



# Normalization T-ND

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 an example like

	SID	Sname	Age
Remarks	1	Shivika	20
	2	Vamika	25
	3	Tanika	20
	4	Shivika	20

These two rows are same

Date \_\_\_/\_\_\_/\_\_\_\_\_

Not



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

So we set SID, no two rows will have same SID 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
---	---------	----	------	----	------	-----

Remarks

These columns are same.

here we have three types of problem —

notes

- 1. Insertion anomaly
- 2. Deletion anomaly
- 3. Updation anomaly
- Insertion



For previous table if we add new course like Pharmacy we can't insert it.

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

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

This is called Insertion Anomaly.

Remarks



## • Deletion

from previous table,

If we delete SID = 2

So simply: Delete from student  
where SID = 2

now two will b. 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

Notes

## • Updation

from previous table,

update sname —

set sname

So query: UPDATE student <sup>^ where</sup>  
sname = 'Tomika Patel'  
where SID=2.

No problem here.

But if we change salary of FL from  
300 to 900 then

in this, it changes all FL  
300 to 900

but you want to change only  
Tomika Patel faculty salary

but here it changes all FL so  
it creates problem.

Remarks

Causes due to column level  
duplicacy.

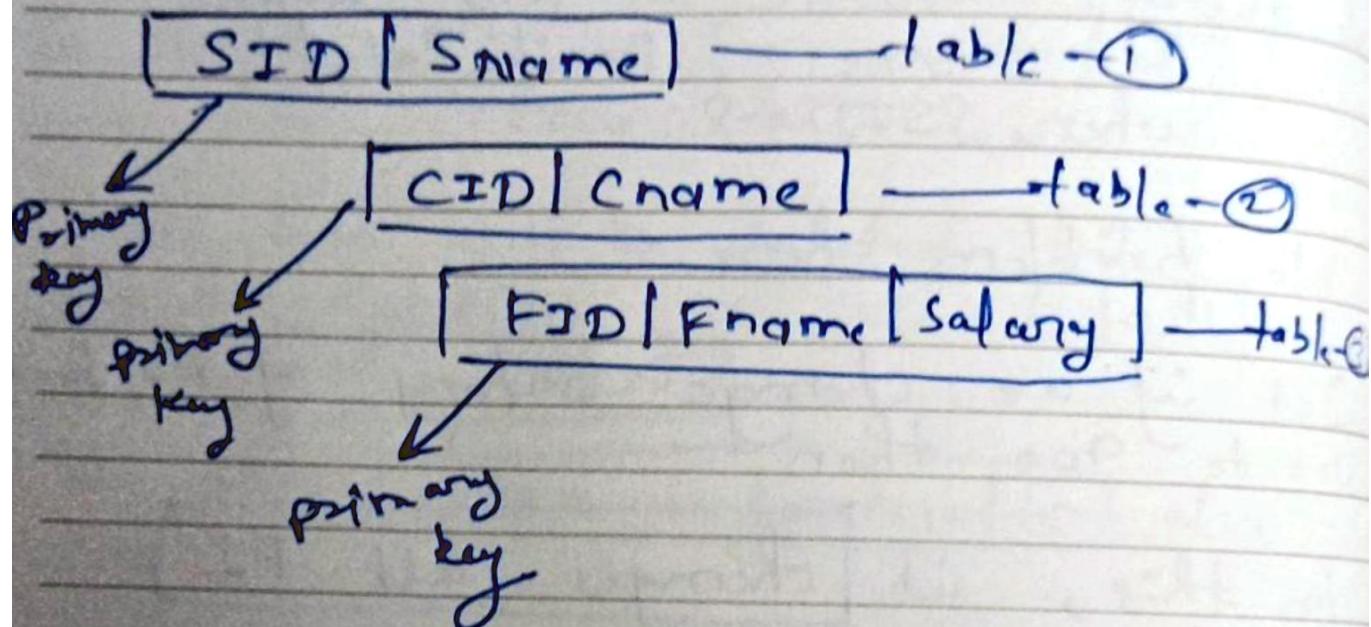
Date \_\_\_/\_\_\_/\_\_\_\_\_

Note



## How Normalization help form these ??

We can break our table



Remarks

notes

# Summary of all Normal forms



## JNF (First Normal form)

→ If a table contains multiple values  
    of one attribute Then  
    ⇒ Not a normal form.

Condition for JNF:

1. Only single value attribute

like:

Roll No.	Course
1	DBMS
2	C N

It is in JNF

Roll No.	Course	Remarks
1	DBMS, C	
2	C N, C++	Invalid not a JNF

~~Key Point~~  
~~JNF follows atomic rule~~

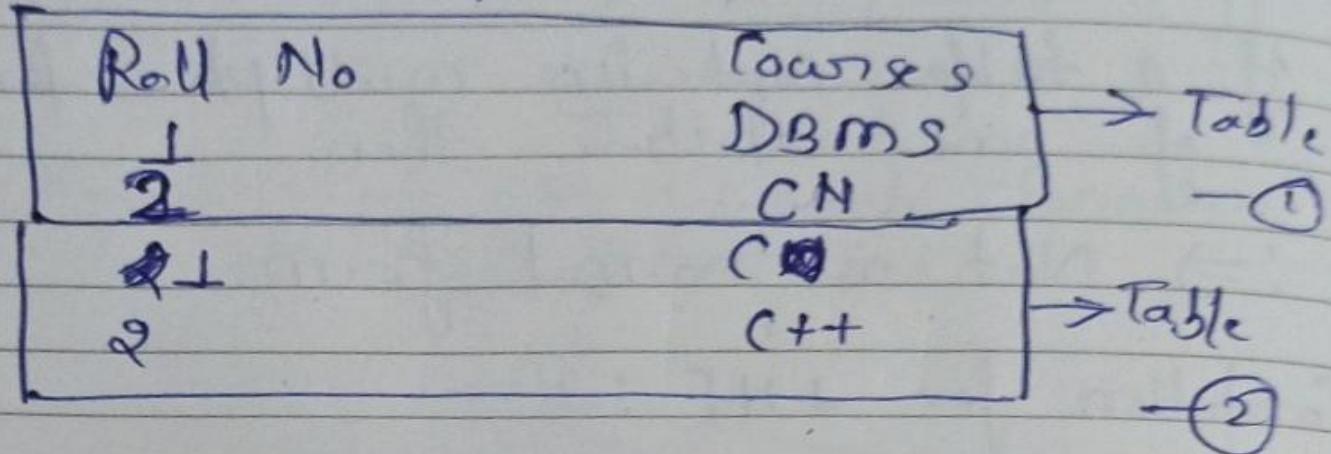
Date \_\_\_/\_\_\_/\_\_\_\_\_

Note

Solution for that:



If we have multiple attributes  
then break table like



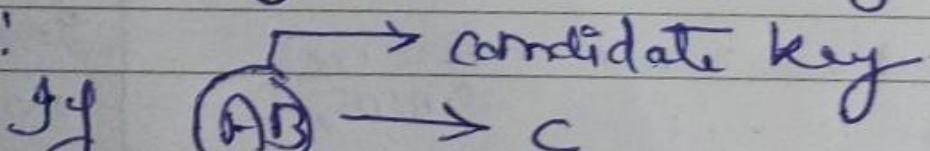
2NF (Second Normal form)

First cond<sup>n</sup>: must be in 1NF

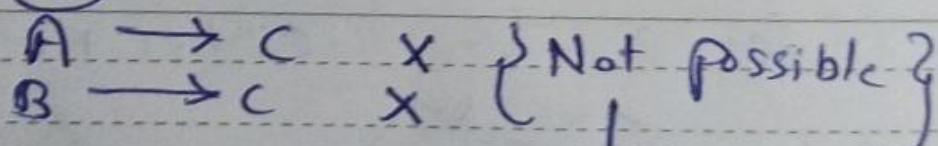
② Partial dependency (CNO\*)

③ Only full dependency

like:



Remarks



$\times \quad \times \quad \} \text{Not possible} \}$

Partial dependency

Notes

## 3NF (Third Normal form)

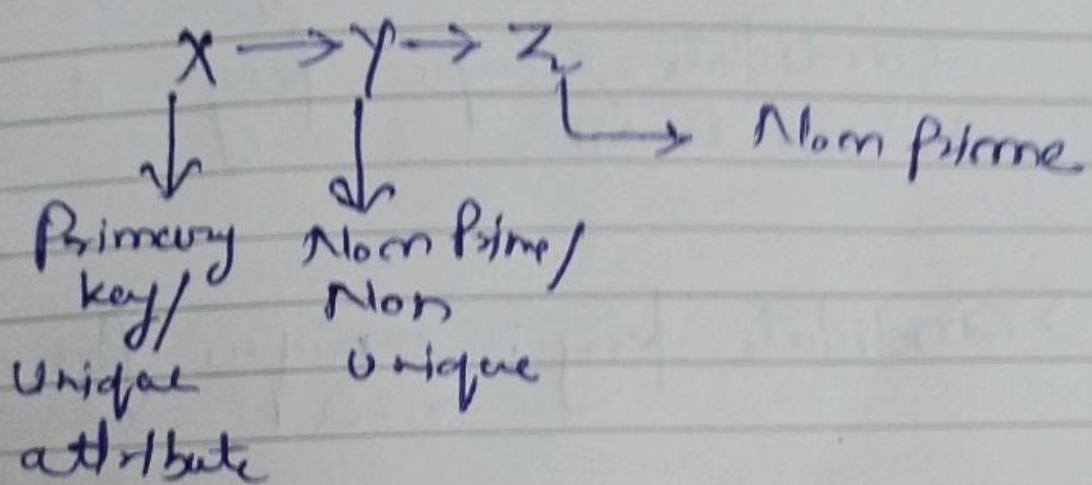


Condition :

① must be in 2NF

② No transitive dependency

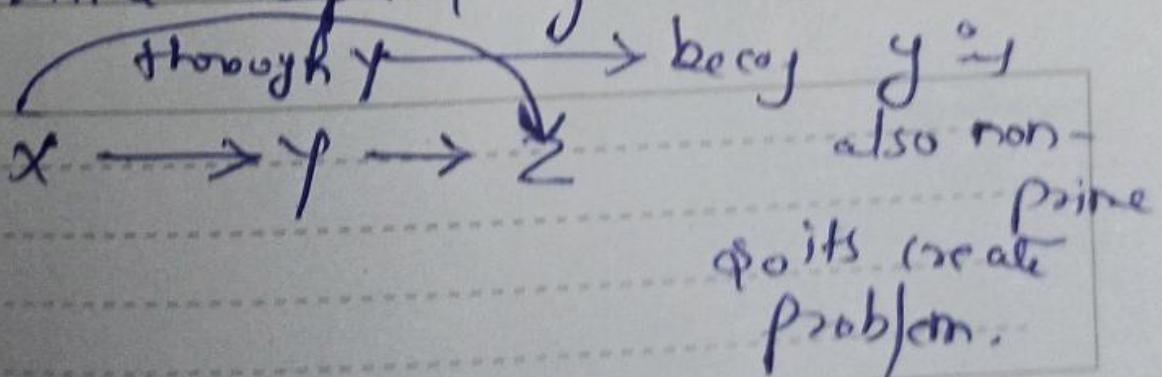
means



+ Non-prime attribute Z, Y

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

Remarks





## BCNF (Boyce-Codd)

Condition:

① must be in 3NF

LHS (<sup>+</sup> left hand side of attribute) always be

Candidate / Super key don't  
is in BCNF.

⇒ candidate key: unique for every row

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

Remarks.....

Notes

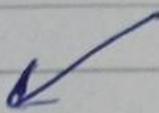
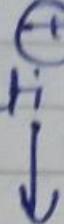
Date \_\_\_\_\_ / \_\_\_\_\_ / \_\_\_\_\_

4NF (Fourth Normal form)



Conditions: must be in BCNF

No multi valued dependency



Shivani → 3 Mobile No  
→ 3 Email |

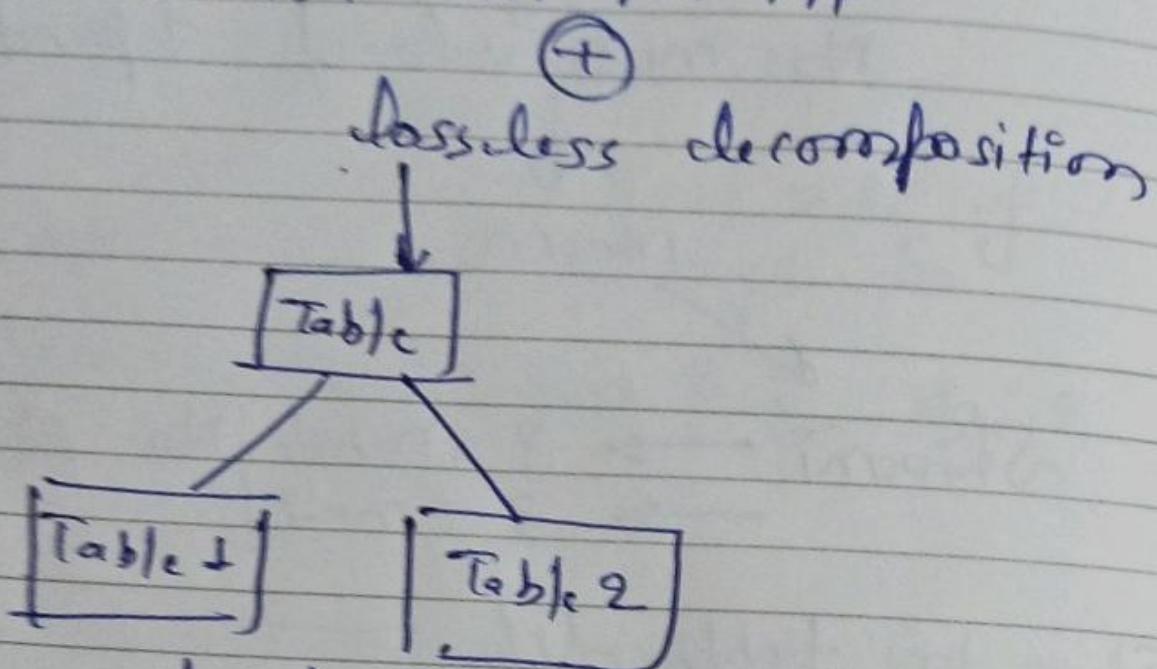
Store in table like —

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



divide in vertical ( $\infty | ^m$ ) 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

# 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

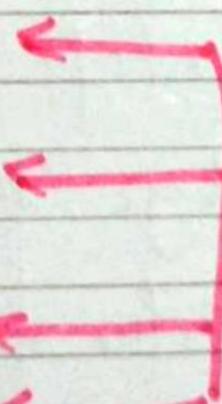
- Candidate key

- Super key

- Foreign key

- Alternate key

- Composite key



These four  
are very  
important.

- Artificial key

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 come's  
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 achieve

unique + Not NULL

Date \_\_\_/\_\_\_/\_\_\_\_\_

Note



## Candidate key

Key → attribute

use of key → uniquely identify

Name	City	Age
------	------	-----

Shivani	Kanchipuram	20
---------	-------------	----

Somya	Lucknow	21
-------	---------	----

Prachi	Delhi	28
--------	-------	----

Shivani	Kanchipuram	20
---------	-------------	----

Same

Take ex. of student data table.

Adhar Card	—
------------	---

Roll no	—
---------	---

Registration	—
--------------	---

Licence No.	—
-------------	---

Remarks	Mobile No	—
---------	-----------	---

	email	—
--	-------	---

All are diff<sup>n</sup>  
for all every  
student

This are  
true set of candidate  
key



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

## Foreign key

If is an attribute or set of attributes that references to primary key of same table or another table of (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

Course Id	Course Name	Roll no
C1	DBMS	1
C2	Networking	2

~~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 is **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).



## 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 = Roll no

if we add

Roll no, name

Roll no, age

Roll no, name, age

Remarks

then they will be

super key.

Roll no Name age



Ques:  $R(A_1, A_2, A_3, A_4 \dots A_n)$

then how many super keys are possible.

$\Rightarrow A_1$  is candidate key.

$\Rightarrow A_1, A_2$  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

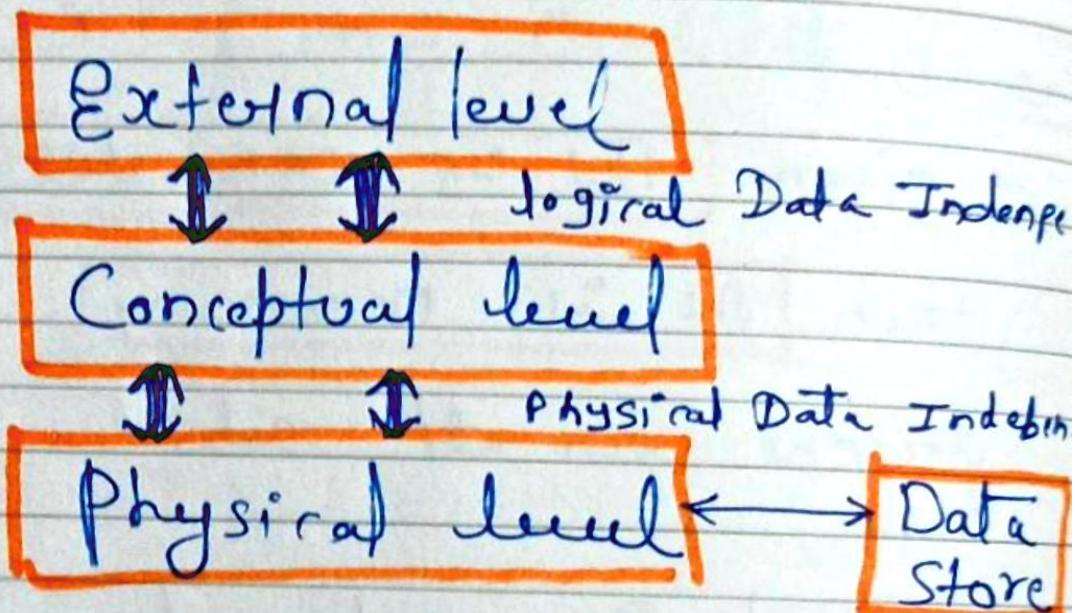
Remarks

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



# DBMS 3-tier Architecture

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



## Data Independence:

Remarks

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



## Physical data Independence:

- Any **change** in the **physical location** of **table** and **indexes** **should not affect** the **conceptual** **internal** 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 layers.

Client layer

Database Server

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

Remarks

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

notes

⇒ Two-tier also known as

**Client - Server Architecture**



live example —

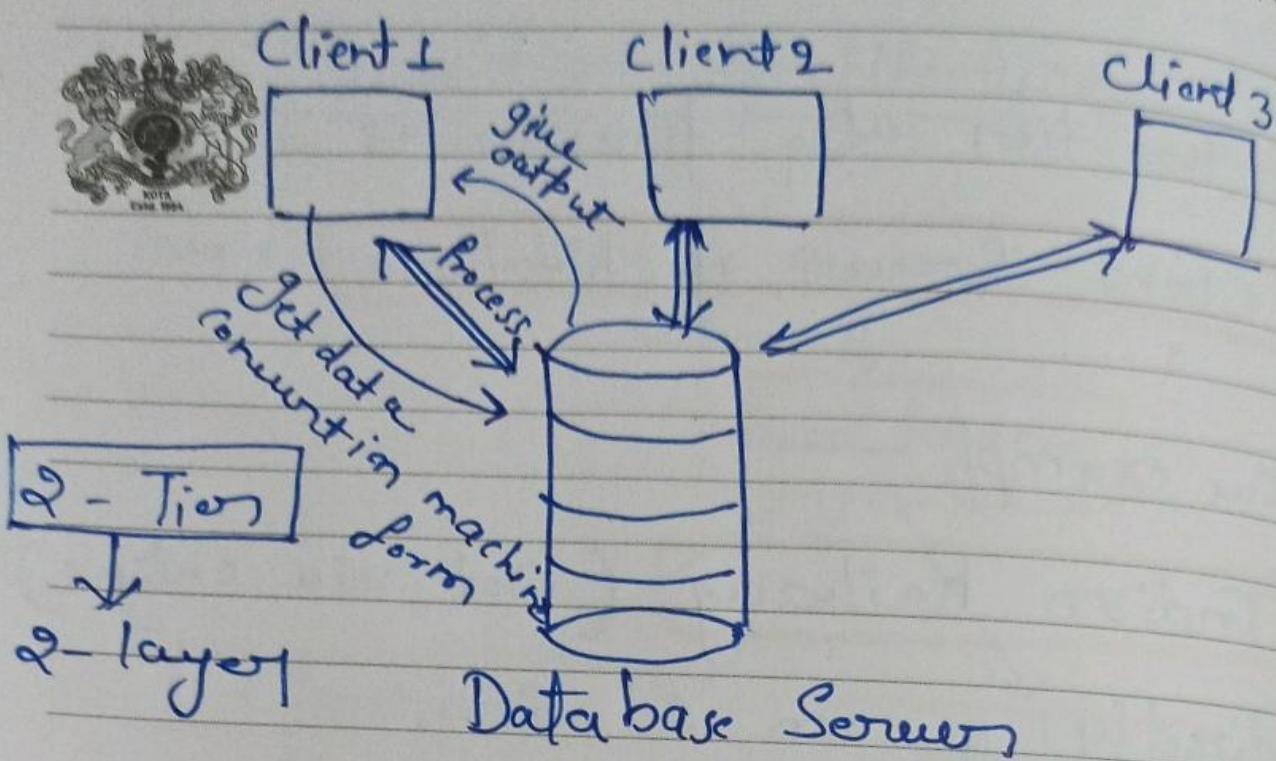
**Indian Railway** (Not via online)

directly go on station.

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

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

Remarks



## Advantage

- two layer  $\Rightarrow$  maintenance easy
- limited client limited database

## Disadvantage

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

\* So at that time 2-tier fail.



Security → here client  
direct interact  
with database  
so if 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.



⇒ Data layer:

At this tier, data layer contains 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 gives direct output bcoz all query verification work done by business layer.

Remarks

⇒ Do load of database server is less.

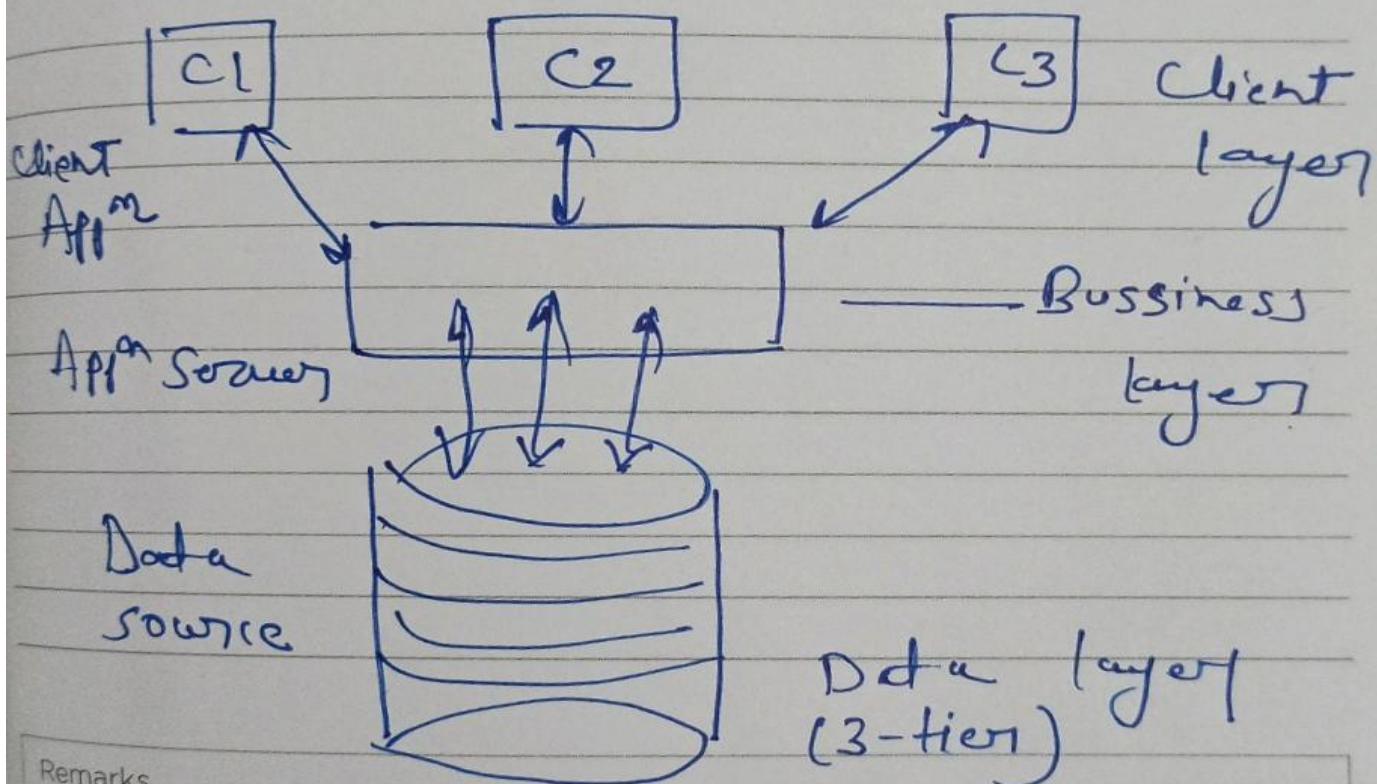


Advantage : 1. Scalability.

2. Security

(User did not direct connect with database)

e.g. like gmail, ICRTC web etc.



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 categories as →

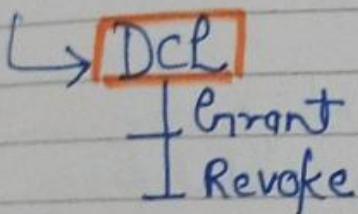
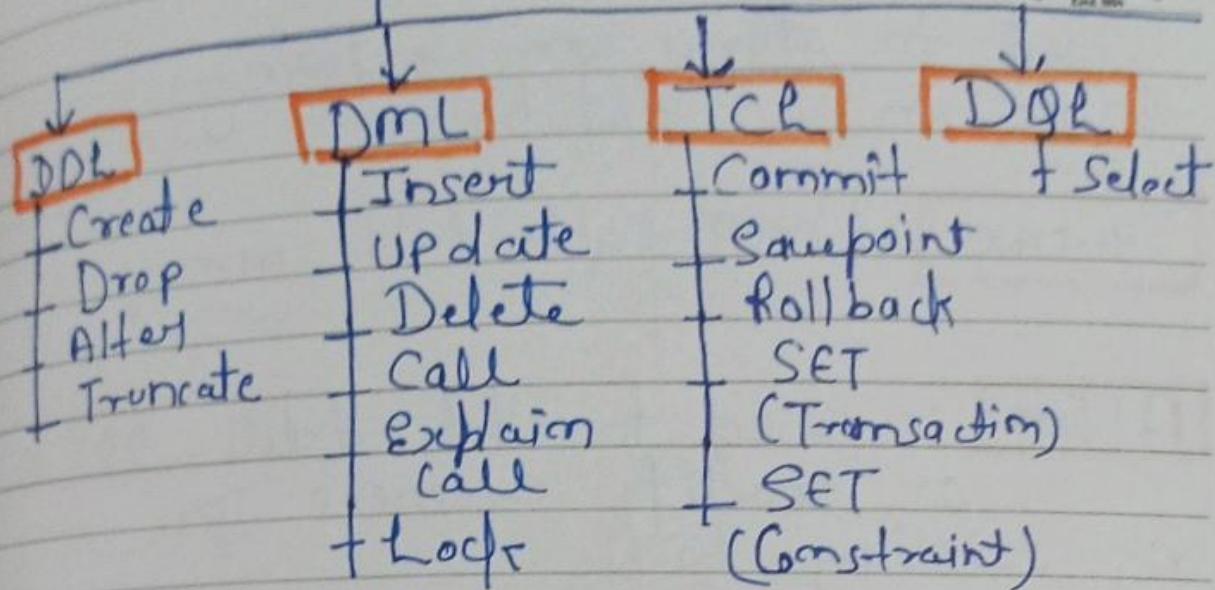
1. **DDL** - Data definition language
2. **DQL** - Data Query language
3. **DML** - Data Manipulation language
4. **DCL** - Data Control language

Remarks

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



# SQL Commands



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

Remarks

## Syntax:

```
Create Table Table_Name ( column1  
datatype, column2 datatype ... );
```



## 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 datatype.

## Difference b/w drop and truncate

Used to remove the whole database or table indexes.

Remarks

remove all rows from table.



## Limit 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;

e.g.: Select \* from customer  
where country = 'Germany'

Remarks

LIMIT 3;



## MIN() AND MAX() fn

→  $\text{min}()$  → smallest value of the selected column.

→  $\text{max}()$  → return largest value of the selected column.

**Syntax:**

eg: Select  $\text{min}(\text{column-name})$   
from  $\text{table-name}$   
where  $\text{cocondition}$ ;

eg: Select  $\text{max}(\text{column-name})$   
from  $\text{table-name}$   
where  $\text{cocondition}$ ;

**COUNT(), AVG() and sum()**

$\Downarrow$ return the no. of rows that matched specified criterion	$\Downarrow$ return avg value of a numeric column	$\Downarrow$ $\rightarrow$ total sum of numeric column
--	---	---



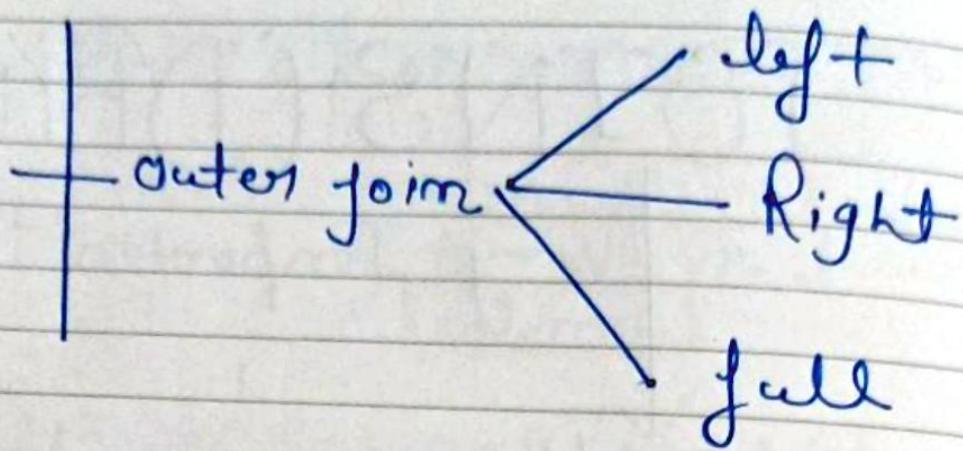
# JOINS (DBMS)

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 on foreign key

example —

E-No	E-name	Add	DepNo	Name	E No
1	Ram	Delhi	D1	HR	1
2	Karan	Chd	D2	IT	2
3	Ravi	Chd	D3	MRKT	4
4	Amrit	Delhi	D4	Finance	5
5	Nitin	Noida	D5		

Employee

Department

Remarks

from this get

only employee  
detail.

Notes

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

From this we can  
see that Ename is  
Employee Table  
and  
in  
table  
HR in department

So we get answer 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 tables — ① Employee  
② Department

E.No	E-name	Add	Dep No	Name	E.no
1	Ram	Delhi	D1	HR	1
2	Varun	Chd	D2	IT	2
3	Ravi	Chd	D3	MKT	4
4	Amrit	Delhi			

Notice : first see or find

Our answer/output  
belongs to which table

Emp / Dept

Start write with query

Remarks

Notes

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

clears there is two table

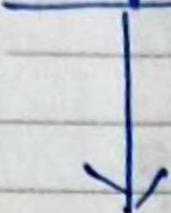
Remarks

Emp & Department

→ So get used of join here



Output:



Process of Natural  
Join

First write what you show  
in output —

Select E-name from Emp, dept

↓  
**Cross Product**

where Emp.Eno = Dept.E.no ;

Output: Ram

Vasan

Amit

Direct query using Natural join

Select E-name from Emp Natural  
join Dept.

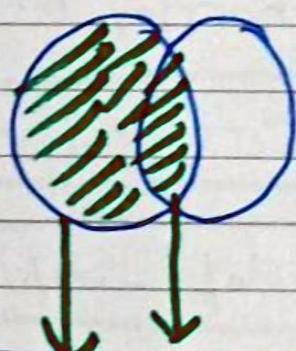
Remarks .....

## 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

Remarks

Date \_\_\_/\_\_\_/\_\_\_\_\_

No.



eg:

Emp-no	E-name	Dept-No	Dept-No	D-name	Loc
E1	Varun	D1	D1	IT	Delhi
E2	Amit	D2	D2	HR	Hyd
E3	Ravi	D1	D3	Finance	Pune
E4	Nitin	-			

Ques: Query is given find output.

Select emp-no, e-name, d-name, loc  
 from the 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	Amit	HR	Hyd
E3	Ravi	IT	Delhi
E4	Nitin	-	-

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 (waitage to right)

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

Ques: Query is given — find output

Select emp-no, e-name, d-name, loc, from 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	Amit	D2
E3	Ravi	D3

⇒ Employee

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

⇒ Department

Remarks

otes

Date / /

Output :



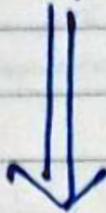
Common + Extra in right

Emp-No	E-name	Dname	Loc
E1	Varun	IT	Delhi
E2	Amrit	HR	Mys
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 \_\_\_\_\_

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)

I → Isolation  
(always talk about parallel  
transaction) should be same.

D → Durability

All **changes** inside **database**  
will be / should be  
**Permanent**.



## Atomicity

Let say transaction T<sub>1</sub>

R(A)

A = A - 50

W(A)

R(B)

R(C) - . . . . .

If own 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]

Remarks: been [saved] in the  
[database.]

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

$$R(B) 3000$$

$$B = B + 1000$$

$$W(B) = 9000 \rightarrow \text{local memory}$$

commit

→ saved in database

Remarks



Now in inside database

$$A = 1000$$

$$B = 9000$$

What is consistency mean here ??

Before Transaction total of

$$A \text{ and } B = 1000 + 9000 \\ = 10000$$

After transaction =  $1000 + 9000 \\ = 10000$

Before transaction = After transaction  
= 10000

↓  
This is consistency.

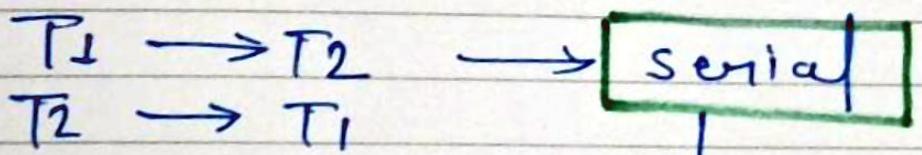
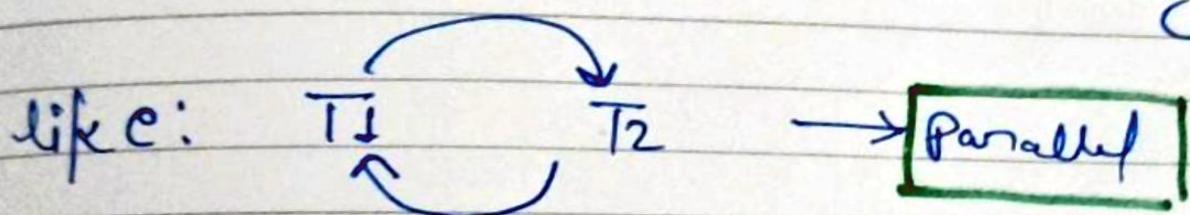
Remarks



## Isolation :

↳ multiple transaction at some time.

↓  
can be coupled it into  
serial schedule not in  
real time but conceptually.



↓  
always  
consistent.

## Durability :

Remarks

updates now become permanent  
and are stored in non-volatile

memory, effect of transaction is never lost



# Schedule In (DBMS)



It is a **chronological** execution

**Sequence** of **multiple** transaction.

↓ types

**Serial**

**Parallel**

↓  
**one by one**

↓

like  $T_1, T_2, T_3$

Start →

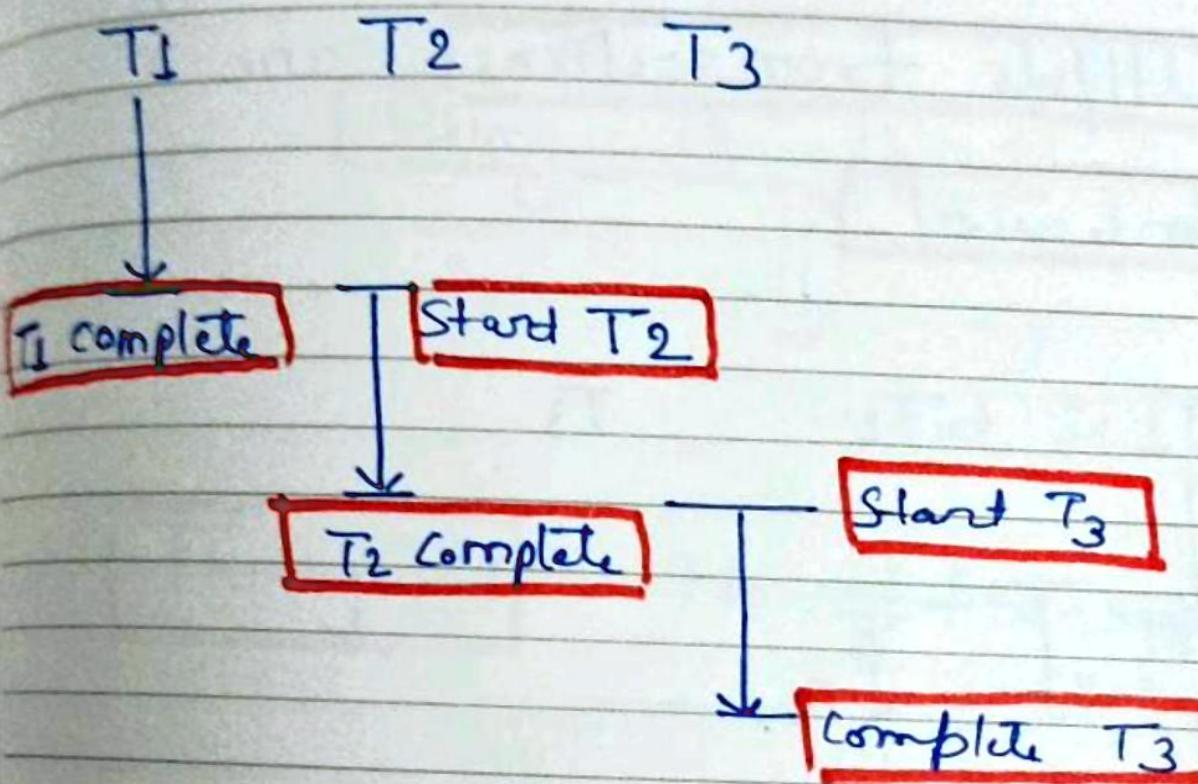
first  $T_1$  is **start** then

Remarks

until it is **not** completed

$T_2$  **can't** start . After

it execute completely,  
then and then T<sub>2</sub> will start.



Advantage → Consistency, Secure  
Problem → Waiting time

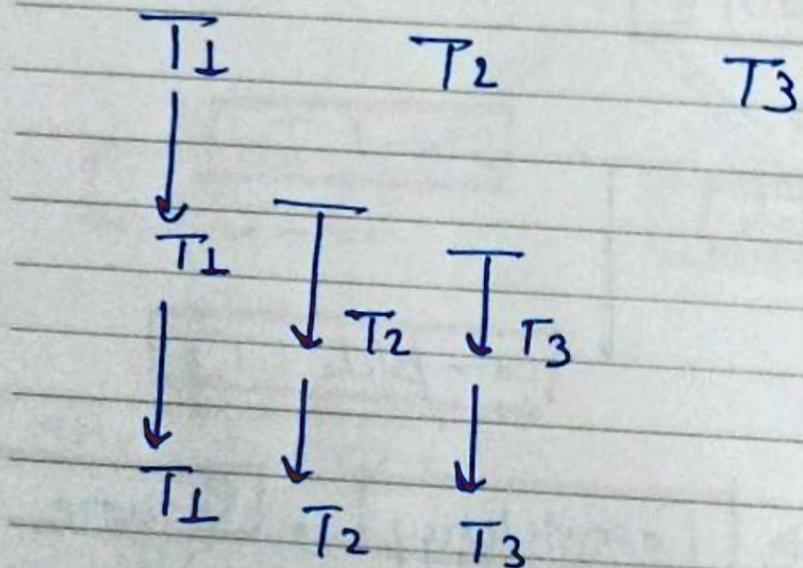
Remarks Real life example → ATM  
like there are 3 people in a queue  
after first then second will start.



## Parallel Schedule



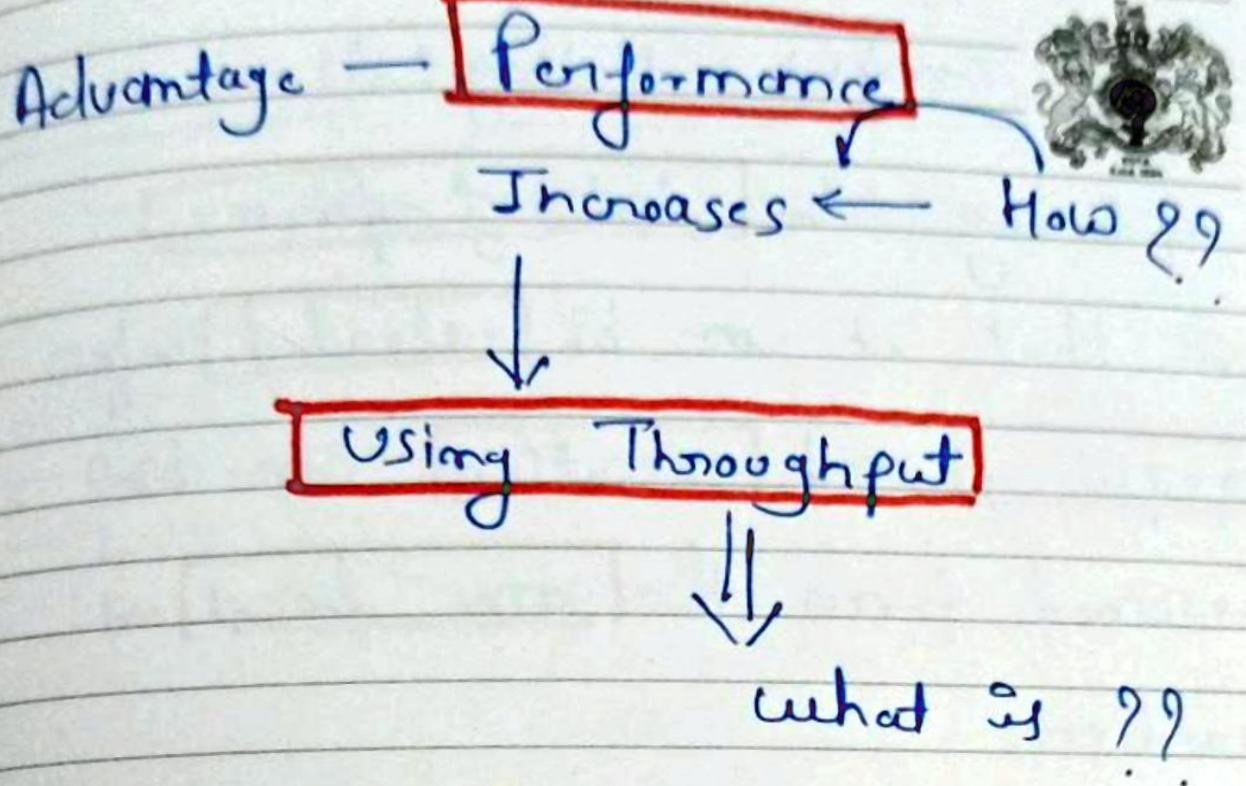
where the **operations** of  
**multiple transactions** are  
**interleaved**.



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

Remarks

Real life ex: **SBI online**



**Throughput** = No. of transactions executed / time.

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

Remarks

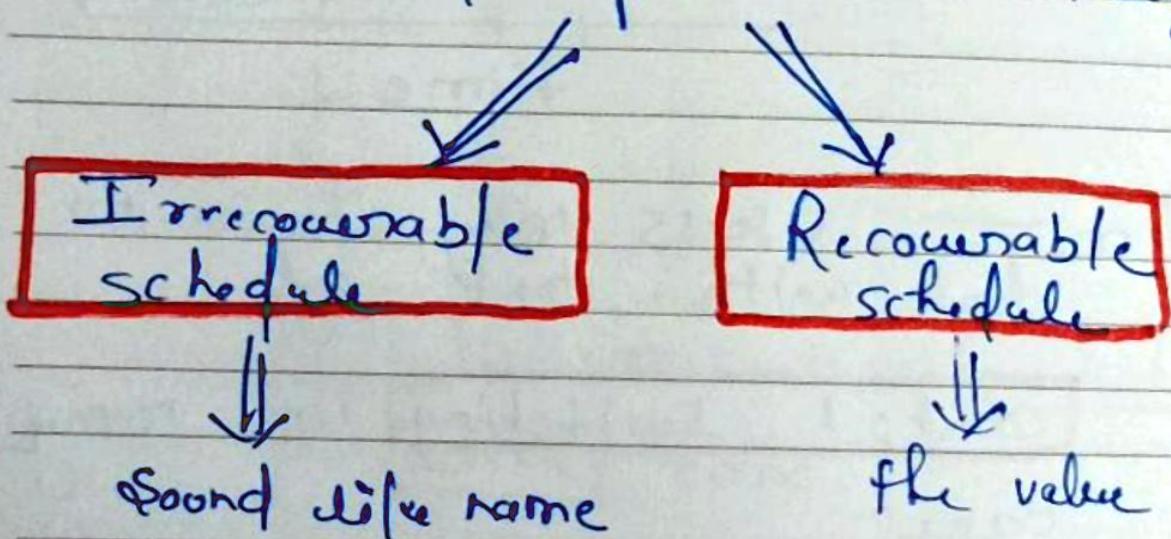


## Context Switching:

of saving 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  
pts value.

(in recover)  
after changes.

↓  
**commit**

## Irrecoverable imm



↳ explain from example -

T<sub>1</sub>  
1. R(A)  
2. A = A - 5  
3. W(A)

T<sub>2</sub>

R(A) 5  
 $A = A - 2 = 3$   
W(A) 3  
commit

R(B)  
\* fail

Given  
A = 10  
B = 20

↳ same changes.

↓  
A = 3

↳ transaction **fail** at point

**B**, then due to **atomicity**,

property it will do **rollback**

Remarks

Again value of A  
||  
10

↓  
**undo**

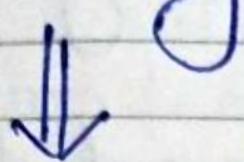
Note



The changes done by T<sub>2</sub>  
is lost and can't be  
recoverable.

Remarks

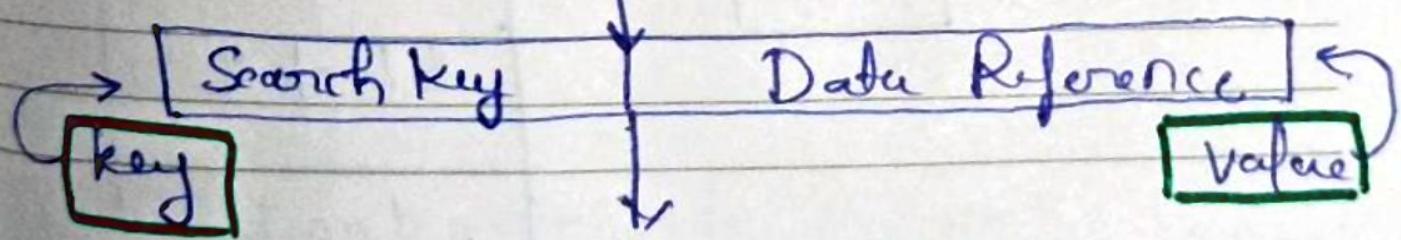
# Indexing in DBMS



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



A single index

Remarks

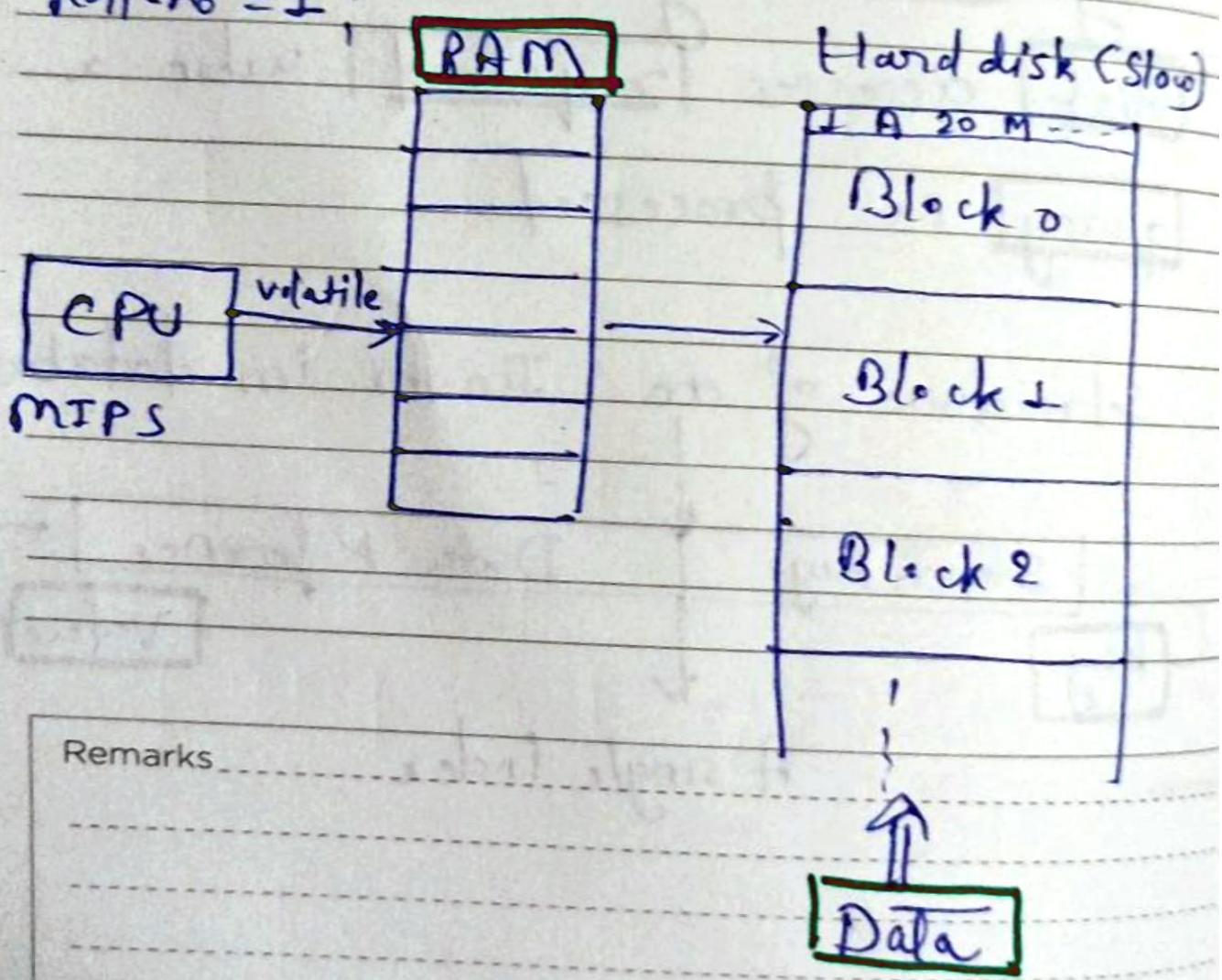


# why Indexing is used ??

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

$\Rightarrow$  Select \* from student where

Rollno = 1;



So **Data** is in **Harddisk**  
↓  
CPU interact  
with  
**RAM** (Random access memory)  
↓, **call** data from  
(Hard-disk) Second memory  
divide into **blocks/Page**



**collection** of boxes called  
**Hard-disk**  
↓  
inside block  
put **records.**

e.g. 10000 records (given)

marks  
1 B S = 100 records

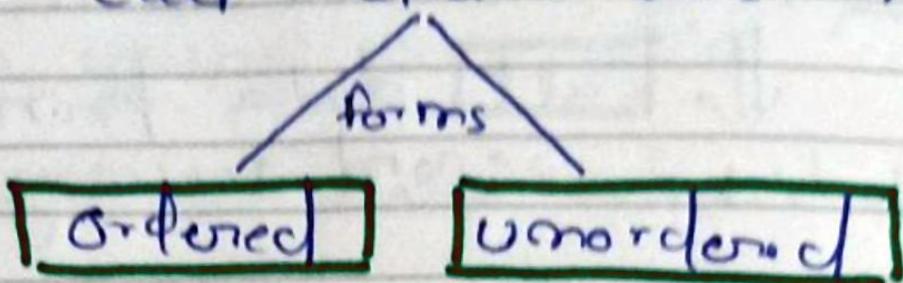
$$\text{No. of blocks} = \frac{10000}{100} = 100$$

↓ required.



# How it works ??

⇒ Call data via CPU



If  
⇒ Our data is unordered

then We call every block  
and Search for own data

So I/O cost increases.

e.g.: book without indexing in  
Pages.

⇒ If own data is ordered

then it reduces the number  
of blocks and I/O cost decreases

Remarks