Made by : Raushan kumar (IIEST,Shibpur(2019-2023))

Gate Smashers Link: [DBMS (Database Management system) Complete Playlist]

Total : 121 videos
Gate smashers Notes :
DBMS(All Videos+Notes)-Gate Smashers is now one stop Solution😇 #shorts#ytshorts

Difference

Basis	DBMS Approach	File System Approach
Meaning	DBMS is a collection of data. In DBMS, the user is not required to write the procedures.	The file system is a collection of data. In this system, the user has to write the procedures for managing the database.
Data Abstraction	DBMS gives an abstract view of data that hides the details.	The file system provides the detail of the data representation and storage of data.
Recovery Mechanism	DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from system failure.	The file system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will be lost.

Concurrency Problems	DBMS takes care of Concurrent access of data using some form of locking.	In the File system, concurrent access has many problems like redirecting the file while deleting some information or updating some information.
Data Redundancy and Inconsistency	Due to the centralization of the database, the problems of data redundancy and inconsistency are controlled.	In this, the files and application programs are created by different programmers so that there exists a lot of duplication of data which may lead to inconsistency.
Data Independence	In this system, Data Independence exists, and it can be of two types. • Logical Data Independence • Physical Data Independence	In the File system approach, there exists no Data Independence.
Integrity Constraints	Integrity Constraints are easy to apply.	Integrity Constraints are difficult to implement in file system.

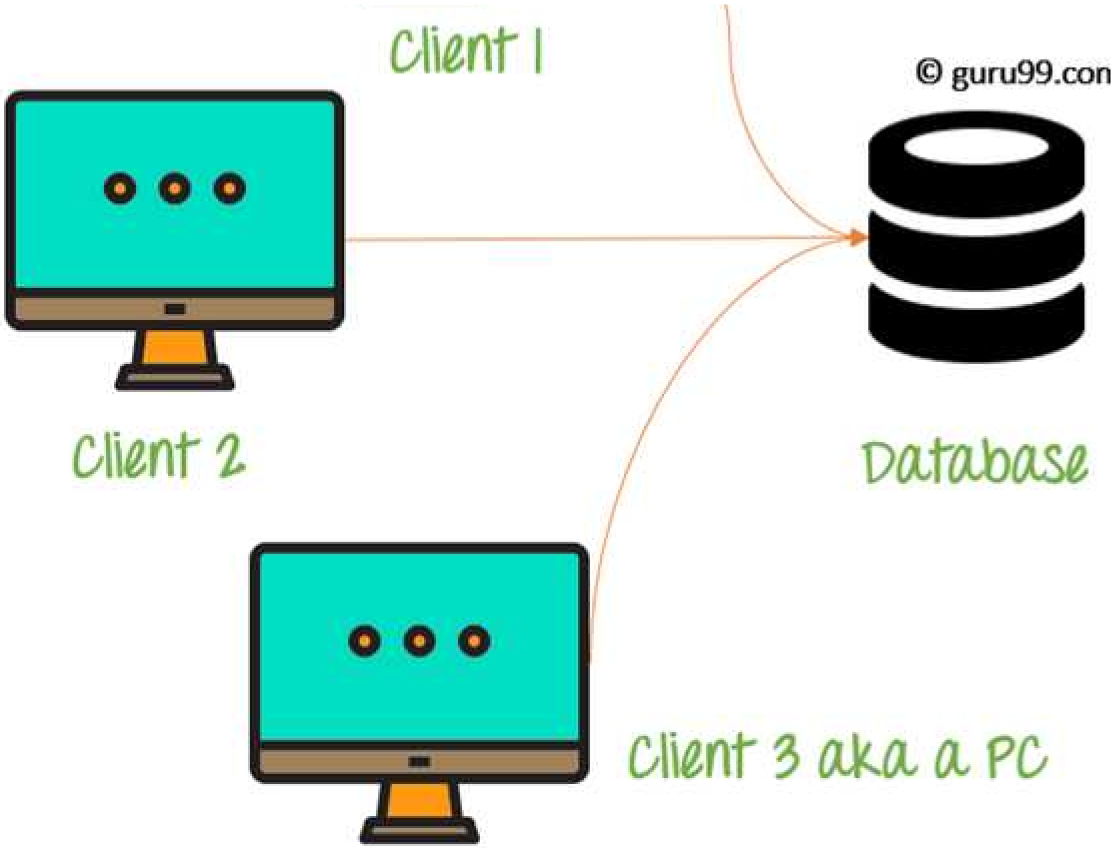
Data Models	In the database approach, 3 types of data models exist: ● Hierarchal data models ● Network data models ● Relational data models	In the file system approach, there is no concept of data models exists.
Examples	Oracle, SQL Server, Sybase etc.	Cobol, C++ etc.

DBMS Architecture

In DBMS there are 3 levels

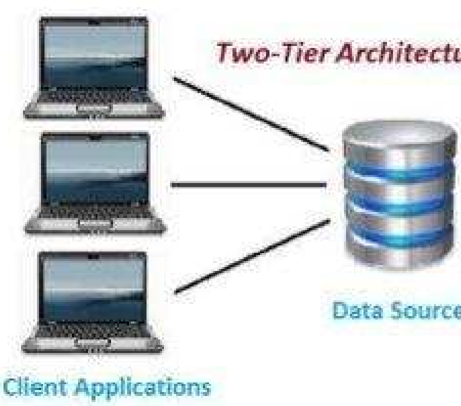
1. External level / View level= describe that is relevant to the users .
2. Conceptual level / Logical level=Describe what data is store in DB and reln among the data
3. Internal level / storage level =Physical representation of the database , how the data is stored in the database . it covers the data structures and file organization

- DBMS can be seen as either Single tier or multi tier
- An n tier architecture divides the whole system into related but independent n modules
    1. 1-tier architecture
       One tier means only one server will be there
       Here communication between client and file server (Database)



2. 2-tier architecture

- It is client-server architecture
- Direct communication
- Run faster(tight coupled)



3 . 3-tier architecture

It follows web based application

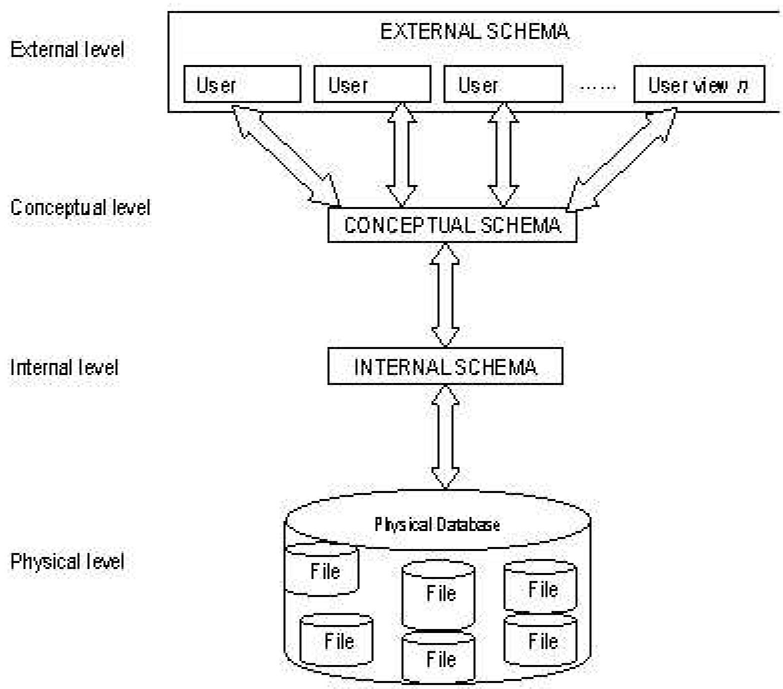
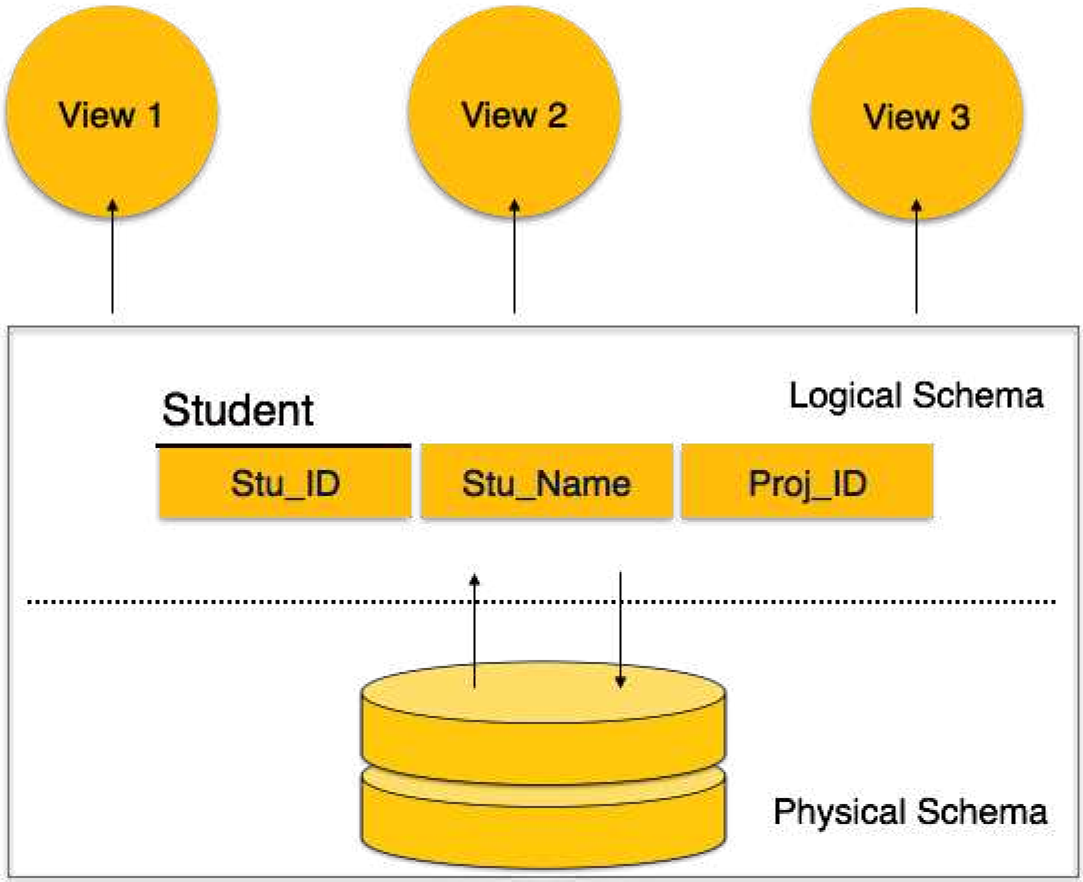


Fig. Three-tier database architecture

3 layers

- Client layer/user(presentation tier) =end user , they don't know beyond this layer
- Business layer /Application server(Abstract view) Its mediator b/w user and db
- Data base layer / Database server (sql)

Schema

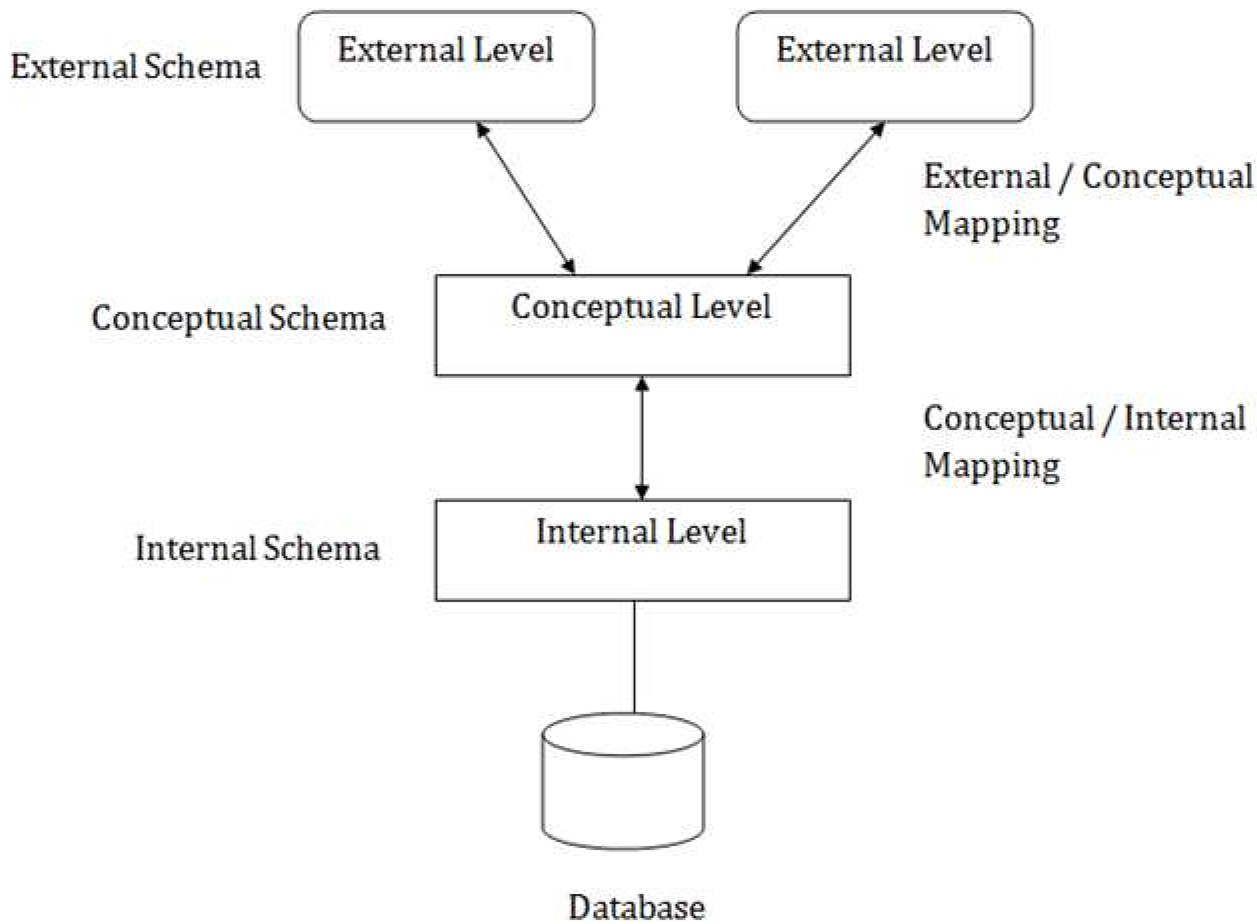


A database schema can be divided broadly into two categories −

- Physical Database Schema − This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- Logical Database Schema − This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Three schema Architecture

Data independence



View level (front end ,executed with the view)
|
Logical data independence
|
Conceptual schema(relation among the table , foregin key , constrains)
|
Physical Data independence
|
Physical Schema
|
Data base

Candidate key

Key is a attribute
It is use to uniquely identify the two tuples
Now the collection of all keys is called candidate key set
{ aadhar card , voter id , license no , roll no , phone no , email}
Now one ek key choose karke usko primary key banate hain and remaining called alternative k

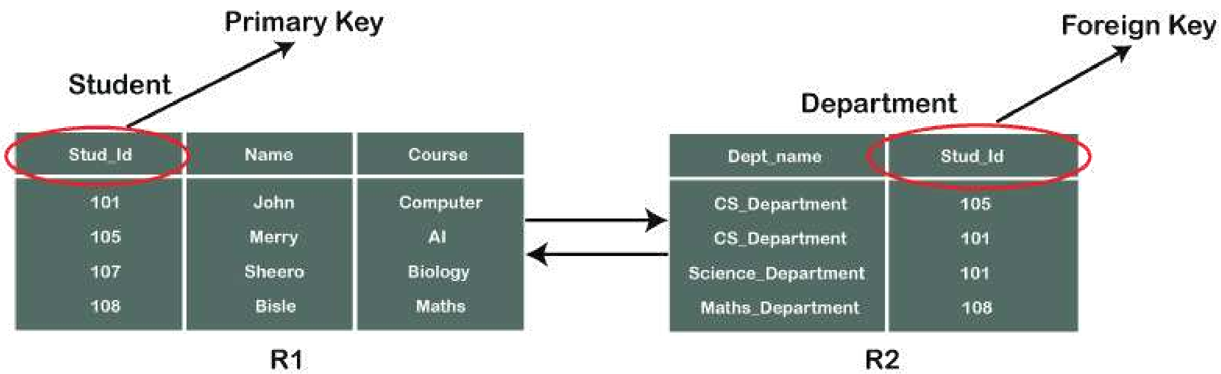
Primary key (Unique + NOT NULL)

Adhaar may be null

Foreign key

It is an attribute or set of attributes that references to primary key of the same table or another table (relation)
→ maintain referential Integrity

Table containing primary key that is called referenced table
Table containing foregin key that is called referencing table



Create table Course
{
Course id varchar(10)
Course_name varchar(10)
Rollno int references student(Rollno)
}
Query after the table is created
Alter table course add constraint fk foreign key (rollno) references student(roll no)

Foregin key can be two in a table but primary key will be one

Lect 11

Referenced table

- Insert no violation
- Delete may cause violation (on delete cascade , on delete set NULL , on delete No action)
- Update : may cause violation

Referencing table

- Insert May cause violation

- Delete no violation
- Updation may cause violation

Super key (lect 13)

Super key : A super key is a combination of all possible attributes which can uniquely identify two tuples in the tables

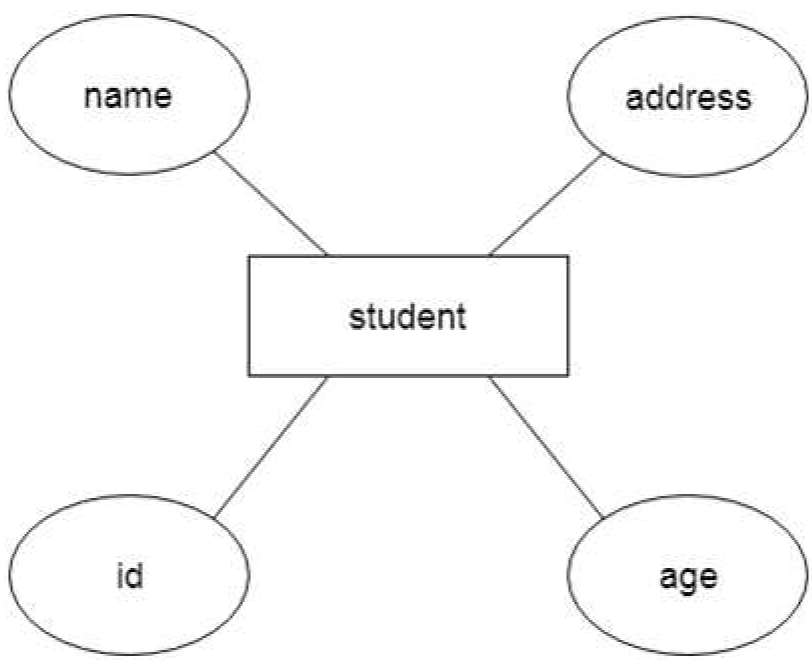
proper subset of any candidate key is Super key

Let Roll no is candidate key
Then {roll no , name},{roll no ,age} , {roll no , name,age} can be super key

If A1 is the candidate key then total possible super key would be 2^(n-1)
If A1 and A2 both are candidate key then total super key will

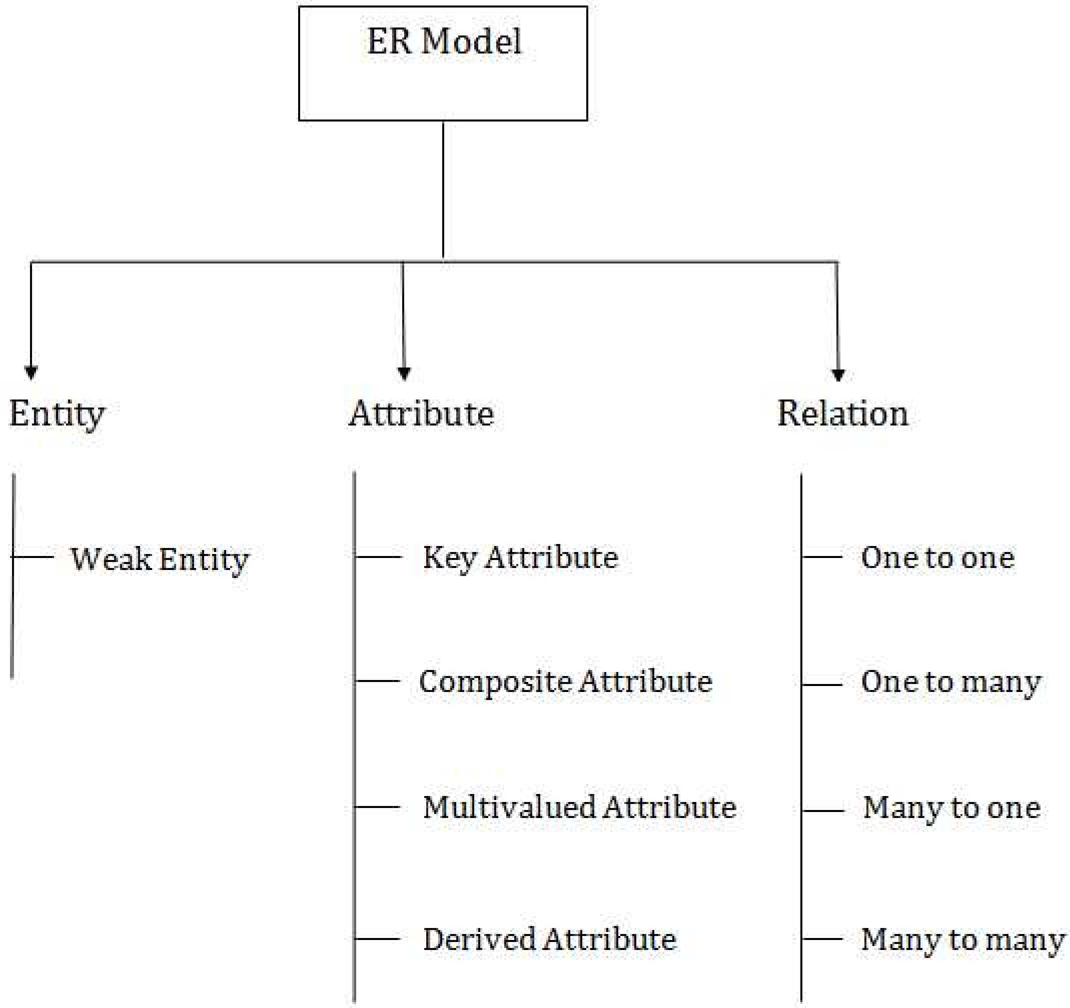
2^(n-1)+2^(n-1)-2^(n-2)(common A1 and A2 are fixed)

ER Model (lect 14)

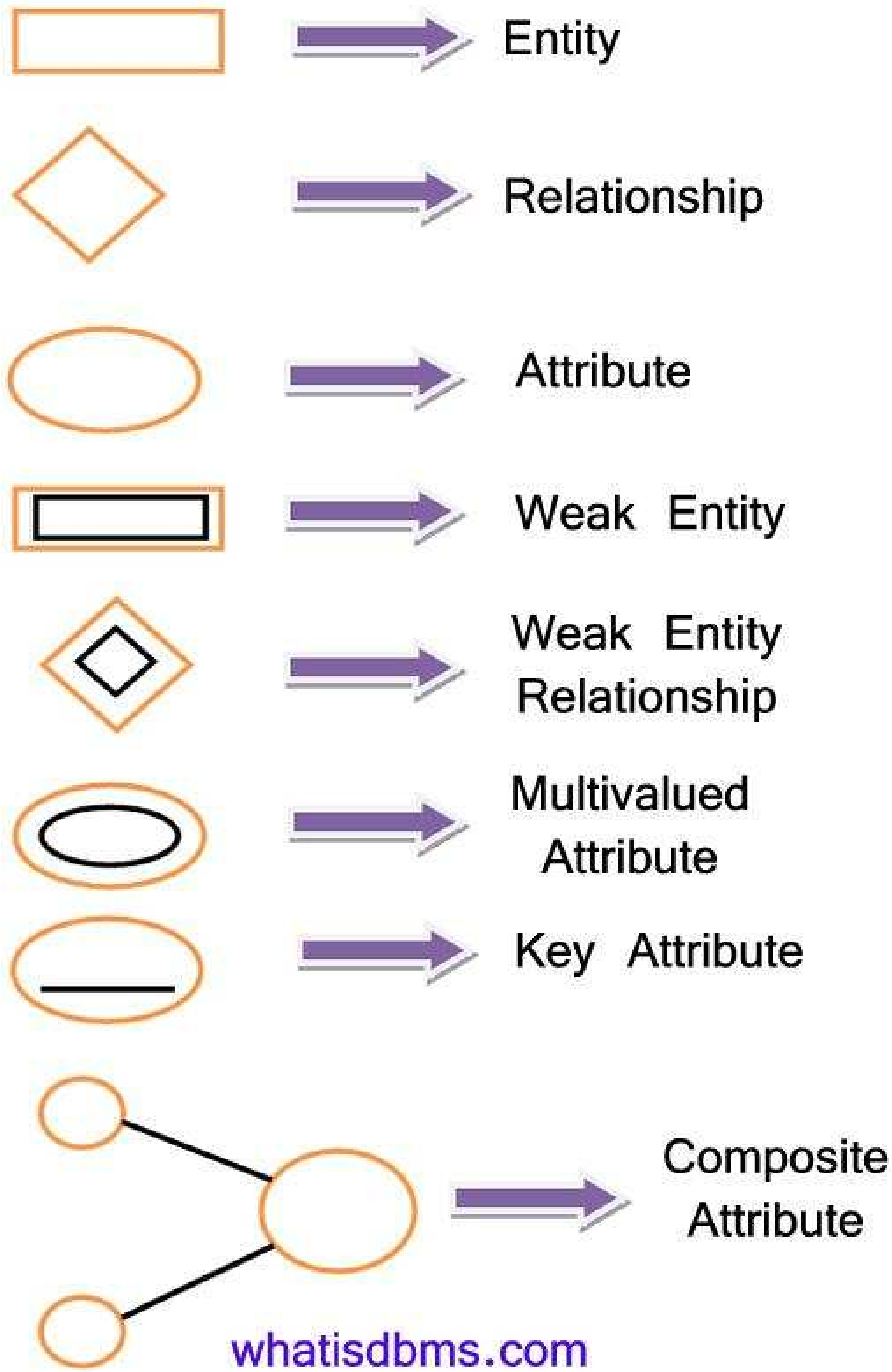


Here student is entity and remaining circle part is attribute

Components of ER models



Representation



Entity
Relationship
Attribute
Weak Entity
Weak Entity Relationship
Multivalued Attribute
Key Attribute
Composite Attribute
whatisdbms.com

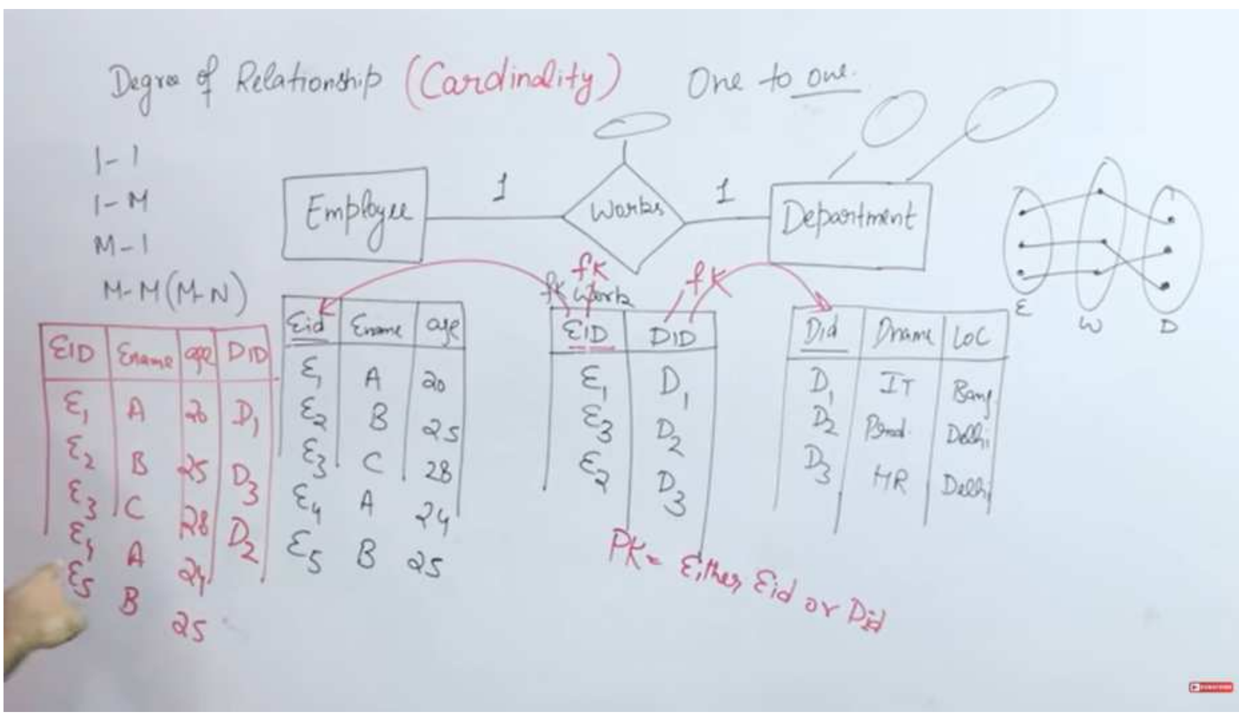
Types of Attributes

1. Single vs Multivalued Attributes
2. Simple vs composite Attributes
3. Stored vs Derived Attributes
4. Key(unique attr) vs non key Attributes
5. Required(mandatory) vs optimal Attributes
6. Complex (composite + multivalued)

Type of relationship(Cardinality)

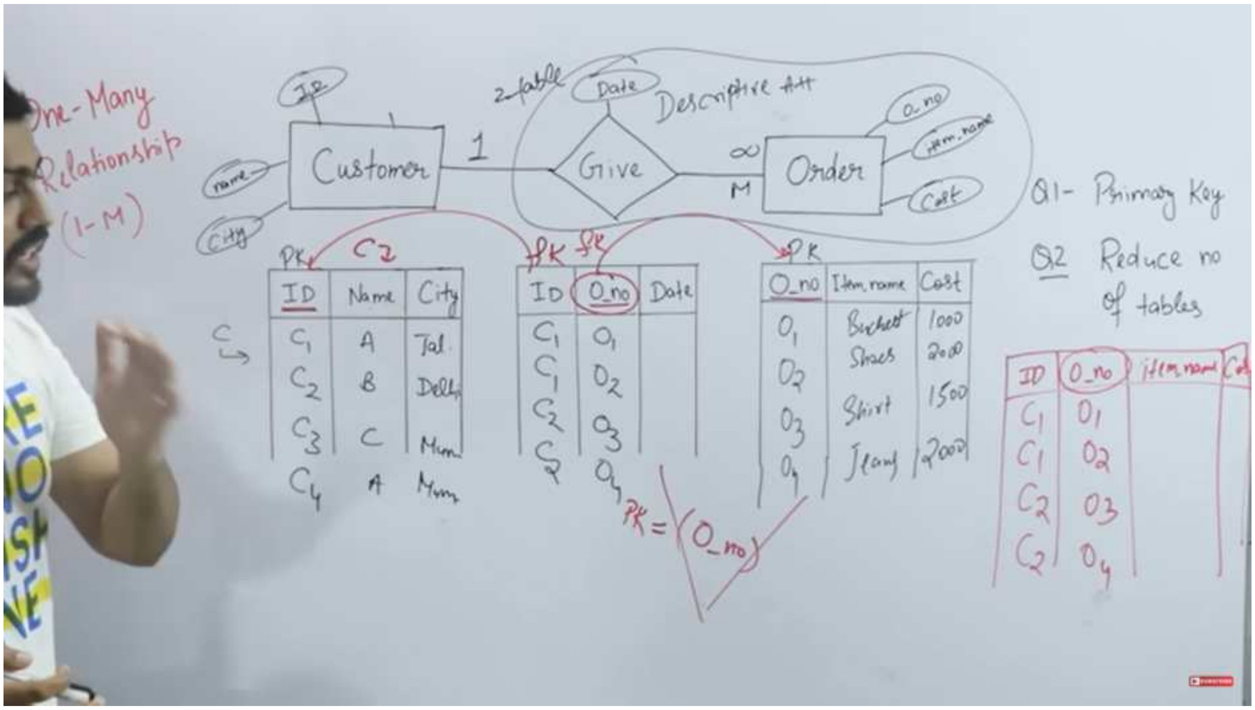
1. 1-1
2. 1-M
3. M-1
4. M-M (M-N)

One to one relationship



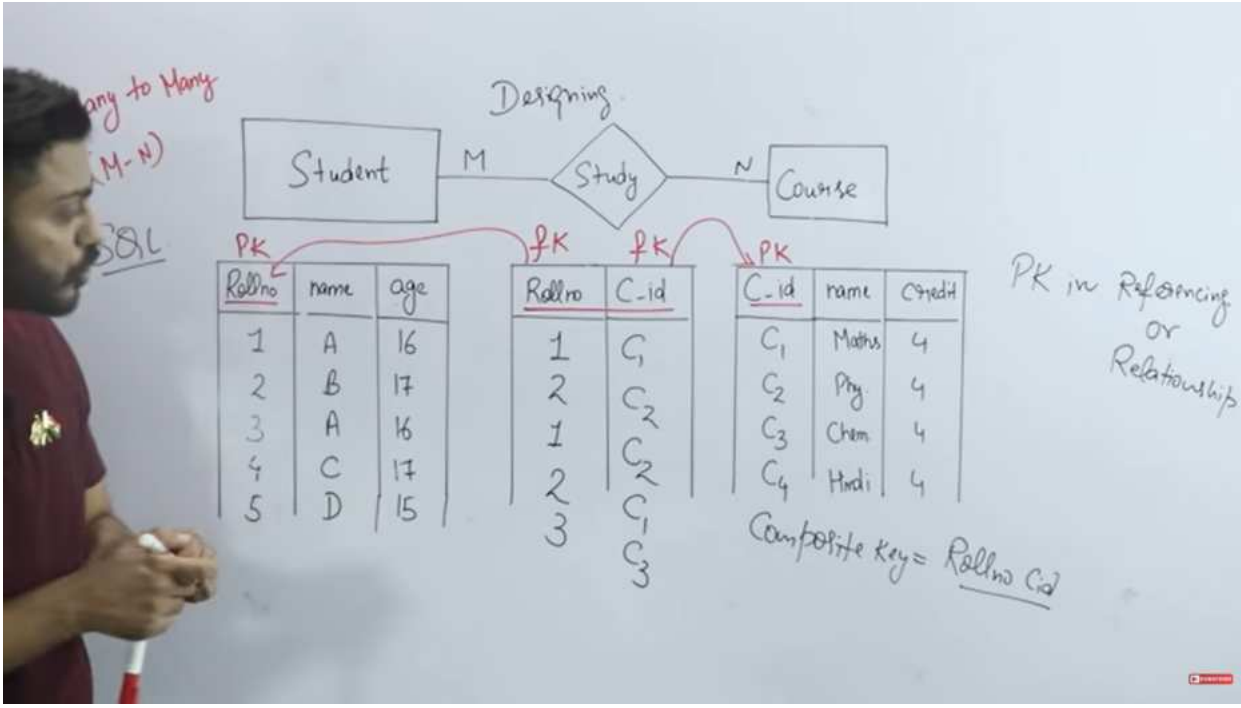
Finally there will be 2 tables

One to many relationship



Finally there will 2 table
And many ke side merge karna hai

Many to Many relationship



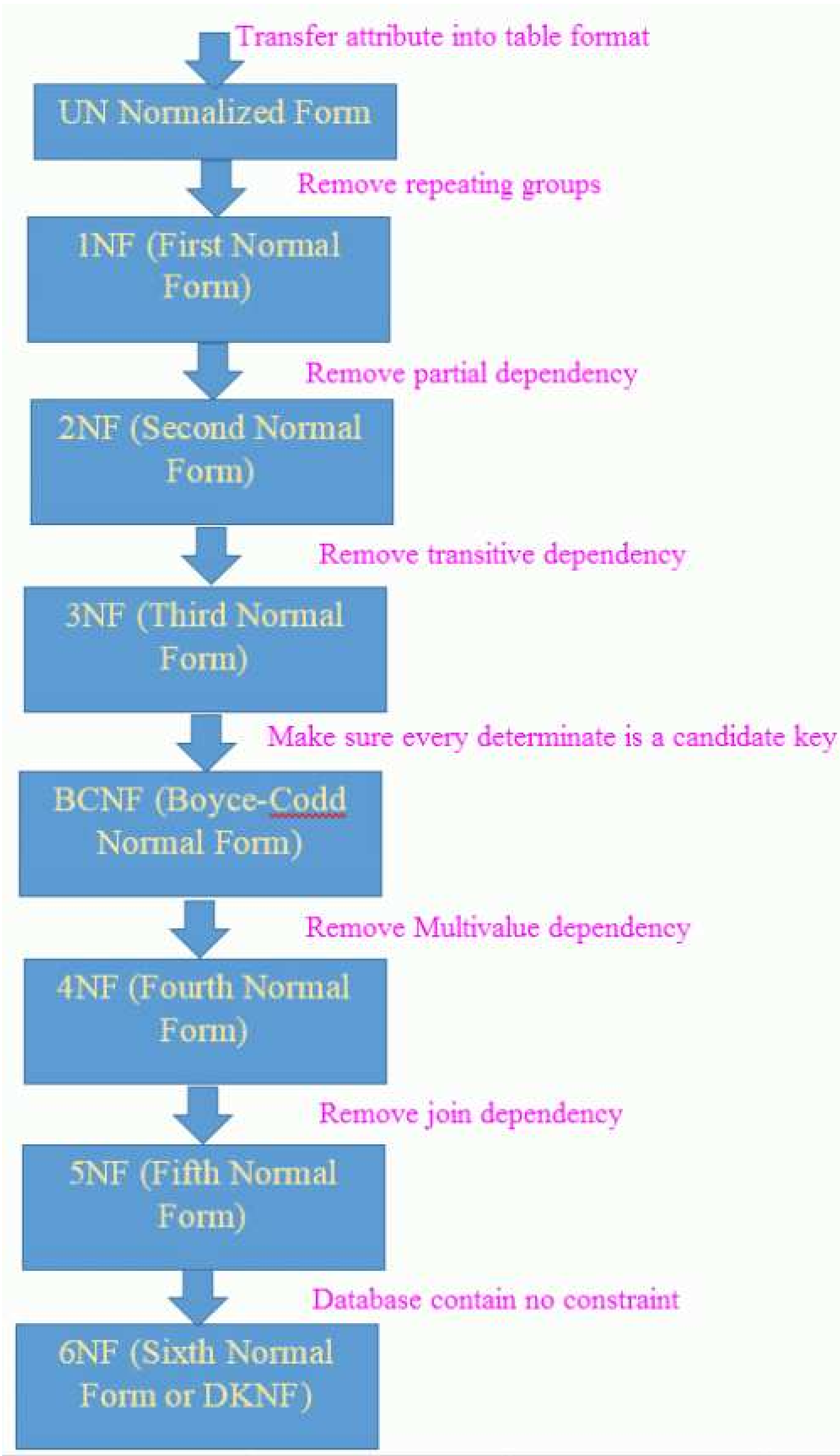
Here primary key will combine (roll no + c_id)
No reduction in Table

Normalization

Normalization is the process of making a table free from insertion , update and delete anomaly and save space by reducing redundant data or duplicate data .
OLTP= normalized data required
OLAP= denormalized data is required

What is an Anomaly?

- Anything we try to do with a database that leads to unexpected (unpredictable) results.
- Three types of Anomaly:
  - insert
  - delete
  - update
- Need to check your logical database design carefully:
  - the only good database is an anomaly free database.


Transfer attribute into table format
UN Normalized Form
Remove repeating groups
1NF (First Normal Form)
Remove partial dependency
2NF (Second Normal Form)
Remove transitive dependency
3NF (Third Normal Form)
Make sure every determinate is a candidate key
BCNF (Boyce-Codd Normal Form)
Remove Multivalue dependency
4NF (Fourth Normal Form)
Remove join dependency
5NF (Fifth Normal Form)
Database contain no constraint
6NF (Sixth Normal Form or DKNF)

1st Normal form

1st Normal Form Example

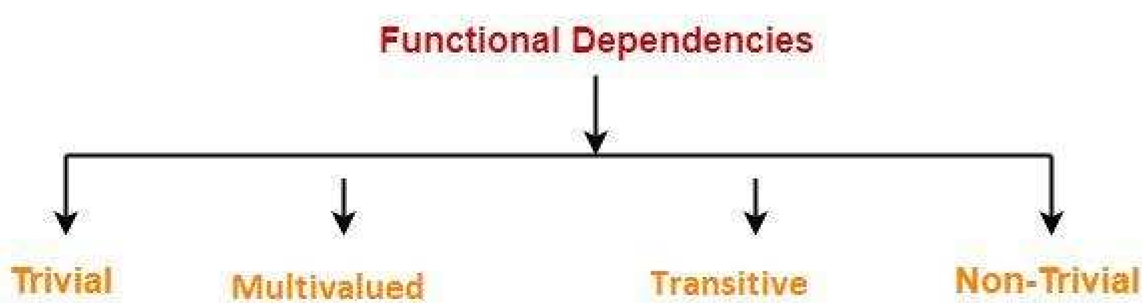
How do we bring an unnormalized table into first normal form?
Consider the following example:

TABLE_PRODUCT

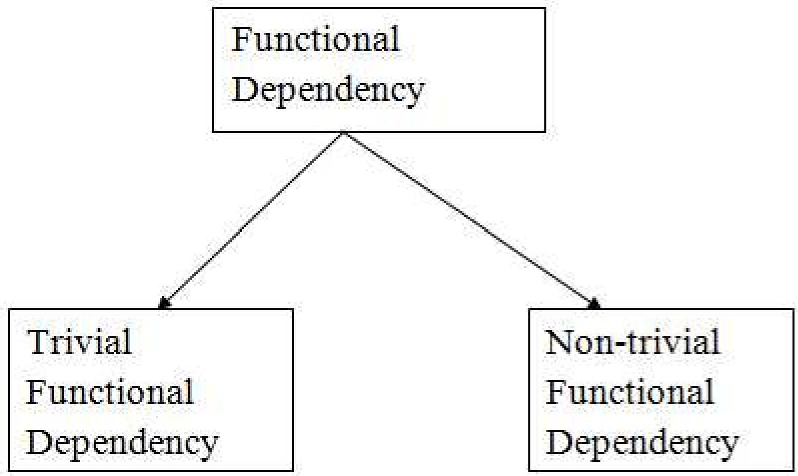
Product ID	Color	Price
1	red, green	15.99
2	yellow	23.99
3	green	17.50
4	yellow, blue	9.99
5	red	29.99

This table is not in first normal form because the [Color] column can contain multiple values. For example, the first row includes values "red" and "green."

Closure



Functional Dependency



Super key : set of attributes whose closure contains all attributes of the given relation
Candidate key : Minimal Super key whose proper subset does not contains super key

Phi does not contain super key

X+ = contains set of attributes determined by x

R(A,B,C,D,E)
Functional dependencies -{A→B,D→E}

ABCDE+ = {A,B,C,D,E}
Here B can be determined by A and similarly E can be determined by D

This can be reduced to
ACD+ = {A,B,C,D,E}

Now {ABCDE , ACD} are super key because they contains all the attributes

Minimal of them ACD
Their proper subset will {A,C,D,AC,AD}
And none of them is super key so ACD is candidate key

Prime attributes ={A,C,D}= ACD
Now check whether from (A,C,D) any of them present in the right side of the function dependencies or not , if no then that will be candidate key otherwise we need to more check neither of (A,C,D) present on right side of the function dependencies

Another question
R(A,B,C,D)
Functional dependencies ={ A→B,B→C,C→A}

ABCD+ = {A,B,C,D}
Here we can get B from A and C from B and A from C
Its reduced to
AD+ ={A,B,C,D}

So { ABCD , AD} super key and minimal of them is AD
{A+}= { A B C}
{D+}= {D}

Here from both A and D none of them is super key so here I can say that AD is candidate key .
The proper subset of AD will be
AD ={A,D} = prime attributes

Now check prime attributes in right side of the functional dependencies
In functional dependencies C→A is available then
CD= {C,D} = prime attributes
BD= {B,D} = prime attributes
So the prime attributes will be { A, B, C ,D}
No prime attributes will be 0 here

Then here (AD, CD,BD} 3 candidate key available

Another question
Relation given R(A,B,C,D,E,F)
FD= {AB→C, C→DE,E→F,D→A,C→B}
Here C→DE can be replaced with C→D and C→E
Find no of candidate key for this question
ABCDEF+ = {A,B,C,D,E,F}
ABDEF= {A,B,C,D,E,F}
ABF= {A,B,C,D,E,F}
AB= {A,B,C,D,E,F}
Now { ABCDEF , ABDEF,ABF,AB} are super key and
AB is candidate key because AB is minimal
Now check for proper subset of AB
{A,B}
{A}+ = {A}*
{B}+ = {B}*
Prime attributes = {A,B}

Now check for right side in functional dependencies for {A,B}

AB

/ \

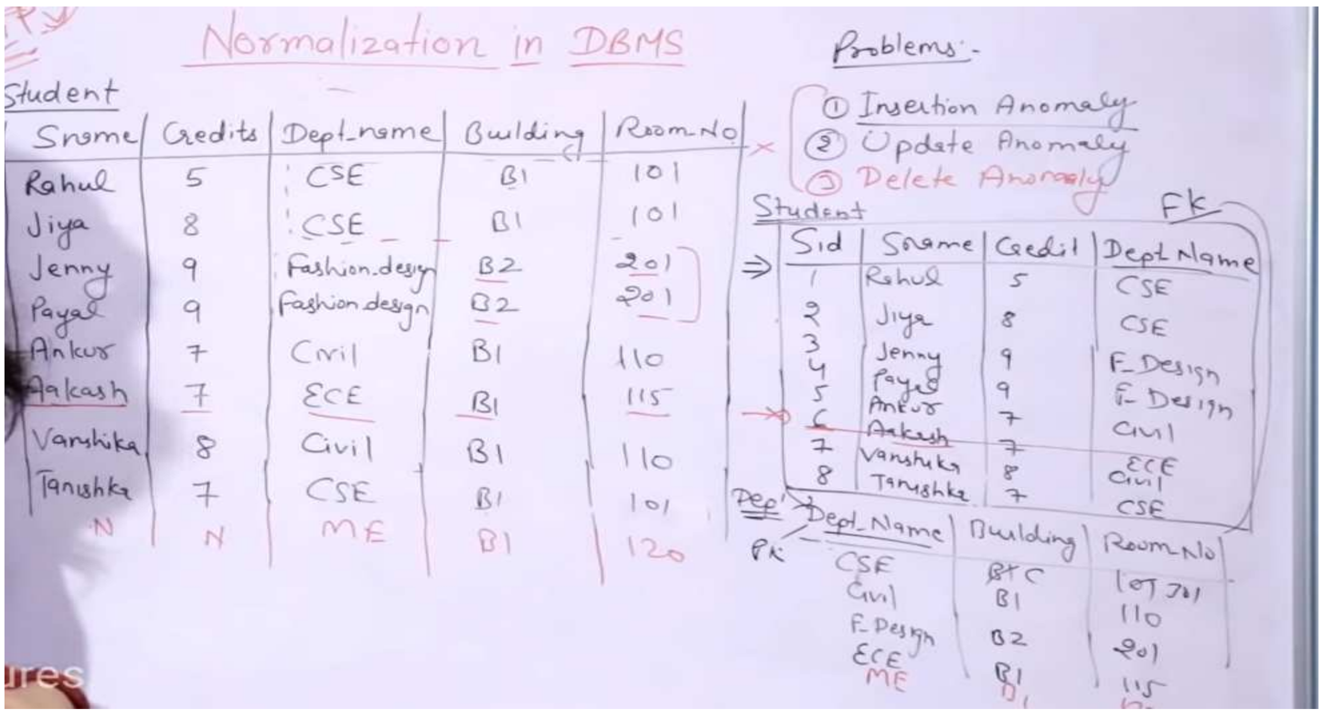
DB AC→sk here Closures of C is all attributes then AC cant be candidate key

{D}+ = {DA}*

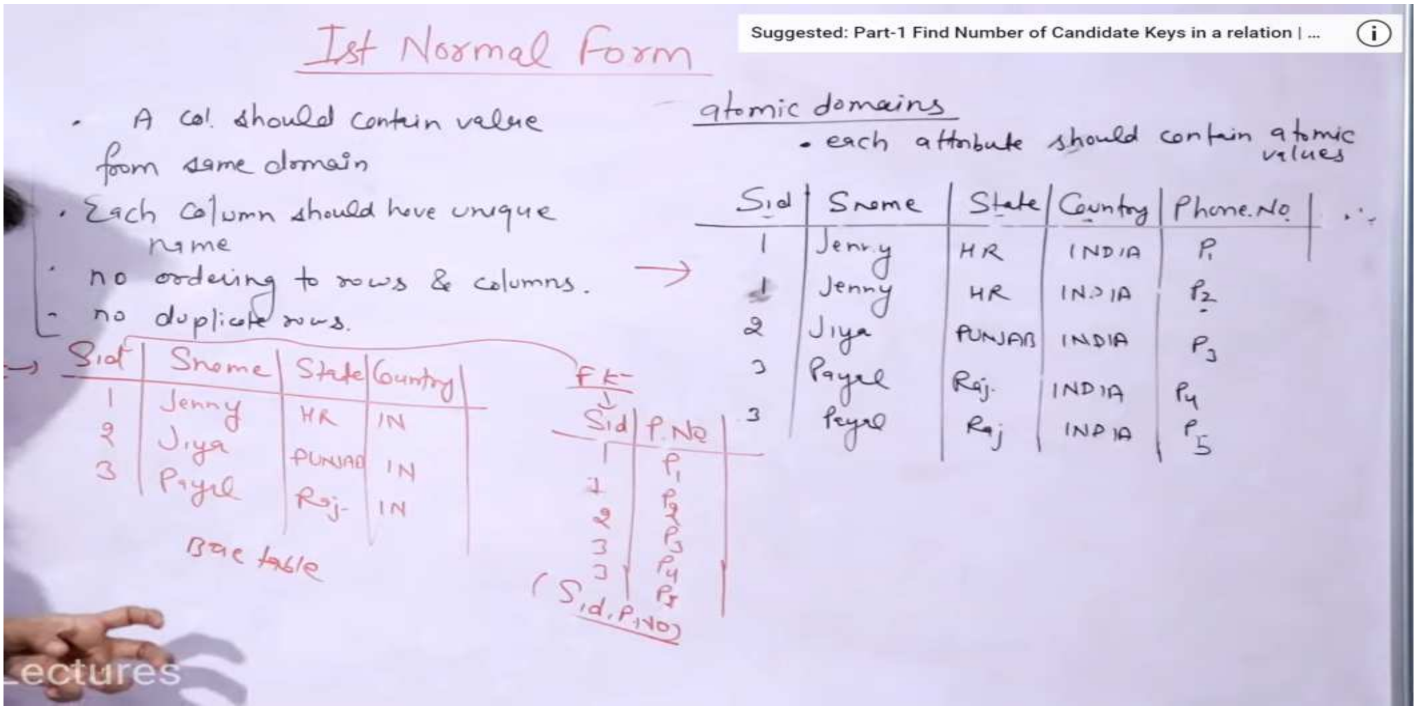
{C}+ = {CDEFAB} here is C is candidate key because their proper subset will be phi and phi is not super key

Then finally prime attribute will be {A,B,D,C}

And there will only 3 candidate key {AB,BD,C}



1st normal form



Second Normal form

1. It is in 1st NF
2. No partial Dependency in the relation
   PROPER SUBSET OF CK⇒ NON PRIME ATTRIBUTE , ye nahi hona chahiye
   Example
   Relation R(A,B,C,D,E,F)
   F.D = {A→B,B→C ,C→D,D→E}
   ABCDEF= {A,B,C,D,E,F}
   AF+ = {A,B,C,D,E,F}

   Here AF is candidate key and Prime attribute = {A,F}
   Non prime attribute ={B,C,D,E}
   Now check for partial dependency
   A→B is available in the functional dependencies so its not in second normal form

Third Normal form

1. It is in 2 NF
2. No transitive dependency for non-prime attributes
   A→B and B→C ⇒ A→C , here A and C is non prime attributes

Non Prime attribute→ Non Prime attribute = for transitive dependency (that will not in 3 NF)

A table is in 3 NF , if and only if for each of its non-trivial functional dependency at least one of the following conditions holds

1. L.H.S is Super key
2. R.H.S is Prime attribute

R(A,B,C,D)
Functional dependencies -(A→B,B→C,C→D)

A+ = {A B C D }
Here is super key and candidate key
Prime attribute = {A}
Non prime attribute = {B,C,D}

In FD there is relation NPA⇒NPA that is B→C , C→D
Then this is not in 3 NF

Another Example
F(A,B,C,D,E,F)
F.D = {AB→CDEF, BD→F}
ABCDEF + = {A B C D E F }
AB+ = {A B C D E F }
Super key AB and prime attribute A and B
So then Non prime attribute C D E F
Now check in F.D
BD→F
This will be considered as NPA == NPA (Not in 3rd Normal form)

BCNF(Boyce-codd Normal form)

R(A,B,C)
FD- {A→B,B→C,C→A}

A relation is in BCNF iff

1. It is in 3 NF
2. For each non-trivial FD x→y , x must be super key

R(A,B,C)
FD- {A→B,B→C,C→A}
ABC+ = {A B C }
A= { A B C}
Here A is candidate key and there is no proper subset of A
Ck = A
And now check for A in the right side
C is also candidate key and B is also candidate key
So A B C are candidate key

So above relation is in BCNF

Another question
Find the highest Normal form in the given relation
R(A,B,C,D,E)
F.D:- {A→BCDE,BC→ACE,D→E}
ABCDE = {A B C D E }
A+ = {A B C D E}
Here A is candidate key because there is no any proper subset of A
And now check for right side
BC = { B , C}
B+= {B}
C+= {C}
So here BC is candidate key because their proper subset does not contain super key
Till now candidate key = A, BC
Prime attribute = A, B, C
From BC
/ \
AC BA these are super key not sure about the candidate key but these both have A is proper subset and A is super key then both can not be candidate key

Now check

	A→BCDE	BC→ACE	D→E
BCNF	correct	correct	incorrect (LHS should be super key)
3NF	correct	correct	incorrect (if BCNF then below than that will also correct , both condition not satisfying for 3 NF)
2NF	correct	correct	correct (No partial dependencies)

Another Example
R(A,B,C,D,E)
F.D : { AB→CDE,D→A}
Here candidate key will be AB and DB and prime attribute will be A B D

	AB→CDE	D→A
BCNF	correct	incorrect (LHS is not super key)
3NF	correct	correct (satisfying second conditions)

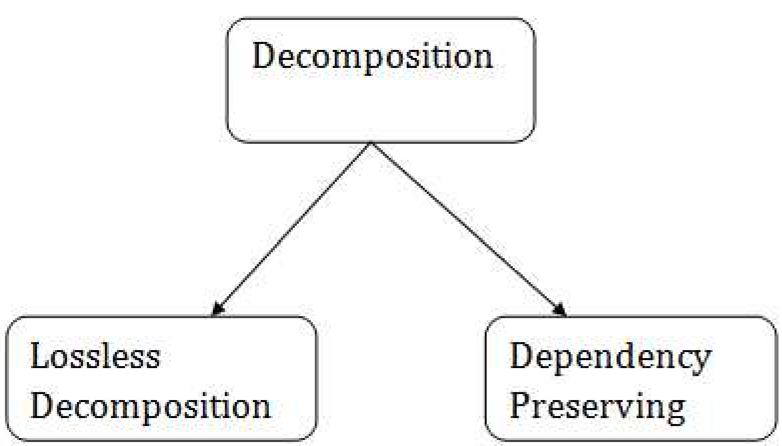
Then the highest Normal form of this relation is 3NF

- [ ] Third Normal from always ensures ‘ Dependency Preserving Decomposition’ but not in BCNF
- [ ] Both third and BCNF ensures lossless decomposition

Jan 14, 2022

Decomposition

- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables.
- If the relation has no proper decomposition, then it may lead to problems like loss of information.
- Decomposition is used to eliminate some of the problems of bad design like anomalies, inconsistencies, and redundancy.



Dependency Preserving Decomposition

R(A,B,C) ⇒ R

1	1	1
2	1	2
3	2	1
4	2	2

Functional dependency can derived like
A→B , A→C then this can be A→BC
And Combinely B and C derive A this can be written as BC→A
F=FD(R) = {A→BC , BC→A}
Let suppose R further divided into n parts
R = R1 , R2 , R3 ……………Rn
FD F = F1, F2 , F3 …………….Fn
F1UF2UF3…………UFn == F then that will be dependency Preserving Decomposition

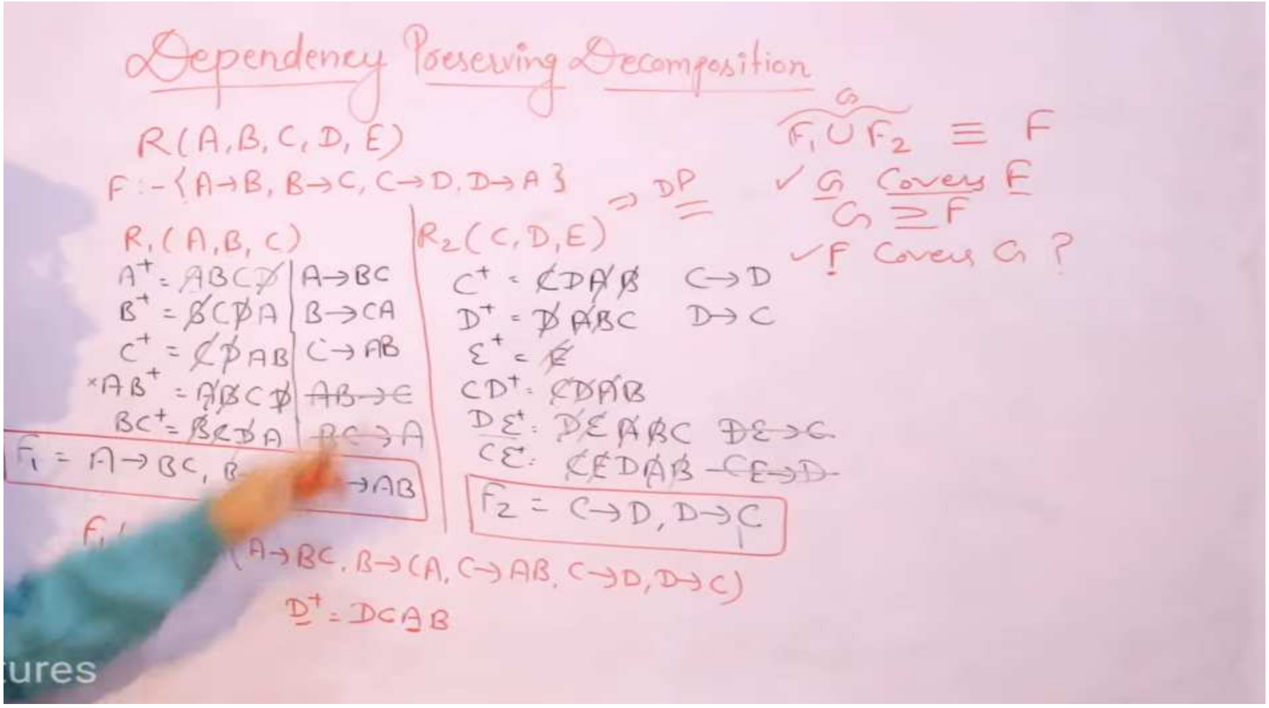
Now let's break R into 2 parts
R1(A,B)

1	1
2	1
3	2
4	2

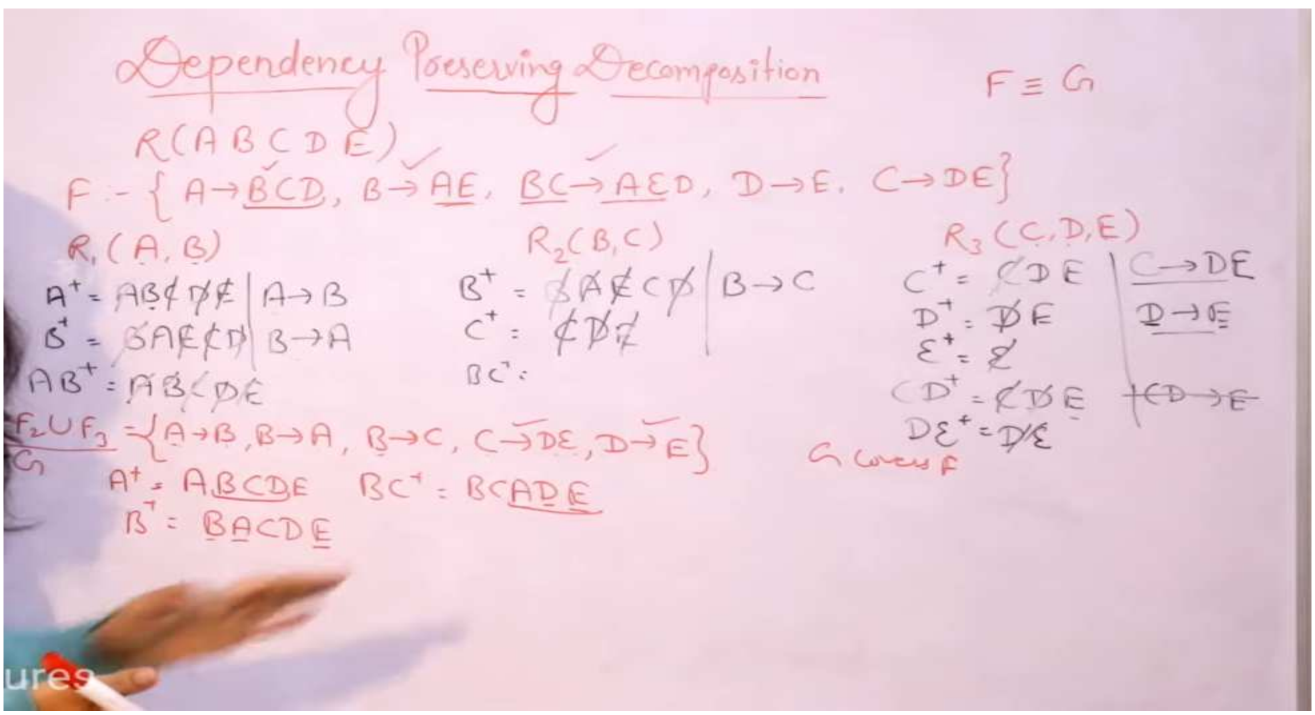
R2(B,C)

1	1
1	2
2	1
2	2

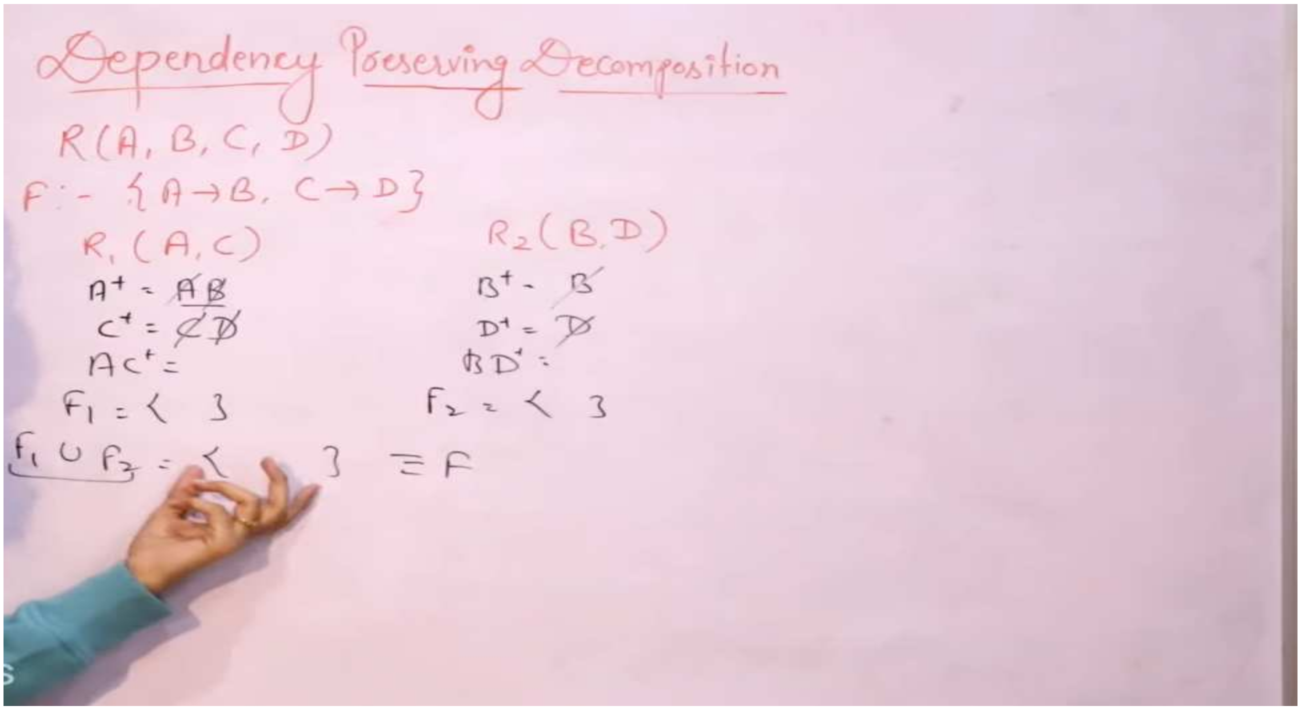
F1=FD(R1) = { A→B}
F2=FD(R2) = {}
So here F!= F1UF2 This is not dependency preserving Decomposition



Here A→A or BA→BA will be trivial functional dependencies , so we can discard this ,only non -trivial functional dependencies will be valid



This is dependency preserving decomposition



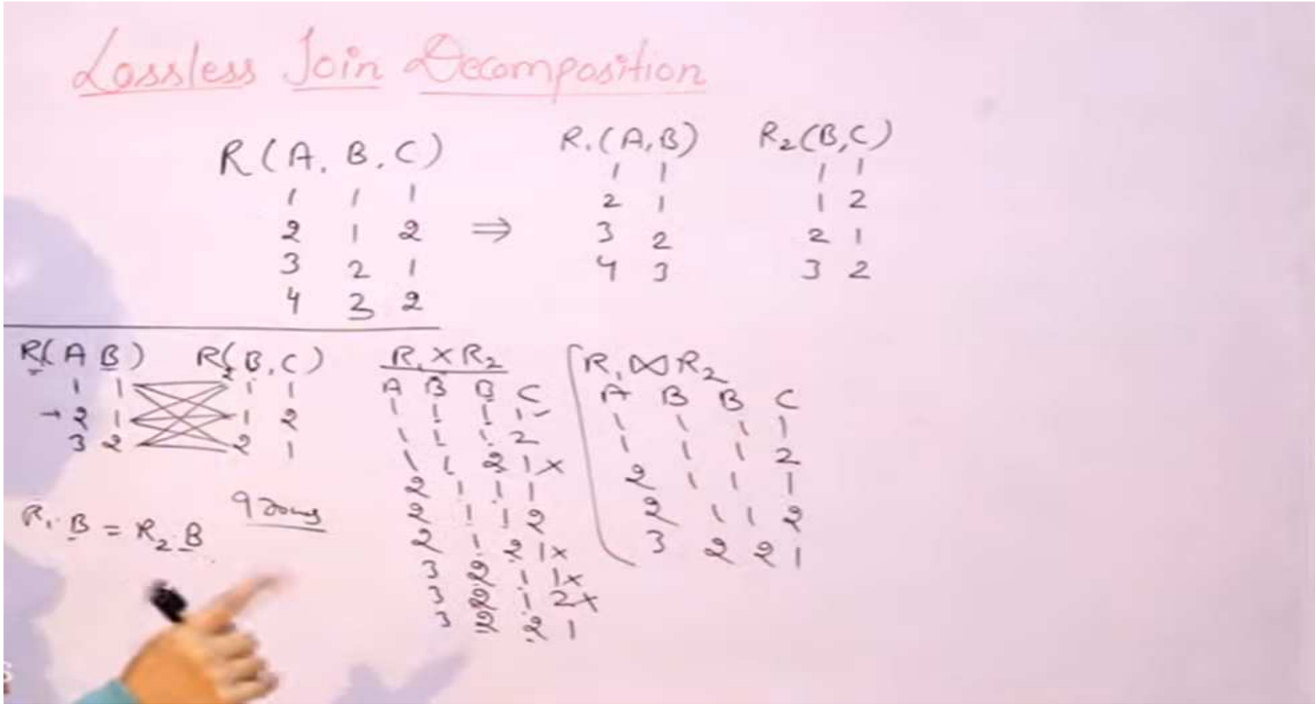
This is not dependency preserving decomposition

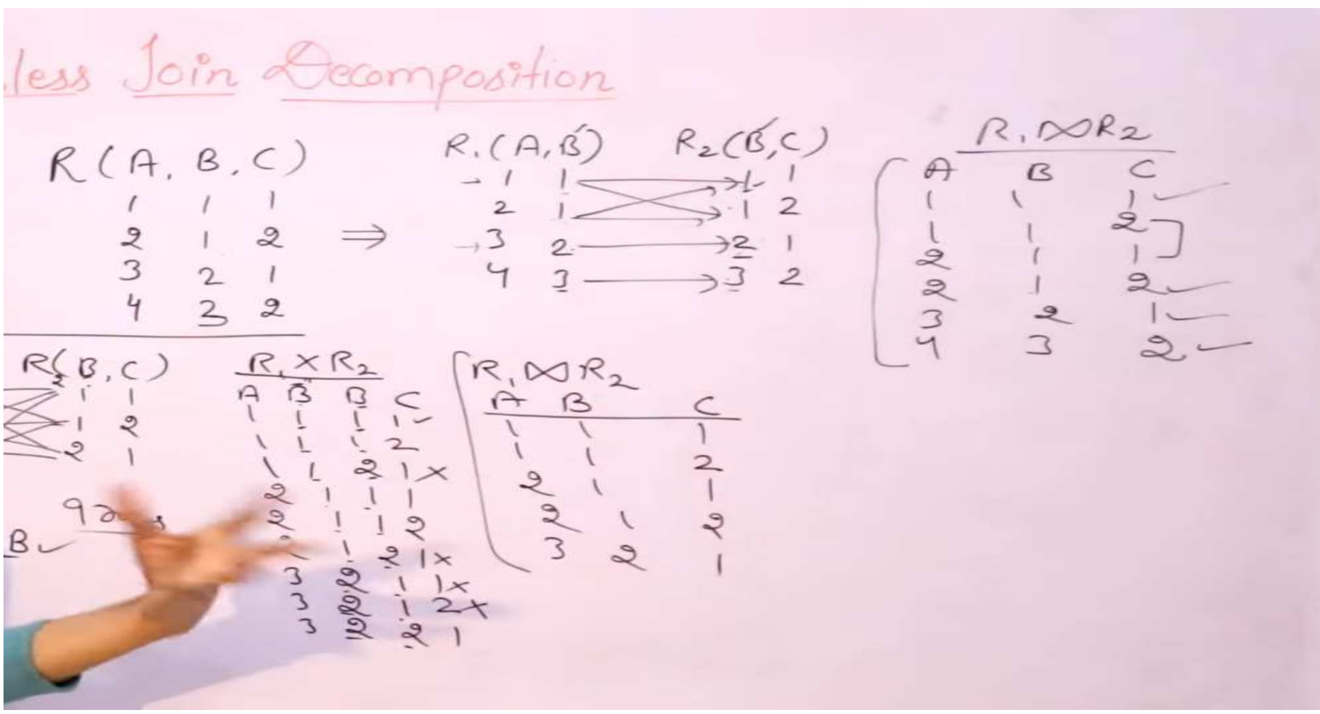
Some time BCNF me dependency preserving decomposition possible nahi ho pata hai

Lossless Join Decomposition

It is a mandatory property for decomposition .
If R divides into R1 and R2 then Natural join of both R1 and R2 should be equal to R
R1 Natural join R2 ===R1
Then that will lossless join Decomposition

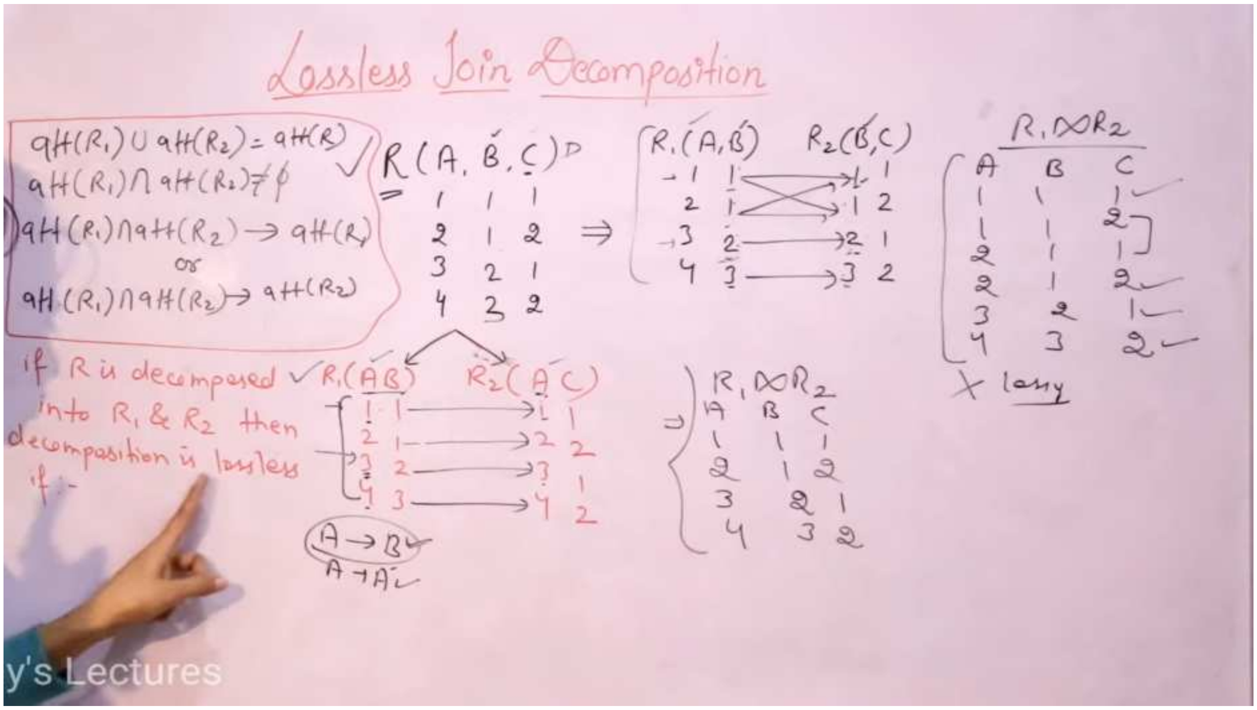
Here Natural join is advanced version of cross product of R1 and R2
Only extra condition R1.B==R2.B then that will consider as natural join





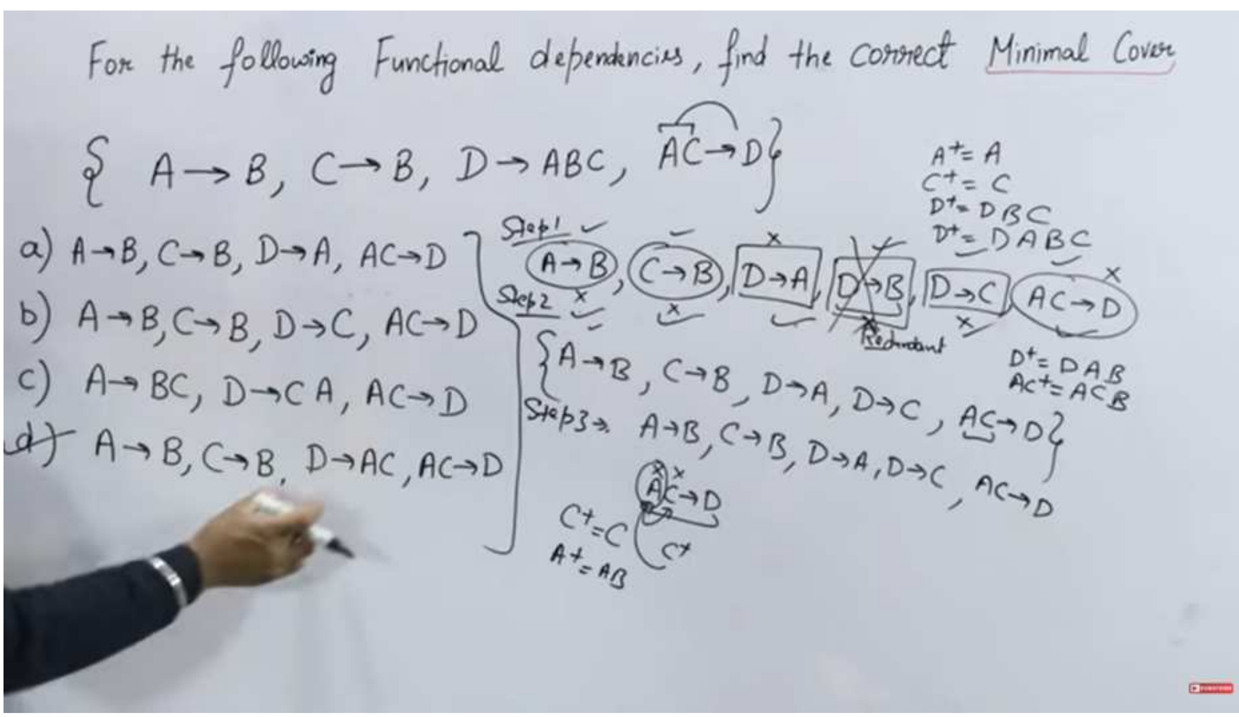
This is not lossless join decomposition because after combining these two relations some extra is coming , but it should be totally equal .

ONE ANOTHER WAY TO CHECK THIS , DONO ME SE JO COMMON ATTRIBUTE HAI , AGAR OO KISI V RELATION ME CANDIDATE KEY BAN JAATA HAI THEN THAT WOULD BE LOSSLESS JOIN DECOMPOSITION .



For lossless join decomposition there should be only 3 conditions required

Minimal cover



Total 3 steps required

1. Write in simplified version
2. Now let assume that ki remove some particular relation and take closures , if all attribute are derived from that then that will be redundant
3. Now try to break left hand side , if left hand side contain more that one attribute
   Way to simplified left hand side is
   Take clouser of removing that attribute , if in clouser of another key contain removed key then that is removal
   If not then that key not removal

Equivalence of functional dependency

X= {A→B, B→C}
y= {A→B,B→C,A→C}

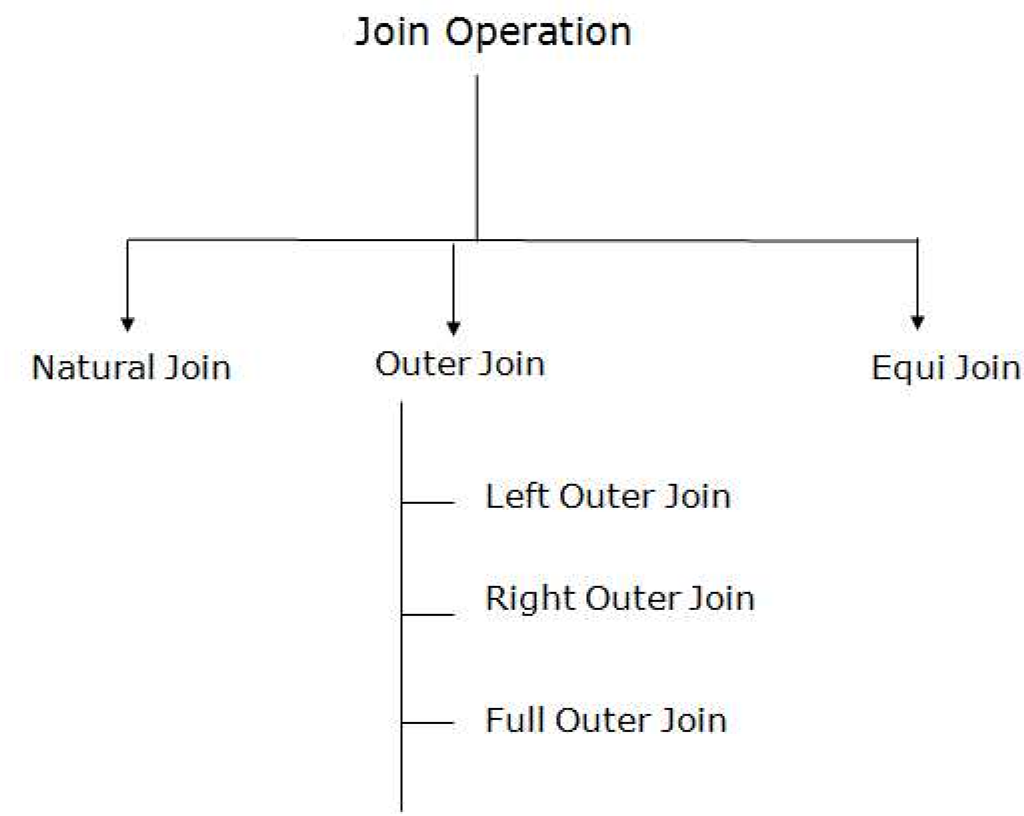
For this we have to check
If X covers Y
And
Y covers X

Now check X covers Y

Let check for Y attributes
A+ = {ABC} (select A from Y and take closures from X)
From here we can say that A→B and A→C , determine
B+ = {BC}
B→C
Then here X covers Y
Now check for Y covers X , or not
A+ ={ABC}
A→B
A→C

B+ = {BC}
B→C
So here Y covers X
Then this is equivalence functional dependency

JOINS



JOINS

1. Cross join
2. Natural join
3. Conditional join
4. Equi join
5. Self join
6. Outer join (there 3 types)

When we need help of 2 or more tables then we need joins to fetch the data and sub query is also a another options
And there must be some common attribute in both of the table then join will be possible

JOINS = CROSS PRODUCT + SELECT STATEMENT(CONDITIONS)

Natural joins

JAB 2 COMMON ATTRIBUTE KEY VALUE KO EQUAL KARNA HAI TO US CASE MEIN HUM NATURAL JOIN KARTE HAIN

Employe table

E_NO	E_NAM	ADD
1	RAM	DELHI
2	VARUN	CHD
3	RAVI	CHD
4	AMRIT	DELHI

Department Table

DEPTN	NAME	E_NO
D1	HR	1
D2	IT	2
D3	MRKT	4

QUESTION

Find the emp name who is working in a department

In cross product , there will be 12 rows (4*3)

Query will be like this

Select E_NAM from emp , dept (this cross product of emp,dept) where emp.E_no =dept.E_No;

emp.E_no =dept.E_No , ye conditions har row ke run hoga , and matching row will be in the output

This can be simply written as

Select E_NAM from emp natural join Dept

Natural join = to common attribute key value ko equal karne ke liye use karte hain

And dono attribute ke name v same ho that means column should be the same for both

If one table contain E_NO and second table should also contain E_NO then natural join will work

SELF JOIN

This is study table

s_id	c_id	since
s1	c1	2016
s2	c2	2017
s3	c3	2017

Find student id who is enrolled in at least two courses
Ans
For join we need two tables so here we will take same table for making joins
For joins here cross product will contains 3*3 = 9 rows that is self join actually
Select T1.s_id from study as T1, study as T2 where T1.s_id=T2.s_id and T1.c_id<>T2.c_id
Here T1 and T2 is two sample table of study table we can not use study directly otherwise it will error in IDE

EQUI JOINS

Question
FInd the emp name who worked in a department having a location same as their address?
emp

E_no	E_name	Address
1	Ram	Delhi
2	varun	CHd
3	Ravi	chd
4	Amrit	delhi

department

Dept no	Location	Eno
D1	Delhi	1
D2	Pune	2
D3	Patn	4

Select Ename from emp, dept where emp.e_no=dept.e_no and emp.address=dept.location

LEFT OUTER JOIN

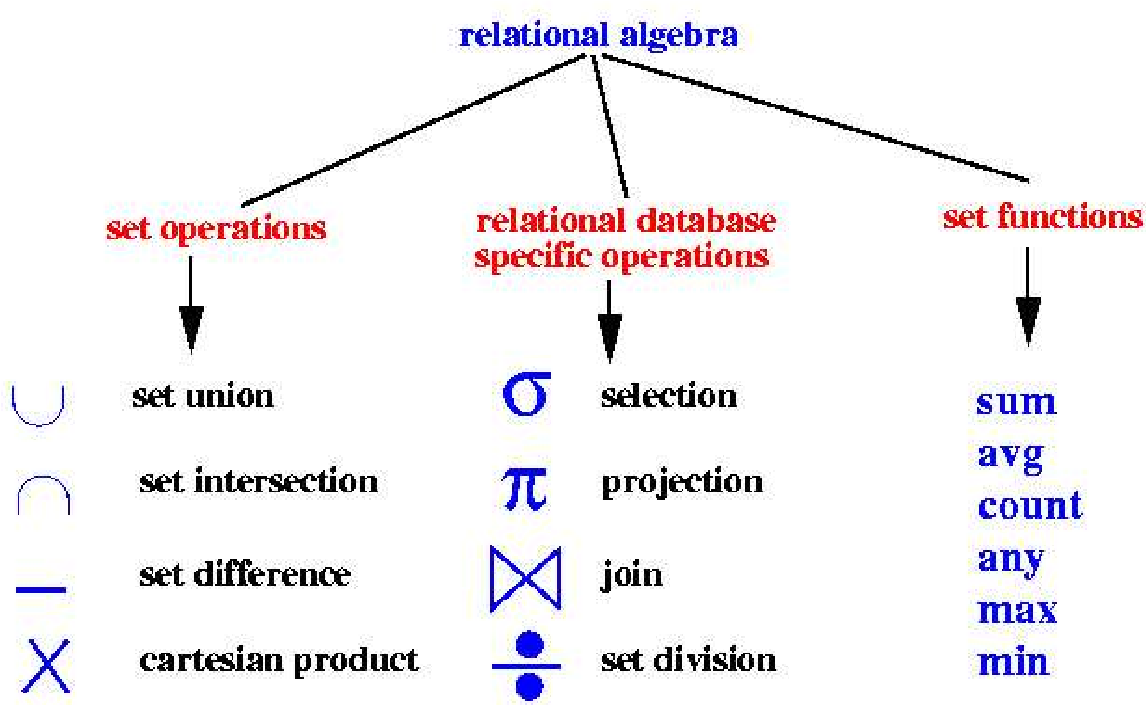
It gives the matching rows and the rows which are in left table but not in right table
Let consider there is two table
EMP and DEPT
And i have to take left join on EMP with DEPT
Query will be
Select emp, e_name, d_name, loc from emp (left me hi hona chahiye) left join dept on(emp.deptno=dept.deptno)
Jo value right me nahi hai oha pe NULL aa jayega

Right outer join

Similarly like left outer join right outer join will work

Those missing value on left side that will NULL
Query
Select emp, ….. From emp right outer join dept(right side hi hona chahiye) on (emp.deptno= dept.deptno)

Relational algebra



Relational Algebra

Six basic operators:

- Select: σ - Selects tuples that satisfy a given predicate
- Project: Π - Projects specified fields.
- Union: $\cup$
- Intersection: $\cap$
- Set difference: $-$
- Cartesian product: $\times$
- Rename: ρ
- *Division:* $\div$
- *Joints*

- The operators take one or two relations as inputs and produce a new relation as a result.

Derived Operator

1. Join
2. Interset
3. Division

Projections

Example - Projection

- Produce a list of salaries for all staff, showing only staffNo, fName, lName, and salary details.

$$\Pi_{\text{staffNo, fName, lName, salary}}(\text{Staff})$$

staffNo	fName	lName	salary
SL21	John	White	30000
SG37	Ann	Beech	12000
SG14	David	Ford	18000
SA9	Mary	Howe	9000
SG5	Susan	Brand	24000
SL41	Julie	Lee	9000

Selection operator

Projection operator bilkul last me aata hai
Lets a example , Retrieve a name whose staffNO SG3

PHAI (name) (Sigma(staffNO='SG3')(staff))

Set difference

A-B ⇒ A but not in B

1. No of column must be same
2. Domain of the data must be same

Division operator

Enrolled

sid	cid
s1	c1
s2	c1
s1	c2
s3	c2

Course table

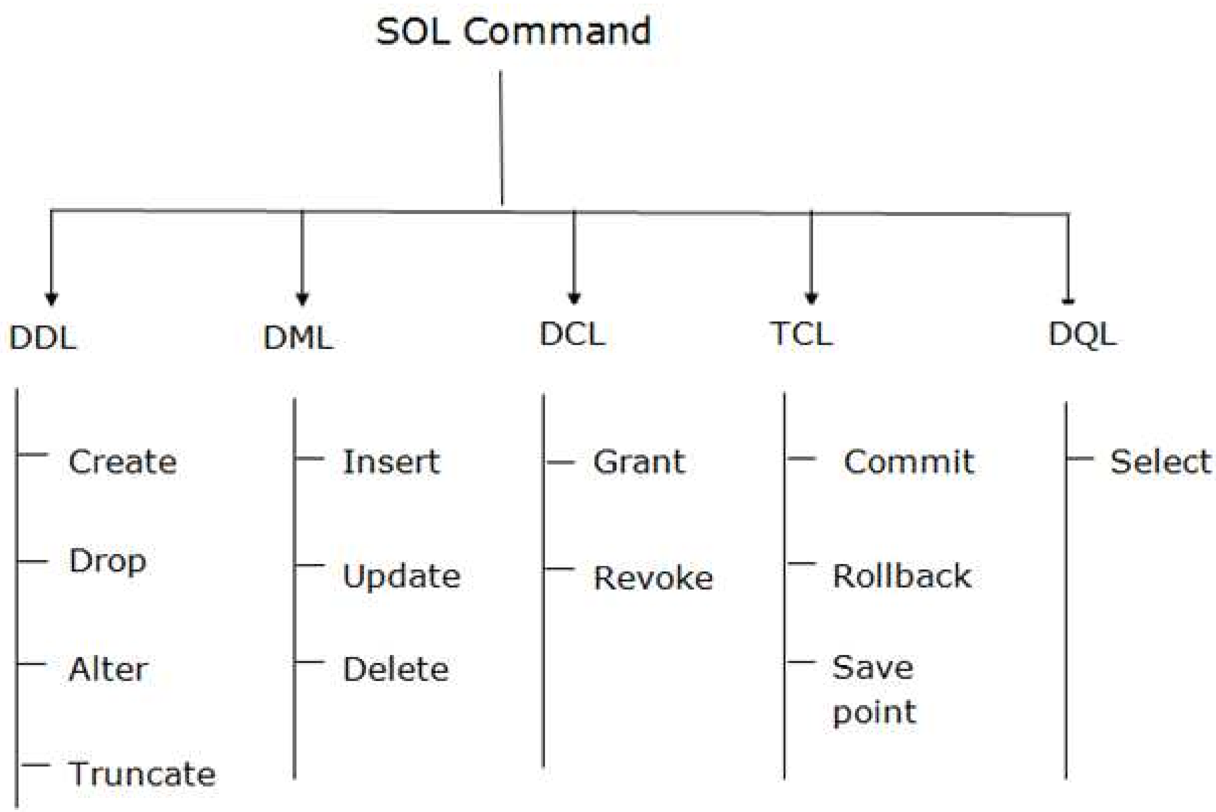
cid
c1
c2

Query : Retrieve sid of students who enrolled in every course
When we get all , every , these two is indicating for division
A(X,Y)/B(Y) = if the result x values for that there should be tuples <x,y> for every y value of relation B.
Enrolled(sid,cid)/ course(cid) , jisse output aayega oo numerator me hona chahiye , jo help kar raha oo denomurator
And actually there should be S1 , because in S1 , both the courses available c1 and c2.
phi(sid)(enrolled)- phi(sid) ((phi(sid) (phi(sid)(enrolled)*phi(cid)(course)))-(enrolled))

S1 c1
S2 c2
S3

Their cross product will be
S1 c1 s1 c1
S1 c2 s2c1
S2 c1 (cross) – s1c2(enrolled) =======> s2 c2 (jo dono course me enrolled nahi hai) matlab disqualified
S2 c2 s3c2 s3 c1
S3 c1
S3 c2
Total-disqualified = qualified (real output)

Structure query Language(sql)



Constraints command

1. Primary key
2. Foreign key
3. Check
4. Unique
5. Default
6. Not null

M= manipulation
C= Control

Desc table name = it will show the attribute of the schema

Create table

```sql
CREATE TABLE users (
  id INTEGER PRIMARY KEY NOT NULL AUTO_INCREMENT,
  email VARCHAR(255) NOT NULL,
  `password` VARCHAR(255) NOT NULL,
  phone_number VARCHAR(15),
  created TIMESTAMP NOT NULL DEFAULT NOW()
);

CREATE TABLE orders (
  id INTEGER PRIMARY KEY NOT NULL AUTO_INCREMENT,
  user_id INTEGER NOT NULL,
  created TIMESTAMP NOT NULL DEFAULT NOW(),
```

Alter command

```sql
ALTER TABLE table_name ADD column_name datatype;
ALTER TABLE table_name DROP column column_name;
ALTER TABLE table_name MODIFY column column_name datatype;
ALTER TABLE table_name rename column old_column to new_column;
ALTER TABLE table_name rename To new_table_name;
Alter table Student add weight integer; Alter table Student add Course
varchar(60);
```

1. Add columns

2. Rename columns
3. Modify datatype
4. Modify datatype length
5. Add constraints
6. Remove Constraints
7. Rename columns/table

ALTER	UPDATE
1. DDL(only structure me change) 2.	1. DML (data me koi v change) 2. We use here update and set

Delete	DROP	Truncate
1. DML 2. Delete from student (rows) 3. Ek ek karke delete hota hai 4. We can delete here specifically where id=1 5. Rollback possible	1. DDL 2. Drop table student (pura structure delete)	1. DDL 2. Delete the rows 3. EK baar me saare row ko delete karta hai 4. No conditions required 5. No rollback possible

Constraints in sql

7. Primary key
8. Foreign key
9. Check
10. Unique
11. Default
12. Not null

Queries and subqueries

Table name = EMP

Eid	Ename	Dept	Salary
1	Ram	HR	10000
2	Amrit	MRKT	20000
3	Ravi	HR	30000
4	Nitin	MRKT	40000
5	Varun	IT	50000

Queries

1. Write a sql query to display maximum salary from emp table
   Select max(salary) from emp ;
2. Write a sql query to display employee name having salary is maximum
   Select ename from emp where salary = (select Max(salary) from emp); outer query = inner query , here phale inner query execute hoge
3. Write a sql query to display the second highest salary from the emp table ?
   Select salary from emp where salary <>(select max(salary) from emp)
4. Write a query to display all the dept names along with no emps working in that ?
   Here we use group by clause
   Select dept from emp group by dept;
   Or we can use aggregate function also
   Select dept , count(*) from emp group by dept ;
5. Write a query to display all the dept names where no of emps are less than 2
   Query :
   Count se sath where clause use nahi ho sacta , because where clause pure table me lagta hai , count sirf ek colum ka data fetch karta hai

   Select dept from emp group by dept having count(*)<2; IT
   For the name we need to check nested query
   Select ename from emp where dept In (Select dept from emp group by dept having count(*)<2;) IT

6. Write a query to display highest salary department wise and name of emp who is taking that salary ;
   Department wise highest salary
   Select dept, max(salary) from emp group by dept;
   HR- 30000
   MRKT- 40000
   IT- 50000

So that final will be select ename from emp where salary In (select Max(salary) from emp group by dept);
Here every row will be compare using (30000,40000,50000)

IN and NOT IN

Question : find the name of emps who are working on a project ?
Select ename from emp where eid IN (select dist(eid) from project) ;

Exists/ Not exists

It always return true or false
Correlated Nested query ke ander use karte hain
Find the details of emp who is working on at least one project ?
Emp

Eid	Ename	Address

Project

Eid	Pid	Pname	Location

Select * from emp where Eid exists (select Eid from project where emp.eid=project.eid)
COrrelated Nested query : yaha pe outer se ke row lekar match karwate hain inner query me , agar match kiya toh print otherwise check for next row .
But for the in or not execution will be totally different because there inner query run first and outer query run later

Aggregate function

1. Max
2. Min
3. Count
4. Sum
5. Avg

   Select max(salay) from emp;

Correlated Subquery(synchronized query)

It is a subquery that uses value from outer query
Top to down approach
Find all the employee detail who work in a department
Table name : EMP

Eid	name	Address

Table Name: DEPT

Did	Dname	Eid

Select * from emp where exits(select * from dept where dept.eid= emp.eid);
Its top to down approach

DIFFERENCE BETWEEN query

1. Nested query (Bottom up)
Select * from emp where e_id in (select e_id from dept)

2. Correlated subquery (top down approach)
   Select * from emp where exits (select id from dept where emp.e_id=dept.e_id);

3. Joins (cross product + condition)
   Select * from emp ,dept where emp.e_id=dept.e_id

Find nth highest salary using sql

Select id , salary from emp e1 where N-1= (select count(distinct salary) from emp e2 where e2.salary>e1.salary)

Transaction

1. It is a set of operations used to perform a logical unit of work
2. A transaction generally represent change in database

Two operation in Transaction

1. Read = Database ko access
2. Write = some change in database

Example transferring the money
Let A=1000 , B=2000
Before commit , everything running in RAM (local memory)
R(A) reading the value of A
A=A-500 removing 500 from A's account
w(A) change to database to 500
R(B)
B=B+500
W(B) =2500
Commit (after committing everything will be stored in harddisk permanently)
Then final value will be
A=500
B=2500

ACID PROPERTIES

A= Atomicity (either all or none)
C= consistency
I= Isolation
D= Durability

Atomicity (either all or none)

Agar commit se pahle kuch v galt hota hai then it will be roll back to previous state
Commit hone ke baad koi transaction fail nahi hote because commit ke baad hard disk me permanently save ho jaata hain
A fail transaction can not be resumed it only restart
Agar transaction 99 pe v fail hota hai to pure ke pure roll back hoge

Consistency

Before the transaction start and after the transaction complete , sum of the money should be the same
Let take for above example
Sum of the money before transaction start is A+B= 3000
Sum of the money after transaction end is A+B = 3000
So then before==After (consistency is valid here)

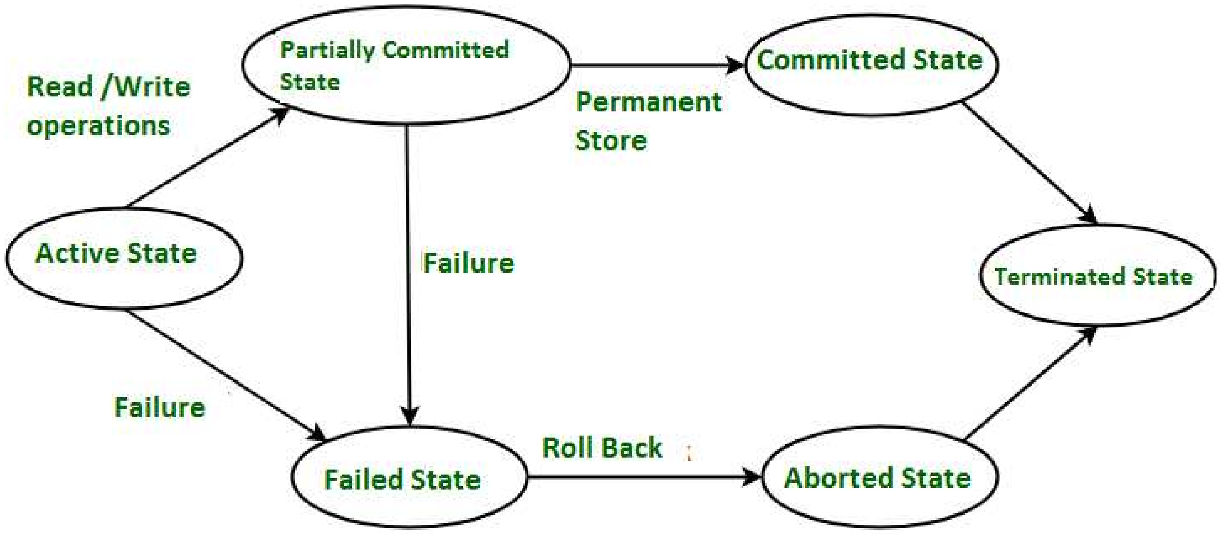
Isolation

It tries to convert parallel schedules to serial schedules , when a lot of parallel scheduling is running . And serial schedule always consistent hota hai

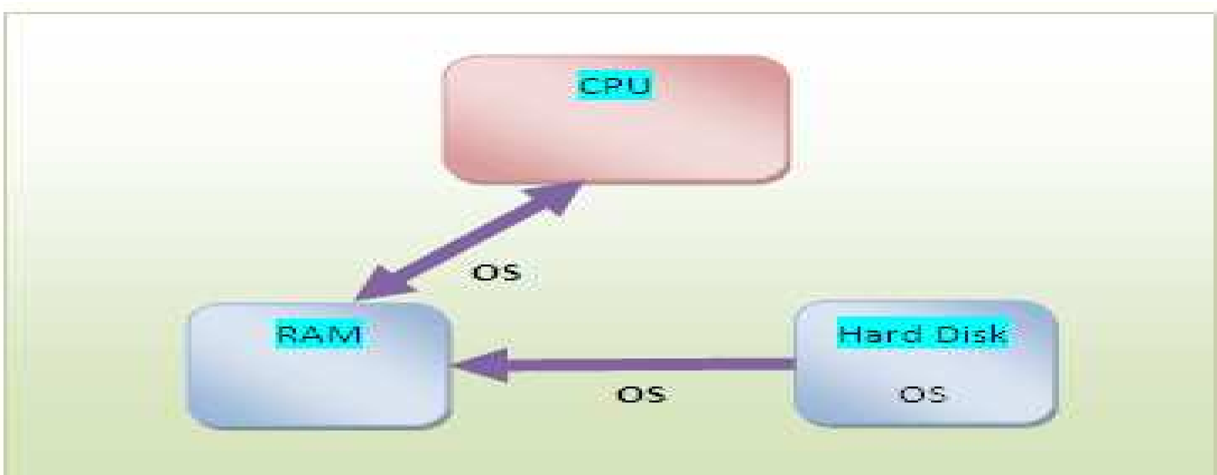
Durability

Data base me jitne v changes ho rahe hain oo saare ke saare permanent hone chahiye

Transaction states



Transaction States in DBMS



Active state = that means data come into the RAM from hard disk and all operations will be hold in RAM and after commit it will permanently save to hard disk ;
RAM= local memory or shared memory

Partially commit = let suppose to n operations there including commit then n-1 operations completed and remaining only commit so that is called partially commit state
Commit me humlog hard disk ke baat karte hain

Schedule

It is chronological execution sequence of multiple transactions
Two types of schedule

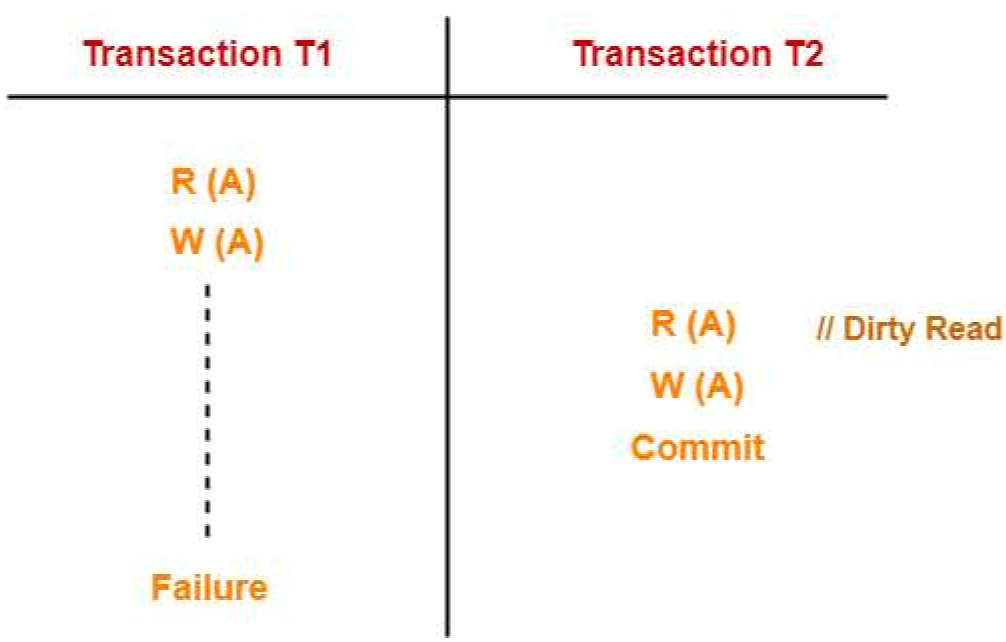
1. Serial (always consistent)
2. Parallel (multiple transaction can come together and start simultaneously)

In parallel performance and throughput is high

Types of problems in concurrency

1. Dirty read
2. Incorrect summary
3. Lost update
4. Unrepeated read
5. Phantom read

Dirty Read



Write- Read conflict(Dirty read Problem)

SCHEDULE

A=70 at starting

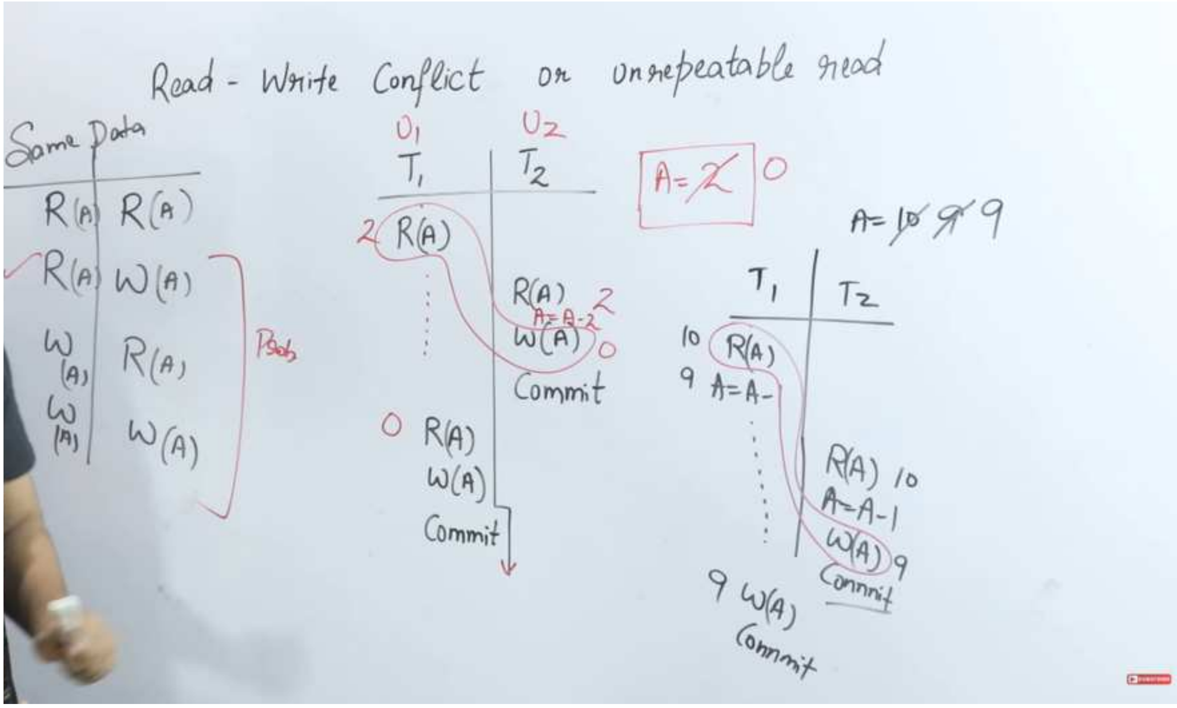
T1	T2
R(A) A= A-50; w(A) R(B) Fail W(B)	 R(A) A= A*2; w(A) 40 commit

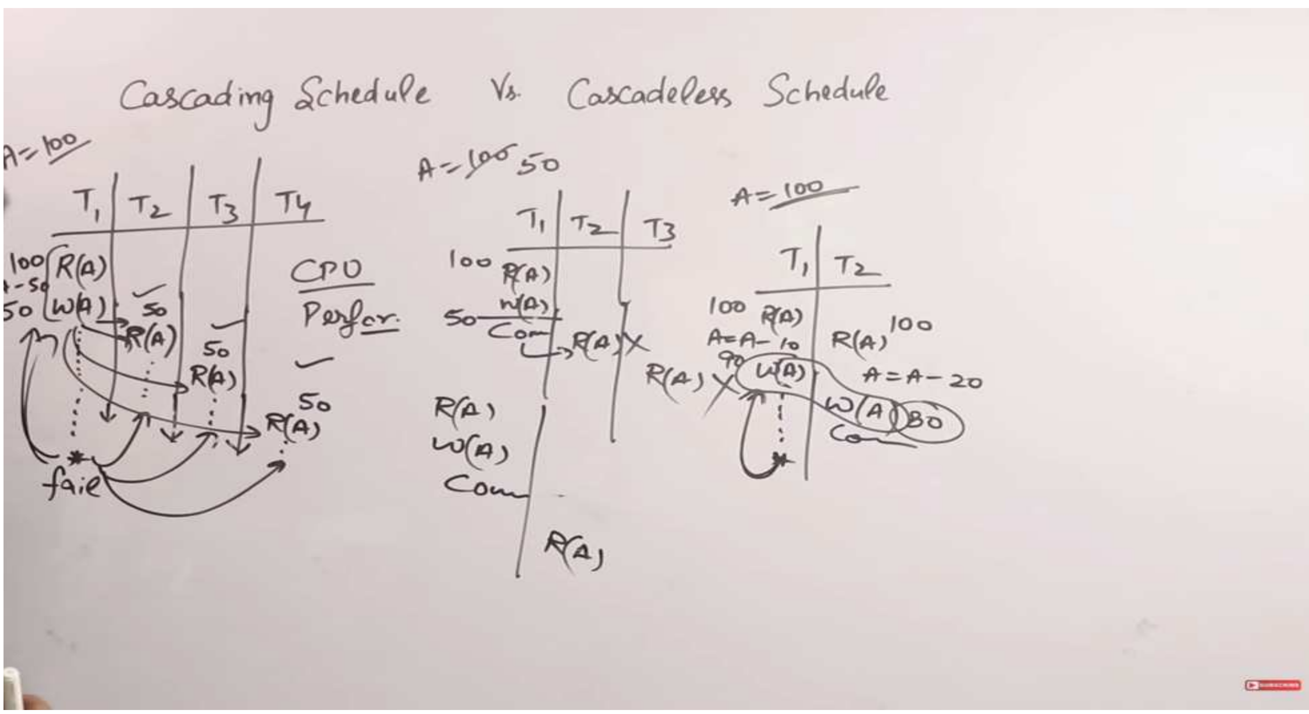
T1 fail ho gaya but T2 to commit kar diya tha that means T2 ne galt read kiya that is dirty read

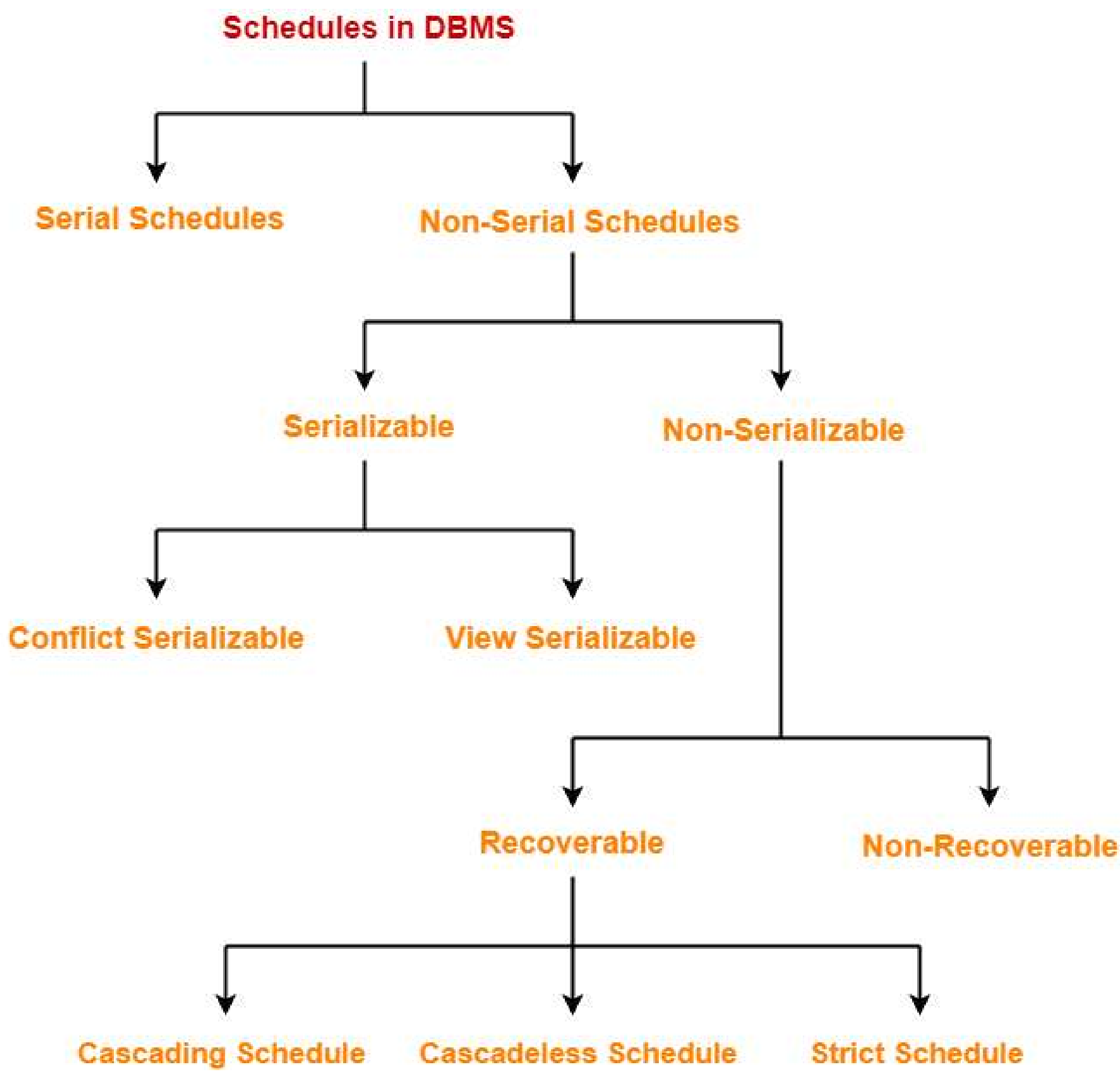
Read-Write Conflict(unrepeatable read)

A=70 at starting

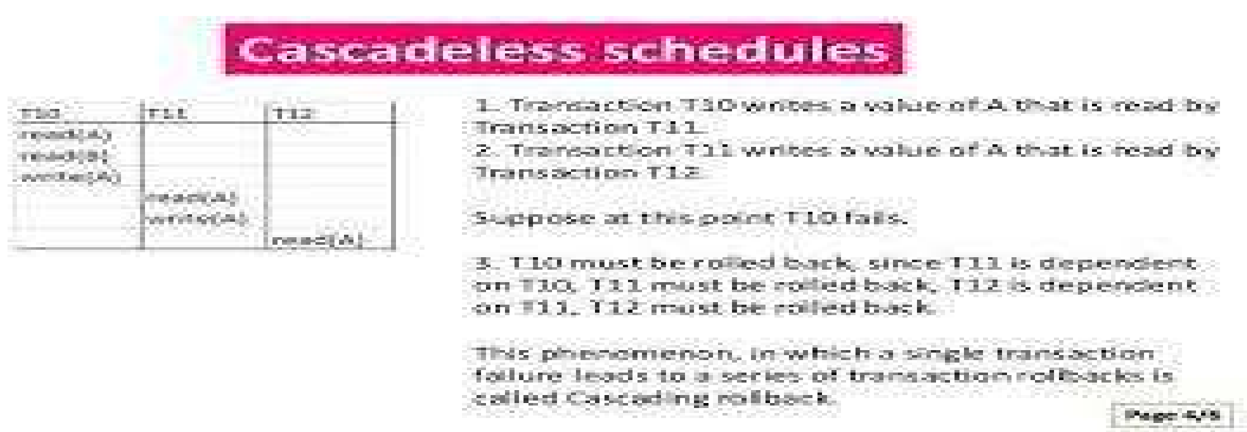
T1	T2
R(A) x R(A) y (we get two diff because ..) W(A) commit	 R(A) w(A) Commit (database update for A)


Read - Write Conflict or unrepeatable read
Same Data
R(A) R(A)
R(A) W(A)
W(A) R(A)
W(A) W(A)
Prob
U1 U2
T1 T2
R(A)
R(A)
A=A-2
W(A)
Commit
R(A)
W(A)
Commit
A=2 0
A=10 9 9
T1 T2
R(A)
A=A-
R(A) 10
A=A-1
W(A) 9
Commit
9 W(A)
Commit


Cascading Schedule Vs. Cascadeless Schedule
A=100
T1 T2 T3 T4
R(A)
W(A)
R(A)
R(A)
R(A)
fail
CPU
Perfor.
A=100 50
T1 T2 T3
R(A)
W(A)
Com
R(A)
R(A)
W(A)
Com
R(A)
A=100
T1 T2
R(A)
A=A-10
W(A)
R(A) 100
A=A-20
W(A) 80
Com



Cascading schedule Vs cascadeless schedule



Cascade-less Schedule

- Schedules requiring cascaded rollback:
  - A schedule in which **uncommitted** transactions that **read** an item from a failed transaction must be **rolled back**.
    - As shown in schedule S_e
- **Cascadeless Schedule**:
  - One where every transaction **reads** only the items that are **written** by committed transactions.
    - $r_2(X)$ in S_d and S_e must be postponed until after T_1 has committed (or aborted), thus delaying T_2 but ensuring no cascading rollback if T_1 aborts.

39

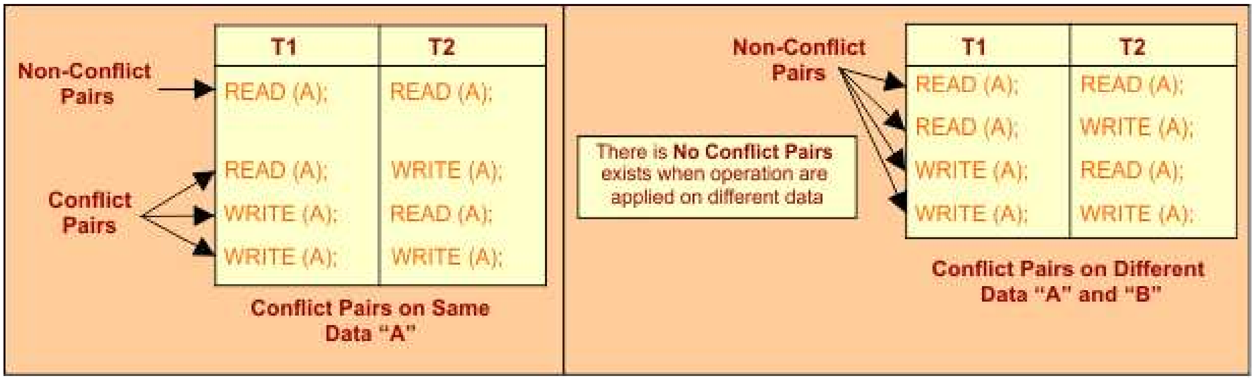
Serializability

There is two type of Serializability

1. Conflict serializability
2. View serializability

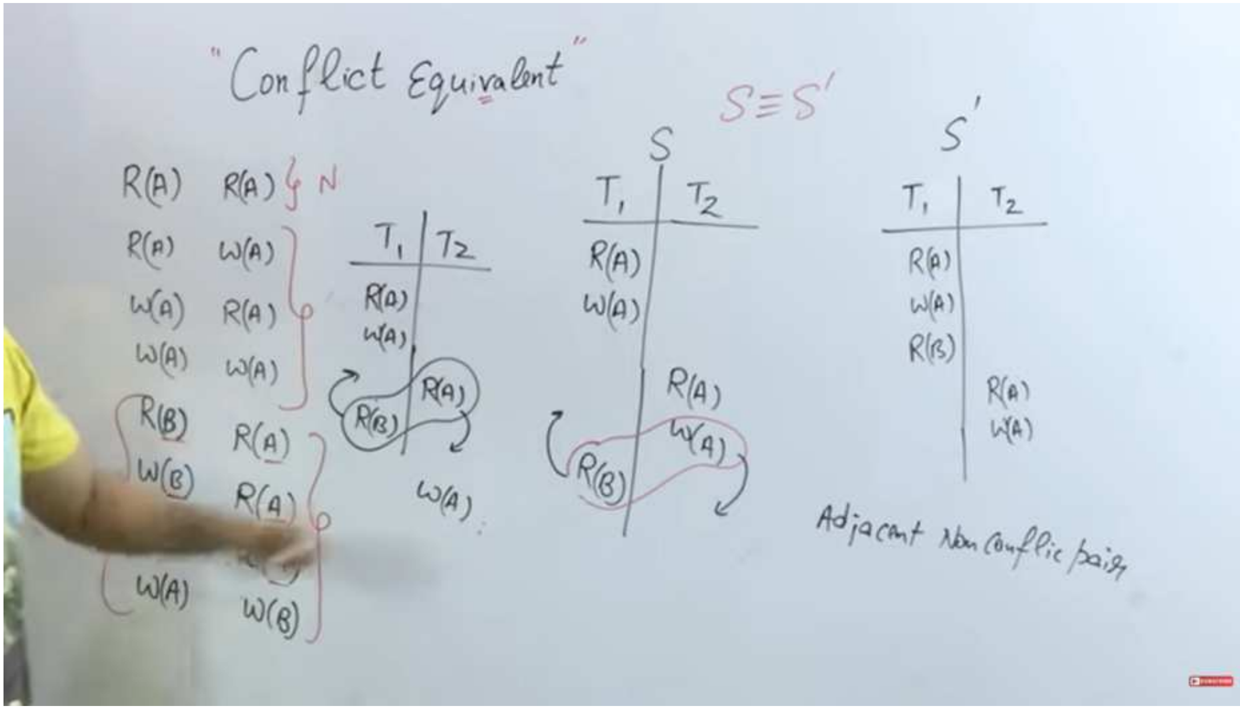
A schedule is serialized if it is equivalent to a serial schedule. A concurrent schedule must ensure it is the same as if executed serially means one after another. It refers to the sequence of actions such as read, write, abort, commit are performed in a serial manner.

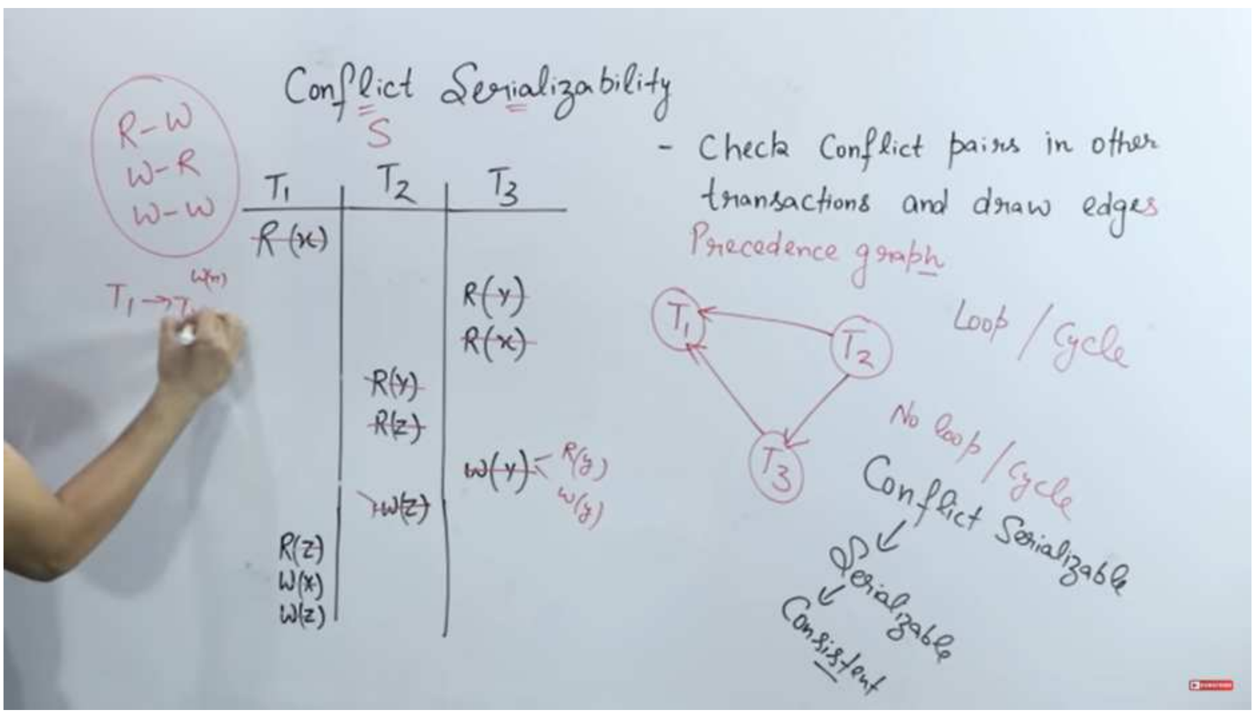
Here we need to find a clone of a parallel schedule , which should be a serial schedule .



Conflict & Non-Conflict Paris in DBMS

Transaction T1	Transaction T2
R (A)	
W (A)	
R (B)	
W (B)	
Commit	
	R (A)
	W (B)
	Commit





For order we should check INDEGREE

T2(0)->T3(1)->T1(2) , yaha pe sabse pahle T2 complete hoga then T3 then T1

(View serializable remaining)

Locking

Compatibility chart

Lock request type

State of the luck	Shared	Exclusive
Shared	Yes	No
Exclusive	No	No

Shared-exclusive locking (1st locking protocol)

Shared Lock(s)=> if transaction locked data item in shared mode then allowed to read only .

Exclusive Lock(x) =>if transaction locked data item in exclusive mode then allowed to read and write both

Problems in shared -exclusive locking

1. May not sufficient to produce only serializable schedule
2. May not free from Irrecoverability
3. May not free from deadlock
4. May not free from starvation

Agar kisi transaction pe only read karna hai toh Shared lock se hi kaam chal jayega no requirement to take exclusive lock

For example

T1
S(A) R(A) U(A)

S(A) = shared lock on A

R(A) = read operation on A

U(A) = unlock operation on A after finishing the read operation

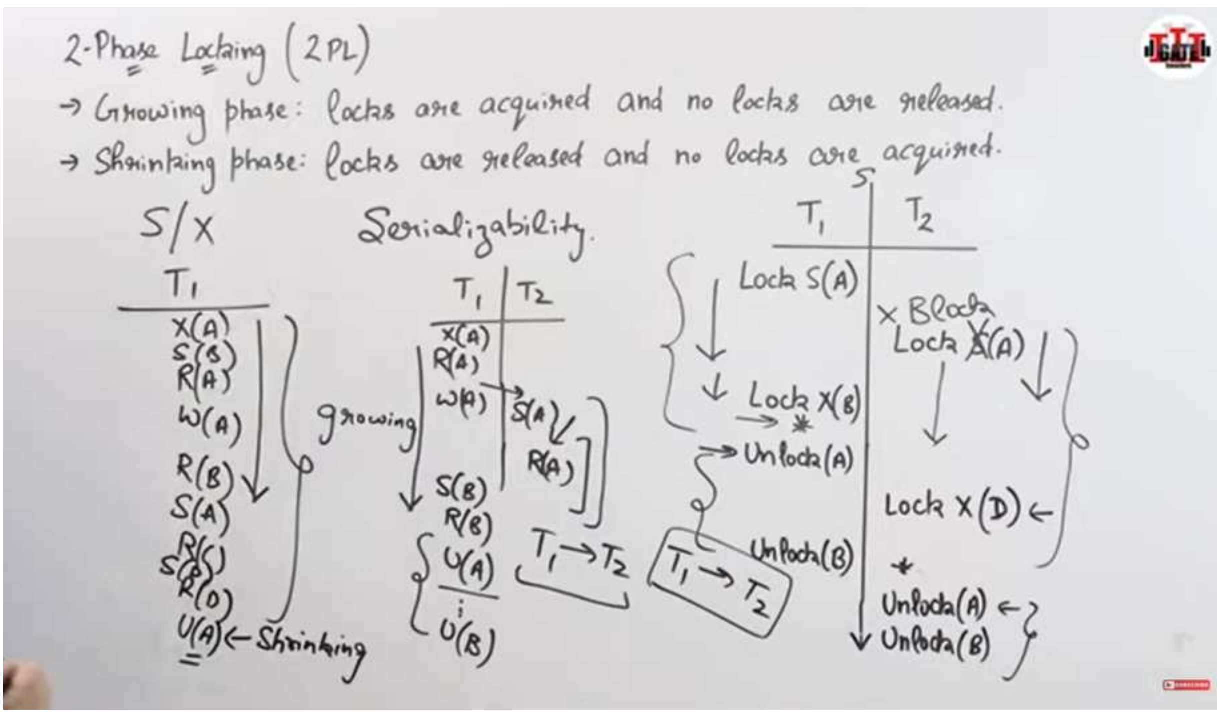
T2
X(A) R(A) W(A) U(A)

Here I have to perform Read and write both on A , so here we need exclusive lock on A for read and write both operations

2-phase locking (2PL)

Growing Phase : locks are acquired and no locks are released
Shrinking phase : locks are released and no locks are acquired

2PL transaction always serializable



S(A) = shared lock on A
X(A) = exclusive lock on B

First always try to growing and after shrinking there is no chance of growing
** star is locking point
Last waala me agar S(A) ko X(A) se replace kar de toh saara T2 block ho jayega , because they have no permission to lock this T2 because T1 already taken the permission

Advantage

Always ensures Serializability

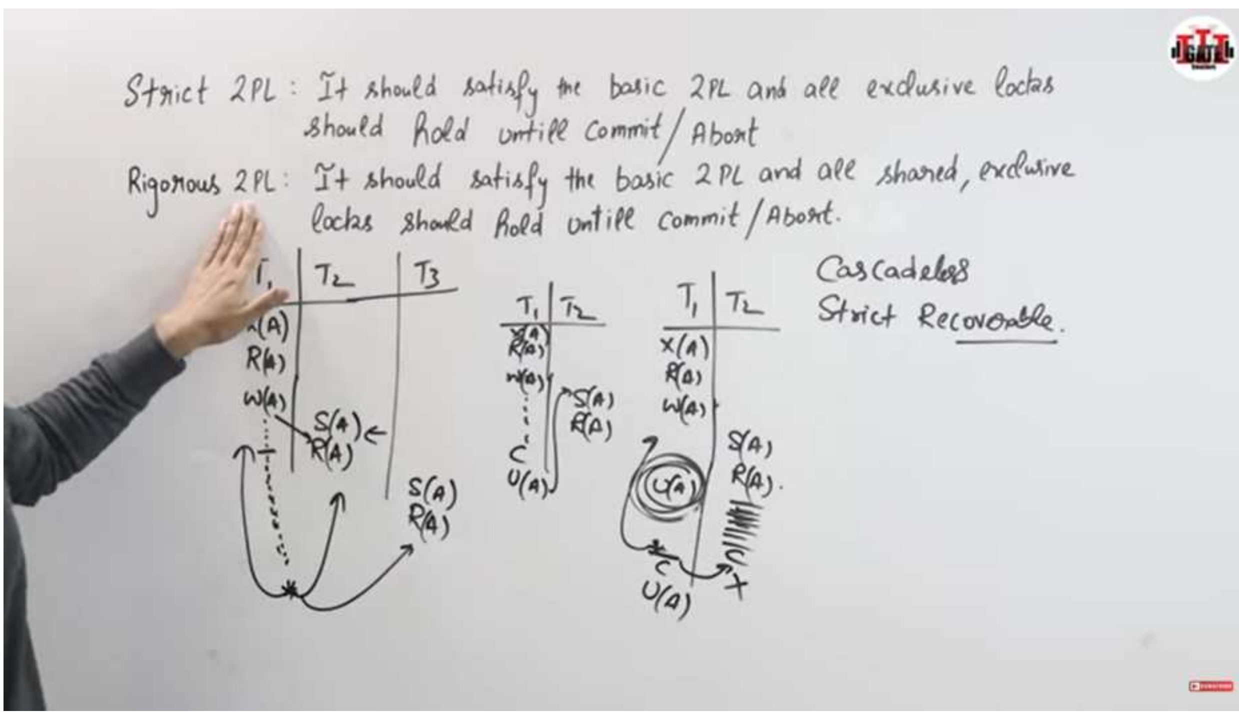
Drawbacks:

- May not free from irrecoverability
- Not free from deadlocks
- Not free from Starvation
- Not free from cascading rollback

Strict 2PL or Rigorous 2PL

Strict 2PL = It should satisfy the basic 2PL and all exclusive locks should hold until commit/ Abort

Rigorous 2PL = It should satisfy the basic 2PL and all shared , exclusive locks should hold until commit/ Abort



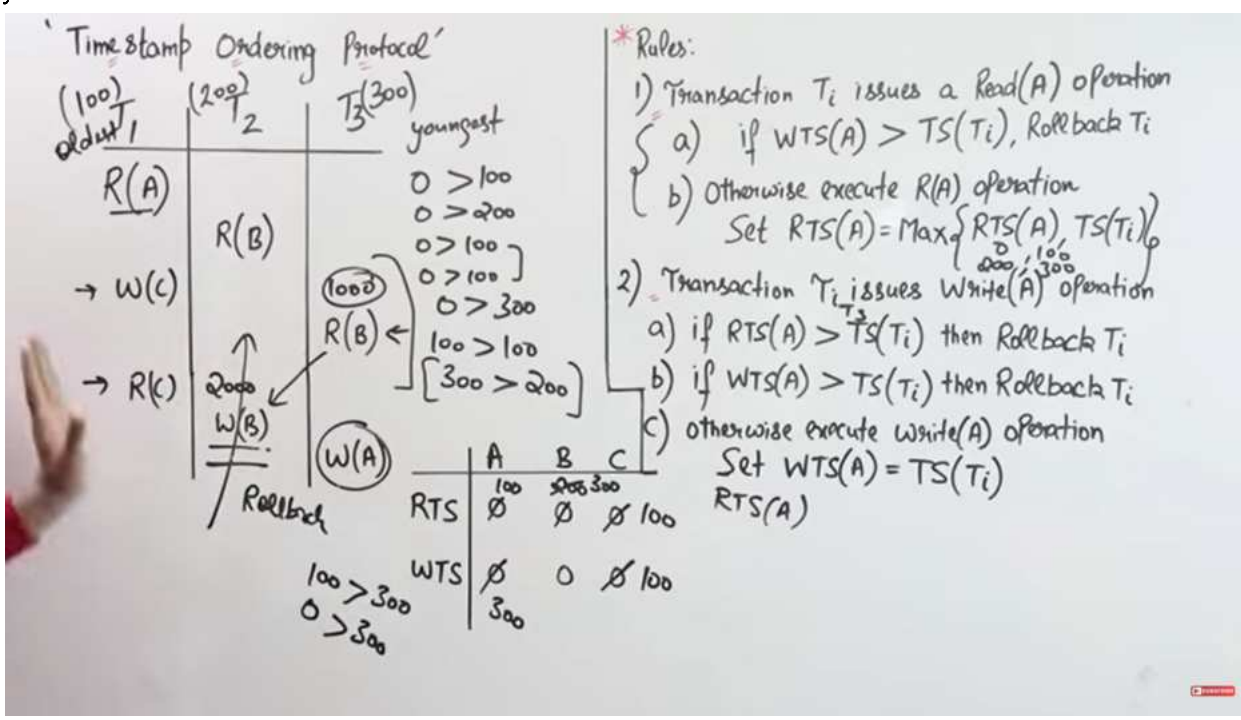
Timestamp ordering protocol

- Unique value assign to every transaction
- Tells the order(when they enters into system)
- Read_TS(RTS) = Last(lastest) transaction no which performed read successfully
- Write_TS(WTS)= last(latest) transaction no which performed write successfully

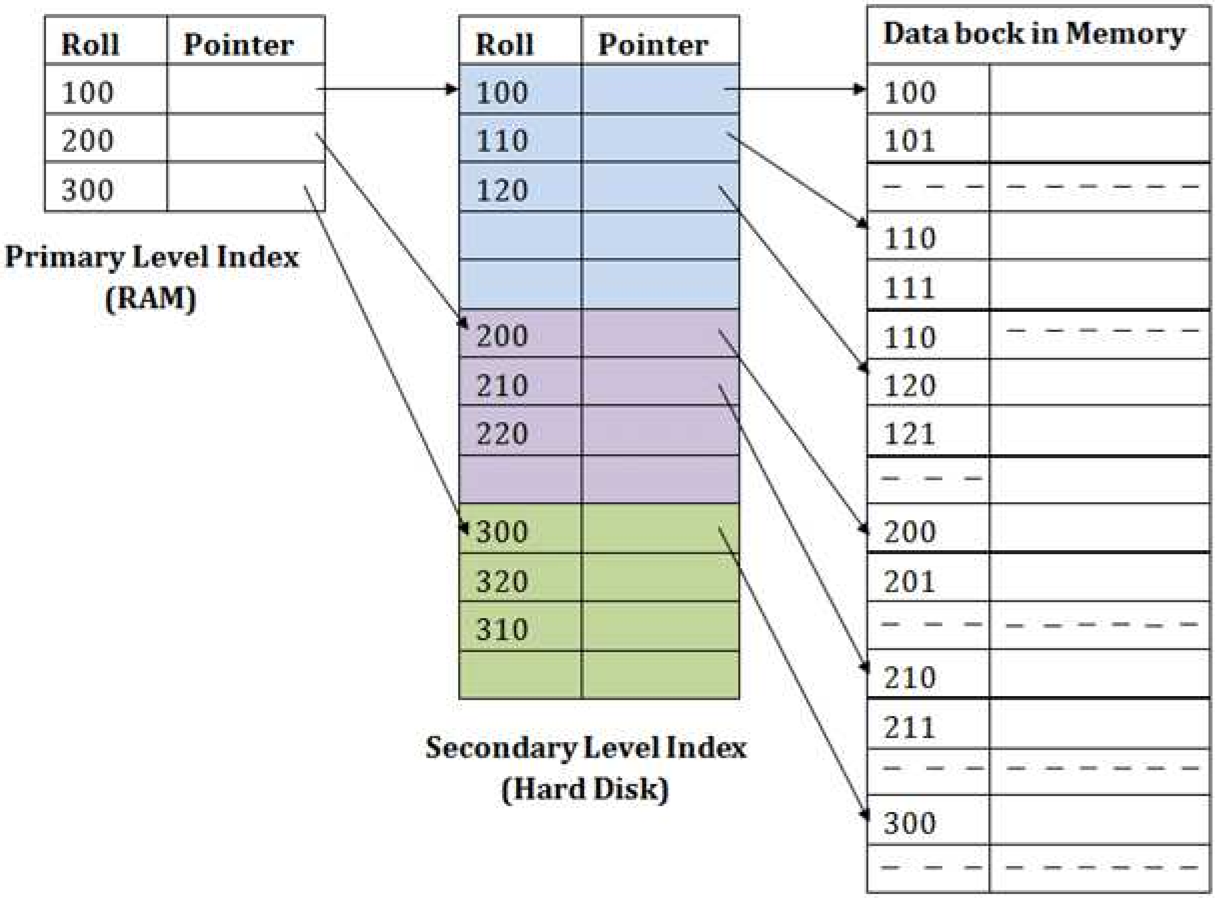
- Rules

1. Transaction Ti issues a Read(A) operation
   a) if WTS(A) >TS(Ti) , Rollback Ti
   b) otherwise execute R(A) operation
   Set RTS(A)= Max{ RTS(A), TS(Ti)}
2. Transaction Ti issues write(A) operation
   a) If RTS(A)> TS(Ti) then rollback Ti
   b) If WTS(A)> TS(Ti) then rollback Ti
   c) otherwise execute write(A) operation set WTS(A)=TS(Ti)

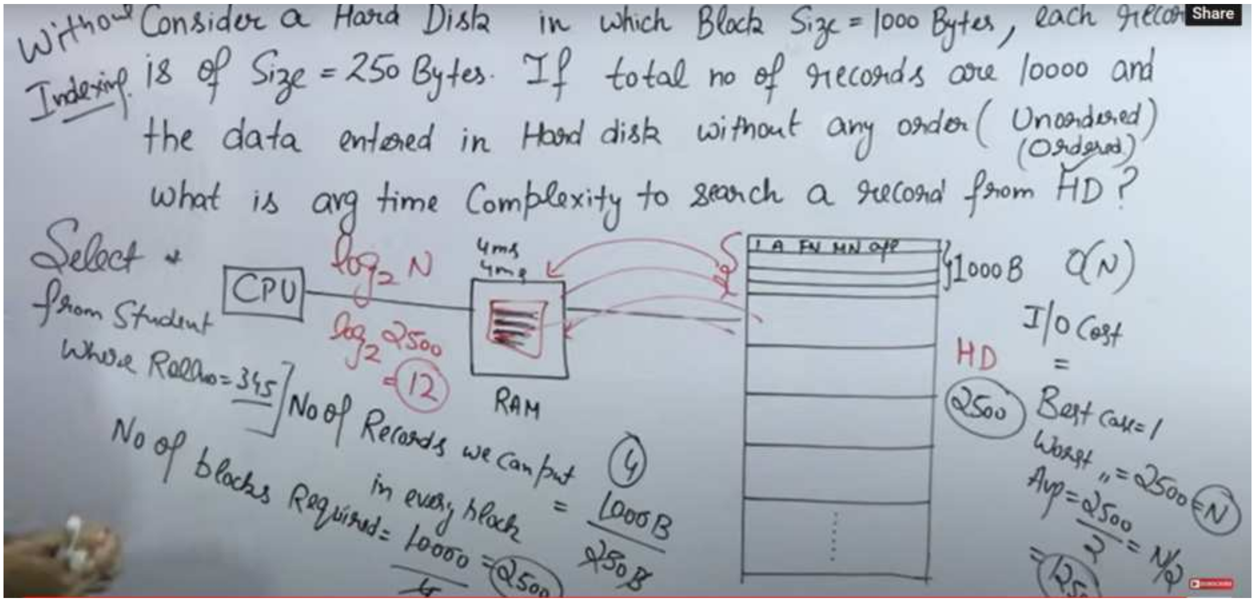
AGAR YOUNGER(jo baad me aaya) ne pahle Read ya write kiya or uske baad older usko read ya write karna chahta hai then we should not allow them .

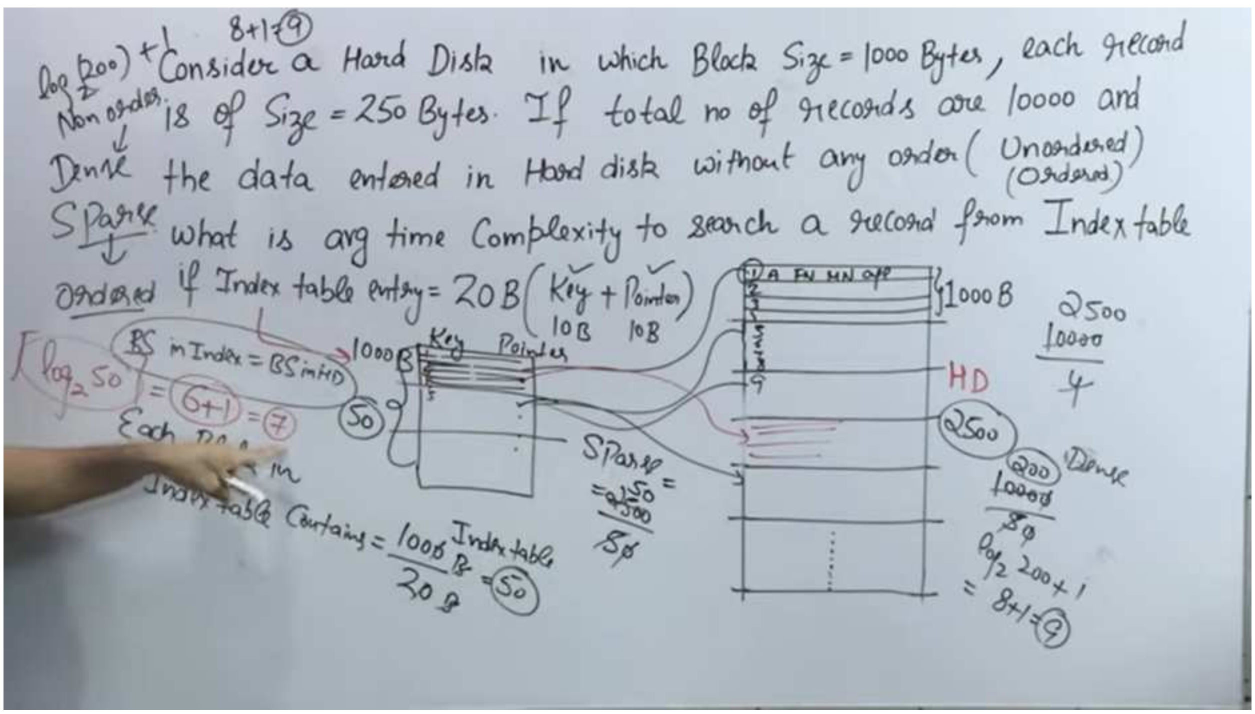


INDEXING



Indexing reduce I/O cost
Question :
Consider a Hard disk in which block size =1000 Bytes, each record is of size = 250 bytes . If total no of records are 10000 and the data entered in hard disk without any order (unordered) what is avg time complexity to search a record from HD ?





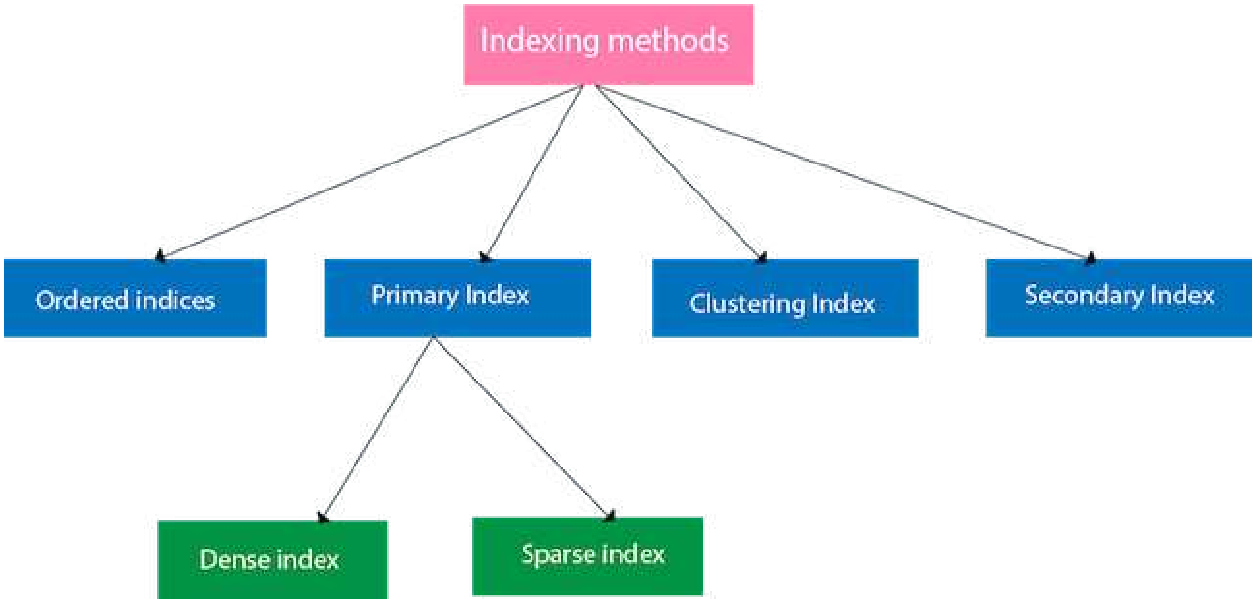
Using indexing

- Dense block = for non order
- Sparse block = for order block

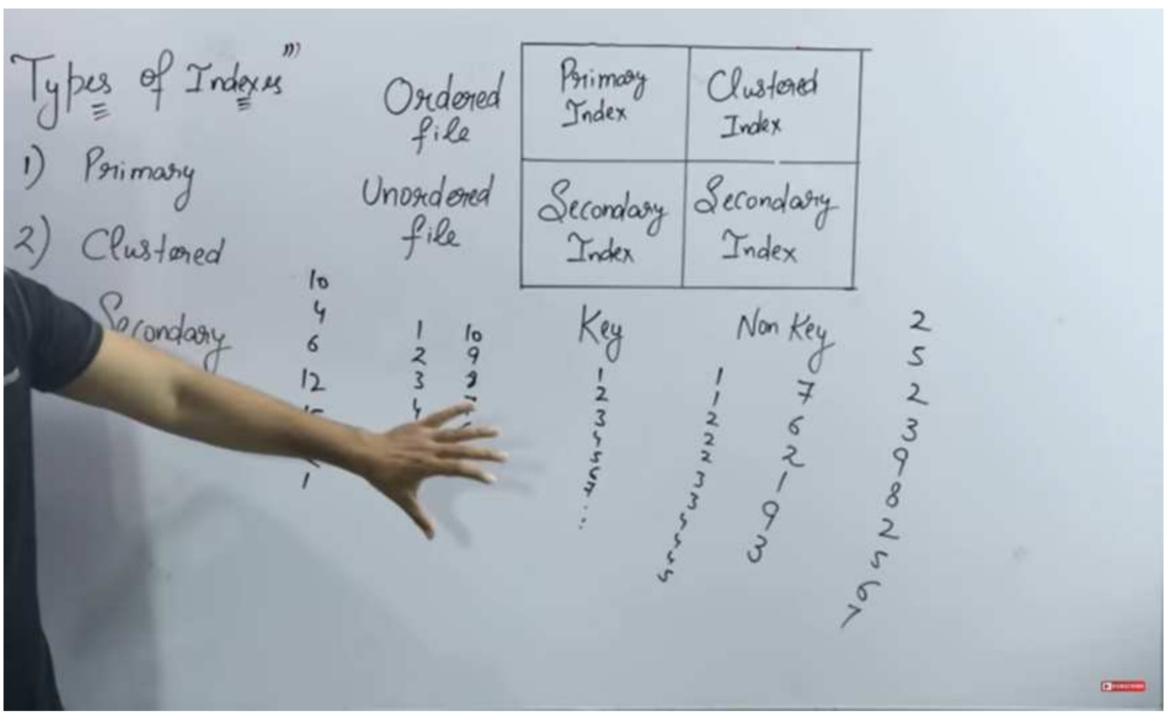
Primary index	Clustered index
Secondary index	Secondary index

Key(unique value) Non key(repeating data)

1st row = ordered file(sorted data) and 2nd row= unordered file

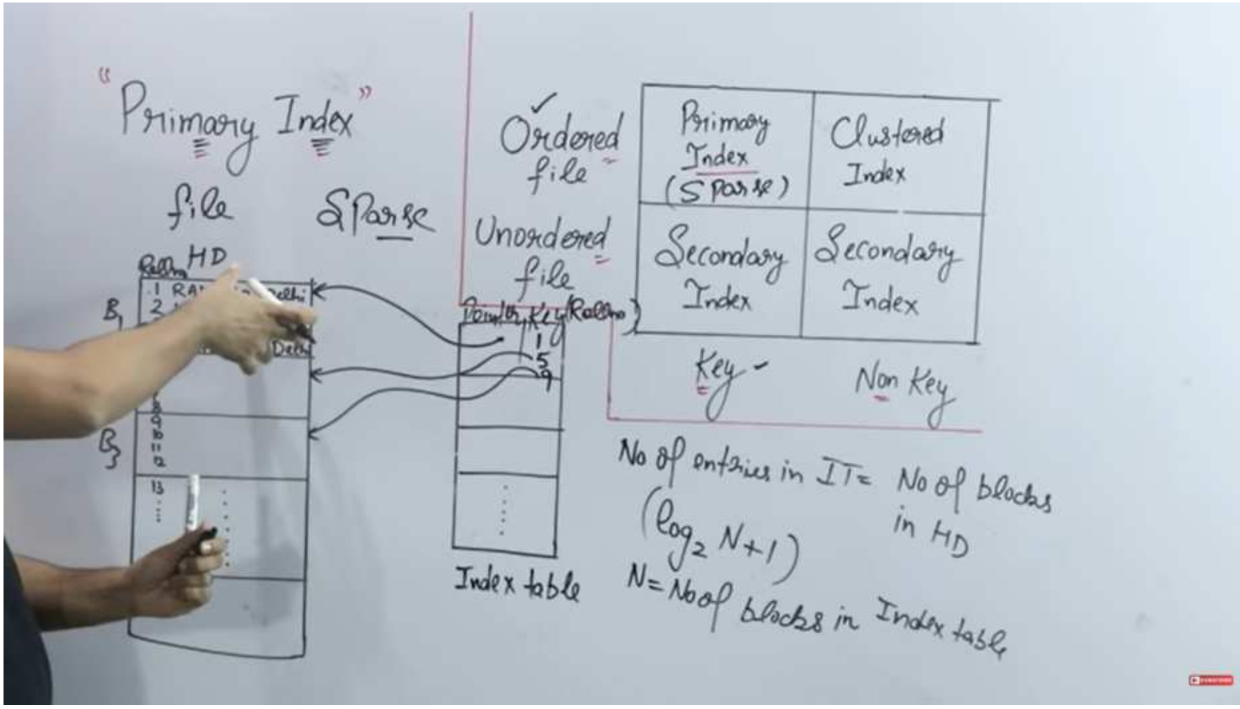


Indexing methods
Ordered indices
Primary Index
Clustering Index
Secondary Index
Dense index
Sparse index

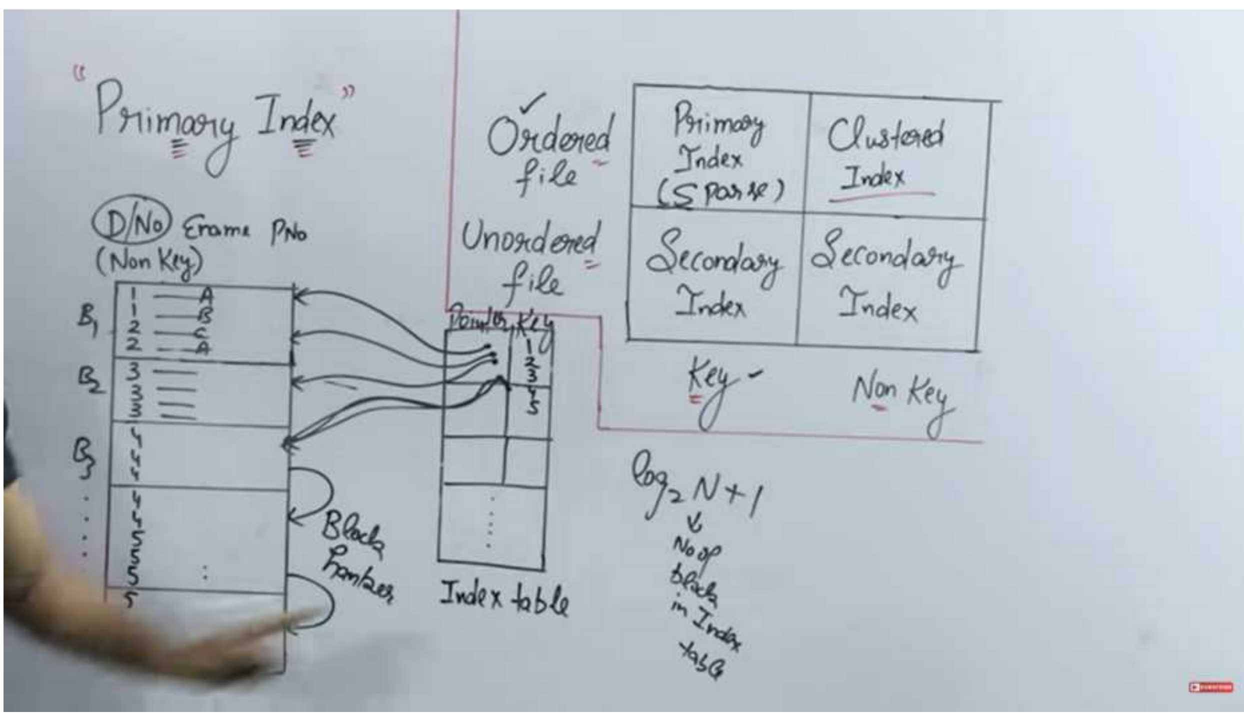


Types of Indexes
1) Primary
2) Clustered
Secondary
Ordered file
Unordered file
Primary Index
Clustered Index
Secondary Index
Secondary Index
Key
Non Key

Primary Index

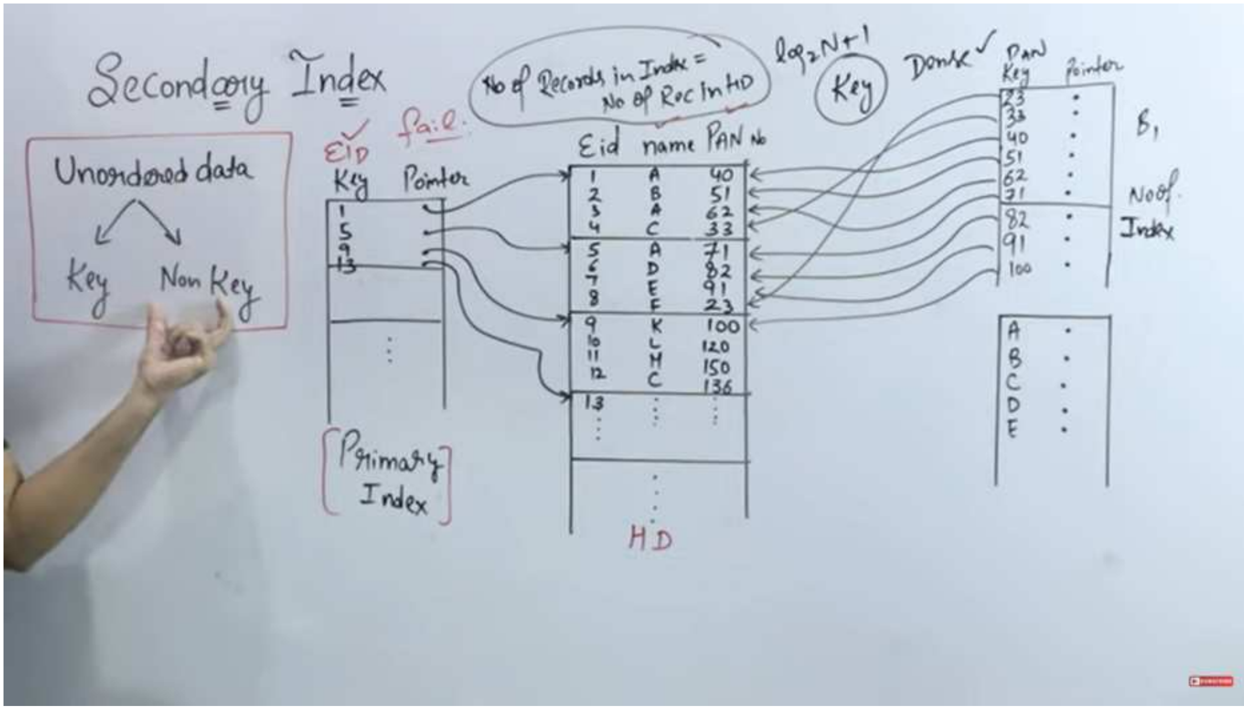


Cluster index

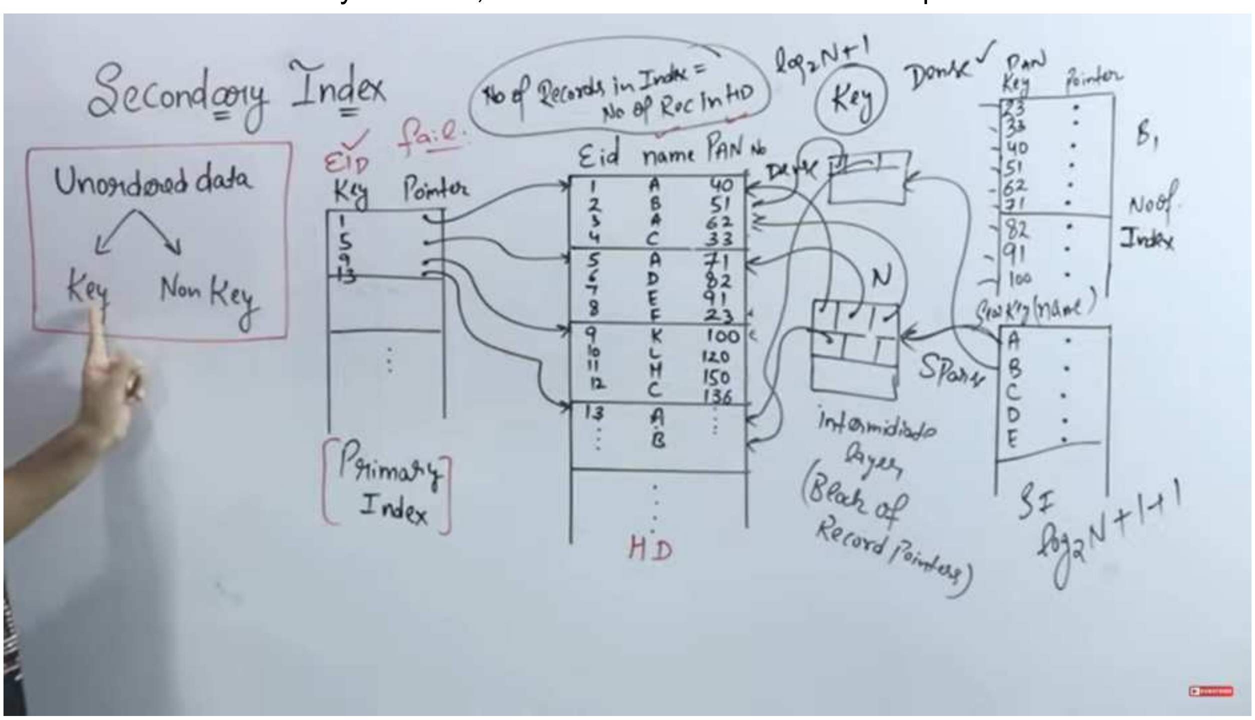


Clustered index v sparse hota hai

Secondary index

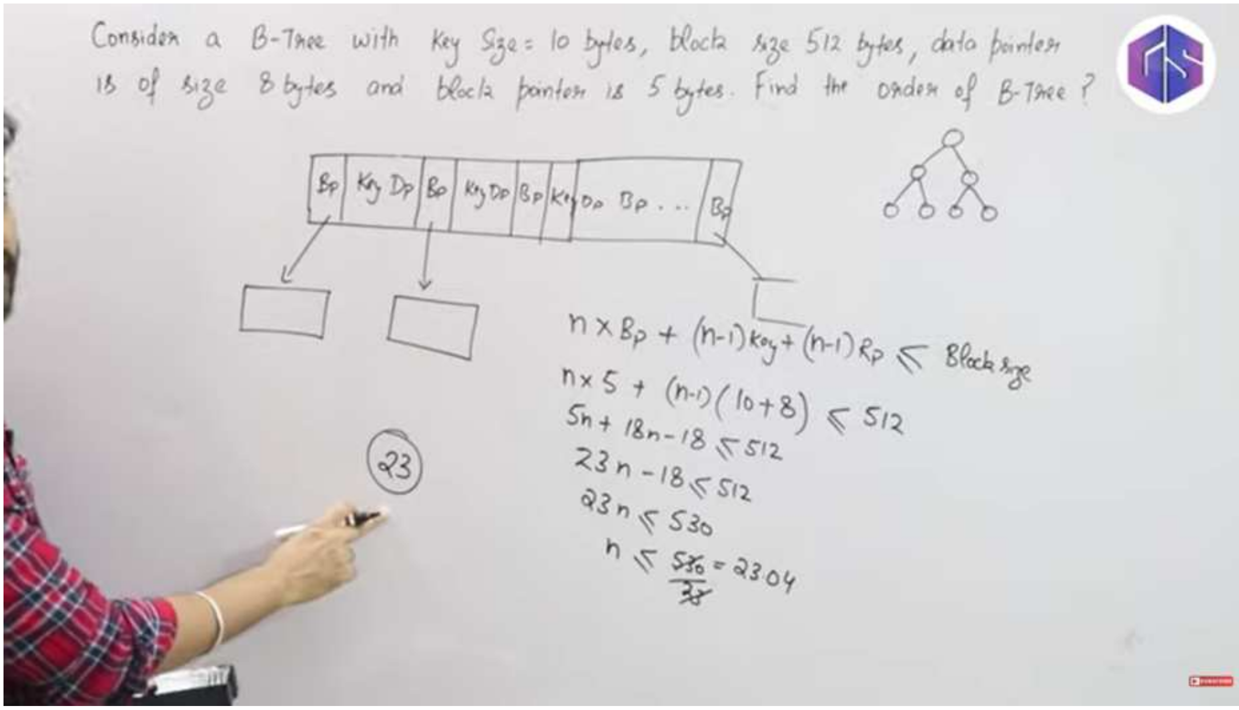


Here PAN No is key attribute , that means its non order and unique

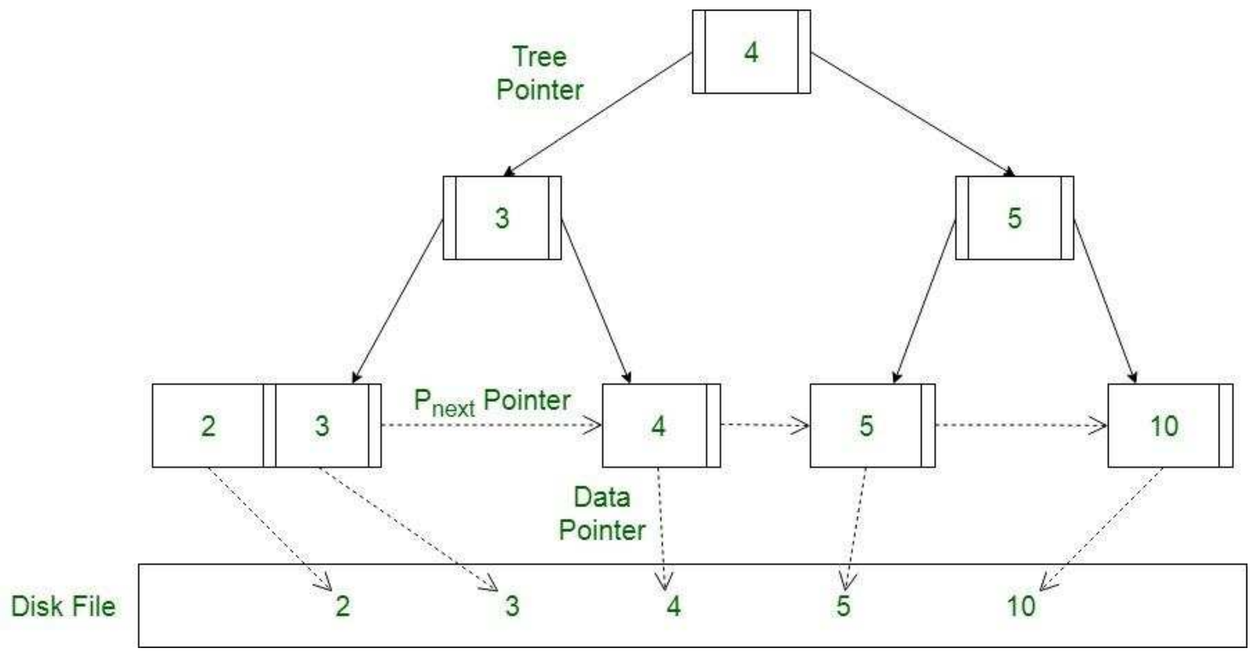


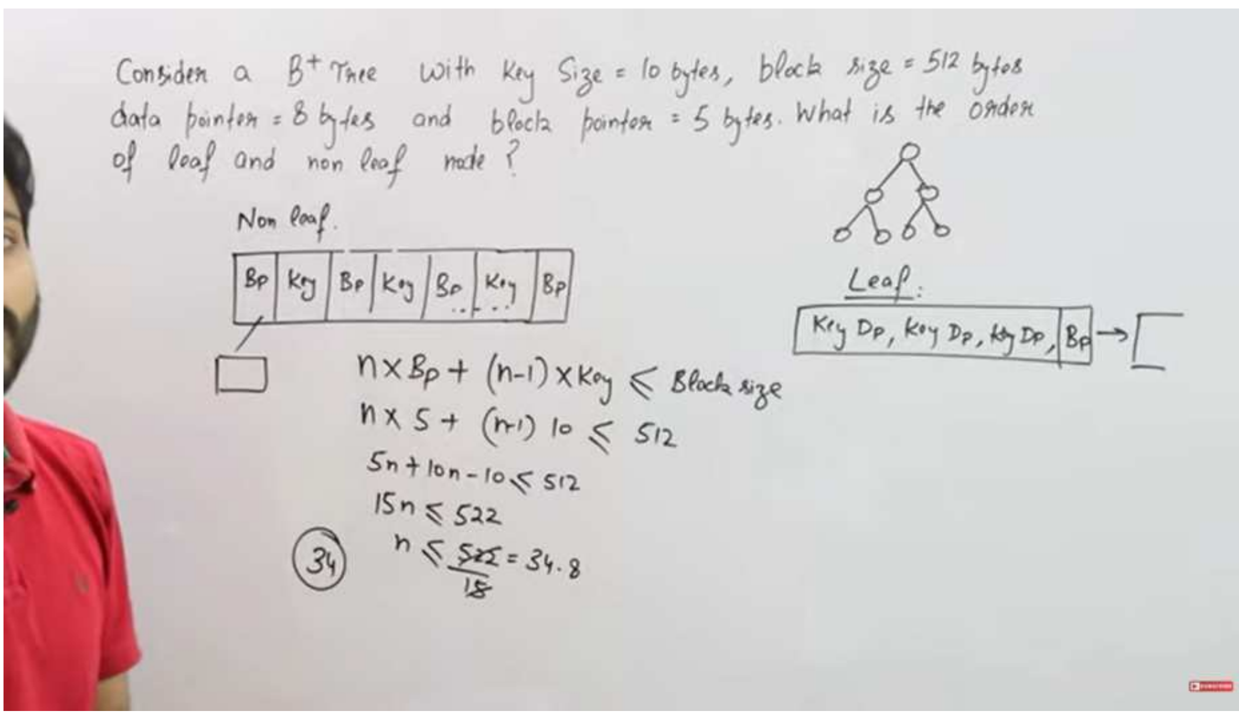
Here indexing done on the basis of Non key attribute that is name (this is repeating and not unique) or ye dense indexing hota hai secondary indexing

B-Tree



B-Tree	B+ Tree
Data is stored in leaf nodes as well as internal nodes.	Data is stored only in leaf nodes.
Searching is a bit slower as data is stored in internal as well as leaf nodes.	Searching is faster as the data is stored only in the leaf nodes.
No redundant search keys are present.	Redundant search keys may be present.
Deletion operation is complex.	Deletion operation is easy as data can be directly deleted from the leaf nodes.
Leaf nodes cannot be linked together.	Leaf nodes are linked together to form a linked list.


Tree Pointer
4
3
5
2
3
Pnext Pointer
4
5
10
Data Pointer
Disk File
2
3
4
5
10


Consider a B+ Tree with Key Size = 10 bytes, block size = 512 bytes data pointer = 8 bytes and block pointer = 5 bytes. What is the order of leaf and non leaf node?
Non leaf.
Bp Key Bp Key Bp Key Bp
n × Bp + (n-1) × Key ≤ Block size
n × 5 + (n-1) 10 ≤ 512
5n + 10n - 10 ≤ 512
15n ≤ 522
n ≤ 522/15 = 34.8
34
Leaf:
Key Dp, key Dp, key Dp, Bp