Q1. What is DDBMS. State its advantages & Disad.
2. Explain types of DDBMS.
3. What is fragmentation and explain types of fragmentation
4. Types of transparency and explain transparency.
5. Three tier client server architecture.
6. What is concurrency control.
7. 2PL protocol (explain)
8. Types of 2PL.

1) DDBMS -

Definition :- It is a database that is scattered or spread over different sites i.e. on multiple computers or over a network of computers. DDBMS is located on various sites.

Another definition: It is a database that runs and stores data across multiple computers and looks like doing everything on a single machine

Advantages -

1) High availability of data as data is replicated on systems and if one copy fails another copy can be used without delay in time

2) High scalability - can be scaled horizontally by adding new servers/nodes

3) Improved performance - As data is divided on multiple computers, there is distribution of

processing by reducing response time

4) <u>Load balancing</u> as data is available on different system and there will not be overload of execution on single machine / server.

5) <u>Parallel processing</u> and concurrency control is handled properly by no. of servers on which data is divided or replicated.

6) ~~Loca~~ ~~Dis~~ Better response - when user requests a query, due to efficient data distribution among servers, faster response will be given to user.

<u>Disadv</u> - 1) <u>Data integrity</u> - As there can be replicas of fragment, data integration from various servers can be an issue.

2) <u>Design will be complex</u> due to distribution of data on systems

3) There can be improper data distribution that can lead to increase in response time for the query.

Types of distributed DBMS

1) Homogeneous DDBMS : In this type of distributed DBMS every server will have same DBMS for storing and accessing data.

✓ All servers will have (installed) same OS, DBMS and data structure (if required)

✓ Simple to handle due to homogeneous nature of DBMS.

e.g.

Mysql	Mysql	Mysql
S_1	S_2	S_3

If there are 3 servers then in homogeneous DDBMS all servers are having same DBMS to handle distributed data.

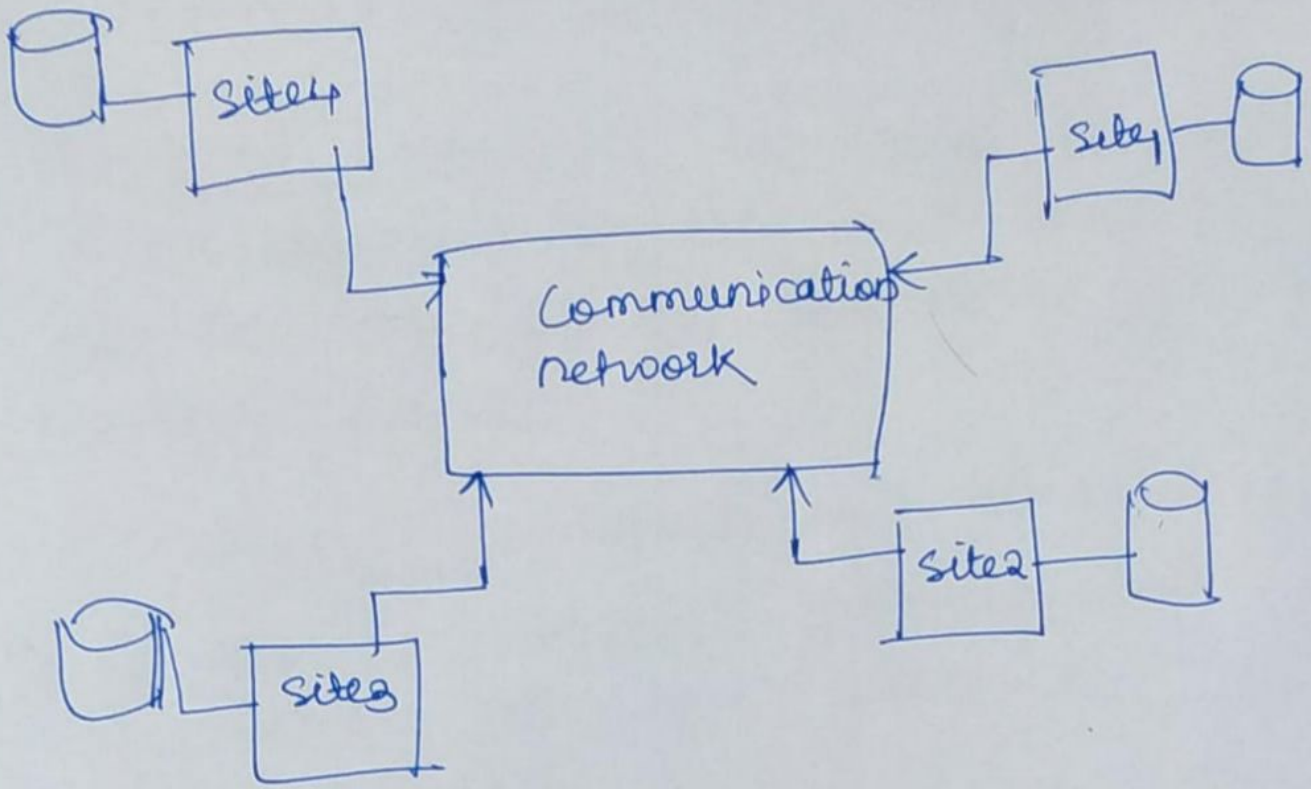
2) Heterogeneous DDBMS – In this type of DDBMS all servers will have different DBMS on systems to handle data that is distributed on systems. Various OS and databases will be installed on different servers. So, translations are necessary for communication among the servers.

→ from one dbms format to other format for communication.

e.g.

Mysql	Oracle	Ms-access
S_1	S_2	S_3

If there are three servers in system to handle database then as an example from above scenario S_1 is installed with Mysql dbms, S_2 with Oracle and S_3 with Ms-access databases respectively. Every server is handling different dbms.



(DDBMS enviornment)

Fragmentation :- The process of dividing data into subtables or smaller relations so that data can be ~~divi~~ stored in different systems.

- The small pieces are called fragments.
  - are called logical units
  - reconstruction of table by combining no. of fragments (call fragments) of table.
  - After reconstruction of table from fragments there should not be loss of data.
  - Data can be divided by 2 ways.
    - By giving condition on attributes to form horizontal fragments of data / table
    - By dividing no. of columns into subtables to form vertical fragments of data / table.

Types of fragmentation

1) Horizontal fragmentation -

It is process of dividing table horizontally by writing condition on attributes to divide rows from table. So data is divided horizontally into set or group of rows after satisfying the given condition on attribute

In relational algebra horizontal fragmentation on table T is represented as,

$$\sigma_C(T)$$

sigma operator for writing the condition

table

condition on attribute

e.g. Student (rollno, name, address, DIV)

$HF_1 = \sigma_{DIV=A}(\text{student})$

↳ one Horizontal fragment of table student for condition on attribute DIV = 'A'

~~Rep~~

$HF_2 = \sigma_{DIV=B}(Student)$

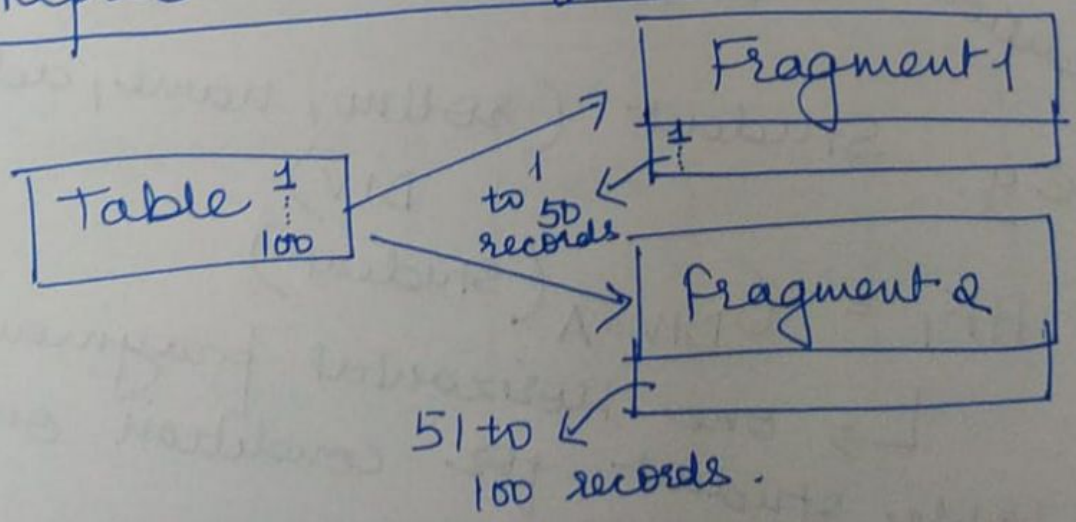
↳ Horizontal fragment with condition of DIV=B.

So in both fragments all rows or records are divided horizontally forming subrows in every fragments HF_1, HF_2. Both fragments will give you original table Student

$\boxed{Student = HF_1 \cup HF_2}$

Union operation combines rows for same column names.

Representation of HF :



SQL example

~~Select~~ create table HF1 as
Select * from student where
DIV = 'A';

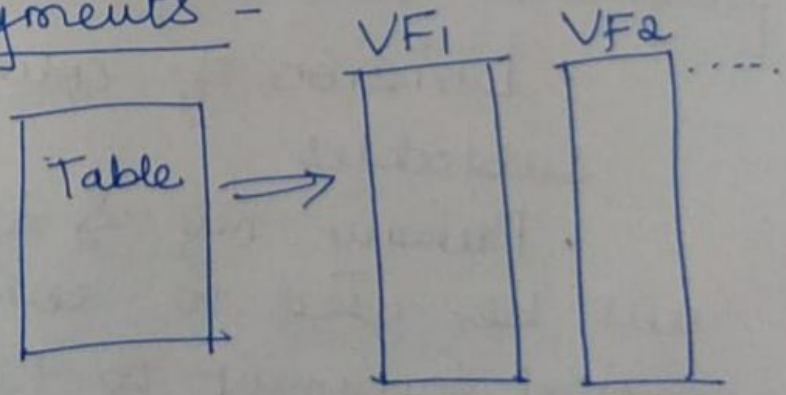
Rollno	name	add	DIV
1			A
⋮			⋮
50			A

create table HF2 as
Select * from student where
DIV = 'B';

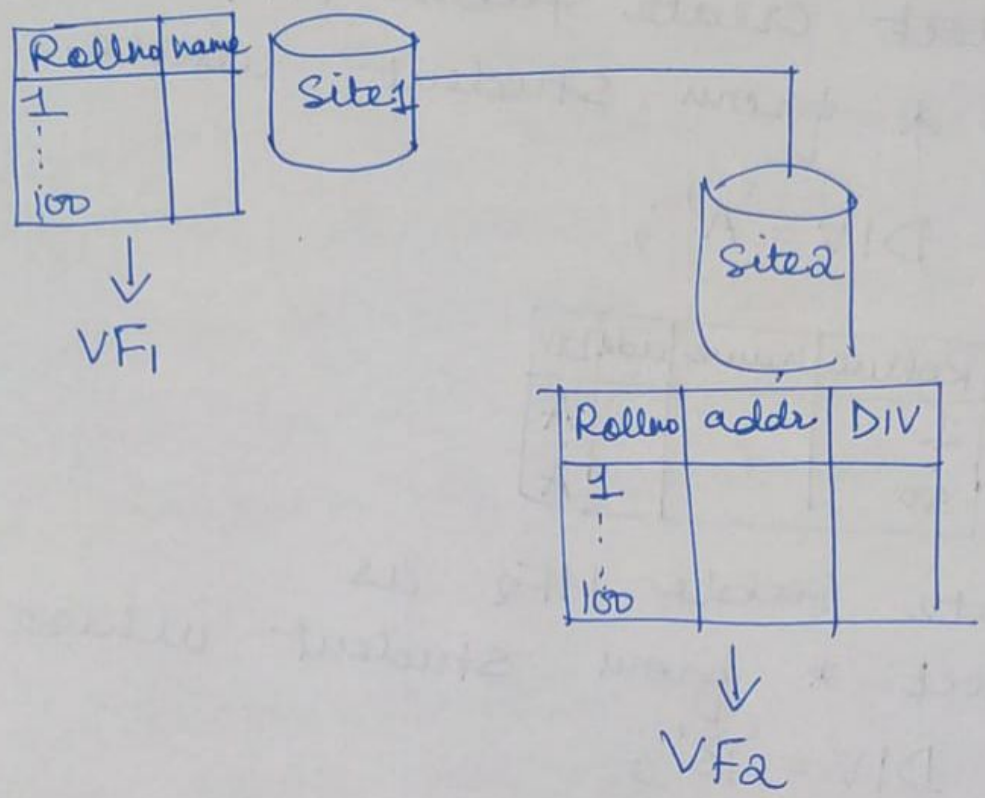
Rollno	name	addr	DIV
51			B
⋮			⋮
100			B

2) vertical fragments -

Representation



Example



Definition :- The process of dividing the table into subtable by dividing the no. of attributes (into) to form the fragments vertically.

- Division of columns to form subtables
- Primary key & foreign key approach will be used to connect the related vertical fragment to have meaningful data. e.g. Rollno in VF1 & VF2. If we don't consider rollno in VF2

then we will not be able to identify whose address and DIV is in VF_2.

- This fragmentation works on columns of table that's why vertical division of table.
- no. of records should be same in all fragments as there is division of attributes not of rows.

- Reconstruction of VF_1 & VF_2 will form the original table student with 100 records.

$$Student = VF_1 \bowtie VF_2$$

natural join will combine attributes of both fragments to get original table student.

In relational algebra :-

$$VF_1 = \Pi_{rollno, name}(Student)$$

$$VF_2 = \Pi_{rollno, addr, DIV}(Student)$$

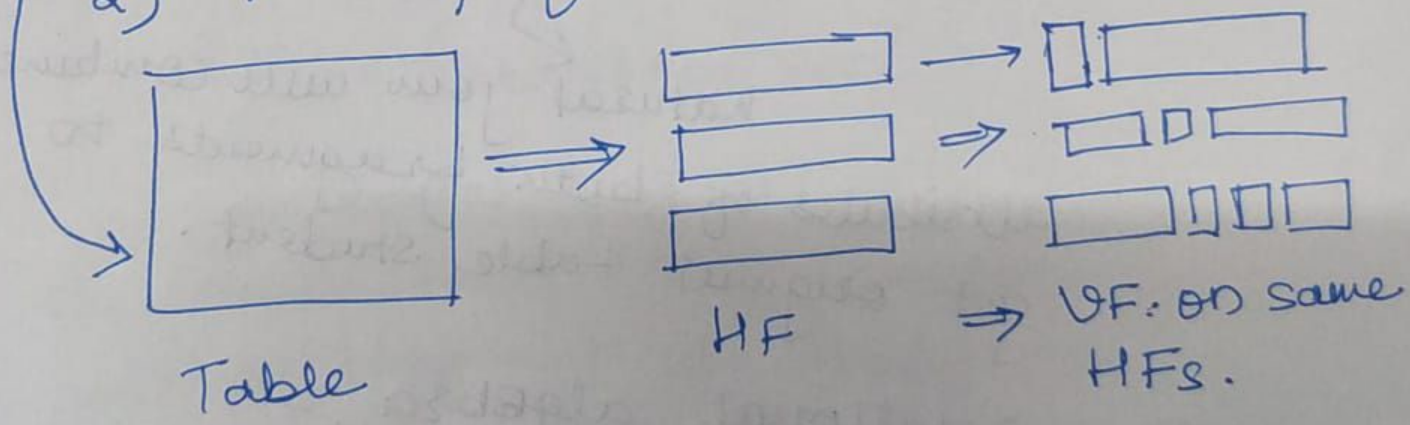
SQL syntax :-

Create table VF1 as select rollno, name from student;

Create table VF2 as select rollno, ~~name~~ addr, DIV from student;

3) <u>Mixed fragmentation</u> - It is combination HF and VF and it can be possible by 2 ways as:

1) First perform HF and then VF
2) First perform VF and then HF.



In Relational algebra =

1) $\sigma_c(\Pi_{a_1, a_2 \ldots}(T))$

writes condition on attribute

2) $\Pi_{a_1, a_2 \ldots a_n}(\sigma_c(T))$

displays columns 1st HF and then vertical

SQL example

Staff - staffNO, Name, DOB, position, salary, branchno

fragments of staff

$S_1 = \Pi_{staffNO, position, DOB, salary}(Staff)$

$S_2 = \Pi_{staffNO, Name, branchno}(Staff)$

vertical fragment on staff.

$S21 = \sigma_{branchno = 'A01'}(S_2)$

Horizantal fragment on S_2 (not on main table. Staff) which is vertical fragment of staff.

Create table S1 as

- Select staffno, position, DOB, salary from staff.

creation of S_1 fragment.

create table S2 as
select staffno, name, branchno
from staff;

create table S21 as select *
from S2 where branchno = 'A01';

mixed fragment created
from S2 which is vertical
fragment of staff.

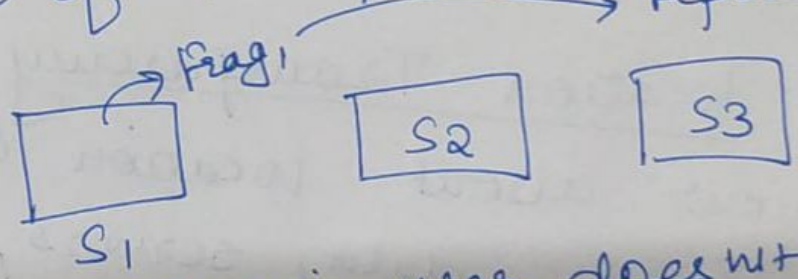
Types of transparency

In distributed system, there is way ~~of it~~ or technique of hiding implementation details from user which is called "transparency".

There are various types of transparency as:

1) Replication transparency :- In this type details of replicas name, location, count will be hidden and reliability feature is exerted by this type of transparency.

e.g.



In above scenario user doesn't know about existence of replica1 and system looks like a centralized system to user

2)

2) Failure transparency - users in background of system doesn't get an idea about failure of S/W or H/W components. If one of system fails, switched to another system where copy of data is available

3) ~~Rep~~ Concurrency Transparency :-
User doesn't know about concurrent execution of application programs or queries by using fragments or replicas.

4) Location Transparency : User doesn't know about location of resources that are data, servers, networks used in system. As well well as region or places of the servers are hidden.

5) fragmentation transparency :-
User is not aware about how many fragments are there of data and where they are stored for the parallel processing.

6) Concurrency transparency :- Users is not aware about concurrent execution of processes in network with the help of distributed data.

7) Performance Transparency : User considers whole system as a centralized system and how the performance is enhanced like by adding servers without affecting the current system's performance.

Three tier client server architecture

Client: is a machine or ~~server~~ process that gives / submits request (query) to server.

Server: is a machine or process that processes and executes query given by client and gives output

There are 3 layers of architecture

1) presentation layer - In this layer user interacts with system through web browser. by using languages like HTML, CSS, javascript.

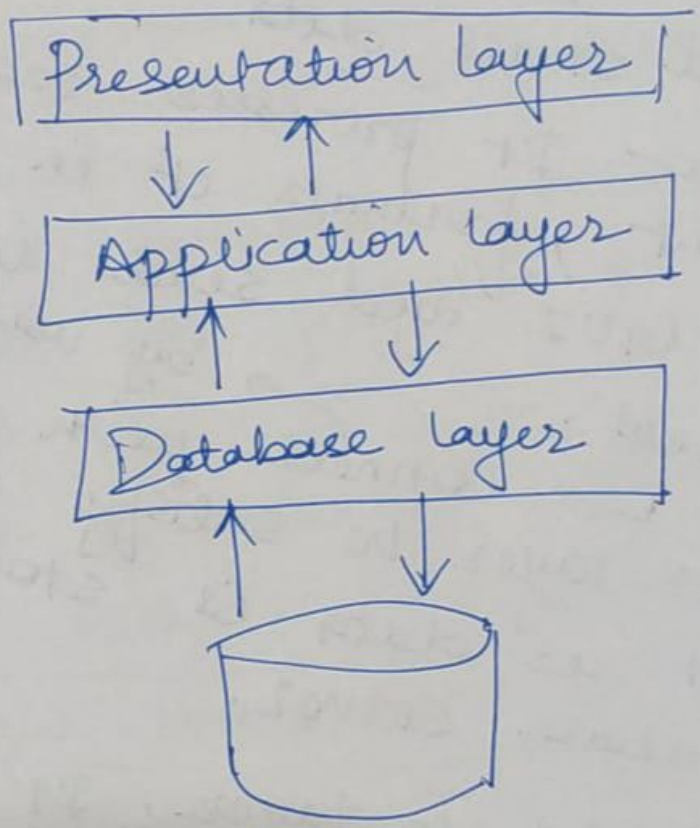
✓ Its purpose is to take request from client and displays information to the client

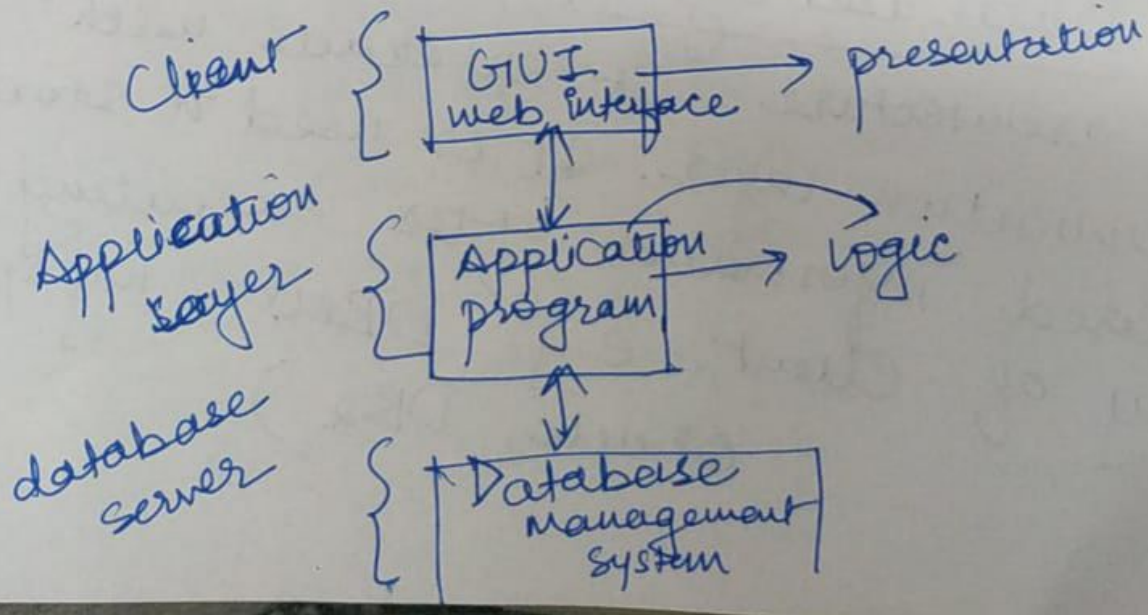
2) Application Layer: It is middle layer in architecture and request taken from presentation layer through client is processed here.

- It interacts with server and stores data.
- It processes request of the client, formats it in proper way by GUI and sends it back to client.
- connects with database or server layer to satisfy request of client as data is stored on the database server. by using ODBC.JDBC

3) Database layer: It is last layer of architecture. It interacts with Application layer. It is used to store processed information after executing query of client. e.g. DBMS (mysql oracle, DB2)

It interacts with application layer with output of query and application layer gives output to present in the form of GUI on the client's desk.





Concurrency control

The mechanism to control concurrent execution of transactions is called as concurrency control.

Locking Mechanism :-

Its mechanism / concept used to control concurrent execution of the transactions.

Lock is keyword in dbms to lock table for performing the transactions.

There are 2 types of locks.

1) Shared lock : LOCK-S(A) } representation
2) exclusive locks : LOCK-X(A)

transaction can only read data in this type of lock

transaction can read + write data in this type of lock.

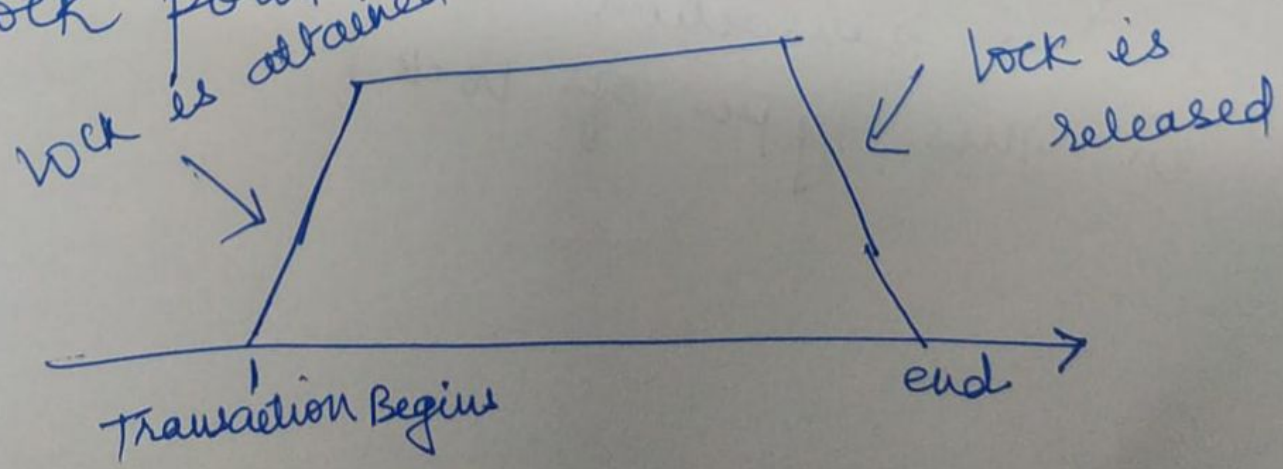
Two phase locking protocol (2PL)

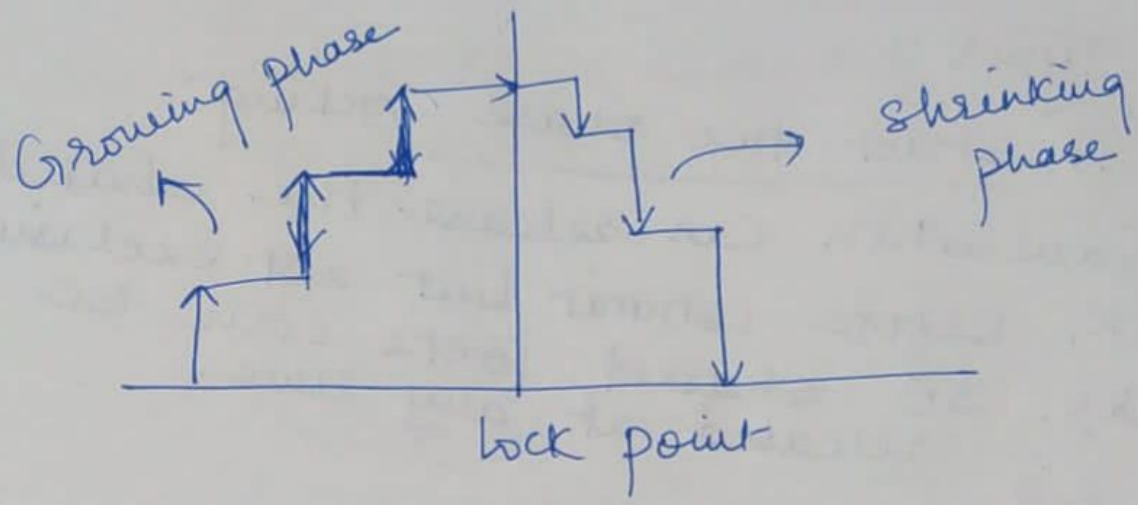
- Divides the execution phase of transaction into 2 phases:

1) Growing phase: The transaction is allowed to gain locks on data for the execution, but no one data is released / unlocked.

2) <u>Shrinking phase</u>: Transaction will start releasing locks that are held in 1st phase and no new locks are allowed to hold on any data.

3) <u>Lock point</u>: The point where growing phase ends is called as lock point.





e.g.

	T_1	T_2
1	Lock-S(A)	
2		Lock-S(A)
3	Lock-X(B)	
4	Unlock(A)	
5	Unlock(B)	

Shrinking phase

Transaction T_1:

growing phase: from step 1 to 3

Shrinking phase from 4 to 5

Types:

1) <u>Strict two phase locking</u>:

Transaction can release the shared lock before commit but not exclusive locks. so shared locks can be released at any time.

e.g.

T_1

S-lock(A)
read(A)
X-lock(B)
Unlock(A)
read(B)
write(B)
Commit
unlock(B)

release of shared lock on A before commit

release of exclusive lock on B after commit

Rigorous 2PL : Transaction can release all locks (shared & exclusive) after only commit;

T_1

S-lock(A)
read(A)
X-lock(B)
read(B)
write(B)
commit
unlock-B
unlock-A

release after commit

release after commit by trans.

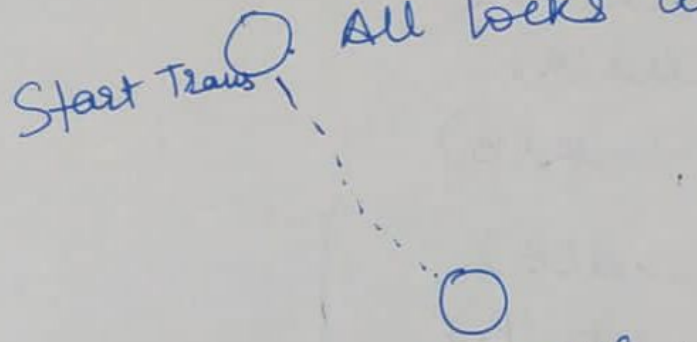
Conservative 2PL : In this protocol transaction ~~se~~ locks all items before transaction begin execution

✓ not used as its not feasible

lock all that you need before transaction starts

T_1 : A, B, C, D

Request lock on all



{ No growing phase in this type

waits till the items are locked before it starts execution.

If T_1 is holding locks on A, B, C and T_2, T_3 needs data then until T_1 commit or abort T_2 or T_3 can not proceed with the execution.

Q. DHF:

01/08/23

MODULE: 2

Derive horizontal fragmentation (5m)

Fragmentation

In which table is fragmented based on constraints on another table (owner table)

owner table → PK

member table → FK

1. Consider owner table = Pay
(title, salary) → attribute

2. member table = emp
(child)
↓
(eno., ename, title) attributes

3. H. Fragmentation of owner table / parent table:

- There are two groups of engineers that are divided as per salary ranges as sal > 30,000 and sal < 30,000.

3.

Fragments on Owner Table

$Pay_1 = \sigma sal > 30000$ & $Pay_2 = \sigma sal < 30000$

semijoin of

4. EMP_1 & EMP_2 are fragments on emp table with HF of owner table (member table)

fragments of owner table

EMP1 = EMP ⋉ Pay1

EMP2 = EMP ⋉ Pay2

2HF on Pay

→ semijoin

- Semijoin:

It is the type of join in which result will contain the columns from one of the join table.

5. O/P: DHF's from HF's of owner table