durability

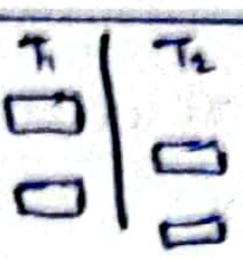
→ Recovery-management component.

→ Shadow-database scheme :



Interleaved manner
⇓
Alternate fashion.

Concurrent execution of Transaction



→ Multi user system - multiple users - same database - same time

→ Simultaneous execution in non-interleaved manner.

→ No operation should affed other executing operations - maintaining the consistency.

→ DB Transactions has 2 main operations : R/W. if ~~not~~ performed in non-interleaved manner, the data may become inconsistent.

Serializability:

→ Serial schedule - executed in series (one after other)

→ Serializability is used to find correct non-serial schedules in DB.

→ Maintains consistency & relates to Isolation features of transaction.

→ Serializability decides if an interleaved non-serial schedule is Serializable or not.

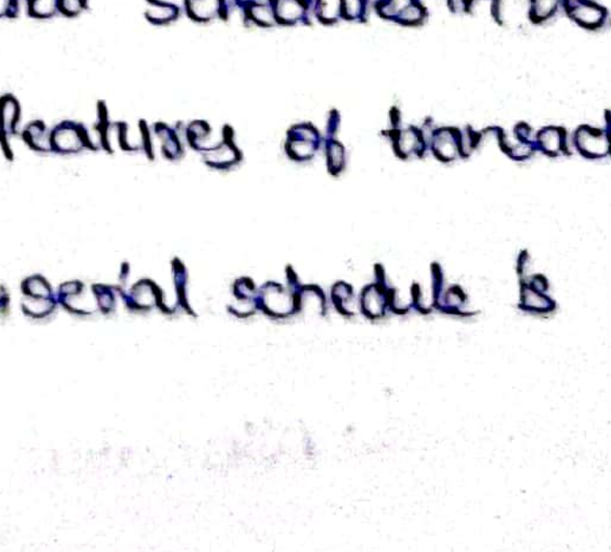
Two-Phase Locking Protocol (2PL)

Execution Phase

Part 1 → Start of transaction (seeks permission for lock)

Part 2 → Transaction acquires all locks

Part 3 → Transaction releases all locks



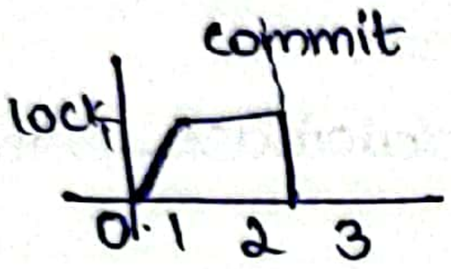
Phases of 2PL:

1. Growing phase: New lock on data can be acquired
   no lock can be released.
2. Shrinking phase: existing locks can be released
   no new locks can be acquired.

Strict 2PL:

→ Same as 2PL but holds all locks till the end of transaction.

→ locks will be released instantly after committing transaction.



Concurrency control protocols:

→ To enforce isolation

→ To preserve DB consistency

→ To resolve conflicts of RW, WR, WW.

1. Lock Based Protocols:

→ A transaction requires appropriate lock to read & write data.

Shared lock: Read-Only lock.

↳ Can be shared blw the transaction as it cannot be written.

Exclusive lock: R/W lock.

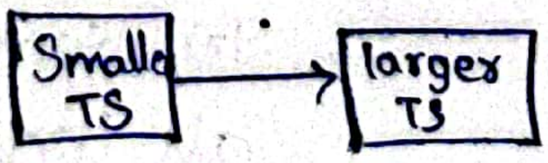
↳ Allocated only to single transaction at particular instance of ti

2. Timestamp based protocols:

→ used to order the transactions based on their timestamps.
(in the chronological order)

→ Timestamp based protocols starts working as soon as transaction is created.

→ Priority ∝ 1/arrival time.



3. Validation based protocols: (also Optimistic concurrency control technique)
↓
very less interference occurs

3 Phases of Execution

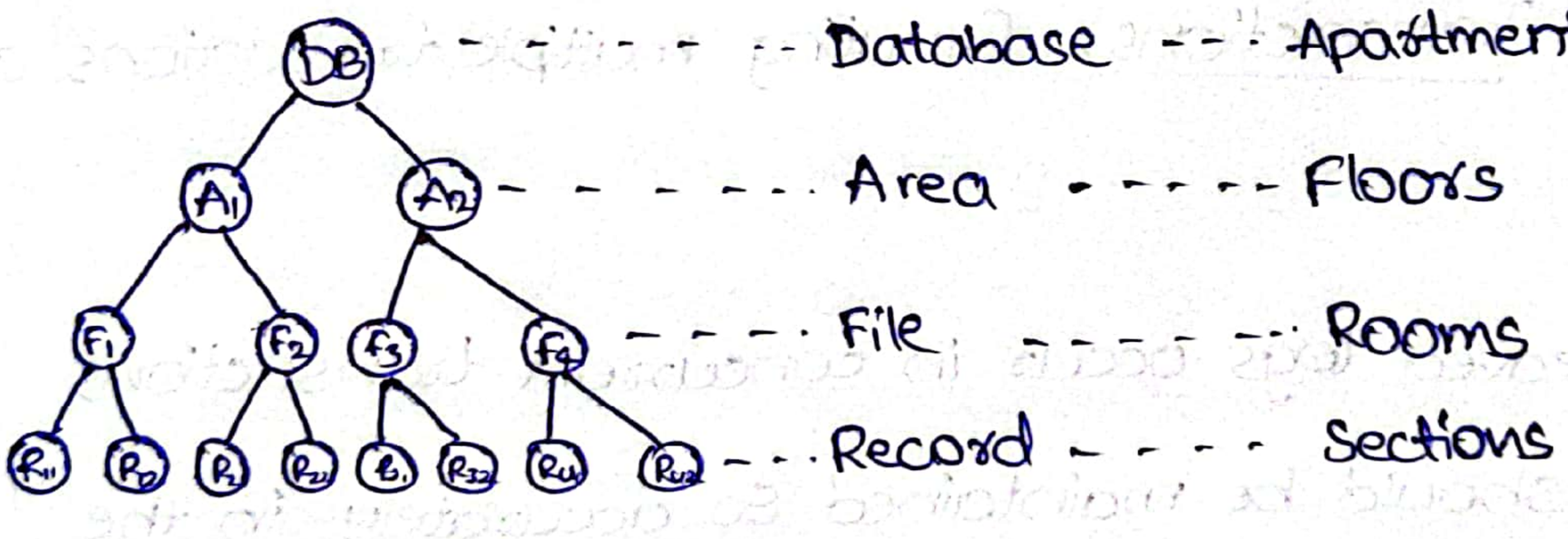
1 → Read Phase : → reads various data items and stores in temporary variables.
→ Can modify temp. variables without changing actual DB.

2 → Validation phase : → Temp. variables will be validated with actual DB to check with serializability.

3 → write phase : → After validation, temp results are written to DB or transaction will be rolled back.

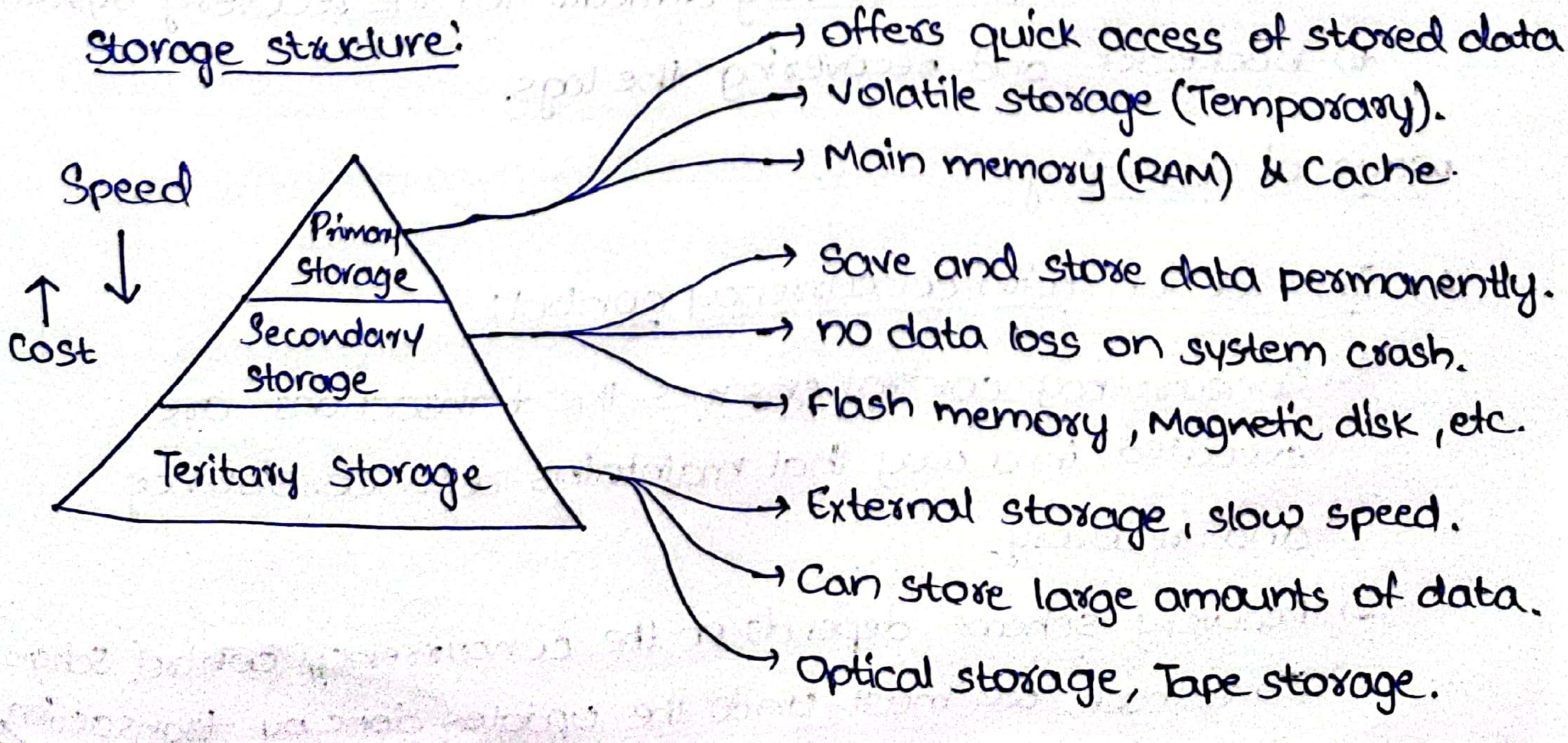
Multiple Granularity: (lol... it's a protocol) [enhances concurrency]

→ Hierarchically breaking up the DB into blocks which can be locked.
→ Maintains the track of what and how to lock.
→ locking and unlocking of a data item made easy.
→ Represented as a tree.



M.G. Tree Hierarchy

Storage structure:



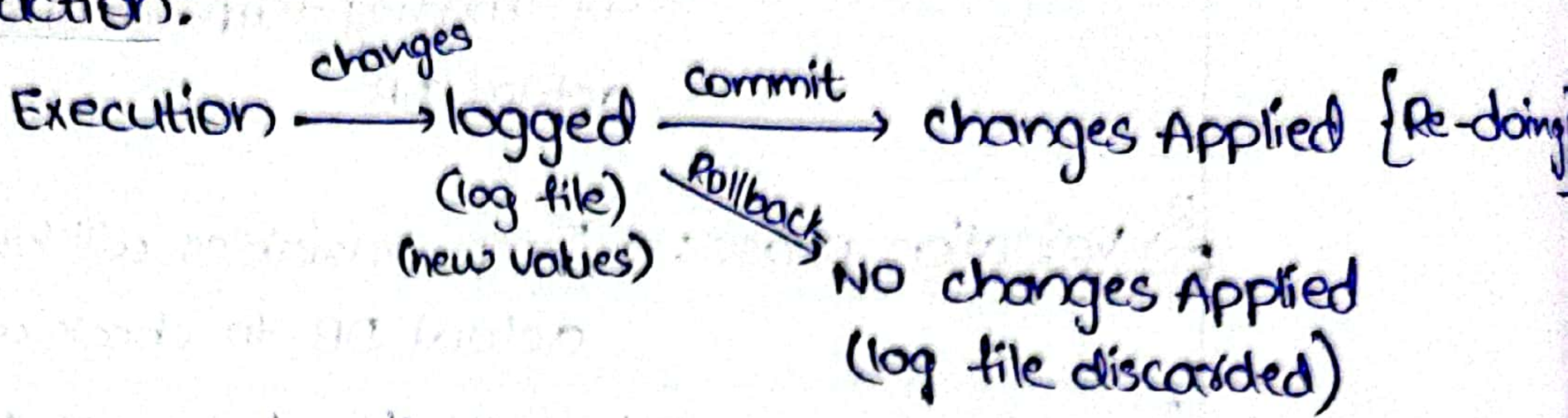
Database Modification:

1. Deferred update: (also NO-UNDO/REDO technique)

   → Maintains log files of transactions.

   → used to recover transaction failures occured unexpectedly.

   → Transaction:



2. Immediate update: (also UNDO/REDO technique)

   → Same as previous but log files contain both old aswell as new values.

   → It rollback, old values of log file are stored in the DB

   → used for recovery of transaction failures that occur due to power, memory, OS failures.

Concurrent Transactions: Executing multiple transactions at a time.

Recovery:

→ Interleaved logs occur in concurrent transactions

→ logs should be maintained so accurately in the order of execution.

→ During recovery, it'd be very difficult for the recovery system to backtrack and recovering the logs.

ways of recovering:

1. Interaction with concurrency control:

   → Concurrency control ensures the transactions are executed in a way that maintains data consistency and recovery.

   → Recovery scheme depends on the concurrency control scheme. To rollback, we must undo the updates done by transaction

2. Transaction Rollback:

→ use logs to rollback a failed transaction.

→ System scans the log backwards a failed transaction, for every log record found in the log, the system restores the data item.

3. Checkpoints:

→ Saving a snapshot of the application state.

→ checkpoints are the predefined points in time where the DBMS records the state of DB & transaction logs.

→ Creation: DBMS creates a checkpoint by writing all the modified data from memory to disk, and recording the checkpoint in the logs.

→ Recovery: System starts applying changes from logs starting from most recent checkpoint rather than from beginning, reducing recovery time.

4. Restart recovery:

→ This process aims to bring the system back to its normal operation after a failure. It involves replaying the transactions that were committed but not fully yet before the failure.

→ Redo, Transaction commit, Resuming normal operations.

Buffer Management:

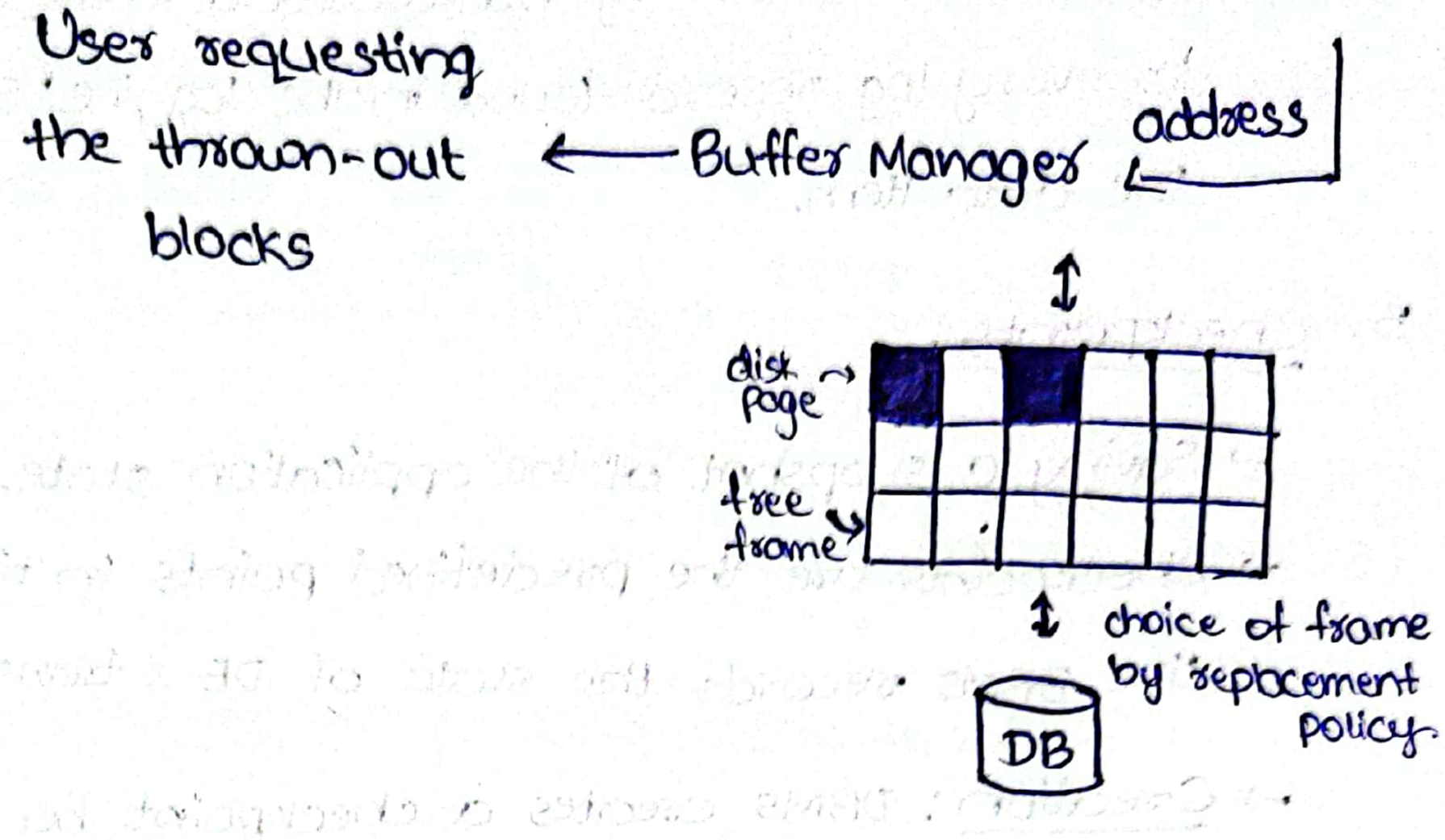
→ Buffer Manages - allocates space to the buffer to store data into buffer.

→ User requesting a particular block → if block available in buffer → Buffer Manages provides the block address in Main memory.

→ if block not available → Buffer Manager allocates the block in the buffer.

→ No free space → existing blocks were thrown from the buffer for space → these were written back into disk.

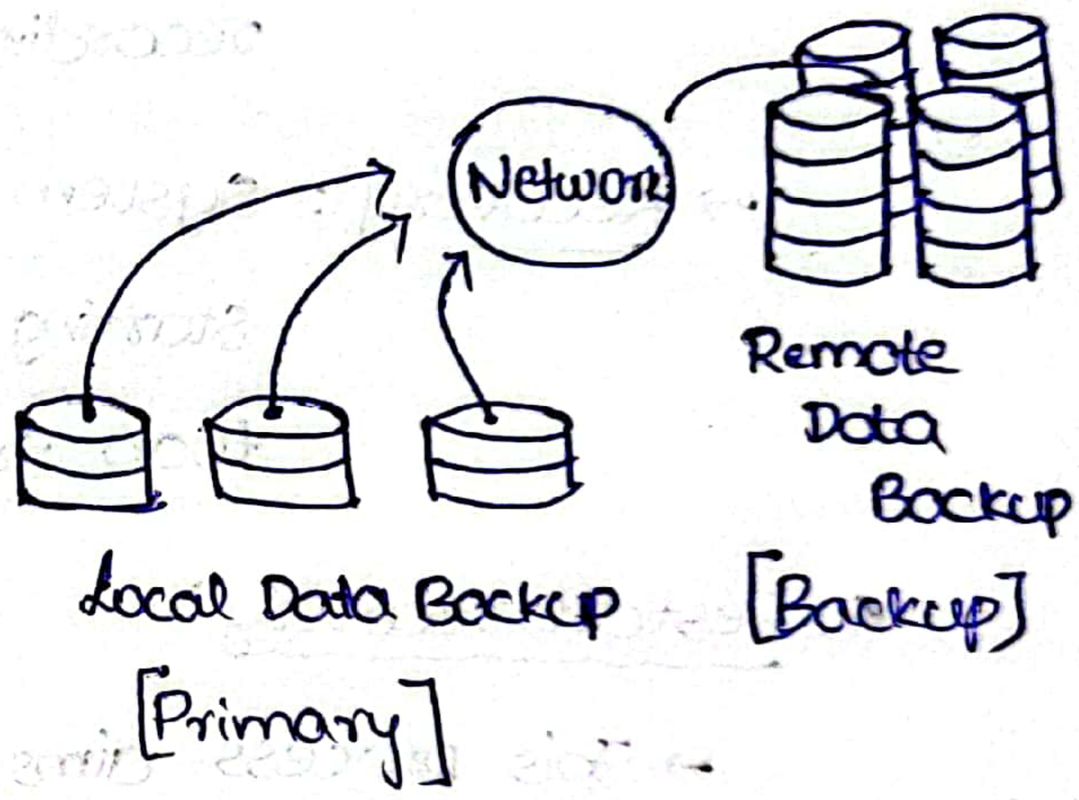
[location in Main memo



Remote Backup Systems:

→ Remote backup provides a sense of security in case the primary location where the DB stored gets destroyed.

→ Remote backup can be online / offline / real-time. In case it is offline, it is maintained manually.



Transfer of Control:

Backup site ready
↓
Recovery at Backup site
↓
Backup site takes over
↓
Transfer control back
↓
Apply updates locally
↓
Old primary recovers
↓
Control transferred

Time to recover:

Backup site ready
↓
Periodic recovery → continuously updating its own copy of DB from prev. state
↓
Checkpoint & deleting old logs
↓
Hot-Spare configuration
↓
Continuous redo processing
↓
Detecting primary failure
↓
Ready for new transactions.