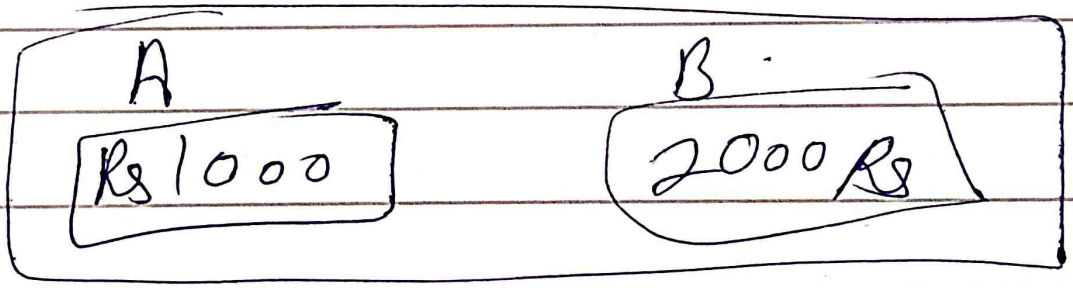
TATA CONSULTANCY SERVICES

Dec-12

# Acid properties && transactions in DBMS

→ Transaction :- A unit of work done against DB in a logical sequence.

Eg :- Task 1 :- To transfer 50Rs from A Bank to B Bank.




Sol :- Steps

Read(A)
A := A - 50;
Write(A) = left 950 Rs.
Read(B)
B :- = B + 50; - Balance = 2050
Write(B);

These all steps should be atomic i.e they are considered to be go in a single task.

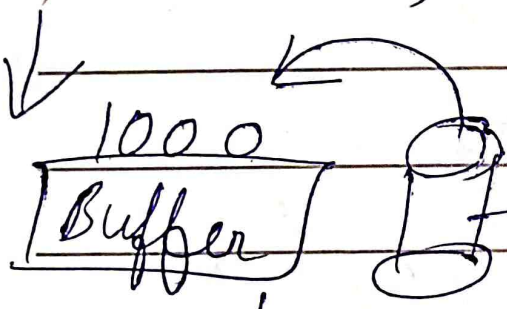
TCS TATA CONSULTANCY SERVICES

Work Question

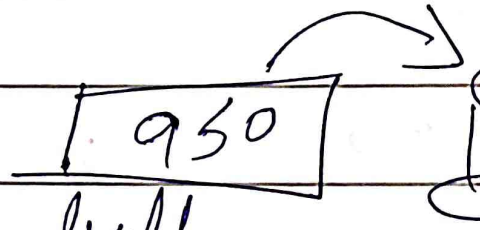
* ACID Properties :- To ensure integrity constraints, if any transaction happen in db then it have to follow ACID Props.

→ ~~Atomicity~~

1) Read (A) ; means it will read bank balance from db & store it in local buffer.




2) Write (A).




read (A)
A' = A - 50.
write (A)

buffer
it will then after transaction return from local buffer to actual db.

1) Atomicity :- if transaction has done successfully in one go → means that all the 6 six steps of transaction will done in one go & then go into db, will call it as successful, if any one transaction fails then will call as failed or rollback happen.

| ⇒ A | B |
|---|---|
| read operation 1000 | read-2000 |
| 950 ← 1000 - 50 | → System crashes |
| Write (A) 950 | Read 2000 2000+50 |
| Rollback 950 stage. | Write (2050) transaction successful |

if after writing (A) / or reducing 50 Rs from acc (A) suppose system crashes & it doesn't add into account (B) then this 50 Rs might get loss so at this stage atomicity is required.

TATA CONSULTANCY SERVICES

→ transaction → successful.
→ fail → (it must be reverted back into its old recovery state).

- DB → Maintain → Old state.
↓
Success not happen
↓
Rollback (old state).

2) Consistency :- DB must be consistent after transaction happens that means if acc A has 1000rs & B has 1000rs then after & before transaction money should be consistent.

(3) Isolation :- Agr suppose kare ke hamne 2 transaction ek tab upi se aur ek debit card se kiya same time pe toh yeah dono transaction ek dusre se isolated rahe aur

TATA CONSULTANCY SERVICES

Concurrently aur sequentially process ho yeah humari db ki responsibility hogi.

→ Banking System :-

Error :- $T_1$ $T_2$ $T_3$ $T_4$

$T_1$ (UPI)
read (A) → 1000
A :- 1000 - 50 = 950
A = 950
Write = 950

$T_2$ (NetBanking)
→ Read (A) → 1000
iss time pe $T_2$ ne bhi 1000 rs read kr liye kyuki $T_1$ ne phele 2 seconds phele execute kiya tha toh usne 950 rs show kr diye lekin $T_2$ ne bhi 1000 rs read kiye h aur 50 rs deduct so left balance in $T_2$ 950 rs.
write (950)

→ So this error comes if data is not concurrent & isolated and if we have to transfer 50 rs from A & B to B then due to both $T_1$ & $T_2$ running

transaction process in A account it will deduct 50 from A & 50 from A-T2 & transfer 50+50 = 100rs into acc B & will show 950 in A account so its not isolated.

Until T1 starts & get completed, it will not start T2..

Multiple transactions happen in system in isolation without interfering each other.

---

* Durability :- after transaction complete successfully. so the change made to db must be there as it is.

T1 → success

(All steps completed) → Success.

↓

so db updates permanent (persist)

↓

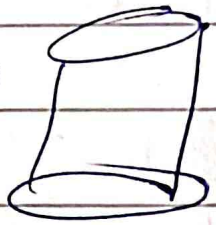
even if there is system failure after transaction T completes.

To recover durability, Recovery manager has this responsibility

→ Recovery Manager component

Main Memory
A = 950
B = 2050

write → system fails



at this time 50rs is debited from A & also credited into B, transaction done, But in db when its going to write in db then system crashes.

& if system fails after restarting it the thing written in main memory will get flushed but there will be any system who recovers or print log at every step.

TATA CONSULTANCY SERVICES

__/__/____

* state of Transaction :-
Whenever transaction start, it comes / starts with Active state, at this point read or write operations happen.

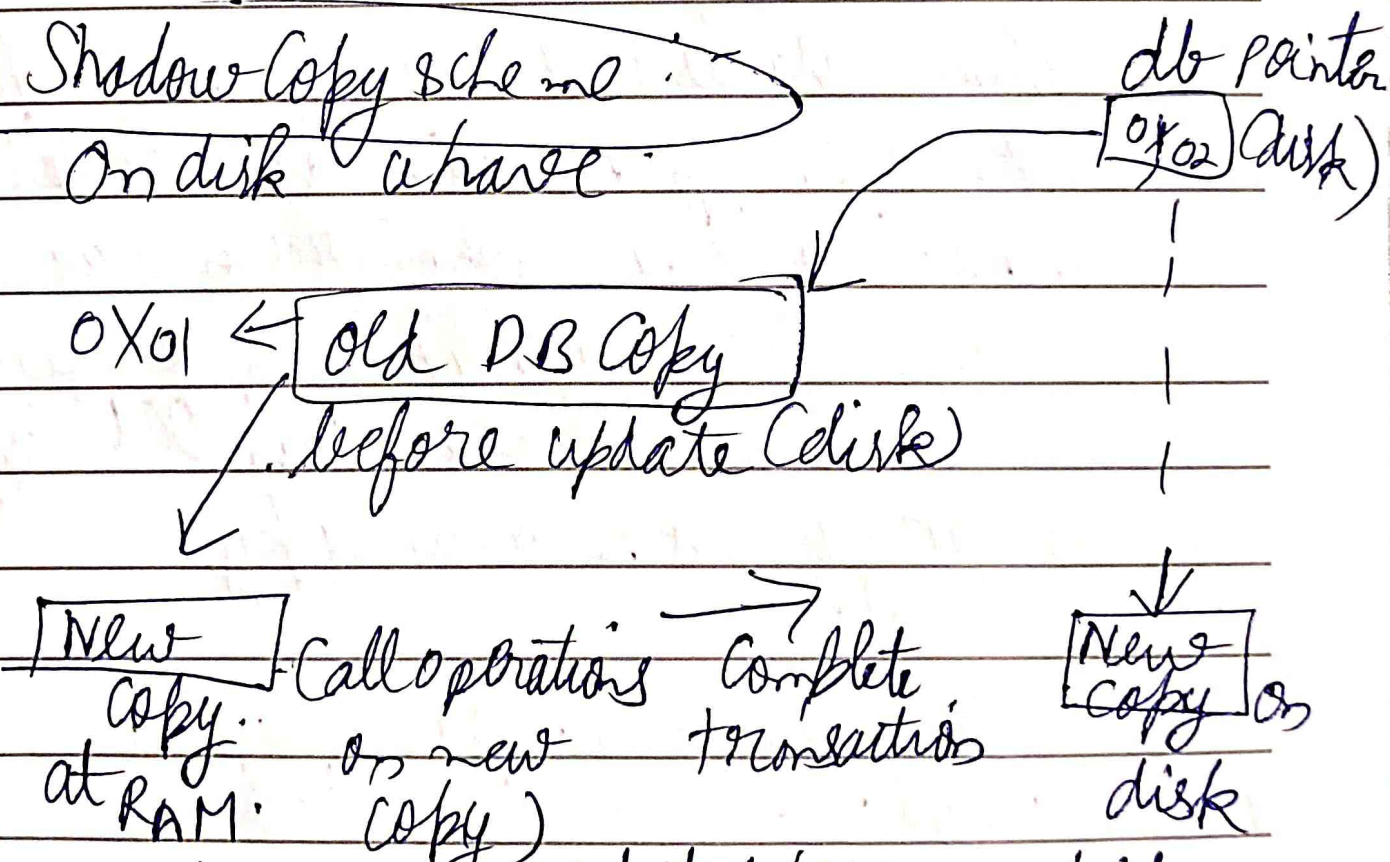
---

Dec-13.

Recovery Management System support durability & atomicity in DBMS

* How to implement atomicity & Durability?

① Shadow Copy scheme :
On disk change.




When db pointer updated to new disk location then notify transaction commit.

tb transaction fully complete mani jayegi jb jo db pointer h vo old db copy se (0X01) jo wki id h usse unlocate hoke new copy of disk (0X02) id h usse locate karega tbhi hum user ko notify karege that yes transaction comitted.

aur agr abort krte h to tb just delete new copy of db from RAM.

agr system restarts kre, Rb jb new copy disk pe top pr gyi h lekin db pointer ke locate ke time pe hi system crash hua, aur db pointer pe 0X02 nhi likha gya, fir system restart ko bad bhi hum old copy disk ki dekh payege na ke new copy.

aur agr system fail kiya after db pointer has updated, then it will read new db copy only.

yeah ~~recovery~~ shadow copy scheme atomicity & durability dono ko reflect kr leki yeah inefficient technique h bcoz every time it will make copies of large bytes.

So to make more efficient scheme **log Based Recovery System comes**:-

Logs (Stable storage)
<T0 start A>
<T0, A, 950>
<T0, B, 2050>
<T0, Commit>

~~Logs~~ logs have to be written before actual performance

⇒ 1) Deferred db modification
(देरी से)
isme hum logs को likh lete h pure for execute krte h.
if system crashed before T completes or if T is aborted, the info in logs ignored.
⇒ Commit krte krte fail hua system to phir hum redo kr skte h mtlb

logs ko read karke commit karte hain.

② Immediate db modifications :-
→ Pehle logs likhege aur uske sath hi saath db bhi change.

<To start> → old value field
new value
<T0, A, 1000, 950>
<T0, B, 2000, 2050>
<T0 commit>

Jb crash hoga, system pura old value hi likh dega toh atomicity maintain hogi.

if txn completes, & system fail hua toh redo used to commit logs in logs.

→ ab logs based recovery ka ek concept
itna sare logs ki storage bahut
zada padegi toh isliye hum

TATA CONSULTANCY SERVICES

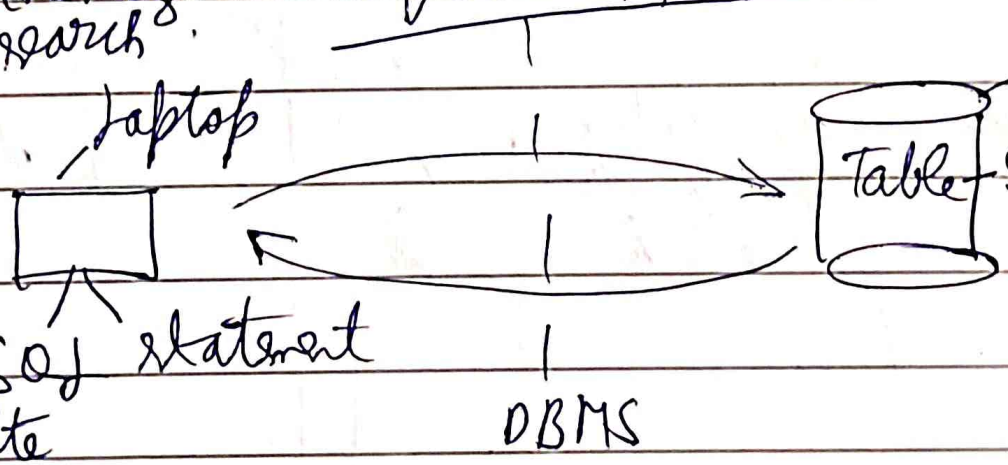
- Checkpoints system Use for Skill b.
- Checkpoints.

C1 → ↑ ETxn → Commit

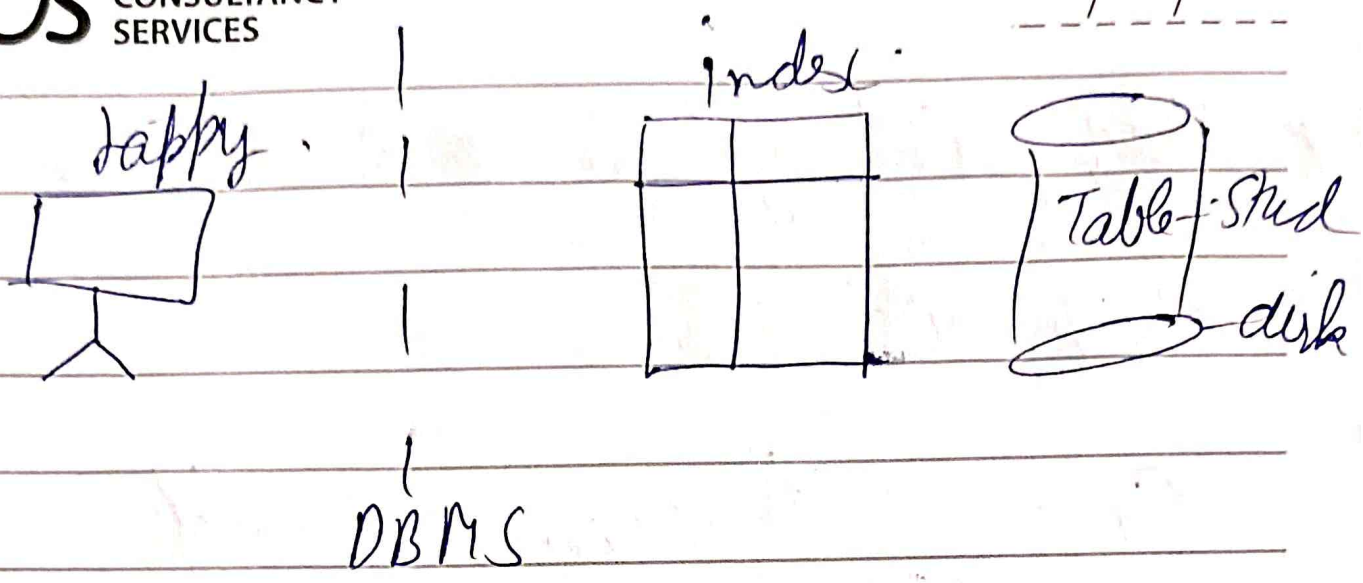
C2 → ETxn — fail → Redo.

Used to optimize performance by minimize no of disk search.

Dec -14 · INDEXING

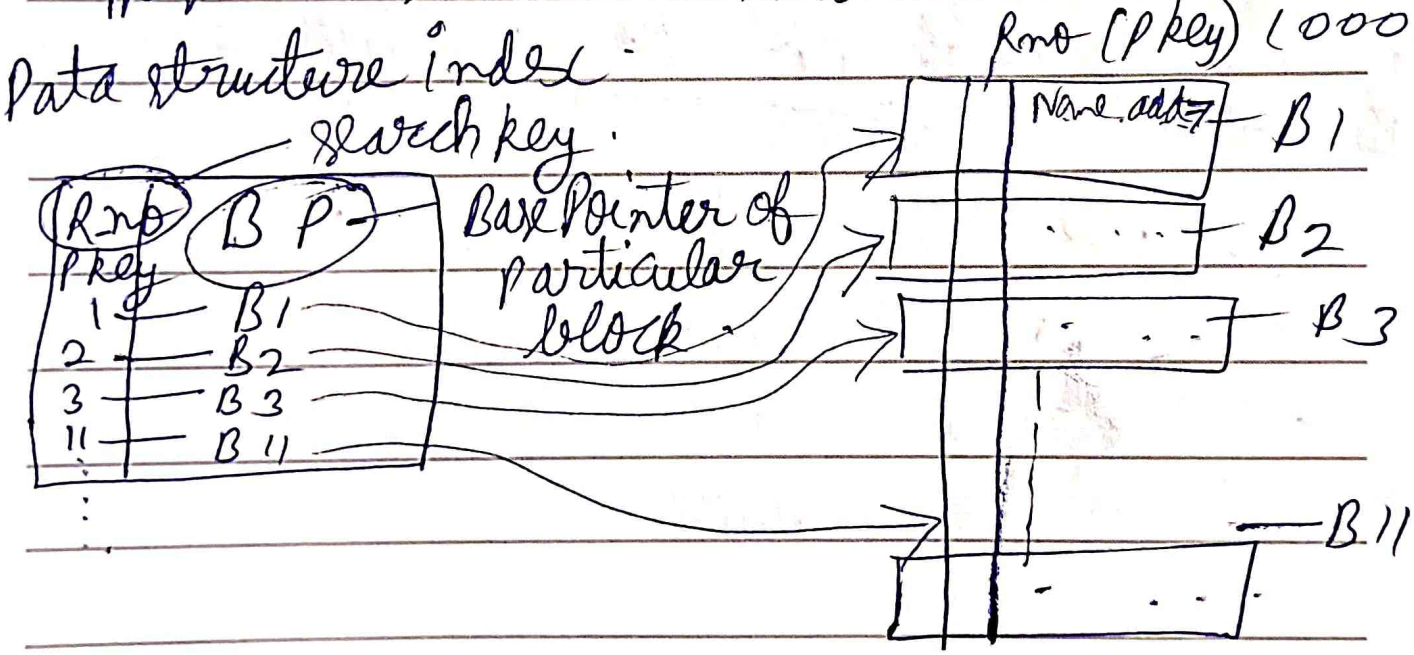



if we go like ↑ above this & try to fetch data simply then it will make process slower because there is file in a disk & search take lot of time so to avoid it we introduce "indexing".




→ in disk (data file in disk 10,000 records store)
Suppose - 1 block → 10 record → Total blocks

Data structure index:




If we have to fetch info of n students in 10,000 records then data structure index is used if we find try to find any base pointer of any sorted

block i.e here $B_1$, $B_2$ are blocks & in that blocks roll-no are sorted so then by getting block we can apply binary search.

Example

for ex:- from previous image; we find out-843 stud record so if we have to try to apply linear/Binary search directly on table then it will take time, rather we can apply Binary search on small sorted index table. (Index file is always small)

| Rno | BP. |
|---|---|
| 1 | B1 — there are block |
| 2 | B2 |
| 3 | B3 |
| ⋮ | ⋮ |
| 9999 | B9999 |

so to find 843 stud record steps are:—

① in 10,000 search put binary search on it and find that what is the BP of 843 record — we find it as

843 record in index file have BP843 and then we go in data file in disk and find 842 → 843 record by applying any linear / Binary search.

* Types of Indexing : also known as (clustering index)

1) Primary indexing :- Only apply that place when data file is sorted in sequential order like on one key for eg roll_no in stud_table not on age student ; if we try to search on stud_age column then roll_no will get disturbed they become non-sorted so always sort on one key.

Imp NOTE :- Its not true that primary index always used on primary key no its not true. it may be primary key but there is no restriction, we can keep non-primary key also.

TATA CONSULTANCY SERVICES

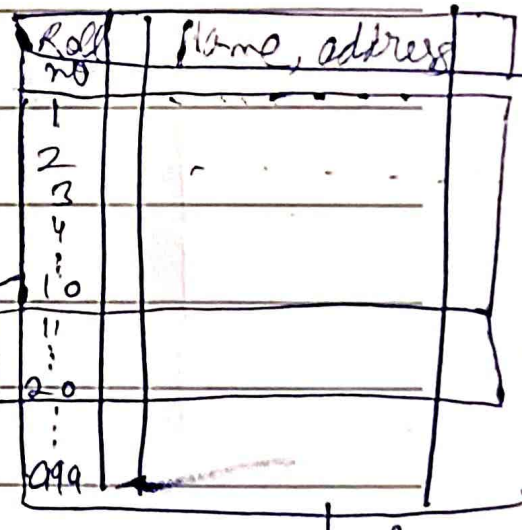
These all comes under primary indexing only.

(i) Dense indexing :- When all the search keys of a disk file present in our index file then its known as dense indexing.

(ii) Sparse indexing - index record appear for only some of search-key values like if there are 10 blocks in disk then from 1 block we keep B1 or 1 roll_no index and then from 2 block B11 roll_no " ".

↳ Sparse indexing always done on primary key / Candidate key if disk is sorted like for ex :-

| Roll_no | Base Pointer |
|---|---|
| B1 | 1 |
| B2 | 11 |
| B3 | 21 |
| B4 | 31 |
| ⋮ | ⋮ |

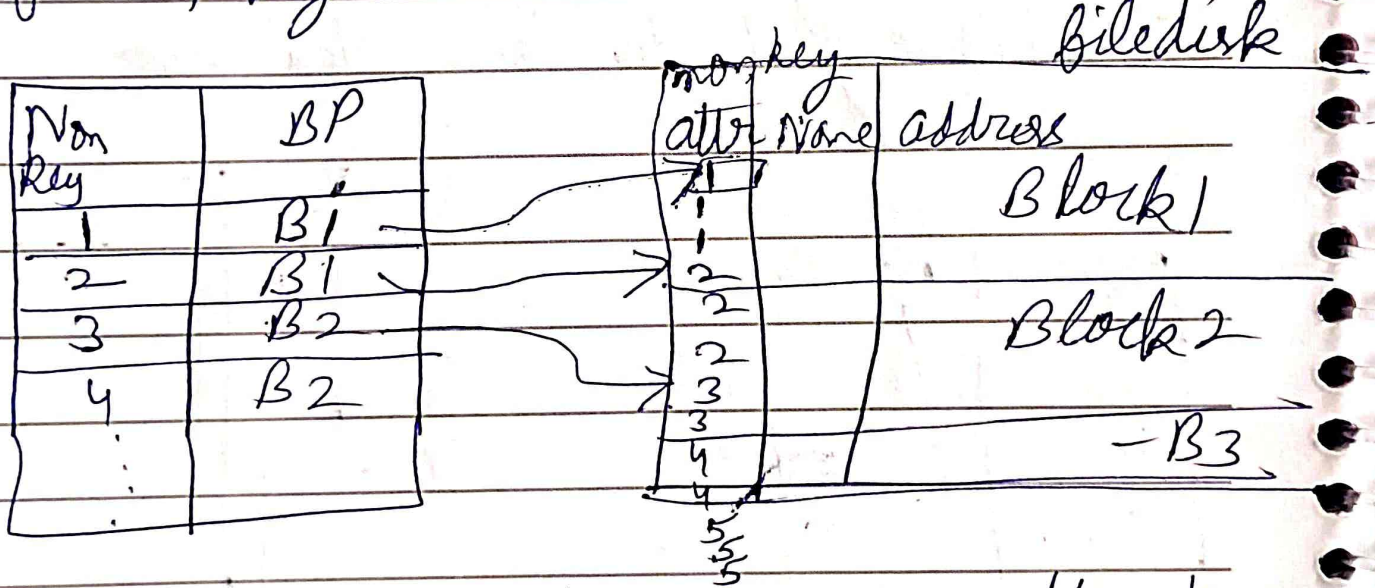



So if in disk pkey / CK is sorted then on indexing we can make block B1 (1-10) & apply sparse indexing.

disk.
B1 (1-10) B2 (11-20)
B3 (21-30) .....

Primary indexing had one notice that based on one key the file stored & used search indexing on it. ---

→ Based on non-key attribute :- (its example are where we use group by clause like if there is dept & it is non-key attri (non primary) so considering dept groupby clause we can find dept wise salary, their avg ages of dept wise etc. So clustering index is also part of primary indexing only & in cluster we sort on basis of non-key attr such as below :-




So u can see that in non-key attr, there are blocks - in B1 block attri 1, 1, 1, 2 and 2 also in block 2, so in cluster index main yeah Bol sakte hu ke index me non-key ka jo phele values jaise iss image

me agr hum dekhe toh [non key me 1 B1 me aa raha h sbse phele] [2 jo value h vo sbse phele hume block 1 me hi dikh rhi h], [3 ki value hume sbse phele block 3 me dikh rhi h], toh hum bolege ke non-key attr ki sorting index me hum Base pointer B1 pe binary search lga ke sari [1 non-key] ki value nikal lege. aise hi, phele jo 2 ki value jis block me dekhi h use block se lekar ke dusre block me jaha tk 2 ki value h vaha tk binary search lga ke 2 ki value search kr lge. toh isse clustering index bolte h, h yeah bhi primary index ka part hi h.

Imp

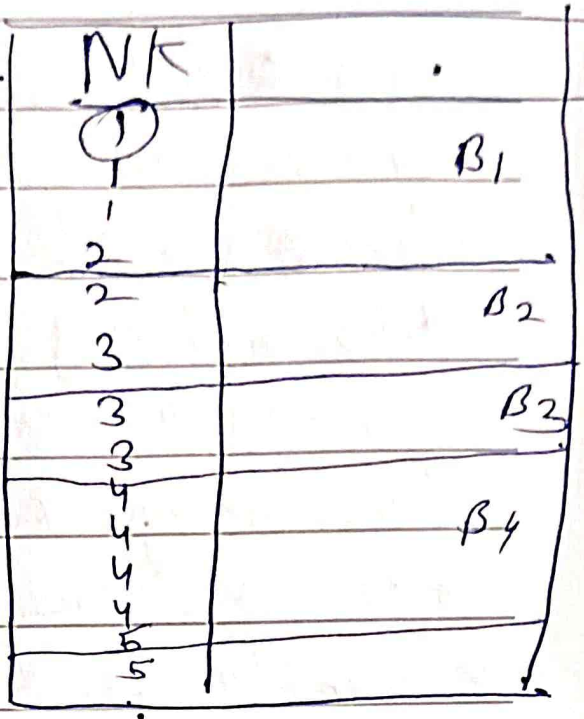
- jb hum primary key / candidate key ke hisab se indexing krte h tb vo sparse indexing hoti h.
  lekin clustering indexing → dense index pe hoti h.

TATA CONSULTANCY SERVICES

# Dense index

Index

| Nonkey | BP |
|---|---|
| 1 | B1 |
| 2 | B1 |
| 3 | B2 |
| 4 | B4 |
| 5 | |




Toh ab iss index me har ek non key search value dali hui h jaise 1,2,3 toh yeah case dense index ka hua

Iss disk file me har non key value repeat to nahi h lekin har ek value ya search key ka index bna hua h

Imp:

- Dense & parse index only one type of idea which is used in based or non-based key attri indexing.

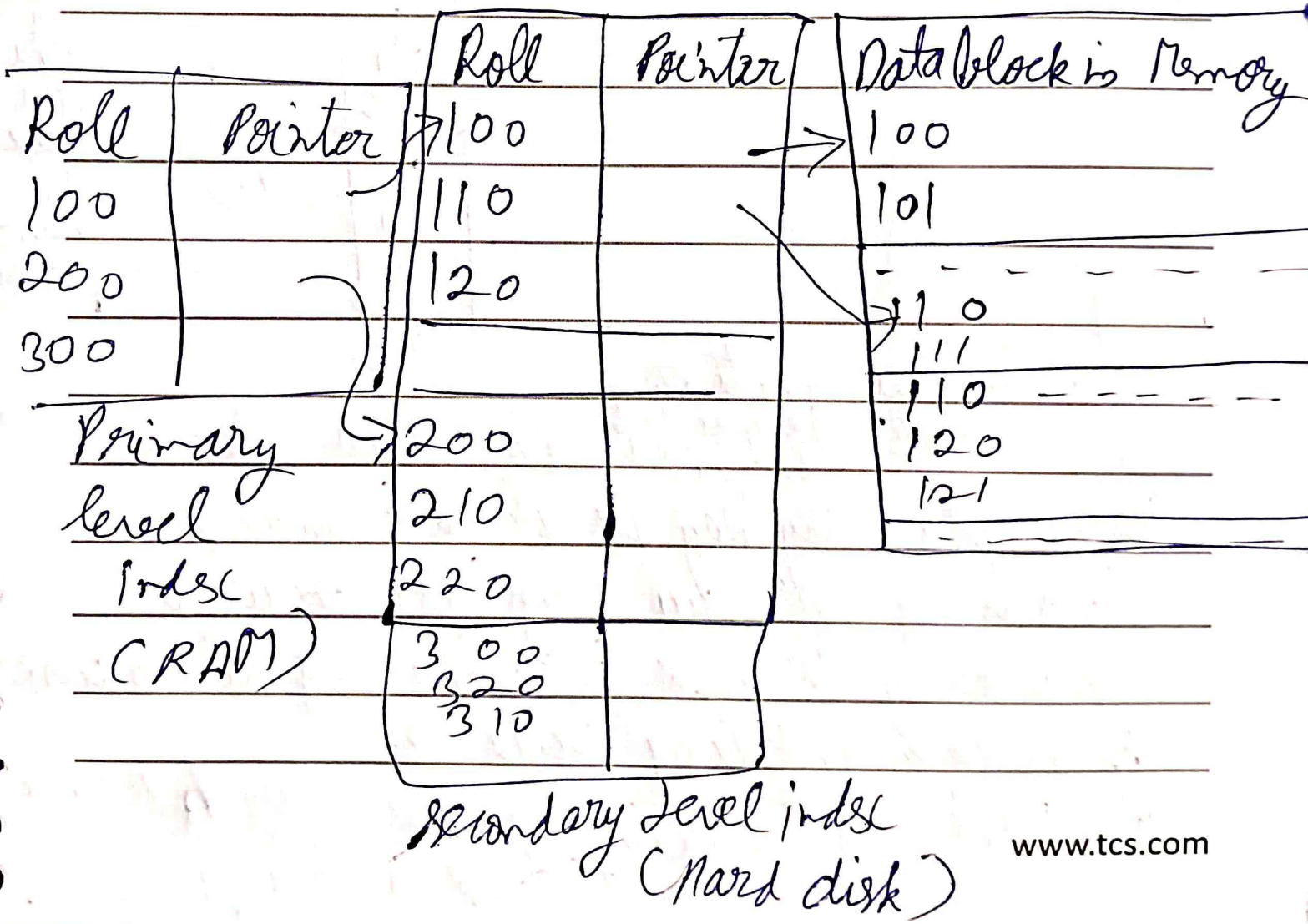
Both ones: primary indexing

→ primary key / Candidate key – sparse index
→ Non primary key — dense index

→ is sparse indexing no of entries in index file = no of blocks in datafile.

→ no of entries in index = Unique non-key attribute value in data file is dense index

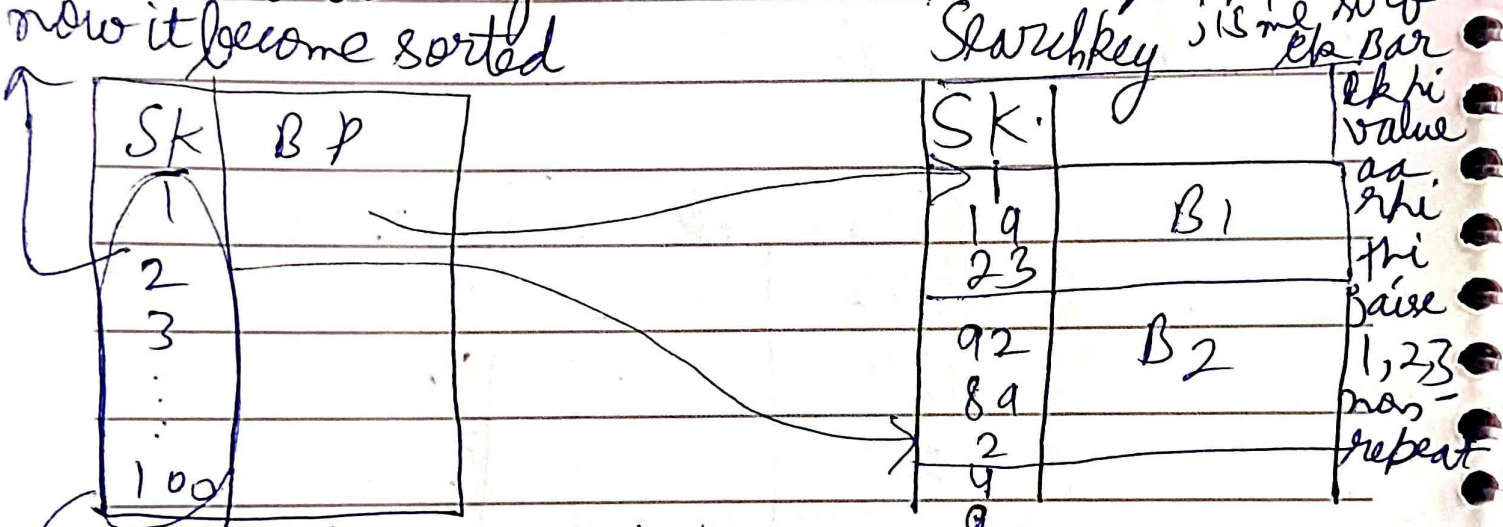
* Multi-level indexing :- jb 2 se zada indexing ek hi disk file pe ghte h index with 2 or more levels.



dense indexing example is secondary indexing.

(Secondary indexing) (datafile unsorted)
Primary indexing me (datafile sorted hoti h) ab secondary indexing me datafile me search key (unsorted hoti h) jaise ke agr primary indexing me stud id pe search h toh vo rank ya age ke column me toh unsorted hogi

now it become sorted

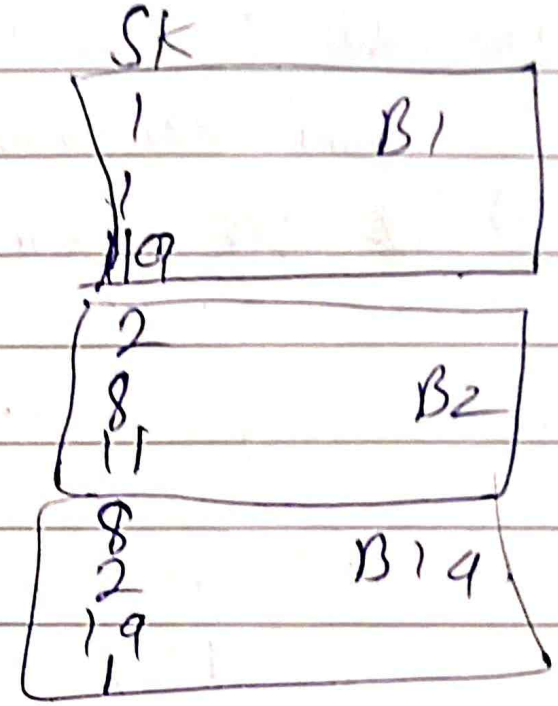



Dense indexing hoti h.
kyuki sare search key ki info index me store kr rkhi h. data file me primary key pe phele primary indexing toh thi hogi, ab mujhe secondary key pe indexing lagani h usse secondary indexing bolte h.
toh secondary indexing linear search krna padega aur dense indexing bolte h.

Secondary Index Also known as non-clustering index.

→ ab agr value datafile ki search key me repeat ho jae tb hum [linked list] lga ke index table bna sketch.

| SK | BP |
|---|---|
| 1 | B1 → B19 |
| 2 | B2 → B19 |
| 8 | B2 → B19. |
| | |
| | |




Interview Ques

→ Secondary indexing me unsorted hoti h toh uska advantage kya h index lgane ka

Ans:- datafile me agr hum kuch search krte tb hume use linear search krna pdta aur vo ek ek value pe iterate hota, lekin indexing krne ke bad, index table ab sorted ho gyi h toh use ab binary search lga sketch.

SQL is mostly vertical scaling

TATA CONSULTANCY SERVICES

Dec-15

NoSQL → Non-Relational SQL

↳ means Not only SQL

data not store in relational tables

in NoSQL :- 1) Can be stored as structured data, semi structure, unstructured data

2) Flexible schema

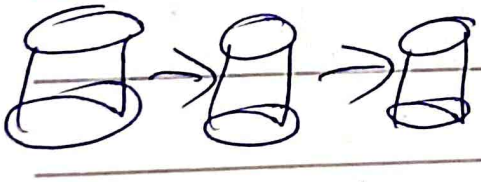
Advantages of NoSQL

1) NoSQL - JSON

{"id" : 1,
"f_name" : "aaru",
"l_name" : "saini"}

2) Scaling — Vertical - Update hardware, RAM, CPU

add node — Horizontal



load shared NoSQL cheap.

in single system in SQL

Costly

Vertical scaling → scale-up
Horizontal ,, → scale-out

Imp interview Q) Why we can't do horizontal scaling in SQL?


whole 3 db in one db & apply joins so it will become slow.

SQL → Table collection

Imp Lines) When in cloud applications have to make a database with no structured or semi or structure with fast response then use → NoSQL

Dec-16

## Types of Databases :-

1. Relational databases :- Features
   - Relational Mode - Table format
   - Related, But cons is that its highly slow when applied joins & no very used in cloud applications & also do not do horizontal scaling.

② Object-Oriented data models :-

↳ works on OOPS similar concept.

class → instance → Objects.

inheritance, polymorphism, encapsulation, abstraction.

class → attributes (characteristics)

↳ methods (Behaviour of Object)

→ stores everything form of objects.

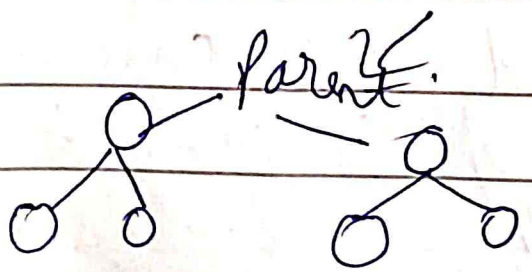
→ objects can have, tables, executable code, object they communicate via methods.

→ handles - Object ID.

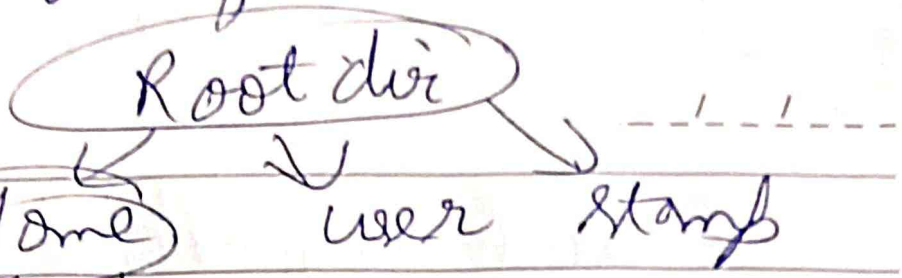
- information stored in db is capable of being represent as object.

③ Hierarcial db :-

eg :- Family tree.



eg:- filesystems (Linux)

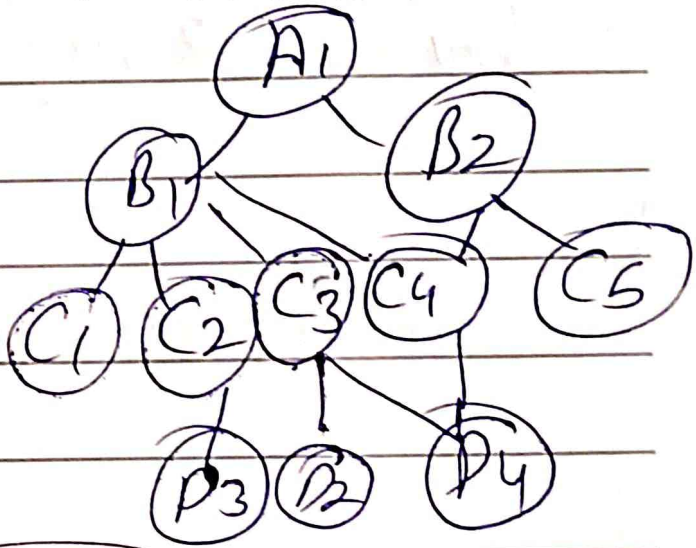



⇒ parent - child relation.

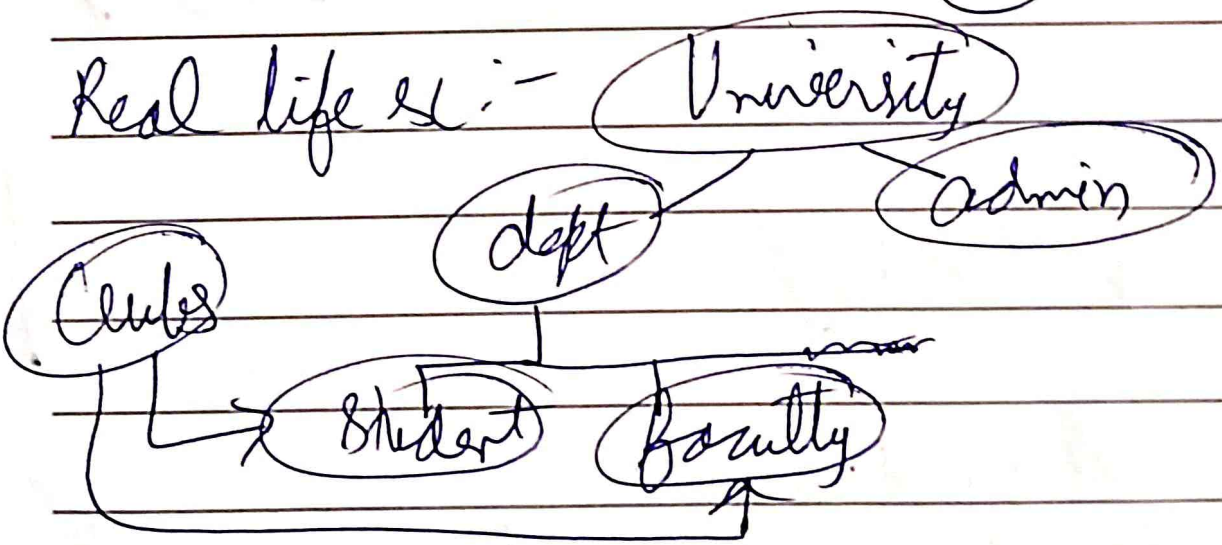
(4) Network databases :-

↳ Multiple parents

↳ Organise in graph structure.




Real life ex :-

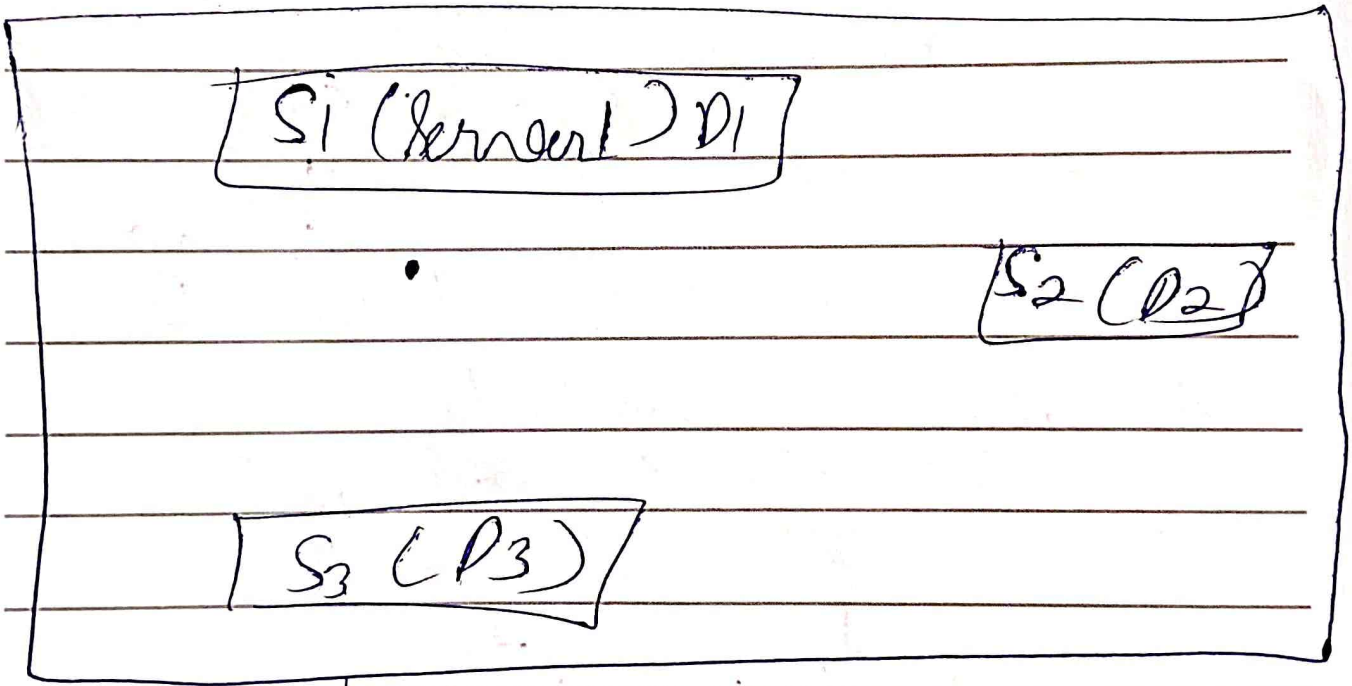


TATA CONSULTANCY SERVICES

Dec-17

# clustering / Replica sets

* clustering / replica set isliye use hoteh haiz. Suppose one website upe bahut sari request aati h and sirf ek hi db upe work karta h lekin system down hua toh site & db band ho jayega, toh us case me clustering use hoga.



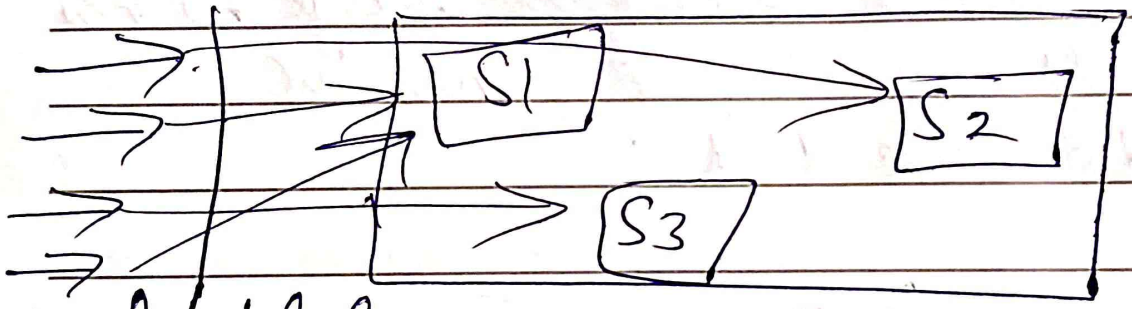

clustering

D1 = D2 = D3 - These all are Creplica sets)

Why we are doing clustering is to do redundancy means of not repeating same value in 1 db but by making replica of it.

Eg:- S1 → shutdown
→ Maintenance
→ etc.

But S2 & S3 works wells.
and if load so much coming on S1 → then it load Balance on S2 & S3.




load balancer

(3) High availability advantage → if S1 is down then S2 & S3 is available. (BackUp).
→ Ex of Replica/set is Content delivery network.

Dec-18.

TATA CONSULTANCY SERVICES

__/__/____

# Partitioning & Sharding in DBMS :-

Data → store → DBMS.

↓

Bhot saara data (Huge data) → Masses like facebook, insta system.

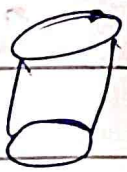
1) Data huge ~~se~~ (Manageability issue)
2) no of Request are ~~Request~~ large so the one data won't work. Then we distribute data.

Data distribution

↓ use

→ data Optimization Technique.

① Scale-Up (Hardware) ↑ increase kr dete h.



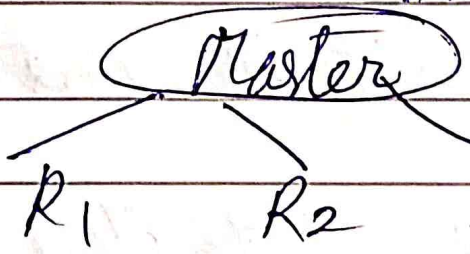
1 TB → 2 TB.

If we ↑ all hardware requirements if request getting more but this doesn't

impact on time consumption like it does not request query in less time & in fact cost of hardware requirement increases so not logical to scale-up.

② not so to scale-up, can do add replica-set (clustering).

On master branch any update comes then it will be distributed update to every node




& this is propagation which cause delay & have to also take care that update go in every node or not. This is good but not Best method.

③ Best Method is Partitioning :-
Partitioning is method of scale out that is horizontal scaling in this also somewhat similar to clustering in

which new node are added.

Student Table ( Response time ↑, due to ↑ in data

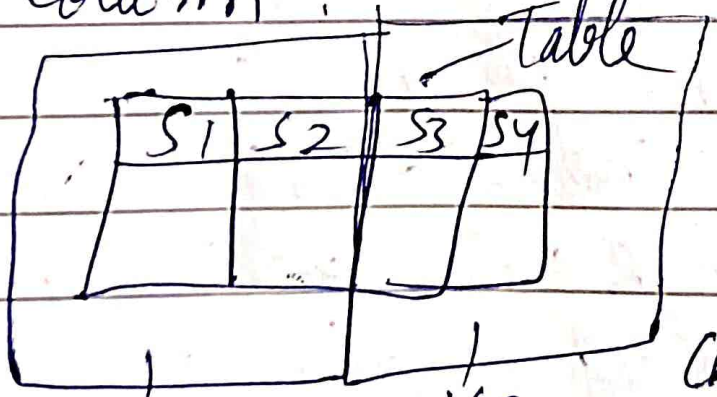
| id | Name | class | address | phone |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Vertical ~~scaling~~ ~~in which servers are~~ partitioning — Horizontal Scaling

⇒ in clustering we are not dividing the data we are making its replica so partitioning is much better.

⇒ Vertical partitioning me vertical slice.

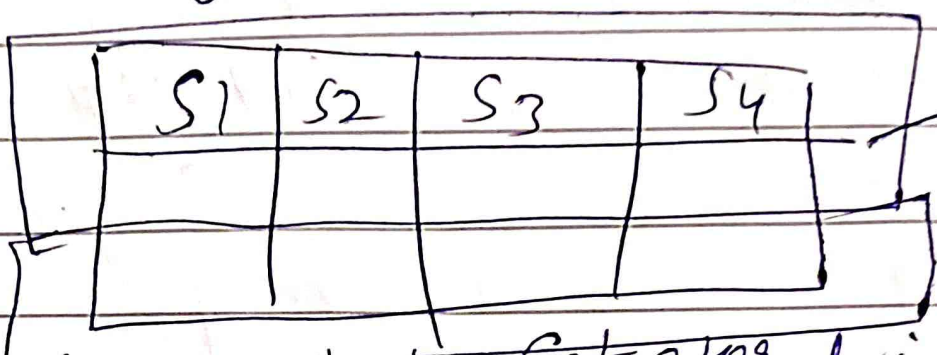
Column :-




if we do V1 & V2 Vertical partition then we have to choose data from V1 also & V2 also to

TATA CONSULTANCY SERVICES

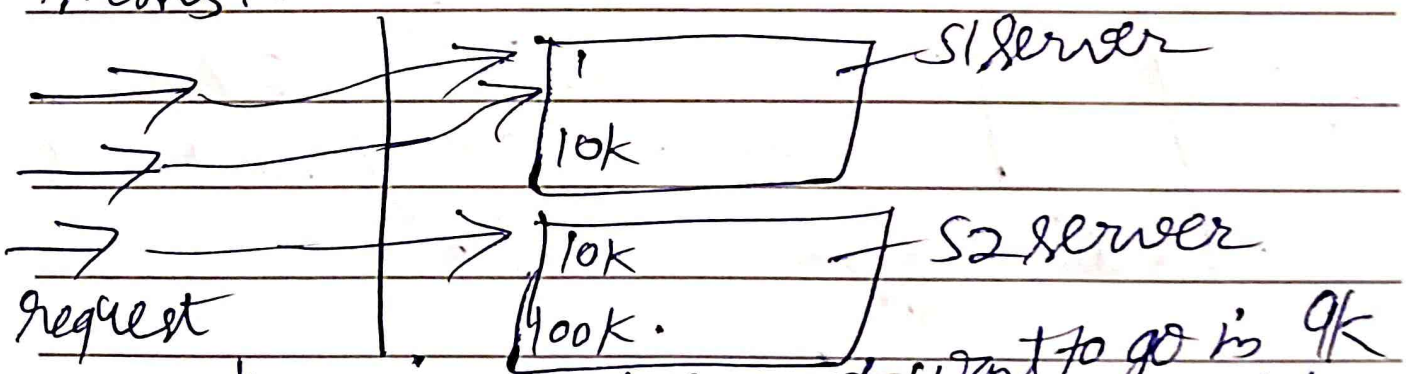
obtain full data but in horizontal partitioning.




all like values from 1 to 25K stored in S1 server & from 25K to 50K stored in S2 server.

→ independent chunks of data got from horizontal partitioning if want 10K record then we fetch from S1 server leaving S2 server do nothing.

→ Advantage of horizontal partition is parallelism means:-



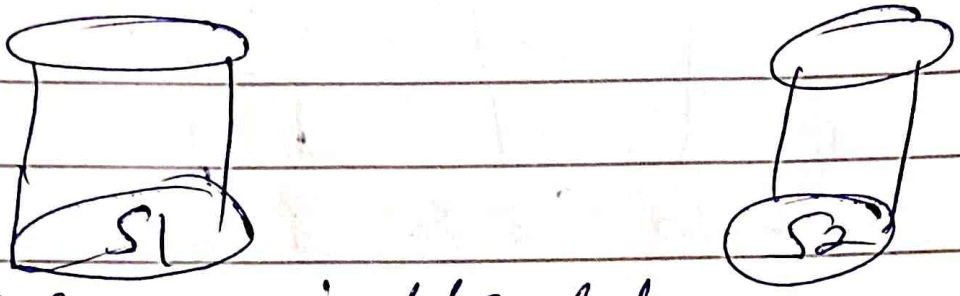

So by knowing that request want to go is 9K records so & another want is 11K records so both S1 & S2 will work toget

to apply.

* Sharding - Horizontal partitioning technique

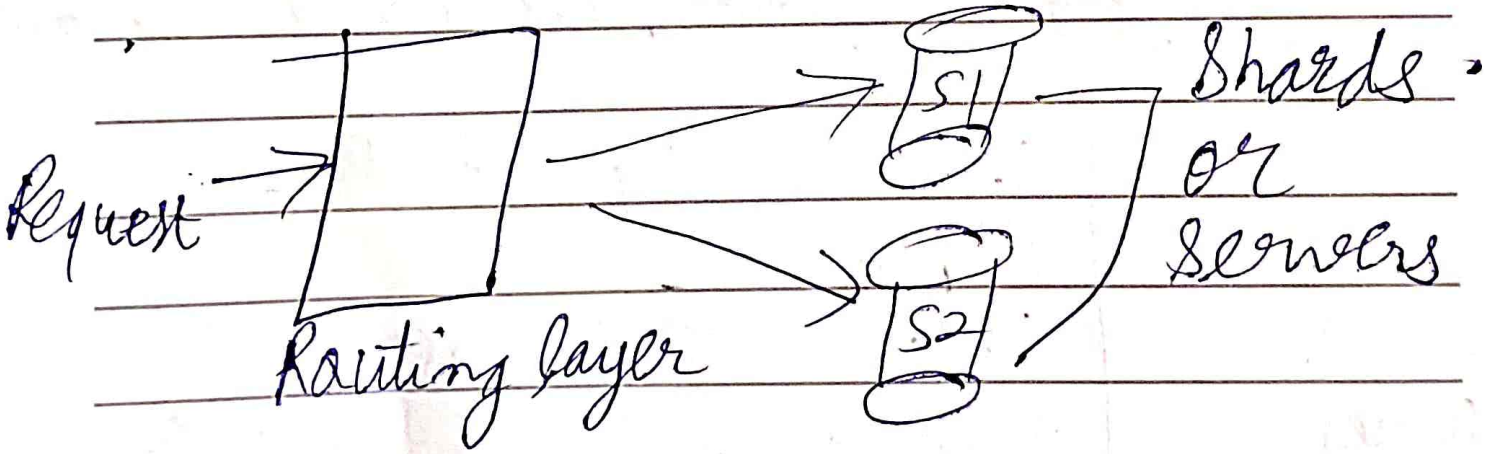
Student data:

1 → 10k → 100d → 100k.




S1 & S2 are independent.

One additional layer knows as routing layer have to introduce in sharding. which filter out that in which shard (layer / server like S1 / S2) have to sent request.



→ shard key or partition key is the key through which we partition like student-id in student table.

→ Sharding ek tarika h partioning ka.

⇒ Non-uniformity feature ek disadvantage h jo ki agr hum ek shard me zada values dal de aur dusre me kam toh dikat h isliye use barbar re-shard krna pdta h time pe

---

Dec-20

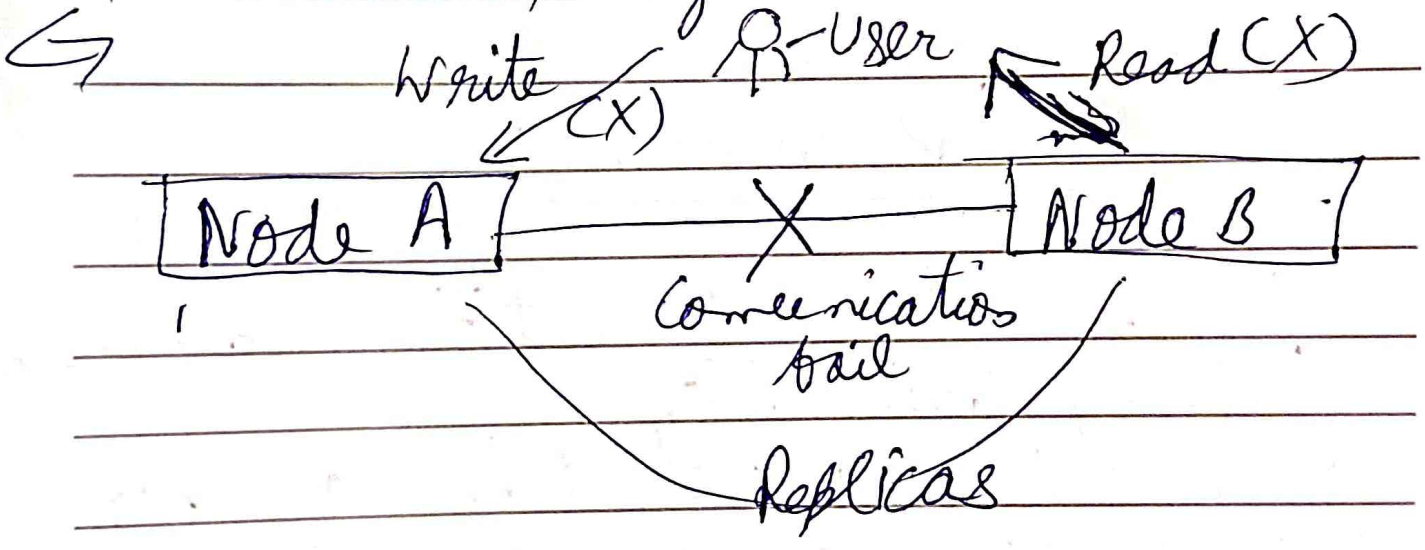
## CAP Theorem

C A P —— Partition Tolerance
Consistency Availability

if system is not distributed then all this 3 CAP will be performed, But if ur system is distributed i.e 2 servers or multiple nodes then from these 3 only 2 will participate/exist.

→ Partition Tolerance means jb ek distributed system partition hota h. it means that break in Communication in nodes, toh partition tolerance ka mtlb h ke unke beech me system fail nhi hona chahiye. bhale hi kuch msgs ka delay ho chalega lekin not fail

→ if ur system becomes too large then have to distribute the system into nodes.

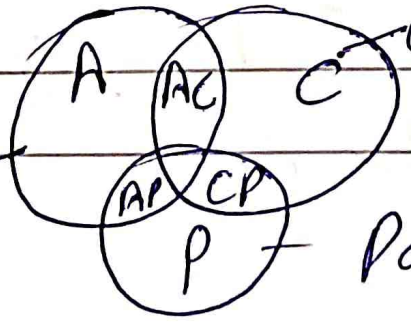



1) Consistency lost → (Both request should process) at same time agr hume node A and node B me distribute kiya & user ne write kiya function node A

pe but vo read Current value nhi previous value hi karega node B se bcoz hume dono system A & B ko available rakha h lekin unke beech me partion ki kaaran Communication nhi hui aur vo consistency nhi rakh pae.

② Availability (lost) - if we fail one request, ab agr hume Consistency maintain krni h pe vo write bhi kare aur read kre lekin previous value toh read function ko hum "failed" kr dege available hi nhi dikhaege toh Consistent hoga

⇒ Koi ek hi rehsktih property ya toh availability ya Consistency, partion pe dono nhi ho skte.



TATA CONSULTANCY SERVICES

__/__/____

Toh agr CA mtbl consistent & available sirf jb hota h tb koi node na ho mtlb system distributed na ho, otherwise

→ AP dbs are eventually consistent mtlb jb partition bir se establish ho jayega tb consistency maintain ho jayegi Badme

Scaling patterns

M. Imp lecture

TATA CONSULTANCY SERVICES

Dec 19

Q) Which Scaling patterns used where large user in db & why scaling is required?

A) We do scaling becoz user are increasing on db; so that performance will increase, all time availability of db, there is no lag while using application; no API latency.

Q) Which scaling option is feasible at this moment?

A) When there is startup so few users are available then there will be no use of big db.

* Scaling Patterns:-

There are some patterns through which we can do scaling, so have to use pattern efficiently so that cost also remain constant.

TATA CONSULTANCY SERVICES

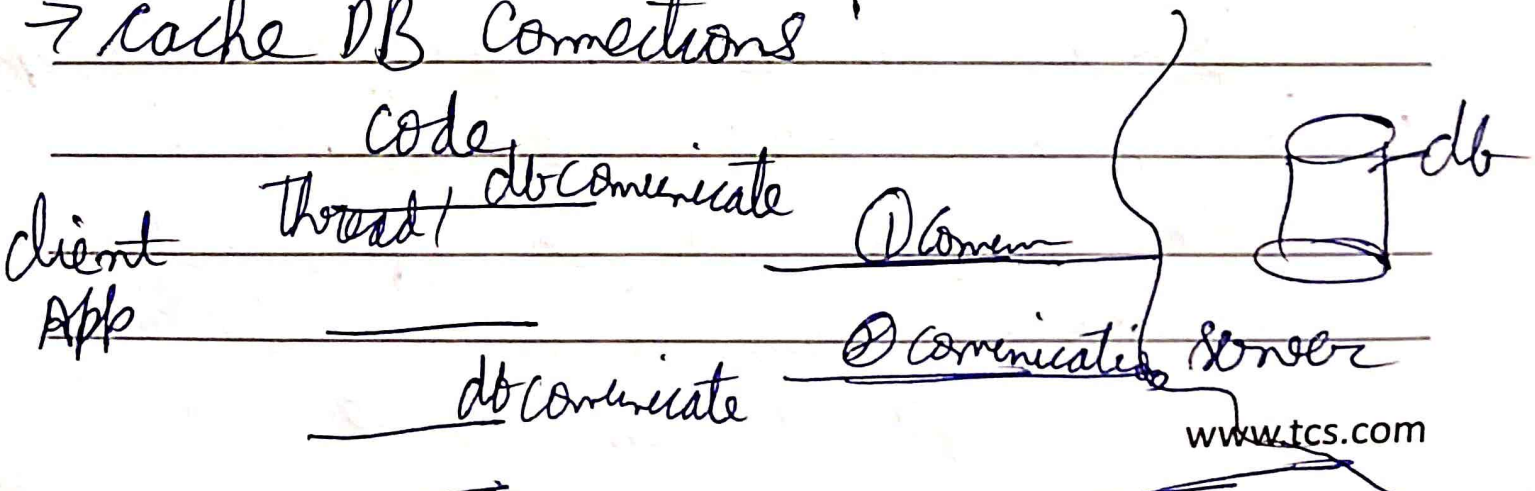
__/__/____

* Pattern 1
Query Optimisation & connection pool implementation

→ 1) iss pattern ko kyon use kete h like non-dynamic data jise booking history, payment history & user profile sirf client ke phone pe hi dikha de toh vo fetch nhi karega db se aur cache me se hi jldi request aa jayegi [dynamic cache me nhi lg skta jisa ke now driver location, current res &]

→ normalise kr skte h lekin iss app me cab booking me highly normalise h bcz joins ki query bahut time le rhi h & db redundancy kr do ya fir [noSQL db use kr skte ho].

→ cache DB connections



TATA CONSULTANCY SERVICES

__/__/____

ek client app h aur usme jo code likha h vo server ke sath kuch communicate karega connections banega lekin har bar request aai toh new connection banane ki jarurat nhi h infact kuch jo connections h vo already jo bhi tech use kr rhe h jaise springboot, java th usme connection pool libraries aati h vo use karega toh humara API latency nhi bhi hoga, isme multiple threads use karo some connection ke liye

→ Use connection pool libraries to cache db connections.

→ iss steps se kuch optimisation ban users pe ban chalega lekin agr bahut users ho gye like 100 booking per minute · to pattern 2 use karo

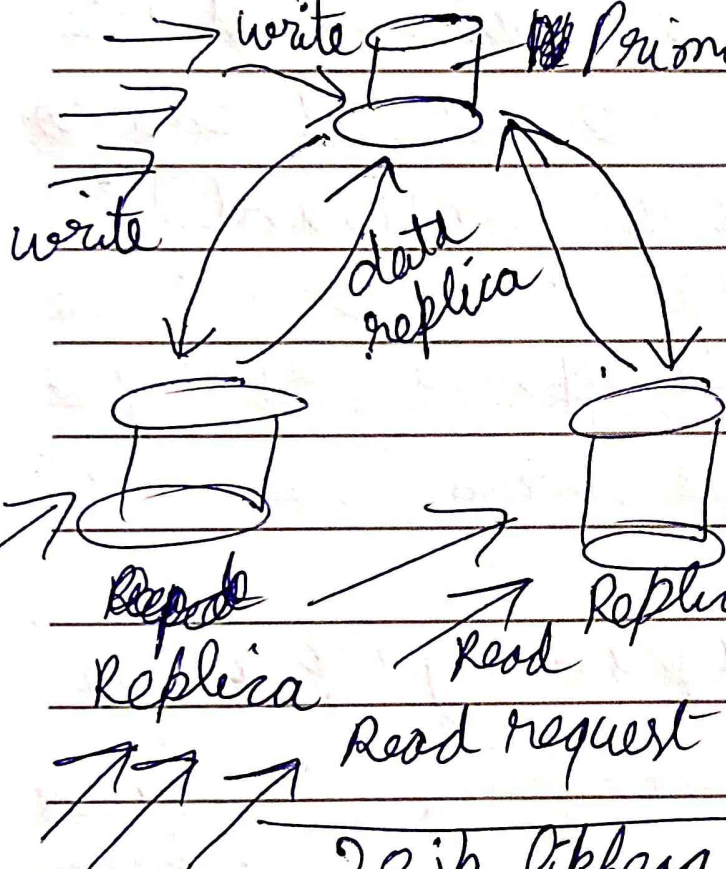
* Pattern 2 :- Vertical scaling or scale-up.

1) tiny machine ko upgrade karo, uski RAM & SSD lagao but yeh swif pocket friendly tk hi option h, cost to nhi km ki abhi thoda optimisation aur ho sakta gya, lekin ab aur users aane

pe 300 booking per minute aa gyi toh firaab pattern 3 choose kr skteh vo6 !-

* Pattern 3 Command Query responsibility segregation (CQRS).

N request ⇒ N/2 (Read request), N/2 (write)




Rumekha master primary db pe write request aayegi aur replicas pe read request aayegi agr write request aai aur bola ke write x=20 phele yeah x=10 lekin ab 20 likh do, toh yeah 20 jb likhega uske bad primary ko data send bhi krna h replica pe ke jb read request aai usne toh vo ab 10 nhi 20 x=20 show kare users ko. toh yeah time to time procedure primary aur replica me chlta h.

TATA CONSULTANCY SERVICES

Interview pr Ques

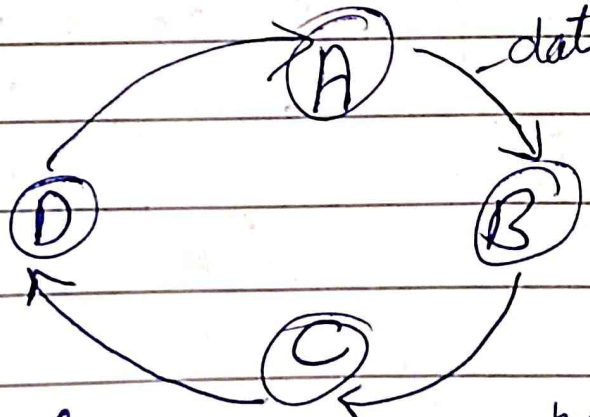
Q) Agr ek write request aai aur usne apna kam kr diya lekin read request phele aake data read krke chli gyi toh kya yeah sahi h.

A) Yeah bilkul sahi h kyuki real world scenario me write request ake se execute honi chaihye h bhale fir read request 1-2 second bad me execute kre updated data ko

(write primary db always consistent)

Now primary is not able to handle all write request then, but zada lag b/w primary & replica is not OK so pattern 4 aga lo :-

Pattern 4 :- Multi primary Replication :-




data replicate

4 Machines 4 Replicas charo me same db hoga A = B = C = D

Push update aayega toh B Push Pushtime me Ape check karega aur update kr lega Push ko aise hi C ← B & D ← C

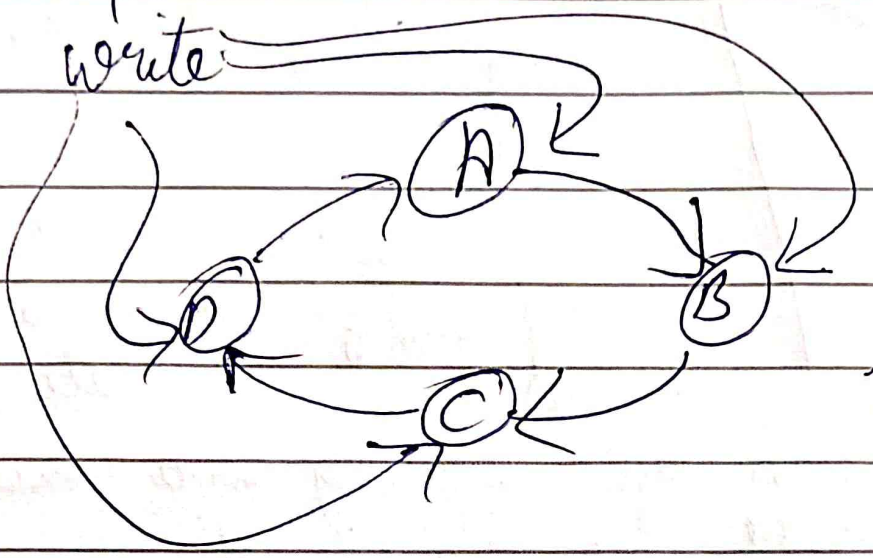
Write → any node will handle it.

TATA CONSULTANCY SERVICES

Write request kisi bhi node ko randomly de do koi ya pe primary which sb equal h.

→ N (write) request

→ write

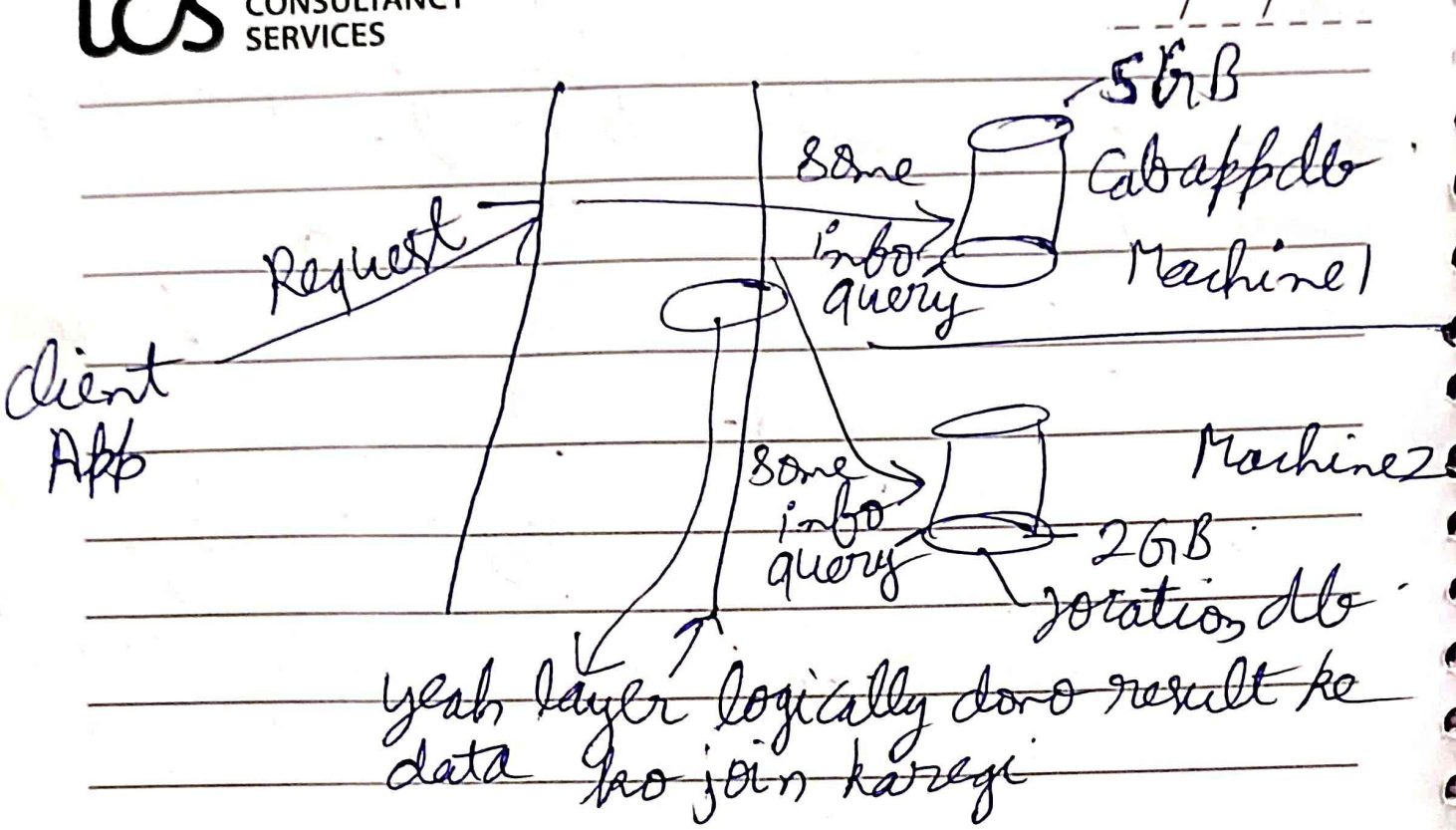



ab read request aai toh vo jis bhi node pe jayegi aur use phele response milega vo read vahi chli jayegi

* Pattern 5 :- Partioning of data by functionality.

↳ isme separating location, tables in separate db schema milte schema ki alag kr do.

↳ ab use dy schema ko separate machines me phi dal do primary-replica / multi-primary configuration krke.




dono hi Machine 1 & 2 pe query lagai aur fir & fir ws layer ke through join kare query result aur usko se JSON file me Big diya data.

→ ab aur Countries me leke jana h business ab toh India me tha fir Germany leke gye to API latency lag aaya to. Sharding / horizontal Scaling karege. or Scale-Out.

__/__/____

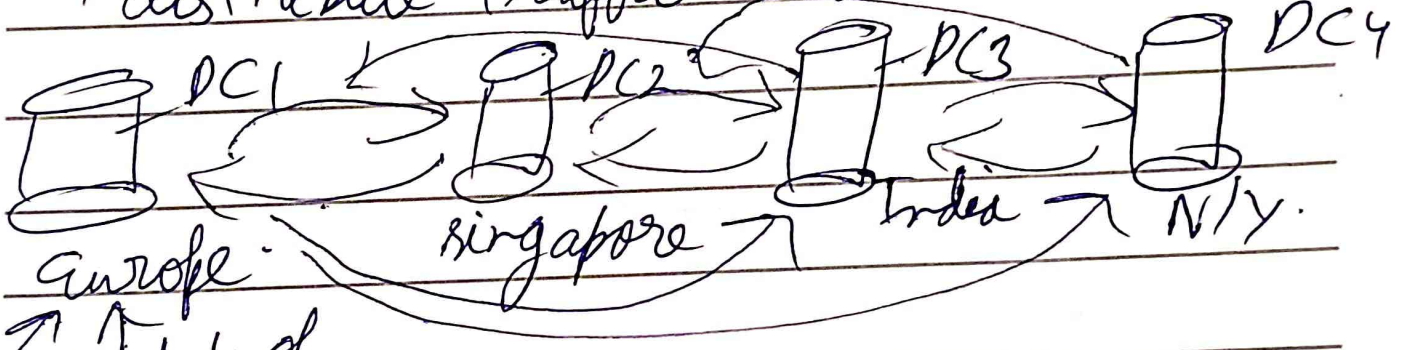
# Pattern-6 Horizontal scaling or Scale-Out:

⇒ ab dusre continents me jana h:-

↓

## Pattern 7:- Data Center Wise Partition:

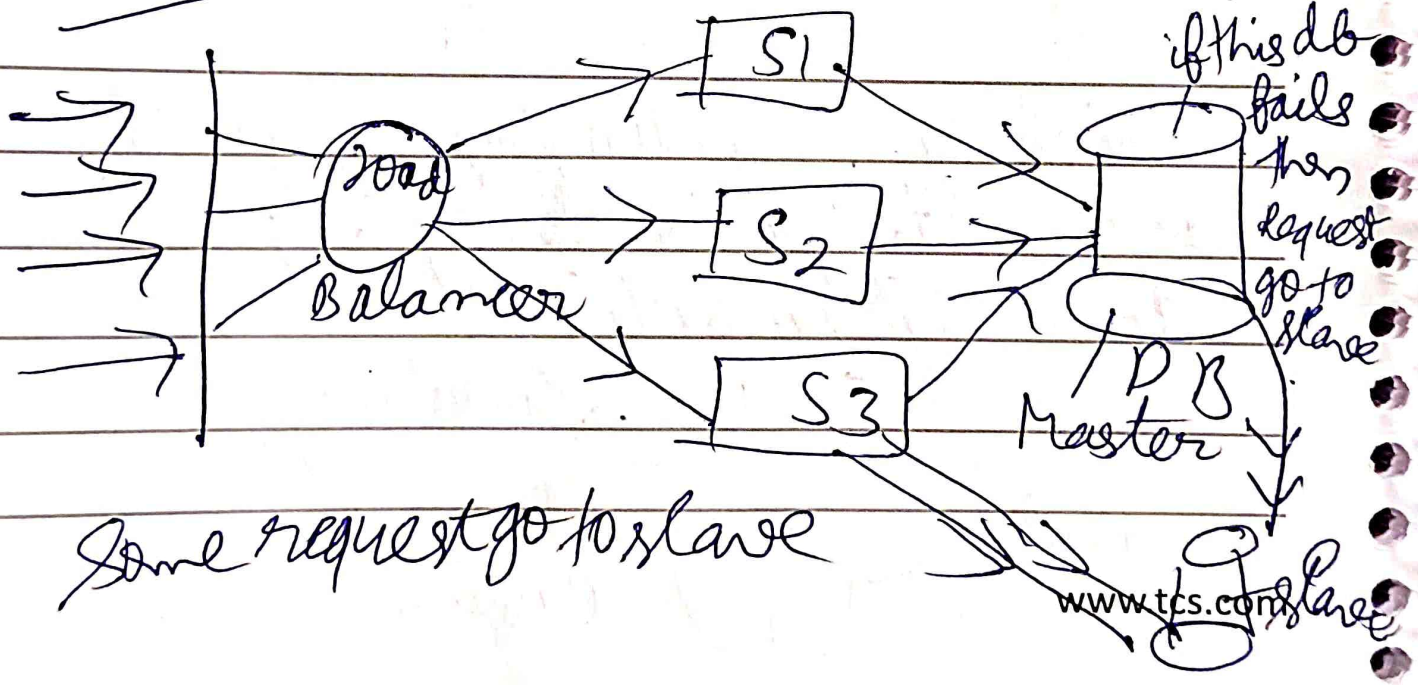
↳ distribute traffic across data centers:




↑ ↑ all data of Europe kept in Europe center only not in another data center; but if any system fail in Europe, so for this reason that of non availability of Europe datacenter if system fail so that's why sometimes these datacenters will make data replications among them. This will make availability of your System.

# Lec-21

## Imp Topic Master-Slave architecture



TATA CONSULTANCY SERVICES

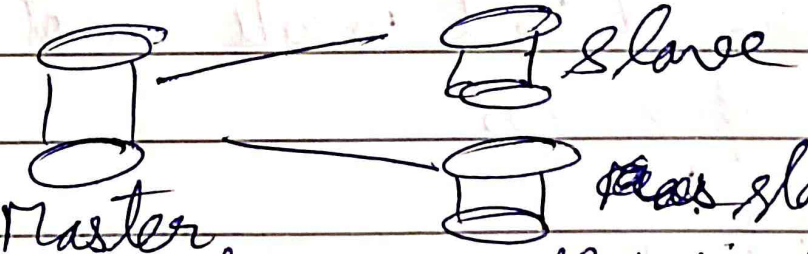
final

if db get crash / fail then bad request it will show.

→ Master node : → Write Operation
↳ Original DB
↳ latest db
↳ Owner db
↳ primary db
↳ it has latest info.

→ slaves / Replicas → Read operation
↳ Db replication (They replicate)

→ How Replication happens?




① Asynchronous replication : - like 1 or 2 sec delay - like SB comment diff.

(2) Syncronous replicatio - But in banking app, 10 sec delay won't work

Ques) Slave gets update Query or write Query ?? Then :-

A) - 1) First think to ignore or never allow slave to accept write or update anything.

2) if u allow it anyhow then we can make a way to propogate it to master so it become master-master model not "master-slave" model anymore So if this 'M-M' model become then we can we pattern-4 on it multi-master / primary replicatio.

* Advantages :-

① Backup :- if write operation fails then can put read operation (if does'nt get failed at least)

(2) Scale-Out read operations

(3) Availability ↑

(4) Reliable

(5) Parallelism / parallely read request.
API

(6) Reduce latency