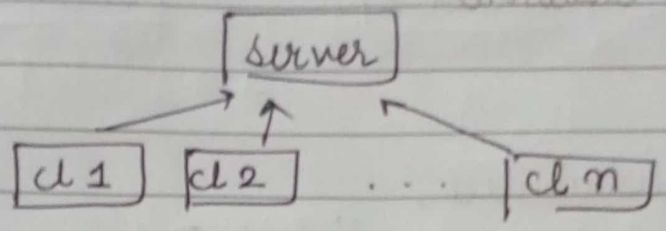
- Centralised database (Handles data at 1 server)
  - client
  - server

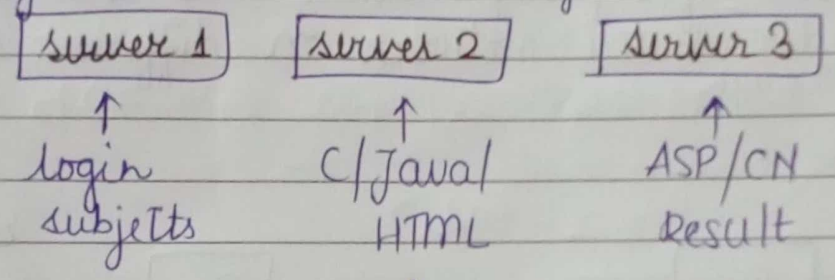
(+ has a backup server)
(If server fails backup server acts as centralised server like archeive)



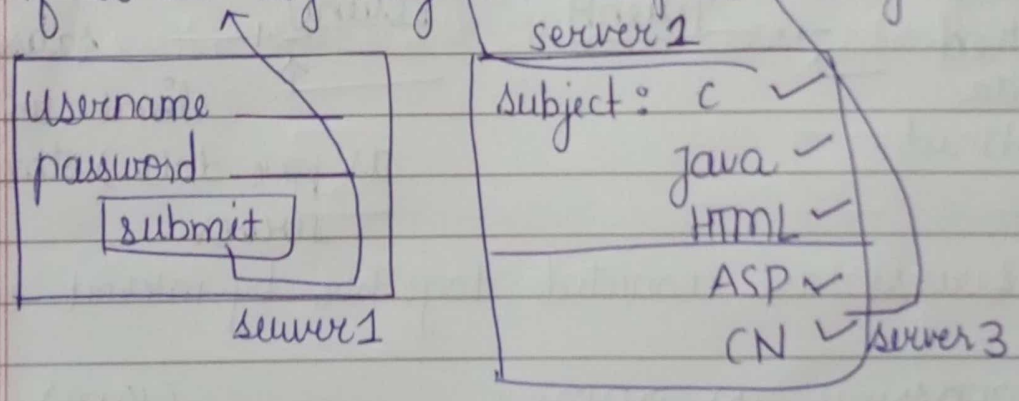
- Distributed database (divided servers for work load balancing)

eg: Online examination system →
1. login
2. subjects
3. questions
4. Result

C, Java, HTML, ASP, CN

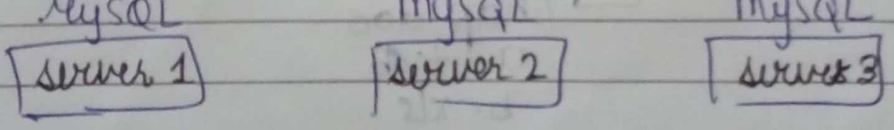


If we are loginning then compiler will go to server 1.



Imp: (Homogeneous & Hetroge) – Types of DDBMS.

If same database (DBMS) is used in all servers that is (eg: MySQL) then it homogeneous DBMS.



- If there is only one server, response time is more. Hence we use distributed database on multiple server.

The DDBMS in which there are different DBMS installed or different servers.

Mongodb

RANKA
DATE / /
PAGE

If there is no exchange of data betn the servers then we can use different DBMS called heterogenous DBMS.

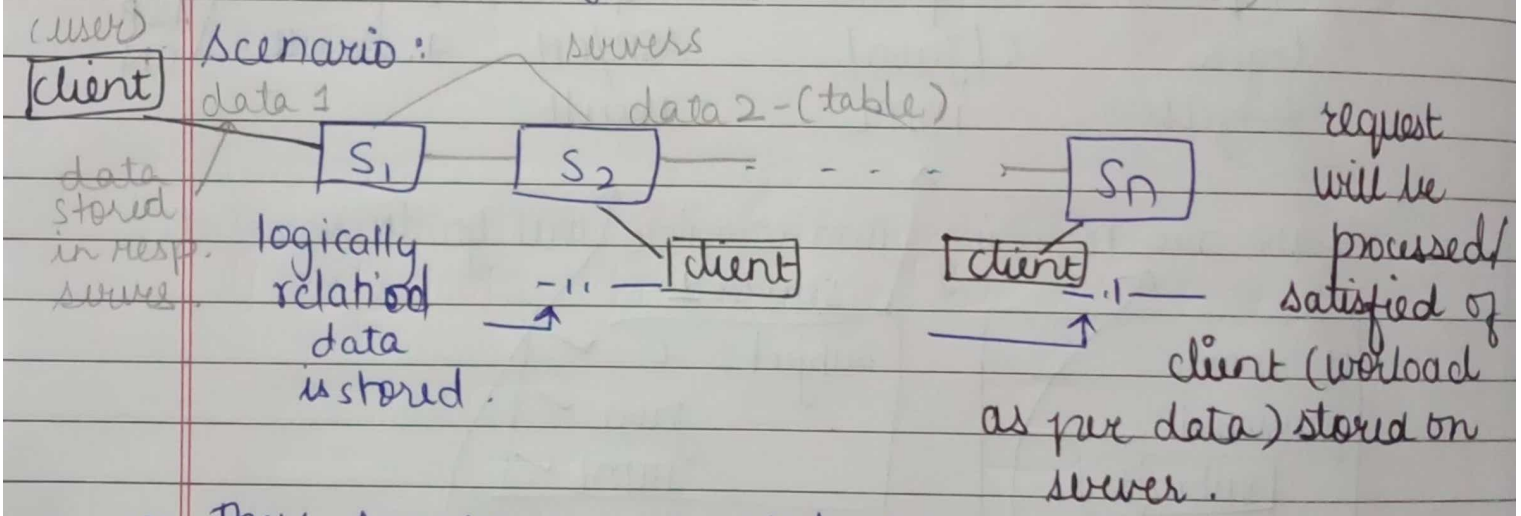
MySQL	Oracle	SQL server
server 1	server 2	server 3

- As database is distributed, processing, load will be distributed.

- DDBMS : Distributed DBMS. (2m)

It is the collection of logically related data that is shared physically through distributed network on different servers. (where servers are connected)

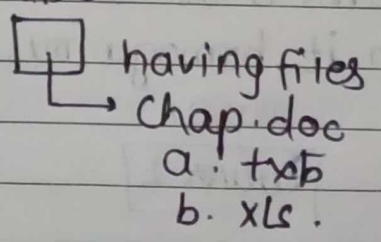
Scenario:



- These servers are connected together by internet etc.

* TRANSPERANCY IN DBMS: (10m)

Consider folder abc



- Copying all the folder is called - duplication
- — " — some part of the folder, some file - replication

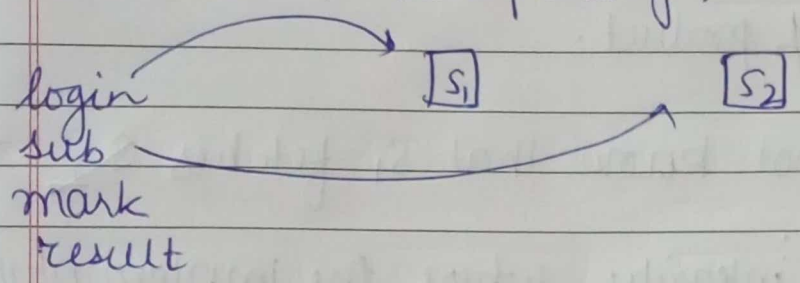
Explain eg

Explain eg.

- It is hiding implementation details from users of an application of an project.

Types:

1. Distribution transparancy:
User will not know where which data is stored called distributed transparancy.

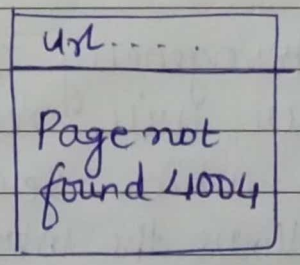


Here, user doesnot know where login is present in which server which is called distribution transparan

2. Failure transperancy:
Hiding the details of replacement system.
If one system fails then the user will not ~~the~~ about the replaced system.

This type of transparancy makes sure that system is continuosly working.

eg: webpage →

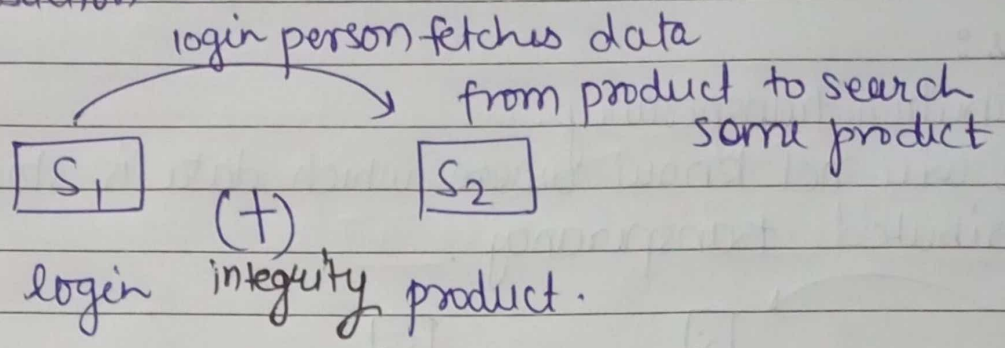


The ~~recovery~~ details of recovery server is hidden.
Hidding imp. details of recovering failded server.

3. Transaction transperancy:

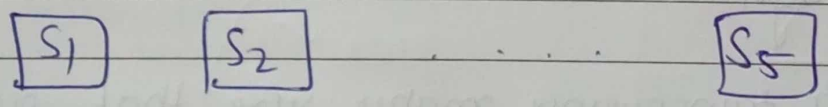
→ (no. of programs that will be executed by GUI's.)

Transaction →



The user will not know that S_1 fetches product from S_2.
Hence this is integrity where the implementation details of transaction are not known to the user.

4. Performance transperancy:
The system looks like centralized DBMS.
It only looks like centralized but not there.



HDFC bank

5. Heterogeneity transperancy:
User will not know which data is stored in which server.
It allows the integrity of several different local DBMS under common schema.

6. Replication
Copying of some data on the servers.
~~We donot know~~ User doesn't know which data is copied where.

7. Fragementation Transperancy:
Division of data on no. of servers.
Fragementation types:
1. Horizontal
2. Vertical.

1. Horizontal : There are more than 1 horizontal fragments depending on conditions.

Student.

Rollno.	Name	Add.	DIV
1			A
2			.
3			.
.			A
.			B
.			A.

DIV A.

Serv. 1	Serv 2

Division of data on logicall inp. div-A.

select * from stu.
where DIV = A.

only divA will be selected and placed in server A.

2. Vertical :
Division of data based on columns/attribute.

Server 1

Rollno.	Name

Server 2

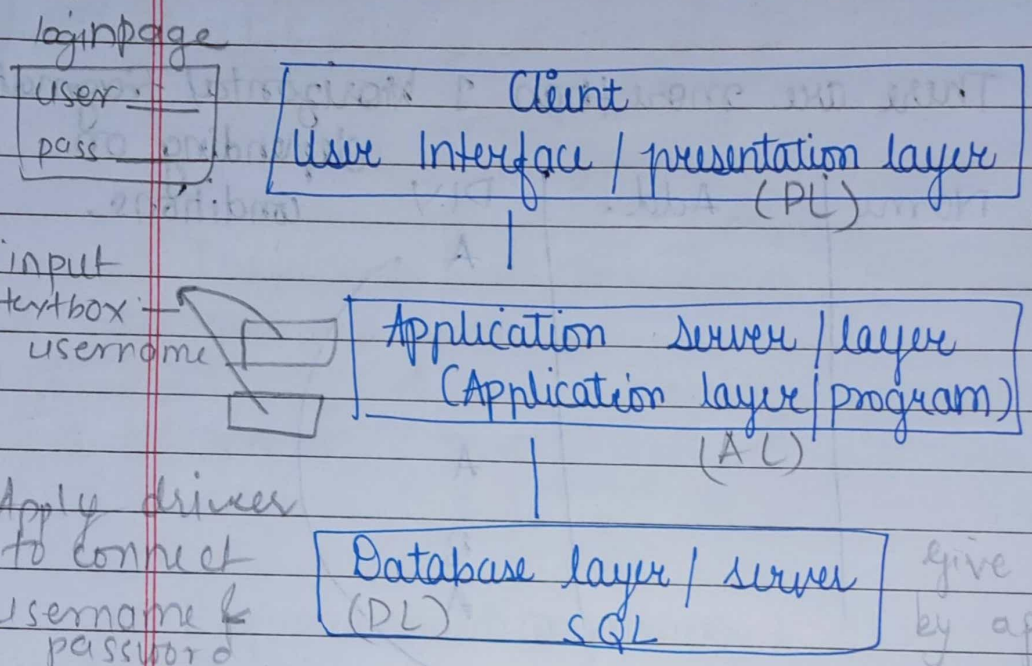
Add	DIV

This type of transperancy is which where user doesnt know where fragments are created on table and where they are stored.

Q 3Tier Arch.

This type of transperancy is which where user doesn't know where fragments are created on table and where they are stored.

• 3 tier client server architecture:



1. Client
2. Server
3. Dia
4. eg. of each layer

Output is generation of presentation by app. layer &

Application layer takes input from presentation layer.

(Relation, exp., eg., app of each layer & explain client & server) (10M)

- Data replication & allocation : (5M)

REPLICATION:

→ It is a technique of copying of some part of data or fragments.

ALLOCATION:

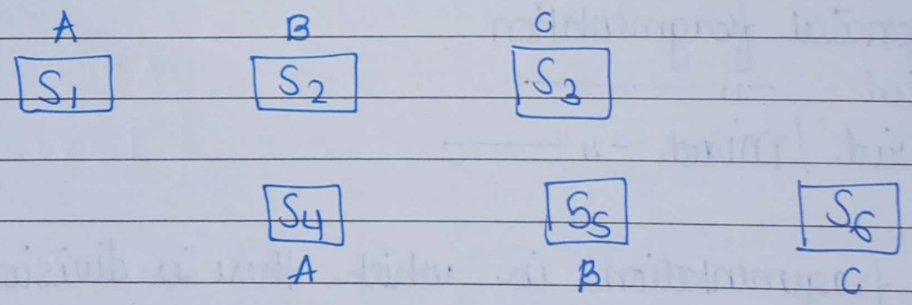
→ It is the way of storing fragments or replicas in server.

- Fully replicated distributed database:

It is db is which there will copy of each data in different servers.

This is the most extreme option in which replication of whole database will at every server in case of distributed system (distributed database).

Assume A, B, C as fragments that are stored on S_1, S_2 & S_3 resp., then in fully distributed database A, B, C will have a replication S_4, S_5, S_6.



- No replication:

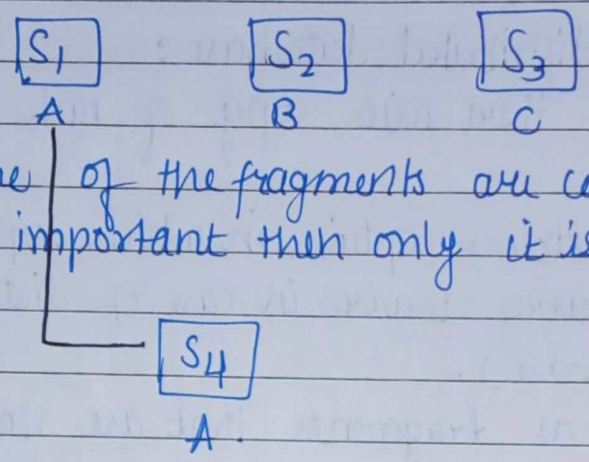
In this case, there will no copies of the fragements on the available servers.

These fragements will not have copies in another servers.

Above case there were copies of A, B, C in S_4, S_5 & S_6 but in this case there will be no copies in S_4, S_5 & S_6.

- **Partial replication:**
In this type of data allocation there will be copying / replication of some parts of ~~st~~ fragments.
In this case important data will be copied on servers.

Assume A, B, C on S_1, S_2, S_3.

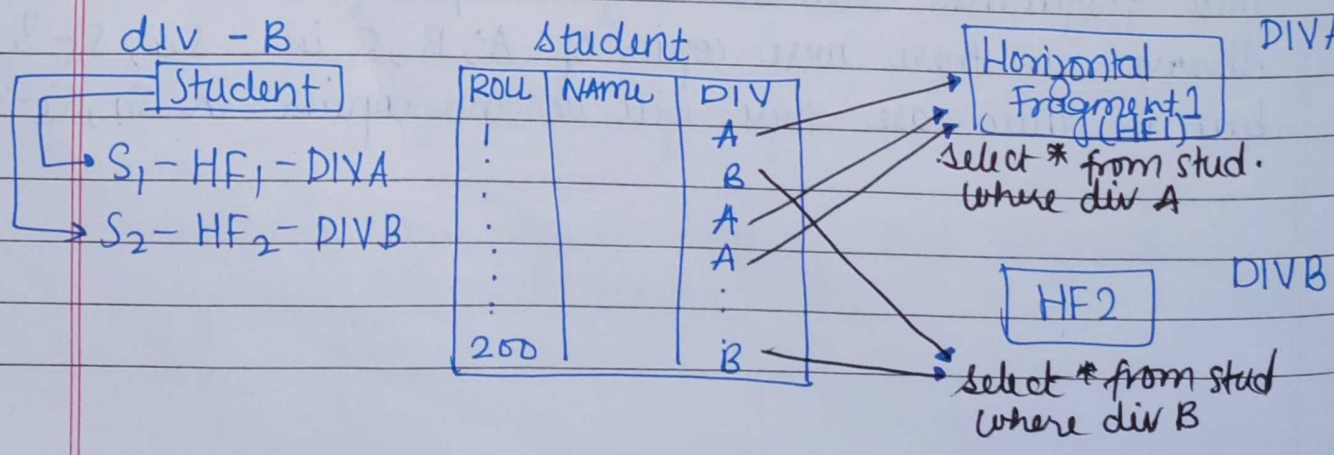


Only some of the fragments are copied.
If A is important then only it is copied.

IMP. **Fragementation in DDBMS:** (10m)
1. Horizonatal fragmentation
2. Vertical —"—
4. Hybrid / Mixed —"—

1. This is fragementation in which there is division / distribution of tables on rows / records horizontally by applying some condition of the attributes.

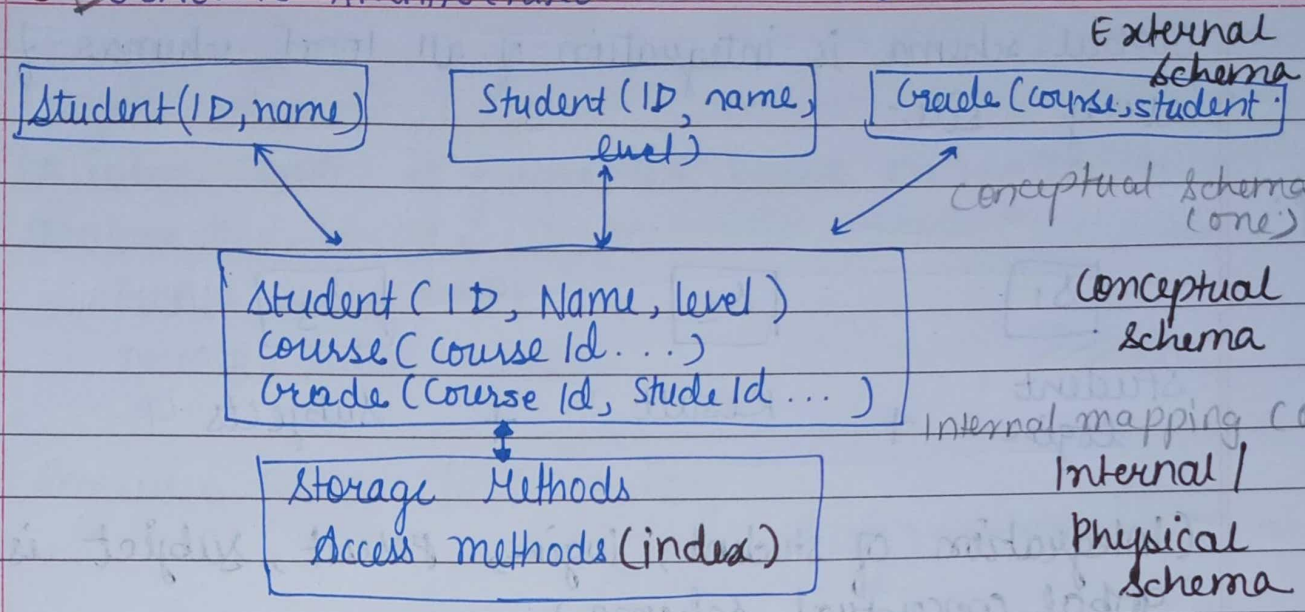
Representation:
Condition on division attribute there are two fragementation
div - A
div - B



10:30am

Friday
28/07/2023

GENERAL ARCHITECTURE OF DDBMS:



eg:

DBMS
Student_table
- Roll no.
- name
- add
- email.Id

} sequence should be same

GUI
Roll no.
name
email.Id

there is no need for all attributes from DBMS:
add in DBMS will have 'NULL value'
if required 'add' is added

- Conceptual schema types:
  - Global schema:
    It is the general architecture of distributed DBMS or a global logical veiw of data / project / app.
  - Local conceptual schema:
    It is the logical data representation at every server / site.

- Global schema is integration of all local schemas from every site.

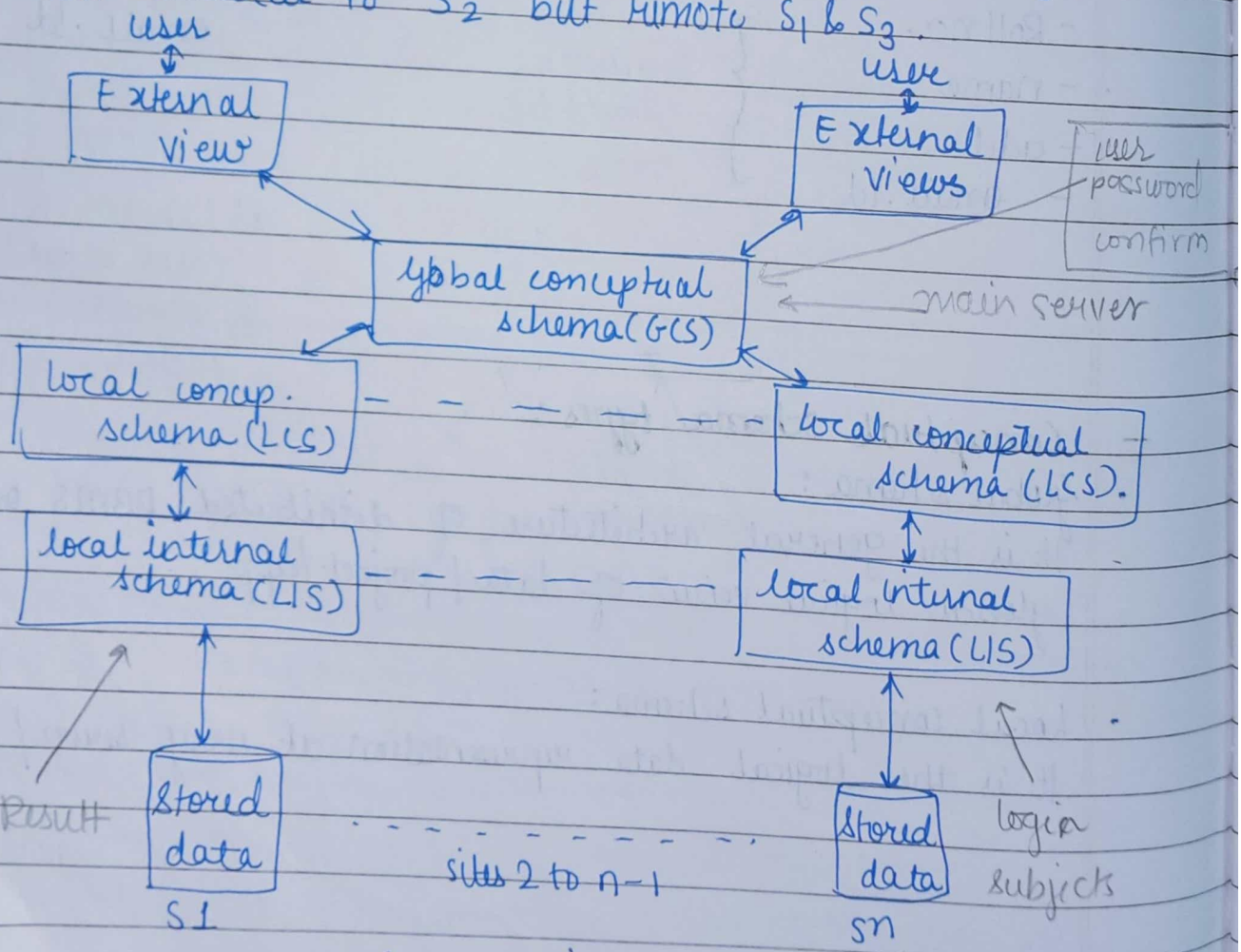
eg:

S_1 — student 1, login 2 + S_2 — Result 3 + S_3 — Subjects 4

(Integration of student, login, result, subject is global conceptual schema).

[Global conceptual schema - student + login + result + subject]

- login is local host to S1 but remote to S_2 & S_3.
  Result is local to S_2 but remote S_1 & S_3.



Logical architecture model

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

2 HF 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.

View & Fragments:
(It is virtual network / table) (doesn't store data)
Syntax:
select * from viewname;

~~doesn't~~ stores data

—— × ——

* Transaction:
Properties: ACID — Durability
Atomicity consistency Isolation

Types of transaction:
1. Online (Performed on daily basis in short time) eg: Gpay, ecommerce site
2. Batch
3. Flat
4. Nested

1. Online Transaction:
In online transaction they are performed in a very short time by accessing the website or apps. (Performed on daily basis)
eg: Gpay, netbanking, ecommerce site

2. Batch Transaction:
In this, transactions take longer time to execute.
eg: Report generation, credit card (monthly bills of customers → with dates will be shown)
(bills will be generated in a batch)

3. Flat transaction:
This transaction has one begin and one end that is only one ~~trasho~~ transaction at a time.
eg: bank (verification by OTP)

Q Discuss issues in case of distributed concurrency control.

Q Explain 2 phase locking protocol with types. (10 m)

4. Nested Transaction :

Nested is opposite of flat transaction. Hence there will be more than one begin & one end. More than one transaction at a time is performed.

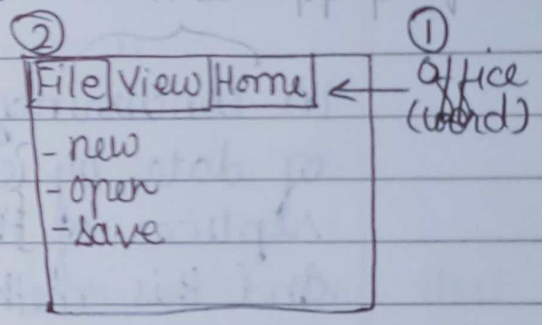
Syntax :

```
Begin Transactions
  Begin
    .
    .
    .
  end
end
```

- No. of transaction will be included to satisfy the requirement of user & the client.

eg : word document

(New → open → save)



* CONCURRENCY CONTROL :

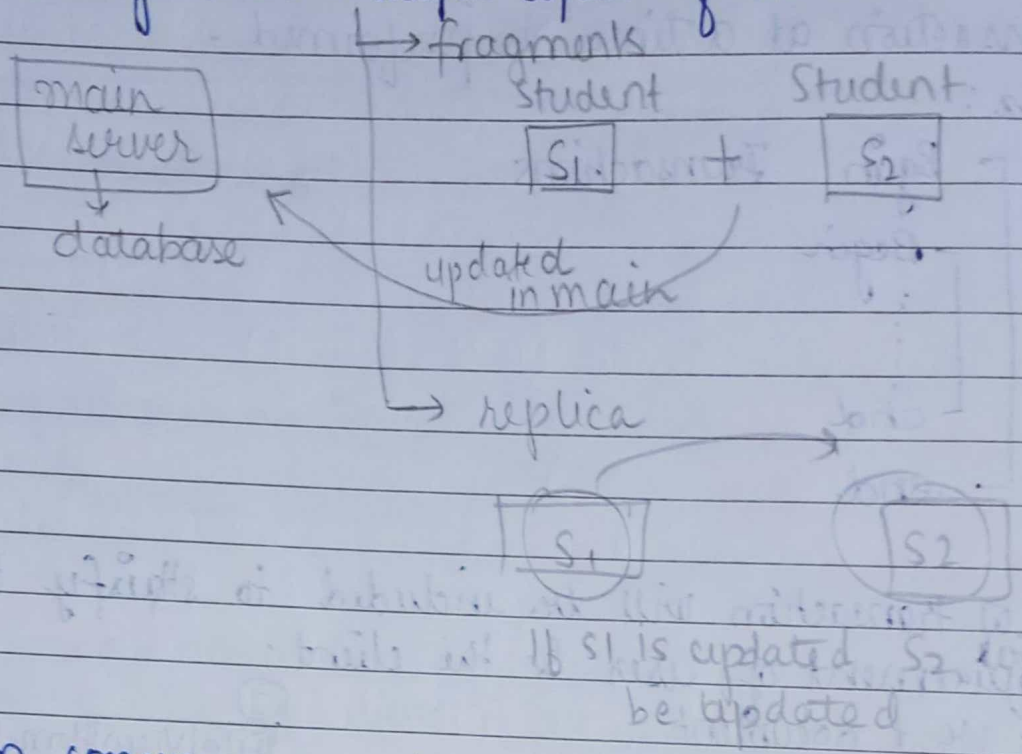
When no. of users are increased for same project / application. Concurrency control is mechanism to make the application consistent.

eg: explain plan for select * from a join b on a.id = b.id ; } inner join.

Vertical = rows = same column = join
Horizontal = column = same
row = union

Q. Discuss issue in case of distributed concurrency control.

1. → Dealing with multiple copies of data:



In concurrency control will deal with multiple copies of data by checking with whether it is fragments or replicas of the tables.
And this method ensurs that updations are done at every copies of replica as well as in main database.

2. → Failure of individual sites:

In failure of fragments,
it will be switched to main server.

In case of replicas,
then it is switched to next working replica.
If R_1 fails
R_2 is activated.

In case of failure of individual sites the distributed dbms will manage continuous execution of the project with the help of replicas or main database.

3. → Failure of communication links:

In case of failure of communication links it should deal with failure - by using network partioning. (Alternate options will be used in failure of comm. link of networks).

Network issue

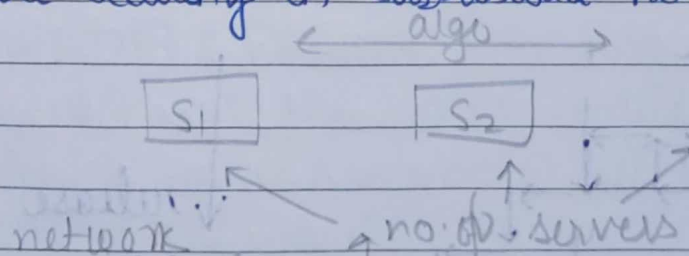
4. → Distributed commit:

Distributed commit will be followed to make sure that each and every copy of fragments or replicas will be updated with current status of data.

* If fail → rollback
* If not → process ahead.

5. → Distributed deadlock:

Concurrency control should be able to solve the deadlocks that are occuring in distributed network of deadlock.



04/08/2023 Q - 2PL & types

RANKA DATE / / PAGE

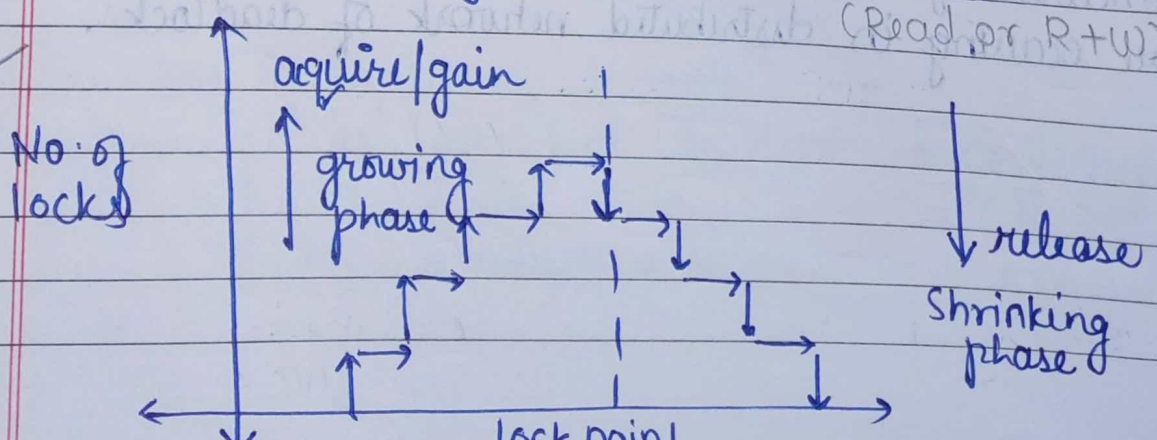
* 2 phase locking protocol & its types:

- A lock is process / mechanism to deal with concurrent execution or parallel execution of data.
  It uses locking concept:
- Types of lock:
  1. Shared lock (Read only) (S) (read) (options) ways of accessing data or table.
     1. read (R)
     2. write (W)
  2. Exclusive lock (X) (Read + Write)

There are types of lock that are shared & exclusive that means data can be accessed in 2 ways for (reading) and (reading + writting) respectively.

- Two phase locking protocol:
  It is way of controlling concurrent transaction.
  ↳ 2 Phase:
  1. Growing phase:
     The phase in which transactions are allowed to obtain or gain the locks.
  2. Shrinking phase:
     Transactions will released of the locks. (unlock)

- LOCK POINT:
  The point where growing phase stops. No transactions are allowed to gain / obtain a lock. (Read or R+W)



2 Phase lock:

eg:

Sr no.	T_1	T_2
1	Lock S(A)	
2		lock S(A)
3	Lock X(B)	
4		
5	Unlock (A)	
6		lock X(C)
7	Unlock (B)	
8		Unlock (A)

growing phase for T_1

lock point

shrinking phase for T_1

Shared lock (Reading only)

Exclusive lock (Read + Write)

growing phase for T_2

lock point

shrinking for T

lock type (data)

For T_1: Consider transaction T_1;

T_1 is accessing data in shared lock A.

Srno. (1,2,3) is growing phase of T_1.

For T_1, lock point is at sr. no 3.

Sr. no. (5,6,7) is a shrinking phase for T_1.

For T_2:

Sr. no. (2 to 6) is growing phase for T_2.

Sr. no. 6 is lock point.

Sr. no. 8 is shrinking phase/point.

- Types of 2PL: Types (explain with example)

1. ~~Straight 2PL~~ Strict 2PL:
2. Rigorous 2PL:
3. Conservative 2PL.

1. Strict 2PL: Exclusive & Reading.

It holds all the locks until the commit point and releases (unlock/uncommit) all locks at a time.

eg: (exclusive locks)

Read lock can be released before commit. S(A)

Read & write lock only is realiased after commit. X(A)

eg:

	T_1	T_2	T_3
1.	lock X(A)		
2.	Read (A)		
3.	lock S(B)		
4.	Read(B)		
5.	Write(A)		
6.	Unlock (A)	lock X(A)	
7.	Unlock(B)	Read (A)	
8.		Write (A)	
9.		Unlock (A)	lock S(A)
10.			Read (A)
11.			Unlock (A)

commit (for rigorous) → 6.

A = exclusive
B = shared

eg: Strict PL:

lock S(A)
⋮
lock X(A)
unlock (A)
commit
unlock (B)

reading is released before commit

only after commit

2. Rigorous PL:

In this type of 2PL are locks are held (shared + exclusive) till commit. That means if transaction is holding locks in shared and exclusive mode then lock will not be released till the moment transaction is not writing commit command.

eg: T_1

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

The above T_1 is released shared & exclusive locks after writing commit command.

(read) & (read + write locks.)

Q Compare two PL & rigorous & strict 2PL with eg.
Q Explain timestamp ordering protocol for concurrency control method (10m)

3. Conservative 2PL:
In this, all locks that Transaction will need will be aquired before start of execution. There will be starvation for other transaction.

eg:
T_1
S-lock(A)
read(A)
X-lock (B)
read (B)
write (B)
commit.

predict locks or no. of tables & will hold till commit.
Therefore, there is starvation for next transaction.

The lock will not be released ~~unit~~ until required.

* Types of failure in distributed dbms:

1. Transaction failure:
This is due to
- deadlock
- unavailability of system resources (data, resources)
- logical errors.

This failure is happening due to the application not going ahead.

2. System crash:
Reason
- no power supply
- network traffic

3. Disk Failure:
- bad sector (corrupted)
- unavailability of disk.

11/08/2023

- TIMESTAMP ORDERING PROTOCOL: → If there is conflict

It is time associated with transaction (TS).
Time can be coded starting with '01' or systime.

This is represented as TS (transaction).

$$TS(T_1) < TS(T_2) < TS(T_3) < \ldots\ldots TS(T_n)$$

Rule 1: If transaction T_i issues / performs Read (X) operation:
(newer) ↑ ith transaction

a) If W_TS(X) > TS(T_i) (older) → this means TS(T_i) is a older transaction but this is not true.
(write/updation) (performing transaction)

It is a timestamp of transaction when updation is done by that transaction at a particular time

This operation will rejected that is there will be rejection of read operation issued by transaction.

b) If W_TS(X) ≤ TS(T_i)
~~(newer~~ (older) newer (This is to be performed so it should be new)

This will be accepted.
There will be acceptance of read / ~~write~~ operation issued by Transaction.

Rule 2: If transaction T_i issues write (X) operation:
9:00am 9:03am
older

a) If TS(T_i) < R_TS(X) → already read so no updation to the old transaction.
then operation is rejected or not updated.
10:40am W → (last write to X) 11:00am

b) If TS(T_i) <> R_TS(X)
then operation accepted.