SHIVANI PATEL(DBMS-NOTES)

(AT INTERVIEW TIME)

SNO.	TOPICS
1.	NORMALIZATION AND ANOMALY
2.	SUMMARY OF ALL NORMAL FORMS
3.	KEYS IN DBMS
4.	3-TIER ARCHITECTURE
5.	BASIC SQL QUERIES
6.	JOINS IN DBMS
7.	ACID PROPERTIES
8.	SCHEDULE IN DBMS
9.	INDEXING IN DBMS

Notes

Date __/__/_____

Normalization AND Anomaly ——...

Normalization : what ??

⇒ It is a technique to remove or reduce redundancy from a table.

* redundancy : multiple copies of same data.

* Duplicacy is of two types —

1. Row level
2. Column level

⇒ Row level : took on example like

SID	Sname	Age	
1	Shivika	20	these two row are same
2	Vanika	25	
3	Tanifra	20	
1	Shivika	20	

Remarks

Date__/__/_____ Not

Here, we do SID is primary key (unique + Not NULL)

So we set SID, no two row will same, so row level duplicacy removed.

⇒ Column level: took an example

SID	Sname	Cid	Cname	FID	Fname	Salary
1	Shivika	C1	DBMS	F1	John	300
2	Vanika	C2	JAVA	F2	Bob	400
3	Tanika	C1	DBMS	F1	John	300
4	Himanya	C1	DBMS	F1	John	300

These column are same.

Remarks

here we have three types of problem —

Notes

1. Insertion anamoly
2. Deletion anamoly
3. Updation anamoly

- Insertion

* In previous table if we add new course like pharmacy we can't inserted it.

Reason: SID because I only said add course not student ID because after course insert then and then student register. and

We set SID as primary key and Primary key can not be NULL.

This is called Insertion Anamaly.

Remarks

• Deletion

from previous table,

If we delete SId = 2

So simply: Delete form Student where SID = 2

row two will be deleted but due to deletion we delete [all data of student.]

In this [extra information] also deleted.

like faculty name, salary etc.

And we [can't recover our data]

Remarks

lotes

• Updation

from previous table,

update sname —

set sname

Eg query: UPDate student where sname = 'TanikaPatel' where SID = 2.

No problem here.

But if we change salary of FI from 300 to 900 then —

in this, it changes all FI 300 to 900

but you want to change only Tanika Patel faculty salary

but here it changes all FI so in create problem.

Remarks

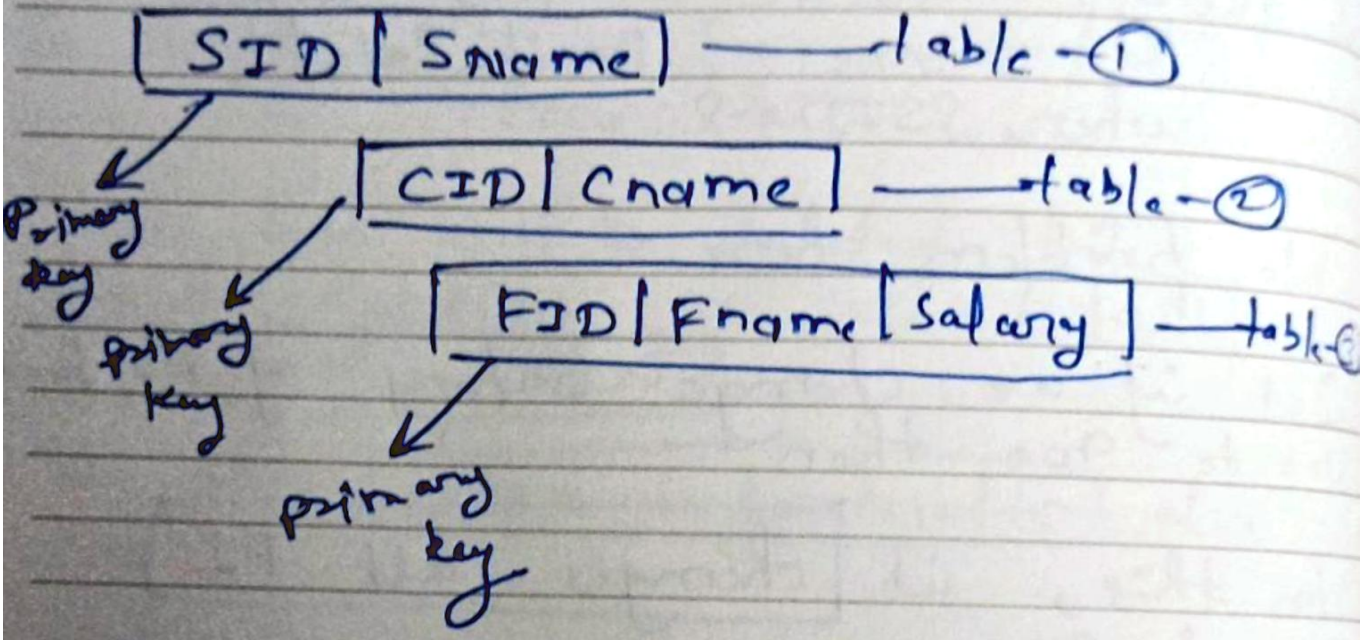
Causes due to column level duplicacy.

Date__/__/_____

Note

How Normalization help form there ??

We can break our table like



Remarks

Notes

Summary of 1/11 Normal forms

1NF (First Normal form)

⇒ If a table contain multiple value of one attribute then

:⇒ Not a normal form.

Condition for 1NF :

1. Only single value attribute

like!

Roll No.	Course
1	DBMS
2	CN

It is in 1NF

If

Roll No.	Course	Invalid not a 1NF
1	DBMS, C	
2	CN, C++	

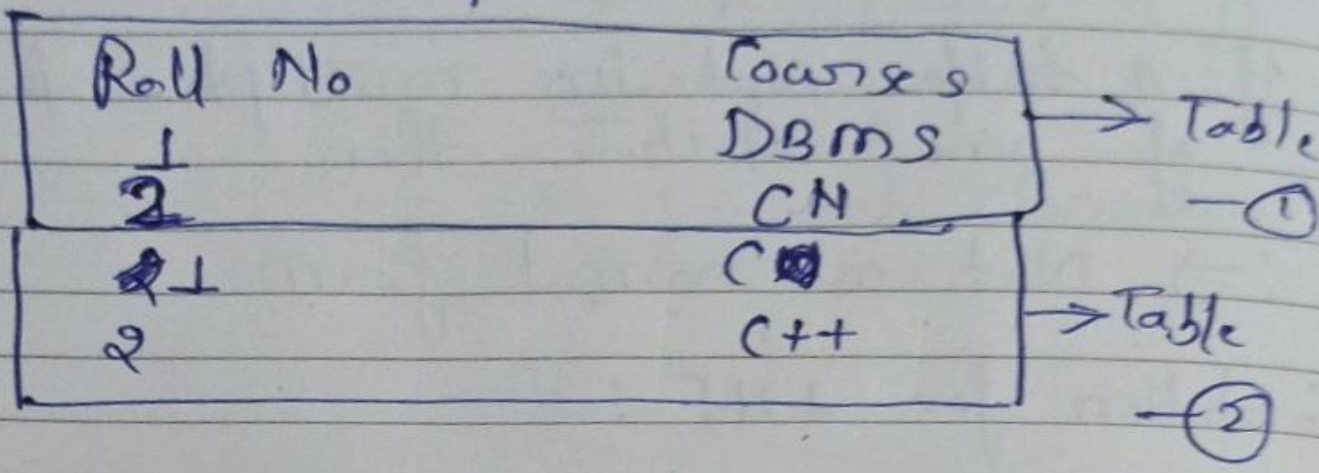
Remarks

Key Point
1NF follow atomic rule

Date__/__/_____ Note

Solution for that:

If we have multiple attribute then break table like



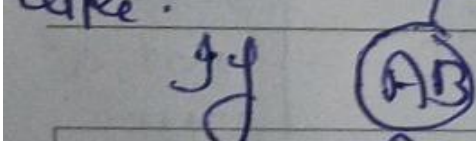
2NF (Second Normal form)

First cond^{n} : must be in 1NF

② Partial dependency (NO*)

③ Only full dependency

like:



Remarks

A → C X
B → C X
{Not possible}

↓

Partial dependency

Notes

3NF (Third Normal form)

Condition :

① must be in 2NF

+

② No transitive dependency

↓

two arrows

$X \rightarrow Y \rightarrow Z$

Z → Non prime

X → Primary key / Unique attribute

Y → Non prime / Non Unique

+ Non-prime attribute Z, Y

but X is indirectly connected through with Z so it is known as transitive dependency.

$X \rightarrow Y \rightarrow Z$ (through Y → because Y is also non-prime so its create problem.)

Remarks

Date__/__/____ Not

BCNF (Boyce-Codd)

Condition:

① must be in 3NF

+

LHS (left hand side of attribute) always be Candidate / Super key so it is in BCNF.

⇒ Candidate key: unique for every row.

⇒ Super key: combination of columns that uniquely identifies any row.

Remarks

Notes

Date __/__/____

4NF (Fourth Normal form)

Conditions: must be in BCNF

⊕

No multivalued dependency

↓

means

↙

Shivani ⟶ 3 Mobile No

⟶ 3 Email

Store in table like ⟶

Shivani	m1	E1
Shivani	m1	E2
Shivani	m1	E3
⋮	m2	E1
⋮	m2	E2
⋮	m2	E3
⋮	⋮	⋮

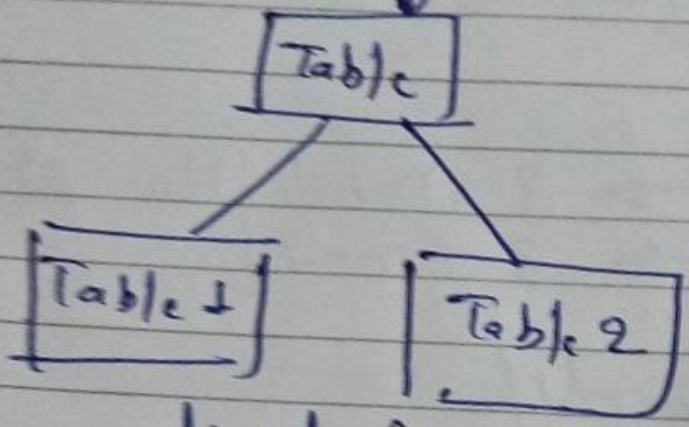
Remarks: This is multivalued dependency.

5th Normal form

condition: In 4th NF

(+)

lossless decomposition



divide in vertical (col/m) form.

There will be chance of extra tuple (row)

it is lossy so remove that use common in both table like candidate key so its reduces at redundancy.

Remarks

otes

Keys In DBMS

- Keys play a very important role in our database.

They → are used to identify one and only one instance of an entity uniquely.

- Types of key:
  - Primary key
  - Candidate key
  - Super key
  - Foreign key
  - Alternate key
  - Composite key
  - Artificial key

These four are very important.

Remarks

I discussed here detail in about primary, candidate, foreign and super key.

⇒ Primary key:

Primary key = { Unique + Not NULL }

Property: like in **every table** **every entity** has their **unique** **id** for identification.

Like in college I. card: We have a key like = Enrollment number / Registration number

Remarks

By using this no. they get all data about particular student.

otes

Unique — what comes inside ??

Phone no.
Aadhar Card
PAN
Reg No.
Enroll No.
:
etc.

} Unique

↓

If it is unique so that is candidate key but for primary key unique is not complete satisfaction.

↓

Not Null — what ??

Never empty (it is compulsory field)

Remarks

For primary key we achieved Unique + Not NULL

Date__/__/_____

Note

Candidate key

Key → attribute

use of key → uniquely Identify

Sname	City	Age	
Shivani	kanpur	20	Same
Samyama	Lucknow	21	
Prachi	Delhi	28	
Shivani	kanpur	20	

Take eg. of student data table

- Adhar Card
- Roll no
- Registration
- License No
- Remarks
  - Mobile No
  - Email

All are diff^n for all every student

These are set of candidate key

Point: We choose primary key from set of candidate key.

Foreign key

It is an attribute or set of attribute that references to primary key of same table or another table (relation).

Point: Maintains referential Integrity.

Remarks

Example

Student table

PK (Primary key)

Roll No	name	address
1	A	Delhi
2	B	Mumbai
3	A	Chandigarh

Course Table

foreign key

CourseId	Course Name	Roll no
C1	DBMS	1
C2	Networking	2

Note

Remarks

Here in course table Roll no is used as a foreign key.

Here,

Student table is **Referenced table** where **primary key** exist.

In Course table, it it **Referencing table**, where **foreign key** exist.

Query: After table creation —

After table course ADD constraint fk foreign key (Roll No) references Student (Roll No).

Remarks

Point —

Table has more than one foreign key (possible).

Date__/__/_____ No

Super keys

- It is a combination of all possible attributes which can uniquely Identify two tuples in a table.
- Super set of any Candidate key is Super key.

Student table

Ck = Rollno

Eg: we add Rollno Name age

Rollno, name
Rollno, age
Rollno, name, age

Remarks: then they will be super key.

Notes Date __/__/_____

Que: $R(A_1, A_2, A_3, A_4 \ldots A_n)$ then how many super keys are possible.

Sol $\longrightarrow$ A_1 is candidate key.

$\longrightarrow$ A_1, A_2 are candidate key

Case-①

// given **A_1 is candidate key** then **A_1 is necessary** to present in all subset.

$\Rightarrow 2^{n-1}$ is super key possible.

Case-②

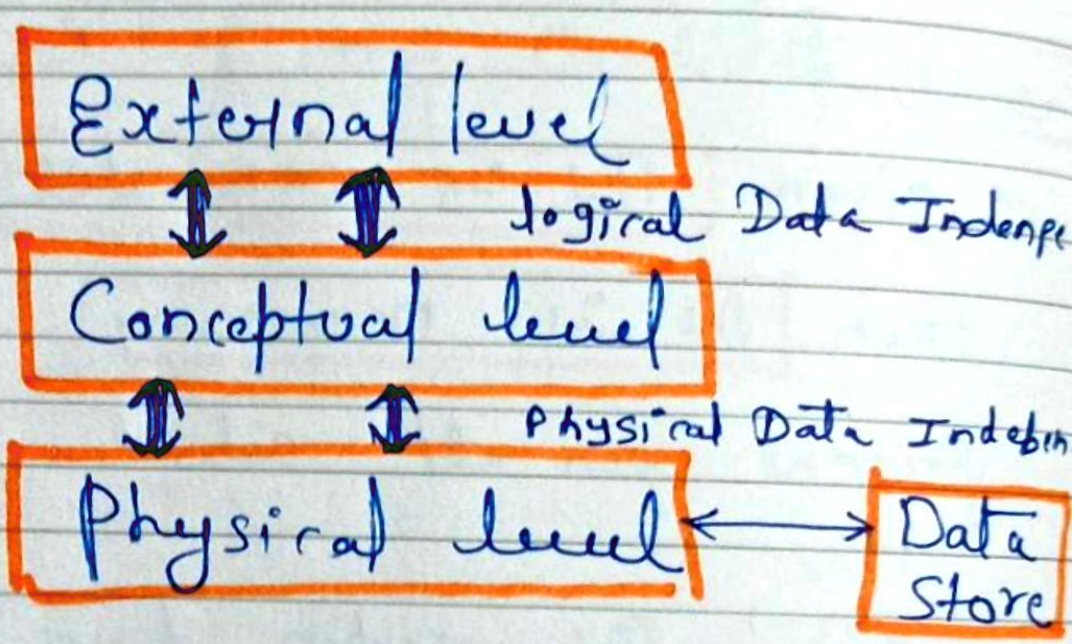
// A_1, A_2 are candidate key

Remarks

So Here $(2^{n-1} + 2^{n-1} - 2^{n-2})$ is no. of super key.

DBMS 3-tier Archite

⇒ It's divide the complete system into 3 inter-related but independent modules.



Data Independence :

means a change of data at one level should not affect another level.

Remarks

Notes

Physical data Independence:

→ Any change in the physical location of table and indexes should not affect the conceptual level or external view of data.

Conceptual data Independence:

⇒ Data at conceptual level schema and external level schema must be independent.

⇒ change in conceptual schema does not affect external schema.

Remarks

Two tier Architecture

Two tier simply means

⇓

two layer.

- Client layer
- Database Server

Client layer means it is a machine that has interface which help to fetch data from database server.

⇒ Client has application programme (fetch a particular data) come on database server then we execute it.

Remarks

⇒ Two-tier also known as

Client - Server Architecture

like example —

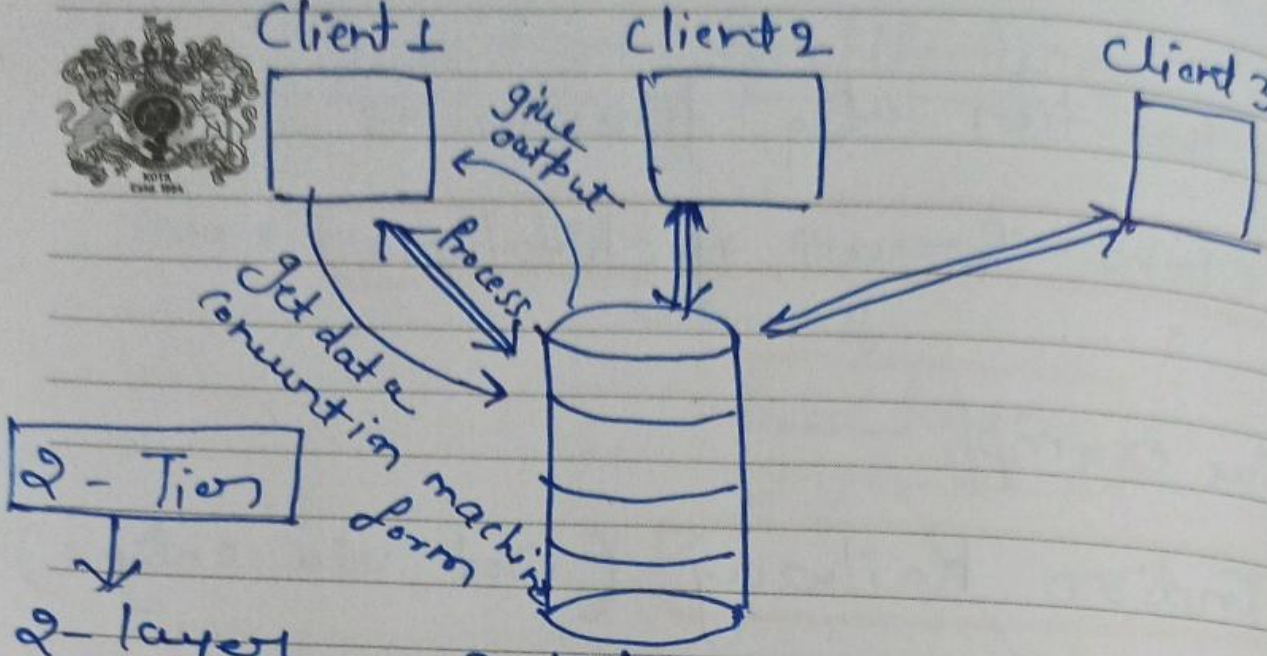
Indian Railway (not via online)

directly go on station.

⇒ form a fill and book our ticket then the member of ticket window fetch our data manually (that is client oractime)

⇒ After filling data they process through database server then they print the ticket and gives us.

Remarks



Advantage

- two layer so maintainance easy
- limited client limited database

Disadvantage

- Problem of Scalability (when no. of user more).

Remarks

* do at that time 2-tier fail.

otes

Date__/__/_____

• Security → here client direct interact with database so it causes vulnerability.

Three tier architecture

This architecture contain three layers

① Client layer

② Business layer

③ Data layer

⇒ Client layer: used to refers to local interfaces used to author, third-party clients.

Remarks

⇒ Business layer: the query send by clients is send to business layer.

Date __/__/_____

No

⇒ Data layer:

At this tier, data layer contain database resides along with its query processing language.

⇒ this layer has multiple views of database can be provided by application.

⇒ When you enter information in an interface it give direct output because all query verification work done by business layer.

Remarks

⇒ So load of database server is less.

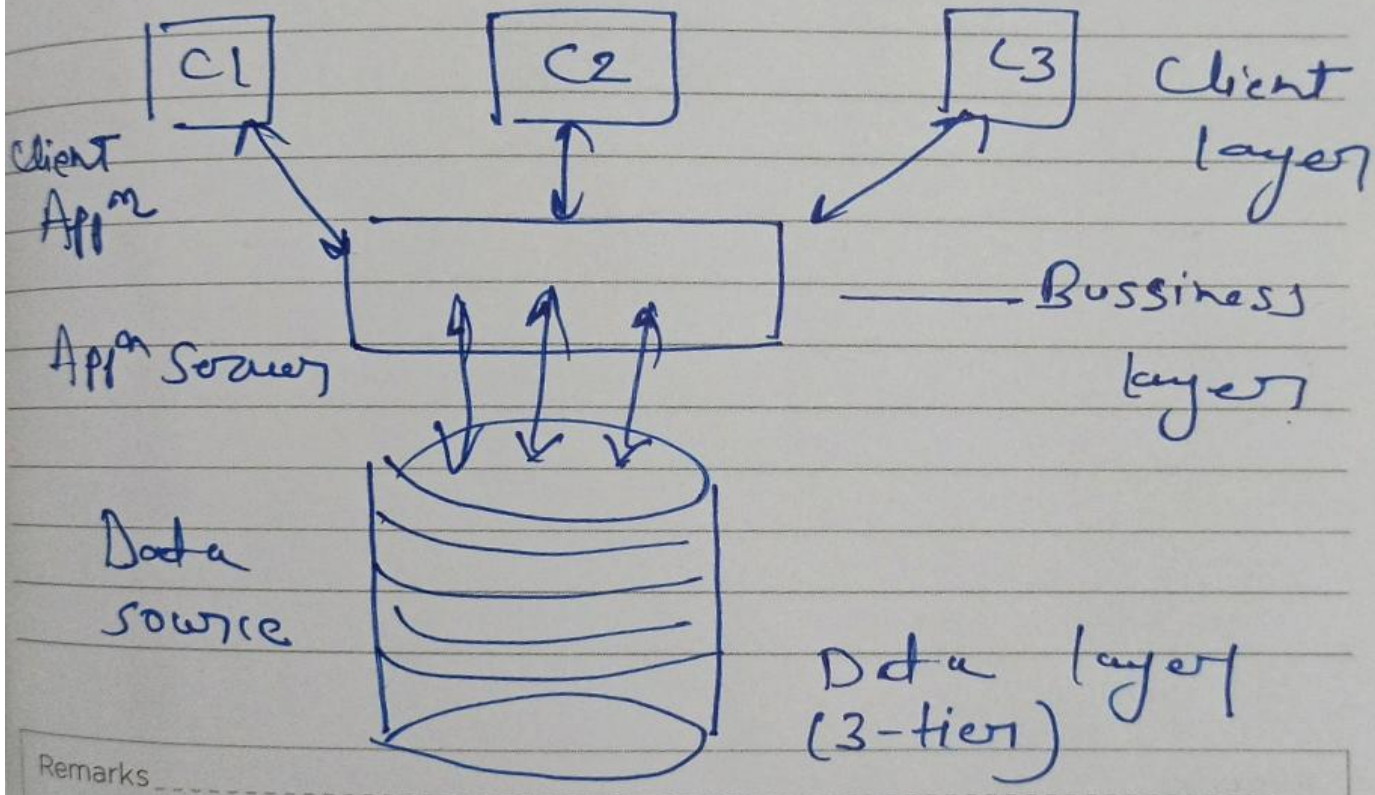
Date __/__/_____

Advantage: 1. Scalability.

2. Security

(user did not direct connect with database)

Ex: like gmail, IRCTC web etc.



Remarks

Disadvantage: low maintenance.

Basic SQL Queries

SQL → Structured Query Language

↳ used for **storing**, **manipulating** and **retrieving** data in database.

SQL commands are mainly into four catagories as —

1. **DDL** — Data definition language
2. **DQL** — Data Query language
3. **DML** — Data Manipulation language
4. **DCL** — Data Control language

Remarks

SQL uses certain commands like **Create, Drop, Insert, etc.**

Date __/__/_____

Notes

SQL Commands

- **DDL**
  - Create
  - Drop
  - Alter
  - Truncate
- **DML**
  - Insert
  - Update
  - Delete
  - Call
  - Explain Call
  - Lock
- **TCL**
  - Commit
  - Savepoint
  - Rollback
  - SET (Transaction)
  - SET (Constraint)
- **DQL**
  - Select
- **DCL**
  - Grant
  - Revoke

CREATE – This command is used to create the database or its objects (like, table, index, function, views, store, procedure, and triggers).

Remarks

Syntax:

```
Create table table_Name ( column1 datatype, column2 datatye ... );
```

DROP

Used to drop an existing table in a database.

Syntax: Drop table table_name;

Alter: used to add, delete, or modify columns in an existing table.

Syntax: Alter table table_name ADD column_name datatypes.

Difference b/w drop and truncate

- drop: used to remove the whole database or table indexes.
- truncate: remove all rows from table.

Remarks

Date __/__/_____

Limit Query (Clause)

↓

used to set an upper limit on number of tuples.

↓

must be a non-negative integer.

Top Clause

↓

used to specify the no of records to return.

Syntax:

Select Top number column-name(s) from table-name where condition;

eg: Select * from customer where country = 'Germany';

Remarks

LIMIT 3;

MIN() AND MAX() fn

→ min() → smallest value of the selected column.

→ max() → return largest value of the selected column.

Syntax:

eg: Select MIN(column-name)
from table-name
where condition;

eg: Select MAX(column-name)
from table-name
where condition;

COUNT(), AVG() and SUM()

- COUNT() → return the no. of rows that matched specified criterion
- AVG() → return avg value of a numeric column
- SUM() → total sum of numeric column.

Remarks

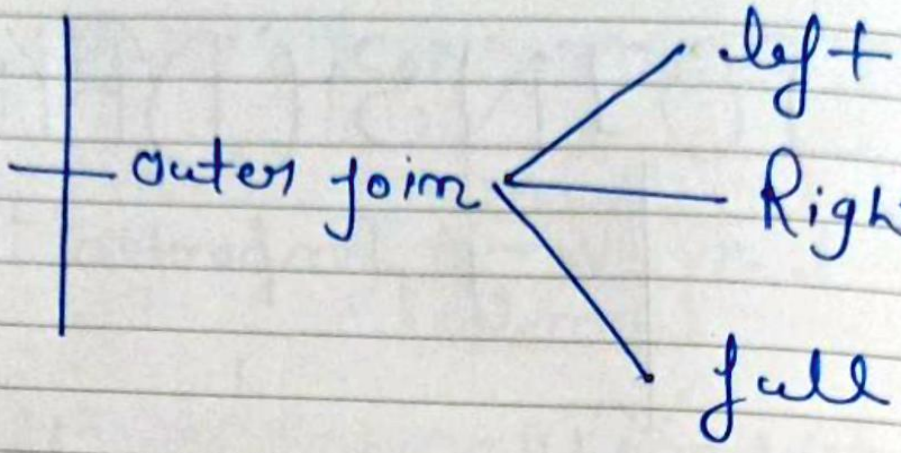
JOINS (DBMS)

Very Important

like the name, it sound to join two or more than two tables in a database to find a result that we can't get from one table.

Types of join:

- Cross Join
- Natural Join
- Conditional
- Equi
- Self Join



Now I explain by taking an example —

E_No	E_name	Add
1	Ram	Delhi
2	Varun	Chd
3	Ravi	Chd
4	Amrit	Delhi
5	Nitin	Noida

Primary key

Employee

DepNo	Name	E No
D1	HR	1
D2	IT	2
D3	MRKT	4
D4	finance	5

Foreign Key

Department

from this get only employee detail.

Remarks

Notes

find Employee name of employee who is working in HR department.

from this we can see that E_name is in employee table and HR in department table

So we get answers using both table with the help of joins !!!

Key Point : In both table atleast one attribute is common.

Join = Cross product + Select (statement)

Remarks

Natural Join {Imp}

Start from example —

take two table — ① Employee

② Department

E.No	E_name	Add
1	Ram	Delhi
2	Varun	chd
3	Raui	chd
4	Amrit	Delhi

Dep No	Name	Eno
D1	HR	1
D2	IT	2
D3	MRKT	4

Notice : first see or find

↓

Our answer/output belong to which table

↓

Emp/Dept

↓

Start write with query

Remarks .....

.....

.....

.....

Query: Select Employee name who belong to delhi.

How to write query ??

first employee belong to Employee table.

→ It is clear our ans get from employee table.

⇒ Select E-name from Emp where Add = 'Delhi';

Que: find the Emp Names who is working in a department.

by read the Que

clear there is two table

Emp & Department

→ So get used join here

Remarks

Date__/__/_____ No

Output:

Process of Natural Join

First write what you show in output —

Select E name from Emp, dept

Cross product

where Emp.Eno = Dept.Eno ;

Output : Ram
Varun
Amrit

Direct Query using Natural join

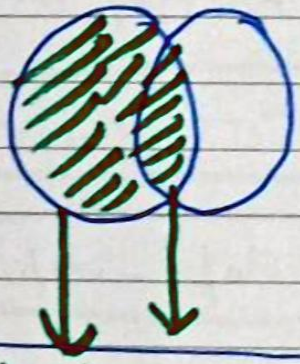
Select E-name from Emp Natural join Dept.

Remarks

Left Outer join

⇒ It gives the matching row and the rows which are in left table but not in right table.

Using Venn diagram ——



left + common part data of table

Date__/__/_____ No

eg:

Emp_no	E_name	Dept_No
E1	Varun	D1
E2	Amrit	D2
E3	Rani	D1
E4	Nitin	—

Dept_No	D_name	Loc
D1	IT	Delhi
D2	HR	Hyd
D3	Finance	Pune

Que: Query is given find output.

Select emp_no, e_name, d_name, loc

from Emp left outer join

dept On (Emp.dept_no = dept_dept_no)

Output

Emp_no	E_name	d_name	Loc
E1	Varun	IT	Delhi
E2	Amrit	HR	Hyd
E3	Rani	IT	Delhi
E4	Nitin	—	—

Remarks

left table

↓ Employee

So all attribute of employee that we print come and common in both that also print from left and right table.

Right Outer Join (weightage to right)

⇒ It gives matching rows and the rows which are in right table but not in left table.

Que: Query is given — find output

Select emp-no, e-name, d-name, loc, room emp Right outer join dept ON (emp.dept-no = dept.dept-no)

Natural join query.

Date__/__/------ No

Emp-No	E-name	Dept-No
E1	Varun	D1
E2	Amrit	D2
E3	Rani	D3

⇒ Employee

Dept-No	D-name	Loc
D1	IT	Delhi
D2	HR	Hyd
D3	Finance	Pune
D4	Testing	Noida

⇒ Department

Remarks

otes

Date__/__/_____

Output :

Common + Extra in right

Emp-No	E-name	Dname	Loc
E1	Varun	IT	Delhi
E2	Amrit	HR	Hyd
E3	Ravi	Finance	Pune
–	–	Testing	Noida

Remarks

Acid Properties

Transaction is a single logical unit of work which modify the content of a database.

In order to maintain consistency in dbms before and after the transaction certain properties are followed. These are called ACID properties.

Remarks

Notes

Key point : As a developer these [property] should be [hold]. If you used [transaction management] system.

A ⟶ Atomicity (either all or none)

C ⟶ Consistency (before the transaction start and after trans. completed → sum of money) ↓ should be same.

I ⟶ Isolation (always talk about parallel transaction)

D ⟶ Durability

↓

All [changes] inside [database] will be / should be [permanent].

Remarks

Atomicity

let say transaction T1

R(A)
A = A - 50
W(A)
R(B)
R(C) .....

If our transaction failed at any point just before commit, then we will do rollback

commit → update in database

roll back → that is used to undo the transactions that have not been saved in the database.

Remarks

Note: A failed transaction cannot be resume it will restart always.

eg: Atom

Consistency

like transaction T_A and T_B

$T_A = 2000$

$T_B = 3000$

T_1
R(A) 2000
A = A - 1000
W(A) 1000 → update in local memory
R(B) 3000
B = B + 1000
W(B) = 9000 → local memory
Commit → saved in database

Remarks

now in **inside database**

A = ₹ 1000
B = 9000

What is **consistency** mean here ??

Before Transaction total of
A and B = 2000 + 3000
= 5000

After transaction = 1000 + 9000
= 5000

Before transaction = After transaction
= 5000

⇓

this is consistency.

Remarks

Notes

Date __/__/____

Isolation:

↳ multiple transaction at same time.

↓

can be converted it into serial schedule not in real time but conceptually

like: T1 ⇄ T2 → Parallel

T1 → T2
T2 → T1 → Serial

↓

always Consistent.

Durability:

updates now become permanent and are stored in non-volatile memory, effect of transaction never lost

Remarks

Schedule In (DBMS)

It is a chronological execution sequence of multiple transaction.

types

- Serial
- Parallel

Serial → one by one

like T_1, T_2, T_3

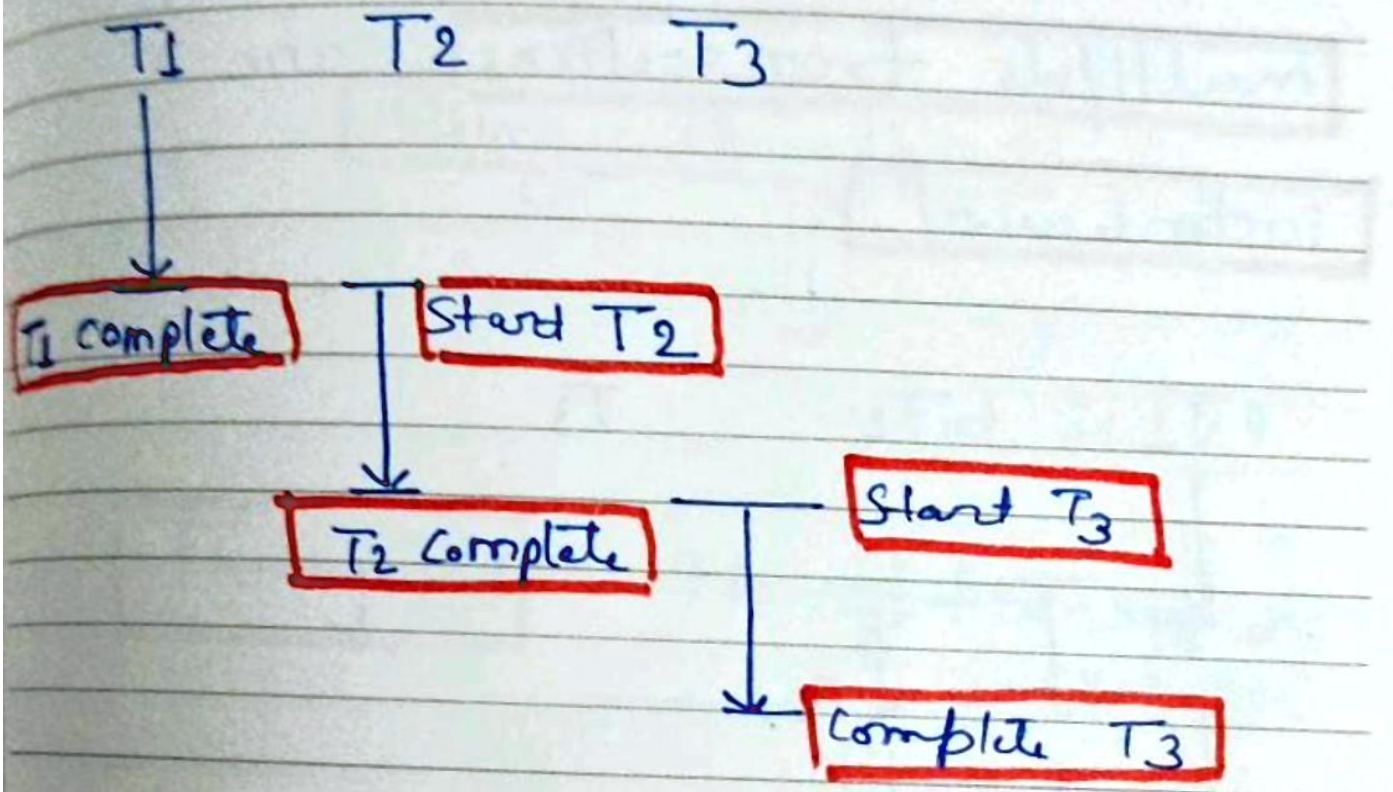
start → first T_1 is start then

Remarks

until it is not completed T_2 can't start. After

T1 execute completely

then and then T2 will start.



Advantage → Consistency, Secure

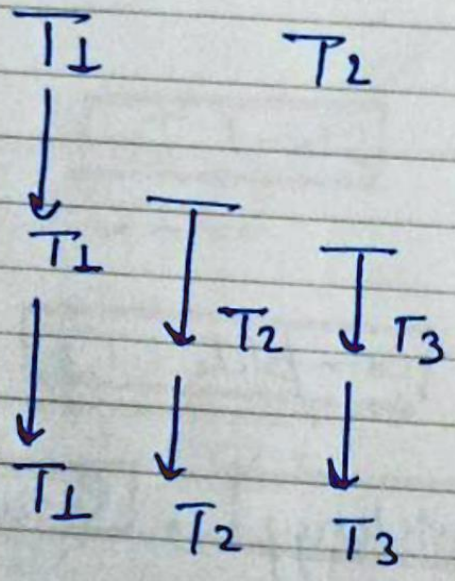
Problem → waiting time

Remarks: Real life example → ATM life there are 3 people in a queue after first then second will start.

Parallel Schedule

⇓

where the **operations** of **multiple transactions** are **interleaved**.



multiple transaction **executed** at the **same** time.

Remarks

Real life ex: **SBI online**

Advantage — Performance

Increases ← How ??

↓

Using Throughput

⇓

what is ??

Throughput = No. of transactions executed / time.

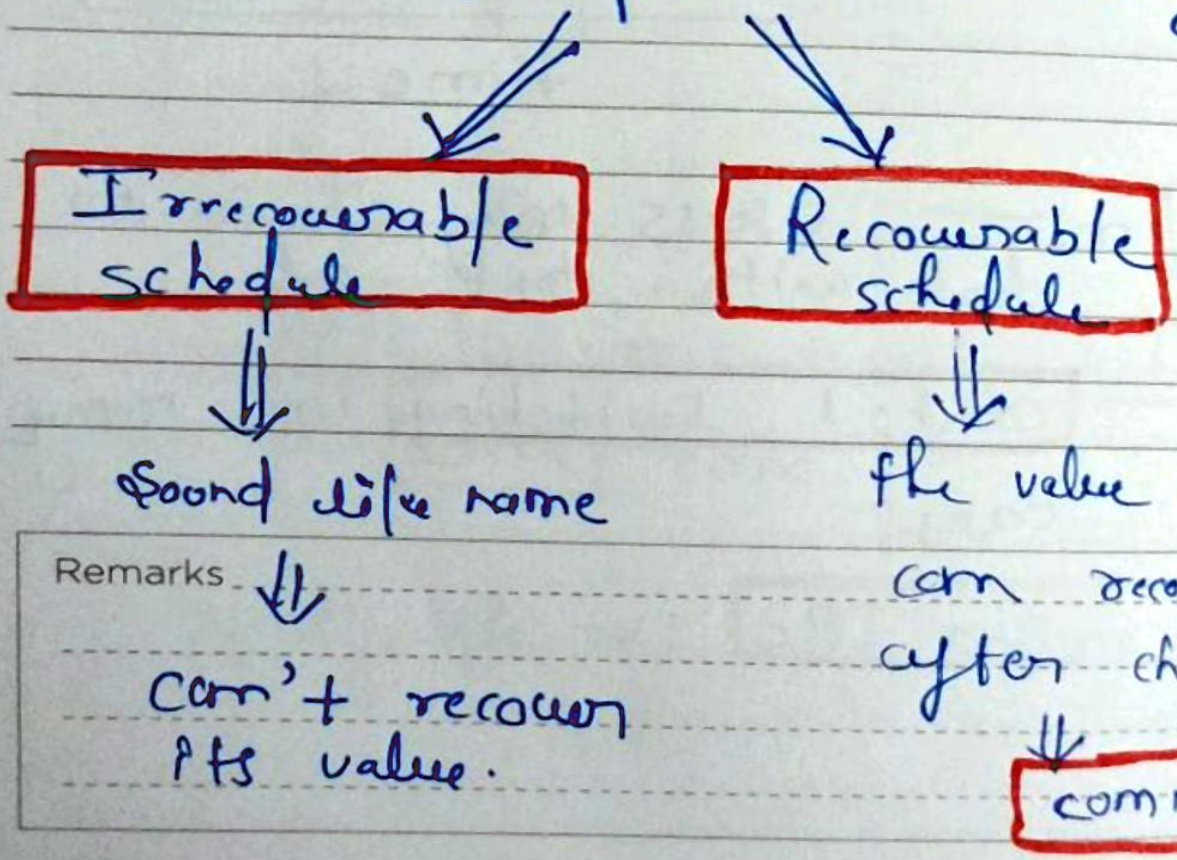
Problem — users may have to deal with tons of context switching in many cases.

Remarks

Context Switching:

Storing the **state of process** so that it can be **reloaded** when **required** and **execution** can be resumed from the **same point** as earlier.

Inside parallel scheduling



Remarks

Can't recover its value.

Irrecoverable issue

↳ explain from example -

T1	T2
10 R(A)	
5 A=(A-5)	
5 W(A)	→ R(A) 5
	A = A-2 = 3
	W(A) 3
	commit
	↳ save changes.
R(B)	↓
*	A=3
fail	

given
A = 10
B = 20

↳ transaction fail at point B, then due to atomicity property it will do rollback

↓

undo

Remarks

Again value of A
||
10

Date__/__/_____ Note

The changes done by T2 is lost and can't be recoverable.

Remarks .....

.....

.....

.....

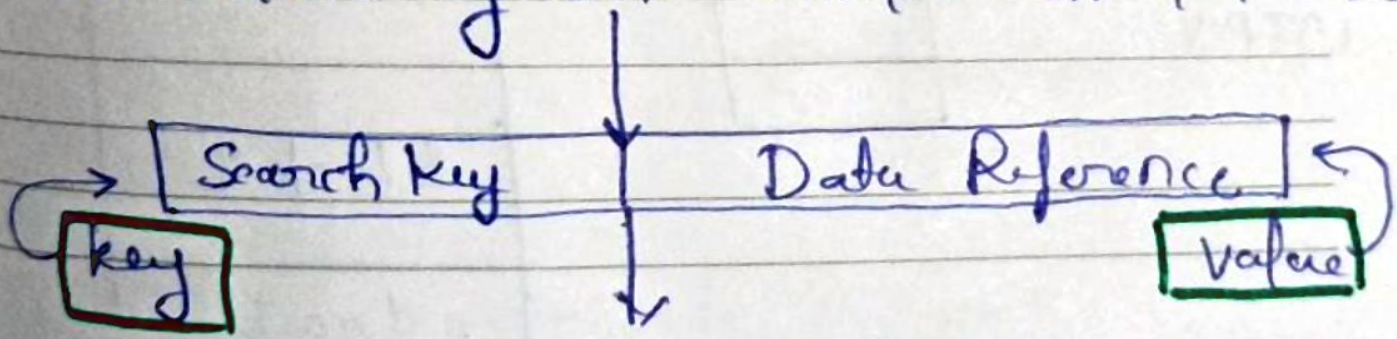
Date __/__/_____

Indexing in DBMS

What is ??

It is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed.

Structure of an Index in database

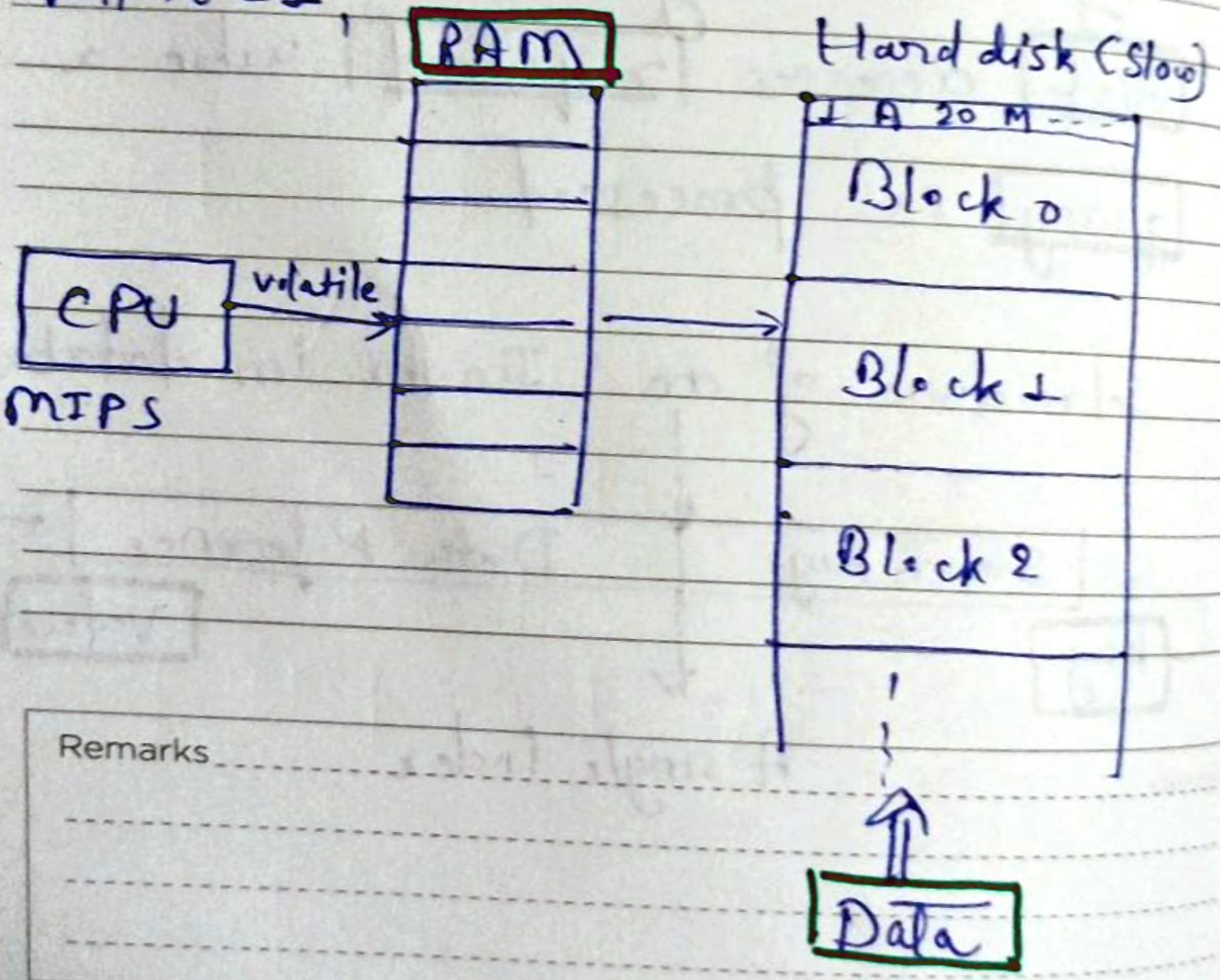


Remarks

Why Indexing is used ??

Take a query in which we find student data whose roll no = 1.

⇒ Select * from student where Roll no = 1;



Remarks

So Data is in Harddisk

⇓ CPU interact with

RAM (Random access memory)

⇓ call data from second memory (Hard-disk)

↳ divide into blocks/Page

⇓

collection of boxed called Hard-disk

⇓

inside block put records.

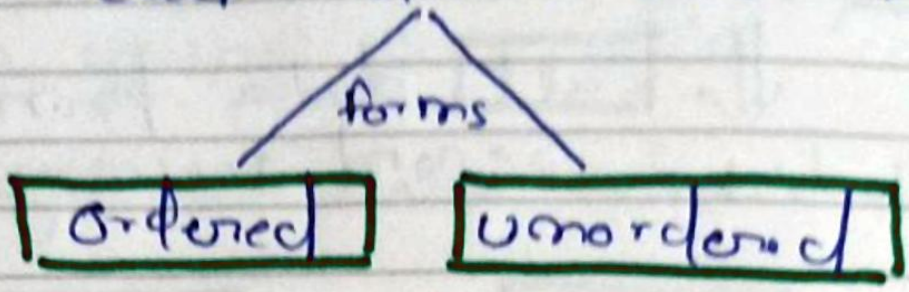
eg: 10000 records (given)

1 BS = 100 records

No. of blocks = $\frac{10000}{100} = 100$ required.

How it works ??

⇒ Call data via CPU



If

⇒ Our data is unordered then we calls every block and search for our data so I/O cost increases.

eg: book without indexing in pages.

⇒ If our data is ordered then it reduces the number of block and I/O cost decreases

Remarks