Segment – 04

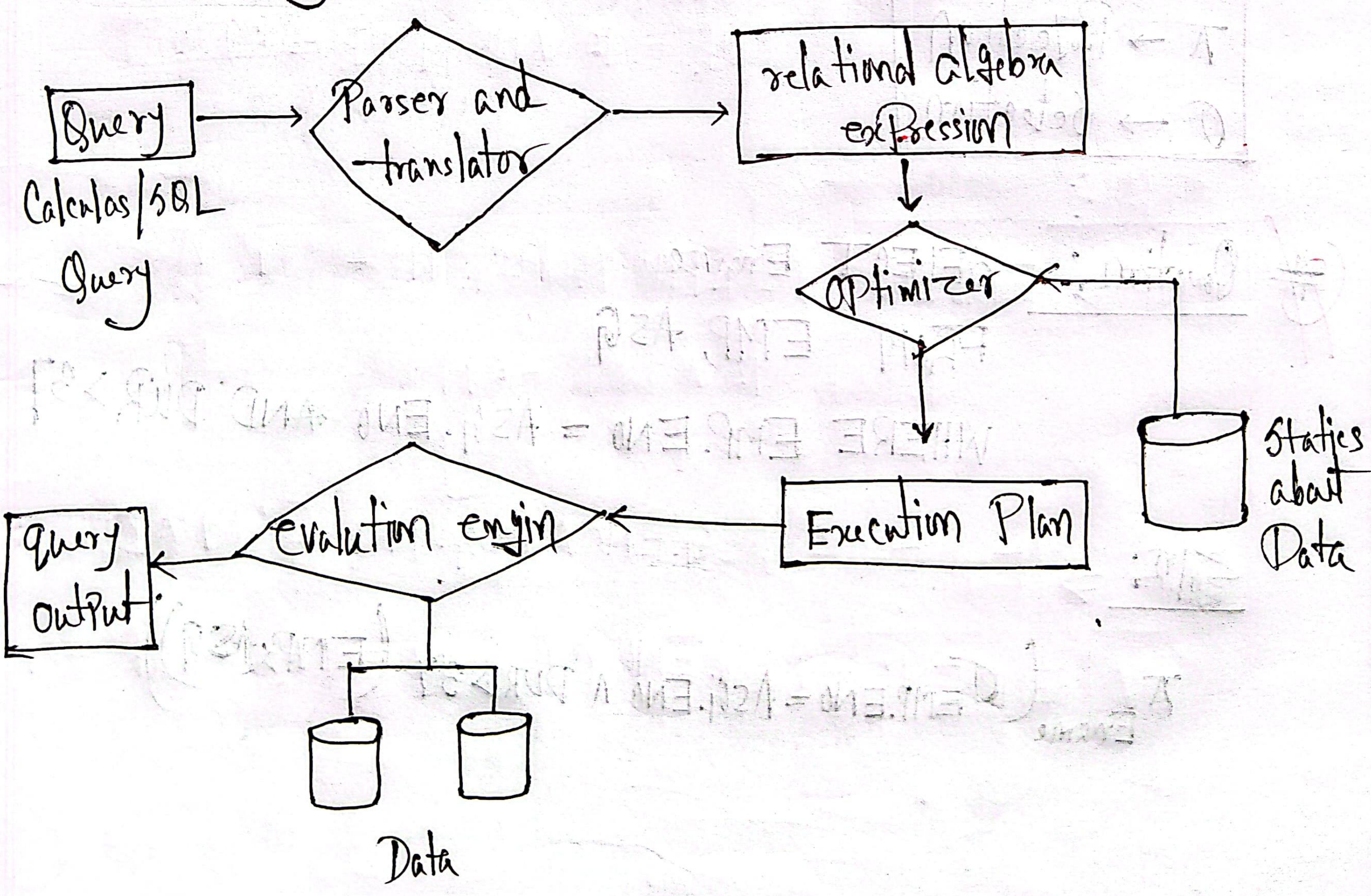
Query Processing :

3 step Process that transforms a high level query (relational Calculas/SQL) into an equivalent and more efficient lower-level query (relational algebra).

(i) Parsing and translation

(ii) Optimizer

(iii) Evaluation : The query evalution engine takes an (optimal) evalution plan, executes that plan and returns the answer to the query.



Question: Retrieve the birth date and address of the employees whose ~~ma~~ name is "John Smith".

```
SELECT Bdate, Address
FROM EMPLOYEE
WHERE Fname = 'John' AND Lname = 'Smith';
```

Soln :

$$\pi_{Bdate,\ Address}\left(\sigma_{Fname = 'John' \wedge Lname = 'Smith'}(EMPLOYEE)\right).$$

$$\boxed{\begin{array}{l} \pi \rightarrow \text{Projection} \\ \sigma \rightarrow \text{Selection} \end{array}}$$

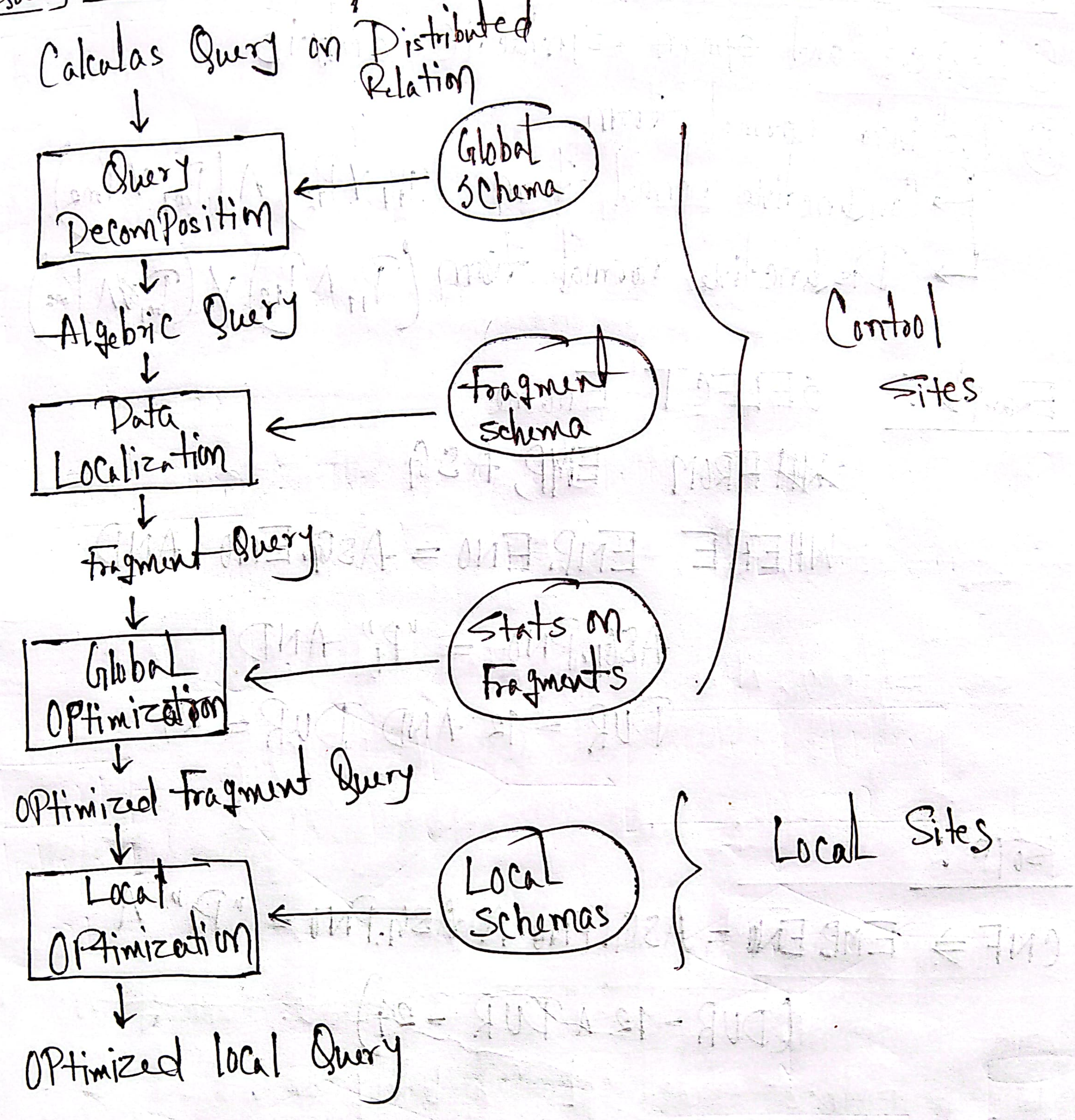
Question:

```
SELECT Ename
FROM EMP, ASG
WHERE EMP.ENO = ASG.ENO AND DUR > 37
```

Soln :

$$\pi_{Ename}\left(\sigma_{EMP.ENO = ASG.ENO \wedge DUR > 37}(EMP \times ASG)\right)$$

Query ~~Processing~~ Decomposition steps / Data Localization :



4-Steps of Query decomposition :

(i) Normalization (CNF, DNF)

(ii) Analysis (Lexical & Syntactic Analysis)

(iii) Elimination of redundancy

(iv) Rewritting

Normalization : 2 steps

(i) Lexical and ~~synon~~ syntactic analysis

(ii) Put into normal ~~form~~

→ Conjunctive normal form $(P_{11} \vee P_{12}) \wedge (P_{m1} \vee P_{m2})$

→ Disjunctive normal form $(P_{11} \wedge P_{12}) \vee (P_{m1} \wedge P_{m2})$

Example:

SELECT Ename
~~WH~~ FROM EMP, ASG
WHERE EMP.ENO = ASG.ENO AND
ASG.PNO = "P1" AND
DUR = 12 AND DUR = 24

Soln:

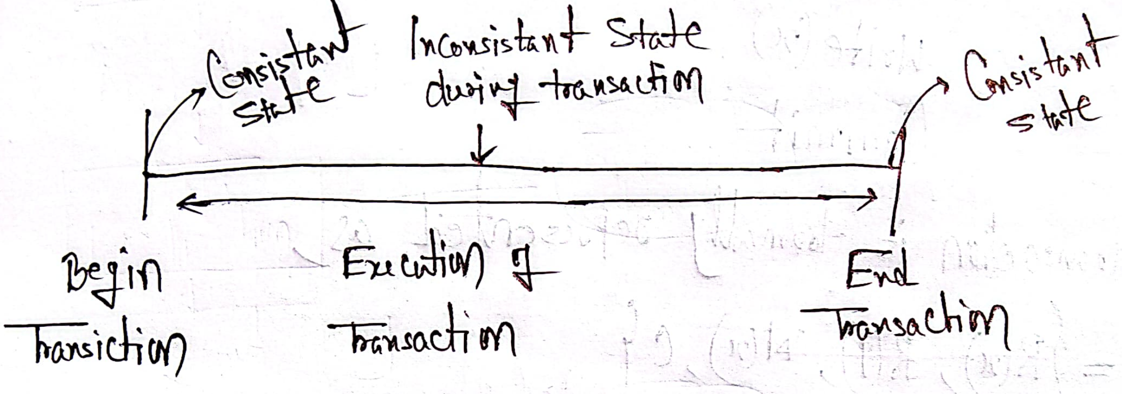
CNF ⇒ EMP.ENO = ASG.ENO ∧ ASG.PNO = "P1" ∧ (DUR = 12 ∧ DUR = 24)

DNF ⇒ (EMP.ENO = ASG.ENO ∧ ASG.PNO = 'P1' ∧ DUR = 12)
∨ (EMP.ENO = ASG.ENO ∧ ASG.PNO = 'P1' ∧ DUR = 24)

Segment - 05

Transaction Management:

A Collection of actions that transform the DB from one Consistent state into another Consistent state; during the execution the DB might be inconsistent.



States of Transaction:

(i) Active → Initial state / during the execution.

(ii) Partially Commited → After the final statement has been executed.

(iii) Commited → After successful Completion.

(iv) Failed → After the discovery that normal execution Can no longer Proceed.

(v) Aborted → After the transaction has been rolled back and the DB restored to its state Prior the start of the transaction. Restart it again or kill it.

Formalization of Transaction :

Example :

Read (x)
Read (y)
$x \leftarrow x+y$
Write (x)
Commit

The transaction is formally represented as,

$$\Sigma = \{R(x), R(y), W(x), C\}$$

$$\alpha = \{(R(x), W(x)), (R(y), W(x)), (W(x), C), (R(x), C), (R(y), C)\}$$

ACID Properties :

(i) **Atomicity :** Either all or None.

A transaction is treated as a single unit of operation and is either executed completely or not at all.

(ii) **Consistency :** Before transaction start and after transaction completed the total amount of data should be same.

(iii) **Isolation :** A transaction is executed as if it would be the only one.

(iv) Durability: The updates of a Committed transaction are Permanent in DB.

Problems in transactions:

(i) Dirty read: T_1 modifies x which is then read by T_2 before T_1 terminates; if T_1 aborts T_2 has read value which never exists in DB.

(ii) Non-repeatable (fuzzy) read: T_1 read x; T_2 then modifies or deletes x and Commits; T_1 tries to read x again but reads a different value or can't find it.

(iii) Phantom: T_1 searches the database according to a Predicate P while ~~P~~ ~~is~~ T_2 inserts new tuples that satisfy P.

Transaction Process issues:

(i) Transaction structure
(ii) Internal database Consistency
(iii) Reliability Protocols
(iv) Concurrency Control algorithms
(v) Replica Control Protocols.

or,

(i) Concurrency Control
(ii) Isolation
(iii) Durability
(iv) Performance

Segment - 06

Concurrency Control : It's the Problem of synchronizing concurrent transaction such that the following two Properties are achived—

(i) The Consistency of DB is maintained.

(ii) The maximum degree of Concurrency of operation is achived.

Conflicting Operation :

- O_{ij} = read(x) and O_{kl} = Write(x)
- O_{ij} = Write(x) and O_{kl} = read(x)
- O_{ij} = Write(x) and O_{kl} = Write(x)

Locking Based Algorithm :

2 Types of lock —

(i) read lock (rl) / Shared lock

(ii) Write lock (wl) / exclusive lock.

Compatibility Matrix :

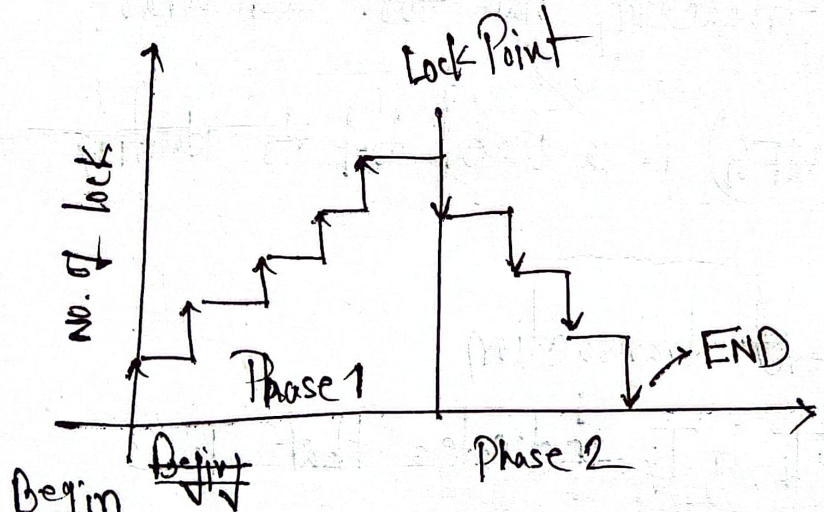
	$rl_i(x)$	$Wl_i(x)$
$rl_i(x)$	Compatible	Not Compatible
$Wl_i(x)$	Not Compatible	Not Compatible

Two Phase Locking (2PL) :

Each transaction is executed in two Phase

(i) Growing Phase → obtain lock

(ii) Shrinking Phase → Releases lock



↑ obtain lock

↓ Release lock

(Serializability, Conflict-Serialzability, Deadlock Freedom, Recoverability

Properties of 2PL Protocol :

→ Generates Conflict-Serialzable schedule

→ But schedules may ~~cau~~ Cause Cascading aborts —

* If a transaction aborts after it releases a lock it may Cause other transactions that have accessed the unlocked data item to abort as well.

Deadlock Management

Deadlock : A set of transaction is in a deadlock situation if several trasaction wait for each other.

A Wait-for-Graph (WFG) is a useful tool to identify deadlocks,

- The nodes represent transaction
- An edge from T_i to T_j indicates that T_i is waiting for T_j
- If the WFG has a Cycle , we have a deadlock situation.

FAHIM REZA | C211046

(Deadlock Prevention, Deadlock detection and recovery)

Deadlock Management:

Assume T_1 and T_2 run at site 1, T_3 and T_4 run at site 2, and the following wait for - relationships between them:

$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_1$. This Deadlock cannot be detected by the LWFGs but, by the GWFG which shows intersite waiting.

Local WFG:

$$\begin{array}{ccc} \text{site 1} & \qquad & \text{site 2} \\ T_1 & & T_3 \\ \downarrow & & \uparrow \\ T_2 & & T_4 \end{array}$$

Global WFG:

$$\begin{tikzcd} \text{site 1} & \text{site 2} \\ T_1 \arrow[d] & T_3 \arrow[l] \\ T_2 \arrow[r] & T_4 \arrow[u] \end{tikzcd}$$

Deadlock Prevention: Guarantee that deadlock never occurs:

(i) Check transaction when it's initiated.

(ii) Start it only if all required resources are available.

(iii) All resources which may be needed by a transaction must be predeclared.

3 phenomena/situation that occur during transaction —

(i) **Dirty read:** T_1 modifies x which is then read by T_2. Then T_1 aborts, T_2 has read a value which never exist in the DB.

(ii) **Non-repeatable (Fuzzy) read:** T_1 read x; T_2 then modifies or deletes x and commits. T_1 tries to read x again but read a different value or can't find it.

(iii) **Phantom:** T_1 searches the database according to a predicate P while T_2 inserts new tuples that satisfy P.

Segment – 7

Reliability:

A reliable DDBMS is one that can continue to process user requests even when the underlying system is unreliable, i.e failure occur.

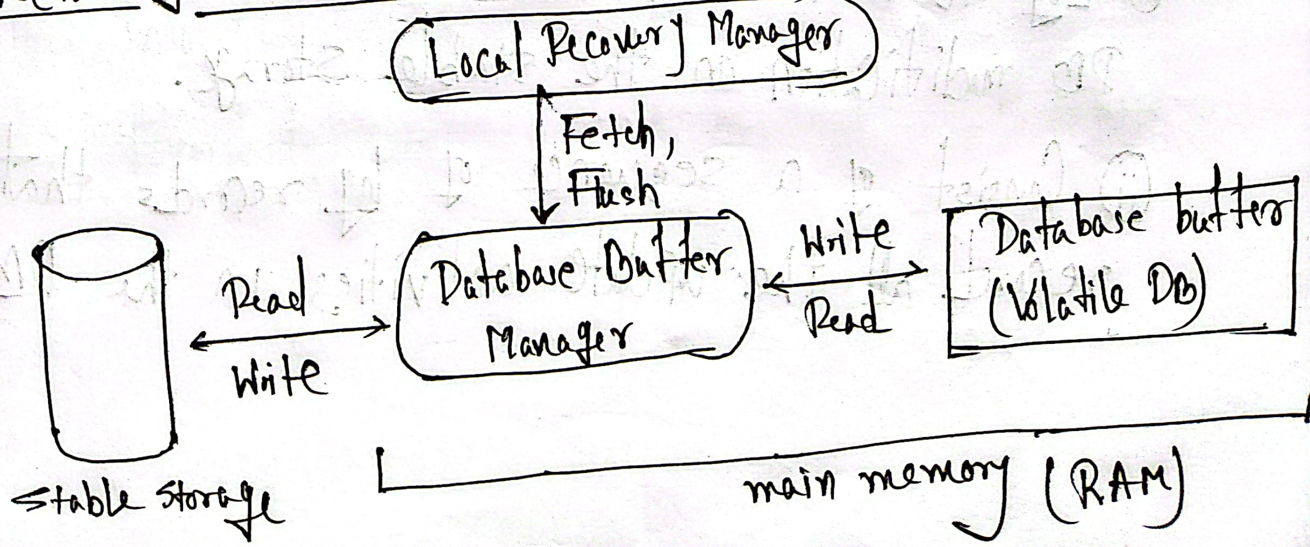
Types of Failure:

(i) Transaction failure,
(ii) System failures, e.g., system crash, Power supply failure.
(iii) Media failures, e.g: hard disk failure.
(iv) Communication failure, e.g: lost/undeliverable messages.

Local Recovery Management:

A local recovery manager (LRM) maintain the atomicity and durability properties of local transaction at each site.

Architecture of LRM:



Two ways for the LRM to deal with update operation:

① **In-Place update:**

→ Physically changes the value of the data item in the stable database.

→ As a result, Previous values are lost.

→ Mostly used in databases.

② **Out-of Place update:**

→ The new ~~it~~ value of updated data item are stored separately from the old values.

→ Periodically, the updated values have to be integrated into the stable DB.

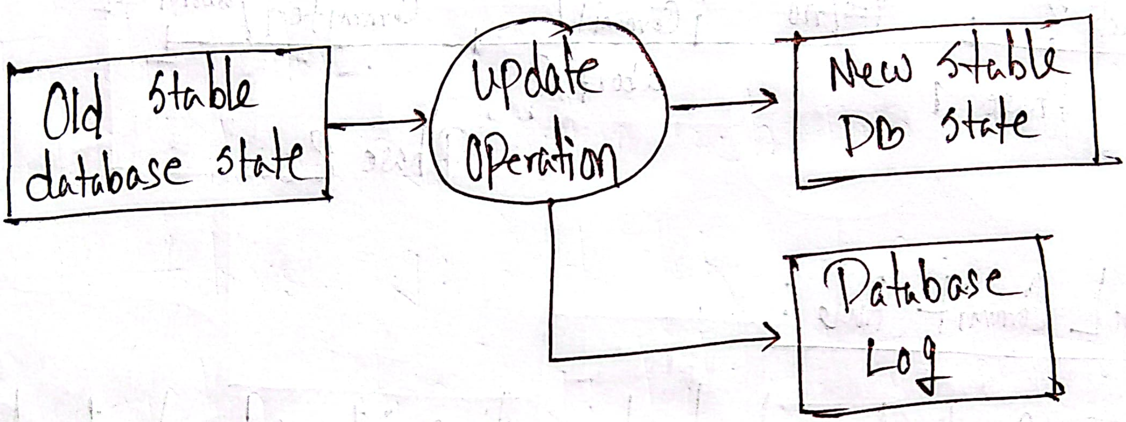
In-Place-update:

① Log file is the most popular structure for recording DB modification on the stable storage.

② Consist of a sequence of log records that record all the update activities in the DB.

(iii) Each log record describes a ~~single~~ significant event during transaction Processing.

(iv) With the information in the log file the recovery manager can restore the consistency of the DB in case of a failure.

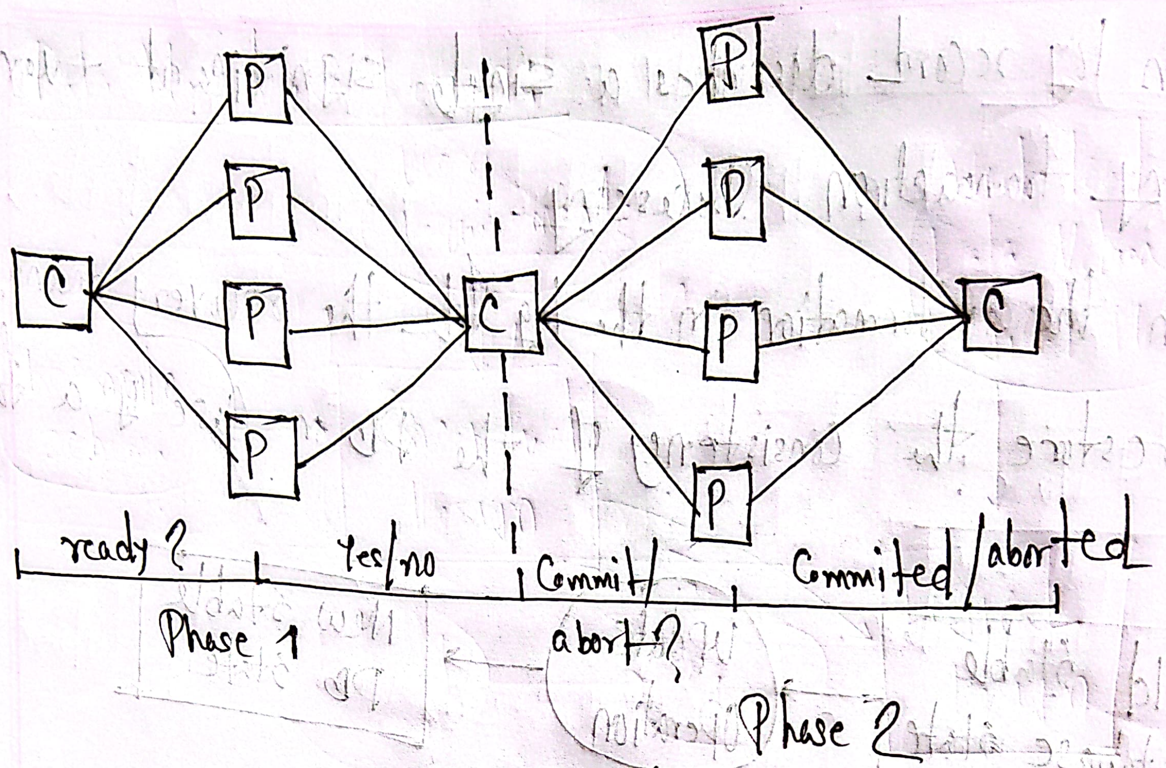


2 Phase Commit (2PC) Protocol:

Simple Protocol that ensures that atomic Commitment of distributed transactions.

Phase 1: The Coordinator get the Participants ready to write the rule into the database.

Phase 2: Everybody writes the results into the database.



Global Commit Rule :

(i) The Coordinator aborts transaction if and only if at least one Participant vote to abort it.

(ii) The Coordinator Commits a transaction if and only if all the Participants vote to Commit it.

Segment - 8

Data Warehouse :

A Data Warehouse is a —

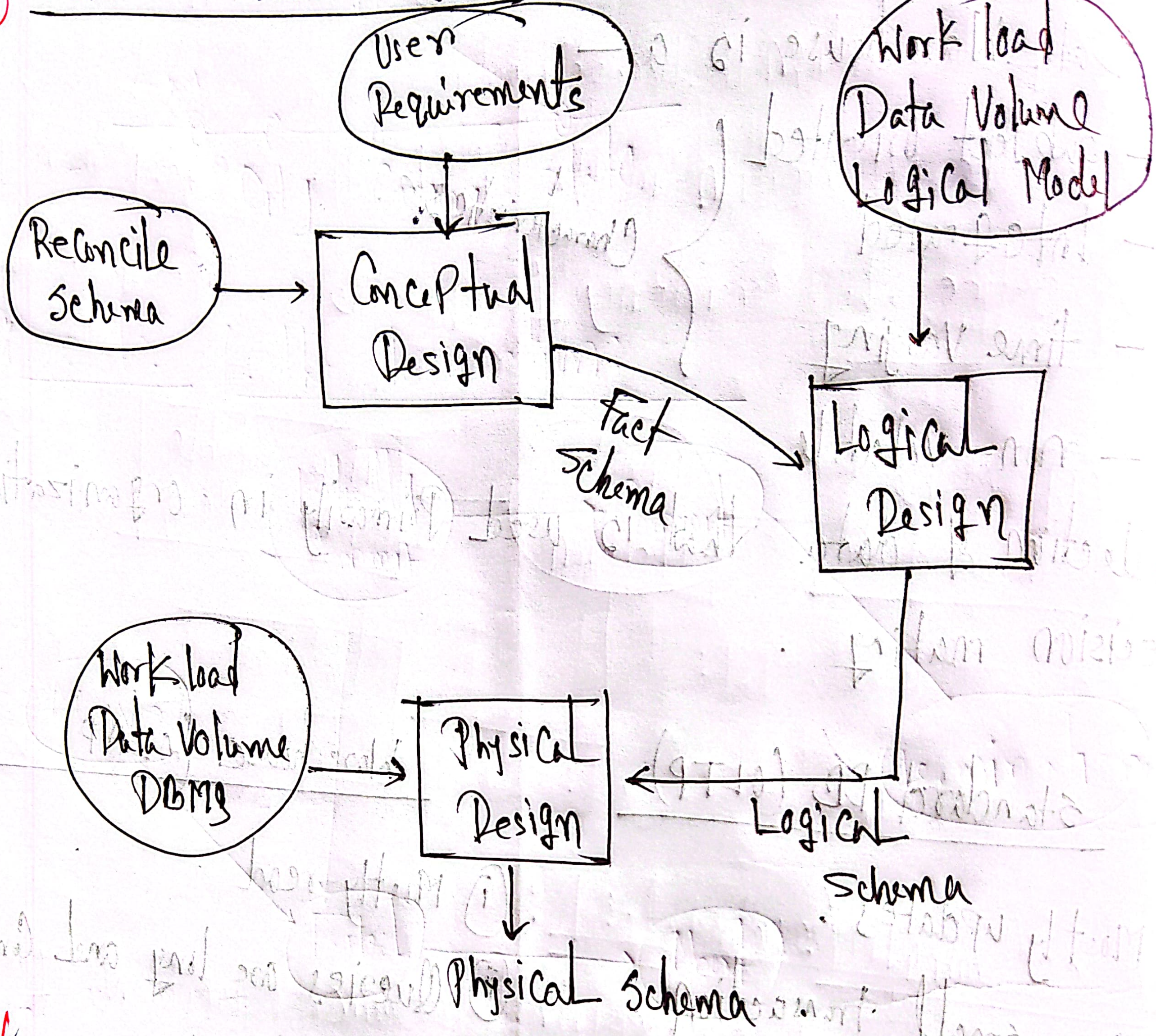
- subject oriented
- Integrated
- time varying
- non-volatile

} Characteristics

Collection of data that is used Primarily in organizational decision making.

# Standard DB (OLTP)	Warehouse (OLAP)
(i) Mostly updates.	(i) Mostly read
(ii) Many small transaction	(ii) Queries are long and Complex
(iii) Mb - Gb of data	(iii) Gb - Tb of data
(iv) Current snapshot.	(iv) History.
(v) Index / hash on P.K.	(v) Lots of scan
(vi) Raw data	(vi) summarized, reconciled data
(vii) Thousands of users.	(vii) Hundreds of users.

Data Ware house Design:



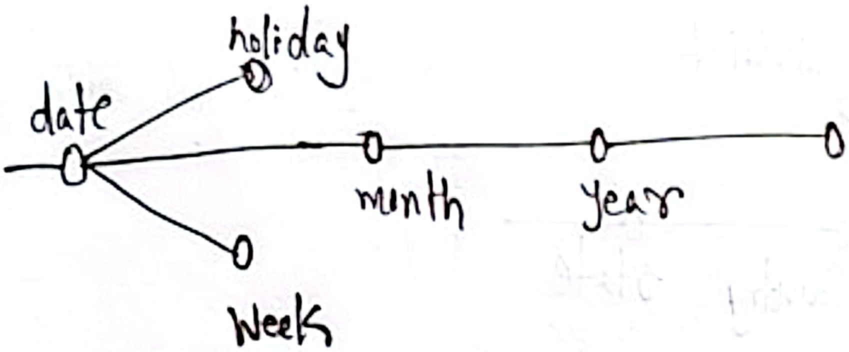
Hierarchy:

A hierarchy is a directional tree whose -

→ Nodes are dimensional attributes

→ Edges describe n:1 associations between Pairs of dimensional attribute

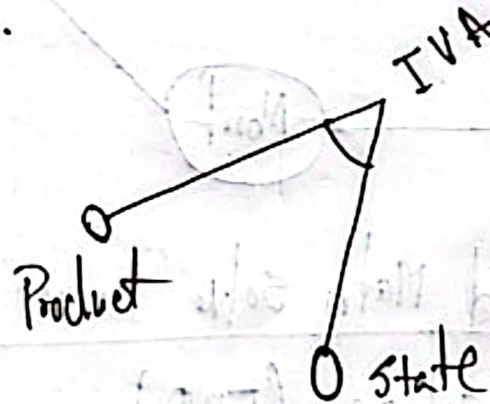
→ root is the Considered dimension.



Cross-dimensional attributes :

A Cross-dimensional attribute is a dimensional or a descriptive attribute whose value is obtained by Combining values of some dimensional attributes.

e.g: 'IVA' is Computed based on the "Product" and the "state".

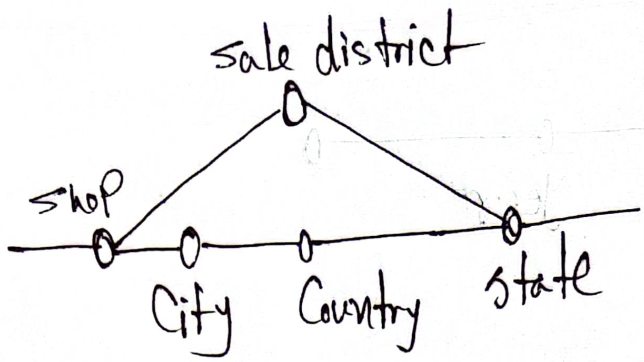


Convergence : In Convergence —

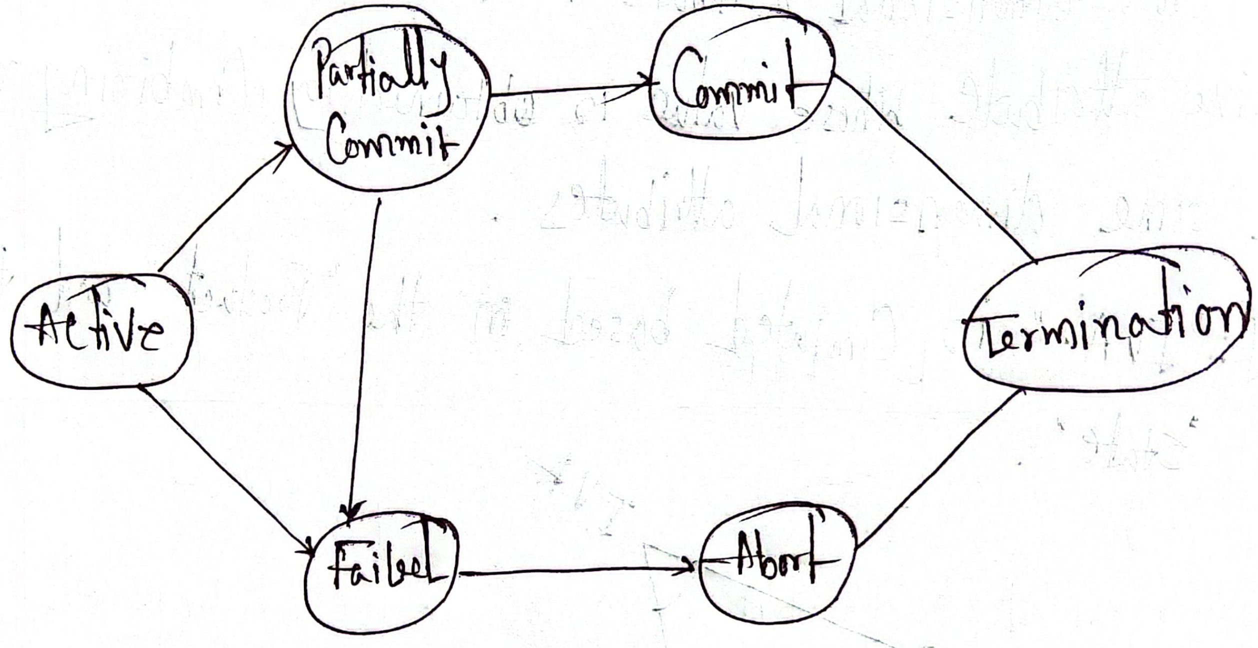
→ Two dimensional attributes can be connected by more than two distinct and oriented edges.

e.g: shop → City → Country → state.

or, shop → sale district → state.



States of the Transaction:



Data Warehouse related Math solve Process:

(i) Design Conceptual schema (ERD)

↓

(ii) Determined Fact, Measures and Dimensions, and draw logical schema diagram (Use grafting and Pruning)

↓

(iii) Draw attribute tree and Fact schema

(iv) Finally, Design a snowflake or star schema Diagram.

Timestamp-based locking: An algorithm that assigns timestams to transactions and uses timestamps to determine the order in which locks are granted. This ensure that transactions are serialized in a consistant order.

FAHIM REZA | C211046
fahim.bdctg51@gmail.com