

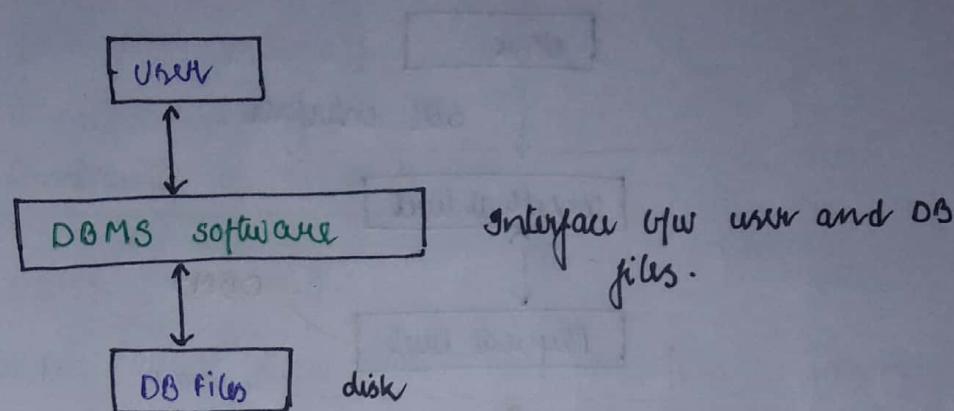
3/11/17

Brief Introduction —

Database — collection of related data.

Eg - set of students info.

DBMS — sw used to manage and access database files in an efficient way.



Flat file system —

(OS file system)

DB files stored in disk are managed without DBMS (i.e. using OS only).

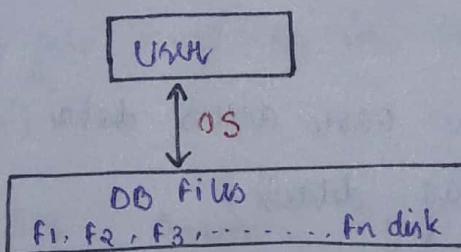


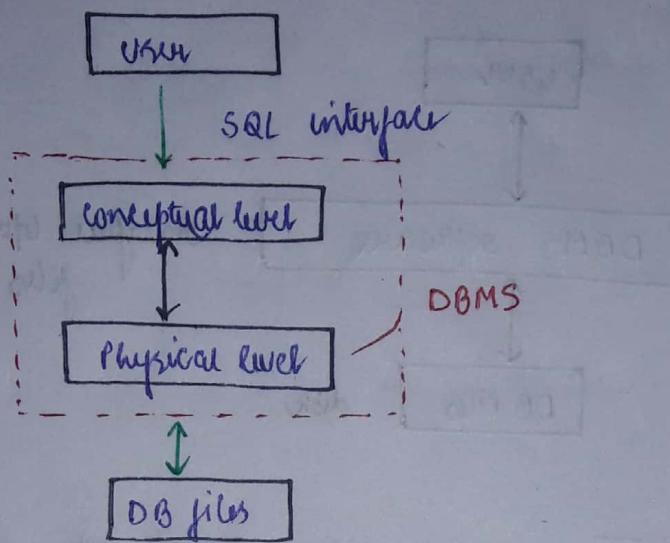
Fig: flat file system

- It fails to manage large DB : If DB size is in GB's / TB's then, flat file system fails to manage DB
- It manages small DB : If DB size is in KB's or MB's, then it can manage using flat files

Disadvantage —

- Too complex to manage storage details of DB files by the user and too difficult to develop application programs and manage application programs.

Advantage of DBMS File system —



- No need to know storage details by user as all details are known to DBMS.
- Data independence : User can access data without storage details.
- Easy to access data.

#

Flat file system

- i) complex to manage storage details of DB files by user.

DBMS file system

- No need to manage storage details of DB files by the user.

- 2) more access cost, to access data from DB files.
- 3) degree of concurrency is very less.
- 4) Too complex to implement different levels of access control (security).

DBMS maintains indexing to reduce access cost.

more degree of concurrency

easy to implement levels of access control because of virtual tables (views)

Integrity constraints of RDBMS —

Relational DBMS [RDBMS] :

- ① widely used data model
- ② R.J Codd (also called as Codd Data Model)
- ③ Codd proposed 12 rules to design RDBMS s/w.

Rule 1 —

Data in DB file must be in tabular format (set of rows and columns).

student →	Sid	Sname	Age
	S ₁	A	20
	S ₂	B	20
	S ₃	C	21
	S ₄	A	22

↑
DBMS file

student
S ₁ , A, 20 # S ₂ ,
B, 20 # S ₃ , C,
S ₁ # S ₄ , A, 22
— — — —
— — — —

↑
Flat file
(OS file)

- No 2 records of DB file is same

Attribute / fields

sid	sname	Age
S ₁	A	20
S ₂	B	21
S ₃	C	22

record / tuple

column name → Attribute / field

row → record / tuple

Cardinality: No. of records of DB file

Arity: No. of fields of DB file (table)

Relational Schema —

definition of DB Table (or) structure of DB table.

student (sid , sname , age)

↑

table name { field name }

Relational instance :

- Rows set of the DB file (also called as snapshot)
- It changes more frequently as compared to relational schema

Candidate Key —

Minimal set of attributes used to differentiate records of table uniquely.

- ① student (Sid , sname , age)

	↑		
S ₁	A	20	$\{ \text{Sid} : \text{candidate key} \}$
S ₂	B	21	$\{ [\text{sid}, \text{sname}] : \text{Not candidate key} \}$
S ₃	C	22	
S ₄	D	20	

Not minimal

- ② Enroll (sid , cid , fee)

student can enroll many courses and course can be enrolled by many students. i.e

\downarrow same sid with many cid and vice versa

candidate key \rightarrow [sid , cid] as

Sid	Cid	fee
S ₁	C ₁	5000
S ₁	C ₂	6000
S ₂	C ₂	8000
S ₃	C ₂	2000

- ③ R

A	B	C
4	6	8
4	5	8
4	6	4
4	5	4
5	6	8

ABC: candidate key

candidate key \rightarrow A X AB X ABC ✓
 B X BC X
 C X AC X

→ x is some set of attributes of R and
 x is candidate key of relation R iff -

- ① No two records with same field values over "x" attributes (unique)

and ② No proper subset of x can differentiate the records uniquely (minimal)

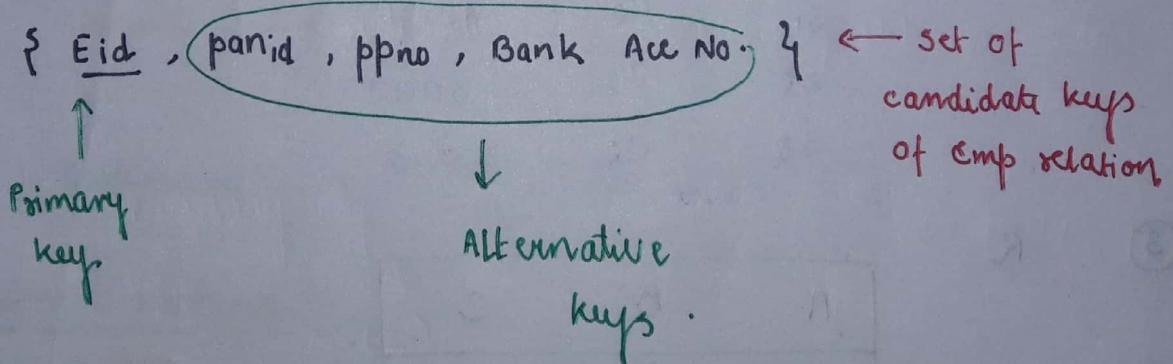
$R (\text{---} \dots \text{---})$
↑
 x
candidate key

if ABC is candidate
key then
ABC

④ Emp -

Eid	Ename	DOB	panid	ppno	Bank	Acc No.
					SBI	1001
					BOI	1001
					SBI	1002

Assume NO 2 emp's with same Bank and Acc No.



- One of candidate keys is assigned as primary key and remaining all candidate keys are called as Alternative keys
- NULL — Unknown value / Un existed value
If any value is not assigned or given by user it is assigned as NULL.

Primary key -

- Any one candidate key of relation whose field values must be not NULL
- Every field of primary key must be not null (NULL value not allowed for any primary key field)
- for any DB relation atmost one primary key is allowed.

Alternative keys -

Any one candidate key of relation whose field values must be not NULL.

Every field of primary key must be not null (NULL value not allowed for any primary key field)
for any DB relation atmost one primary key is allowed.

- All candidate keys of relation except primary key
- Alternative key fields allows NULL values
- More than one alternative key is allowed to define.

NOT

Table Creation -

```
create Table Emp  
( EId    varchar(10) Primary key,  
Ename   varchar(30),  
DOB     date,  
panid   varchar(10) UNIQUE NOT NULL,  
ppno    varchar(15) UNIQUE  
bank    varchar(20),  
Acc No  integer(15),  
UNIQUE ( bank, Acc no)  
);
```

Keywords used

Primary key → Primary key
Alternative key → UNIQUE

Candidate keys

Primary key
(unique and not
NULL)

Alternative keys
(Unique)

Prime Attribute -

Attribute which belong to any candidate key of relation.

Prime attribute set : set of all candidate key's attribute of
relation / table.

Eg - in Table Emp.

Non prime attribute set = { Eid, fname, ppno, bank, accno }

Non prime attribute — (Non key attribute)

Attribute which does not belongs to any candidate key of the relation.

Non-prime attribute set — set of all non prime attribute of relation R.

Eg - in table Emp.

Non prime attribute set: { Ename, DOB }

→ For any RDBMS table, there must be atleast one candidate key whose field value must be NOT NULL.

For eg → create table R

```
( A int UNIQUE,  
  B int UNIQUE,  
  C int  
);
```

X Not Allowed



Two candidate keys
but none is both
unique and NOT NULL.

(Atleast one key should be
unique and not null.)

R (A, B, C)

2	NUL	-
4	3	-
6	4	-
NUL	5	-

Relation R
not in RDBMS.

Eg. - create table R

```
( A int UNIQUE NOT NULL,  
  B int UNIQUE,  
  C int  
);
```

✓ Allowed
(for RDBMS)

Primary key → NO

Candidate keys → 2 (A, B)

Eg. - create table R

```
( A int primary key,  
  B int UNIQUE,  
  C  
);
```

✓ Allowed for RDBMS
(preferred)

Primary keys → 1 (A)

Candidate keys → 2 (A, B)

Alternate keys → 1 (B)

Note -

UNIQUE NOT NULL \neq PRIMARY KEY

It means
alternative key
which does not
allow NULL value

Super Key -

- set of attributes of relation (R) which can differentiate records uniquely but may not be the minimal attribute set.

- candidate key is minimal super key.
- super key is not used in table implementation (theoretical approach), used only in design stage

Eg - stud (Sid , Sname , age)

if candidate key = Sid



super key = { Sid ,
Sid Sname ,
Sid age ,
Sid Sname age }



Set of candidate
keys of R .

Set of superkeys of Relation R



Every superkey is not candidate key while every
candidate key is superkey .

Ques: R (A, B, C, D)

How many super-keys in relation R if A is candidate
key ?

$${}^3C_0 + {}^3C_1 + {}^3C_2 + {}^3C_3 = \frac{3}{[2 \cdot 1]} + \frac{3}{[1 \cdot 2]} + 1 + 1$$

$$= 3 + 3 + 1 + 1 = 8 \text{ i.e. } 2^3$$

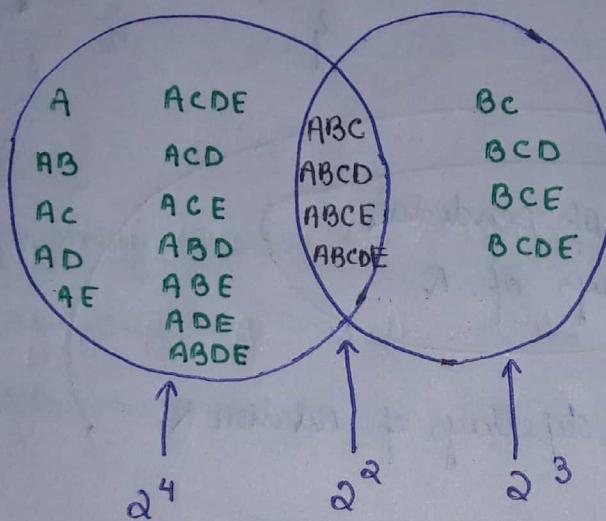
Superkey = A ∪ any subset of B, C, D)

$$= \{ A, ABC, ABCD, AB, ACD, AC, ABD, AD \} \Rightarrow 2^3 \text{ superkeys}$$

Ans

Ques: R(A, B, C, D, E). How many superkeys in Rel R if
 $\{ A, BC \}$ candidate keys.

Soln:



A, B, C, D, E

Take A
Now 4 left
 $\therefore 2^4$

Take BC
Now 3 left
 $\therefore 2^3$

For common ABC
2 left
 $\therefore 2^2$.

$$\begin{aligned} n(X \cup Y) &= n(X) + n(Y) - n(X \cap Y) \\ &= 2^4 + 2^3 - 2^2 \\ &= 16 + 8 - 4 \Rightarrow 20 \text{ Ans} \end{aligned}$$

Ques: R(A, B, C, D, E, F)

$\{ AB, BCD \}$ superkey = ?

$$\begin{aligned} \text{Soln: } \text{No. of superkey} &= \underline{AB} \{ CDEF \}^4 + \underline{BCD} \{ AEF \}^3 - \underline{ABCD} \{ EFG \}^2 \\ &= 2^4 + 2^3 - 2^2 \\ &= 20 \text{ Ans} \end{aligned}$$

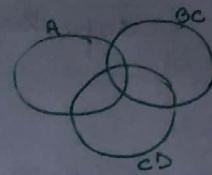
Ques: $R(A, B, C, D, E)$. How many super keys in R ; A, BC, CD are candidate key?

$$\text{superkey} = A(BEDE) + BC(ADE) + CD(ABE) \leftarrow ABC(DE) - BCD(AE) \\ - ACD(BE) + ABCD(\emptyset)$$

$$= 2^4 + 2^3 + 2^3 - 2^2 - 2^2 + 2^1$$

$$= 18 + 4$$

$$= 22 \text{ Ans}$$



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

How many super keys if

a) $\{A_1\}$ cand key.

b) $\{A_1 A_2, A_3 A_4\}$

c) $\{A_1, A_2, A_3\}$

d) $\{A_1 A_2, A_2 A_3, A_4 A_5\}$

e) If each attribute of R cand-key

Solu"

(a) superkey = $A_1 \cdot \{A_2 A_3 \dots A_n\}$
 $= 2^{n-1} A_1$

(b) superkey = $A_1 A_2 \{A_3 A_4 \dots A_n\} + A_3 A_4 \{A_1 A_2 \dots A_n\} -$
 $A_1 A_2 A_3 A_4 \{A_5 \dots\}$
 $= 2^{n-2} + 2^{n-2} - 2^{n-4}$
 $= 2^{n-4} (2^3 - 1)$
 $= 7 \cdot 2^{n-4} \text{ Ans}$

(c) superkey = $A_1 \{A_2 A_3 \dots A_n\} + A_2 \{A_1 A_3 A_4 \dots A_n\} +$
 $A_3 \{A_1 A_2 A_4 A_5 \dots A_n\} -$
 $A_1 A_2 \{A_3 \dots A_n\} - A_2 A_3 \{ \dots \} - A_3 A_1 \{ \dots \}$
 $+ A_1 A_2 A_3 \{ \dots \}$

$$\Rightarrow 2^{n-1} + 2^{n-1} + 2^{n-1} - 2^{n-2} - 2^{n-2} - 2^{n-2} + 2^{n-3}$$

$$\begin{aligned} &\Rightarrow 2^n - 2^{n-2} + 2^{n-3} \\ &\Rightarrow 2^{n-3} [2^3 - 2 + 1] \\ &\Rightarrow 7 \cdot 2^{n-3} \quad \underline{\text{Ans}} \end{aligned}$$

d) superkey = $A_1A_2 + A_2A_3 + A_4A_5 - A_1A_2A_3 - A_2A_3A_4A_5 - A_1A_2A_4A_5 + A_1A_2A_3A_4A_5$

$$\begin{aligned} &= 2^{n-2} + 2^{n-2} + 2^{n-2} - 2^{n-3} - 2^{n-4} - 2^{n-4} + 2^{n-5} \\ &= 2^{n-1} + 2^{n-2} - 2^{n-3} - 2^{n-3} + 2^{n-5} \\ &= 2^{n-1} + 2^{n-2} - 2^{n-2} + 2^{n-5} \\ &= 2^{n-5} [16 + 1] \Rightarrow 17 \cdot 2^{n-5} \quad \underline{\text{Ans}} \end{aligned}$$

e) suppose, $R(A_1, A_2, A_3)$

if each attribute of R is cand. key,

$$\{A_1, A_2, A_3, A_1A_2, A_2A_3, A_1A_3, A_1A_2A_3\}$$

$\Rightarrow 7$ superkeys

for n attributes $\Rightarrow 2^n - 1$ max possible SK.
 \downarrow
 for \emptyset

Note → No. of possible superkeys in R with n attribute
 is $= \frac{2^n - 1}{2^n}$ (if each attribute of R is cand. key)

→ if all attribute form one candidate key, then

$$\text{No. of superkeys of } R = \frac{1}{2^n}$$

Eg - $R(\underline{A_1 A_2 A_3}) \Rightarrow A_1A_2A_3 \{\emptyset\}$

$$\therefore 2^0 = 1 \text{ superkey}$$

Foreign Key (Referential Key)

"It is used to relate data b/w tables"

Eg -

stud (sid, sname, age, login)	enroll (sid, cid, fee)	Foreign key
S1	S1 C1 5000	
S2	S2 C2 5000	
S3	S3 C2 4000	
S4	S4 C3 5000	
S5	S5 C4 6000	
	X S6 C1 5000	restricted

all students

not allowed
i.e. does not exist

→ Enroll sid always belongs to stud sid.
↑
Foreign key.

Create table stud

```
(  
    sid varchar(10) Primary key,  
    sname varchar(30),  
    age integer,  
    login varchar(30) UNIQUE  
);
```

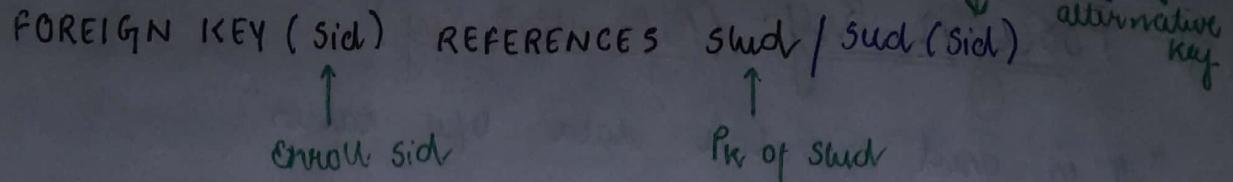
Create table Enroll

```
(  
    sid varchar(10);  
    cid varchar(10);  
    fee integer  
    Primary key (sid, cid));  
FOREIGN KEY (sid) REFERENCES  
stud (sid);
```

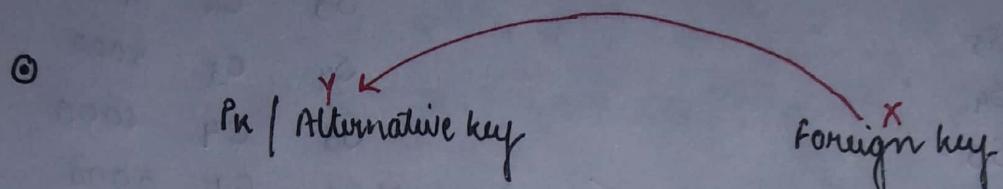
ON DELETE CASCADE
ON UPDATE CASCADE

Referential
constraint

by default, it is NO ACTION

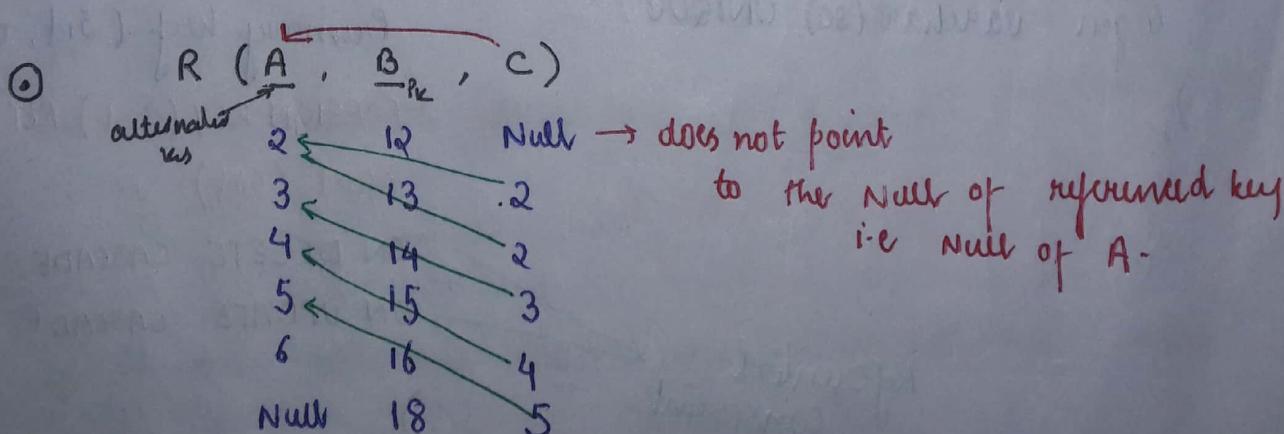
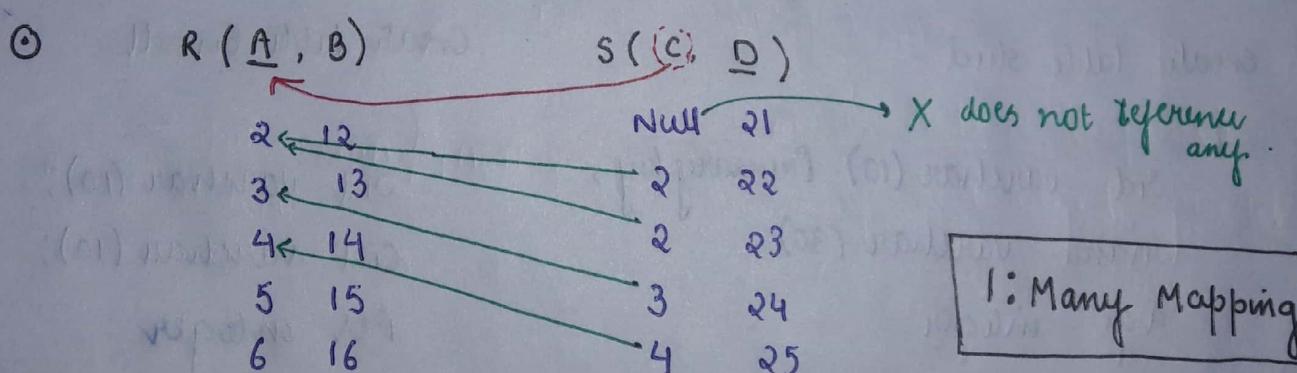


→ Foreign key: set of attributes references to primary key / alternative key of same relation or some other relation.

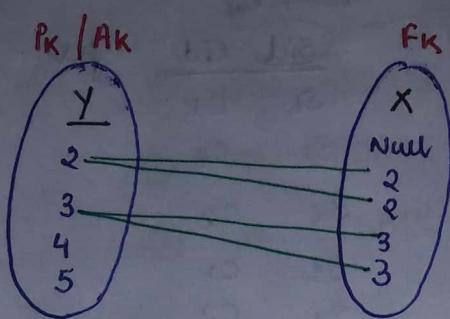


②

PK	R (A, B, C)	Foreign key
2	12	Null
3	13	2
4	14	2
5	15	3
7	17	10 X cannot because $10 \notin A$



- Record with FK value as NULL is not in the relationship



{ 1:M + 0 relationship }

- Each record of referencing relation (foreign key) can relate almost one record of referred relⁿ.
 - 0 for NULL
 - 1 for rest
- Each record referred can be related by 0 or more (many) records of referencing relation.

Referential Integrity constraints

4/11/17

Sid	Sname	age
S1		
S2		
S3		
S4		
S5		

Sid	Cid	fee
S1	C1	
S1	C2	
S2	C2	
S5	C1	
S4	C2	

Referenced Relation

Table which contain foreign key
Referencing Relation

foreign key value
must always
be in
corresponding
primary key
(i.e. referred
rel's PK)

→ Foreign key Constraints —

① Referenced Relation [Student]

- Insertion — No violation
- Deletion — May cause violation

↓ solve for this

① ON DELETE NO ACTION (default)

deletion of referenced record restricted
if foreign key violation occurs.

② ON DELETE CASCADE

Because of deletion of referenced
record, DBMS deletes the related
referencing record.

③ ON DELETE SET NULL

Allowed to delete referenced record if
related f k field values are set to
NULL (i.e not a primary key)

Now → this is deleted - 5

R(A, B)	S(C, D)
2	2 1
3	2 12
5	6 13
	7 14

not a primary key

- Updation — may cause violation

↓
soln

- ① ON UPDATE NO ACTION (default)
- ② ON UPDATE CASCADE
- ③ ON UPDATE SET NULL

② Referring Relation [enroll]

- Insertion — may cause violation
- Deletion — NO violation
- Updation — may cause violation

Eg—

create table Enroll

{ Sid

Cid

for

Primary ()

foreign key (Sid) references student

ON DELETE CASCADE

ON UPDATE CASCADE

);

→ Check constraint :

(constraint to attribute values)

It is used to specify set of accepted values of attribute (or) range of value to given attribute.

Eg — create table stud

' sid varchar(10) Primary key ,

name varchar(30),

age integer check age ≥ 20 and age ≤ 30 ,

NULL is also
not allowed.

games varchar(10) check ("cricket", "chess", NULL)

);

Here,
NULL is
allowed.

Ques: Assume a relation R(AB C) where AB combined is a primary key and another relation S(DE F) where D is primary key. Attribute 'E' of rel 'S' ; foreign key reference to Rel R is allowed or not

R (A B C)

4 5

4 6

5 4

5 6

S (DE F)

X Not allowed to
refer

2 fields are
combined.

i.e. PK.

so the referring must also be
2 fields

2) check constraint can be replaced by foreign key . T or F ?

Not Allowed , false as check constraint is for range
and FK is for values that belong to other table

Normalization

- It is used to eliminate or reduce redundancy in DB tables, or RDBMS table.
 - Redundancy in DB table is present if two or more independent relations are stored in single RDBMS table.

Above are the 3 independent
relations

↓ when these are maintained
in a single table.

R

Sid	Sname	age	Cid	Cname	Instr	fee
S1	B	20	C1	DB	Karthik	5000
S2	B	20	C1	DB	Karthik	4000
S3	A	22	C1	DB	Karthik	4000
S3	A	22	C2	Algo	Coaxman	6000
S3	A	22	C3	Algo	Allen	5000
S4	B	23	C4	CO	Hayes	

→ if S3 → to delete record → doing so, we need to delete course C1, by doing so, we delete student also (drawback)

redundancy
as some student enrolled with different courses.

(②) if we need to insert a course, we need to add student info also (insertion problem)

(④) if we change the age of S3 in one place
↓
May cause inconsistency

Problems because of redundancy

Because of redundancy, it may cause inconsistency

Database Anomalies

① updation Anomaly -

If one redundant copy is updated and some other redundant copy is not updated, it causes inconsistency

② deletion Anomaly -

Because of deletion of some data, we may lose some other important data

③ insertion Anomaly -

It is not possible to insert one independent data without other data (independent data)

If redundancy is reduced / eliminated,

then DB anomalies are also reduced
or eliminated.

↓ soln.

{ Requirement
of Normalization

→ Decompose the DB relation into two or more subrelations to reduce / eliminate redundancy.

Eg -

$R_1 (\underline{S_id} \text{ Sname age})$

S₁

S₂

S₃

S₄

can be deleted
so all data
will be
in R₂ in
deletion

can be
inserted
without
forgetting
course.

$R_2 (\underline{S_id} \text{ } \underline{C_id} \text{ } f\&e)$

S₁ C₁

S₂ C₁

S₃ C₁

S₃ C₂

S₃ C₃

$R_3 (\underline{C_id} \text{ Cname Inst})$

C₁

C₂

C₃

C₄

Normalized DB design → : No redundancy
: No Anomalies

can be added
without adding
stirkey info

Functional Dependency (FD) —

(also called Single Valued Dependency)

- It is represented by ' \rightarrow ' mark.

Eg -

$$X \rightarrow Y$$

(where X and Y are 2 different sets of attributes)

- $X \rightarrow Y$ implied in R

only if $t_1 \cdot X = t_2 \cdot X$ then $t_1 \cdot Y = t_2 \cdot Y$

(i.e. for each X' value in relation R, there must be only one Y value)

Eg -

R

	A	B	C
$t_1:$	4	8	2
	6	5	3
$t_2:$	4	8	4
	6	5	6
$t_3:$	4	8	5
	7	5	8
	9	6	7

Whenever A is same
B must also be same
 $\therefore A \rightarrow B$

Whenever A is not same
B may be same or different.

C be the candidate key (i.e. all the records are unique) $\rightarrow C \rightarrow A$ ✓ $C \rightarrow B$ ✓

$$A \rightarrow B \quad \checkmark$$

$$B \rightarrow A \quad \times$$

(because for same B, different A)

$$\begin{cases} C \rightarrow A \\ C \rightarrow B \end{cases}$$

candidate key
(or)
Superkey

Always ✓
if (C is candidate key or Superkey).

$$X \rightarrow Y$$



determinant



Determiner

① Trivial FD -

$X \rightarrow Y$ is trivial FD if and only if $X \supseteq Y$
(self dependency)

Eg-

$$A \rightarrow A$$

$$B \rightarrow B$$

$$C \rightarrow C$$

$$AB \rightarrow A$$

$$(AB) \rightarrow (AB)$$

$$X \supseteq Y$$

[every possible trivial FD always exist in relation R]

$$R(ABC) \quad \{ AB \rightarrow A \}$$

$$\begin{bmatrix} 45 \\ 45 \end{bmatrix}^6 \\ \begin{bmatrix} 45 \\ 45 \end{bmatrix}^7$$

X should contain all elements from Y sides

② Non Trivial FD -

No common attribute over determinant and determiner i.e.

$$X \cap Y = \emptyset, \text{ thus } X \rightarrow Y \text{ is non trivial FD}$$

Eg -

$$A \rightarrow B$$

$$C \rightarrow A$$

$$C \rightarrow B$$

$$AC \rightarrow B$$

③ Semi - Non Trivial FD -

combination of trivial and non trivial

Eg -

$$AC \rightarrow BC$$

can be split as :

$$AC \rightarrow B \quad (\text{non trivial})$$

$$AC \rightarrow C \quad (\text{trivial})$$

$$A \rightarrow AB$$

$$A \rightarrow A \quad (\text{trivial})$$

$$A \rightarrow B \quad (\text{non trivial})$$

$BC \rightarrow AC$

$BC \rightarrow A$ (non trivial)

$BC \rightarrow C$ (trivial)

Ans:

R	A	B	C
2	4	6	
3	4	7	
2	4	5	
5	4	7	
6	5	7	

Find set of Non trivial FD's?

$A \rightarrow B$ ✓

$B \rightarrow A$ X

$A \rightarrow BC$ X

$BC \rightarrow AX$

$B \rightarrow C$ X

$C \rightarrow B$ X

$B \rightarrow AC$ X

$AC \rightarrow B$ ✓

$C \rightarrow A$ X

$A \rightarrow C$ X

$C \rightarrow AB$ X

$AB \rightarrow C$ X

{ $A \rightarrow B$, $AC \rightarrow B$ } Ans

if $A \rightarrow B$
and
 $A \rightarrow C$
true

$\iff A \rightarrow BC$

Ans: R (A B C)

4 4 7

4 7 7

4 4 5

4 7 5

6 4 7

Find set of non trivial FD's

for given instance

$A \rightarrow B$ X

$B \rightarrow A$ X

$A \rightarrow BC$ X

$BC \rightarrow A$ X

$B \rightarrow C$ X

$C \rightarrow B$ X

$B \rightarrow CA$ X

$CA \rightarrow B$ X

$C \rightarrow A$ ✗

$A \rightarrow C$ X

$C \rightarrow AB$ X

$AB \rightarrow C$ X

{ 4
Ans

→ Stud (Sid Sname age)

S₁ A

S₂ B

S₃ C

S₄ C → Even though

all Sname are same

still we can't say

Sname → Sid

X

as in future

S₄ with some new C can come

MEid : emp who manages dept

e₁ d₁

e₁ d₂

e₂ d₃

e₃ d₄

"each dept must have only one manager"

"each emp can manage many dept" (constraints)

{ did → MEid }

↳ dependences are defined based on given constraint

If all MEid are unique, then also we can't say MEid → did as in future same employel can manage any other dept.

→ R (A B C)

4 5 1

{ B → A not valid implied }

True

4 5 2

{ A → B implied in R }

we cannot say

6 7 3

(because based on instance dependency cannot be derived)

7 7 4

∴ May or may not

as in future
(same we could add
after B)

\therefore Functional dependencies of relation R may or may not be defined over given data.

Armstrong Rules over FD's —

x, y, z are some set of attributes over the relation R.

① Reflexivity —

$x \rightarrow x$ always holds true
(as trivial FD is always valid)

② Transitivity —

If $x \rightarrow y$ and $y \rightarrow z$ implied in relation R, then
 $x \rightarrow z$ also holds true in relation R.

Eg -

R	(A	B	C	D)	if $\{B \rightarrow C, C \rightarrow D\}$ exist in R
	4	5	8		
	4	5	8		then, $B \rightarrow D$ also
	5	7	9		exist in R
	5	7	9		
	6	7	9		
	6	7	9		

③ Augmentation —

If $x \rightarrow y$ is true, then $xz \rightarrow yz$ will be true.

Eg - R (A B C D)

4	4	5	8
5	4	5	8
5	5	7	9
6	5	7	9
7	6	7	9
8	6	7	9

if $B \rightarrow C$

then

$AB \rightarrow AC$ is true

whatever be the A value

but reverse is false

if $AB \rightarrow AC$, then $B \rightarrow C \times$ false.

④ Split and Merge Rule -

- If $X \rightarrow Y \cup Z$, then $X \rightarrow Y$, $X \rightarrow Z$
(split rule)

but however i.e. $X \rightarrow Y \cup Z$ exist in R,

then $\begin{matrix} X \rightarrow Y \\ \text{or} \\ X \rightarrow Z \end{matrix}$ May not exist in R
 ↓
 they exist together.

- If $X \rightarrow Y$, $X \rightarrow Z$, then $X \rightarrow Y \cup Z$

(Merge Rule)

⑤ Attribute closure (X^+)

$X^+ =$ set of all possible attributes which are determined by X recursively.

Ques: Given FD set

$$\{ A \rightarrow B, BC \rightarrow D, CD \rightarrow E, B \rightarrow C, EF \rightarrow G \}$$

Find A^+ ?

$$A^+ = \{ A, B, C, D, E \}$$

Ans

$$A \rightarrow ABCDE$$

$A \rightarrow A$ reflexive

$$A \rightarrow B$$

$$B \rightarrow C$$

$$BC \rightarrow D$$

$$CD \rightarrow E$$

A

A, B

A, B, C

A, B, C, D

A, B, C, D, E

$$B^+ = \{ B, C, D, E \}$$

Ans

$$B \rightarrow BCDE$$

$$(AF)^+ = \{ A, F, B, C, D, E, G \}$$

Ans

$$AF \rightarrow ABCDEF G$$

Superkey —

(It can be identified by FD's)

Relation R with FD set (F),

R (X Y Z)

2
3
4
5
6

X: superkey of R

then $\{X \rightarrowYZ\}$

X is superkey of relation R iff X^+ determines all attributes of relation R

Eg -

R (A B C D)

2 4 given $\{A \rightarrow B, B \rightarrow CD\}$
3 5

Not allowed X 7 6 7 8 allowed ✓ 8 6 7 8 $B^+ = \{B, C, D\}$

(A must be different)

Since, it does not determine all the attributes, it does not determine A

∴ Not superkey

$A^+ = \{A, B, C, D\}$

A is superkey as it determines all

for superkey



All the values are unique

(different)

Candidate key -

(Minimal superkey)

X is candidate key of rel R iff

① X must be superkey of relation R

(i.e X^+ should determine all attributes of R)

and ② No proper subset of X is superkey

($\nexists Y \subset X$ such that

Y^+ should not determine all attributes)

Ex-

for relation $R(A B C D E)$ with dependencies

$$\{AB \rightarrow C, B \rightarrow D, A \rightarrow E\}$$

$$(AB)^+ = \{A, B, C, D, E\}$$



AB is superkey

Now, proper subset of $AB \rightarrow \{A, B\}$

not
superkey

$$\left. \begin{array}{l} A^+ = \{A, E\} \\ B^+ = \{B, D\} \end{array} \right\}$$

$\therefore AB$ is candidate key.

* If one attribute is superkey i.e A is superkey, then it itself will be candidate key.

Ques: Find no. of candidate keys of given relation R?

① R (A B C D) $\rightarrow \{ A \rightarrow B, B \rightarrow C \}$

$$A^+ = \{ A, B, C \}$$

D is not determined

D cannot be determined by any

$$\therefore (AD)^+ = \{ A, B, C, D \}$$

superkey

Also,

$$A^+ = \{ A, B, C \}$$

$$D^+ = \{ D \} \quad \text{NOT super key}$$

$\therefore (AD)$ is candidate key.

determinant

determiner

A
B

B
C

only in determinant but not in determiner

(so it will be a part of candidate key)

Note - ① X must be a part of every candidate key of R, only if -

① X is not present in non trivial FD

set of relation R

(Here 0)

(08)

② X belongs to some determinant but does not belong to determiner of non trivial FD

(Here A)

③ if $X \rightarrow Y$ is non trivial FD with 'Y' as prime attribute of relation R, then R has at least 2 candidate key

$$X \rightarrow Y$$

prime attribute (i.e. Y belongs to one of candidate key)

$X \rightarrow Y$
 \downarrow
 Prime
 attribute
 $(WY)^+ = \{ \text{All attributes of } R \}$
 if $X \rightarrow (Y)$
 \Downarrow (we can replace, Y with X)
 one more possible candidate key exist
 $(WA)^+ = \{ W, X, Y, \text{ all } ? \}$
x determinates
and
with
w, y all

Here, in this ques,

prime attribute is $(AD)^+$

A or D is not present on RHS

of any FD

\therefore only 1 candidate key.

② $R(ABCDE)$

$\{ AB \rightarrow C, C \rightarrow D, D \rightarrow EA \}$

B is not on RHS of any FD

$\therefore B$ will be in all candidate key.

$\xrightarrow{\text{super key}} (AB)^+ = \{ A, B, C, D, E \}$

Now Testing for minimal superkey.

a) Find proper subset of superkey closure and it should not determine all attribute

$$A^+ = \{ A \} \times$$

$$B^+ = \{ B \} \times$$

AB is minimal. \Rightarrow candidate key.

b) some other attributes of superkey should not determine other attributes of superkey.

$$AB \rightarrow B \times \text{i.e } A^+ = \{ A \}$$

$$B \rightarrow A \times \text{i.e } B^+ = \{ B \}$$

AB is minimal.

AB



More candidate keys are

{BD, BC}

∴ {AB, BD, BC}

3 candidate keys in R.

BD

We need to check
each for minimal

$B^+ = \{B\}$

$D^+ = \{D, E, A\}$

Here it is minimal

↓

D is on RHS

$C \rightarrow D$

∴ BC

③ R(ABCD)

{AB → CD, C → A, D → B}

$AB^+ = \{A, B, C, D\}$.

check for minimal.

$A^+ = \{A\}$

$B^+ = \{B\}$

A^+ should
not contain
B and

B^+ should
not contain A

∴ AB



CB

= {C, B, A, D}



CD

= {C, D, A, B}

AD

= {A, B, C, D}

∴ AB, CB, CD

$B^+ = \{B\}$

$C^+ = \{C, A\}$

$C^+ = \{C, A\}$

$D^+ = \{D, B\}$ ✓

(4)

④ R(ABCDE)

{A → BC, CD → E, E → A, B → D}

$A^+ = \{A, B, C, D, E\}$

$E^+ = \{A, B, C, D, E\}$

$CD = \{A, B, C, D, E\}$

$CB = \{C, B, D, E, A\}$

$AD = \{A, B, C, D, E\}$

$ED = \{A, B, C, D, E\}$

(4)

⑤ $R(ABCDEF)$

$$\{ AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow B \}$$

A will be part of every

$$AB = \{ A, B, C, D, E, F \}$$

AB

$$AF = \{ A, F, B, C, D, E \}$$

AF

$$AE = \{ A, B, C, D, E, F \}$$

AE

$$AC = \{ A, C, D, E, F, B \}$$

(4)

⑥ $R(ABCDEF)$

$$\{ AB \rightarrow C, C \rightarrow D, CD \rightarrow E, DE \rightarrow F, EF \rightarrow B \}$$

A will be part of all

✓ $AB = \{ A, B, C, D, E, F \}$

$$\begin{array}{l} A^+ = A \\ B^+ = B \end{array} \checkmark$$

✓ $AEF = \{ A, E, F, B, C, D \}$

$$\begin{array}{l} A^+ = \{ A \}, E^+ = \{ E \}, F^+ = \{ F \} \\ (AE)^+ = \{ A, E \}, (EF)^+ = \{ E, F, B \} \\ AF^+ = \{ A, F \} \end{array}$$

✓ $AED = \{ A, E, D, F, B, C \}$

$$AD^+ = \{ A, D \}, DE = \{ D, E, F, B \}, AE = \{ A, E \}$$

X $ACD = \{ A, C, D, E, F, B \}$

$$AC = \{ A, C, \underline{D}, E, F, B \}$$

containing D

✓ $AC = \{ A, C, D, E, F, B \}$

$$A = \{ A \}$$

~~$$AC = \{ A, C \}$$~~

$$C = \{ C, D, E, F, B \}$$

(4)

7) $R(ABC)$

No non-trivial FD in R .

ABC is candidate key.

Note — if no non-trivial FD, then candidate key is all the attribute collectively.

Membership test —

Given FD set (F),

$X \rightarrow Y$ FD is logically implied (member) in FD set iff
 X^+ must determine ' Y ' in FD set (F).

$$\{ \dots \dots \dots \} \Leftarrow X \rightarrow Y$$

if X^+ consist Y , then

$X \rightarrow Y$ is member of FD set

otherwise $X \rightarrow Y$ not member.

Given: $\{ AB \rightarrow C, BC \rightarrow D, D \rightarrow E, DE \rightarrow F \}$

a) Test $AB \rightarrow F$ member or not.

$$(AB)^+ = \{ A, B, C, D, E, \underline{F} \}$$

contain F

\therefore Member

b) Test $BC \rightarrow A$ member or not.

$$(BC)^+ = \{ B, C, D, E, F \}$$

Not Member

as A is not in $(BC)^+$

Closure of FD set ((NP complete problem / takes exponential time))

Given FD set F

F^+ : set of all possible FD's which can be derived by using given FD's

$$\text{Eg} - F = \{ A \rightarrow B, B \rightarrow C \}$$

$A^+ = ABC$	$B^+ = BC$	$C^+ = C$	$(AB)^+ = ABC$	$(BC)^+ = BC$	$(AC)^+ = ABC$	$(ABC)^+ = ABC$
$A \rightarrow A$	$B \rightarrow B$	$C \rightarrow C$	$AB \rightarrow A$	$BC \rightarrow B$	$AC \rightarrow A$	$ABC \rightarrow A$
$A \rightarrow B$	$B \rightarrow C$		$AB \rightarrow B$	$BC \rightarrow C$		
$A \rightarrow C$	$B \rightarrow BC$			$BC \rightarrow BC$		
$A \rightarrow AB$						
\vdots						
$A \rightarrow ABC$			$AB \rightarrow ABC$			
					$AC \rightarrow ABC$	$ABC \rightarrow ABC$

Note - ① FD set closure identification

→ NPC problem

(Exponential TC Problem)

$$\textcircled{2} \quad F \equiv F^+$$

↓
equal expressive power.

Equality of FD set —

Given FD set F and G

1)

F and G have equal expressive iff
 $f^+ \equiv g^+$

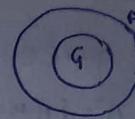
→ Theoretically,
and
is complex
method.

2) f and g are equal expressive iff -

a) F covers G :

Every FD of G set must be a member of F set i.e.

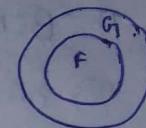
$$F \supseteq G$$



and b) G covers F :

Every FD of F set must be member of G set i.e.

$$F \subseteq G$$



F covers G	G covers F	
True	True	$F \equiv G$
True	False	$F \supsetneq G$
False	True	$F \subset G$
False	False	F and G are not comparable i.e. $\emptyset \subset G$ or $F \subset \emptyset$

Ans: $f = \{ A \rightarrow B, B \rightarrow C, C \rightarrow A \}$

$g = \{ A \rightarrow BC, B \rightarrow AC, BC \rightarrow A, AB \rightarrow C \}$

Which is true -

- a) $F \subset G$
- b) $F \supsetneq G$
- c) $F \equiv G$
- d) None of them

F covers G (every G member of F)

$A^+ \rightarrow A, B, C$
Take each member of G and check if present in F and

from G $A \rightarrow BC$ $B \rightarrow AC$ $BC \rightarrow A$ $AB \rightarrow C$
in F $A^+ = \{ A, B, C \}$ $B^+ = \{ B, C, A \}$ $(BC)^+ = \{ B, C, A \}$ $(AB)^+ = \{ A, B, C \}$
 \therefore True.

G covers F

from F $A \rightarrow B$ $B \rightarrow C$ $C \rightarrow A$
in G $A^+ = \{ A, B, C \}$ $B^+ = \{ B, A, C, A \}$ $C^+ = \{ C \}$
 \therefore False

Ques: $F = \{ A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow F, D \rightarrow E \}$

$G_1 = \{ A \rightarrow BC, BC \rightarrow AD, B \rightarrow F, D \rightarrow E \}$

F covers G_1 —

From G_1 $A \rightarrow BC$ $BC \rightarrow AD$ $B \rightarrow F$ $D \rightarrow E$
in F $A^+ = \{ A \underline{BCDEF} \}$ $(BC)^+ = \{ BC \underline{ADEF} \}$ $B^+ = \{ B \underline{F} \}$ $D^+ = \{ D, E \}$

G_1 covers F —

$A \rightarrow BCDEF$ $BC \rightarrow ADEF$ $B \rightarrow F$ $D \rightarrow F$
 $A^+ = \{ A \underline{BCDEF} \}$ ✓ ✓ ✓

$\therefore F \cong G_1$

Minimal cover (or) Canonical cover —

Given FD set F ,

Minimal cover of FD set (F):

Minimal possible FD's which are logically equal to F .

Procedure to find minimal cover:

Given FD set (F)

- ① Remove extraneous attributes from every determinant of FD set (F)

If $wxy \rightarrow z$, $w^+ = x$; then

attr 'x' is extraneous in

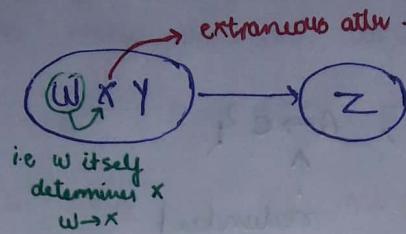
$wxy \rightarrow z$

$: f_1$

② Remove all redundant FD's from step ① result

$X \rightarrow Y$ redundant in FD set (F_1) in above
 iff $X \rightarrow Y$ is member of
 $\{F_1 - \{X \rightarrow Y\}\}$ in FD set

Note — Extraneous attribute over determinant —



$\{W \dot{\times} Y \rightarrow Z, W \rightarrow X\} \equiv \{WY \rightarrow Z, W \rightarrow X\}$
 ↓
 we remove this
 extr.

Ques: ① $\{ABC \dot{\times} D \rightarrow E, AB \rightarrow F, F \rightarrow D\}$

$$AB^+ = \{A, B, F, D\}$$

extra

$\{ABC \rightarrow E, AB \rightarrow F, F \rightarrow D\}$

② $\{ABC \dot{\times} D \rightarrow E, AC \rightarrow F, F \rightarrow B, C \rightarrow D\}$

$$(AC)^+ = \{ACFBD\}$$

$\therefore \{AC \rightarrow E, AC \rightarrow F, F \rightarrow B, C \rightarrow D\}$

③ $\{AB \rightarrow C, A \rightarrow B, B \rightarrow A\}$

$$A^+ = \{A, B\}$$

Here B extraneous (or)

$$B^+ = \{B, A\}$$

A is extraneous

$$\{A \rightarrow C, A \rightarrow B, B \rightarrow A\}$$

$$\{B \rightarrow C, A \rightarrow B, B \rightarrow A\}$$

Either A or B is extraneous but not both.

Redundant FD in FD set —

$X \rightarrow Y$ in FD set (F) is redundant iff

$X \rightarrow Y$ must implied (member) in FD set $\{F - (X \rightarrow Y)\}$

$$\{F - (X \rightarrow Y)\}$$

After this also

$X \rightarrow Y$ is member of F

$\therefore X \rightarrow Y$ is redundant in FD set (F)

Ques: Find redundant —

① $F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$

↑
redundant

② $F = \{AB \rightarrow C, BC \rightarrow D, AB \rightarrow D, CD \rightarrow E, BC \rightarrow E\}$

hold this
assume it is
not there now (implication)
find $(AB)^+$ if
it gives C, then
redundant else not

↑
redundant

Ques: $F = \{A \rightarrow BCDEF, BC \rightarrow ADEF, AC \rightarrow D, D \rightarrow E, B \rightarrow F\}$

Find Minimal cover.

↓
extraneous

Remove extraneous attr. —

$$F = \{A \rightarrow BCDEF, BC \rightarrow ADEF, A \rightarrow D, D \rightarrow E, B \rightarrow F\}$$

Find redundant —

Here split the FD

$$A \rightarrow B \checkmark$$

$$A \rightarrow C \checkmark$$

$$A \rightarrow D \times$$

$$A \rightarrow E \times$$

$$A \rightarrow F \times$$

$$BC \rightarrow A \checkmark$$

$$BC \rightarrow D \checkmark$$

$$BC \rightarrow E \times$$

$$BC \rightarrow F \times$$

$$A \rightarrow D$$

$$D \rightarrow E$$

$$B \rightarrow F$$

$$only B \rightarrow F$$

$$\therefore BC \rightarrow F$$

$$(B^+)^* = B, C, A, D, E$$

$$(BC)^* = BC, A, D, E, F$$

$$\therefore BC \rightarrow F$$

\therefore Minimal cover is

$$F = \{ A \rightarrow BC, BC \rightarrow AD, D \rightarrow E, B \rightarrow F \}$$

Ans.

Here we have 2 choices

$$A \rightarrow D : BC \rightarrow D$$

X removed ✓

$$A \rightarrow D : BC \rightarrow D$$

✓ retained is redundant

\therefore one more minimal cover exist.

$$F = \{ A \rightarrow BCD, BC \rightarrow A, D \rightarrow E, B \rightarrow F \}$$

- Note - ① Minimal cover of FD set (F) may not be unique but, all minimal covers are logically equal.
- ② Also, minimal cover of is equal to FD set (F)

$$F_{m_1} \equiv F_{m_2} \equiv F$$

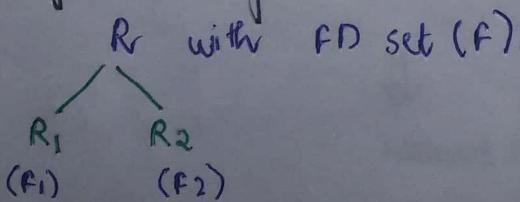
expressive power remains same.

Properties of Decomposition —

Decomposition of Relation (R) into sub-relations not causes any integrity violation if.

- it is loss less join decomposition (and)
- dependency preserving decomposition.

Dependency Preserving decomposition -

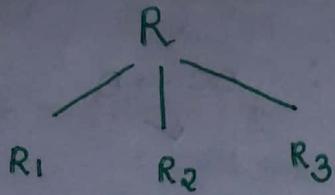


$$F_1 \cup F_2 \equiv F$$

Because of the fact that decomposition should not lost any FD of FD set

Loss less Join decomposition —

- JOIN result of subrelations must be equal to the original relation.



$$R_1 \bowtie R_2 \bowtie R_3 \sqsubseteq R$$

- Relation (R) is decomposed into $R_1 R_2 R_3 \dots R_n$ sub relations.

In general,

$$[R_1 \bowtie R_2 \bowtie \dots \bowtie R_n] \sqsubseteq R$$

- If $(R_1 \bowtie R_2 \bowtie \dots \bowtie R_n) \equiv R$, then it is loss less join decomposition.

- If $(R_1 \bowtie R_2 \bowtie \dots \bowtie R_n) \supset R$, it is called as lossy join decomposition.



Here, some extra tuples are generated because of wrong decomposition, these are called spurious tuples.

R

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S2	B	C2
S3	B	C3

{ Sid → Sname }

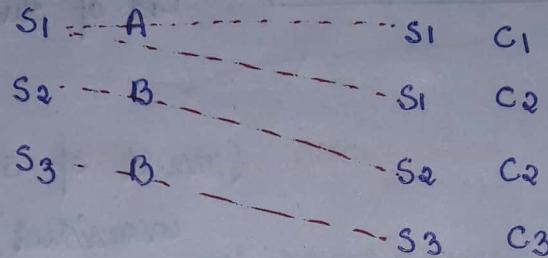
candidate key : Sid Cid

Cid of student S2 = C2

① decomposed into

R₁ (Sid Sname)

R₂ (Sid Cid)



↓ Join

Sid	Sname	Cid
S1	A	C1
S1	A	C2
S2	B	C2
S3	B	C3

Cid of S2 = C2

R₁ ⋈ R₂ ≅ R

Loss less decomposition

② decomposed into

R₁ (Sid Sname)

R₂ (Sname Cid)



↓
Natural Join R₁ ⋈ R₂

$R_1 \bowtie R_2$

Sid	sname	Cid
S1	A	C1
S1	A	C2
S2	B	C3
S2	B	C3
S3	B	C2
S3	B	C3

extra
tuple
(spurious
tuple)

$$(R_1 \bowtie R_2) \supseteq R$$



Lossy Join decomposition
(consistency is lost)

$$\text{Cid of student } S_2 = C_3, C_2$$

X
wrong op

(result of SQL query is
inconsistent)

Ans: $R(A B C) \setminus A \rightarrow B, B \rightarrow C$

4	5	4
5	5	4
6	7	5
7	7	5
8	6	5

candidate key

A

① decomposed into $R_1(\underline{A} \ \underline{B})$ $R_2(\underline{B} \ C)$

$A \rightarrow B$

$B \rightarrow C$

4	5	—	—	5	4
5	5	—	—	7	5
6	7	—	—	6	5
7	7	—	—	8	—
8	6	—	—	—	—

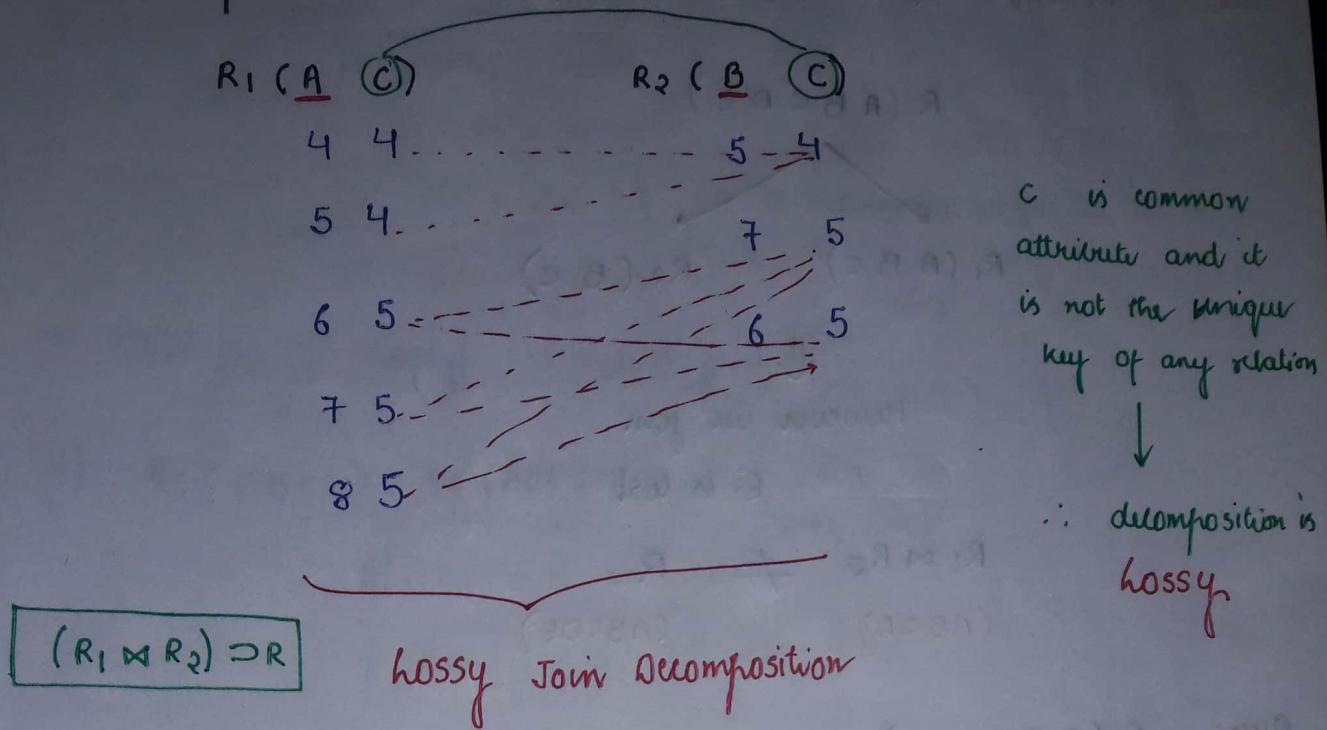
$B \rightarrow C$
candidate key
of R_2 B

$$R_1 \bowtie R_2 \equiv R$$

Total record generated = 6

Lossless join decomposition.

② decomposed into



C is common attribute and it is not the unique key of any relation



∴ decomposition is lossy

Total record generated = 8

Note-

Relation R with FD set (F) decomposed into R_1, R_2 subrelation

→ given decomposition is lossless join decomposition iff.

$$① (R_1 \cup R_2) \equiv R \quad // \text{every attr of } R \text{ is in some sub rel.}$$

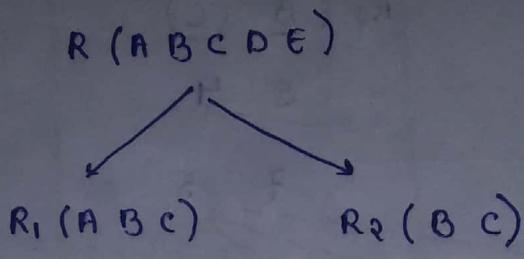
and ② $(R_1 \cap R_2) \rightarrow R_1 \quad // R_1 \cap R_2 \text{ is superkey for } R_1$
(or)

$$(R_1 \cap R_2) \rightarrow R_2 \quad // R_1 \cap R_2 \text{ is superkey for } R_2$$

because of
2nd condition
no spurious record

$$\left\{ (R_1 \bowtie R_2) \equiv R \right\} \Leftrightarrow \left\{ (R_1 \cup R_2 = R) \text{ and } (R_1 \cap R_2 \rightarrow R_1 \text{ or } R_1 \cap R_2 \rightarrow R_2) \right\}$$

$$\rightarrow R_1 \cup R_2 = R \quad (\text{it ensures that no column is lost})$$



↓
However we join
E is lost.

$$R_1 \bowtie R_2 \neq R$$

$$(A B C D) \qquad (A B C D E)$$

Ques: $R(A B C D E)$

$$\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$$

Test whether given decompositions are lossless or lossy.

i) $R_1(A B C) \quad R_2(C D)$

lossy . $R_1 \cup R_2 \neq R$ \times E is lost

ii) $R_1(A B C) \quad R_2(D E)$

lossy . $R_1 \cup R_2 = R$ \checkmark No common attribute
 $R_1 \cap R_2 = \emptyset$ \times

iii) $R_1(A B C) \quad R_2(C D E)$

lossy

$$R_1 \cup R_2 = R \quad (A B C D E)$$

$$R_1 \cap R_2 = C \quad \times$$

$$C^+ = \{C, D\}$$

↑
Not super key of R_1 or R_2

iv) $R_1(ABCD)$ $R_2(BE)$

$$R_1 \cup R_2 = R \checkmark$$

$$R_1 \cap R_2 = B$$

$$B^+ = \{B, E\} \text{ ie it determines all for } R_2$$

↓
super key (R_2) \checkmark

lossy. loss less join

v) $R_1(ABCD)$ $R_2(ABE)$

$$R_1 \cup R_2 = R \checkmark$$

$$R_1 \cap R_2 = \{A, B\}$$

$$(AB)^+ = ABCDE$$

determines both R_1 and R_2

∴ super key in both relation

∴ lossless join

5/11/17

ques:

$R(ABCDEF)$

$$\{ AB \rightarrow C, BC \rightarrow A, AC \rightarrow B, B \rightarrow D, DE \rightarrow F, F \rightarrow G \}$$

i) $D_1 \{ABC, DEF, FG\}$

$R_1(ABC)$ $R_2(DEF)$ $R_3(FG)$

$$R_1 \cup R_2 \cup R_3 = R \checkmark$$

Join relations only if common attribute key of any subrelation
 $\Rightarrow (L L J)$
lossless join

$R_1(ABC)$ $R_2(DEF)$ $R_3(FG)$

$R_1(ABC)$

$R_2(DEF)$

$$R_2 \cap R_3 = F$$

$F^+ = \{FG\}$ ∴ F key for R_3 join

No common attribute

∴ lossy

ii) $R_2 \{ABC, BD, DEF, FG, ABDE\}$

$R_1 (\underline{ABC}) \quad R_2 (\underline{BD}) \quad R_3 (\underline{DEF}) \quad R_4 (\underline{FG}) \quad R_5 (\underline{ABDE})$

$$R_1 \cap R_2 = \{B\}$$

$$B^+ = \{B, D\}$$

$\therefore B$ is key for R_2

\therefore Join

$R_{12} (\underline{ABCD}) \quad R_3 (\underline{DEF}) \quad R_4 (\underline{FG}) \quad R_5 (\underline{ABDE})$

$$R_{12} \cap R_3 = D$$

$$D^+ = \{D\}$$

does not determine
any \therefore

Not allowed to join

$$R_3 \cap R_4 = F$$

$$F^+ = \{F, G\}$$

\therefore key for R_4

\downarrow Join

$R_{12} (\underline{ABCD}) \quad R_{34} (\underline{DEF}) \quad R_5 (\underline{ABDE})$

$$R_{12} \cap R_5 = \{ABD\}$$

$$(ABD)^+ = \{ABC\}$$

$\therefore g_1$ is key for R_{12}

\downarrow Join

$R_{125} (\underline{ABCDE}) \quad R_{34} (\underline{DEF})$

$$R_{125} \cap R_{34} = \{DE\}$$

$$(DE)^+ = \{DEF\}$$

\therefore Key for R_{34}

\downarrow Join

$R_{12345} (\underline{ABCDEFG})$

\therefore lossless Join decomposition

Dependency Preserving decomposition -

- Relational schema R with FD set F decomposed into $R_1 R_2 R_3 \dots R_n$ subrelations with $F_1 F_2 F_3 \dots F_n$ FD's.
- In general,

$$\{F_1 \cup F_2 \cup F_3 \cup \dots \cup F_n\} \leq F$$

- If $[F_1 \cup F_2 \cup \dots \cup F_n] = F$, then decomposition is dependency preserving decomposition (every FD of F is preserved in sub relation)
- If $[F_1 \cup F_2 \cup \dots \cup F_n] \subset F$, then decomposition is Not dependency preserving decomposition.



Here, because of decomposition,
some FD of F is lost.

Ques: $R(ABCDEF)$

$$\{AB \rightarrow C, BC \rightarrow A, AC \rightarrow B, B \rightarrow D, DE \rightarrow F, F \rightarrow G\}$$

$$D_2 \{ABC, BD, DEF, FG, ABDE\}$$

$R_1(ABC)$	$R_2(BD)$	$R_3(DEF)$	$R_4(FG)$	$R_5(ABDE)$
\downarrow	\downarrow	\downarrow	\downarrow	\downarrow
$\left\{ \begin{array}{l} AB \rightarrow C \\ BC \rightarrow A \\ AC \rightarrow B \end{array} \right.$	$B \rightarrow D$	$DE \rightarrow F$	$F \rightarrow G$	$B \rightarrow D$

All the dependencies are preserved in some relation.

$$\therefore [F_1 \cup F_2 \cup F_3 \cup F_4 \cup F_5] \leq F$$

∴ Dependency preserved

Ques: $R(ABCDE)$

$$\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow BE\}$$

decomposed to $\{AB, BC, CD, DE\}$

$R_1(AB)$

$$\downarrow$$

$$A \rightarrow B$$

$R_2(BC)$

$$\downarrow$$

$$B \rightarrow C$$

$R_3(CD)$

$$\downarrow$$

$$C \rightarrow D$$

$R_4(DE)$

$$\downarrow$$

$$D \rightarrow E$$

$$B^+ = \{B, C, D, E\}$$

A is not present
so $B \rightarrow A$ will
not be there.

$$C^+ = \underline{CD}BE$$

$$\therefore C \rightarrow B$$

Here, $D \rightarrow B$ is not present
directly.

so, we find all hidden FD.

$$D^+ = \underline{DBEC}$$

$$D \rightarrow C$$

$$E^+ = E$$

NOW,

$$A \rightarrow B \quad \checkmark$$

$$B \rightarrow C \quad \checkmark$$

$$C \rightarrow D \quad \checkmark$$

$$D \rightarrow BE \quad \checkmark \quad \text{as } D^+ = \underline{DBEC}$$

$$\therefore F_1 \cup F_2 \cup F_3 \cup F_4 \cong f$$

dependency preserved.

Ques: $R(ABCD)$

$$\{AB \rightarrow CD, D \rightarrow B\}$$

decomposed into $\{ACD, BD, ABC\}$

$R_1(ACD)$

$$f_1$$

compute closure
of all proper subsets of (AC)

$$\begin{aligned} A^+ &= A, B^+ = B, C^+ = C \\ AB, BE &\vdash \dots \\ (AB)^+ &= ADB, C \end{aligned}$$

$$AD \rightarrow C$$

$$AB \rightarrow C \quad \checkmark$$

$R_2(BD)$

$$f_2$$

$$D \rightarrow B$$

$$B^+ = \{B\}$$

$R_3(ABC)$

$$f_3$$

$$AB \rightarrow C$$

$$C^+ = \{C\}$$

$$(AB)^+ = ABCD$$

All possible
FD's

$(AB)^+$ in subrelation

$$(AB)^+ = ABC$$

D is not here

$\therefore \text{DDST}$

$$AB \rightarrow D \quad \times$$

$$D \rightarrow B \quad \checkmark$$

\therefore Dependency Not preserved

$(AB \rightarrow D)$ is lost because
of decomposition

Ques: $R(ABCDEF)$

$\{ A \rightarrow BCDEF, BC \rightarrow \underline{ADEF}, B \rightarrow F, D \rightarrow E \}$

decomposed to $\{ ABC, BCD, BF, DE \}$

$R_1(ABC)$	$R_2(BCD)$	$R_3(BF)$	$R_4(DE)$
\downarrow	\downarrow	\downarrow	\downarrow
$A \rightarrow BC$	$BC \rightarrow D$	$B \rightarrow F$	$D \rightarrow E$
$(BC)^+ = BC \underline{ADEF}$	$D^+ = \{ D, E \}$	$F^+ = \{ F \}$	$E^+ = \{ E \}$
$BC \rightarrow A$	(F_1)	(F_2)	(F_3)
$(AB)^+ = \cancel{ABCDEF}$			(F_4)
$A \rightarrow BCDEF$			

A^+ in subrelation = ABCDEF

\therefore all are satisfied ✓

$BC \rightarrow ADEF$

$(BC)^+$ in subrel = BCDEF A

all satisfied ✓

$B \rightarrow F$ ✓
 $D \rightarrow E$ ✓

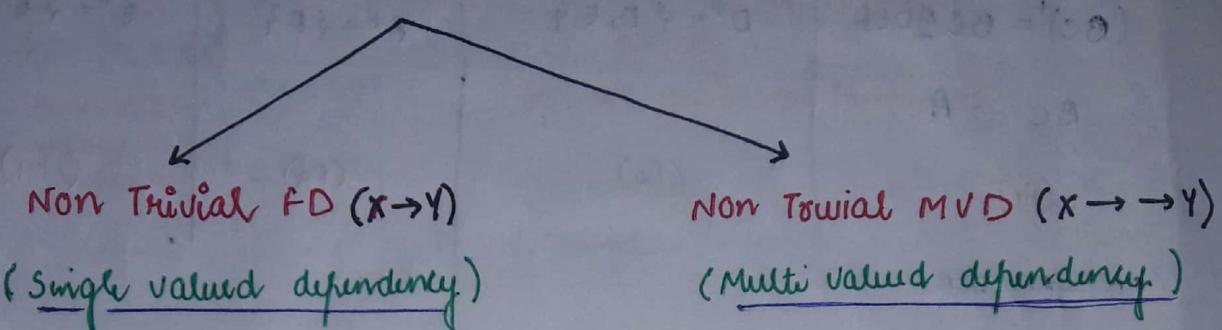
$F_1 \cup F_2 \cup F_3 \cup F_4 \rightleftharpoons F$

\therefore dependency preserved.

Normal Forms —

- Used to identify degree of redundancy (also used to eliminate/reduce redundancy)

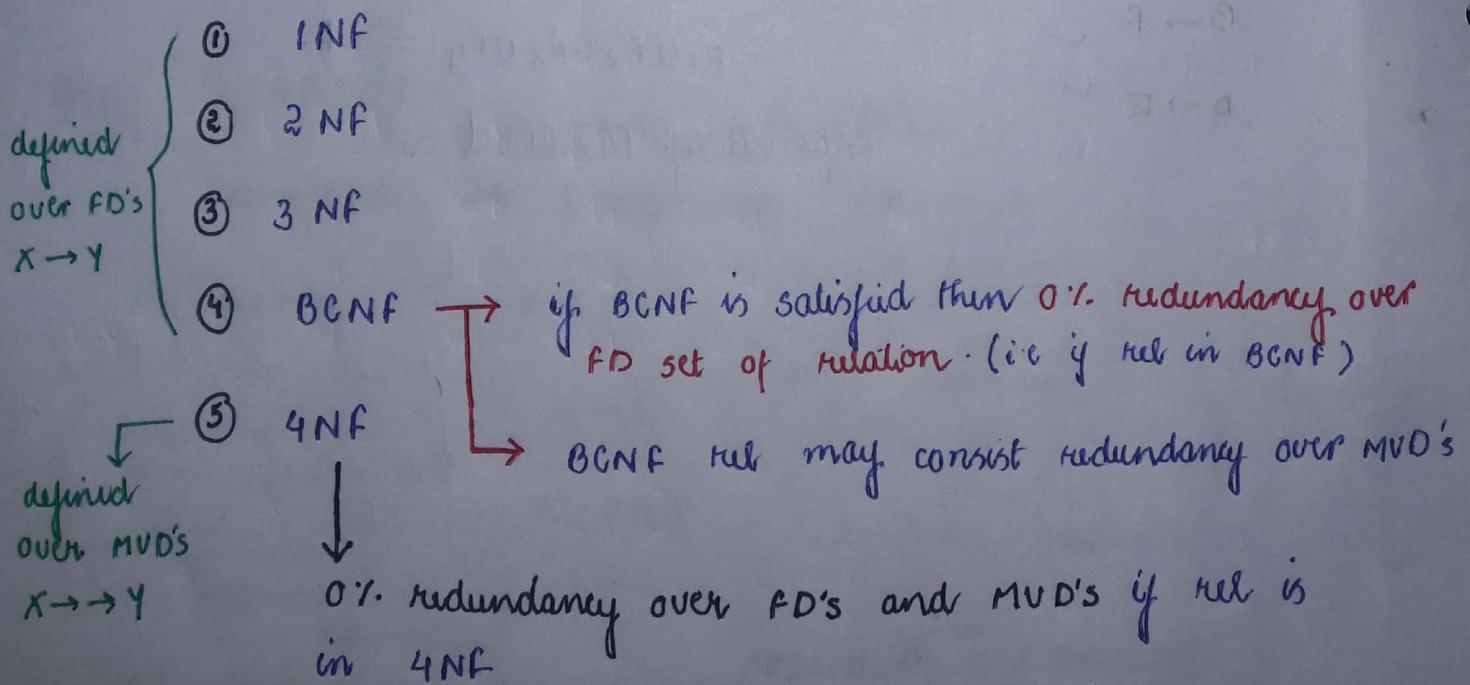
Redundancy in relation (R) can be because of



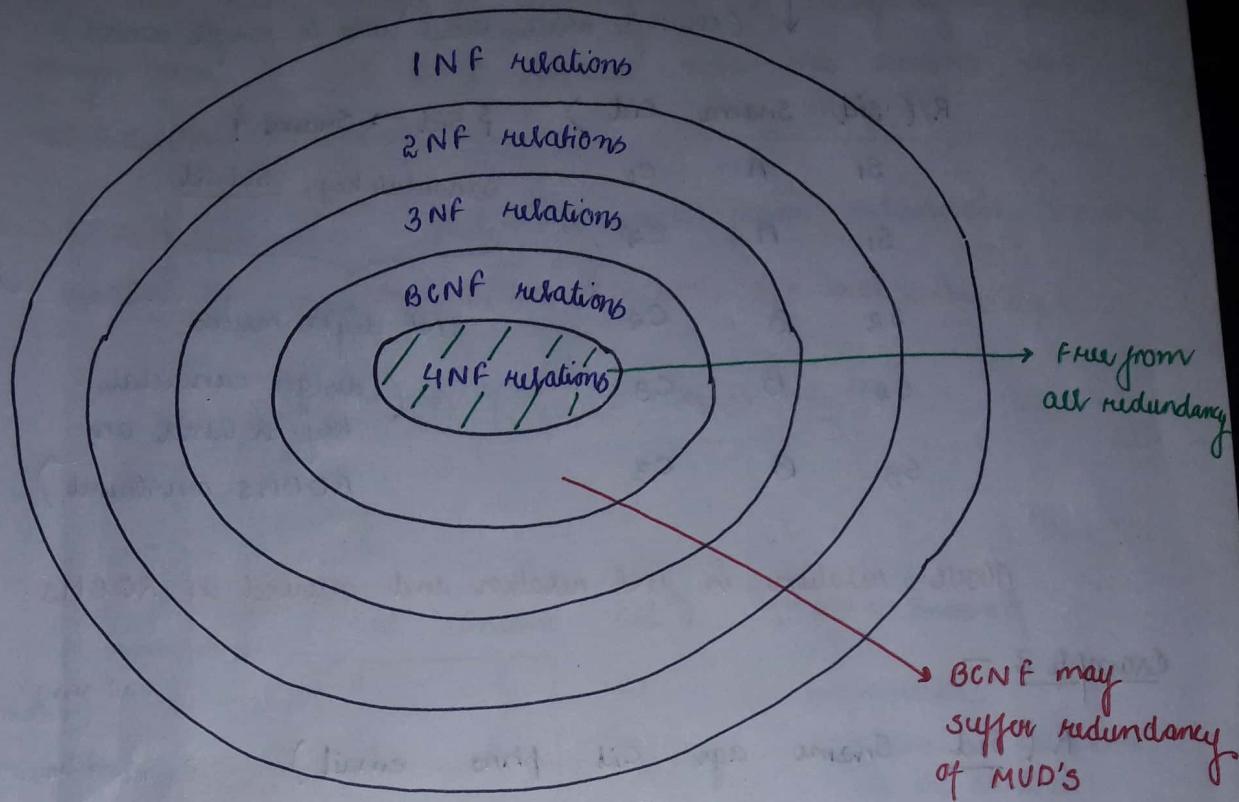
To eliminate redundancy over FD's, relation should decomposed into "BCNF"

To eliminate redundancy over MVD's, relation should decompose into "4NF"

Various Normal Forms —



If the reln is not in INF, then it does not satisfy RDBMS rules.



First Normal Form (1NF) —

- It is default NF of RDBMS relation.
- definition : Relation R in 1NF iff NO multivalued attribute in R
 - (or)
- Every attr of rel R must be atomic (single value).
- NO 2 records of relation must be same (i.e. the relation must have unique candidate key)

R (sid sname cid)	multivalued attribute (set of multiple value over attribute for any record)
S ₁ A C ₁ /C ₂	
S ₂ B C ₂ /C ₃	
S ₃ B C ₃	

{ sid → sname ? }

Not in 1NF

(1-many relation)

{ sid → cid } X

↓ converting to INF design
(change multivalued attr to single valued)

R (sid, Sname, cid) { Sid → Sname }

S₁ A C₁ candidate key: Sid Cid

S₁ A C₂

S₂ B C₂

S₂ B C₃

S₃ B C₃

INF design means

(design candidate key is based on RDBMS constraint)

Above relation is INF relation and allowed in RDBMS.

Example 2 -

R (Sid, Sname, age, cid, phno, email) { Sid → Sname, age }

S₁ A 20 C₁/C₂ P₁/P₂/P₃ E₁/E₂/E₃/E₄

S₂ B 22 C₂/C₃ P₄ E₅/E₆

all single value.

↓ converting to INF design
(RDBMS design)

R (Sid, Cid, phno, email, Sname, age) { Sid → Sname, age }

S₁ C₁ P₁ E₁ A 20

S₁ C₁ P₂ E₂ A 20

S₁ C₁ P₃ E₃ A 20

⋮ ⋮ ⋮ ⋮ ⋮ ⋮

S₁ C₂ P₁ E₁ A 20

⋮ ⋮ ⋮ ⋮ ⋮ ⋮

S₂ C₂ P₄ E₅ B 22

S₂ C₂ P₄ E₆ B 22

S₂ C₃ P₄ E₅ B 22

S₂ C₃ P₄ E₆ B 22

cond key: Sid Cid phno email

all multivalued values.

Total record
for S₁ = 2 × 3 × 4
= 24

for S₂ = 2 × 1 × 2
= 4

Note
In INF even degree of redundancy is very high because design goal of INF is to convert data into RDBMS, but not to decrease redundancy.

→ $X \rightarrow Y$ is non trivial FD which form redundancy in any relation R iff Non trivial FD with X is not superkey



Eg - R (Sid, Sname, cid) { Sid \rightarrow Sname ? }

redundancy

S ₁	A	C ₁
S ₁	A	C ₂
S ₁	A	C ₃
S ₂	B	C ₃

↑
not superkey
(as does not determine all the attribute)
∴ separated in table (eg. S₁)

→ $X \rightarrow Y$ FD in R does not form any redundancy

iff ① Trivial FD

$$X \equiv Y$$

(or)

② X must be a super key.

Eg - R (Sid, Sname, age) { Sid \rightarrow Sname, age ? }

S ₁	A	20
S ₂	A	20
S ₃	B	22
S ₄	B	20

superkey
NO Redundancy.

Eg - $R(A B C)$ $\{ A \rightarrow BC \}$
NO Redundancy

A	B	C
1	2	5
2	2	5
3	3	7
4	3	7
5	3	7

Here
No
relation
b/w
B and C.

Eg - $R(A B C)$ $\{ A \rightarrow B, (B) \rightarrow C \}$

1	2	5
2	2	5
3	3	7
4	3	7
5	3	7

↑
super
key

↓
Not a
super key

Forms redundancy
in relation R

(i.e whatever will be
B's value ^{and} C ^{value} both
be same ~~all the time~~)
in all case

Ques: Which of the dependency can form redundancy in R
 $R(ABCDEF)$ *points out* *not points out*

cand key $\rightarrow AB, BC$

$\{ AB \rightarrow C, B \rightarrow D, D \rightarrow E, AE \rightarrow F, C \rightarrow A \}$

• $AB \rightarrow C$

$$(AB)^+ = ABCDEF$$

$\therefore AB$ is a superkey. ✓

• $B \rightarrow D$

$$B^+ = BDE$$

B is not a superkey.

Redundant

• $D \rightarrow E$

$$D^+ = DE$$

not a superkey

Redundant

• $AE \rightarrow F$

$$(AE)^+ = AEF$$

not a superkey

Redundant

$$C \rightarrow A$$

$$C^+ = C, A$$

Not a superkey

Redundancy

2nd Method

$$R(A B C D E F)$$

Prime attr

Not prime attr.

$$\text{candidate key} \rightarrow \{AB, BC\}$$

$$\{AB \rightarrow C, B \rightarrow D, D \rightarrow E, AE \rightarrow F, C \rightarrow A\}$$

Prime \rightarrow Prime
(superkey)

Prime \rightarrow not prime

Not prime \rightarrow not prime

not prime \rightarrow not prime

proper subset
of candidate key

(Not SK)
①

(Not SK)
②

(Not SK)
③

proper subset
of candidate key
(BC)

∴ Not SK

Forms Redundancy

Note —

X is not a superkey of relation R if — $X \rightarrow Y$

① X is proper subset of candidate key.

(08)

② X is non prime attribute

(08)

③ X is proper subset of candidate key combines with non prime attributes.

\Rightarrow Non trivial FD $X \rightarrow Y$
with X not superkey.
(forms redundancy)

1 NF

2 NF

3 NF

BCNF

(does not allow any non trivial FD which form redundancy)

① $\left[\begin{matrix} \text{proper subset} \\ \text{of candidate key} \end{matrix} \right] \rightarrow \left[\begin{matrix} \text{non} \\ \text{prime} \\ \text{attribute} \end{matrix} \right]$	Allowed	Not allowed	Not allowed	Not allowed
② $\left[\begin{matrix} \text{non prime} \\ \text{attribute} \end{matrix} \right] \rightarrow \left[\begin{matrix} \text{non} \\ \text{prime} \\ \text{attr} \end{matrix} \right]$	Allowed	Allowed	Not allowed	Not allowed

	1NF	2NF	3NF	BCNF
③ $\left[\begin{array}{l} \text{Proper subset of} \\ \text{candidate key} \\ \text{Non prime attri} \end{array} \right] \rightarrow \left[\begin{array}{l} \text{Non} \\ \text{prime} \\ \text{attri} \end{array} \right]$	Allowed	Allowed	Not allowed	Not allowed
④ $\left[\begin{array}{l} \text{Proper subset of} \\ \text{one candidate} \\ \text{key} \end{array} \right] \rightarrow \left[\begin{array}{l} \text{Proper subset} \\ \text{of other} \\ \text{candidate key} \end{array} \right]$	Allowed	Allowed	Not allowed	Not allowed

Ques: If rel R is in 3NF but not in BCNF, then which FD must be in R.

solutn.
(from above table)

$$\left[\begin{array}{l} \text{Proper subset of one} \\ \text{candidate key} \end{array} \right] \rightarrow \left[\begin{array}{l} \text{Proper subset of other} \\ \text{candidate key} \end{array} \right]$$

must exist in R.

Ques: If rel R is in 3NF and no non trivial FD such that

$$\left\{ \begin{array}{l} \text{Proper subset of} \\ \text{cand key} \end{array} \right\} \rightarrow \left\{ \begin{array}{l} \text{Proper subset of} \\ \text{other cand key} \end{array} \right\}, \text{ then rel R}$$

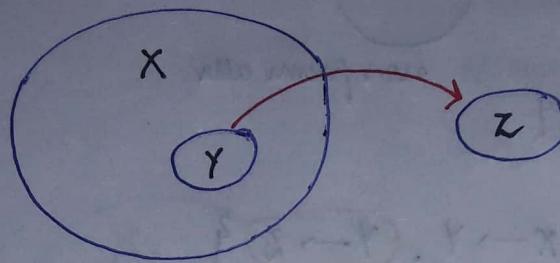
must also be in?

BCNF

Second Normal Form (2NF) -

Relational schema (R) is in 2NF iff no partial dependency in relation R .

Partial Dependency :



X : any candidate key

$Y \subset X$

Z : Non prime attribute

$Y \subset X$

$Y \rightarrow Z$ Partial dependency.

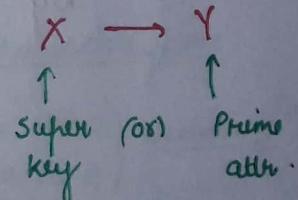
Third Normal Form (3NF) -

Relational schema R is in 3NF iff

Every nontrivial FD $X \rightarrow Y$ in R with.

① X must be superkey
(or)

② Y must be prime attribute.



{ $X \rightarrow Y$, $Y \rightarrow Z$ }
↑
super key .
↑
prime attr.

(candidate key) → Non Prime

✓

{ in 3NF

(superkey or
Not superkey) → Prime

✓

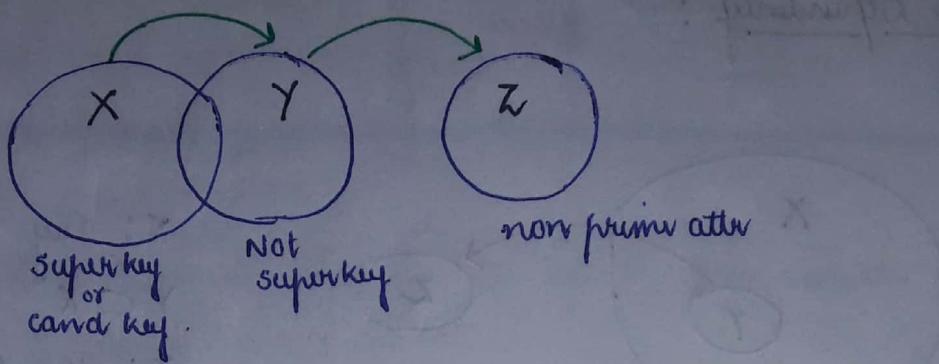
{

in 3NF

definition -

A relation R is in 3NF if there is no transitive dependency.

Transitive dependency:



$$\{ x \rightarrow y, y \rightarrow z \}$$

↓
Transitive dependency

(Non prime attr (z) transitively determined by superkey or cand key)



Not Allowed in 3NF

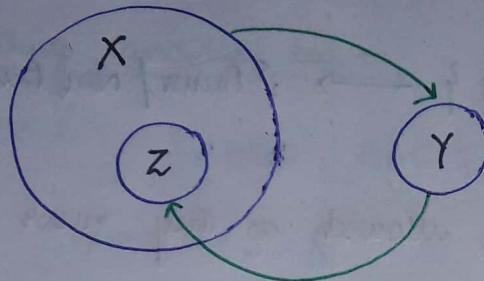
→ If $x \rightarrow y, y \rightarrow z$
↓
super key ↓
super key non prime

↳ If this y is super key, then it is not transitive dependency as z (non prime) is determined by super key.

↓
Allowed in 3NF

→ Prime attribute transitively determined by superkey is allowed in 3NF.

$\{ \underbrace{x \rightarrow y}_{\text{superkey}}, \underbrace{y \rightarrow z}_{\text{not superkey}} \}$ prime
↓
since it is prime attribute
Allowed in 3NF



X: candidate key
Y: Not superkey
Z: prime attr.

Example,

$R(ABC) \nmid AB \rightarrow C, C \rightarrow A$

Candidate key: AB, BC

↑
super key ↑
prime attr.

(∴ Allowed in 3NF)

But $C \rightarrow A$ forms redundancy in the relation as

(proper subset of one cand. key) determines (proper subset of another rel.)



Soln is BC NF

Boyer Codd NF (BCNF) —

- Relational schema R is in BCNF iff every non-trivial FD $X \rightarrow Y$ in R is with X as superkey

All non-trivial FD's
whose determinant is
a super key



Allowed in BCNF

- BCNF : $\{ \text{super key} \} \longrightarrow \{ \text{Prime / Non Prime} \}$

Note— Trivial FD are always allowed as they never cause redundancy.

$$\left\{ \begin{array}{l} \text{NO Attri. redundancy} \\ \text{in } R \text{ over FD's} \end{array} \right\} \iff \text{Rel } R \text{ in BCNF}$$

$$\left\{ \begin{array}{l} \text{Redundancy exists in } \\ R \text{ over FD's} \\ \text{Non-trivial FD} \\ X \rightarrow Y \\ \downarrow \\ \text{not superkey} \end{array} \right\} \iff \text{Rel } R \text{ not in} \\ \text{super BCNF}$$

~~*****~~ Ques: Find the highest NF of given relation R.

① R(ABCDE)

$$\{ ABD \rightarrow C, BC \rightarrow D, CD \rightarrow E \}$$

Soln: check for BCNF

$$(ABD)^+ = ABDCE$$

(super key)

$$(BC)^+ = BCDE \quad \times \quad \therefore \text{Not in BCNF}$$

$$(CD)^+ = CDE \quad \times$$

Finding candidate key →

$$\{ ABD, ABC \}$$

Prime attr: ABCD

Non prime attr: E

Note
if there exist a trivial F.D, do not check for it, it is always allowed for all. Eg. $AB \rightarrow A$

check for 3NF →

$$ABD \rightarrow C$$

$$BC \rightarrow D$$

allowed
as C,D are
prime attr

$$CD \rightarrow E \quad \times$$

↑ not prime

∴ Not in 3NF

check for 2NF →

Here we check for partial dependency.

$$ABD \rightarrow C$$

prime attr

No partial dep

$$BC \rightarrow D$$

"

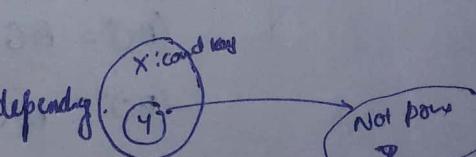
$$CD \rightarrow E$$

non prime attr

"

Not proper subset
of any candidate key

All are having no partial dependency but still it could fail



But $BC \rightarrow D$ and $CD \rightarrow E$

↓
 $\underbrace{BC \rightarrow E}_{\substack{\text{proper} \\ \text{subset}}}$ $\underbrace{\text{not prime}}_{\text{prime}}$

\therefore Partial dependency exist
 \therefore not in 2NF

don't check by this method

↓ use

2NF test: $\nexists (\text{proper subset of candidate key})^+ = \text{always prime}$

1st candidate key $\rightarrow ABD$

$$\left. \begin{array}{l} A^+ = A \\ B^+ = B \\ D^+ = D \\ (AB)^+ = AB \\ (BD)^+ = BD \\ (AD)^+ = AD \end{array} \right\} \text{all prime} \quad \therefore \text{No partial depend}$$

2nd candidate key $\rightarrow ABC$

$$\left. \begin{array}{l} A^+ = A \\ B^+ = B \\ C^+ = C \\ (AB)^+ = AB \\ (AC)^+ = AC \\ (BC)^+ = BCD(E) \end{array} \right\} \text{prime}$$

\uparrow
non prime \therefore Partial dependency exist

\therefore Not in 2NF

Since All fails \therefore highest NF \rightarrow 1NF Ans

② $R(ABCDE)$

$$\{ AB \rightarrow C, C \rightarrow D, B \rightarrow E \}$$

candidate key \rightarrow AB

prime attr: AB

non prime: CDE

$$B \rightarrow E$$

partial dependency

non prime

\therefore 2NF fails

$$\begin{aligned} A^+ &= A \\ B^+ &= B, E \\ &\uparrow \\ &\text{non prime} \end{aligned}$$

\Rightarrow highest NF = 1NF AB

③ $R(ABCD)$

$$\{ AB \rightarrow C, C \rightarrow A, AC \rightarrow D \}$$

• for BCNF:

$$(AB)^+ = ABCD$$

$C^+ = CAD \times$ Not BCNF

candidate key \rightarrow (AB, BC)

Prime: ABC

Non prime: D.

• 3NF:

$$AB \rightarrow C \checkmark$$

$$C \rightarrow A \checkmark$$

Not 3NF

$$AC \rightarrow D \not\checkmark$$

• 2NF:

Cond:

$$\underline{AB} \rightarrow \begin{aligned} A^+ &= A \\ B^+ &= B \\ \hline AB &= ABGD \end{aligned}$$

$$\underline{BC} \rightarrow \begin{aligned} B^+ &= B \\ C^+ &= C, A, D \end{aligned}$$

Not 2NF
Non FR

\therefore INF AB

④ R(ABCD)

$$\{ AB \rightarrow C, BC \rightarrow D \}$$

BCNF

$$(AB)^+ = ABCD \quad \checkmark \text{ superkey}$$

X Not in BCNF

$$(BC)^+ = BCD \times \text{not}$$

candidate key = AB

prim: AB

non prim: CD

3NF

$$\begin{array}{c} AB \rightarrow C \\ \uparrow \quad \uparrow \\ \text{candi} \quad \text{not prim} \end{array} \quad \checkmark$$

X Not in 3NF

$$\begin{array}{c} BC \rightarrow D \\ \uparrow \quad \uparrow \\ \text{not} \quad \text{non pr} \\ \text{candi} \end{array} \quad X$$

2NF

candidates
AB

$$A^+ = A$$

$$B^+ = B$$

\therefore 2NF

⑤ R(ABCDEF)

$$\{ AB \rightarrow C, C \rightarrow D, D \rightarrow AE, DE \rightarrow F, EF \rightarrow B \}$$

BCNF

$$(AB)^+ = ABCDEF \quad \checkmark$$

X Not in BCNF

$$C^+ = CDAE \cancel{FB} \quad \checkmark$$

$$(EF)^+ = EFB \quad X$$

candidate key = AB, AEF, ~~ADBF~~, D, C

prim: ABCDEF

Non prim: X

- 3NF
 $AB \rightarrow C \checkmark$
 All are prime
 \therefore kanthi ho if RHS is pair \therefore 3NF

⑥ R (ABCDEF)

$$\{ AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow B \}$$

- BCNF

$$(AB)^+ = \{ ABCDEF \} \checkmark$$

$$C^+ = CDEFB X$$

candidate key \rightarrow AB, AF, AE, AC

prime : ABCEF

Non prime : D

- 3NF

$$AB \rightarrow C \checkmark$$

$$C \rightarrow DE X$$

NOT 3NF

Not candi

- 2NF

$$\underline{AB} \rightarrow A^+ = A$$

$$B^+ = B.$$

$$AF \rightarrow F^+ = F, B$$

$$AE \rightarrow E^+ = E, F$$

$$AC \rightarrow C^+ = C, \underline{D}, E$$

X Not in 2NF

↓
Not fp

\therefore 1NF

⑧ R(ABCD)

$\{ AB \rightarrow C, BC \rightarrow A, AC \rightarrow B \}$

BCNF

ABC \times Not in BCNF

Candidate key \rightarrow ABD, BCD, ACD

pkns \rightarrow ABCD

Not pkns \rightarrow X

3NF

$AB \rightarrow C$
kuch bhi ho. \checkmark pkns

$BC \rightarrow A \quad \checkmark$

$\therefore \underline{3NF}$

$AC \rightarrow B \quad \checkmark$

⑨ R(ABCD)

$\{ AB \rightarrow CE, BD \rightarrow F, E \rightarrow F \}$

BCNF

$(AB)^+ = ABCE \times$ Not in BCNF

Candidate key =

we remove all dependency
which are not identified
by relation R

$AB \rightarrow C \quad \checkmark$

$AB \rightarrow E \quad X$

$BD \rightarrow F \quad X$

$E \rightarrow F \quad X$

\therefore only FD is $\{ AB \rightarrow C \}$

Candidate key = ABD

prim = ABD

non prim = C

$AB \rightarrow C$ is partial dependency
 \therefore 2NF X

Hence 1NF

⑦ R(ABCDE)

{ $A \rightarrow BC$, $CD \rightarrow E$, $E \rightarrow A$, $B \rightarrow D$ }

candidate key $\rightarrow A, E, BC, CD$

prim = ABCDE

all are prim

\therefore 3NF

⑧ R(ABC)

{ $AB \rightarrow E$, $DE \rightarrow C$ }

Remove FD which are not in R.

All are removed

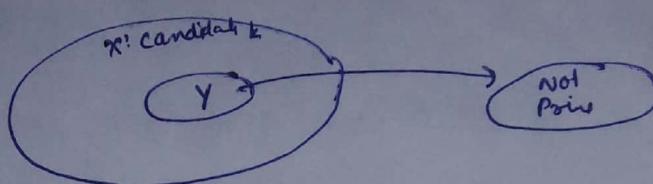
i.e. No dependency left

\therefore R is in BCNF

Ques: If relation R with only simple candidate key, then which is true about relation R?

- a) R in BCNF
- b) R in 3NF but may not BCNF
- c) ✓ R in 2NF but may or may not 3NF
- d) R in 1NF but " " " 2NF

If candidate key is simple



This condition is not possible as only simple candidate keys in R.

i.e. if simple candidate key \rightarrow then proper subset = null i.e. \emptyset only.

$$\{Y\} \subset \{A, B, C, D\}$$

\therefore 2NF

Eg - $R(A B C D)$ $\{A \rightarrow B, B \rightarrow A C, C \rightarrow D\}$

candidate key $\rightarrow \{A, B\}$

$C \rightarrow D$
not prime
not candidate

X 3NF fails

But 2NF ✓

∴ 3NF may or may not but 2NF will always be True.

Eg - $R(A B C)$ $\{A \rightarrow B, B \rightarrow A C\}$

candidate key = $\{A, B\}$

2NF ✓
3NF ✓
BCNF ✓

Ques: If every attribute of R is prime attribute then R is
in 3NF but may or may not BCNF

- a) 1NF ~~c) 3NF~~
b) 2NF d) 4NF

Ans: If R is in 3NF (and) at most one compound candidate key in rel R (remaining candidate keys are simple then —

~~a) R also in BCNF~~

- b) R may not BCNF
c) R not BCNF
d) can not decide

R is in 3NF

→ prime attr
→ non prime
candidate key

$$\begin{array}{l} ABCDE \\ AB \xrightarrow{A \rightarrow B} D \xrightarrow{D \rightarrow E} E \\ AB \xrightarrow{B \rightarrow A} C \xrightarrow{C \rightarrow D} D \\ (AB)^+ = ABCDEF \\ C^+ = CDEABC \\ D^+ = DEABC \\ E^+ = ABCDE \end{array}$$

D, C

Ques: $R(ABC)$

with no non-trivial FD's, then highest NF of

R BCNF

No non-trivial FD's \rightarrow No redundancy over FD's



BCNF relⁿ

Ques: Relation R with only 2 attributes, then relation R is always in BCNF (also in other higher NF)

$R(AB)$

There will be 4 condition.

① $R(AB)$ $\{ \underline{A} \rightarrow B \}$ ✓ BCNF

② $R(AB)$ $\{ \underline{B} \rightarrow A \}$ ✓ BCNF

③ $R(AB)$ $\{ \underline{A} \rightarrow B, \underline{B} \rightarrow A \}$ ✓ BCNF

④ $R(AB)$ $\{ \text{No non-trivial FD's} \}$ ✓ BCNF

Decomposition into higher NF -

Relation R decompose into '2NF', '3NF', 'BCNF' with

- loss less join decomposition
and
- dependency preserving decomposition

① $R(A B C D E)$

$$\{AB \rightarrow C, C \rightarrow D, \underbrace{B \rightarrow E}\}$$

candidate key $\rightarrow AB$

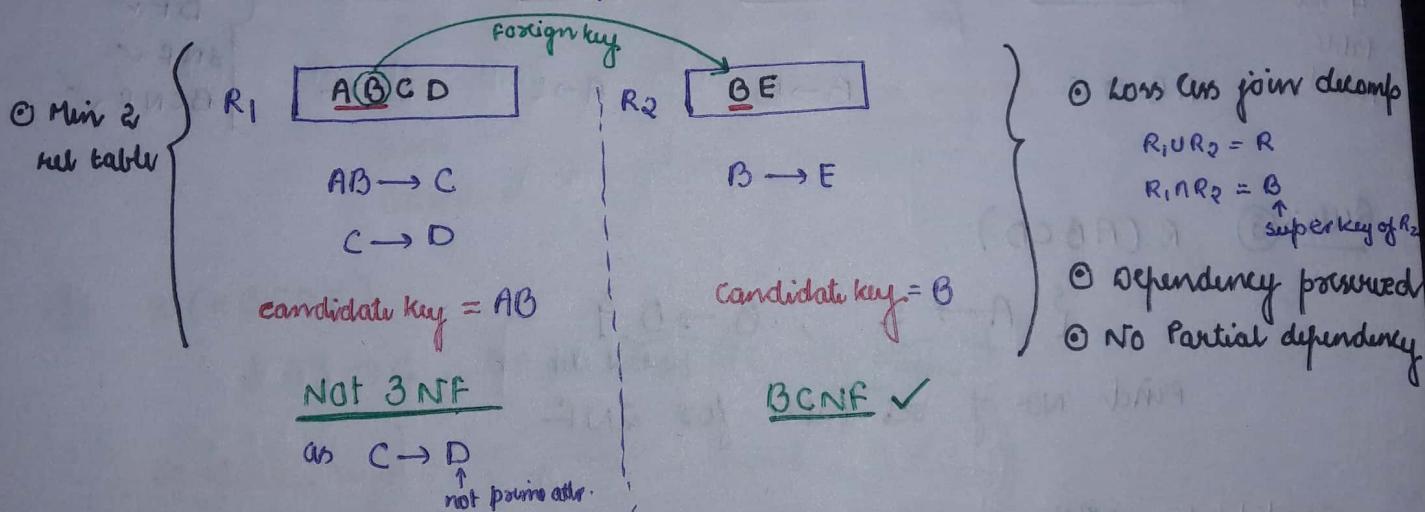
partial dependency

$\therefore R$ not in 2NF

$\Rightarrow R$ in 1NF

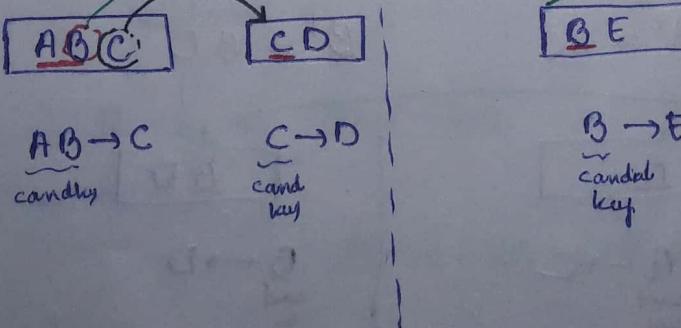
2NF decomposition -

NOW decompose $B \rightarrow E$ as it is partial dependency



3NF design -

① Min 3 rel table



- ① Loss less join decomp
- ② Dependency preserved
- ③ 3NF ✓
- ④ BCNF ✓

② R (ABCDEF)

candidate key = AB
 prime = AB non prime = CDEF

$\{AB \rightarrow C, BE \rightarrow D\}$, $A \rightarrow E, E \rightarrow F$

Partial dependency

Partial dependency $(A \rightarrow E, A \rightarrow F)$

2NF design —

$$B^+ = BD$$

$$A^+ = AEF$$

3 rel
table

$AB \rightarrow C$
 cand key ✓

$B \rightarrow D$
 cand key ✓

$A \rightarrow E, E \rightarrow F$
 cand key ✗
 ✗
 ∴ NOT 3NF

lossless
dependency present
NO Partial dep
∴ 2NF

3NF design —

4 rel
table
needed

$AB \rightarrow C$

$B \rightarrow D$

$A \rightarrow E, E \rightarrow F$

LLJV
DP ✓
3NF ✓
BCNF ✓

Ques: ③ R (ABCD)

$\{A \rightarrow C, B \rightarrow D\}$
 find no. of rel table for 2NF

Candidate key = AB

prime = AB

non prime = CD

$$A^+ = AC$$

$$B^+ = BD$$

$$AB$$

$$AC$$

$$BD$$

$A \rightarrow C$
 cand key

$B \rightarrow D$
 cand key

3 tables

LLJV ✓

DP ✓

2NF ✓

3NF ✓

(for only 2 table
it is lossy)

④ R(ABCDEF)

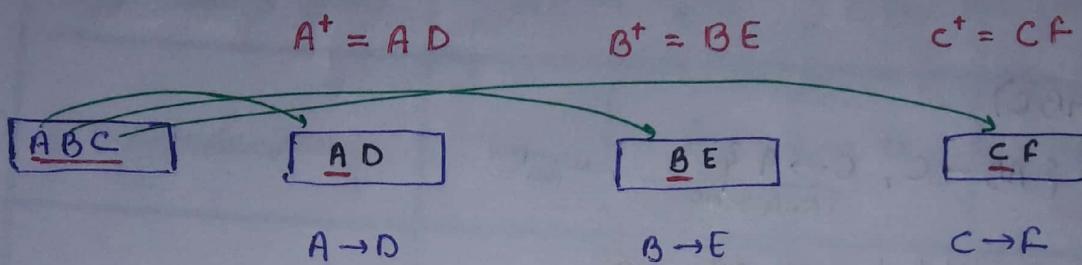
$$\{ \begin{array}{l} A \rightarrow D \\ B \rightarrow E \\ C \rightarrow F \end{array} \}$$

partial dep
partial "
partial "

Find no. of rel. tables for 2NF?

Candidate key. \rightarrow ABC

prime = ABC not prime = DEF



Loss less join ✓

DP ✓

2NF ✓

3NF ✓

BCNF ✓

∴ 4 tables needed

⑤ R(ABCDE)

$$\{ AB \rightarrow C, BC \rightarrow A, AC \rightarrow B \}$$

Candidate keys = { ABDE, ACDE, BCDE }

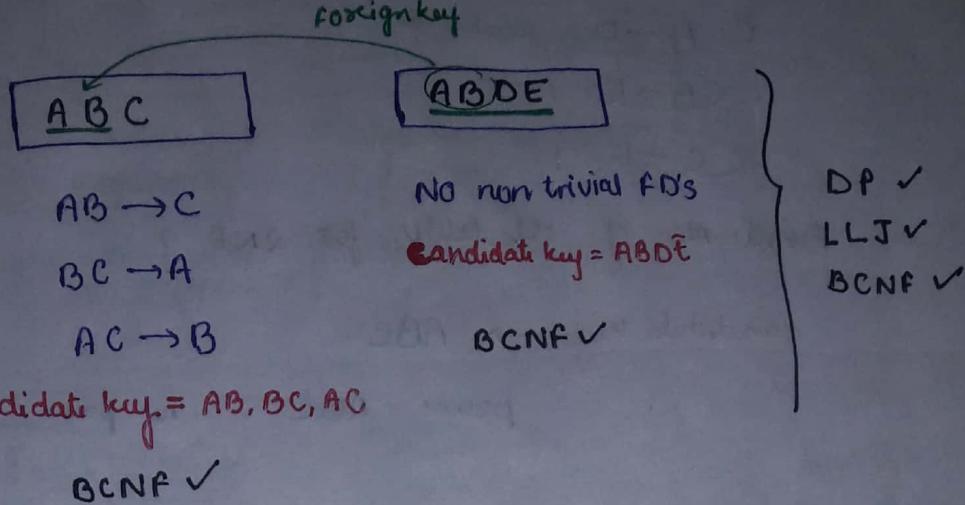
prime = ABCDE

not prime = X

∴ R is in 3NF but Not BCNF

BCNF decomposition

BCNF decomposition -



⑥ $R(ABC)$

$\{ AB \rightarrow C, C \rightarrow A \}$ Fails BCNF \therefore decompose thus.

candidate key = AB, CB

prime = ABC , non prime = X

All are prime

\therefore It is in 3NF but not BCNF

↓ BCNF decomposition

<u>B C</u>

<u>C A</u>

~~AB → C~~
 NO Non-trivial
 FD's
 BCNF ✓

$C \rightarrow A$
 BCNF ✓

LLJ ✓
 BCNF ✓
 but
 DP X
 fails

↓
 Re-design

ABC

$AB \rightarrow C$

$C \rightarrow A$

LLJ ✓

DP ✓

but not BCNF

X

∴ Above relation R is not possible to decompose BCNF with dependency preserving decomposition.

Most accurate NF

DB design Goals	1NF	2NF	3NF	BCNF
① 0% redundancy	NO	NO	NO	Yes over FD's No over MUDs
② * Loss less Join decomposition	Yes	Yes	Yes	Lossless decompr of 1NF, 2NF, 3NF ^{BCNF} possible for every relation.
③ * Dependency Preserving decomposition	Yes	Yes	Yes	May Not Not every rel can decompose into BCNF with dependency preserving decompo.

Queries

① Relational Algebra
(mathematical formula)

Procedural Query lang
formulation of what data
retrieved (and) how data
is retrieved

② SQL (used to run
in computer)

Non procedural Query lang
formulation of what data
is retrieved.

③ Relational calculus
(mathematical formula)

Relational Algebra

→ Basic Operators :

π	→ Projection	} unary operator
σ	→ selection	
\times	→ cross product	} Binary operator
\cup	→ Union	
$-$	→ Minus (set difference)	} Unary operator
ρ	→ Rename	

→ Derived operators -

Binary operators	\cap	: intersection (using "-")
	\bowtie	: Join (using " π, σ, \times ")
	$/$: division (using " $\pi, \times, -$ ")

Projection (π) -

It is used to project required list of attributes from relation R.

$$\pi_{\text{attr-list}}(R)$$

Selection (σ) -

It is used to retrieve records which are satisfying the given predicate condition (P) from relation R.

$$\sigma_P(R)$$

Eg -

R

	A	B	C
4	6	8	
7	5	8	
5	6	8	
8	3	5	

$$\pi_{BC}(R) :$$

B	C
6	8
5	8
3	5

{ distinct tuple in result }

$$\sigma_{A \leq 6}(R) :$$

A	B	C
4	6	8
5	6	8

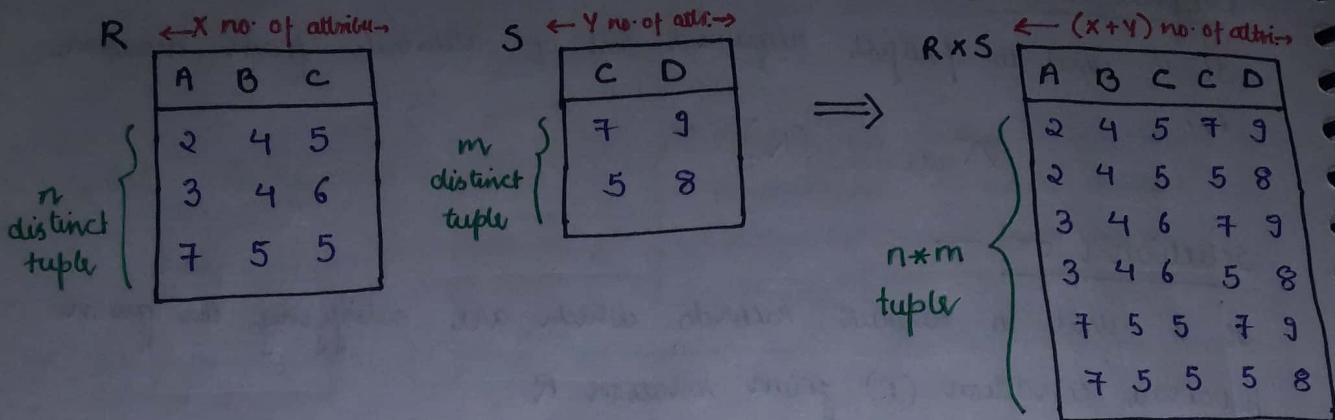
$$\pi_{BC}(\sigma_{A \leq 6}(R)) :$$

B	C
6	8

Cross Product (\times) -

$R \times S$ results in all attributes of R followed by all attributes of S (and) each record of R pairs with every record of S.

Eg -



→ Relation R with n tuples and Relation S with 0 tuples, then No. of Tuples in $R \times S = \underline{0 \text{ tuple}} \text{ (empty)}$

Eg - $A = \{1, 2, 3\}$
 $B = \{3\}$
 $A \times B = \{3\}$ Ans

Joins —

- ① Natural Join (\bowtie)
- ② Conditional Join (\bowtie_c)
- ③ Outer Joins
 - left outer join (\bowtie_l or \bowtie_{lc})
 - Right outer join (\bowtie_r or \bowtie_{rc})
 - full outer join (\bowtie_f or \bowtie_{fc})
- ④ Equi Join ($\bowtie_=$)
 conditional join with equal condition.

Natural Join -

$$R \bowtie S \equiv \pi_{\text{distinct attributes}}(\sigma_p(R \times S))$$

p : equality of common attributes of R and S

Eg -

R	A	B	C	S	C	D	R \times S
2	4	5		7	9		2 4 5 7 9
3	4	6		5	8		3 4 6 7 9

(Ans)
Now for σ_p i.e selection where common attributes are same.

A	B	C	D
2	4		5

only 1 column
of C will
be there

↓
Now π_{dist} i.e project it.

Eg - Now for above,

$$R \bowtie S \equiv \pi_{A B C D}(\sigma_{R.C = S.C}(R \times S))$$

Eg - $T_1(A \underline{B} C)$ $T_2(B \underline{C} D)$

$$T_1 \bowtie T_2 \equiv \pi_{A B C D}(\sigma_{T_1.B = T_2.B \wedge T_1.C = T_2.C}(T_1 \times T_2))$$

Eg - $T_1(A, B)$ $T_2(C, D)$. find natural join

$$T_1 \bowtie T_2 \equiv T_1 \times T_2$$

A	B
4	5
6	7

C	D
3	5
4	5

\Rightarrow

A	B	C	D
4	5	3	5
4	5	4	5
6	7	3	5
6	7	4	5

$n \times m$

Natural join is equal to
cross product if

Join condition is empty

Conditional Join —
 (\bowtie_c)

$$R \bowtie_c S \equiv \sigma_c(R \times S)$$

Eg -

R

A	B	C
2	4	5
3	4	6
7	5	5

S

C	D
7	9
5	8

$$R \bowtie S \underset{R.C < S.C}{\equiv} \sigma_{R.C < S.C}(R \times S)$$

↓

A	B	C	C	D
2	4	5	7	9
3	4	6	7	9
7	5	5	7	9

Ans

Note —

$$R \bowtie S \neq \sigma_{R.A < 5} (R \bowtie S)$$



$$\sigma_{R.A < 5} (R \times S)$$

$$\downarrow$$

$$\sigma_{\substack{R.C = S.C \\ R.A < 5}} (R \times S)$$

- (Equality of common attribute is not checked)

(Natural Join, ∵ equality of common attribute is also checked)

- Duplicate tuples exist

A	B	C	C	D
2	4	5	7	9
2	4	5	5	8
3	4	6	7	9
3	4	6	6	8

A	B	C	D
2	4	5	8

NO similar tuples.

R → A	B	C
2	4	5
3	4	6
7	5	5

S → C	D
7	9
5	8

Outer Join —

① Left Outer Join (R ⋈ S)

$$R \bowtie S \equiv R \bowtie S \text{ tuples}$$

and

tuples of R, which
failed the join condition

Eg -

R

A	B	C
4	5	7
4	6	3
4	7	5

Failed
join
cond

S

C	D	E	F
4	3	5	
7	5	6	
6	8	9	

$$R \bowtie S$$

left part

A	B	C	D	E	F
4	5	7	4	3	5
4	5	7	7	5	6
4	6	3	Null	Null	Null
4	7	5	Null	Null	Null

② Right Outer Join (R ⋈ S)

right part

A	B	C	D	E	F
4	5	7	4	3	5
4	5	7	7	5	6
Null	Null	6	9	8	9

③ Full Outer Join (\bowtie)

$$R \bowtie S \equiv (R \bowtie S) \cup (R \bowtie S)$$

Eg -

A	B	C	D	E	F
4	5	7	4	3	5
4	5	7	7	5	6
4	6	Null	Null	Null	Null
4	7	5	Null	Null	Null
Null	Null	6	9	8	9

6/11/17

Ans: $R(A\ldots) \quad S(B\ldots)$

Retain 'A' values of R , those are more than some
 'B' values of S .

↓
 any /
 atleast one

Soln:

$R(A\ldots)$

2

4

6

$S(B\ldots)$

3

5

$R \times S$

⇒

A...	B...
2	3
2	5
4	3
4	5
6	3
6	5

⇒

A
4
6

$$\therefore \pi_{R:A} \left(\sigma_{R.A > S.B} (R \times S) \right)$$

Ans

(Q8)

$$\pi_{R,A} (R \bowtie S)$$

Ans

← using conditional join

Note -

$$\begin{array}{l} R \bowtie_c S \\ \sigma_c(R \times S) \\ R \bowtie S \end{array}$$

} used to retrieve records of rel "R" which ~~are~~ satisfy the given specific condition with atleast one/any/some records of S

Ques: $R(A\ldots) \text{ } S(B\ldots)$

Retrieve 'A' val of R those are more than every 'B' of rel (S)

\downarrow
all

Soluⁿ
 $R(A\ldots)$

	$R(A\ldots)$	$S(B\ldots)$
2	✓ 2	3
4	✓ 4	5
✓ 6	✗ 6	

$\cap (> \text{All}) \equiv \langle \text{some or any}$

$\Rightarrow (\text{All } A \text{ value of } R \leq \text{any } B)$

$$\pi_A(R) = \pi_A(R \bowtie S) \underset{R.A \leq S.B}{\text{Ans}}$$

A val of R
those are more
than every B
of S

$$= \left(\text{All } A \text{ value of relation } R \right) - \left(\text{'A' value of rel } R \text{ those are less than equal to some 'B' of } S \right)$$

$\Rightarrow P$ (Rename) -

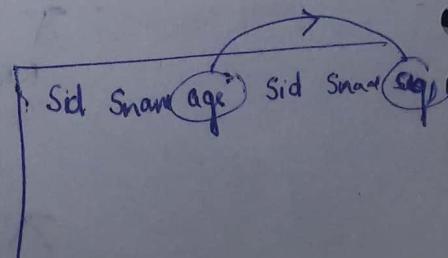
for condition where cross product of same table is done

$$\sigma(\text{stud} \times \text{stud})$$

$\underbrace{\text{stud.age} > \text{stud.ag}}$

\hookrightarrow produced ambiguity

Hence we use rename operator



- Rename is used to rename table name or attribute.

Eg - There is a table named stud (sid sname age)

For renaming table name -

$\rho(\text{Temp}, \text{stud})$: stud is renamed as
Temp (sid, sname age)

For renaming the attribute of table -

$\rho(\text{stud})_{I, N, A}$: Attributes of table stud are changed
stud (I, N, A)

For renaming some attribute of table -

$\rho(\text{stud})_{1 \rightarrow I, 3 \rightarrow A}$: stud (I, sname, A)

↑ just by giving
column no. we can change the name

Ques: stud (sid sname age)

Retrieve Sid's of student whose age is more than age
of some student .

For some we use
cond. join



stud

	Sid	sname	age
x	S1		10
-	S2		20
✓	S3		30
✓	S4		40

> stud

	Sid	sname	age
	S1		10
	S2		20
	S3		30
	S4		40

Rename this table
name

$$\pi_{\text{stud} \cdot \text{sid}} \left(\text{stud} \bowtie_{\text{stud} \cdot \text{age} > \text{Temp} \cdot \text{age}} P(\text{Temp}, \text{stud}) \right)$$

OR
Method 2

By renaming column —

$$\pi_{\text{sid}} \left(\text{stud} \bowtie_{\text{age} > A} f_{I, N, A}(\text{stud}) \right)$$

Ques: Retrieve sid's of student whose age is maximum.

*	10	10
*	20	$\geq \text{all}$ 20
*	30	\rightarrow 30
✓	40	40

\geq Every = All value - < some i.e.
at least one

$$\text{Sid's with max age} = \text{Sid's of all Student} - \text{Sid's of student whose age is less than some student age}$$

\bowtie_c
used

$$\therefore \pi_{\text{sid}}(\text{stud}) - \pi_{\text{sid}} \left(\text{stud} \bowtie_{\text{age} < A} f_{I, N, A}(\text{stud}) \right)$$

Ans.

<u>stud</u> -		<u>stud</u> -		<u>sid</u> -		<u>sid</u> -	
Sid	age	Sid	age	Sid	-	Sid	-
s1	10	s1	10	s1		s1	
s2	20	s2	20	s2		s2	
s3	30	s3	30	s3		s3	
s4	40	s4	40	s4			

\Rightarrow

s4

Ques: Retrieve sid's of student whose age is minimum.
 ↓
 ≤ all

✓ 10	10
✗ 20	20
✗ 30	30
✗ 40	40

$$\pi_{\text{sid}}(\text{stud}) - \pi_{\text{sid}} \left(\text{stud} \setminus_{\text{age} > A} \rho(\text{stud}) \right)_{I,N,A}$$

(OR)

$$\pi_{\text{sid}}(\text{stud}) - \pi_I \left(\text{stud} \setminus_{\text{age} < A} \rho(\text{stud}) \right)_{I,N,A}$$

Ques: stud (sid sname age)

Retrieve sid's whose age is max for each name.

stud

Sid	Sname	age
s1	A	10
s2	A	20
s3	B	30
s4	B	40

Sid	Sname	age
s1	A	10
s2	A	20
s3	B	30
s4	B	40

i.e Sid's whose
age is ≥ every
student of
same
name.

Sid's whose age
is max for
each name = Sid's of
all student — Sid's whose age < some
student age of same
name
 cond
join
 Δ_2

$$= \pi_{\text{sid}}(\text{stud}) - \pi_{\text{sid}} \left(\text{stud} \setminus_{\substack{\text{sname} = N \\ \text{age} < A}} \rho(\text{stud}) \right)_{I,N,A}$$

$$\begin{bmatrix} s1 \\ s2 \\ s3 \\ s4 \end{bmatrix} - \begin{bmatrix} s1 \\ s3 \end{bmatrix} \\ \cong \begin{bmatrix} s2 \\ s4 \end{bmatrix}$$

Ques: stud (sid, marks, gender)

- ① Retrieve sid's of female student who scored more marks than some male student.

↓
at least one
(Σ)

Sid	Mark	Gend	X	Sid	Mark	Gend
s ₁						
s ₂						
s ₃						
s ₄						

$\exists_{\text{Sid.}} \left(\text{stud.} \bowtie \rho(\text{stud}) \right)$
Gender = 'F' \wedge I, M, G.
 $G = 'M'$
Marks > M

- ② Retrieve Sid's of female student who scored less marks than every male student.

$\forall (\text{All}) \equiv \geq \text{some}$

$\Rightarrow \exists_{\text{Sid.}} \left(\cancel{\forall} (\text{stud.}) - \exists_{\text{Sid.}} \left(\text{stud.} \bowtie \rho(\text{stud}) \right) \right)$
Gender = 'F'
 $G = 'M'$
Marks \geq M

SET Operators -

- ① \cup : union
- ② $-$: minus (set difference)
- ③ \cap : intersection \rightarrow derived operator

} basic operators

\rightarrow To apply set operations, relations must be union compatible.

\rightarrow R and S are union compatible iff.

- ① No. of attributes of R \equiv No. of attributes of S
- and ② Domain of each attribute of R must be same as that of S

$R(A_1 A_2) \quad S(B_1 B_2)$

domain (A_1) \rightarrow possible value accepted by attribute A_1

Eg -

$\pi_{sid}(\dots)$ set operator ($\cup, \cap, -$) $\pi_{sid sname}(\dots)$

X Not allowed

(as no. of attributes are different
in one) $\boxed{sid} \neq \boxed{sid \atop sname}$

Eg -

$\pi_{sid sname}(\dots)$ set operator $\pi_{sid age}(\dots)$

X Not allowed

(as domain is different)
 $sname \neq age$

Eg -

$\pi_{sid sname}(\dots)$ set operator $\pi_{IN}(\dots)$

↑ set of student id ↗ name of student

✓ Allowed

Note ① RUS, RNS, R-S results are always treated as distinct tuples.

R	A	S	B	RNS	A
	2		2		2
	2		2		3
	2		3		
	3		3		
	3		3		
	4		5		
			5		

② RUS, RNS, R-S expressions resulted schema (i.e structure) is same schema of left side relation R.

For R set operator S

name of result table = R
 name of column's name = column name of R.

Eg -

R	A	B	C	S	D	E	F
	2	4	6		3	5	7
	3	5	7		4	6	5
	4	6	5		2	4	8

Union -

$$RUS = \{ x \mid x \in R \cup x \in S \}$$

OR

RUS

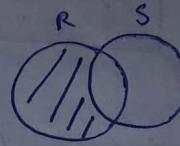
A	B	C
2	4	6
3	5	7
4	6	5
2	4	8

Minus —

$$R - S = \{ x \mid x \in R \wedge \underbrace{x \notin S} \}$$

But not

(Records of R but not S) \equiv Records of only R



$$\therefore R - S =$$

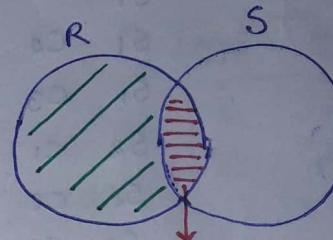
A	B	C
2	4	6

Intersection —

$$R \cap S \equiv R - (R - S)$$

$$R \cap S = \{ x \mid x \in R \wedge \underbrace{x \in S} \}$$

AND



$$R - (R - S) \equiv R \cap S$$

$$R \cap S =$$

A	B	C
3	5	7
4	6	5

Note → If T_1 and T_2 are with same no. and name of attribute, then

$$T_1 \cap T_2 = T_1 \bowtie T_2$$

$T_1 (A \ B)$	$T_2 (A \ B)$
2 4	2 4
6 8	6 5

$$T_1 \cap T_2 = 2 \ 4$$

$$T_1 \bowtie T_2 = 2 \ 4 \rightarrow \text{matches } A \text{ to } A \text{ from } T_1 \text{ and } T_2 \text{ also } B \text{ to } B.$$

$$T_1 \cup T_2 = T_1 \bowtie T_2$$

$$T_1 \cap T_2 = T_1 - (T_1 - T_2) \equiv T_1 \bowtie T_2$$

Division (/ or ÷)

enroll	(<u>sid</u> , <u>cid</u>)
	S ₁ C ₁
	S ₁ C ₂
	S ₁ C ₃
	S ₂ C ₁
	S ₂ C ₂
	S ₃ C ₁

Course	(<u>cid</u> , ...)
	C ₁
	C ₂
	C ₃

all courses

Retrieve sid's enrolled for all courses.

$$\pi_{\text{sid}, \text{cid}}(\text{enroll}) / \pi_{\text{cid}}(\text{course}) \equiv \boxed{\begin{array}{|c|} \hline \text{sid} \\ \hline \text{s}_1 \\ \hline \end{array}}$$

✓ {	S ₁ C ₁	C ₁
	S ₁ C ₂	C ₂
	S ₁ C ₃	C ₃
✗	S ₂ C ₁	
✗	S ₂ C ₂	
✗	S ₃ C ₁	

sid which is mapped
or paired with
every course
↓
represented by $\frac{\circ}{\circ}$

Expansion of "÷"

Sid's enrolled for every course

$$\pi_{\text{sid}, \text{cid}}(\text{enroll}) / \pi_{\text{cid}}(\text{course})$$

Step 1: Sid's of student not enrolled every course
(Sid's enrolled proper subset of all courses)

$$\pi_{\text{Sid}} \left(\underbrace{\pi_{\text{Sid}}(\text{enroll}) \times \pi_{\text{cid}}(\text{course})}_{\text{Every student enrolled, for every course}} - \pi_{\text{Sid}, \text{cid}}(\text{enroll}) \right)$$

Actual pairs enrolled

$\{ C_1 \}$
 $\{ C_2 \}$
 $\{ C_3 \}$
 $\{ C_1, C_2 \}$
 $\{ C_2, C_3 \}$
 $\{ C_3, C_1 \}$

if it is any of it then not enrolled for all courses

Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₁	C ₃
S ₂	C ₁
S ₂	C ₂
S ₂	C ₃
S ₃	C ₁
S ₃	C ₂
S ₃	C ₃

Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₁	C ₃
S ₂	C ₁
S ₂	C ₂
S ₂	C ₃
S ₃	C ₁
S ₃	C ₂
S ₃	C ₃

\Rightarrow

Sid	Cid
S ₂	C ₃
S ₃	C ₂
S ₃	C ₃

Sid
S ₂
S ₃

Sid's not enrolled every course

Step 2:

$$[\text{Sid's enrolled every course}] = [\text{Sid's enroll some course}] - [\text{Sid's not enrolled every course}]$$

$$\pi_{\text{Sid}, \text{cid}}(\text{enroll}) / \pi_{\text{cid}}(\text{course}) \equiv \pi_{\text{Sid}}(\text{enroll}) - \pi_{\text{Sid}} \left(\pi_{\text{Sid}}(\text{enroll}) \times \pi_{\text{cid}}(\text{course}) - \pi_{\text{Sid}, \text{cid}}(\text{enroll}) \right)$$

$$\pi_{AB}(R) / \pi_B(S) \equiv \pi_A(R) - \pi_A(\pi_A(R) \times \pi_B(S) - \pi_{AB}(R))$$

↓
'A' value of rel R
which are paired
by every 'B' value
of S.

π , $-$, \times basic operators
are used to express '/'

Eg - $R(ABCD) / S(CD)$
 $\Rightarrow R(AB)$ Ans

Ques: $\pi_{\text{stud}}(\text{sid sname age})$ [all student]

$\pi_{\text{course}}(\text{cid cname instrutor})$ [all courses]

$\pi_{\text{enroll}}(\text{sid cid fm})$ [Mapping] Sid enrolled courses

① Retrieve Sid's enrolled some course taught by KORTH
 \uparrow
instructor.

$\pi_{\text{Sid}}(\text{sid}) (\cancel{\text{enroll}} \bowtie \cancel{\text{course}})$

instruct = "KORTH"
enroll.cid = course.cid

OR

Enroll

Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₂	C ₂
S ₃	C ₃
S ₃	C ₄

 \times course

Cid	Instr
C ₁	KORTH
C ₂	KORTH
C ₃	Nawath
C ₄	Nawati

$\pi_{\text{Sid}}(\text{O}_{\text{Enroll.cid} = \text{course.cid}}(\text{Enroll} \times \text{course}))$

Instr = KORTH

cost of query = $2 * (7 \times 4) = 56$

OR

$$\pi_{\text{sid}} \left(\sigma_{\text{enroll.cid} = \text{course.cid}} \left(\text{enroll} \times \sigma_{\text{inst} = \text{KORTH}} (\text{course}) \right) \right)$$

for cross product
 7×2
 4 compare
 (2 record returned)
 $\text{ginst} = \text{KORTH}$

$$\begin{aligned} \text{cost of query} &= 4 + (7 \times 2) \times 1 \\ &= 18 \end{aligned}$$

OR

$$\pi_{\text{sid}} \left(\sigma_{\text{inst} = \text{KORTH}} (\text{enroll} \bowtie \text{course}) \right)$$

Natural Join

Here also instead of joining
will pull courses

↓
if we retrieve KORTH
course and then
join

$$\pi_{\text{sid}} \left(\text{enroll} \bowtie \left(\sigma_{\text{inst} = \text{KORTH}} (\text{course}) \right) \right)$$

- ② Retrieve Sid's enrolled by every course taught by KORTH
 ↓
 division is used

$$\pi_{\text{sid.cid}} (\text{enroll}) / \pi_{\text{cid}} \left(\sigma_{\text{inst} = \text{KORTH}} (\text{course}) \right)$$

(OR)

$$\pi_{\text{sid}} (\text{enroll}) - \pi_{\text{sid}} \left(\pi_{\text{sid}} (\text{enroll}) \times \pi_{\text{cid}} \left(\sigma_{\text{inst} = \text{KORTH}} (\text{course}) \right) - \pi_{\text{sid.cid}} (\text{enroll}) \right)$$

↑
 Sid's
 not enrolled every
 KORTH course

Every sid enrolled
 every KORTH course

- ③ Retrieve cid's enrolled by some student whose age more than 20)

$$\pi_{cid} \left(\sigma_{age=20} (Enroll \bowtie stud) \right)$$

(OR)

$$\pi_{cid} (Enroll \bowtie \sigma_{age > 20} (stud))$$

- ④ Retrieve cid's enrolled by every stud whose age more than 20

$$\pi_{sid,cid} (Enroll) / \pi_{sid} (\sigma_{age=20} (stud))$$

Note -

Pair with every - \Rightarrow use \div

> every ... \Rightarrow All val - { \leq some}

- ⑤ Retrieve some sid's enrolled some course taught by Korth or Nawathe.

$$\pi_{sid} (Enroll \bowtie \sigma_{inst=KORTH} (course))$$

\checkmark
inst = Nawathe

OR

$$\pi_{sid} (Enroll \bowtie \sigma_{inst=KORTH} (course)) \cup \pi_{sid} (Enroll \bowtie \sigma_{inst=Nawathe} (course))$$

⑥ Retrieve sid's enrolled some course taught by Korth and Navathe

$$\pi_{\text{Sid}} (\text{Enroll} \bowtie \sigma_{\text{inst} = \text{KORTH}} \text{course})$$

$\text{inst} = \text{KORTH}$
 $\text{inst} = \text{Navathe}$

X wrong as never true for AND cond.

(OR)

$$\pi_{\text{sid}} (\text{Enroll} \bowtie \sigma_{\text{inst} = \text{KORTH}} \text{course}) \cap \pi_{\text{sid}} (\text{Enroll} \bowtie \sigma_{\text{inst} = \text{Navathe}} \text{course})$$

⑦ Retrieve Sid's every course taught by Navathe and Korth.

<u>Enroll</u>	
Sid	Cid
S1	C1
S1	C2
S2	C2

<u>Course</u>	
Cid	Inst
C1	Korth X
C2	Navathe X
C3	Ullman X

Condition taken is

$\text{inst} = \text{Korth} \wedge$
 $\text{inst} = \text{Navathe}$

↓

This will never be
true (as inst. can't be
same both
Korth & Navathe
together)

Result of this is empty

⑧ Retrieve Sid's where every course is taught by Korth and Navathe.

Cid	Inst
C1	Korth
C2	Korth
C3	Navathe
C4	Navathe

<u>Enroll</u>	
Sid	Cid
S1	C1
S1	C2
S2	C3
S2	C4
S3	C1
S3	C2
S3	C3
S3	C4
S4	C1
S4	C3

O/p is
S3

⇒

$$\pi_{\text{Sid}} \text{ Cid } (\text{Enroll}) / \pi_{\text{cid}} (\sigma_{\text{inst} = \text{KORTH}} \text{course})$$

$\text{inst} = \text{KORTH} \checkmark$
 $\text{inst} = \text{Navathe}$

C1
C2
C3
C4

OR

$$\{ \begin{array}{l} S_1 \\ S_3 \end{array} \} \setminus_{\text{Sid Cid}} (\text{enroll}) / \setminus_{\text{Cid}} (\sigma(\text{course})) \} \cap \{ \begin{array}{l} S_1 \\ S_2 \end{array} \}$$

Inst = Korth

common \leftarrow

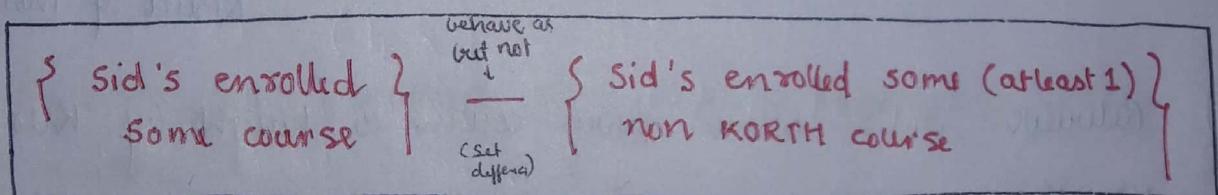
$$\{ \begin{array}{l} S_2 \\ S_3 \end{array} \} \setminus_{\text{Sid Cid}} (\text{enroll}) / \setminus_{\text{Cid}} (\sigma(\text{course})) \}$$

Inst = Navathy

- ⑧ Retrieve sid's enrolled only Korth courses
- \downarrow
set diff
- \downarrow
It should be only student
of KORTH and nobody else

Enroll		Course	
Sid	Cid	Cid	Inst
S1	C1	C1	Korth
- S1	C2	C2	Navathy
- S2	C2	C3	Ullman
S3	C1		
- S3	C3		
S4	C1		

\Rightarrow O/p will be
S4



$S_4 \Leftarrow \{ \begin{array}{l} S_1 \\ S_2 \\ S_3 \\ S_4 \end{array} \} - \{ \begin{array}{l} S_1 \\ S_2 \\ S_3 \end{array} \}$

$$\setminus_{\text{Sid}} (\text{enroll}) - \setminus_{\text{Sid}} (\text{enroll} \bowtie \sigma(\text{course}))$$

Inst \neq Korth

- ⑨ Sid's whose age only 20
- \hookrightarrow does not relate to set diff

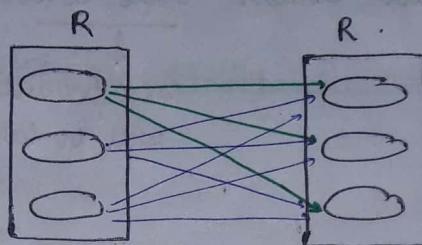
$$\setminus_{\text{Sid}} (\sigma_{\text{age}=20} (\text{stud}))$$

(T) ~~Set~~

Self Cross Product -

$\sigma_c(R \times R)$: To compare every possible 2 records of R.

$\sigma_c(R \times R \times R)$: To compare every possible 3 records of R.



Ques: ① Retrieve Sid's who enrolled atleast 2 courses.

Here we need to compare every possible 2-2 records

$\overset{=}{\text{enroll}}$

	Sid	Cid	fuv		Sid	Cid	fuv
t ₁	S ₁	C ₁	→		S ₁	C ₁	
t ₂	S ₁	C ₂	→		S ₁	C ₂	
	S ₁	C ₃	→		S ₁	C ₃	
	S ₃	C ₁	→		S ₃	C ₁	
	S ₃	C ₂	→		S ₃	C ₂	
x	S ₂	C ₁			S ₂	C ₃	

↓
for comparing using cross product.

$$\left(t_1 \cdot \text{Sid} = t_2 \cdot \text{Sid} \right) \wedge \left(t_1 \cdot \text{Cid} \neq t_2 \cdot \text{Cid} \right) \Rightarrow \text{ensures that } t_1 \cdot \text{Sid} \text{ enrolled atleast 2 courses}$$

$$\pi_{\text{Enroll} \cdot \text{Sid}} \left(\sigma_{\begin{array}{l} \text{Enroll} \cdot \text{Sid} = T \cdot \text{Sid} \\ \wedge \text{Enroll} \cdot \text{cid} \neq T \cdot \text{cid} \end{array}} (\text{Enroll} \times \rho(\text{Enroll})) \right)$$

(OR)

$$\pi_{\text{Sid}} \left(\text{Enroll} \bowtie_{\begin{array}{l} \text{Sid} = S \\ \wedge \text{cid} = C \end{array}} \rho(\text{Enroll}) \right)$$

② Retrieve Sid's enrolled atleast three courses.

\downarrow
cross product of 3 instances
will be taken.

Firstly Rename it,

$$\rho(T_1, \text{Enroll})$$

$$\rho(T_2, \text{Enroll})$$

$$\rho(T_3, \text{Enroll})$$

$$\sigma(T_1 \times T_2 \times T_3)$$

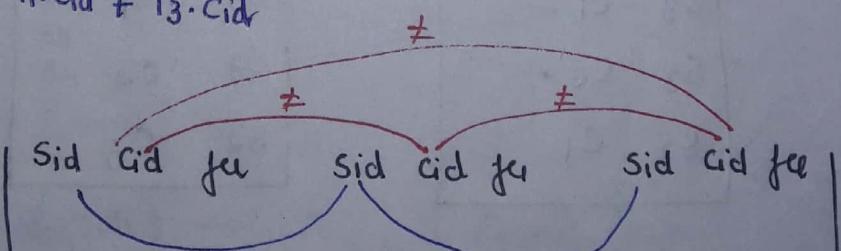
$$T_1 \cdot \text{Sid} = T_2 \cdot \text{Sid} \wedge$$

$$T_2 \cdot \text{Sid} = T_3 \cdot \text{Sid} \wedge$$

$$T_1 \cdot \text{cid} \neq T_2 \cdot \text{cid} \wedge T_2 \cdot \text{cid} \neq T_3 \cdot \text{cid} \wedge$$

$$T_1 \cdot \text{cid} \neq T_3 \cdot \text{cid}$$

Taken because
equality does
not follow
equality rule



③ Retrieve Sid's enrolled only one course.

↓

(Sid's enrolled at least one course but not
enrolled at least 2 courses)

⇒ All enrolled — Enrolled in at least 2 courses

$\pi_{\text{sid}}(\text{enroll}) - \pi_{\text{sid}}(\text{Enroll} \bowtie_{\substack{\text{sid}=\text{s} \\ \wedge \text{cid} \neq \text{c}}} \rho_{\substack{\text{s,c,f}}}(\text{enroll}))$

④ Retrieve Sid's enrolled exactly 2 courses

(Sid's enrolled
at least 2 courses) — (Sid's enrolled
at least 3 courses)

⑤ Sid's enrolled at most one course.
(zero or 1)

⇒ Sid of all student — Enrolled in at least 2 courses.

$\pi_{\text{sid}}(\text{stud}) - \pi_{\text{sid}}(\text{Enroll} \bowtie_{\substack{\text{sid}=\text{s} \\ \wedge \text{cid} \neq \text{c}}} \rho_{\substack{\text{s,c,f}}}(\text{enroll}))$

⑥ Sid's enrolled atmost 2 courses

↓
enrolled in 0, 1 or 2 courses

All student Sid's — Sid's enrolled atleast 3 courses

$\pi_{\text{Sid}}(\text{stud}) \rightarrow \pi_{\text{Sid}}(\text{atleast } 3)$

⑦ Retrieve Sid's who paid max fees.

Enroll		
Sid	Cid	fees
S ₁		80
S ₁		70
S ₁		50
S ₃		50
S ₃		60
S ₂		80

Enroll		
Sid	Cid	fees
S ₁		80
S ₁		70
S ₁		50
S ₃		50
S ₃		60
S ₂		80

$\pi_{\text{Sid}}(\text{Enroll}) - \pi_{\text{Sid}}(\text{Enroll} \bowtie_{\text{fee} < f} \rho(\text{Enroll}))$

$\begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix} \leftarrow \begin{matrix} S_1 \\ S_2 \\ S_3 \end{matrix}$

↑
Error

(as S₁ also paid
max fees)

o/p should be
S₁ and S₂

(this is because
S₁ is not
unique)

\Rightarrow So we will project Sid,Cid and
these will be subtracted from all
Sid,Cid record.

$$\pi_{\text{Sid}} \left(\pi_{\text{Sid Cid}} (\text{Enroll}) - \pi_{\text{Sid Cid}} (\text{Enroll} \bowtie_{\text{Fee} < f} \rho_{\text{s,c,f}} (\text{Enroll})) \right)$$

$\pi_{\text{Sid}} \left\{ \begin{array}{l} S_1, C_1 \\ S_1, C_2 \\ S_1, C_3 \\ S_3, C_1 \\ S_3, C_2 \\ S_2, C_1 \end{array} \right\} \Leftrightarrow \left\{ \begin{array}{l} S_1, C_1 \\ S_1, C_2 \\ S_1, C_3 \\ S_3, C_1 \\ S_3, C_2 \\ S_2, C_1 \end{array} \right\} - \left\{ \begin{array}{l} S_1, C_2 \\ S_1, C_3 \\ S_3, C_1 \\ S_3, C_2 \end{array} \right\}$
 \Downarrow
 S_1
 S_2 (required ans)

⑧ Retrieve Sid's who paid maximum fee for each course.

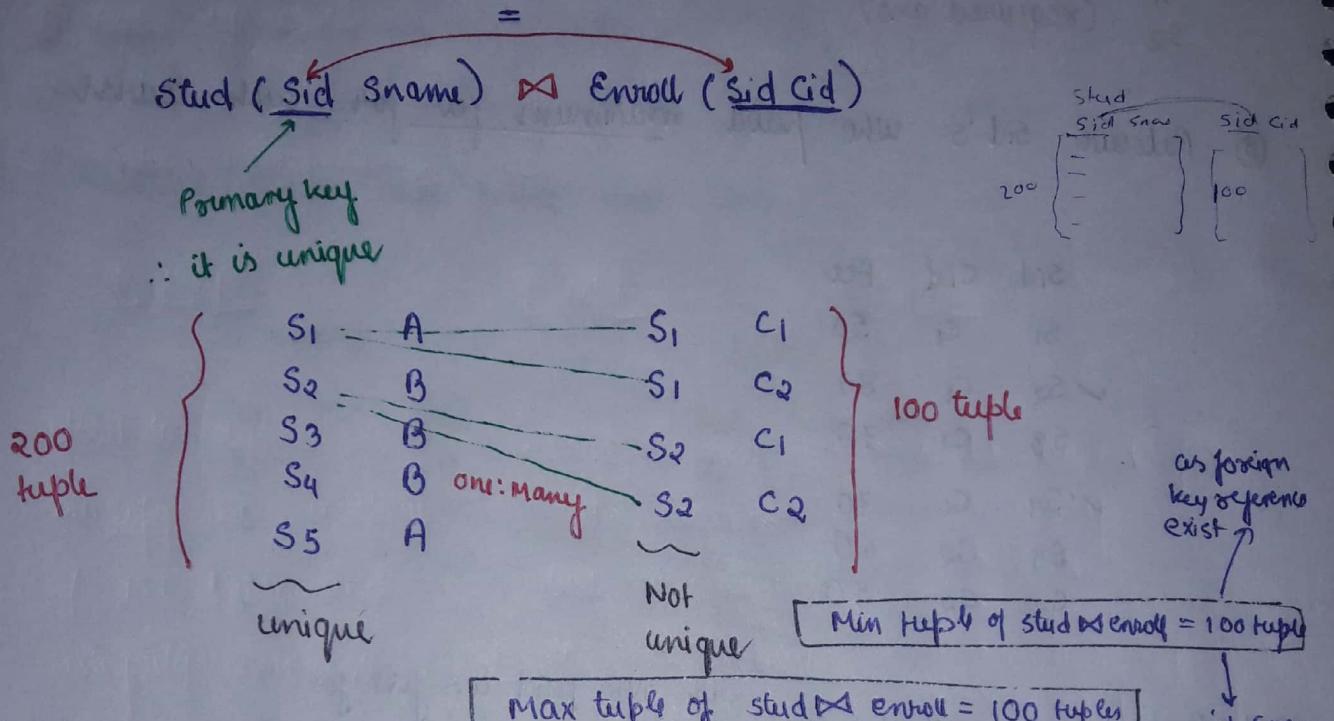
Sid	Cid	Fee
S1	C1	50
✓ S2	C1	80
S3	C1	70
✓ S4	C2	70
S5	C2	60
S2	C2	50

Sid's paid
fee \geq every
stud for same
course
 \Rightarrow All Stud - Sid who paid less fees for
same course

$$\pi_{\text{Sid}} \left(\pi_{\text{Sid Cid}} (\text{Enroll}) - \pi_{\text{Sid Cid}} (\text{Enroll} \bowtie_{\substack{\text{Fee} < f \\ \text{Cid} = c}} \rho_{\text{s,c,f}} (\text{Enroll})) \right)$$

Ques: Stud (sid sname) with 200 tuples and
Enroll (sid cid) with 100 tuples. How many max and
min records in result of stud ⋈ enroll?

- a) (200, 100)
 b) (200, 0)
 ✓ c) (100, 100)
 d) (20,000, 0)



Note →

In one to many mapping b/w A and B sets

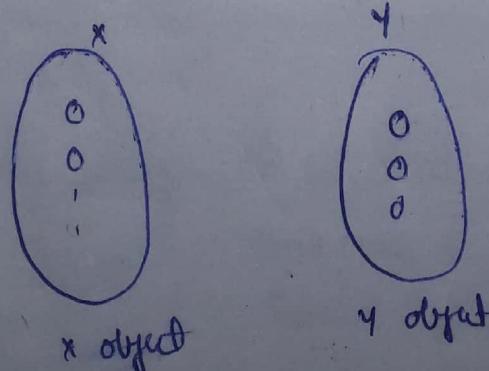
① Max no. of pair at any time = y pairs

② M:1, Max no. of pair = x pairs

③ 1:1, Max no. of pair = min(x,y) pairs

④ M:N, Max no. of pair = x · y pairs

↓
 if Foreign key does not exist
 $\Rightarrow 0$
 (from 0 to 100)



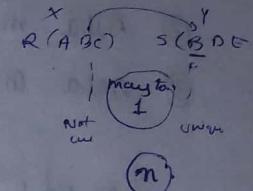
Ques:

$R(\underline{A} \ B \ C)$ with n tuples and
 $S(\underline{B} \ D \ E)$ with m tuples. What is

(many:1)

Max tuples in $R \bowtie S$: n tuples

Min tuples in $R \bowtie S$: n tuples with foreign key
(Assume there is no NULL value)
0-tuples with no foreign key



Ques:

$R(\underline{A} \ B \ C)$ with n tuple and
 $S(\underline{A} \ D \ E)$ with m tuple.

(1:1)

Max tuple in $R \bowtie S$: $\min(n, m)$ tuple

Min tuple in $R \bowtie S$: $\min(n, m)$ tuple with FK
0 without FK

Ques:

$R(\underline{A} \ \underline{B} \ C)$ with n tuples

$S(\underline{D} \ \underline{B} \ E)$ with m tuples

(many:many)

Max tuple in $R \bowtie S$: $n \times m$ tuple.

Min tuple in $R \bowtie S$: 0 (because no 'B' is candidate key)

Ques:

$R(\underline{A} \ B \ C)$ $S(\underline{B} \ D \ F)$ with n and m tuples and

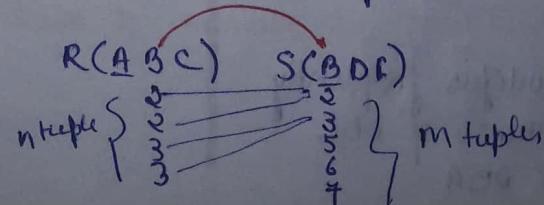
$\{ A \rightarrow B, B \rightarrow C, B \rightarrow D, D \rightarrow F \}$ are FD's.

we could find primary key

$A^* = ABCDF$

$\therefore A$ as primary key
and B for 2nd

Max tuple in $R \bowtie S$: n -tuples



SQL

(Structured Query Language)

Sublanguages of SQL -

- ① Data definition lang (DDL)
 - ② Data manipulation lang (DML)
 - ③ Data control lang (DCL)

<u>DDL</u>	<u>DML</u>	<u>DCL</u>
<ul style="list-style-type: none"> Used to <u>define</u> or <u>modify</u> the structure of DB table and integrity constraints (Primary key, alternative key, f K, check, etc) <u>DDL commands</u> - <ul style="list-style-type: none"> → CREATE TABLE <tname>... → DROP TABLE <tname>... → ALTER TABLE <tname> <ul style="list-style-type: none"> • Add / remove attribute • Modify integrity constraint → CREATE INDEX ... , etc. NOT modified frequently and control of DDL under DBA 	<ul style="list-style-type: none"> Used to <u>modify</u> data records but not allowed to modify structure of tables (and) Used to <u>access data</u> from DB table <u>DML Commands</u> - <ul style="list-style-type: none"> { INSERT INTO <tname> DELETE FROM <tname> UPDATE <tname> SET ... { SELECT * from table where Condition; 	<ul style="list-style-type: none"> Data control for <u>consistency</u> <ul style="list-style-type: none"> → Lock() → Shared() → Exclusive lock() → Commit → Rollback Data control for <u>security</u> <ul style="list-style-type: none"> → grant → revoke <p>} Used for transaction and concurrency control</p>

#

SQL query (vs) R.A expression

① SQL : $\xrightarrow{\text{selection operator } (\sigma)}$
 $\textcircled{3} \text{ SELECT DISTINCT A}_1, A_2, \dots, A_n$

Cross Product (\times) $\leftarrow \textcircled{1} \text{ FROM R}_1, R_2, \dots, R_n$

Selection operator (σ_p) $\textcircled{2} \text{ WHERE P;}$

Execution order is ①, ②, ③

\downarrow In R.A form

$$\pi_{A_1, A_2, \dots, A_n} \left(\sigma_p (R_1 \times R_2 \times \dots \times R_n) \right)$$

Example,

- ① Retrieve Sid's enrolled some course taught by KORTH

In R.A \rightarrow

$$\pi_{\text{Sid}} \left(\sigma_{\substack{\text{Enroll.id} = \text{Course.id} \\ \text{Inst} = \text{KORTH}}} (\text{Enroll} \times \text{Course}) \right)$$

In SQL \rightarrow

SELECT DISTINCT Sid

FROM Enroll AS T₁, Course AS T₂

WHERE T₁.Cid = T₂.Cid and

T₂.Inst = "KORTH";

Note -

DISTINCT keyword is used to remove the duplicate records.

Basic SQL clauses —

⑤ $\text{SELECT } [\text{DISTINCT}] \ A_1, A_2, \dots, A_n$ ⑥

① $\text{FROM } R_1, R_2, R_3, \dots, R_m$

② $[\text{WHERE } P]$

③ $[\text{GROUP BY } (\text{Attribute})]$

④ $[\text{HAVING condition}]$

⑦ $[\text{ORDER BY } (\text{Attributes}) \ [\text{DESC}]]$

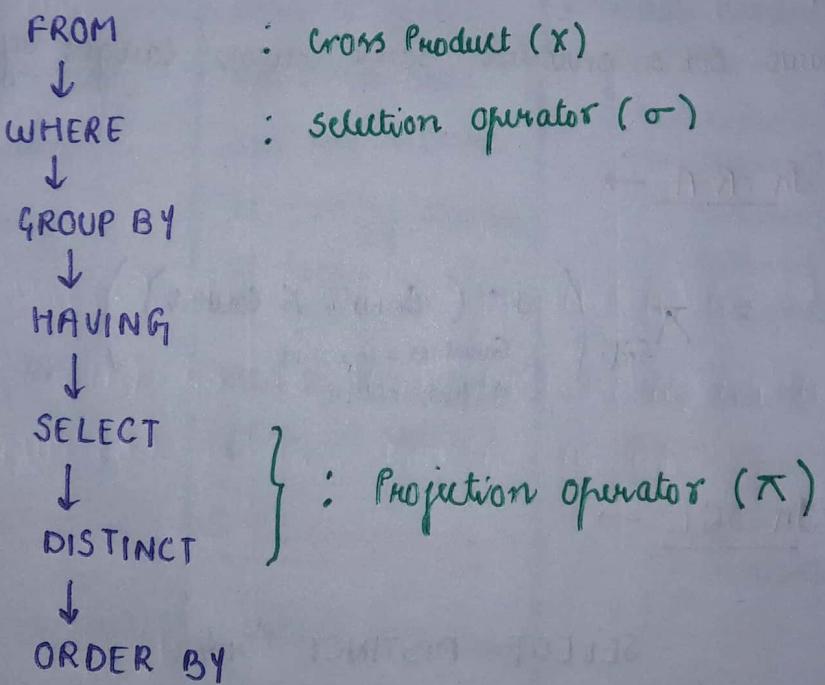
* By default
ORDER BY
is Ascending
ord

* Bracket denote optional part

* Execution flow —

① \rightarrow ② \rightarrow ③ \rightarrow ④ \rightarrow ⑤ \rightarrow ⑥ \rightarrow ⑦

i.e



Aggregate functions of SQL —

- COUNT()
 - SUM()
 - AVG()
 - MIN()
 - MAX()
- }
- 'NULL' value is not considered
for aggregation.
↓
discard NULL value

→ Aggregate function computes aggregation of 'NON NULL' values.

Eg — Create table R

```
( A int  
B int  
);
```

R	Row-id	A	B
	1	6	8
	2	8	7
	3	NULL	10
	4	8	NULL
	5	5	5
	6	NULL	NULL
	7	5	6

- ① One extra attribute is defined by.
- ② It is defined by default for every RDBMS table.
- ③ User cannot modify or manipulate it.
- ④ It is unique it default Primary key.

Ques: Write SQL query to ^{delete} duplicate record?

FAQ
Interview

Count —

count(*) : Counts no. of records of table

count(A) : Counts no. of non NULL value of attribute A

COUNT(DISTINCT A) : Counts no. of distinct non null values of attr 'A'.

Eg -

```
SELECT COUNT(*) AS A1 , COUNT(A) AS A2 ,  
COUNT (DISTINCT A) AS A3  
FROM R;
```

↓ output

for table →

Rowid	A	B
1	6	8
2	8	7
3	NULL	10
4	8	NULL
5	5	5
6	NULL	NULL
7	5	6

⇒

A1	A2	A3
7	5	3

Ques: SELECT SUM(A) AS A1 , SUM (DISTINCT A) A2 ,
AVG(A) AS A3 , AVG (DISTINCT A) AS A4 ,
MIN (A) AS A5 , MAX (A) AS A6 .

FROM R;

↓ o/p

A1	A2	A3	A4	A5	A6
32	19	6.4	6.33	5	8

$$\frac{\text{sum}(\text{colP}(A))}{5}$$

$$\frac{\text{dist}(\text{sum})}{\text{count}(A)}$$

Ques: Select Avg (A), B
From R;

Soluⁿ:

Avg (A)	B
Single value → 6.4	□
	□
	□

} Many values wrong X → gives error

∴ If aggregate funcⁿ is used in SELECT clause, then any other attribute is not allowed to SELECT by if GROUP BY clause does not exist

Ques: SELECT A
FROM R

WHERE A = MAX (A);

↓
defined
for condition
for each
record

∴ does not give 8 as answer

It computes as —

$$A = \text{MAX} (A)$$

$$6 = \text{Max} (6)$$

$$8 = \text{Max} (8)$$

$$8 = \text{Max} (8)$$

⋮

∴ O/p →

6
8
8
5
5

Aggregation is computed
for each record

↓
∴ No use of
using it
(but does not
quer error)

Eq -

Select sid
from stud
where age = max(age); X Avoid using
Aggregate in 'where clause'

↓
does not give error but
computes each record

∴ returns sid of each
record.

7/11/17
GROUP BY clause -

Used to group data records based on specified
attribute values.

R	A	B	C
	a ₁	b ₁	80
	Null	b ₂	70
	a ₃	b ₃	50
	Null	b ₂	90
	a ₁	b ₁	60
	a ₃	b ₃	60
	a ₁	b ₅	Null

Group by A →

R	A	B	C
	a ₁	b ₁	80
	a ₁	b ₁	60
	a ₁	b ₅	Null
	Null	b ₂	70
	Null	b ₂	90
	a ₃	b ₃	50
	a ₃	b ₃	60

" Group by (AB) →
(more than 1 attr is used)
Used for OLAP application
(Data Mining
Queries)

R	A	B	C
	a ₁	b ₁	80
	a ₁	b ₁	60
	a ₁	b ₅	Null
	Null	b ₂	70
	Null	b ₂	90
	a ₃	b ₃	50
	a ₃	b ₃	60

SQL Standard for group by -

Every attribute of group by must be in select clause and is allowed to use aggregation funcn in SELECT clause.

↓
(Aggregation of SELECT clause computes aggregation of each group)

But it is not allowed to select attributes which are not in group.

Eg -
1. Select A , Avg(c)
2. From R
3. Group by (A);

✓ Aggregation allowed as group by (A) → so A can be selected

Soluⁿ.

A	Avg (c)
a1	70
NULL	80
a3	55

- for last table

Ques:

Select A

From R

Group by (A);

A
a1
Null
a3

Ques: select B
from R
group by A; Error
 ↑
 It is group by A
 but is selecting B
 ∴ Error.

Ques: select A, B
from R
group by A;
 → B is selected
 and
 it is group by A
 ∴ Not allowed
 Error.

Ques: select Avg(c)
from R
group by A;
 In SQL standard -
 It is not allowed as (data can't be
 grouped by attribute i.e. 'A' placed in tabular
 format)
 is not in select clause. ∴ Error

A	B	values
a1	0	
b1	0	
c1	0	
d1	0	

But, in Oracle SQL -

It is allowed.

Avg(c)
70
80
55

But we don't know
which is avg of which
groups.

same functionality when implemented in SQL standard can be written as —

Select C₁

From (Select A , Avg(c) as C₁

From R

Group by A)



Inner query o/p is

A	Avg (c)
a ₁	80
a ₁	70
a ₁	70
Null	80
Null	80
a ₃	55
a ₃	55

after outer
query

C ₁
70
80
55

Ans: Select A
From R
Group by (A,B)

Error as Group by attribute B
is not in the select

In ~~SQL~~ Oracle SQL —

Allowed ✓

A
a ₁
a ₁
Null
a ₃

We cannot say
which value is
group by A and which
is group by B.

Ques: which is not allowed as per SQL standard.

a) Set A , Avg (B)
From R ; X

b) Set Avg (c)
From R X
Group by A;

c) Select A,
From R X
Group by (A,B);

d) Select A , count (*).
From R ✓
Group by A

option a), b), c) all are wrong . but

⑥ and ⑦ are accepted
by Oracle SQL i.e some
compiler (but not a.)

Ques: Which is not allowed.

a) Select A
From R
Group by (A,B);
group by B
↑
but B not projected

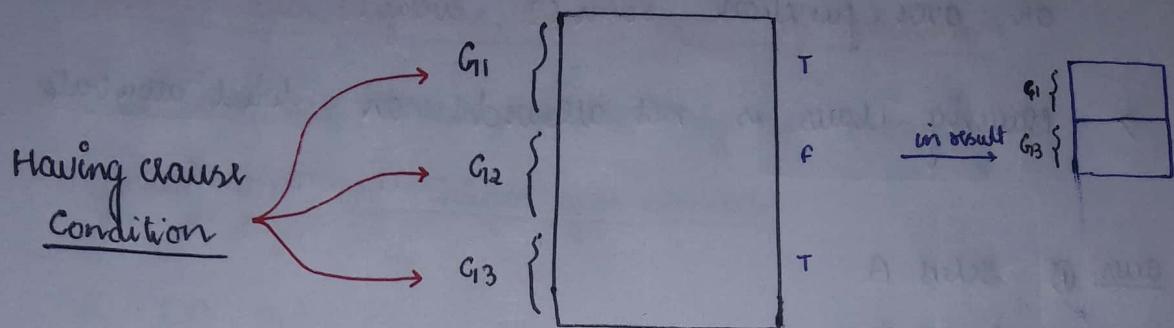
b) Select A
From R
Group by A; ✓

c) Select A , count (*)
From R
Group by A; ✓

d) Select A,B
From R
Group by (A,B) ✓

Having Clause -

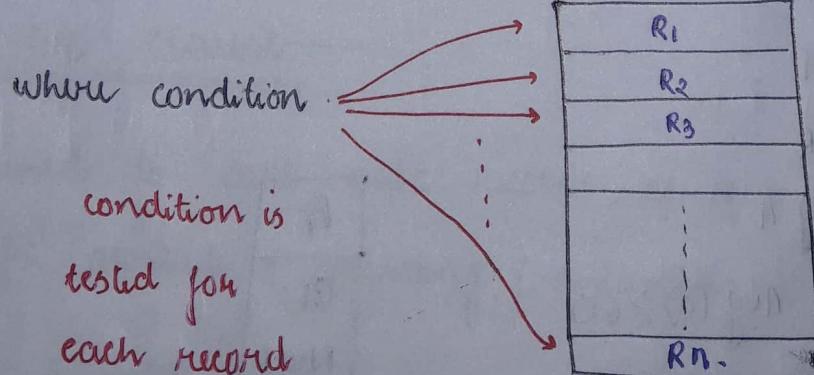
- It is used to select group based on some specified condition.
- Having clause condition is tested for each group.



condition is
tested for each group

(if any group fails the condition, then
that group is not in result)

In 'where' clause



SQL standard for Having clause

- Having clause is allowed to use only if Group by clause exist.
- Condition of having clause must be over aggregation or over function sum(), every()
- Having clause is not allowed on direct attribute

Ques: ① Select A

From R

Group by A

Having C > 70;



direct attribute

C > 70 cannot be applied as condition of group.

∴ Error

② Select A

1. From R

2. Group by A

3. Having Avg(c) > 60

↓
aggregation

is allowed

A
a1
NULL

③ Select A

1. From R
2. Group by A
3. Having some (c) ≥ 70;

Allowed

A
a1
NULL

Note -

Having clause can use direct group by attribute

It can use direct attribute only if group by is based on one integer type attribute

R (A B)

{	2
	2
✓ {	4
	4
	4
✓ {	8
	8

Select A
From R
Group by A
Having A > 3

o/p →

A
4
8

Allowed

Order by clause —

It is used to sort data records of query result, based on specific attribute (ascending / descending order)

Eg -

1. Select X, Y, Z
2. From —
3. Where —
4. Group by —
5. Having —
6. Order by (X, Y);

→ ordering can be done base on 1 or more attribute

X	Y	Z
2	6	10
2	8	8
3	5	0
3	4	1
NULL	6	4
NULL	5	3

O/P for
Order by (X, Y)

X	Y	Z
2	6	10
2	8	8
3	4	1
3	5	0
NULL	5	3
NULL	6	4

NULL values will
always be at the last

Nested Query —

- ① Nested Query (without Correlation)
- ② Co-related Nested Query

→ Nested Query :

Select Attr

From Tables

Where Condition

group by Attr

Having Condition

Order by attr

(Inner Query)

← Inner query
result treated
as
table / cond

for outer query

- In nested query without correlation,
 - inner query is independent (\therefore executed 1st)
 - outer query uses the result of inner query

Eg. -

```

Select T1.Sid
From stud T1
Where marks = (
    Select max(marks)
    From stud T2 );
  
```

result of it is
80 (scalar value)
① Inner query / Bottom
② Outer query / Top

- Execution flow is from Bottom to Top.
- Inner query executes only once.

\rightarrow Correlated Nested Queries:

- Inner query is not independent.
- Inner query uses the attributes defined in outer query.
- Outer query uses the result of inner query.
- WHERE clause and HAVING clause of outer query is used to correlate with inner query.

Eg - select A

① FROM R

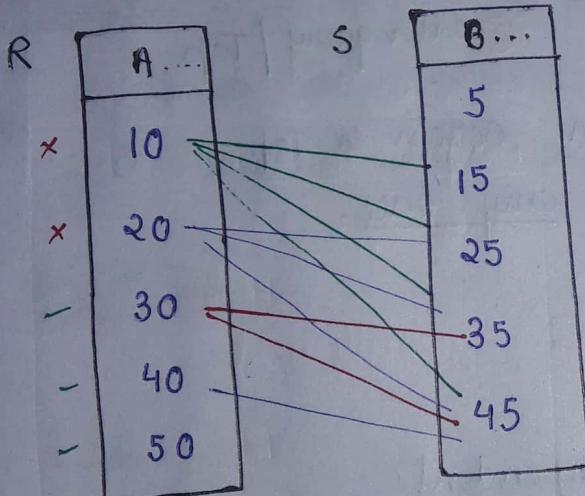
② WHERE (

select count (*)
③ FROM S
where $(R.A) \leq (S.B) \leq 2$

inner query

To compute inner query (It's result is compared with 2)
attr. of outer query is required

if its less, then outer where clause is True
else False



Result of inner query	Where cond
4	F
3	F
2	T
1	T
0	T



O/P will be

A
30
40
50

$$\text{No. of inner query execution} = \frac{\text{No. of record in outer query}}{= 5}$$

- If co-relation is present in where clause, then for each record of outer query, from clause computes inner query.
- Execution flow is (Top - Bottom - Top) in loop.

Eg - Select A

① From R ~~a₁~~

② Group by A

③ having Avg (C) > (Select Avg (c))

75 55

From S

where $S.A = \underbrace{R.A}_{\substack{a_1 a_3}};$

group name
of relation R

solutn:

R	A	C
X	a ₁	40
	a ₁	70
✓	a ₃	80
	a ₃	90

S	A	C
a ₁	a ₁	80
	a ₁	70
a ₃	a ₃	60
	a ₃	50

- If co-relation is in having clause, then for each group of outer query, inner query recomputes.

$$\text{No. of inner query execution} = \text{No. of groups in order query}$$

Eg-

```
Select Sid  
From Stud
```

where marks = (

Select Marks
From Stud
where branch = CS

);

Inner query

The result of this inner query is set of marks

i.e. not a single scalar value

∴ Direct comparison not allowed

"Error"

Stud

Sid	Branch	marks
S ₁	CS	70
S ₂	IT	80
S ₃	CS	90
S ₄	IT	95
S ₅	CS	85

Result of
inner
query

Marks
70
90
85

Now here direct
comparison is not
possible

i.e.

$$S_1 \text{ CS } 70 = \{ 70, 90, 85 \}$$

set of values

To avoid this problem, SQL supports

"SQL funcn"

SQL funcⁿ used in Nested Queries —

- ① IN / NOT IN
- ② ANY
- ③ ALL
- ④ EXISTS / NOT EXISTS → Best for co-related Nested Query.

Best suited for nested queries with no correlation

These SQL funcⁿ are used to compare set of values of inner query result with records of value of outer query in where / having clause

* So, for last ques. instead of '=' we can use 'IN'.

Note- Above funcⁿ are not used when inner query results in single scalar value.

① IN — (Membership Test)

$X \text{ IN } [\text{Set of Y value of inner query result}]$



Returns True if x value is member of set of Y value

Eg - $X \text{ IN } \{2, 3, 5, 6\}$



True if $X \in \text{set}$

$X \text{ IN } \{\text{Empty Set}\} : \text{False}$
 $X \text{ NOT IN } \{\text{Empty Set}\} : \text{True}$

→ IN can be used in place of Equi Join i.e
(works similar to Equi Join)

Eg —

①

 $R(A|B) \quad S(C|D)$

$$\pi_{A,B} (R \bowtie S) \underset{R \cdot B = S \cdot C}{\cong} \begin{array}{l} \text{Select *} \\ \text{From } R \\ \text{where } B \text{ IN (select } C \\ \text{from } S) \\ \text{works same} \\ \text{as equi Join} \\ (\text{Equal to atleast one value of } C \\ \text{or some value or any.}) \end{array}$$

② $T_1(A|B|C) \quad T_2(D|E|F)$

$$\pi_{ABC} (T_1 \bowtie T_2) \underset{\begin{array}{l} T_1 \cdot B = T_2 \cdot D \\ \wedge T_1 \cdot C = T_2 \cdot E \end{array}}{\cong} \begin{array}{l} \text{Select *} \\ \text{FROM } T_1 \\ \text{where } (B,C) \text{ IN (select } D,E \\ \text{from } T_2) \\ \text{comparison b/w equal no.} \\ \text{of columns must be there} \\ \text{b/w inner and outer query.} \end{array}$$

③ $R(A|B) \quad S(C|D)$

$$\pi_{A,B} (R \bowtie S) \underset{R \cdot B > S \cdot C}{\cong}$$

Not possible to express
using IN func'n

② ANY and ALL

→ X operator ANY [set Y values]

\uparrow
 $<$,
 $>$,
 \leq ,
 \geq ,
 $=$,
 \neq

TRUE if X satisfies the given comparison in operation with some value of set Y
 ↓
 (at least 1)

\neq : Not equal

$X \text{ op ANY } \{ \text{empty set} \} : \text{false}$

→ X operator ALL [set of Y values]

True if X satisfies the given comparison (operator) with every value of set Y .

Eg - Select *

FROM R

WHERE A IN (Select B
From S)

$X \text{ op ALL } \{ \text{empty set} \} : \text{True}$.

Here = ANY can be used.

R	A
x	10
-	30
-	40

S	B
	20
	30
	40

A
30
40

Here, IN can be replaced by (= ANY)

IN $\approx (= \text{ANY})$

Rewriting it using = ANY

Select *

FROM R

where A = ANY (select B
From S)

- But, 2 field comparison of IN cannot be replaced by = ANY
i.e IN support more than one field comparison.

Eq - Select *

FROM T1

where (B,C) IN (select D,E
From T2)

cannot be done by = ANY

- Also, = ANY support many more operations like <, >,
<=, >=, etc that are not supported by IN .

<> ALL can be replaced by NOT IN func

Ques:

R	A	B
	4	2
	6	4
	7	5

S	C	D
	4	3
	7	6
	9	8

What is the no. of tuples in the result of
SQL query?

① Select *
from R
where $B > \text{ANY}$ (select c
from S
where $D > 10$);

a) 0

b) 1

c) 2

d) 3

$\begin{matrix} 8 \\ 2 \\ 4 \\ 5 \end{matrix} > \text{Any}$

$\therefore 0 \text{ records}$

② Select *
from R

Where $B > \text{ALL}$

(select c
from S
where $D > 10$);

$\boxed{\text{ANY} = \exists}$

$\boxed{\text{ALL} = \forall}$

a) 0

c) 2

b) 1

$\checkmark d) 3$

$\begin{matrix} 2 \\ 4 \\ 5 \end{matrix} > \text{All}$

$\therefore 0 \text{ records}$

$\exists x(P(x)) \rightarrow$ True for some x , $P(x)$ is true
False for every x , $P(x)$ is false.

$\forall x(P(x)) \rightarrow$ True for every x , $P(x)$ is true
False for some x , $P(x)$ is false.

Eg - $x = \{5, 10, 15\}$

$\exists x(x > 10) = T$

$\exists x(x > 20) = F$

$\forall x(x > 2) = T$

$\forall x(x > 5) = F$

Eg - $x = \{ \}$

$\exists x(x > 10) = \text{False}$

$\forall x(x > 10) = \text{True}$

③ EXISTS

function returns True if inner query result is not empty, otherwise returns false

$R(A \dots)$	$S(B \dots)$
x 20	25
- 30	35
- 40	45

Select A

From R

Where EXISTS (

Select *
 From S
 Where $R.A > S.B$

)

True: if some record of S, it satisfies $R.A > S.B$

F
T
T

Empty ... return false \Rightarrow record not in O/P

20
30
40

O/P \rightarrow

A
30
40

Ques:

R

A	B
4	2
6	4
7	5

S

C	D
4	3
7	6
9	8

Find No. of Tuples in result of -

①

Select *

From R

Where EXISTS (

Select C
 From S
 Where D > 10 and R.B < S.C;

)

$\therefore 0$ tuples

② Select *

From R

Where EXISTS (

Select count(*)

From S

Where D > 10 and R.B < S.C);

returns 0 i.e. {0}

Since it has
a value
∴ Not empty
⇒ True
(true for all)

∴ 3 records in the o/p.

Ques: R(A...) S(B...)

Retrive 'A' values of R which are more than some 'B' of
relation S.

In R.A —

$\pi_A (R \bowtie S)$
 $R.A > S.B$

In SQL —

Select distinct A
From R, S
Where R.A > S.B

(OR)

We can't write it using ANY.

Select distinct A

Query: Emp (Eid, dno, sal, gender)

① Retrieve Eid's who get highest salary

Select Eid
From Emp

All Eid's

~~where~~ EXCEPT

Select T₁. Eid
From Emp T₁, Emp T₂
Where T₁.sal < T₂.sal

Eid's who do not
get highest salary

EXCEPT
keyword is
used for
SET difference

In R.A —

$\pi_{Eid}(\text{Emp}) - \pi_{Eid}(\text{Emp} \bowtie_{\substack{\text{sal} < s \\ \text{I}, \text{D}, \text{S}, \text{G}}} P(\text{EMP}))$

(OR)

Select Eid
From Emp

Where sal = (Select Max(sal)
From Emp);

② Retrieve Eid's who get 2nd highest salary.

In R.A —

$P(\text{Temp}, \pi_{Eid, \text{sal}}(\text{Emp} \bowtie_{\substack{\text{sal} < s \\ \text{I}, \text{D}, \text{S}, \text{G}}} P(\text{EMP})))$

This gives Eid, sal of all employees except who gets highest salary.

$\pi_{Eid}(\text{Temp}) - \pi_{Eid}(\text{Temp} \bowtie_{\substack{\text{sal} < s \\ \text{I}, \text{S}}} P(\text{Temp}))$

In SQL —

(Join Query)

'Temp' used in R.A needed to be computed 3 times, for this SQL supports WITH clause.

WITH Temp (Eid, sal) as

```
(Select Distinct T1.Eid, T1.Sal  
  From Emp T1, Emp T2  
 Where T1.Sal < T2.Sal )
```

if with clause
is not used
then, this
is written 3
times
in place of
Temp .

SELECT Eid

From Temp

EXCEPT

Select T1.Eid

From Temp T1, Temp T2

Where T1.Sal < T2.Sal ;

(OR)

Using Aggregation (Nested Query)

Select Eid

From Emp

Where Sal = (Select max(sal)

From Emp

Where Sal < (Select max(sal)

From Emp))

2nd Max

1st Max

Above is non-correlated query

(Now this if we want kth highest, then it become
① (K+1) levels of nested query
② K+1 instance of Emp table is used -

OR

Emp (Eid sal)

T ₁	Eid	sal
E ₁	80	
E ₂	80	
E ₃	70	
E ₄	70	
E ₅	60	
E ₆	50	

Emp (Eid sal)

T ₂	Eid	sal
E ₁	80	
E ₂	80	
E ₃	70	
E ₄	70	
E ₅	60	
E ₆	50	

For each record of T₁ count no. of distinct records of T₂ whose salary is more than sal of T₁



$$\text{count}(\text{distinct}) = 0$$

T₁.sal is 1st max

$$\text{count}(\text{distinct}) = 1$$

T₁.sal is 2nd max

* For Eid's of all Emp with 1st, 2nd and 3rd max sal.

$$\text{where } T_1.\text{sal} < T_2.\text{sal} \leq 2$$

Select T₁.Eid

From Emp T₁

Where (Select count(Distinct T₂.Sal))

From Emp T₂

Where T₁.sal < T₂.sal) = 1

inner query

for kth highest

- Only 2 level of query
- 2 instances of Emp table

* for Eid who get Top 3 least salary.

Where cond will be changed as

$$T_1.\text{sal} > T_2.\text{sal} \leq 2$$

* for Eid who get 3rd highest

$$T_1.\text{sal} < T_2.\text{sal} = 2$$

* Eid's who get 3rd last salary

where $T_1 \cdot \text{sal} > T_2 \cdot \text{sal} = 2$

Ques: Emp (Eid, dno, sal, gender)

① Retrieve dno which consist of atleast 3 employees.

In R.A —

$\rho(T_1, \text{Emp})$

$\rho(T_2, \text{Emp})$

$\rho(T_3, \text{Emp})$

$\circ (T_1 \times T_2 \times T_3)$

atleast 3

$T_1 \cdot \text{dno} = T_2 \cdot \text{dno} \wedge$

$T_2 \cdot \text{dno} = T_3 \cdot \text{dno} \wedge$

$T_1 \cdot \text{Eid} \neq T_2 \cdot \text{Eid} \wedge$

$T_2 \cdot \text{Eid} \neq T_3 \cdot \text{Eid} \wedge$

$T_1 \cdot \text{Eid} \neq T_3 \cdot \text{Eid};$

In SQL —

select distinct $T_1 \cdot \text{dno}$

from Emp T_1 , Emp T_2 , Emp T_3

where $T_1 \cdot \text{dno} = T_2 \cdot \text{dno}$ and $T_2 \cdot \text{dno} = T_3 \cdot \text{dno}$

and $T_1 \cdot \text{Eid} \neq T_2 \cdot \text{Eid}$ and

$T_2 \cdot \text{Eid} \neq T_3 \cdot \text{Eid}$ and

$T_1 \cdot \text{Eid} \neq T_3 \cdot \text{Eid};$

Ans

(OR) By using GROUP BY —

This is easier and has less computation.

Eid	dno
e ₁	d ₁
e ₂	d ₁
e ₃	d ₁
e ₄	d ₂
e ₅	d ₂
e ₆	d ₃
e ₇	d ₃
e ₈	d ₃
e ₉	d ₃

{ ≥ 3

{ ≥ 3

{ ≥ 3

Select dno

From Emp

Group by dno

having count(*) ≥ 3

Ans

having clause can be replaced by where clause.

(∴ Having clause is not mandatory)

Select dno

From (select dno, count(*)) C

From Emp

Group by dno) temp

where C ≥ 0



Temp

dno	count(*)
d ₁	3
d ₂	2
d ₃	4

The computation by these 2 queries i.e. with having clause and where clause are same

② Retrieve d-no such that average sal of female emp's of department more than avg sal of all male employees of comp

Select dno

1. From Emp
2. Where gender = female
3. Group by dno
4. having $\text{Avg}(\text{sal}) > (\text{Select Avg}(\text{sal})$

Avg sal of all male employe
of all dept.

From Emp
 $\text{Where gender = male);}$

Avg sal
of female employee
of each dno.

Emp

Eid	dno	sal	gender
e1	d1		f
e3	d1		f
e5	d2		f
e7	d2		f

↙ p

d no
d1

OR By cross producting

T₂

dno	Avg of Fem
d1	80
d2	50

dept wise

T₂

Avg of Males
60

Select $T_1.dno$

From (Select dno, Avg(sal) sal
From Emp

Where gender = female

Group by dno) T_1 ,

(Select Avg(sal) sal

From Emp

Where gender = male) T_2

Where $T_1.sal > T_2.sal$;

- ③ Retrieve dno such that avg sal of female emp of each dept more than avg sal of male emp of same dept

co-related query with having clause —

Select dno

From Emp T_1

Where $T_1.gender = \text{female}$

Top ① Group by dno

Top ③ Having Avg(sal) > (Select Avg(sal))

From Emp T_2

② Where $T_2.gender = \text{male}$ and
 $T_2.dno = T_1.dno$;

Bottom

(OR)

Select T₁.dno
 From (select dno, Avg(sal) sal
 From Emp
 Where gender = female
 Group by dno) T₁ ,
 (select dno, Avg(sal) sal
 From Emp
 Group by dno
 Where gender = male) T₂
 Where T₁.sal > T₂.sal and
 T₁.dno = T₂.dno ;

③ Retrieve Eid's whose salary more than $\Delta_c^{\text{salars of some}}$ salary
 Emp of dept 5.

In R.A —

$\pi_{Eid} (\text{Emp} \bowtie_{\substack{\text{Sal} > S \\ \wedge D = "5"} } \rho_{(Eid)} (\text{Emp}))$

In SQL —

directly cross producting

SELECT (distinct) T₁.eid

FROM Emp T₁, Emp T₂

Where T₁.sal > T₂.sal

and T₂.dno = 5 ;

(OR)

OR Using ANY function -

complexity = n

Select Eid

From Emp

Where Sal > ANY (

Select Sal
From Emp
Where dno = 5)

Here, we
can replace

Sal > ANY (sal of dept 5)

↓ equivalent to

Sal > (min sal of
dept 5)

OR

Using co-related query -

Select T₁. Eid

From Emp T₁

Where EXISTS (Select *

From Emp T₂

Where T₂. dno = 5

and T₁. Sal > T₂. Sal);

10 20 30 40 50

n tuples

T ₁	Eid	dno	Sal
X	e ₁	4	10
X	e ₂	5	20
✓	e ₃	4	30
✓	e ₄	5	40
✓	e ₅	3	50

T ₂	Eid	dno	Sal
	e ₁	4	10
	e ₂	5	20
	e ₃	4	30
	e ₄	5	40
	e ₅	3	50



Complexity = O(n²)

at least 1
record of
inner query
should be true
↓
so Exist is True
else False

④ Retrieve Eid's whose sal more than every emp of dept 5

 'ALL' can be used.

In R.A -

$$\pi_{Eid}(\text{Emp}) - \pi_{Eid}(\text{Emp} \bowtie_{\substack{\text{sal} > \\ \text{I}, \text{D}, \text{S}, \text{G}}} p(\text{Emp})) \\ \wedge D = 5$$

In SQL -

using cross product and set difference (JOIN query)

Select Eid from Emp ← All

EXCEPT ← -

Select T₁. Eid
 From Emp T₁, Emp T₂
 Where T₁.sal <= T₂.sal and
 T₂.dno = 5;

All employee whose
 sal is less than
 atleast 1

(OR) Using ALL function - (Nested Query)

Select Eid

From Emp T₁

Where sal > ALL

sal > ALL

50
20
40

(Select sal

From Emp

Where dno = 5);

* Instead of 'ALL', max can also be used

OR

+ using co-related nested query -

Select $T_1.Eid$

from Emp T_1

where **NOT EXISTS** (Select *

From Emp T_2

where $T_2.dno = 5$

$T_2.sal \geq T_1.sal$)

$\neg(T_1.sal \leq \text{some sal of dept}) \equiv T_1.sal \text{ is}$

more than
every sal
of dept = 5

T_1	Eid	dno	sal
x	e1	4	10
x	e2	5	20
x	e3	4	30
x	e4	5	40
✓	e5	3	50

$\exists (T_1.sal \leq T_2.sal)$

$\neg \exists (T_1.sal \leq T_2.sal)$

i.e. at least one $T_1.sal \leq T_2.sal$

i.e.

$T_1.sal > \text{every } T_2.sal$.

Ques: R (A....) S (B....)

- ① Retrieve 'A' value of R which are more than some 'B' value in relation S.

In R.A —

$\pi_A(R \bowtie S)$
R.A > S.B

In SQL —

Join Query:

Select distinct R.A
from R, S
where R.A > S.B;

(OR) Nested Query: Using ANY

Select R.A
from R
where R.A > ANY (select B
from S);

(OR) Co-related Nested query.

Select R.A
from R
where EXISTS (select *
from S
where S.B < R.A);

→ R.A > Some value of B in
relation S

Ans: $R(A \rightarrow) S(B \rightarrow)$

Retrieve 'A' value which are more than every 'B' of S
↓
ALL

in R.A →

$\pi_A(R) - \pi_A(R \bowtie_{R.A \leq S.B} 5)$

in SQL →

Join Query :

```
select A
from R
EXCEPT
select A
from R,S
where R.A <= S.B
```

Nested Query: Using ALL

```
select A
from R
where A > ALL (select B
from S);
```

Co-related Nested Query :

```
Select A
from R
where NOT EXISTS (select *
from S
where R.A <= S.B)
```

SQL queries equal to "/" of RA

Ques: stud (Sid sname age) course (Cid cname ginst) Enroll (Sid Cid fa)

- Retrieve Sid's of student who enrolled every course taught by KORTH.

In R.A :

$\pi_{\text{Sid}, \text{Cid}}(\text{Enroll}) / \pi_{\text{Cid}} (\sigma_{\text{ginst} = \text{KORTH}}(\text{course}))$

In SQL :

co-related query —

Enroll T ₁	
Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₁	C ₃
S ₂	C ₁

course	
Cid	Ginst
C ₁	KORTH
C ₂	KORTH
C ₃	KORTH
C ₄	Navathy

Enroll T ₂	
Sid	Cid
S ₁	C ₁
S ₁	C ₂
S ₁	C ₃
S ₂	C ₁

Retrieve T₁. Sid from Enroll (T₁) only if —

$$\left[\text{All Cid's of } \text{KORTH} \right] - \left[\text{Cid's of Enroll (T₂) which are enrolled by T₁. Sid} \right] = \emptyset \quad \text{Empty}$$

when T₁ = S₁ $\begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} - \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} = \emptyset \quad \checkmark \text{ in result}$

when T₁ = S₂ $\begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} - \begin{bmatrix} C_1 \end{bmatrix} = \begin{bmatrix} C_2 \\ C_3 \end{bmatrix} \quad \text{Not empty} \quad \times$

Select $T_1 \cdot \text{Sid}$

① From Enroll T_1

③ Where NOT EXISTS (

Select Cid
From course
Where ginst = KORTH

EXCEPT

② Select $T_2 \cdot \text{Cid}$
From Enroll T_2

Where $T_2 \cdot \text{Sid} = T_1 \cdot \text{Sid}$)

OR

Not Co-related Query -

$\pi_{\text{Sid}, \text{Cid}}(\text{Enroll} / \pi_{\text{Cid}} (\sigma_{ginst = \text{KORTH}}(\text{course}))$

Above can be expanded as:

$\equiv \pi_{\text{Sid}}(\text{Enroll}) - \pi_{\text{Sid}} (\left[\pi_{\text{Sid}}(\text{Enroll}) \times \pi_{\text{Cid}} (\sigma_{ginst = \text{KORTH}}(\text{course})) - \right.$

$\left. \pi_{\text{Sid}, \text{Cid}}(\text{Enroll}) \right])$

R

Now writing above equⁿ in SQL form.

(select Sid
From Enroll
Except
Select Sid

```

from ( select Enroll.sid , course.cid
      from Enroll , course
      where inst = KORTH
      except
      select sid , cid
      from Enroll
    ) R ;

```

- ② Retrieve Cid's enrolled by every student whose age is more than 20.

All -

Retrieve $T_1 \cdot cid$ from Enroll only if -

$$[\text{All sid's whose}] - [\text{sid of enroll } (T_2) \text{ who}] = \phi \\ \text{age more than 20} \quad \text{are enrolled by } T_1 \cdot cid$$

Now implementing above in SQL

Select $T_1 \cdot cid$

① from Enroll T_1

③ where NOT EXISTS (

Select sid
 from Stud
 where age > 20
 Except
 ② Select $T_2 \cdot sid$
~~Enroll~~
 from Enroll T_2
 where $T_2 \cdot cid = T_1 \cdot cid$);

catalog (Sid Pid)		parts (pid Col)	
S ₁	P ₁	P ₁	Red
S ₁	P ₂	P ₂	Red
S ₁	P ₃	P ₃	Red
S ₂	P ₂	P ₄	Green

Retrieve Sid's who supplied every red part.

~~All Sid~~ (

in SQL,

(All Sid's) - (Sid's whose Pid's are not red).

All - (P₁,
P₂,
P₃ not Red
↓
(S₁ S₂) - (S₂)
① ✓)

Retrieve T₁. Sid from Catalog only if

[all red parts Id's] - [Pid's of catalog (T₂) which are supplied by T₁. Sid] = \emptyset

Select T₁. Sid

from Catalog T₁

Where not exist (

Select Pid

from Part

where color = Red

Except

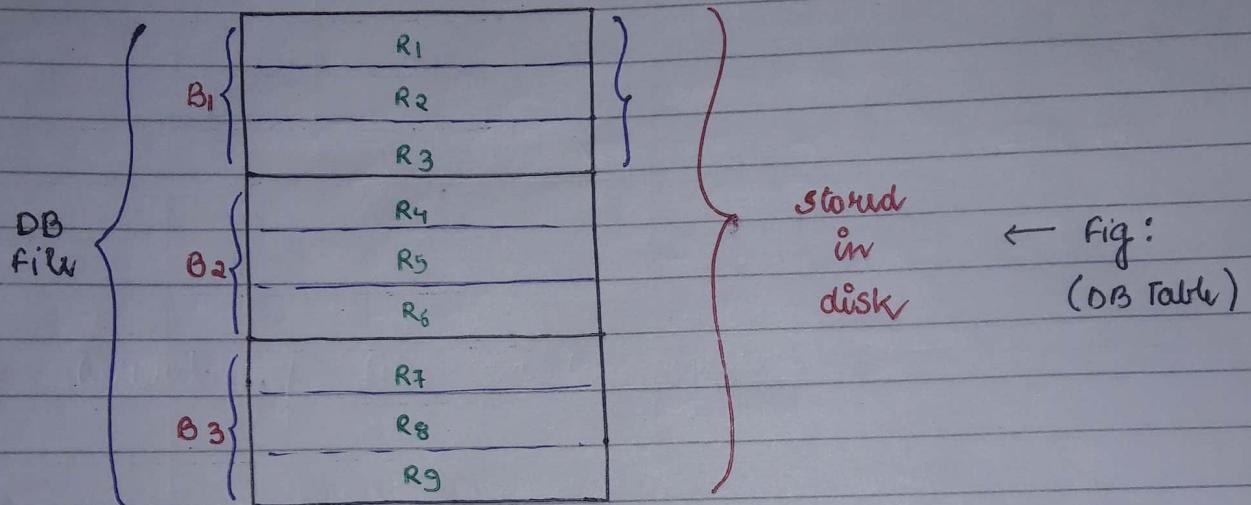
Select T₂. Pid

from Catalog T₂

Where T₂. Sid = T₁. Sid)

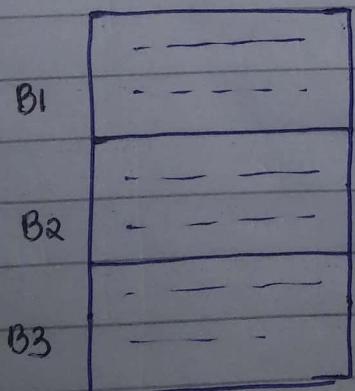
File Organization and Indexing

- Database is collection of files (tables)
- File is a collection of pages (blocks)
- Block is a collection of records

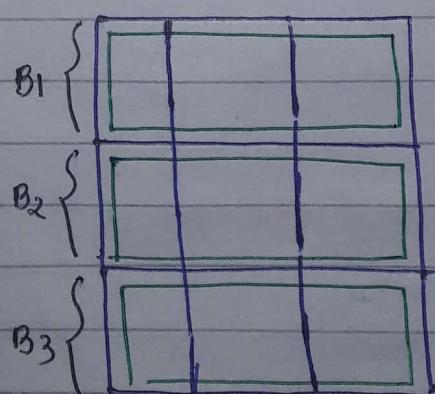


- Data is accessed from disk, block by block.

In operating system
↓
(page file)



In DBMS file



Records of Database file —

fixed length file

variable length file

① Fixed length file :

create table R

(

A char(100),

B char(50),

C char(50)

);

Size of the record \approx 200 Bytes

Block	R ₁	\approx 200 Bytes
	R ₂	\approx 200 Bytes
	R ₃	⋮
	⋮	→ Block header

(offset table to access
records within the block)

② Variable length file :

create table S

(

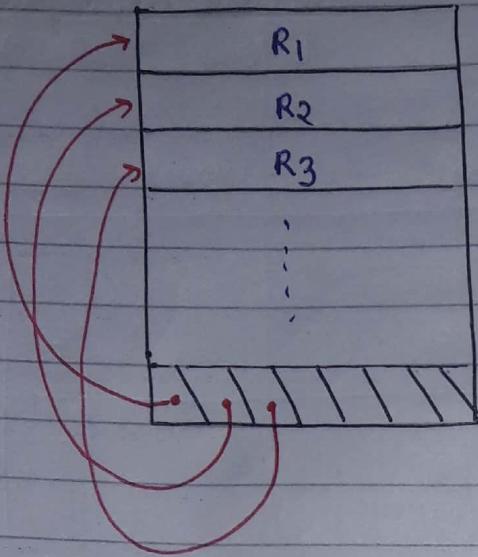
D char(100),

E char(50),

); F text

size of F could be any
it depends on data we store

Record of file(s) may not be fixed length



Records of file can be organised as —

variable length record

Spanned organisation

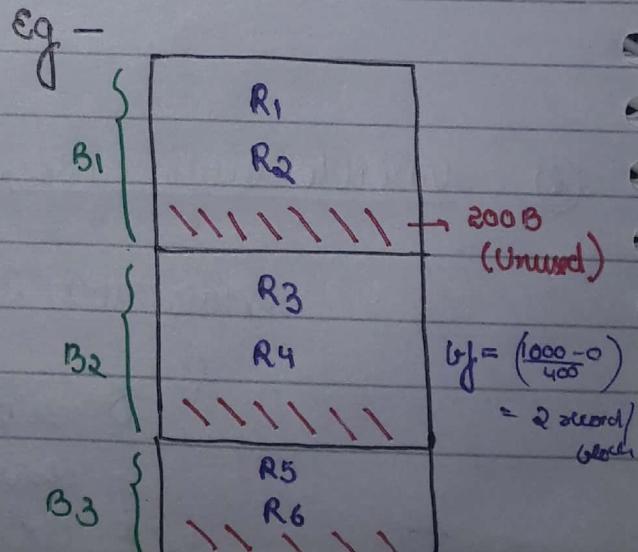
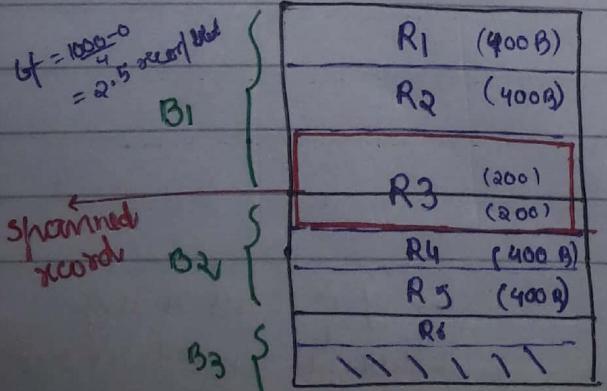
for fixed length record

Unspanned organisation

- Record is allowed to span or store in more than one block.

Complete record ~~may~~ must be stored in one block.

Eg - Block size = 1000 Byte
Record size = 1200 Byte



- Possible to allocate file without internal fragmentation
- More access cost (as record is in more than one block) and is complex to organize (drawback)
- DB uses spanned by default for variable length record.

may not be possible to allocate file without internal fragmentation.

less access cost and easy to organize

DB uses unspanned by default for fixed length record

Block factor —

- Maximum possible records per block is called **block factor**
- It is only for fixed length records.

Block size : B bytes

Block header size : H bytes

Record size : R bytes

*
by default
use this

Block factor for unspanned organization → $\left\lfloor \frac{B-H}{R} \right\rfloor$ records/block

Block factor for spanned organization → $\frac{B-H}{R}$ records/block.

Indexing

DB file [Emp]

ordered field → eid ename ----- Pno unordered field.

	eid	ename	Pno
B1	2		15
	4		18
B2	5		25
	6		19
B3	8		3
	10		35
Bn			

- Indexing is used to reduce the access cost.

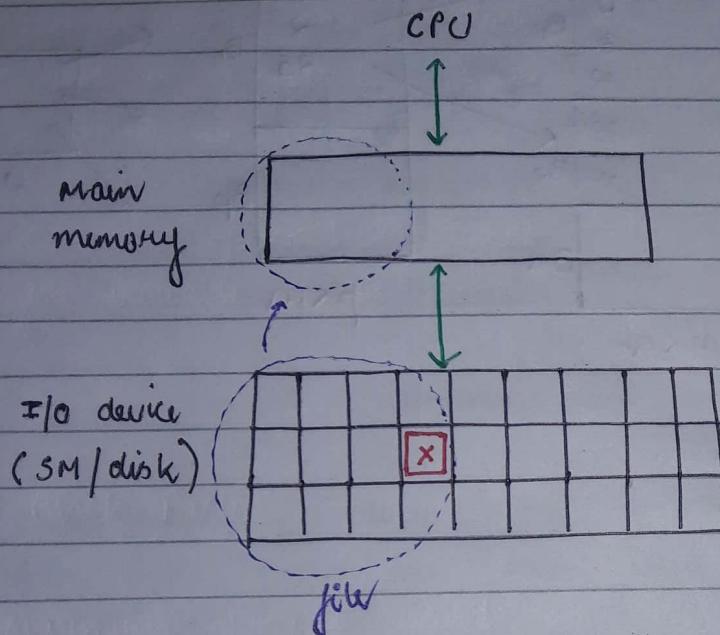
ordered → A B ← unordered

2	4
3	5
9	8
10	9
15	10
16	11
17	2

But, here B is also ordered, but this is not known to DB, there is no guarantee that in future, same type of record will be there

→ I/O cost (Access Cost) :

Number of secondary memory blocks (disk blocks) required to transfer data from disk to main memory in order to access some record is called I/O cost.



- I/O cost to access record from database file of n blocks without index

ⓐ Based on ordered field access:

```
select *  
from emp  
where Eid = X;
```

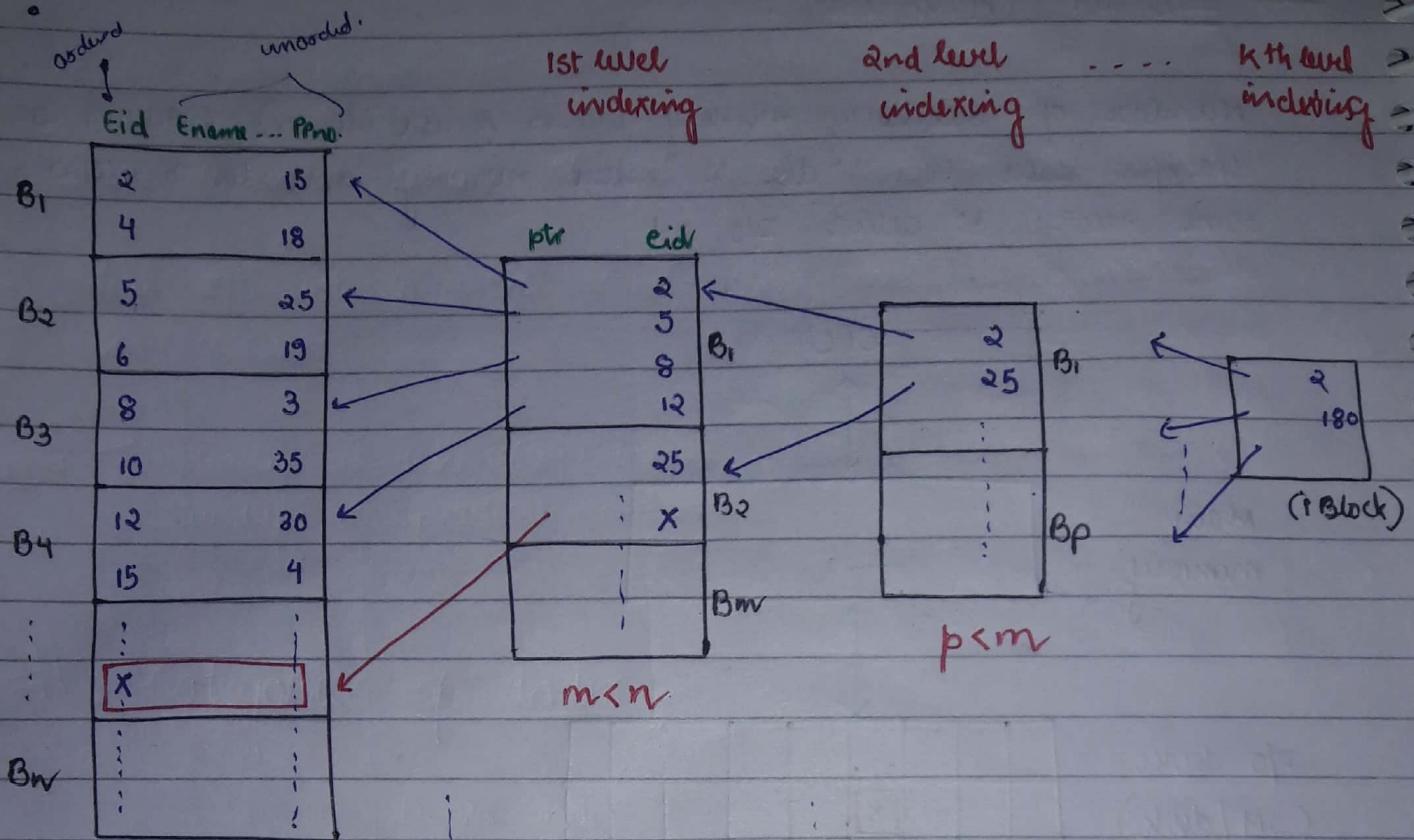
$\lceil \log_2 n \rceil$ block access cost

with assumption that middle of
the file can be found by DBMS.

ⓑ Based on unordered field access:

```
select *  
from emp  
where Pno = X;
```

' n block access cost'
(deadlock or waiting)



I/O cost without
indexing is : $\lceil \log_2 n \rceil$ or n blocks

I/O cost with indexing using
1st level indexing
 $\lceil \log_2 m \rceil + 1$ blocks

I/O cost to access records using multilevel index
 $(k+1)$ blocks

Index file -

- Each entry of index file is 2 fields.
< search key , pointer >
 - field used for indexing.
 - It can be a key (or) non key.
 - Can be an ordered field (or)
Unordered field

- Index file file block factor is max possible index entries
[searchkey, ptr] pairs in index block.

$$\text{Block factor of index file} = \left\lfloor \frac{B - H}{P + K} \right\rfloor \text{entries/block}$$

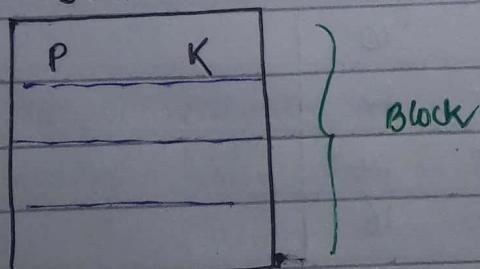
B : size of block

H : size of block header

K : size of search key

P : size of pointer

Index



Multilevel index -

- Index to index until, at last level, there is one block.
- Total I/O cost to access record using multi level index is $(k+1)$ blocks.

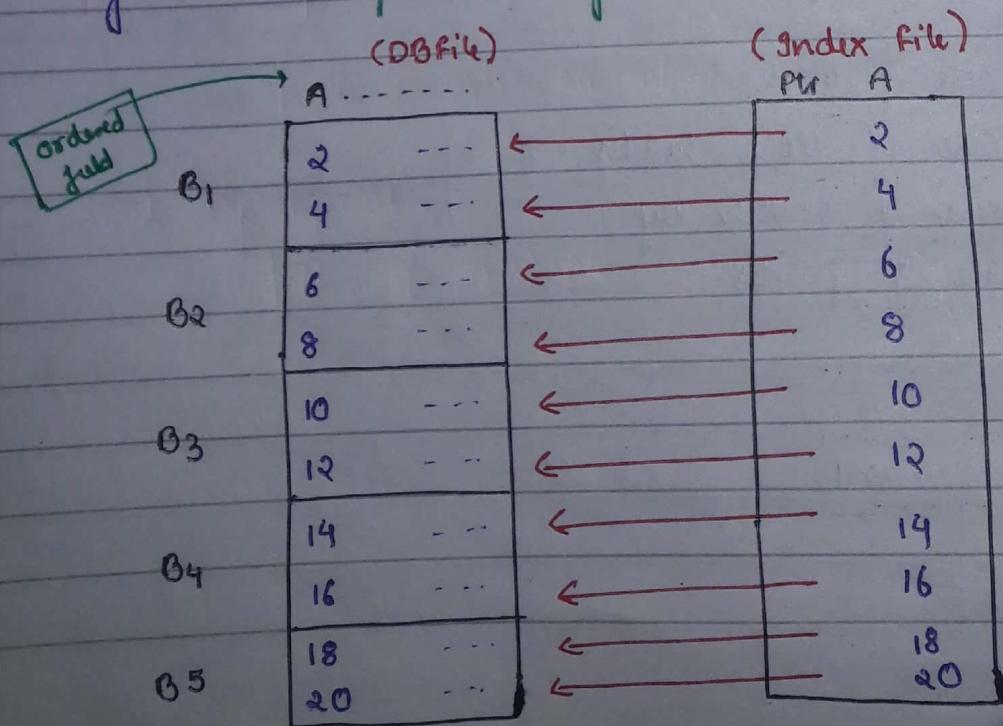
Note — Ideally we take $(k+1)$ blocks, but it may increase slightly in practical conditions.

Categories of Index :

- ① Dense index
- ② Sparse index

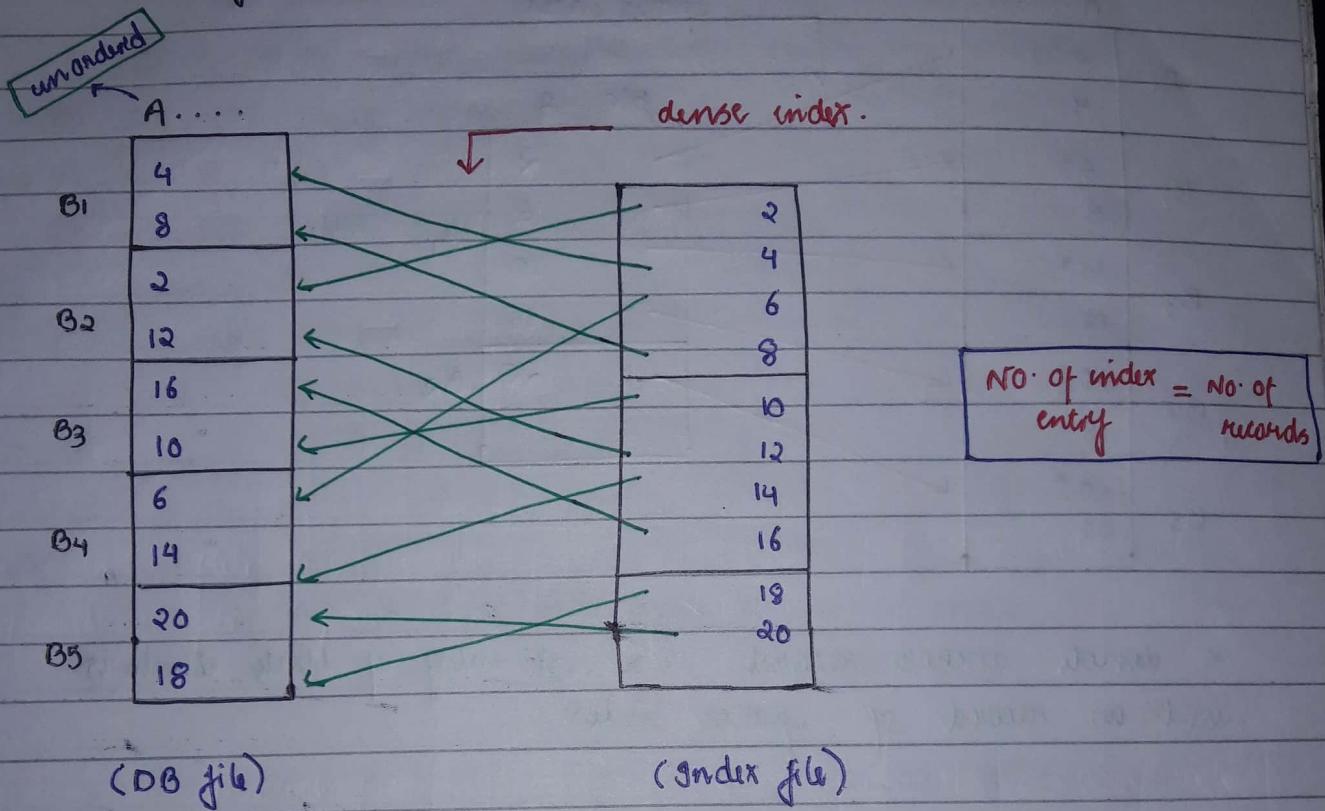
Dense Index —

- More entries are used for the indexing of database files. (max possible entry in index)



- For each record of DB file, there exist an entry in index file.

Indexing for unordered field of DB file -

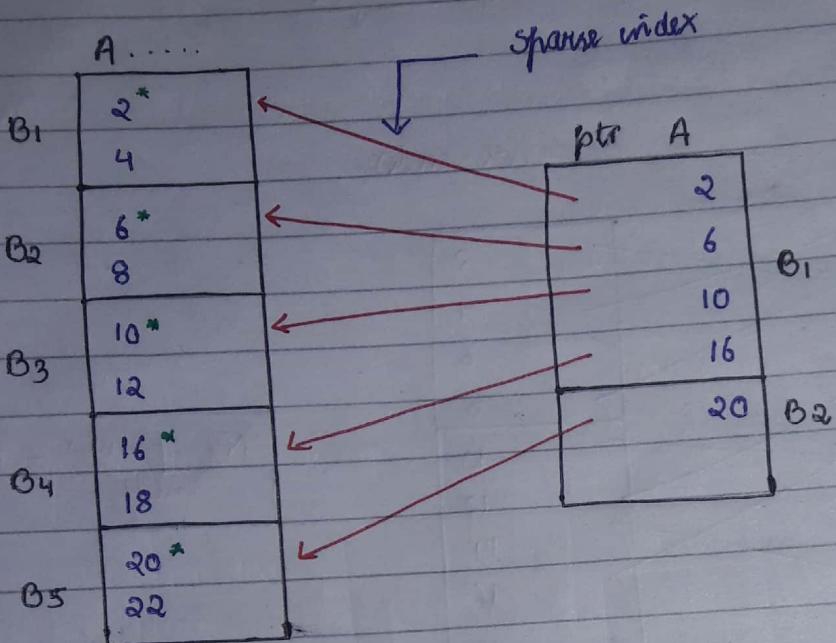


- Dense undersampling can be done for ordered as well as unordered fields.

Sparse Index —

- less entries in index file (objective)
 - for set of records there exist an entry in index file (ie. for many ^{record} entry there is one entry)

- sparse index is possible only if search key is ordered field of the DB file.



* denotes anchor record, i.e. 1st entry of block which is used as records of sparse index

for sparse index,

$$\text{No. of index file entries} < \text{No. of records of DB files.}$$

but, by default,

$$\text{No. of index file entries} \equiv \text{No. of blocks of DB files.}$$

Ques: File consists 50,000 records with record size 100 Bytes
 Block size = 1024 Bytes
 search key = 10 Bytes
 Pointer = 5 Bytes

* ① No. of dense index blocks?

② I/O cost using dense index?

* ③ No. of sparse index blocks?

④ I/O cost using sparse index?

⑤ I/O cost without index

a) Based on ordered field?

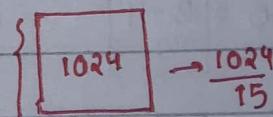
b) Based on unordered field?

Soluⁿ:

$$\text{No. of dense index block} = \frac{\text{No. of entries}}{\text{Block size}}$$

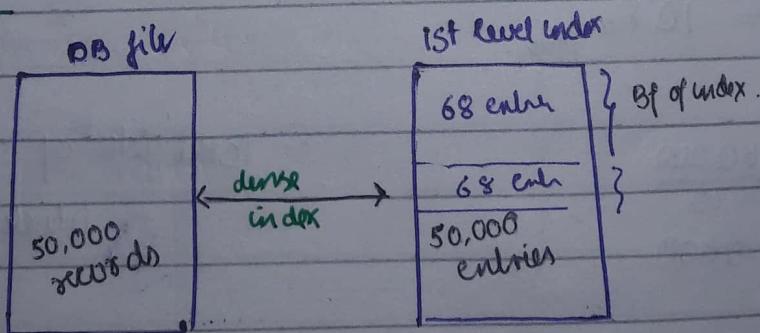
$$= \frac{50,000}{1024} \times \frac{15}{10} \quad \begin{matrix} \times \text{ gt qns} \\ \Rightarrow \text{wrong ans} \end{matrix}$$

(as all 1024 block cannot be used because of unspan)



$$\Rightarrow 68 \quad 68 \times 15 = 1024 \\ \text{i.e. } 4B \text{ is unused}$$

dense index —



$$\text{Block factor of index} = \left\lfloor \frac{B-H}{K+P} \right\rfloor = \left\lfloor \frac{1024-0}{10+5} \right\rfloor = 68 \text{ entries/block}$$

$$\therefore \text{No. of dense index blocks} = \left\lceil \frac{50,000}{68} \right\rceil$$

$$= 736 \text{ blocks} \quad \underline{\text{Ans}}$$

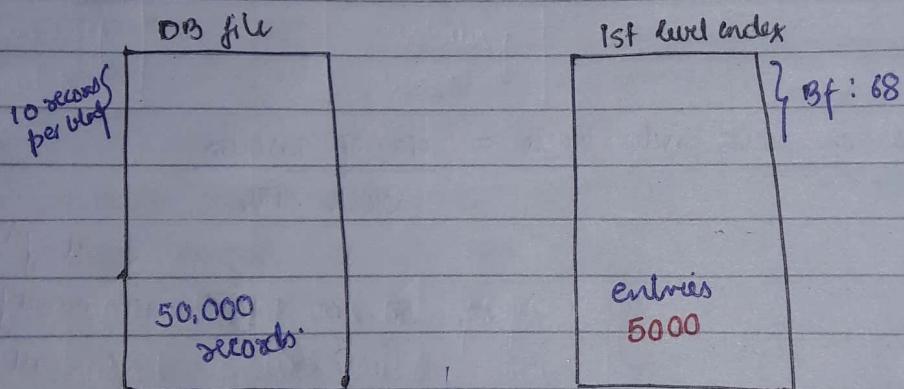
2)

$$\text{I/O cost using dense index} = \lceil \log_2 m \rceil + 1$$

$$= \lceil \log_2 736 \rceil + 1 \Rightarrow 10 + 1$$

$$\Rightarrow 11 \text{ blocks}$$

Sparse index block -



$$\text{Block factor of DB} = \left\lceil \frac{B-H}{R} \right\rceil$$

$$= \left\lceil \frac{1024-0}{100} \right\rceil$$

$$= 10 \text{ records/block}$$

$$\text{Block factor of indexed} = \left\lceil \frac{B-H}{K+P} \right\rceil$$

$$= \frac{1024-0}{10+5}$$

$$= 68 \text{ entries/block}$$

$$\therefore \text{Total block} = \frac{50,000}{10}$$

$$= 5000$$

$$\therefore \text{Total No. of sparse index}$$

$$\text{block} = \left\lceil \frac{5000}{68} \right\rceil$$

$$= 74 \text{ blocks}$$

Ans

4) I/O cost using sparse index = $\lceil \log_2 m \rceil + 1$

$$= \lceil \log_2 74 \rceil + 1$$

$$= 8 \text{ blocks} \quad \underline{\text{Ans}}$$

5) I/O cost without

a) based on ordered field.

(we can use sparse)

$$= \lceil \log_2 n \rceil$$

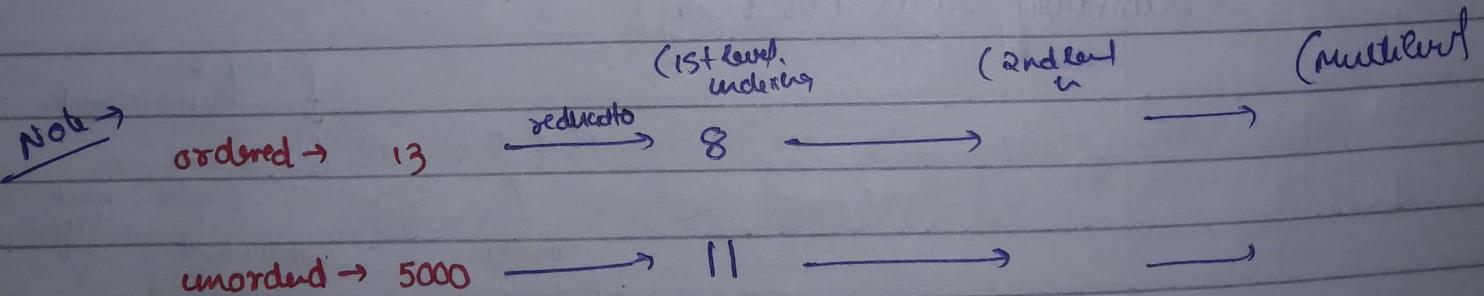
$$= \lceil \log_2 5000 \rceil$$

$$= 13 \text{ blocks} \quad \underline{\text{Ans}}$$

b) based on unordered field ?

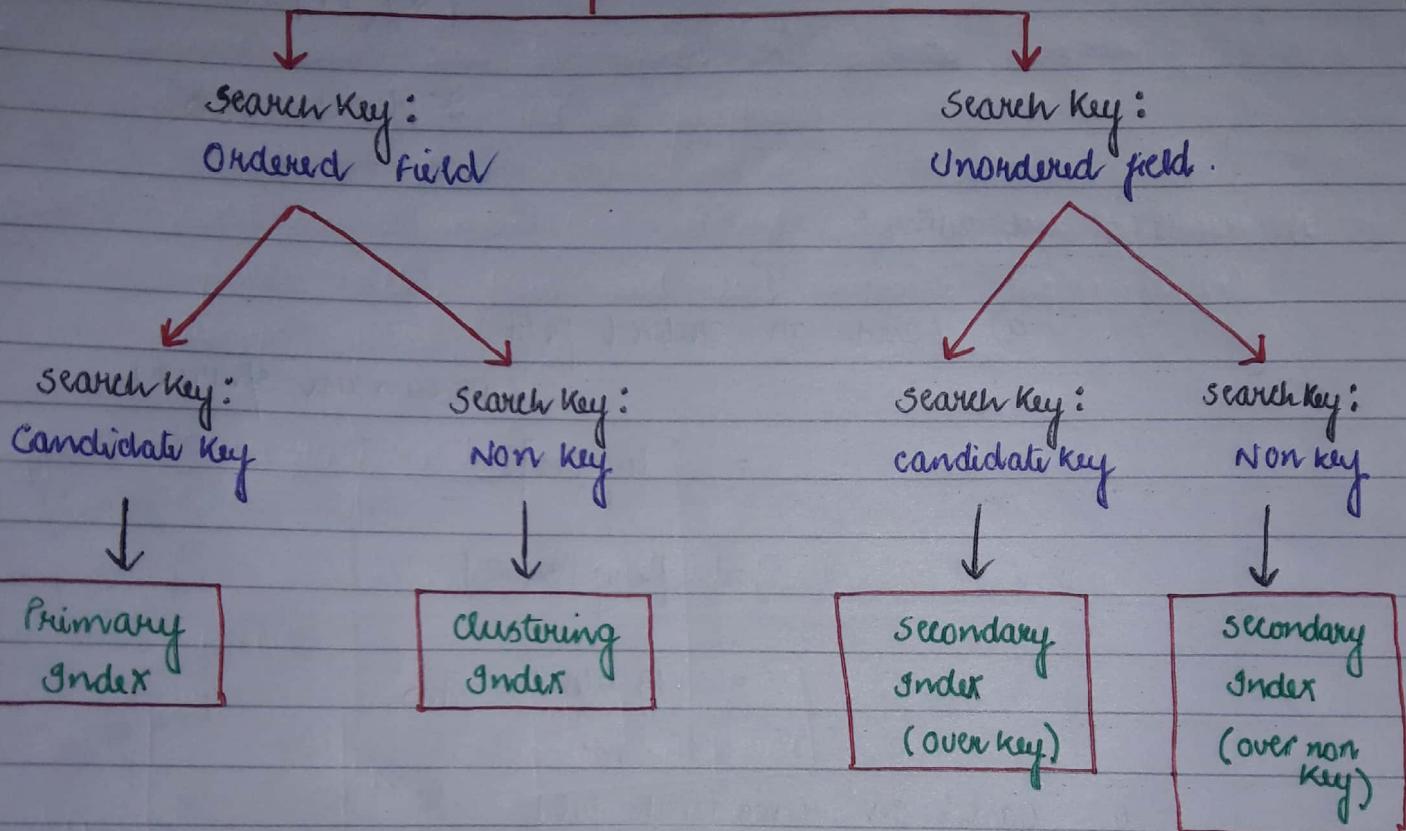
$$= n$$

$$= 5000 \text{ blocks} \quad \underline{\text{Ans}}$$



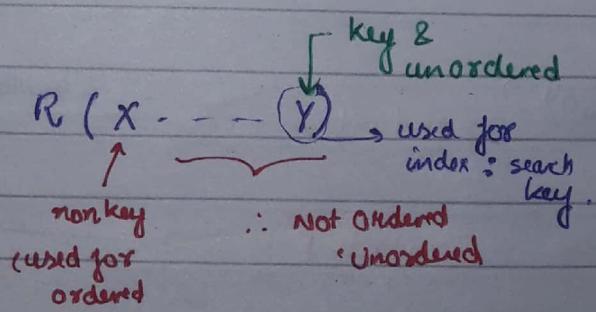
search key: fields used for indexing DB file

Types of Index



Ans: Records of rel R physically ordered over non key field X, and index build over key field of rel R.

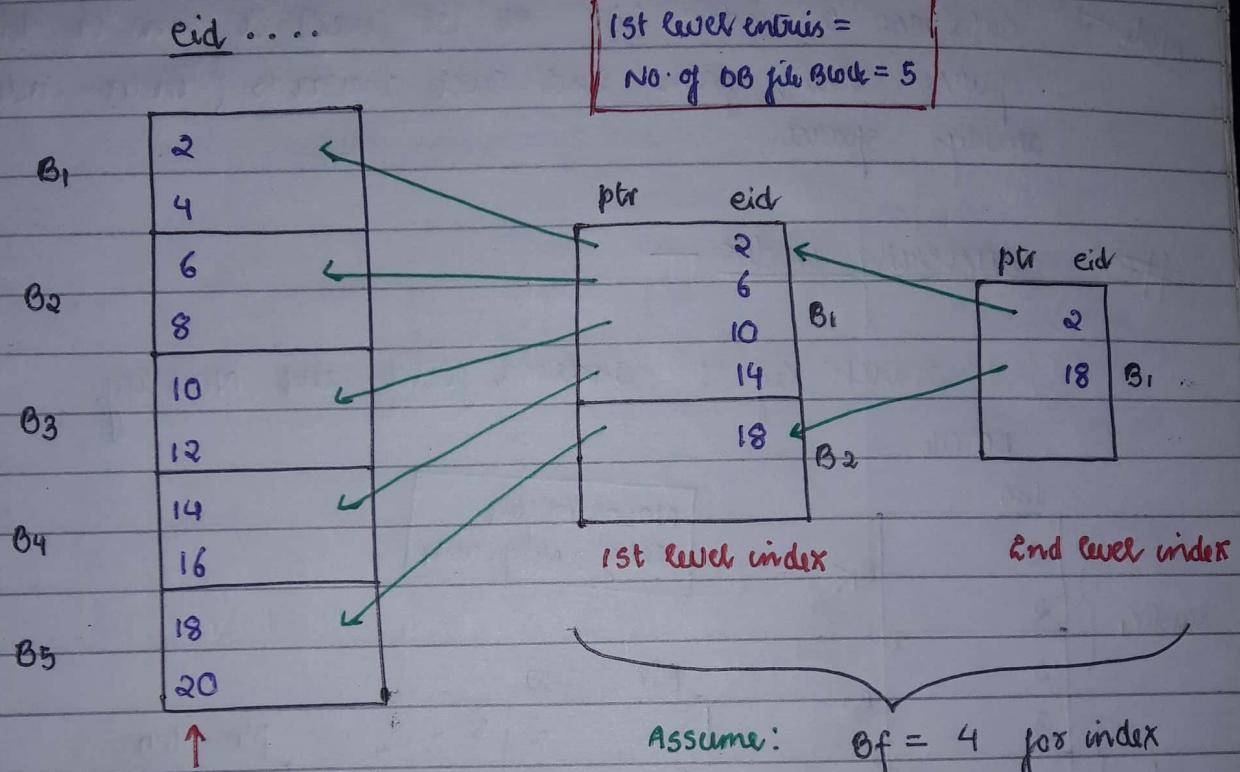
'Secondary Index (over key)'



Primary Index —

where search key is key and ordered field of DB file, it is primary index.

Eg -



key and ordered field

Assume: (DB with B.f = 2)

I/O cost to access record using primary index with multilevel index : (k+1) blocks

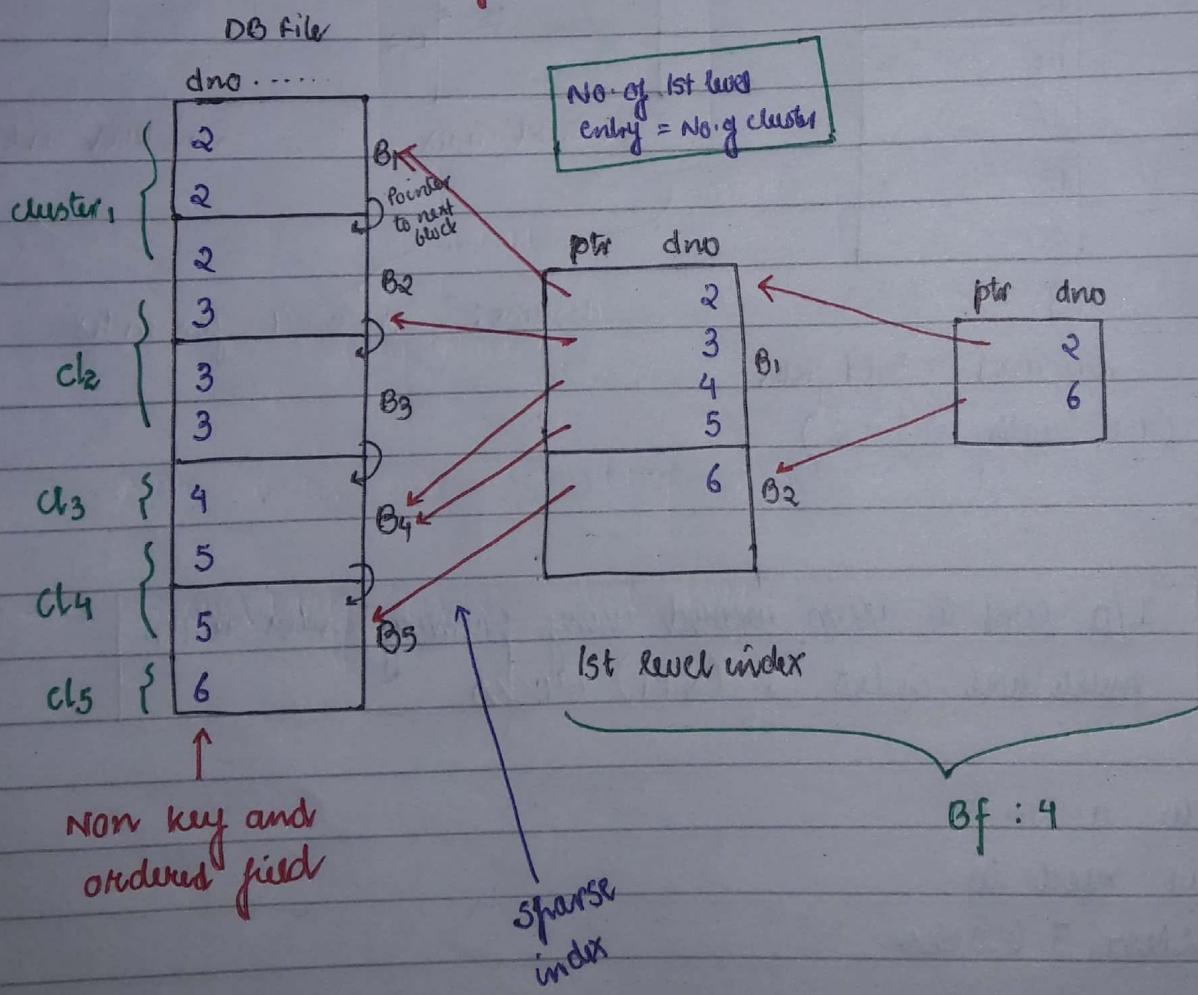
Hence, to access 18,
we need to
access 3 blocks.

- For any DB relation almost one primary index is possible. (as search key is ordered field)
- Primary index can be dense or sparse. (because of ordered field), but sparse PI is preferred.

Note - Database to index (i.e. at 1st level) it can be dense or sparse but, from 2nd level onwards (index-index), always sparse.

Clustering index -

Search Key : ordered field and Non key

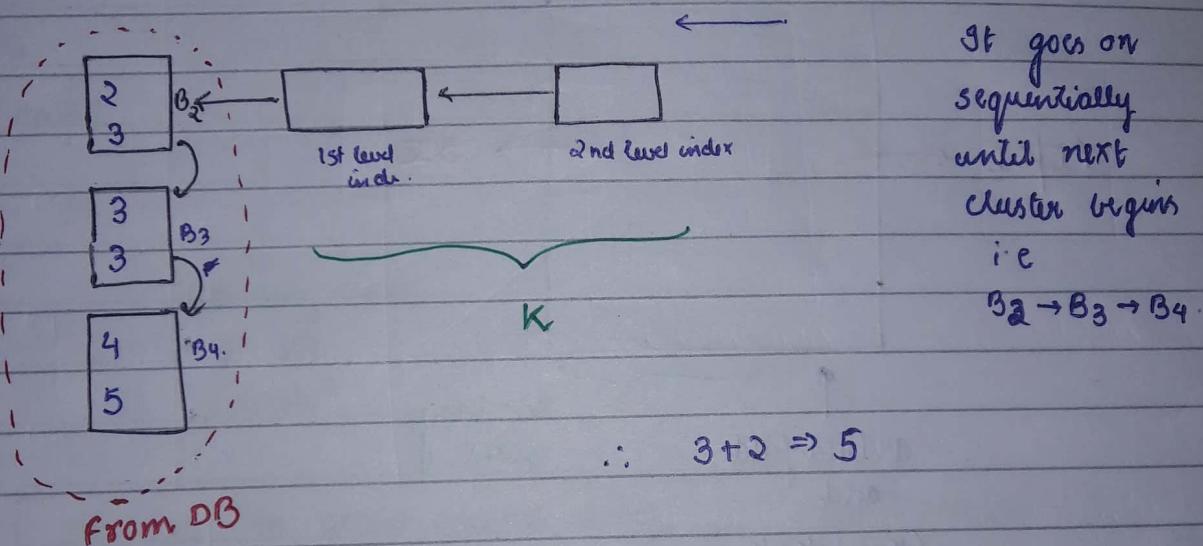


I/O cost to access cluster of records using clustering index at 1st level followed with multilevel indexing =

$K + \text{one or more DB blocks access until next cluster begin}$

Eg -

select * from emp
where dno = 3;



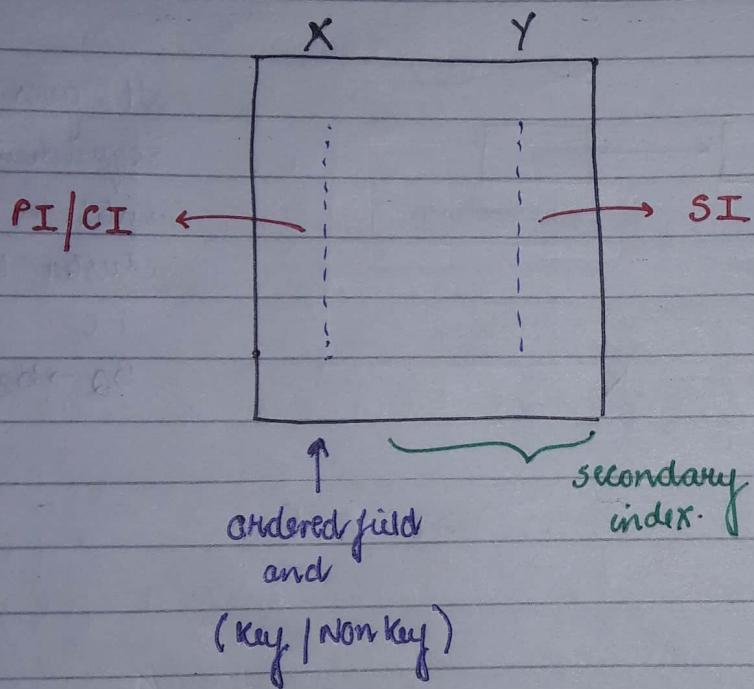
- Clustering index is mostly **sparse index**. (clustering index can be dense if each cluster is with only 1 record)
- For any DB relation almost one clustering index is possible (because search key is ordered field)
- For any DB relation, either primary index or clustering index is possible but not both.

Secondary Index —

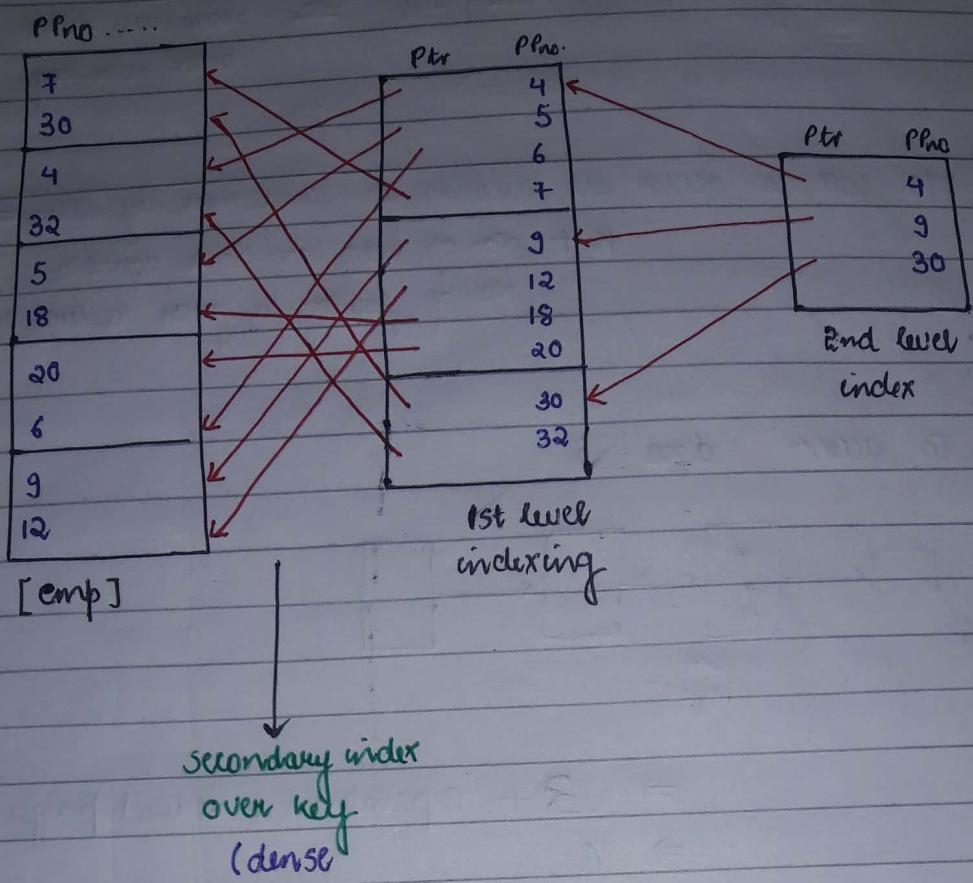
Search Key : unordered field
(and)

Key / Non Key

- It is called as secondary index as it is secondary possible way to access data even when primary index / clustering index already exist.

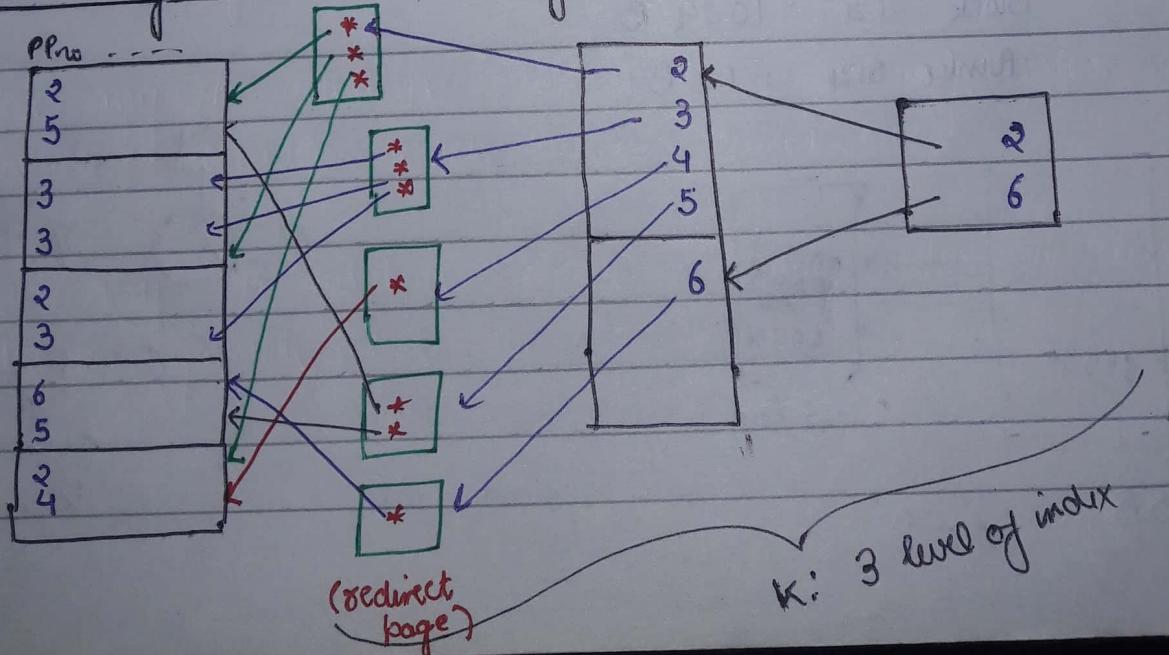


- More than one secondary index are possible.
- secondary index over key :



I/O cost to access record using = $(k+i)$ blocks
 SI with multi-level indexing

→ secondary index over non key:

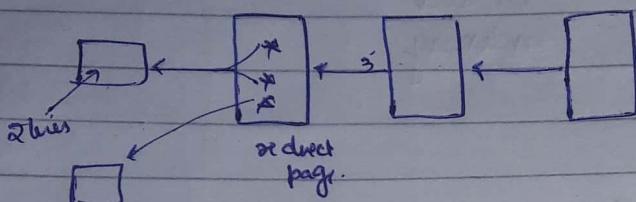


Note -

clustering index + redirect page \leftarrow Secondary index
over non key

I/O cost to access all records of same non key value =
 $k + \text{No. of data base block equal to}$
 $\text{No. of pointers in given redirect page}$

Eg- To access dno = 3



$$\Rightarrow 3 + 3 \text{ pointers in redirect page}$$

$$\Rightarrow 6.$$

Pg 65
(ii)

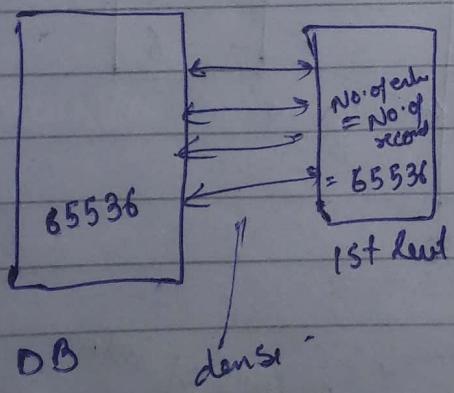
No of records = 65536

Record size = 32 B

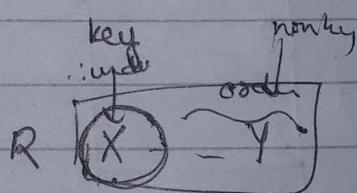
Search key size = 6 B

Block size = 1024 B

Pointer size = 12 B



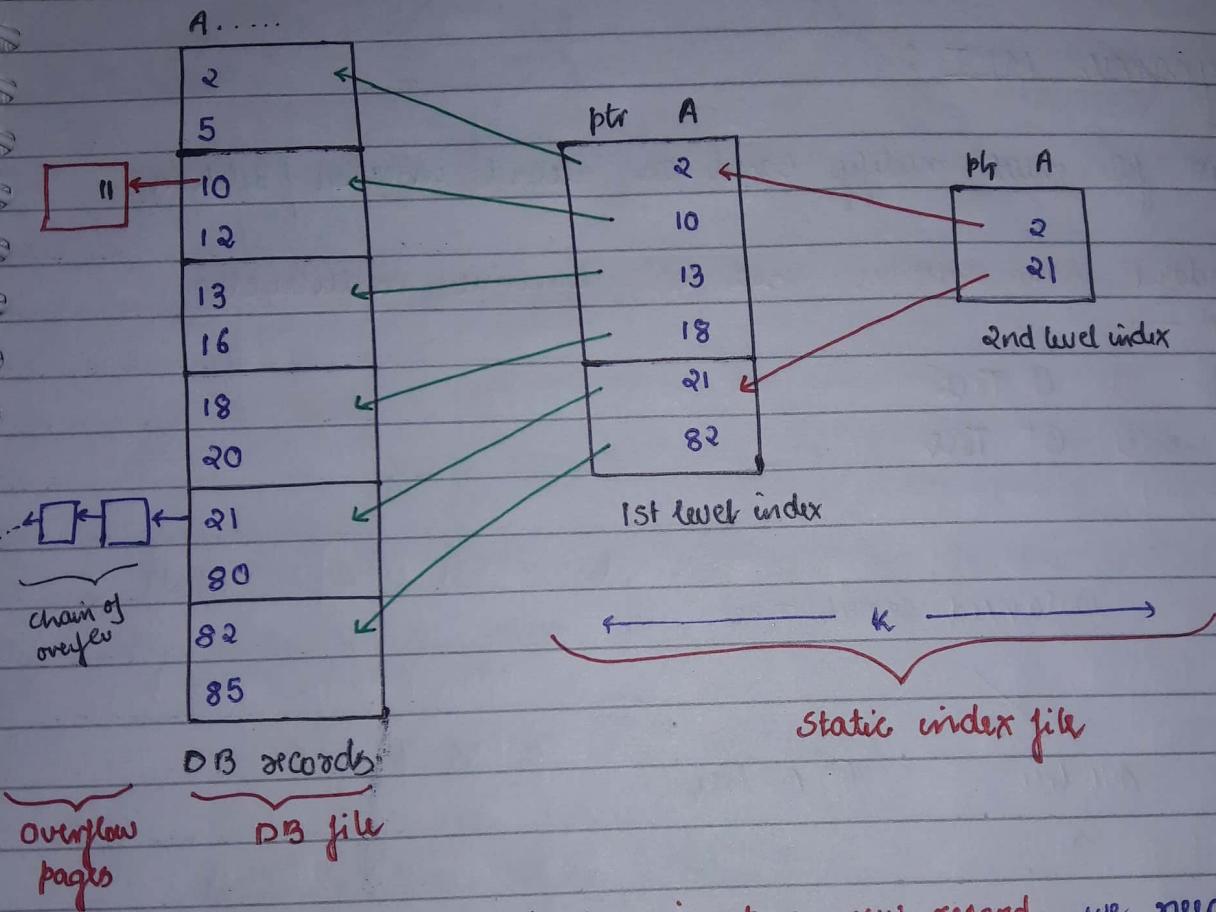
$$I/O \text{ cost} = k+1$$



Secondary
index
over
key

Multilevel Indexing

- ① Static MLI: Index file is constant and newly inserted records are stored in overflow pages.



NOW, when we need to insert a new record, we need to change the index completely. (which is complicated)

So, we make index file static and add overflow pages

↓ but an issue arises

Access cost increases

Disadvantage of static MLI -

- Worst case access time = $O(n)$, where n is no. of record.
- Minimum usage of index block can become 0 %.

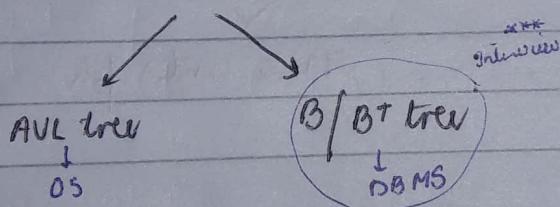
② Dynamic MLI :

→ Index file must modify based on record insertion / deletion.

→ Standard Data structure used for dynamic multilevel index —

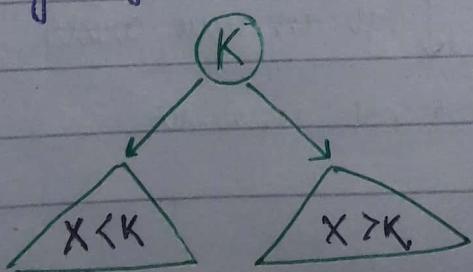
- ① B Tree
- ② B⁺ Tree

Balanced search Tree



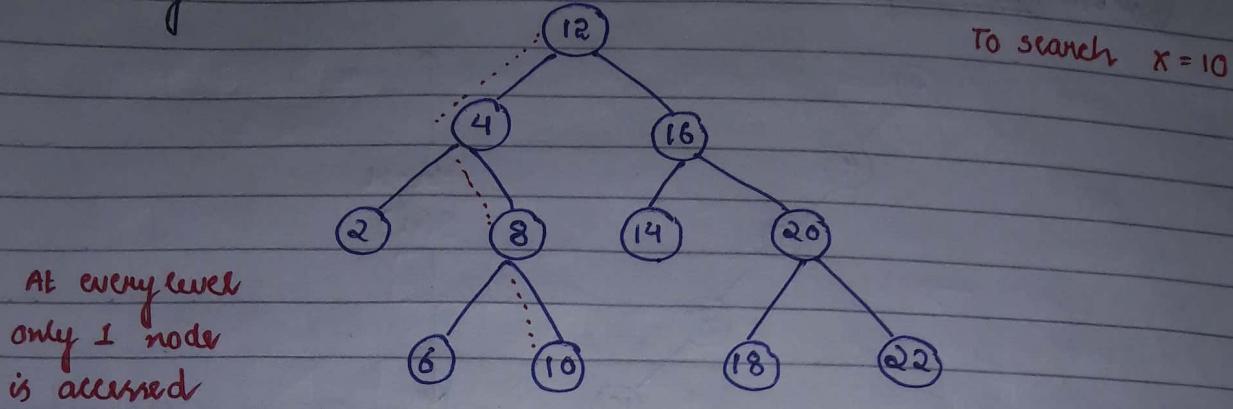
Search Tree -

Data structure used to store keys in tree format and every parent key is more than keys of left subtree and less than keys of right subtree



to perform search
open n efficiently.

Eg - BST



Probe sequence : 12, 4, 8, 10 (succes)

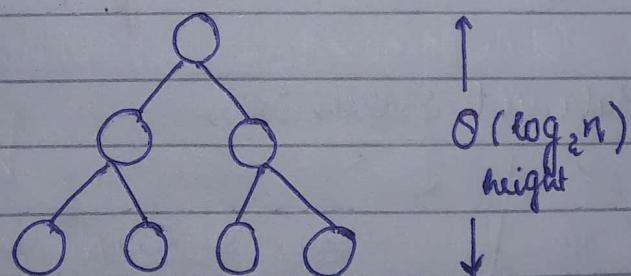
To search $x = 17$

probe sequence : 12, 16, 20, 18 (fails)

∴ Probe sequence to search any key in search tree is exactly one node access from each level.

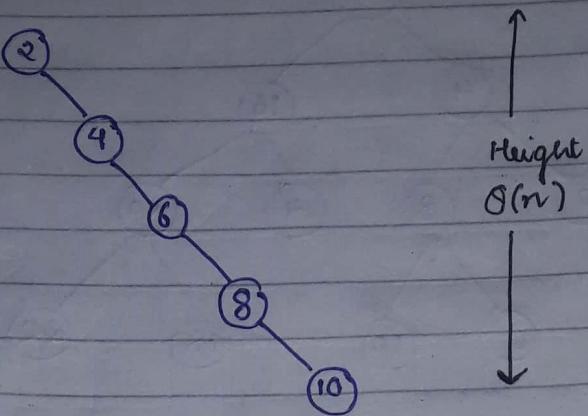
For a search tree of 'n' key

advantage a) Min height : $O(\log n)$



search cost = $O(\log n)$

disadvantage Max height : $O(n)$



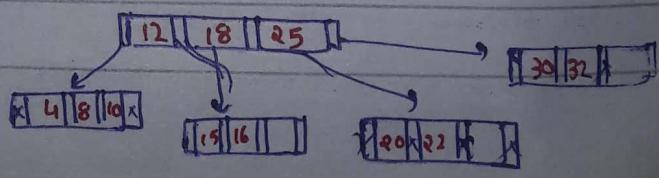
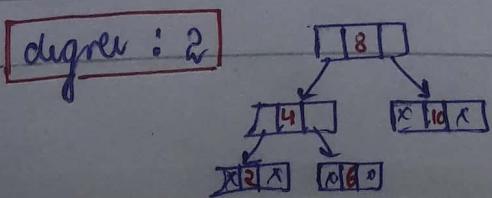
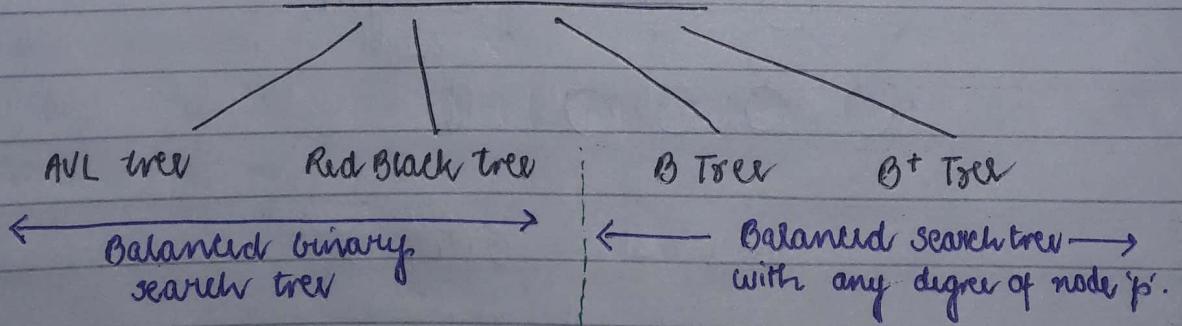
$$\text{search cost} = O(n)$$

To solve this problem, we have BST.

Binary Search Tree (Height Restricted search trees)

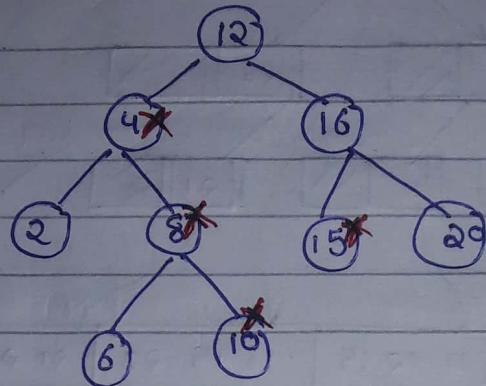
- Max height of search tree for n distinct keys should not exceed $O(\log n)$ → objective of BST
- Advantage: worst case search cost to search the n distinct element is $O(\log n)$.

Balanced search Trees



Since there is no restriction of degree in B and B⁺ tree, ∴ it is preferred for database index.

Note - lazy deletion:



If we wish to delete 8, we simply set the flag and do not delete it

↓
Later when many are deleted i.e. many flags are set

↓
we re-create the tree

as for n deletion, it takes $O(n \log n)$

Ques: Why B/B⁺ tree used for DB files index rather than BST ??

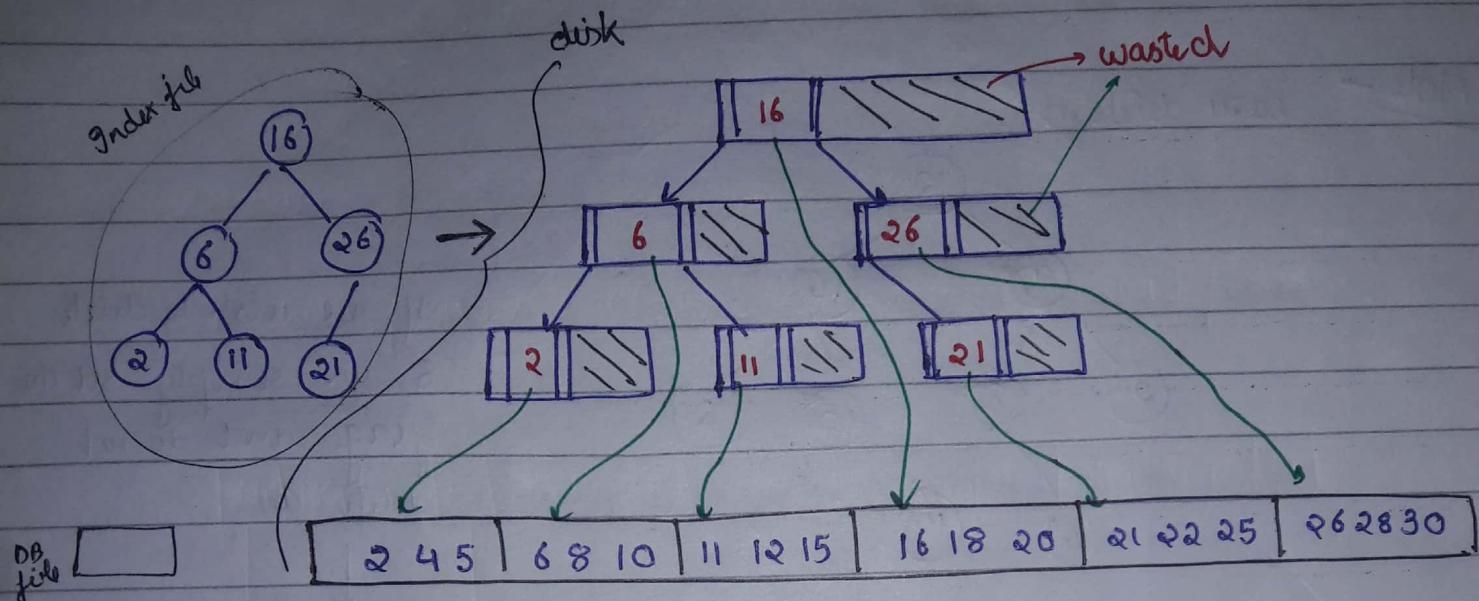
① Because to reduce access cost

(DB file stored in disk index to DB file must also be in disk and

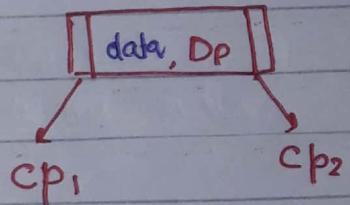
'Data access from disk is in terms of blocks' (drawback))

② If AVL tree is used, as DB index :

2	4	5	6	8	10	11	12	15	16	18	20	21	22	25	26	28	30
---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----



child pointer: points to other tree node (blue line)
 data pointer: " " original data file (green line)



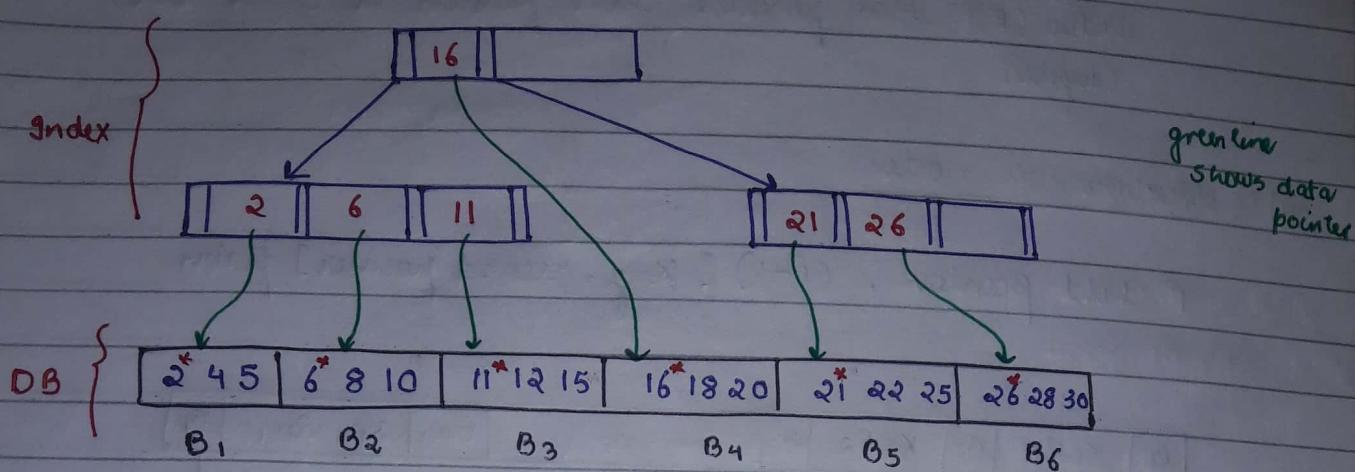
If AVL tree is used for DB index, one entire disk block should allocate for index node and only one key is stored in one node which leads to wastage of disk space allocated for index and more access cost



To avoid this problem

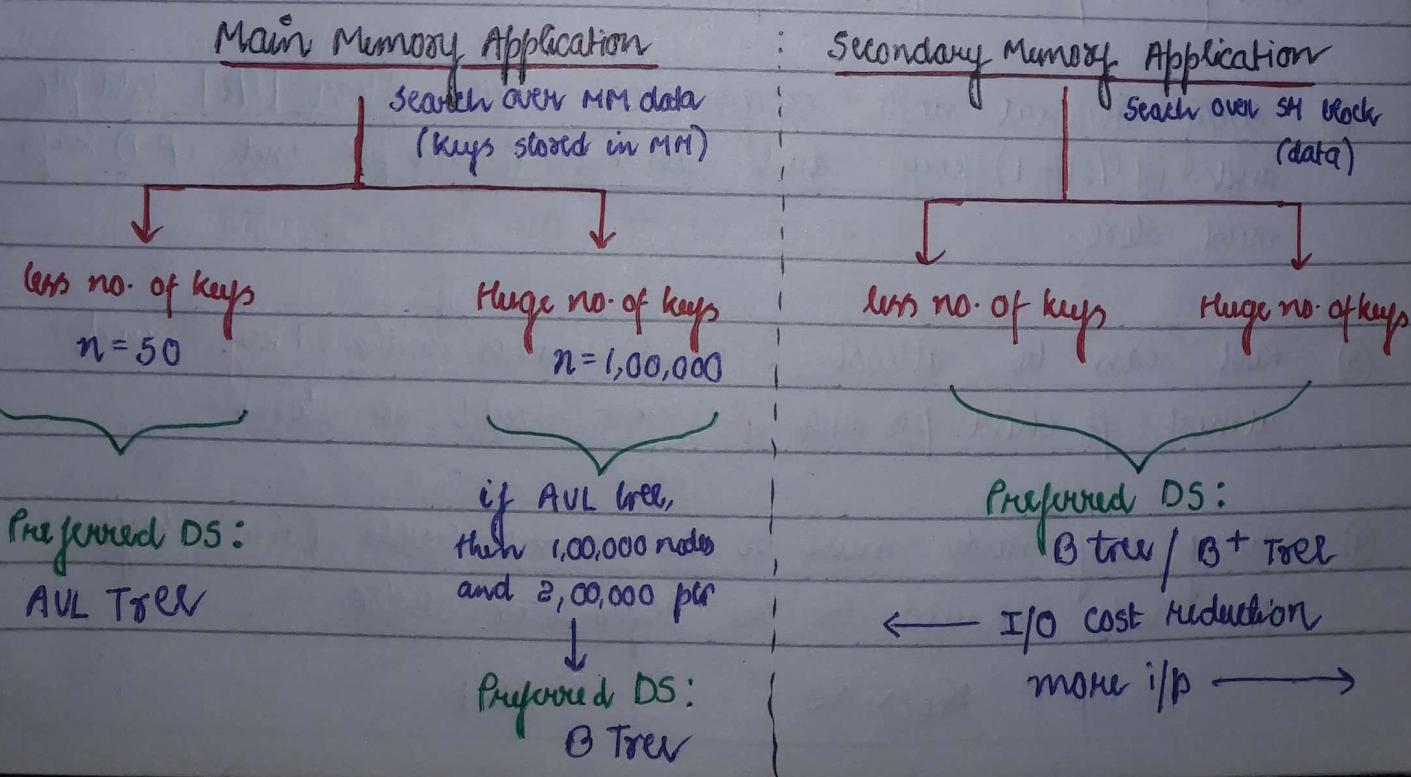
B/B⁺ Tree is
Used

③ If B/B⁺ tree used for DB index



- ① less I/O cost
- ② less wastage of disk space.

Conclusion



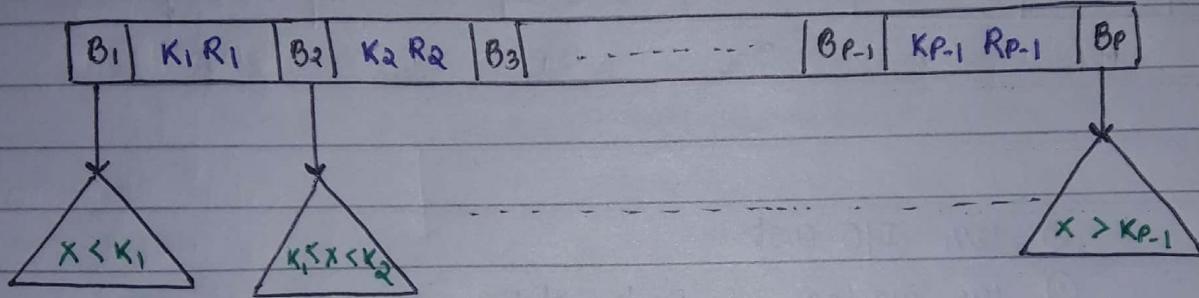
12/11/17

B-Tree →

order (P): Max possible child pointers in B Tree node
(degree)

① Structure of node

P child pointers, $(P-1)$ [key, record pointer] pairs



- child pointer / block pointer / tree pointer points to the index node.
- data pointer / record pointer points to the database block file.

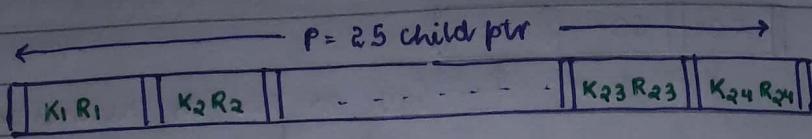
② Every internal node except root with at least $\lceil P/2 \rceil$ child ptr, and $(\lceil P/2 \rceil - 1)$ keys and at most P child ptr and $(P-1)$ keys must store.

③ Root can be at least 2 child pointers with 1 key and atmost P child ptr with $(P-1)$ keys should be stored.

④ Every leaf node must be at same levels (and) keys in node should be in sequential order

$$K_1 < K_2 < \dots < K_{P-1}$$

eg - Assume the child pointers $P = 25$



\Downarrow
Node split
(in order to insert new node)

Assume, we need to insert (K_{25}, R_{25})

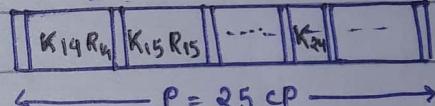
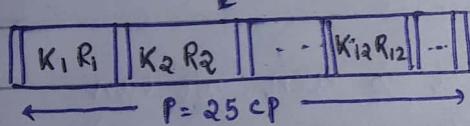
\downarrow
we cannot do as it is full
 \therefore split.

$K_1 < K_2 < \dots < K_{13} < \dots < K_{25}$

Min 2 cp and 1 key



Min $\lceil \frac{P}{2} \rceil$ CP and $(\lceil \frac{P}{2} \rceil - 1)$ key.



Ques: Construct B Tree with order $P: 4$ [max CP per node] with following sequence of keys

20, 60, 30, 40, 90, 50, 10, 15, 25, 70, 80

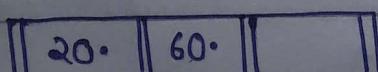
insert 20 -



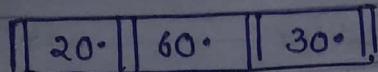
$P=4$

$\therefore (P-1) = 3$ keys

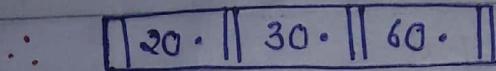
insert 60 -



insert 30 -



↑ cut as per its restriction
(it must be a sequence data)



insert 40 -

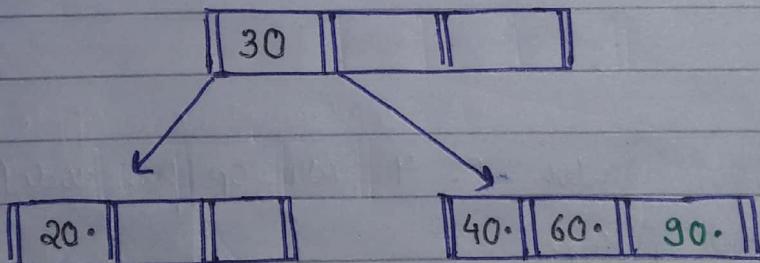
At this level, the node is full ∴ Node splitting.



Node splitting

20 {30; 40} 60

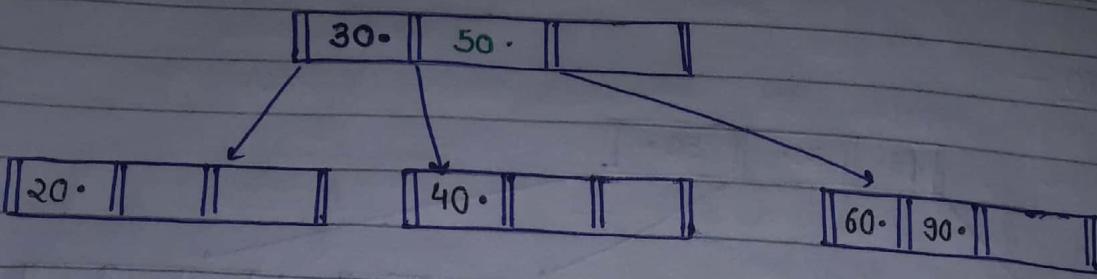
(Since it is even, we can take any 30 or 40)



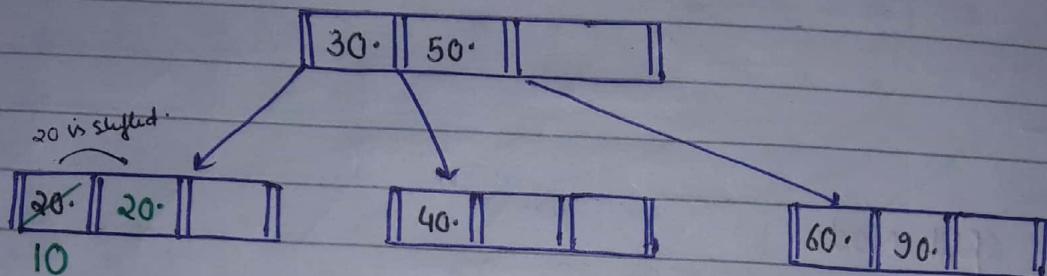
insert 50 -

Now to insert 50 .. node is full

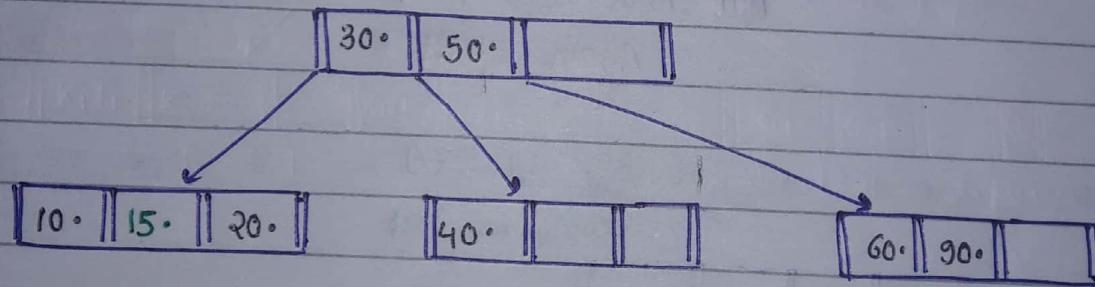
∴ Split 40, (50), 60, 90
 ↓
 moved to root



insert 10 —



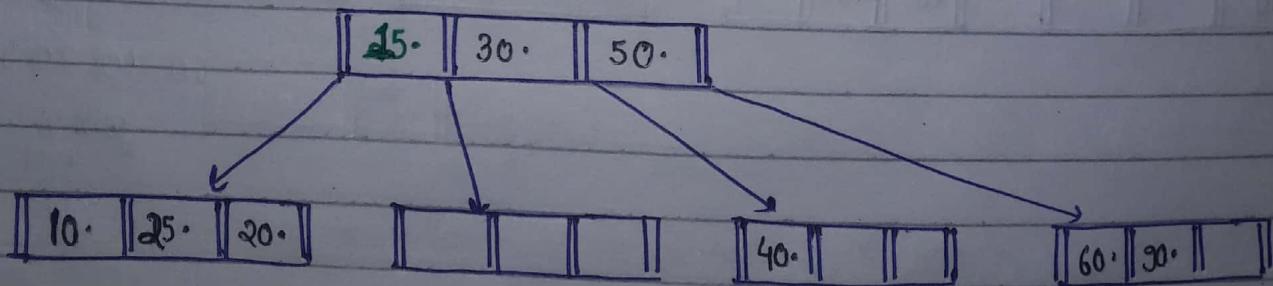
insert 15 —



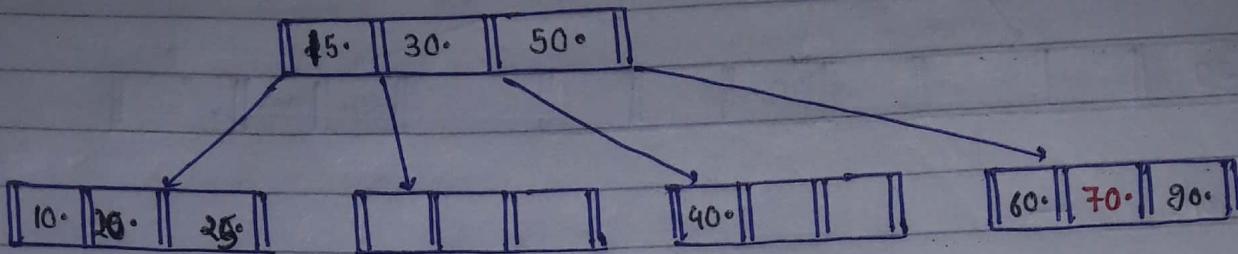
insert 25 —

split node goes to root

10, 15, 20, 25



Insert 70 —



Insert 80 —

Split Node

60 (70) 80 90

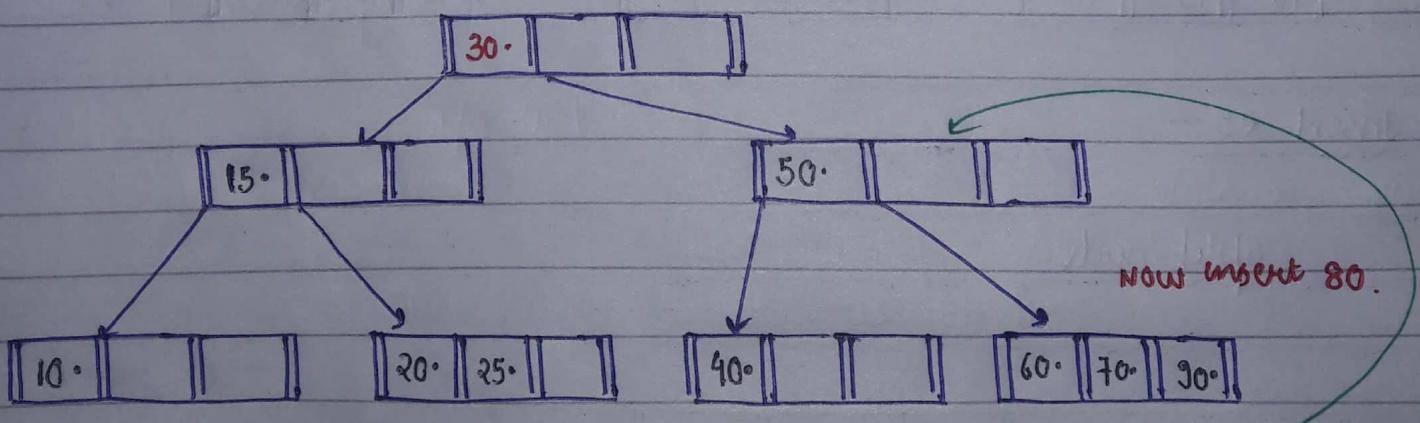
goes to root

but root also full

∴ Again split

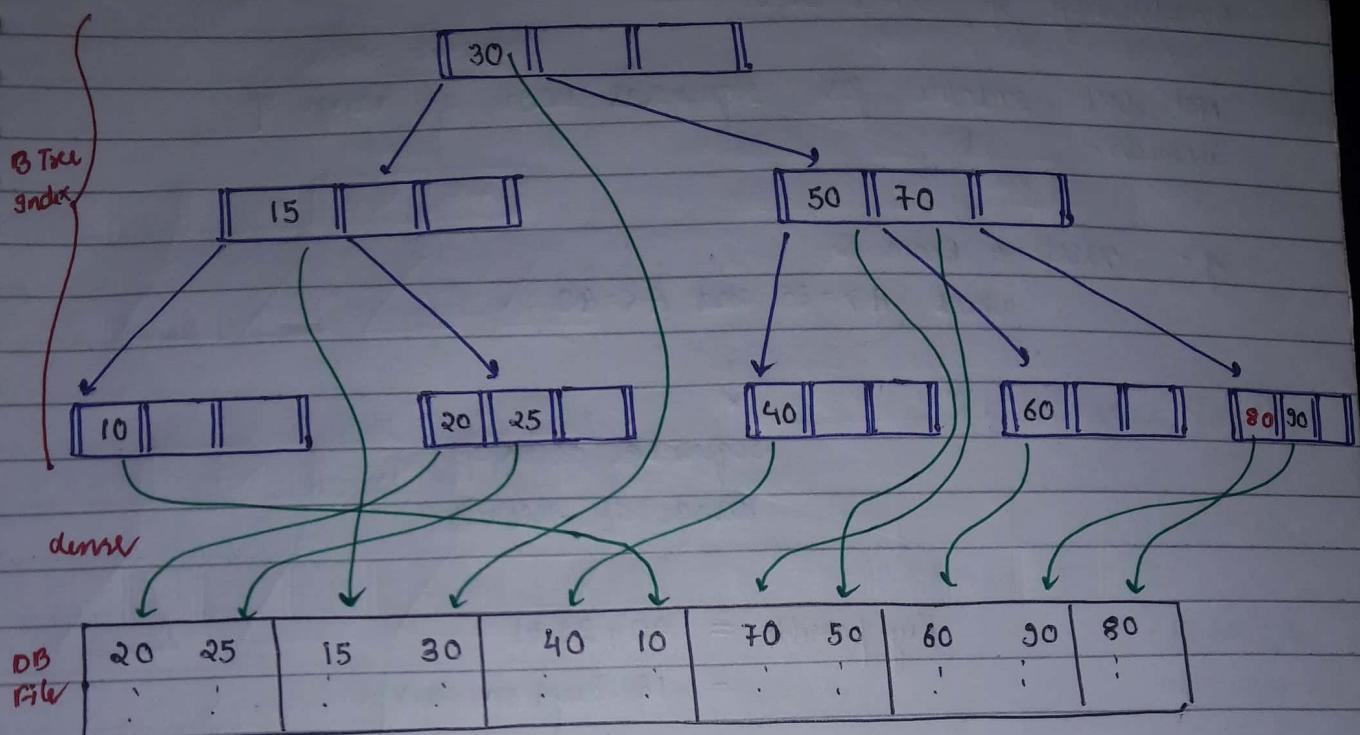
15 (30) 50 70

becomes new root



60, (70) 80, 90

goes to root



Advantage of B-Tree index —

B-Tree index is best suitable for random access of any one record.

Eg - select *
from R

where $A = 25$ ← Any one record access of file

I/O cost : $(K+1)$ block access

$$K \approx O(\log_p n)$$

I/O cost of random access query : $O(\log_p n)$

Disadvantage of B Tree index

Not best suitable for sequential access of range of records.

Eg - select * from R
where $A \geq 25$ and $A \leq 40$;



sequential access of
range of records.

$$\begin{aligned}\text{Key possible} &= 40 - 25 + 1 \\ &= 16 \text{ keys in range.}\end{aligned}$$

$$\begin{aligned}\therefore \text{Total access cost} &= 16 * (\text{access cost}) \\ &= 16 * \log_{pn} \text{ (large)}\end{aligned}$$

cost of successful = $k+1$ Block

cost of unsuccessful = k .

$$\therefore I/O \text{ cost} = \Theta(k \cdot \log_{pn})$$

B^+ Tree -

Order (P): Max possible pointers in B^+ tree node.
(degree)

① Structure of node:

a) Leaf Node -

It consists only (key, R_P) pairs and one block pointer points to next leaf node.

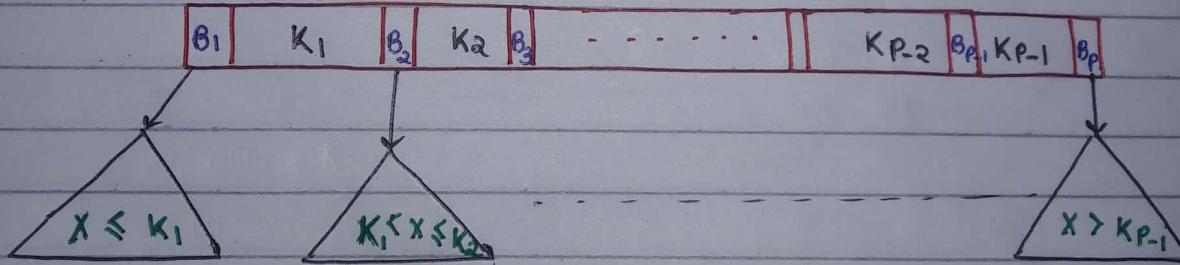


P : NO. of pointers.

(pointing to
next leaf node)

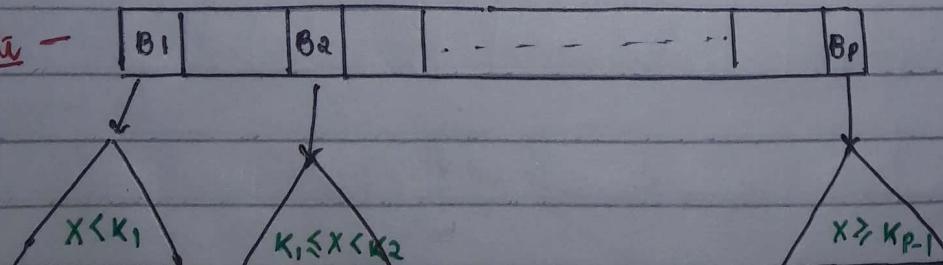
b) Internal Node -

It consists only keys and child pointer and no record pointer.

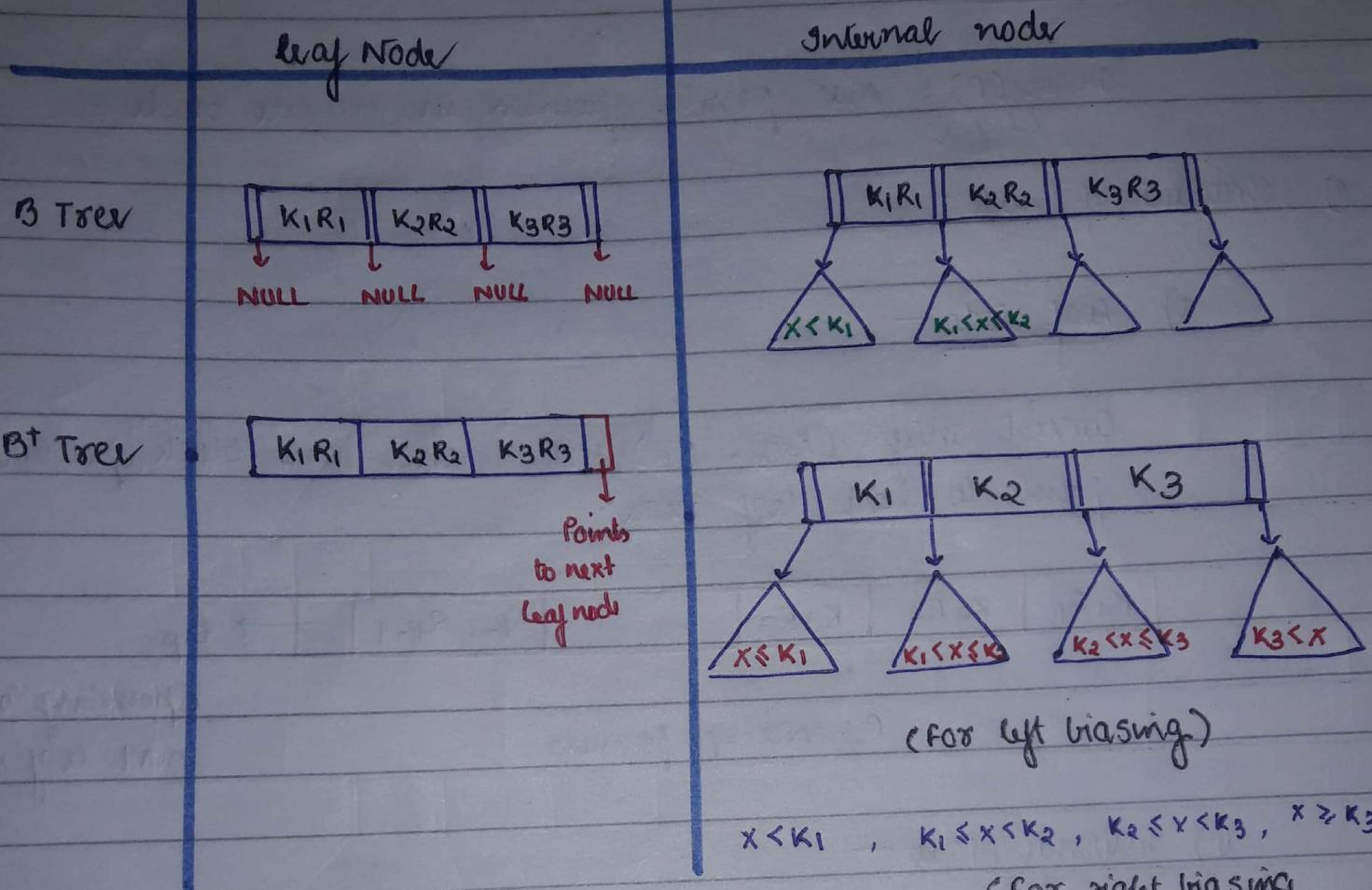


↑
left biasing
(parent key is maintained towards its left pointer)

Note -



↑
Right Biasing
(parent maintained towards right pointer)



Ques: construct B^+ tree with order $P=4$ (max possible pointer per node) with following sequence of keys -

20, 60, 30, 40, 90, 50, 10, 15, 25, 70, 80.

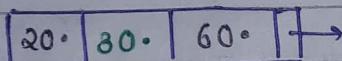
Insert 20 -



Insert 60 -



Insert 30 -

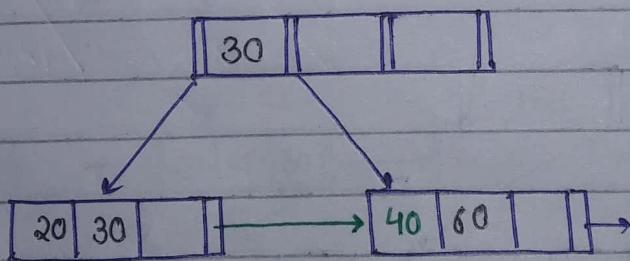


Insert 40 -

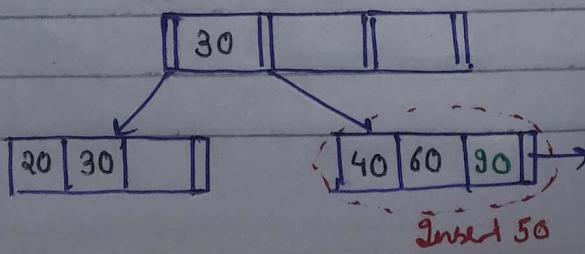
leaf node splitting

20, 30, 40, 60

left biasing.



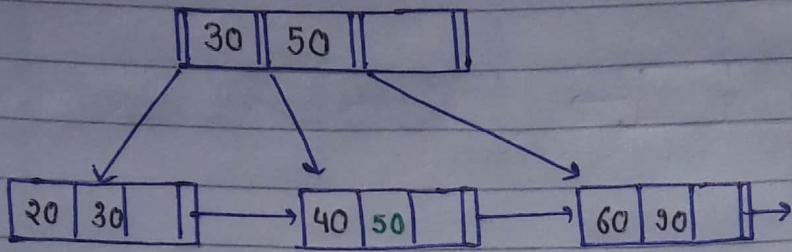
Insert 90 -



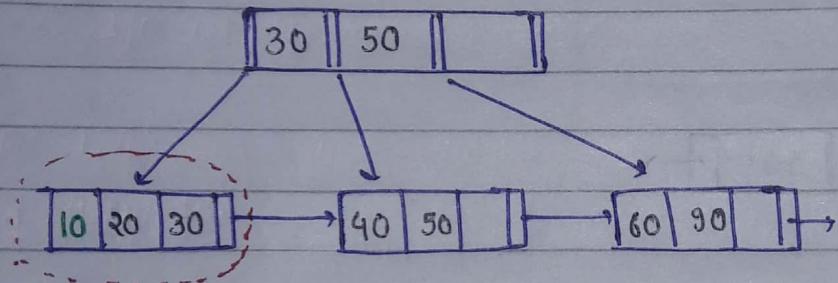
insert 50 —

split the leaf node

40, 50, 60, 90



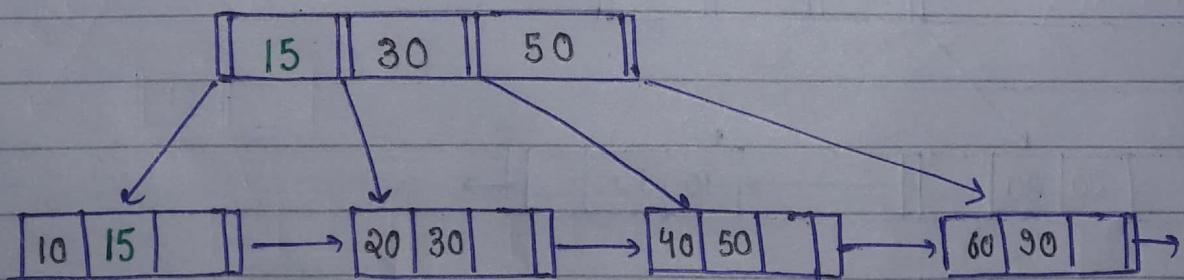
insert 10 —



insert 15 —

split leaf node

10, 15, 20, 30



→ Advantage of B^+ Tree index -

- ① B^+ tree index is best suitable for random access of any one record.

Eg -

Select *

From R

Where $A = 25$; → any one record access of file.

I/O cost : $(K+1)$ block access

$$K = \Theta(\log_p n)$$

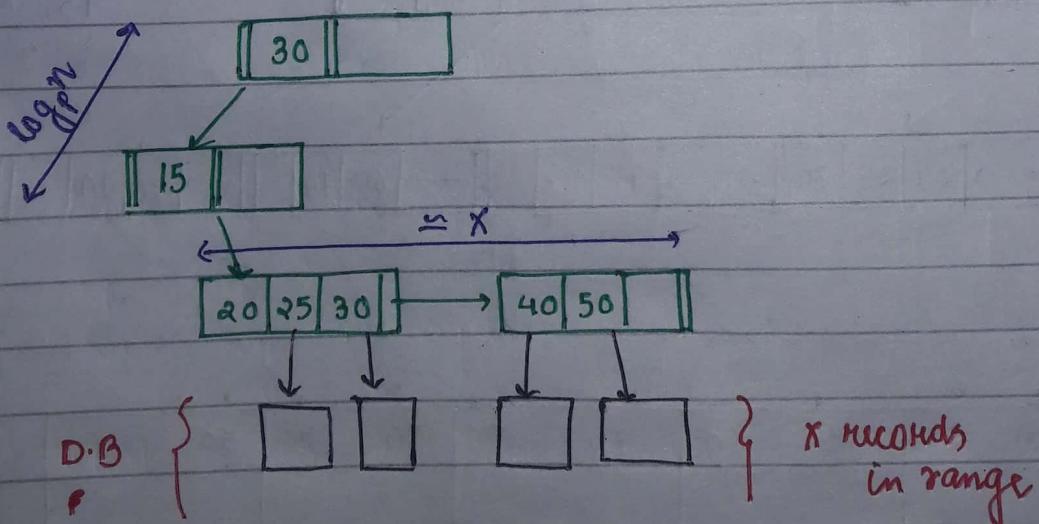
∴ I/O cost of random access query : $\Theta(\log_p n)$

- ② B^+ tree index best suitable for sequential access of range of records.

Eg - Select *

From R

Where $A \geq 25$ & $A \leq 40$.

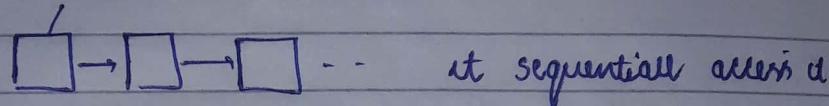


$$\therefore \text{Total access cost} = \log_p n + x + x \\ \approx \Theta(x + \log_p n)$$

for successful search.

* for unsuccessful search, there is no cost.

Note - If the file is ordered, then index is not required



- ① In B^+ tree, all keys that are required for indexing must be stored at leaf level.
- ② Every leaf node, pointed to next leaf node.
- ③ Leaf node consist (key, data pointer) pairs and one block pointer to point next leaf node.
- ④ In Internal node consist only keys and child pointer.
- ⑤ Entire tree is fully left biased (or) right biased.
- ⑥ In left biasing B^+ tree, keys stored in internal nodes are maximum key of each leaf node except last leaf node.
(No. of internal node keys are equal to No. of leaf node - 1)

Ques: Find best possible order of B Tree node / B⁺ Tree node.

① Block size = 1024 Bytes

Search key = 9 Bytes

Block pointer = 6 Bytes

Record ptr = 8 bytes.

what is best possible order of B Tree node if

Order P : max possible block ptr in B Tree node?

Soln:

Order P : Max Bp per node

Block (B Tree node)

$$P * Bp + (P-1) (keys + Rp)$$

$$P * Bp + (P-1) [Key + Rp] \leq \text{Block size}$$

$$\Rightarrow P * 6 + (P-1) (9+8) \leq 1024$$

$$\Rightarrow 23P \leq 1041$$

$$\therefore P = \left\lfloor \frac{1040}{23} \right\rfloor$$

$$= 45 \leftarrow \text{max child pointer per node}$$

Ans (order)

② Block size = 512 Bytes

Bp = Rp = 10 Bytes

Search key = 15 Bytes

What is the best possible order of B tree node if
Order P is max possible keys per node.

Soln:

order P : Max possible keys / node

Block (B Tree)

$$(P+1) B_p + P (key + R_p)$$

$$\therefore (P+1) B_p + P (key + R_p) \leq \text{Block size}$$

$$\Rightarrow (P+1) 10 + P (15 + 10) \leq 512$$

$$\Rightarrow 35P \leq 502$$

$$\therefore P = \left\lfloor \frac{502}{35} \right\rfloor$$

$$= 14 \quad \leftarrow \begin{matrix} \text{Max possible keys per node} \\ \text{Ans} \end{matrix}$$

③ Block size = 2048

$$B_p = R_p = 20 \text{ Bytes}$$

$$\text{Search key} = 40 \text{ Bytes}$$

What is best possible order of B tree node, if order P:

order P: b/w 1 to $2P$ keys in root

b/w P to $2P$ keys in other nodes?

↑
Space allocation is based
on maximum capacity.

Order P: Maximum $2P$ keys per node.

(We compute order w.r.t P)

$$(2P+1)B_p + 2P(K_{key} + R_p)$$

Block (B Tree node)

$$\Rightarrow (2P+1)B_p + 2P(K_{key} + R_p) \leq \text{Block size}$$

$$\Rightarrow (2P+1)20 + 2P(40 + 20) \leq 2048$$

$$\Rightarrow 160P \leq 2048$$

$$\begin{array}{r} 40 \\ 80 \\ 40 \\ \hline 160 \end{array} \quad \begin{array}{r} 20 \\ 20 \\ 20 \\ \hline 60 \end{array}$$

$$\therefore P = \left\lfloor \frac{2048}{160} \right\rfloor$$

$$= 12 \xleftarrow{\text{Ans}} \text{order}$$

$$④ \text{ Block size} = 1024 \text{ B}$$

$$\text{Search key} = 9 \text{ B}$$

$$\text{Block ptr} = 6 \text{ B}$$

$$\text{Record ptr} = 8 \text{ B}$$

What is the best possible order of B⁺ tree leaf node and

⑤ B⁺ tree internal node if

Order P: Max possible pointers in B⁺ tree.

Soln: Internal Node -

Order P: No. of Block pointers.

$$P * B_p + (P-1) K_{key}$$

Block (B⁺ tree node)

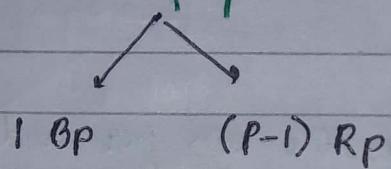
$$\begin{aligned}\therefore P * B_p + (P-1) \text{ key} &\leq \text{Block size} \\ \Rightarrow P * 6 + (P-1) 9 &\leq 1024 \\ \Rightarrow 15P &\leq 1033\end{aligned}$$

$$P = \left\lfloor \frac{1033}{15} \right\rfloor$$

$$= 68 \quad \underline{\text{Ans}}$$

leaf node -

P: No. of pointers



$$(P-1)(\text{key} + R_p) + 1 * B_p$$

$$\therefore (P-1)(\text{key} + R_p) + 1 * B_p \leq \text{Block size}$$

$$(P-1)(9+8) + 6 \leq 1024$$

$$17P = 1035$$

$$P = \left\lfloor \frac{1035}{17} \right\rfloor$$

$$= 60$$

59 key
in
leaf node 1 key
for
Block point.

Note -

If size of Block pointer, B_p = size of record pointer R_p ,
then,

order of B^+ tree leaf node = B^+ tree internal node.

⑤ Block size = 512 B

$$B_p = R_p = 10 \text{ B}$$

$$\text{Search key} = 15 \text{ B}$$

what is the max possible order of B^+ tree node if
order P is max possible keys per node.

Soln:

order P : max possible keys per node.

Here we can compute any i.e. order of internal node or leaf node.

Order of
internal
node :

$$(P_i) * B_p + (P_i \text{ keys})$$

$$(P_i) * B_p + (P_i \text{ keys}) \leq \text{Block size}$$

$$(P_i) * 10 + (P_i \text{ keys}) 15 \leq 512$$

$$10P + 15P + 10 \leq 512$$

$$25P \leq 502$$

$$P = \left\lfloor \frac{502}{25} \right\rfloor$$

$$= 20$$

OR

order of
leaf node :

$$P(\text{key} + R_p) + 1B_p$$

$$\therefore P = 20$$

⑥ Block size = 2048 B
 $BP = RP = 20 B$ → we can compute order for any node
 Search key = 40 B
 what is the best possible order of B+ tree node if
 Order P : G/W 1 to $2P$ keys in root
 G/W P to $2P$ keys in other node.
 ↑
 max limit taken

leaf
internal
node :

$$1BP + 2P(\text{keys} + RP)$$

$$\therefore 1BP + 2P(\text{keys} + RP) \leq \text{block size}$$

$$20 + 2P(40 + 20) \leq 2048$$

$$20 + 80P + 40P \leq 2048$$

$$120P \leq 2028$$

$$P = \left\lfloor \frac{2028}{120} \right\rfloor$$

$$(\text{OR}) = 16 \text{ Ans}$$

internal
node :

$$(2P+1)BP + 2P * \text{keys}$$

$$\Rightarrow P = 16 \text{ Ans}$$

Note-
 If same size blocks are allocated for B Tree node and B+ node, then

Order of B+ tree node > Order of B tree node.

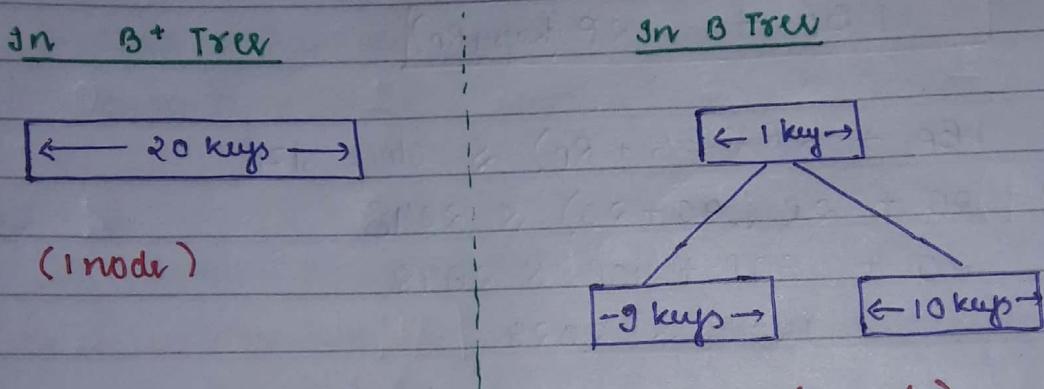
$$\left[\begin{matrix} \text{No. of keys stored in} \\ \text{B+ tree node} \end{matrix} \right] > \left[\begin{matrix} \text{No. of keys stored in} \\ \text{B tree node} \end{matrix} \right]$$

F05 Secondary Memory App.

If same size blocks are allocated for B tree index and B⁺ tree index, then

$$\left[\begin{array}{l} \text{No. of B Tree index} \\ \text{nodes for } n \text{ distinct} \\ \text{keys} \end{array} \right] \Rightarrow \left[\begin{array}{l} \text{No. of } 3+ \text{ tree index} \\ \text{nodes for } n \text{ distinct} \\ \text{keys} \end{array} \right]$$

Eq - for $n = 20$



$$\rightarrow \left[\begin{array}{l} \text{No. of levels of B Tree} \\ \text{index for } n \text{ distinct} \\ \text{keys} \end{array} \right] \geq \left[\begin{array}{l} \text{No. of levels of B}^+ \text{ tree} \\ \text{index for } n \text{ distinct} \\ \text{keys} \end{array} \right]$$

I/O cost I/O cost

B^+ tree index preferred
for DB table over B tree
because I/O cost of range
and I/O cost of random
access query is less in B^+ tree.

(B⁺ also preferred for random access)

FOR Main Memory
App.

→ If order of B tree node = order of B^+ tree node,
then,

[No. of B tree index
nodes for n distinct
keys]



[No. of B^+ tree index
node for n distinct
keys]

[No. of B tree index
levels for n distinct
keys]



[No. of B^+ tree index
levels for n distinct
keys]

Eg -

B Tree order = B^+ tree order

(P: 14 keys)



15 BP + 14 (Keys + Rp)

512 B

(P: 14 keys)



14 (Rp + keys) + BP

(or)

14 keys + 15 BP

360 B

Both are different



Not possible in secondary memory



∴ Used for MM application
as any bytes can be
allocated in it.

Min/Max keys in BTree/B⁺ tree order for given height (h) -

- ① Min possible keys in B/B⁺ tree of order P (max CP per level) and height h [0, 1, 2, ..., h]

Assume, for root : Min $\geq B_p$ and 1 key

other node : Min $\lceil \frac{P}{2} \rceil B_p$ and $(\lceil \frac{P}{2} \rceil - 1)$ key

Height	Min Node	Min B_p	Min Key
(Root) 0	1	2	1
1	$2 \lceil \frac{P}{2} \rceil$	$2 \lceil \frac{P}{2} \rceil$	$2 (\lceil \frac{P}{2} \rceil - 1)$
2	$2 \lceil \frac{P}{2} \rceil$	$2 \lceil \frac{P}{2} \rceil^2$	$2 \lceil \frac{P}{2} \rceil (\lceil \frac{P}{2} \rceil - 1)$
3	$2 \lceil \frac{P}{2} \rceil^2$	$2 \lceil \frac{P}{2} \rceil^3$	$2 \lceil \frac{P}{2} \rceil^2 (\lceil \frac{P}{2} \rceil - 1)$
...
h	$2 \lceil \frac{P}{2} \rceil^{h-1}$	$2 \lceil \frac{P}{2} \rceil^h$	$2 \lceil \frac{P}{2} \rceil^{h-1} (\lceil \frac{P}{2} \rceil - 1)$

Min possible keys in B Tree with order P and height h :

$$\Rightarrow 1 + 2 (\lceil \frac{P}{2} \rceil - 1) + 2 \lceil \frac{P}{2} \rceil (\lceil \frac{P}{2} \rceil - 1) + 2 \lceil \frac{P}{2} \rceil^2 (\lceil \frac{P}{2} \rceil - 1) \dots$$

$$\Rightarrow 1 + 2 (\lceil \frac{P}{2} \rceil - 1) \left[1 + \lceil \frac{P}{2} \rceil + \lceil \frac{P}{2} \rceil^2 + \dots + \lceil \frac{P}{2} \rceil^{h-1} \right]$$

$\underbrace{\hspace{10em}}$ GP series

$$\Rightarrow 1 + 2 \left(\lceil \frac{P}{2} \rceil - 1 \right) \left[\frac{\lceil \frac{P}{2} \rceil^h - 1}{\lceil \frac{P}{2} \rceil - 1} \right]$$

$$\Rightarrow 1 + 2 \left(\lceil \frac{P}{2} \rceil^h - 1 \right)$$

$$\Rightarrow \boxed{2 \lceil \frac{P}{2} \rceil^h - 1} \approx \Theta \left(\lceil \frac{P}{2} \rceil^h \right)$$

Also,

$$\text{Min Nodes} = \Theta \left[\left(\frac{P}{2} \right)^h \right]$$

\uparrow
Min keys for B tree.

Min possible keys in B^+ tree with order P and height (h) -

Only nodes at leaf level is counted
as all other are separator value



$$\boxed{2 \lceil \frac{P}{2} \rceil^{h-1} \left(\lceil \frac{P}{2} \rceil - 1 \right)} \approx \Theta \left(\lceil \frac{P}{2} \rceil^h \right)$$

② Max possible keys in B/B^+ tree with order P and height h -

Assume, Max P B_P and $(P-1)$ keys per node.



Height	Max node	Max BP	Max keys
0	1	P	$P-1$
1	P	$P \times P$	$P \times (P-1)$
2	P^2	$P^2 \times P$	$P^2(P-1)$
3	P^3	$P^3 \times P$	$P^3(P-1)$
\vdots	\vdots	\vdots	\vdots
h	P^h		<u>$P^h(P-1)$</u>

P^h

$$\text{Max node} = \left(\frac{P^{h+1} - 1}{P-1} \right)$$

$$\text{Max key} = \left(P^{h+1} - 1 \right)$$

Max keys in B^+ tree with order P and height (h) = $P^{h+1} - 1 \approx \Theta(P^h)$

Also,

Max key in B^+ tree with order P and height (h) = $P^{h+1} - 1$

Height of B tree with order P and n distinct key.

a) Min Height

(Max possible key per node)

$$n = p^{h+1} - 1$$

$$h = \log_p(n+1) - 1 \quad \approx \quad O(\log_p n)$$

b) Max Height

(Min possible key per node)

$$n = 2 \left\lceil \frac{p}{2} \right\rceil^h - 1$$

$$\frac{n+1}{2} = \left\lceil \frac{p}{2} \right\rceil^h$$

$$h = \log_{\frac{p}{2}} \left(\frac{n+1}{2} \right) \quad \approx \quad \log_{\frac{p}{2}} n/2 \approx O(\log_p n)$$

Note -

Keys of B/B⁺ tree : $O(\lceil \frac{p}{2} \rceil^h) \dots \xrightarrow{\text{to}} O(p^h)$

Height $\begin{cases} \text{B tree} : & \log n \\ \text{B}^+ \text{ tree} : & \frac{p}{2} \end{cases}$ to $\log n/2$

Ans: Order P : b/w 2 to P block ptr in root node
 b/w $[P/2]$ to P block ptr in other internal node
 b/w $[P/2]-1$ to $(P-1)$ keys in leaf node
 of B Tree / B^+ tree.

- ① What is the min keys and nodes in B Tree / B^+ tree of order P: 5 and level: 4?
- ② What is the max keys and nodes in B Tree / B^+ tree of order P: 5 and level 4.

design
From
root
to
leaf

Soluⁿ:

① Min Keys / Nodes of level 4 and order P: 5

level	Min node	Min BP	Min Key
1	1	2	1
2	2	$2 * [P/2]$ $\Rightarrow 2 * [5/2] = 2 * 3$	$2 * 2$
3	3	$6 * 3$	$6 * 2$
4	<u>18</u>	-	<u>$18 * 2$</u>
	<u>27</u>		
			<u>Min key in B tree = 53</u>

$$\text{Min key in } B^+ \text{ tree} = 36 \quad (\text{only last level i.e leaf node } 18 * 2)$$

$$\text{Min nodes in } B/B^+ \text{ tree} = 27$$

② Max keys and nodes.

level	Max nodes	Max BP	Max keys
1	1	5	4
2	5	5×5 xP	5×4
3	25	25×5	25×4
4	<u>125</u>	-	<u>125×4</u>
	<u>156</u>	Max key in B tree → <u>624</u>	

$$\text{Max nodes in } B/B^+ \text{ tree} = 156.$$

$$\begin{aligned} \text{Max key in } B^+ \text{ tree} &= 125 \times 4 \\ &= 500 \quad (\text{only last i.e leaf}) \end{aligned}$$

Bulk loading B⁺ Tree -

(Construction of B⁺ Tree in bottom up Approach)

- It is the design of B⁺ tree from the leaf nodes to the root node.

Procedure :

- Sort the keys in ascending order which are used for indexing.
- Design the leaf nodes and, distribute the keys into leaf nodes (No. of leaf nodes)

③ Design internal node

If 'X' nodes at level (l) , then
 X child pointers at level ($l-1$)

Distribute 'X' child pointer into nodes (No. of nodes at ($l-1$))

Repeat step ③ until root.

Ques: No. of keys for index = 2500

order $P = 7$ (max ptr per node of B^+ tree)

[Assume: 2 to P pointers for root

$\lceil P/2 \rceil$ to P pointers for other nodes]

a) How many min levels of B^+ tree index?
⇒ (Min nodes)

b) How many max levels of B^+ tree index?
(max nodes)

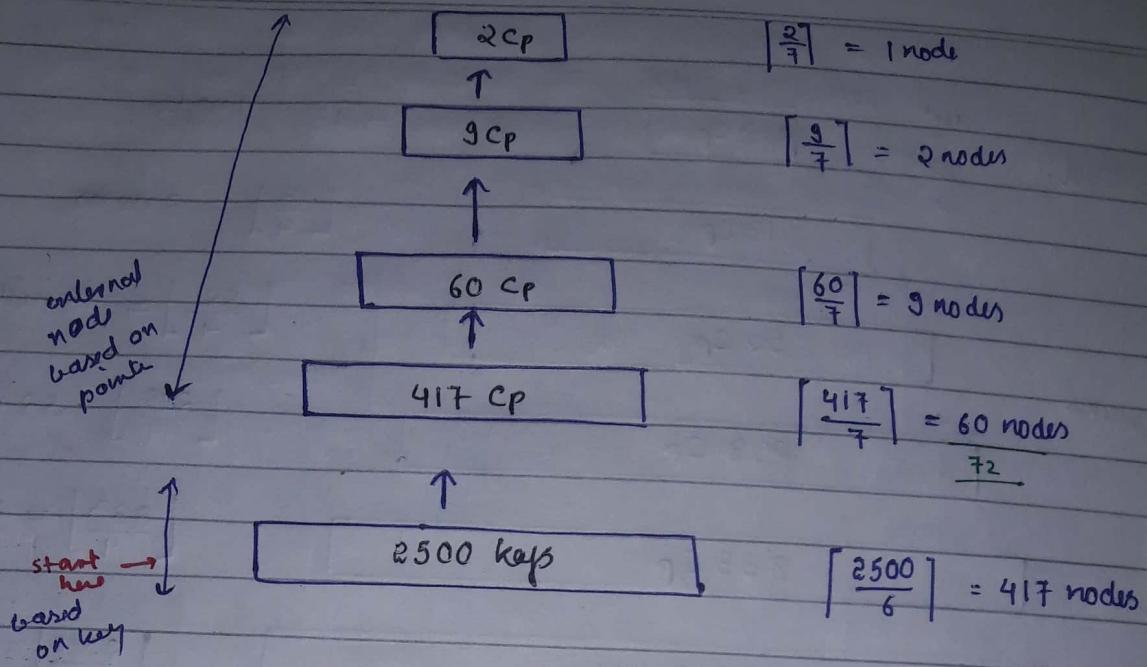
II solve using Bulk loading B^+ tree. (when leaf to root)

Soluⁿ:

① Min levels of B^+ Tree index:
(Max node fill factor)

$$\text{MAX: } P = 7 \text{ ptr}$$

$$\begin{aligned} \text{key} &= P-1 \\ &= 7-1 \Rightarrow 6 \text{ keys.} \end{aligned}$$



\therefore Min index level = 5 Ans

$$\begin{aligned} \text{Min index blocks} &= 417 + 72 \\ &= 489 \text{ nodes. Ans} \end{aligned}$$

② Max level of B+ tree index

Max fill factor Root : Min = 2 CP and ~~Max~~ 1 key

$$\begin{aligned} \text{Other nodes} : \text{Min} &= \lceil p/2 \rceil \text{ and } \cancel{\text{Max}} \text{ key} = p-1 \\ &= \lceil 7/2 \rceil \\ &= 4 \text{ CP} \end{aligned}$$

$$\left\lfloor \frac{2500}{4} \right\rfloor = 833$$

We take floor value

i.e.

If we take ceil value

$$\left\lceil \frac{10}{3} \right\rceil = 4 \text{ nodes}$$

3 key

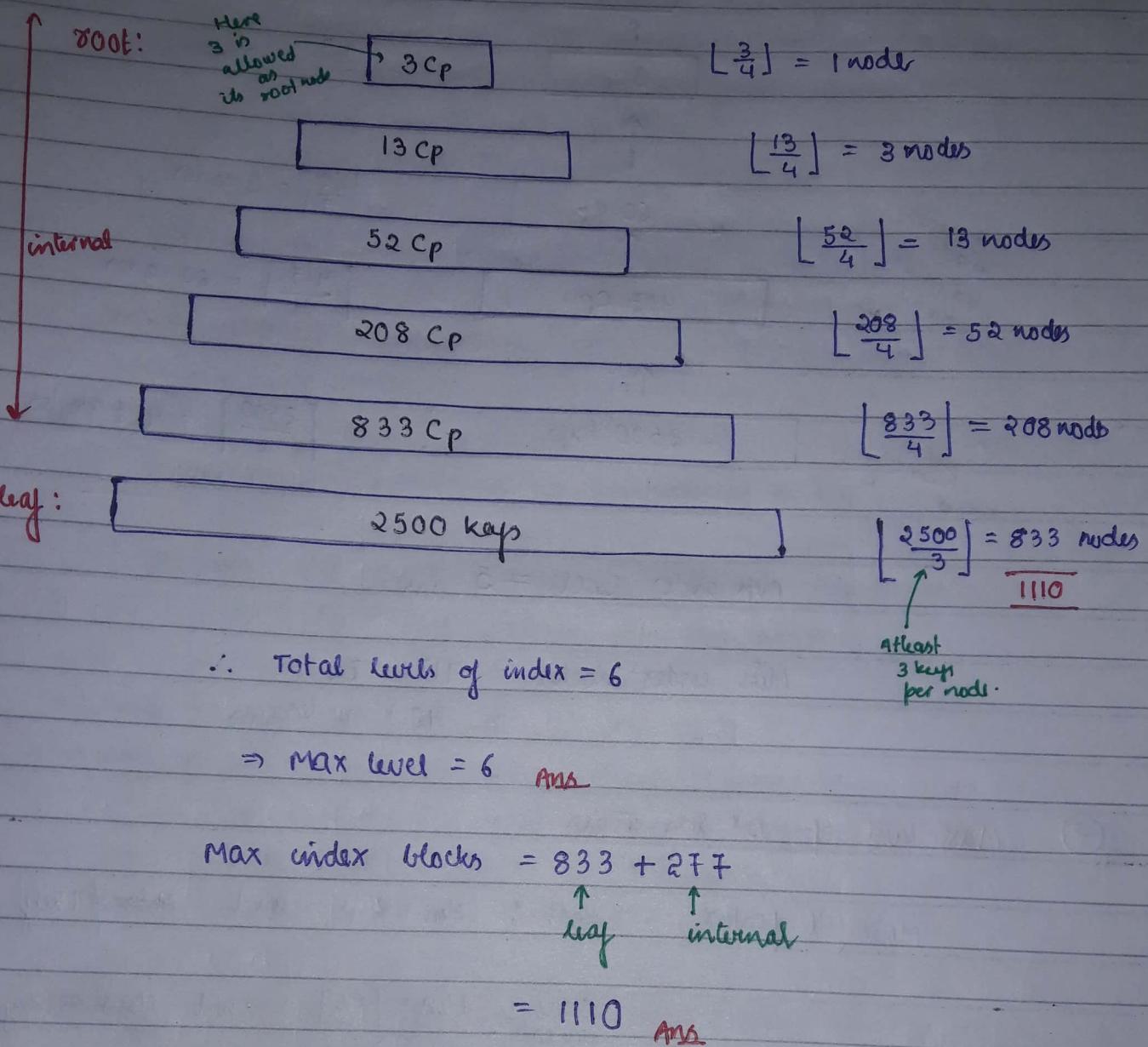
3 key

3 key

1 key

This is not allowed

\therefore we take ceil value



Notes

B-Tree Problem -

- { ① Find Min/Max keys of given l levels
 - ② " " " " levels of given n keys }
- design root to leaf

B+ tree Problem -

- ① Find min/max key of given l levels
design root to leaf
- ② Find min/max levels of given n keys?
design leaf to root (Bulk loading B+ tree)

Join Algorithms —

- ① Nested loop Join Algo
- ② Block nested loop Join Algo

Nested loop Join Algo :

Let relation R with n tuples occupied in x blocks
relation S with m tuples occupied in Y blocks.

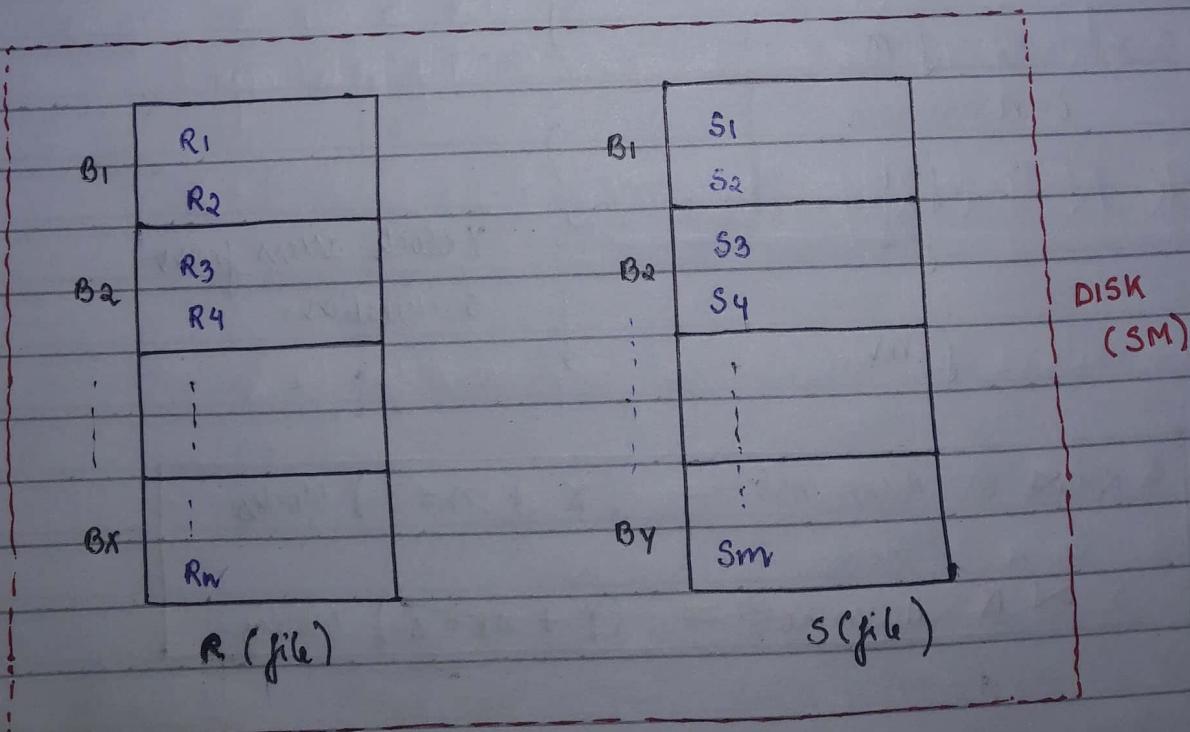
$R \bowtie S \Rightarrow$

```
for (i=1; i≤n; i++)  
  for (j=1; j≤m; j++)  
    comp (Ri, Rj) join condition
```

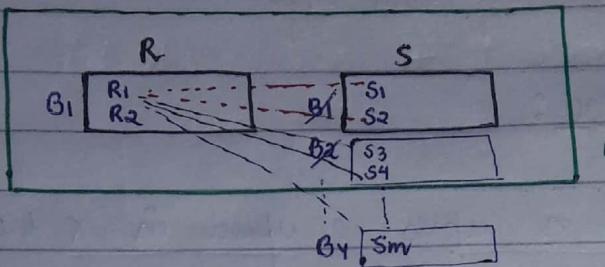
record no.
of R

Record no.
of S

drawback — It takes more access cost to join if, the main memory space allocated to join is limited.



Case 1: Assume only 2 blocks of MM allocated for Join
 (At any time for one block of R data and one block
 of S data can be stored in MM)



Main memory
space for Join

for $i = 1$

$j = 1$

$j = 2$

:

$j = m$

γ block access from
S relation

$i = 2$

$j = 1$

$j = 2$

:

$j = m$

γ blocks access from
S relation

$i = n$

$j = 1$

$j = 2$

:

$j = m$

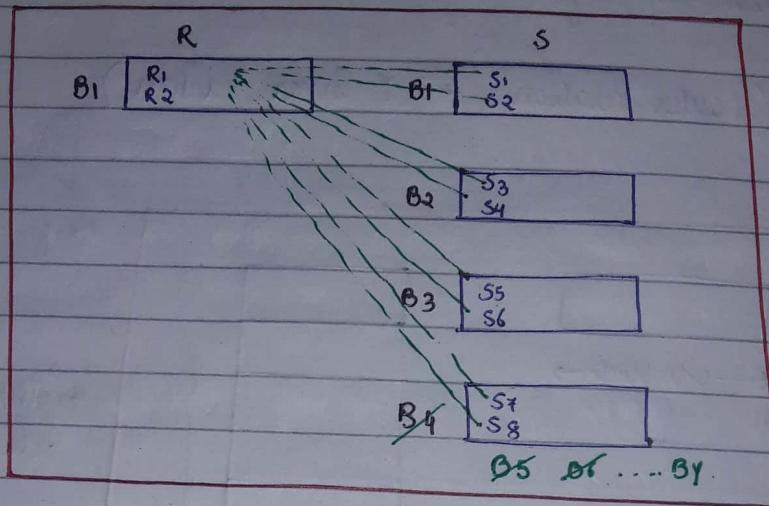
γ block access from
S relation.

$$R \bowtie S \text{ Access cost} = (x + n * \gamma) \text{ blocks}$$

$$S \bowtie R \text{ Access cost} = (\gamma + m * x) \text{ blocks}$$

Case 2 : Assume 5 blocks of MM allocated for join

a) $R \bowtie S$ Access cost :



for $i = 1$

$j = 1$
 $j = 2$
 \vdots
 $j = m$

$\} Y$ block of S relation accessed

$i = 2$

$j = 1$
 $j = 2$
 \vdots
 $j = m$

$\} (Y-3)$ blocks of S relation accessed

$i = n$

$j = 1$
 \vdots
 $j = m$

$\} Y-3$ blocks of S relation accessed

$$\textcircled{1} \quad \text{RMS Access cost} = X + Y + (m-1)(Y-3) \text{ blocks}$$

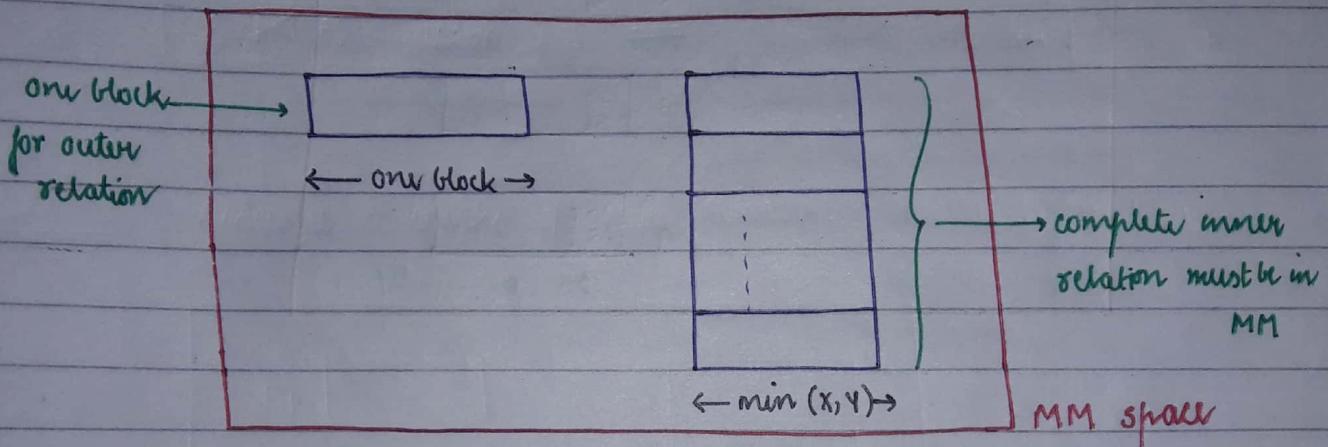
$$\textcircled{2} \quad \text{SMR Access cost} = Y + X + (m-1)(X-3) \text{ blocks}$$

same for both the algo (Nested and Block Nested loop join)

Case 3:

How many min main memory blocks are required to Join R, S without any repeated data access from Disk to main memory using Nested loop join algo.

(1 outer relation + Y inner relation)



$$\text{Min MM space} = \min(X, Y) + 1$$

∴ { $\min(X, Y) + 1$ } block of MM is required to avoid repeated data access from disk to MM to perform join of R and S using Nested loop join Algo.

Block Nested Loop Join Algo

- Here, instead of records, the loop runs for blocks
- Rel R with X blocks of record and Rel S with Y blocks of record.

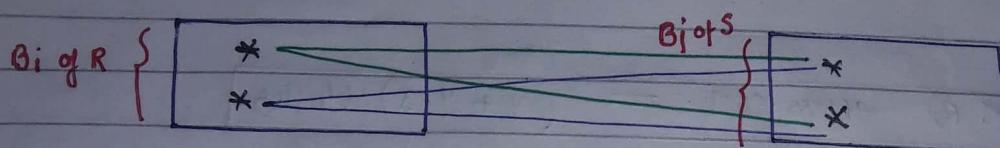
$R \bowtie S \Rightarrow$

for ($i=1 ; i \leq X ; i++$)

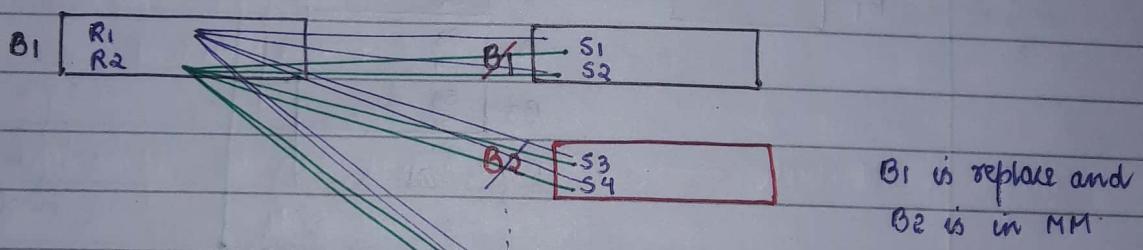
for ($j=1 ; j \leq Y ; j++$)

JOIN (ith block record of R,

jth block record of S)



Case 1: Only 2 blocks of MM allocated for JOIN



for $i = 1$

$j = 1$
 $j = 2$
 $j = Y$

} Y blocks of S data
accessed

$i = 2$

$j = 1$
 $j = 2$
 $j = Y$

} Y blocks accessed from S

$i = X$

$j = 1$
 $j = Y$

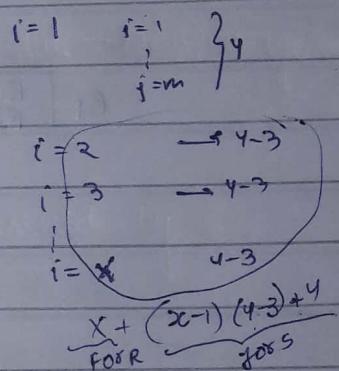
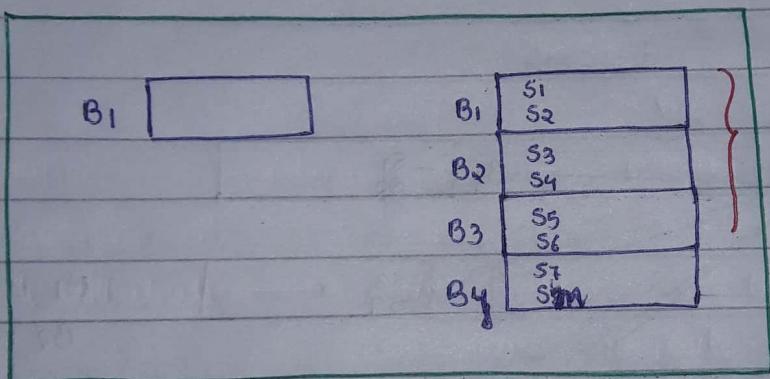
} Y block accessed from S

Advantage — Less access cost as compared to Nested Loop Join algo, if main memory space is limited.

$$R \bowtie S \text{ access cost} = (X + X \cdot Y) \text{ blocks}$$

$$S \bowtie R \text{ access cost} = (Y + Y \cdot X) \text{ blocks}$$

Case 2: Assume 5 blocks of Main Memory allocated for Join.



$$R \bowtie S \text{ access cost} = X + (X-1) * (Y-3) + Y$$

$$S \bowtie R \text{ access cost} = Y + (Y-1)(X-3) + X$$

ER Model

- ER diagrams are used to represent high level database designs (diagrammatic representation of DB design)

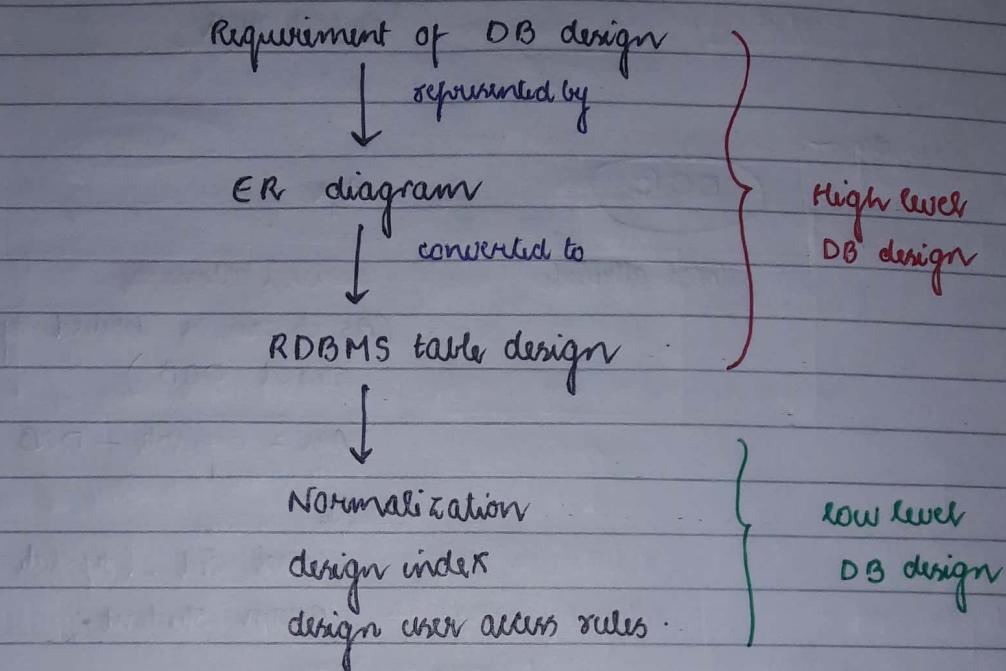


Fig: Database development steps

Main components in ER model -

- ① Attribute
- ② Entity sets
- ③ Relationship sets.

Attribute :



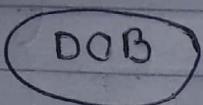
• Key attribute →

• multivalued attribute →

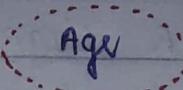
- derived attribute → value of attribute is derived from value of stored attribute

Eg -  'dotted oval'

Eg -



stored attribute



derived attribute

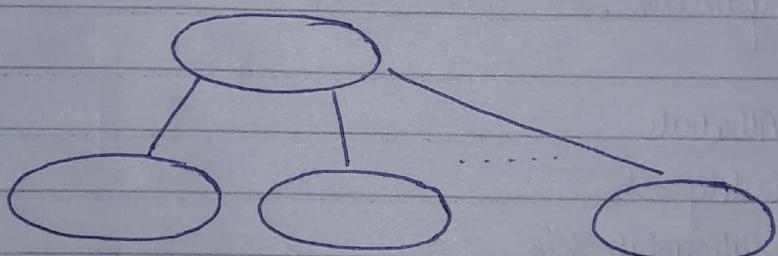
(As it can be derived by stored DOB)

$$\text{Age} = \text{Sysdate} - \text{DOB}$$

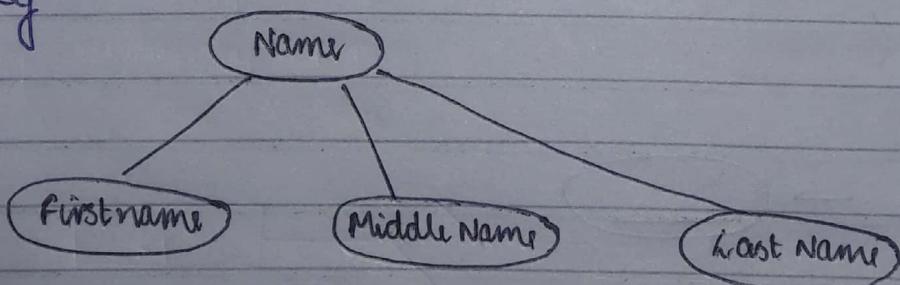
In SQL:

```
select Sid , (sysdate - DOB) as age
From student;
```

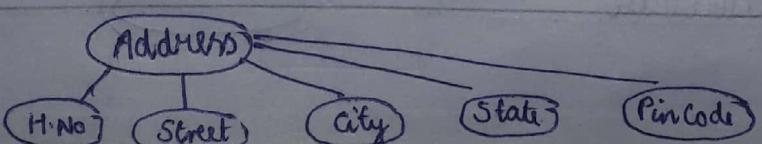
- Composite Attribute → Attribute that can be represented by 2 or more attribute.



Eg -



(or)



Tables are designed based on type of query performed -

- ① If mostly application uses 'full address only'

stud (sid, address)

→ take address as a single field

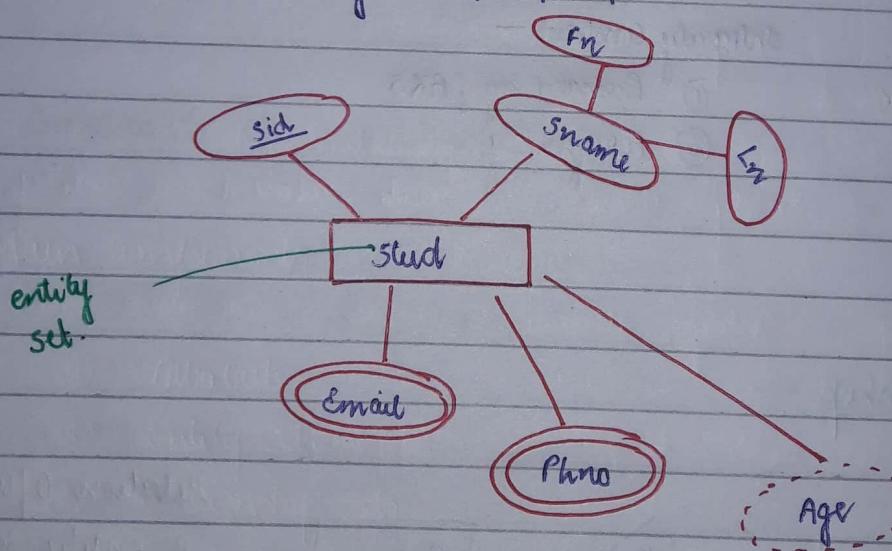
- ② If mostly application uses city, state, pincode

stud (sid, Hno, street, city, state, Pin)

together address.

Entity set —

Set of similar entity (object / tuples)

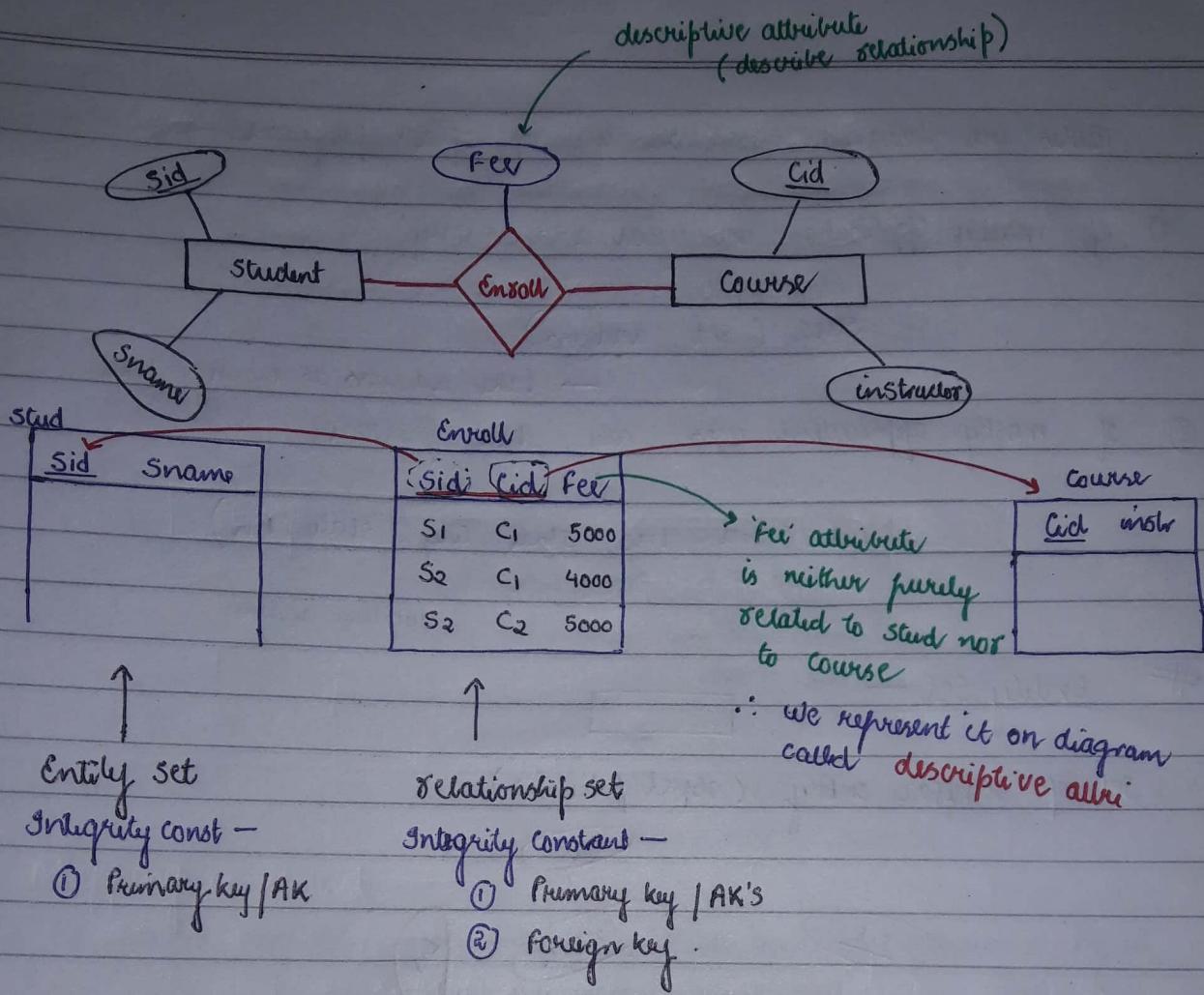


Here, 'stud' is entity set.

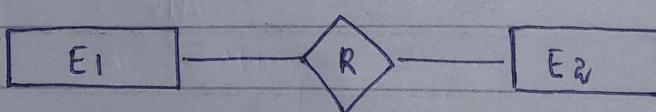
Relationship set —



Used to relate two or more entity set.

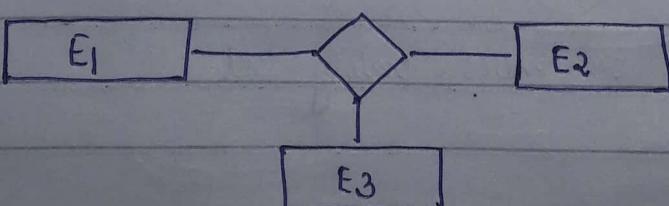


Binary relationship —



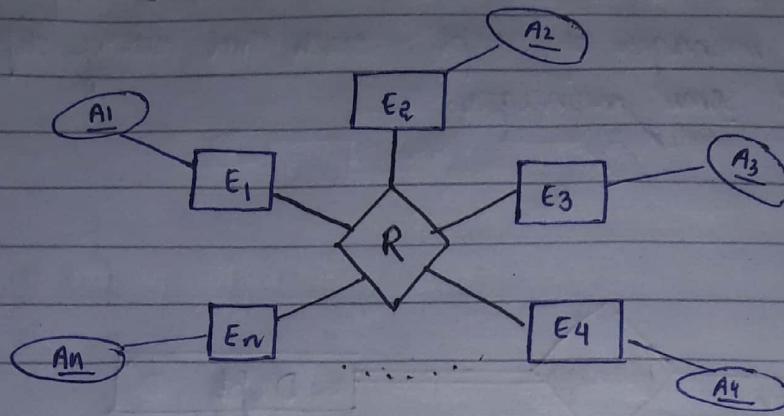
relation b/w
2 entity set

Ternary relationship set —



relation b/w
3 entity set

n-ary relationship set —



$R (A_1, A_2, \dots, A_n)$

some descriptive attr.

Min attributes needed = n

Foreign key = n

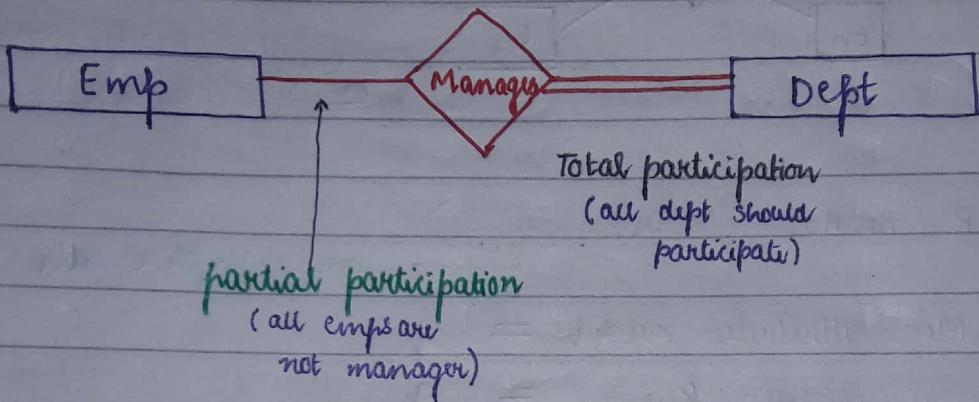
~~#~~ constraints of relationship set —

- ① Mapping (cardinality)
- ② Participation

Participation —

- If every entity of entity set is related to relationship set (must cond.) , then it is called **Total Participation** (i.e must be 100% participation) denoted by =
- Otherwise, called **Partial Participation** ($< 100\%$ participation) denoted by -

Ex - Emp and Dept are entity sets, if
 manages is relationship set used to relate emp,
 who are manager of dept, such that every dept
 must have some manager.
 (at least 1)



Mapping Constraint —

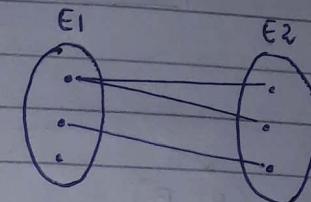
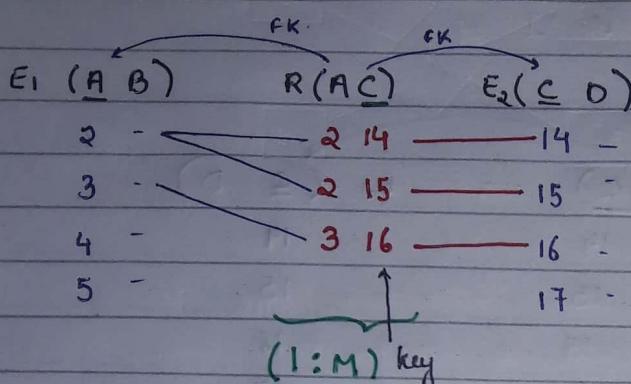
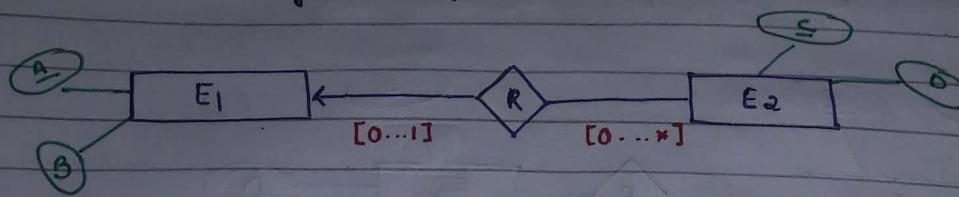
one - Entity allowed to relate almost one entity (\rightarrow)

many - Entity allowed to relate 0 or more entities (\rightarrow)

* Binary relationship set possible mapping —

- | | | |
|--|---|--|
| ① One : Many mapping
② Many : One "
③ Many : Many "
④ One : One " | } | 'Candidate keys' of
relationship set depends
on mapping constraint |
|--|---|--|

⇒ One : Many mapping relationship betw (R) w/o E₁, E₂ entities:

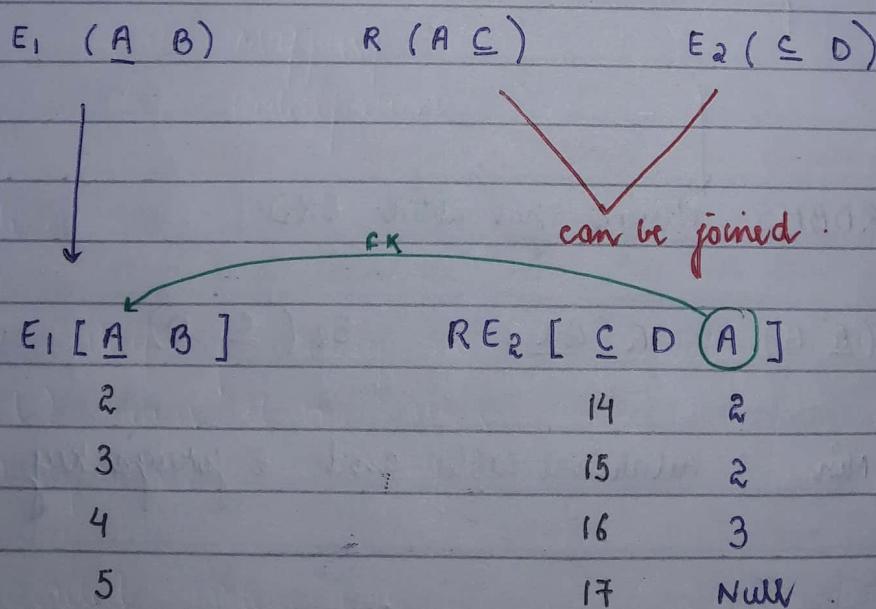


one object of E₁ relate many E₂
 one " " E₂ " only one E₁
 (Many)
 (One)

candidate key of rel set R (AC) = {C} of
 for 1:M mapping

E₁—E₂ many
 E₂—E₁ one

⇒ RDBMS tables design for ERD —



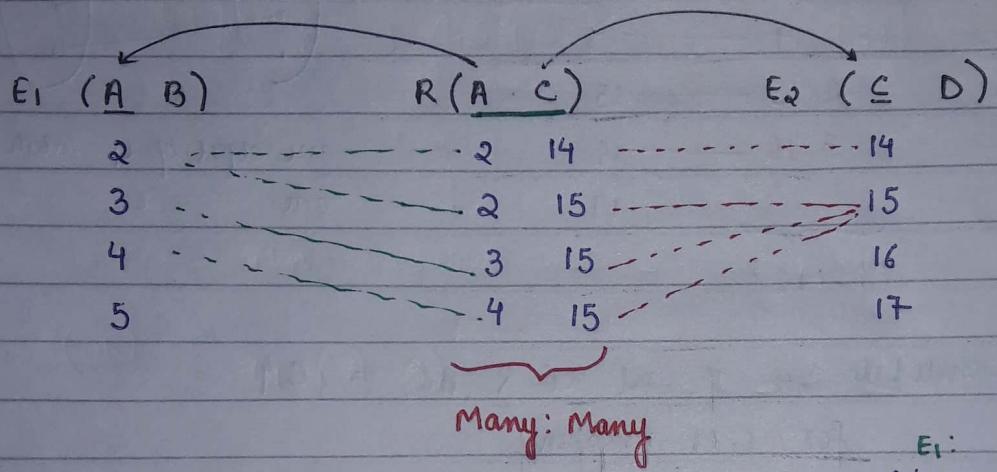
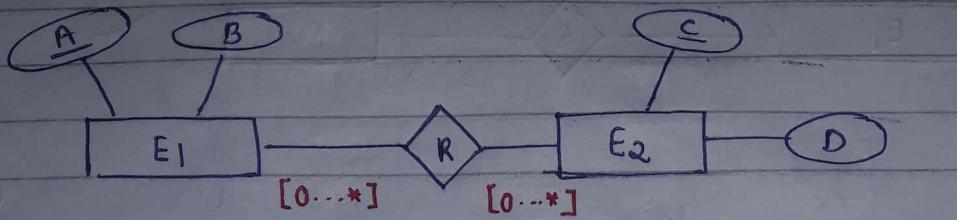
∴ Min 2 tables with 1 foreign key

→ Many : One mapping
 Similar to above just



[cand key = A]

→ Many : Many mapping



Candidate key of rel set $R(AC) = \{AC\}$
for M:M relation

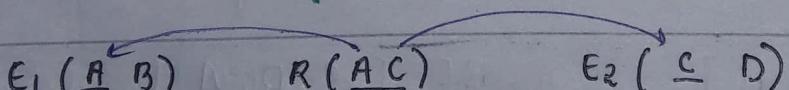
E1:
2 has many mapping

$2 \rightarrow 14$ many

E2:
15 has many mapping

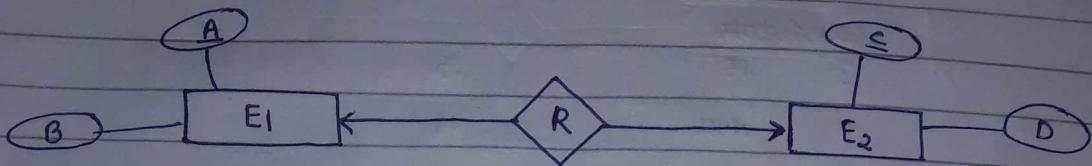
$15 \rightarrow 2$ many
 $15 \rightarrow 3$ many
 $15 \rightarrow 4$ many

RDBMS design for above ERD



Min 3 relational table and 2 foreign key

→ One : One Mapping relationship b/w E_1 and E_2 with Partial participation at both ends.



$E_1 (A \ B)$	$R (A \ C)$	$E_2 (C \ D)$
2	2 15	14
3	3 17	15
4	5 14	16
5	5 16 ↑ key 2 14	17
	\times Not allowed	

∴ candidate keys of relationship set $R(A \ C)$ is = {A, C} for 1:1 mapping



RDBMS design for above ER diagram

Even if A and C both are key, still we cannot join all into a single table because of partial participation (as it causes some violation i.e. NULL values occur) over left side (E_1 side) and right side (E_2 side).

A	B	C	D
2	-	15	-
3	-	17	-
4	-	NULL	NULL
5	-	14	-
NULL	NULL	16	-

cand key = {A, C}

\times no primary key
(i.e. no field whose value does not contain NULL)

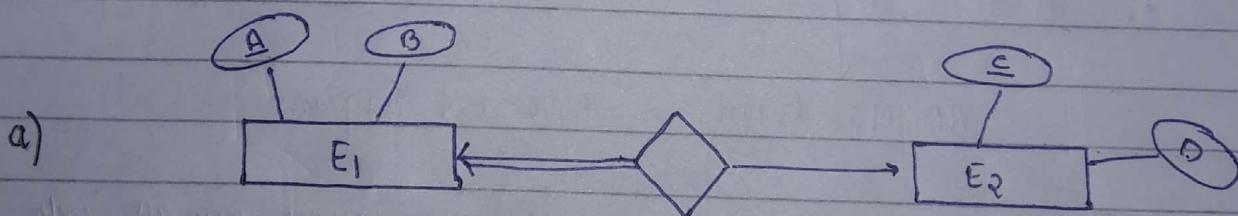
	$E_1 R (A \underline{B} C)$	$E_2 (C \underline{D})$
2	15	16
Candidate		
3	17	15
4	Null	16
5	14	17

candidate key = { A, C }
 ↑ ↑ AK.
 Primary key

∴ Min 2 tables and 1 foreign key.

(OR) $E_1 (\underline{A} B)$ $R E_2 (\underline{C} D \underline{A})$

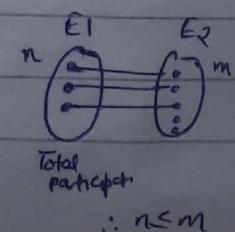
→ One: One Mapping relationship b/w E_1, E_2 entity set with at least one end with total participation.

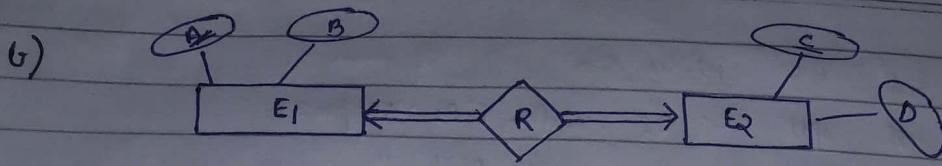


① $E_1 R E_2 (\underline{A} B \underline{C} D)$
 One relational table
 alternate key
 Primary key.

$E_1 \rightarrow C \Leftarrow \text{Non null}$
 \uparrow
 Total participation

② E_1, E_2 with $n & m$ entities respectively.
 'R' rel with 1:1 and Total participation at E_1 ,
 $(n \leq m)$ guaranteed





One relational table,

$E_1, R, E_2 \quad (A \ B \ C \ D)$

candidate key = {A, C}

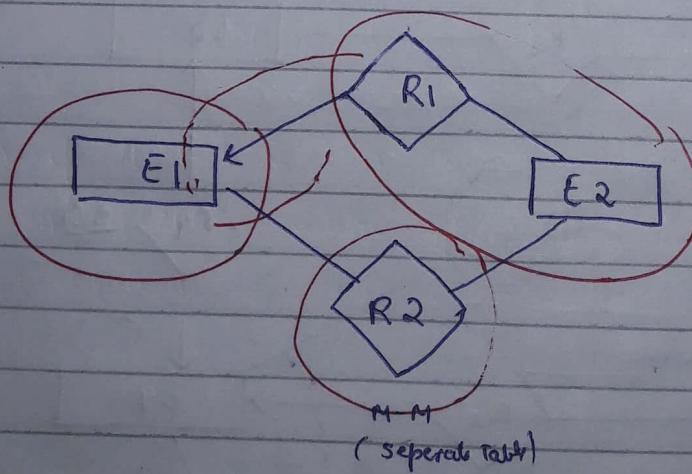
↑
key with
no nulls

At both sides there is total participation.

$$\therefore n = m$$

13/11/17

Ques: E_1, E_2 entity sets. R_1, R_2 relationship sets b/w E_1 and E_2 with 1:M and M:N mapping.
How many min relational tables required.



3 tables

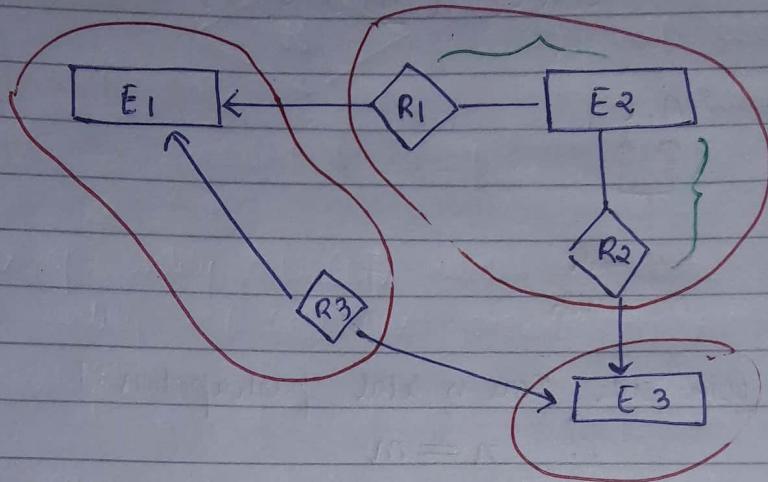
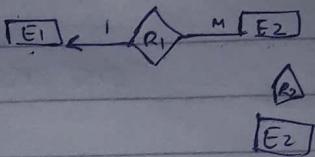
Ques: E₁, E₂, E₃ entity sets

R₁ rel b/w E₁, E₂ with 1:M mapping

R₂ rel b/w E₂, E₃ with M:1 "

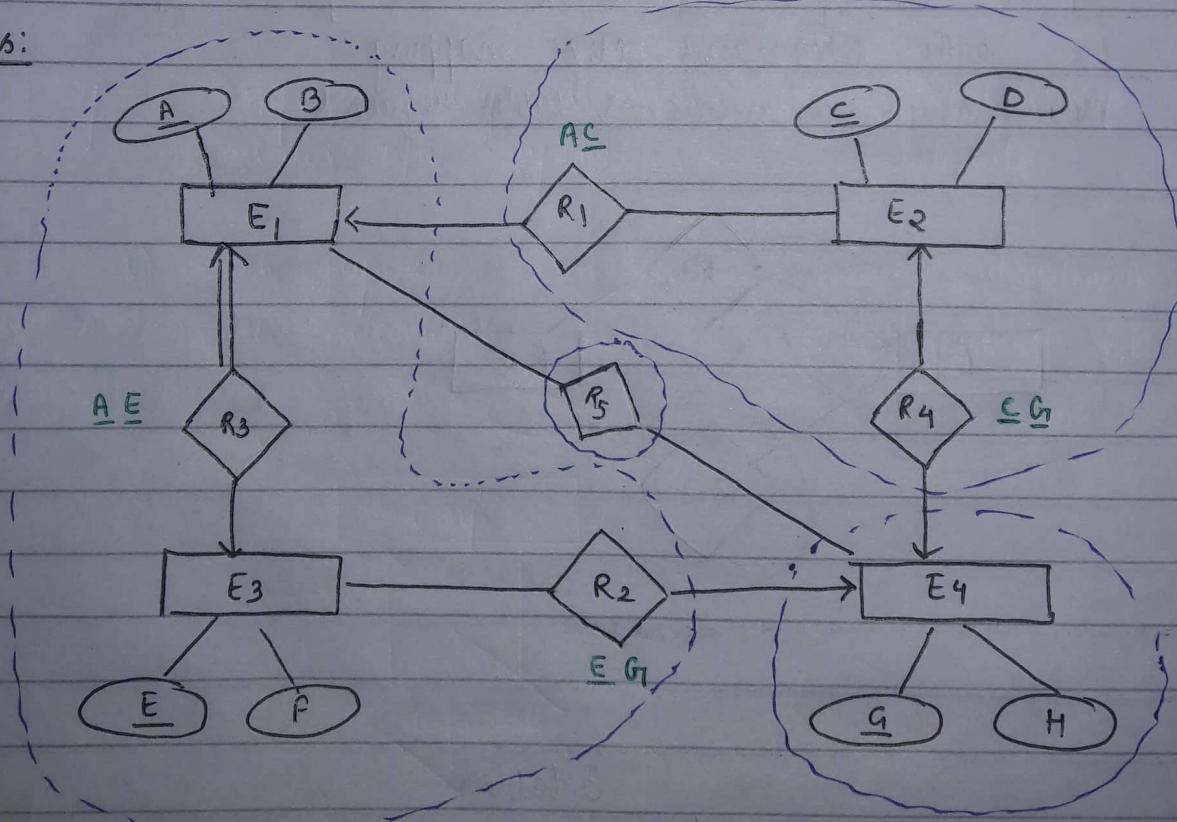
R₃ .. " E₁, E₃ " 1:1 "

How many min rel table required.



∴ 3 relational tables are required.

Ques:



- i) Main rel tables?
- ii) No. of FK in RDBMS design?
- iii) " " attributes in " " ?

Soln:

$E_1 R_3 E_3 R_2 [\underline{A} \ B \ \underline{E} \ F \ \underline{(G)}]$

Primary key $\rightarrow \underline{E}$
 Alternative key $\rightarrow \underline{A}$
 Foreign key $\rightarrow \underline{(G)}$

$R_1 E_2 R_4 [\underline{C} \ D \ \underline{A} \ \underline{(G)}]$

Primary key $\rightarrow \underline{C}$
 Alternative key $\rightarrow \underline{G}$
 F.K $\rightarrow \underline{(A)}, \underline{(G)}$

$E_4 [\underline{G} \ H]$

Primary key $\rightarrow \underline{G}$

$R_5 [\underline{A} \ \underline{(G)}]$

Primary key $\rightarrow \underline{AG}$

F.K $\rightarrow \underline{(A)}, \underline{(G)}$

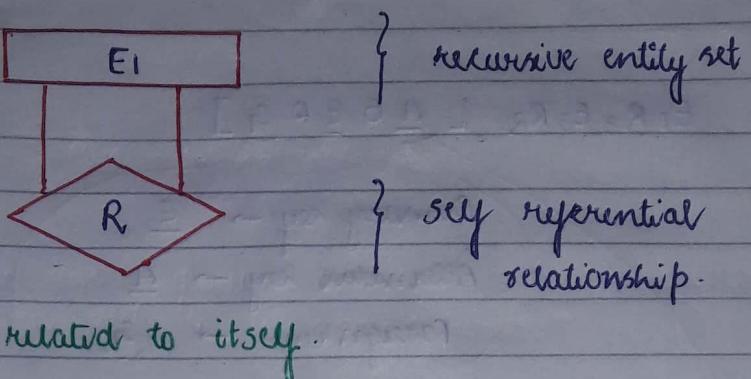
∴

Main rel table Aug = 4 tables. Ans

No. of F.K = 5 F.K. Ans

Total no. of attributes = $5 + 4 + 2 + 2$
 $= 13$ Ans

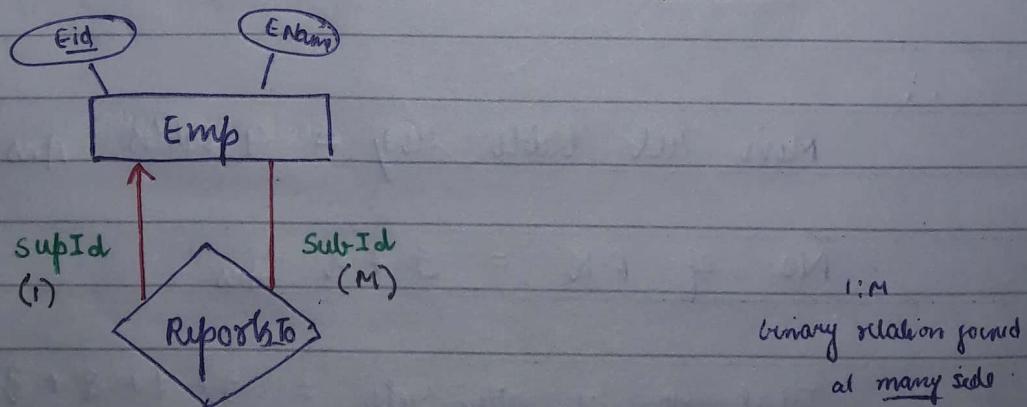
Self Referential Relationship set —



- Entities of entity set (E_1) related to same other entity of same entity set (E_1) is called as **self referential relationship set**.
- It is same as a **binary relationship**
- 1:M , M:1 , 1:1 , M:N mappings possible.

Ex - Emp is entity set and reports-to relationship set related b/w supervisor and subordinate .

- a) Each supervisor can supervise many subordinate and each subordinate reports to one supervisor (1:M)



Emp

ReportsTo (1:M)

Eid	Ename	SupID	SubID
e1	A	E1	E2
e2	B	E1	E3
e3	A	E2	E4
e4	C		

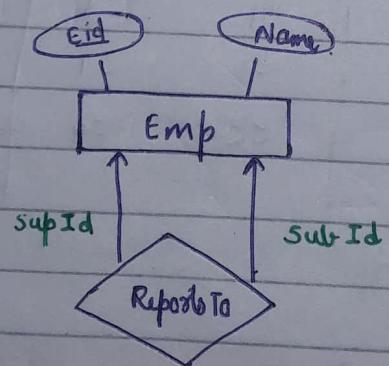
↓ RDBMS design

eid ← FK

eid	ename	supID
e1	A	Null
e2	B	E1
e3	A	E1
e4	C	E2

∴ Min 1 relational table
and
1 foreign key for ERD

- (b) each supervisor should supervise one subordinate and
each subordinate " report one supervisor (1:1)



Emp

Reports To (1:1)

Eid	Name	supID	SubID
e1	A	E1	E2
e2	B	E2	E4

Here both
supID and subID
is unique
(as they both
can belong only
1 relation)

<u>Eid</u>	<u>Ename</u>	<u>SupId</u>
e1	A	Null
e2	B	e1
e3	A	e3

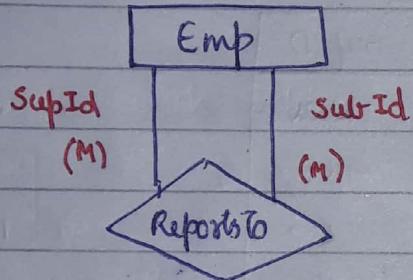
<u>Eid</u>	<u>Ename</u>	<u>SubId</u>
e1	A	Null
e2	B	e3
e3	A	e4

cand : {Eid, supId}

Min relational Table = 1

cand key = {Eid, subId}

- c) each supervisor can supervise many subordinates and each subordinate can report to many supervisors (M:M)



Emp

<u>Eid</u>	<u>Ename</u>
e1	A
e2	B
e3	A
e4	C

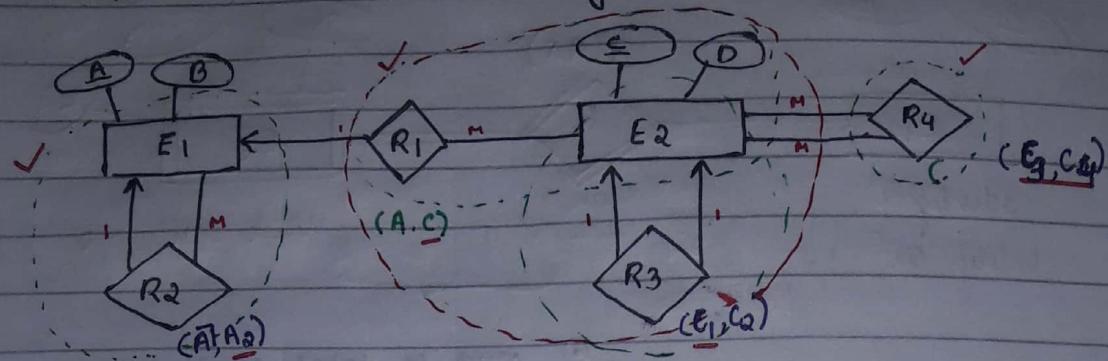
Reports To (M:M)

<u>SupID</u>	<u>SubID</u>
e1	e3
e1	e4
e2	e4

Min rel table = 2

1:M → composite key (key of many sides)
 1:1 → Both keys
 M:1

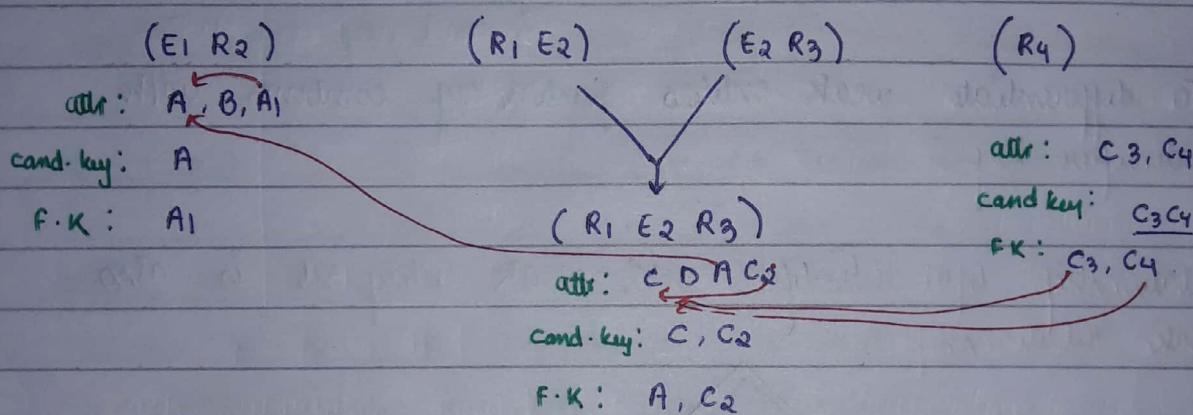
Ques: Find min relational table from ER diagram.



a) Min rel table $\rightarrow 3$

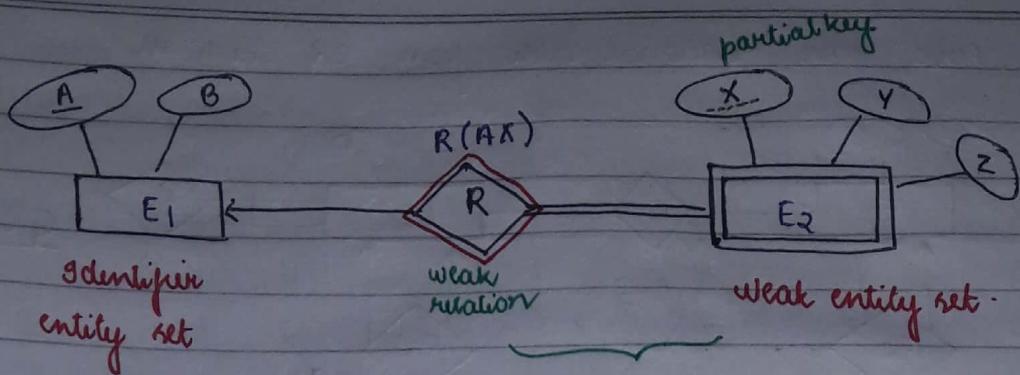
b) NO. of FK in RDBMS design. $(1+2+2) \Rightarrow 5$

c) Total no. of attributes in RDBMS design. $\Rightarrow 9$.



Weak Entity Set -

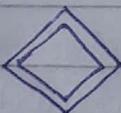
- Entity set with no key.
- Attributes of entity sets not sufficient to differentiate records.
- Represented by
- Entities of weak entity set depends on some strong entity set called as identifier (Owner Entity Set)



① Must be total participation at weak entity set end.

② Mapping b/w identifier to weak entity set must be 1:M

- ③ To differentiate weak entities partial key combines with identifier key.
- ④ Relationship b/w identifier and weak entity set is also weak relationship



RDBMS design :

$E_1 (A \ B)$

$E_2 \ R (AX \ YZ)$

Primary key - {AX}

Foreign key (A) ref to E1

- ⑤ Weak entity set combines with relationship set in RDBMS design which consist of proper candidate key.

For above diagram \rightarrow 2 rel table

- ◎ Multivalued attribute set and weak entity set allowed in ER diagram but, not allowed in relational database (RDBMS)

Multivalued Dependency and 4NF -

→ If relation R is in BCNF
and

Redundancy exists in relation R,
then

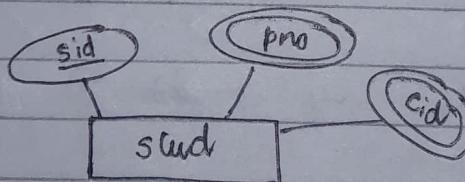
At least two or more multivalued attributes are converted
into single valued attribute in R (forms redundancy over MVD's)

→ If relation R is in BCNF (i.e no redundancy over FD's)
(and)

Almost one multivalued attribute converted into single valued
attribute in R (no redundancy over MVD's)

then, relation R is free from redundancy. (i.e no chance of
redundancy over FD's and MVD's)

Eg -



stud (Sid Pno Cid)

S₁ P₁/P₂ C₁/C₂/C₃

S₂ P₃ C₃



candidate key (Sid Pno Cid)

stud (Sid Pro Cid)

S ₁	P ₁	C ₁	}
S ₁	P ₁	C ₂	
S ₁	P ₁	C ₃	
S ₁	P ₂	C ₁	}
S ₁	P ₂	C ₂	
S ₁	P ₂	C ₃	
S ₂	P ₃	C ₃	

same data/
repeated 3
times

S ₁	C ₁	}
S ₁	C ₂	
S ₁	C ₃	

repeated 2
times

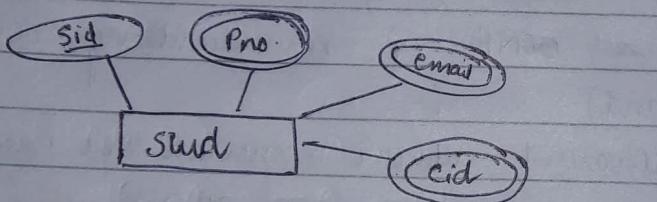
① Stud rel is in BCNF.

Redundancy over MVD's

can be solved by
dividing it in 2
tables

R₁(Sid Pro) R₂(Sid Cid)

Eg -



stud (Sid Pro Email Cid)

S₁ P₁/P₂ e₁/e₂/e₃ c₁/c₂/c₃

S₂ P₃ e₄ C₃



stud (Sid Pro email Cid)

2x3x3
⇒ 18 tuples
for S₁

S ₁	P ₁	e ₁	C ₁
S ₁	P ₁	e ₂	C ₁
S ₁	P ₁	e ₃	C ₁

stud rel in BCNF .

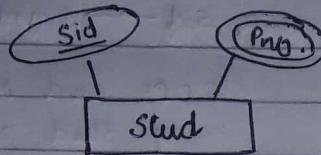
S₂ P₃ e₄ C₃

$s_1 \quad p_1$
 $s_1 \quad p_2$

same data
repeated 2 times

∴ Redundancy

Eg -



Stud (Sid Phno)

$s_1 \quad p_1/p_2$
 $s_2 \quad p_2$



Stud (Sid Phno)

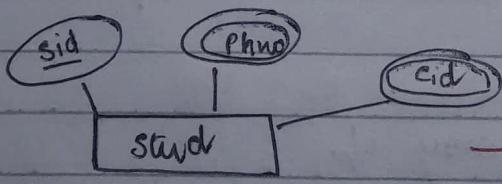
$s_1 \quad p_1$
 $s_1 \quad p_2$
 $s_2 \quad p_2$

BCNF relation and.
No redundancy over MVD's

Multivalued dependency — (MVD)

If multi-valued attribute (Y), which depends on key attribute (X),
if converted into single valued attribute in R , then

$X \rightarrow \rightarrow Y$ MVD exists in R



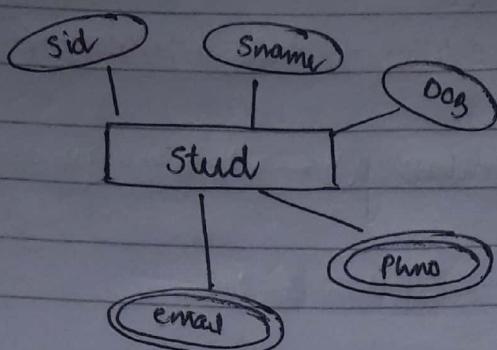
INF
design.

Stud (Sid Phno Cid)

$\text{Sid} \rightarrow \rightarrow \text{Phno}$
 $\text{Sid} \rightarrow \rightarrow \text{Cid}$

cand key : { Sid Phno Cid }

Eg -



{ $sid \rightarrow Sname$

$sid \rightarrow Dob$

$Sname \rightarrow Dob$

$sid \rightarrow Phno$

$sid \rightarrow Email$ }

FDPs and MVD set
of Stud relation.

Properties of MVD's —

① Complementation rule :

If $X \rightarrow Y$ is MVD of R, then

$X \rightarrow R - (X \cup Y)$ is also MVD of rel R.

Eg - R(ABCD)

If $A \rightarrow B$ exist

then $A \rightarrow CD$ also exist.

R(A B C D)

4 5 6 7

$A \rightarrow B$ ✓

4 6 7 7

$A \rightarrow CD$ must also be true.

4 5 6 7

4 6 7 7

② Trivial MVD —

$X \rightarrow Y$ is trivial MVD only if

$X \supseteq Y$ (or) $X \cup Y = R$

Eg - stud (Sid Phno)

free from redundancy.

Note - NO non trivial MVD's possible in relation R (AB) with only 2 attributes

{ Sid $\rightarrow\rightarrow$ Phno }

Trivial MVD

Sid $\rightarrow\rightarrow$ Sid }

Trivial MVD

Phno $\rightarrow\rightarrow$ Phno }

Eg - stud (Sid Pno Cid)

{ S₁ P₁ C₁ } { Sid $\rightarrow\rightarrow$ Cid }

Non Trivial MVD

{ S₁ P₂ C₂ }

{ S₁ P₁ C₂ } { Sid $\rightarrow\rightarrow$ Pno }

Non Trivial MVD

{ S₁ P₂ C₂ }

Eg - R (ABCD)

{ AB $\rightarrow\rightarrow$ CD }

trivial MVD

(3) MVD's not allowed to split over determiner also; also not allowed to merge if determinants are same.

In FD's : If $X \rightarrow YZ$ is implied in R, then $X \rightarrow Y$, $X \rightarrow Z$ also exist in R.

In MVD's : If $X \rightarrow YZ$ implied in R, then

$X \rightarrow Y$, $X \rightarrow Z$, may/may not implied in R

Eg - stud (Sid Cno Rno Phno) { Sid $\rightarrow\rightarrow$ Cno Rno }

S₁ $\rightarrow\rightarrow$ C₁R₁ | C₂R₂ / C₃R₃

{ Sid $\rightarrow\rightarrow$ Cno } { Sid $\rightarrow\rightarrow$ Rno } may not in R'

④ Replication Rule -

Every FD ($X \rightarrow Y$) can be replaced by MVD ($X \rightarrow \rightarrow Y$)
i.e.

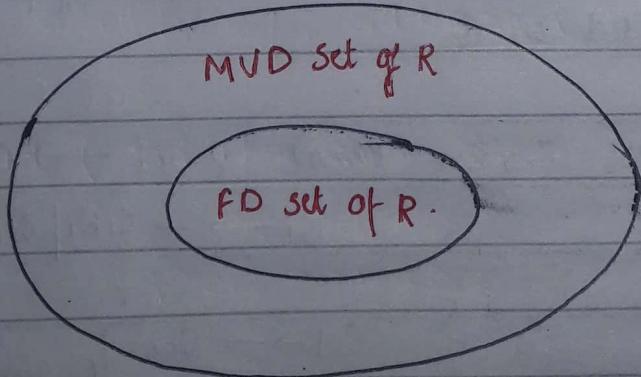
$\text{if } X \rightarrow Y, \text{ then } X \rightarrow \rightarrow Y$

Eq -

$$\begin{array}{ccc} \text{Sid} \rightarrow \text{Sname} & \rightleftharpoons & \text{Sid} \rightarrow \rightarrow \text{Sname} \\ \text{S}_1 \rightarrow [A] & & \text{S}_1 \rightarrow \rightarrow [A] \\ \begin{array}{l} \uparrow \\ \text{Single Sname} \\ \text{for } S_1 \\ (\text{only 1 value}) \end{array} & & \begin{array}{l} \uparrow \\ \text{Set of Sname} \\ \text{for } S_1 \\ (\text{i.e. one or more} \\ \text{value}) \end{array} \end{array}$$

reverse of it is not allowed.

$$\begin{array}{ccc} \text{Sid} \rightarrow \rightarrow \text{Sname} & \not\rightleftharpoons & \text{Sid} \rightarrow \text{Sname} \\ \text{S}_1 \rightarrow [] & & \text{S}_1 \rightarrow [] \\ \begin{array}{l} \text{Set of Sname} \\ \text{for } S_1 \end{array} & & \begin{array}{l} \text{Single Sname for } S_1 \\ (\text{may not be true}) \end{array} \end{array}$$



Ques:

which is correct?

- 1) if $A \rightarrow BC$, then $A \rightarrow B$, $A \rightarrow C$
- 2) if $A \rightarrow BC$, then $A \rightarrow B$, $A \rightarrow C$
- 3) if $A \rightarrow BC$, then $A \rightarrow B$, $A \rightarrow C$
- 4) if $A \rightarrow BC$, then $A \rightarrow B$, $A \rightarrow C$

(a) 1, 2
✓ 1, 3

(b) 2, 4
(d) All true

4NF -

definition: Relational schema R is in 4NF iff

(a) Every non-trivial FD, $X \rightarrow Y$ in R
with X as superkey.

and (b) Every non-trivial MVD $X \rightarrow\rightarrow Y$ in R
with X as superkey.

sufficient for 4NF

as (a) is subset of it

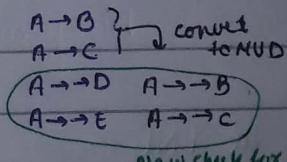
Note-

Attribute closure

Candidate keys identification

Testing of attribute set superkey or not

} should
use only
FD set
of relation



Eg- R (ABCD)

$\{ A \rightarrow B, B \rightarrow C, A \rightarrow\rightarrow D \}$

$A^+ = \{ A, B, C \}$

Not be taken.

• candidate key

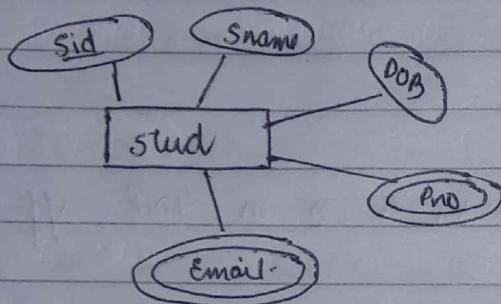
$$(AD)^+ = \{ AD, BC \}$$

$$\therefore \{ AD \}$$

• No. of S.K's in R

$$\Rightarrow \{ AD, ADB, ADC, ABCD \}$$

Ans:



Design in INF

$$Sid \rightarrow Sname \Rightarrow Sid \rightarrow \rightarrow Sname$$

$$Sid \rightarrow DOB \Rightarrow Sid \rightarrow \rightarrow DOB$$

$$Sid \rightarrow \rightarrow Phno$$

$$Sid \rightarrow \rightarrow Email$$

candidate key : Sid Phno Email

In INF -

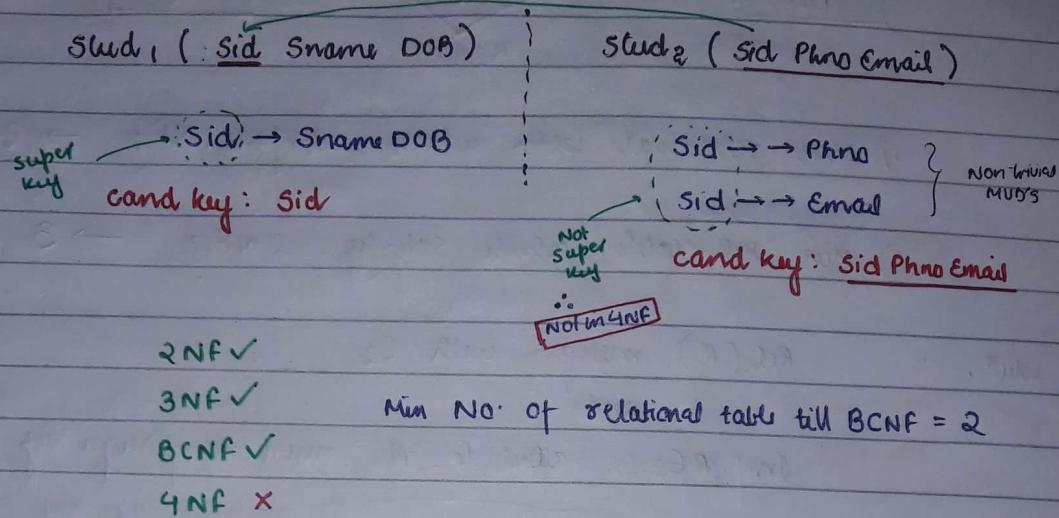
stud (Sid Phno Email Sname DOB)

? Sid \rightarrow Sname DOB , Sid $\rightarrow \rightarrow$ Phno , Sid $\rightarrow \rightarrow$ Email ?

Not allowed
in 2NF

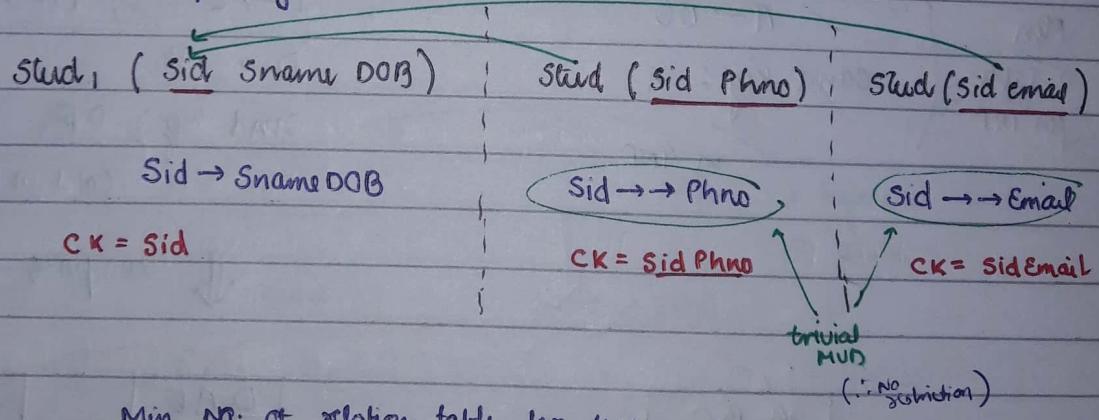
3NF -

we need to decompose it into 2 tables



Min No. of relational tables till BCNF = 2

4NF - decomposing it



Min No. of relation table for 4NF = 3

Note -

If BCNF rel R
(and)

at least 2 MVA stored in R
by converting into SVA,
then

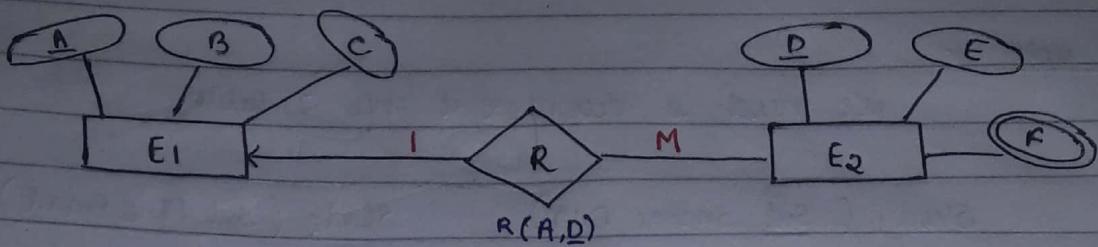
\downarrow
R not in 4NF

If R is in BCNF
(and)

Atmost one MVA
stored in R by converting
into SVA, then

\downarrow
R also in 4NF

Ques:



1) Min relational tables which satisfy 1NF? $\rightarrow 2$

2) Min rel. tabs which satisfy 2NF? $\rightarrow 3$

Solutn:

Rel (R) merges with E_2

and

In RE_2 , attribute 'A' must be foreign key ref to E_1 .

$E_1 (A B C)$

Foreign key
 $E_2 R (DEF(A))$ reference allowed.

Min
2 Table
for 1NF

$A \rightarrow BC$

cand key: A

✓ in 4NF

$D \rightarrow E$

$D \rightarrow A$

partial
dependency

cand key: DF

(NOT in 2NF)

↓ decompose

$A B C$

$D A E$

$D F$

$A \rightarrow BC$

$D \rightarrow E$

$D \rightarrow F$

$D \rightarrow A$

∴ Min rel table for 2NF

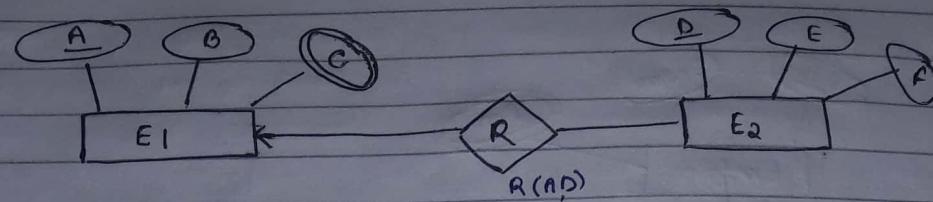
3 NF

BCNF

4NF

→ 3

Ques:



Min rel req for INF?

solutn:

Rel (R) merge with E₂ and
'A' of rel RE₂ must be foreign key ref to E₁

} design constraint

E₁ [A B C]

A → B

A → C

cand. key = AC

RE₂ [DEF A]

D → E F A

cand. key = D

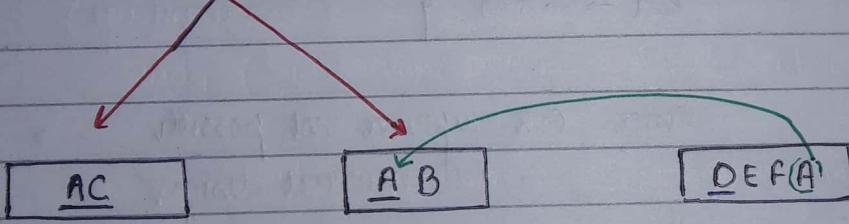
foreign key

reference

Not Possible

as A is not
key
(constraints are
lost)

∴ we cannot do it
by 2 tables



A → C

A → B

D → E F A

F-K reference
is allowed ✓

∴ Min 3 tables for INF

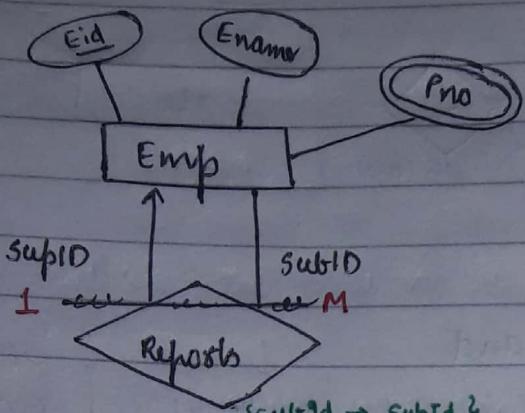
H/W → ① inv above take 1:1 ?

② .. " " 1:1 with multivalue both sides ?

Note— When converting ER diagram to RDBMS'

{ INF if multivalued attribute exist }
4NF if MV attribute does not exist }

Ques:



Main rel table

Required?

1:M

⇒ M side combines
(SubID combines with Eid)

as SubID is superkey
it defines attr.

Sol'n:

Emp Reports (Eid, Ename, Pno, SupID)

F.K reference not possible

(as Eid not a key)

{ eid → Ename
eid → SupID
eid → Pno }

Cand key: { eid Pno }

Since F.K reference not possible
∴ incorrect design

Emp, (Eid, Pno)

eid → Pno

Cand key: eid Pno

Emp₂ Reports (Eid, Ename, SupID)

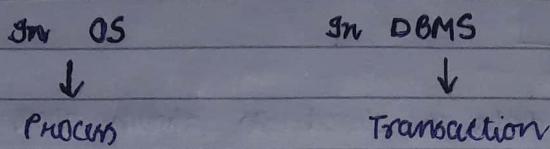
eid → ename SupID

Cand key: eid

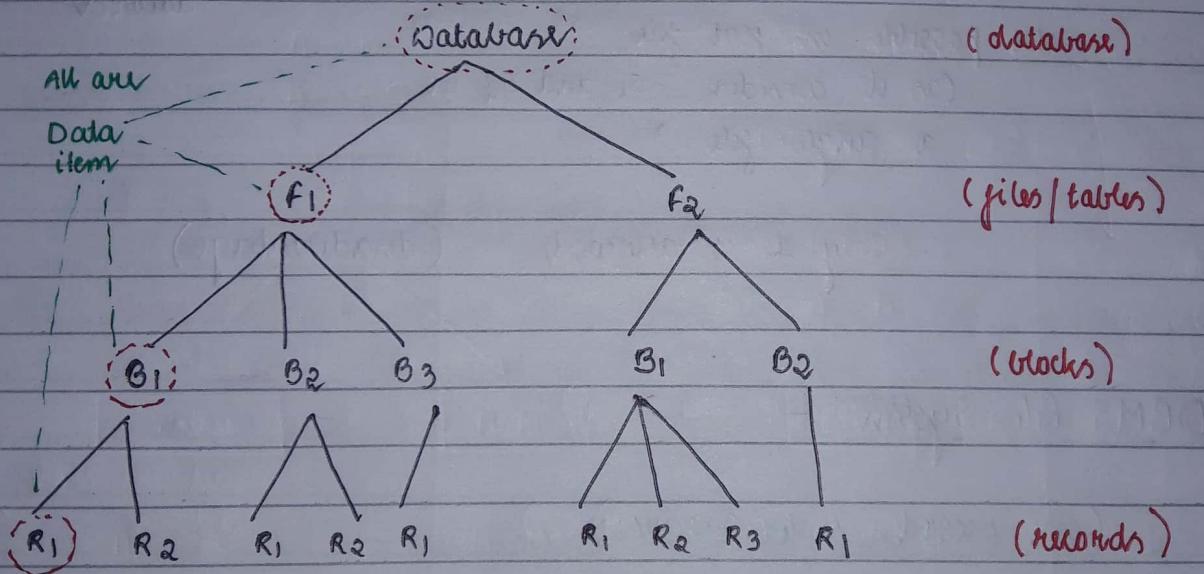
∴ Min 2 relational table required!

Transactions and Concurrency Control

- Transaction is set of logically related operations to perform unit of work.



- Data item (shared resource) — DB element which is required to access by many transaction.



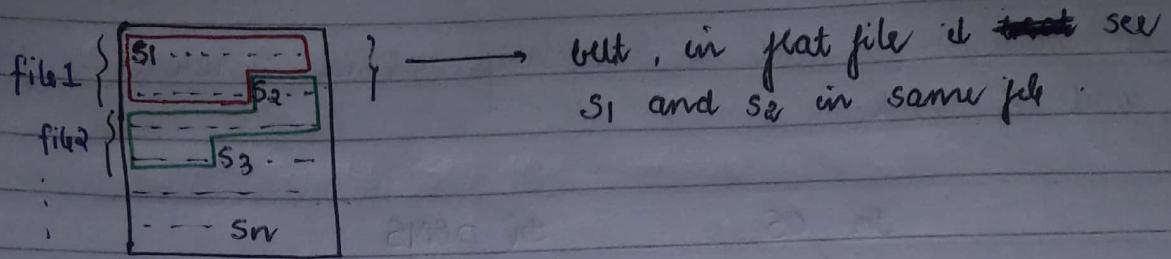
Degree of concurrency — No. of users who can use DB simultaneously (concurrently)

Flat file system

(OS file system)

DBMS file system

(1 file is a resource)



In flat file, the data is stored as shown above.

↓
 suppose more than 2 update but
 in different place

↓ U1: update S1 } concurrent execution
 U2: update S2 of U1, U2 req. not
 allowed

NOT possible in flat file
 (as it consider S1 and S2 in
 a single file)

∴ Only 1 is allowed (disadvantage)

DBMS File System —

(one record is a resource)

R1	S1			↳ resource
R2	S2			↳ Res
R3	S3			↳ Res
R4	S4			↳ Res

Since, DBMS point of view, record is a resource

↓
so it allows as many user, as the no. of records



More degree of concurrency

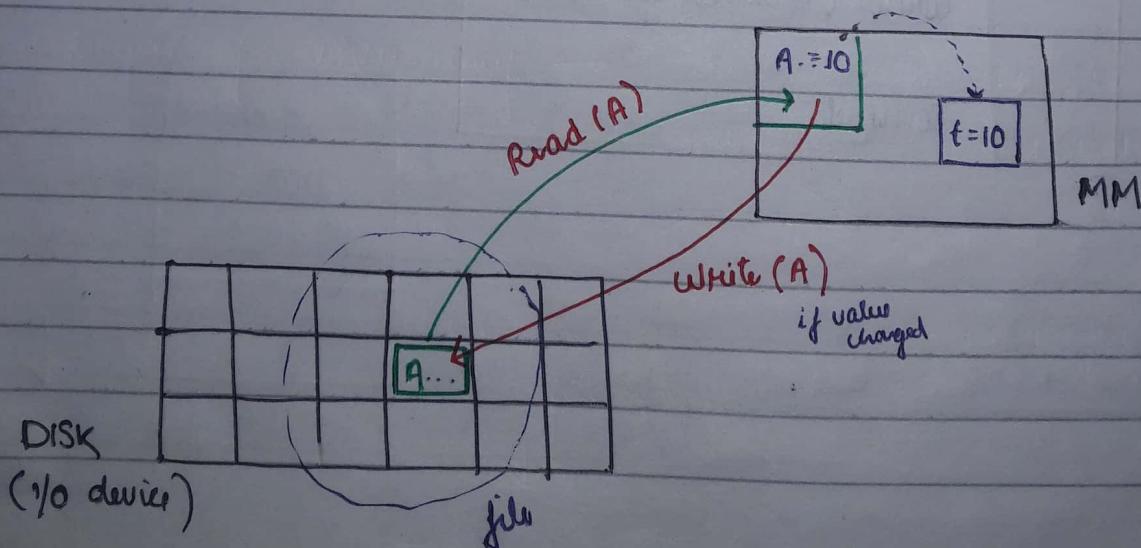
using DBMS file system,
because concurrency control over record level.

* We cannot do the same thing in OS, as the file structure is not under user (fixed)

Main Operations in Transaction -

A : data item

- ① **Read (A) —** Access data item 'A' from DB file (DISK) to programmed variable (MM) in order to use current value of 'A' in transaction logic.
- ② **Write (A) —** Modification of data item 'A' in DB file.



→ Account (Cid bal)

Trans (T_1): Transfer 500 from customer C_1 to C_2

SQL queries → begin trans T_1

update account set bal = bal - 500
where Cid = C_1 ;

update account set bal = bal + 500
where Cid = C_2 ;

end trans

↓ SQL parser executes

Trans (T_1)

A

Read (Balance of C_1)

$$A = A - 500$$

A

Write (Balance of C_1)

Read (Balance of C_2)

B

$$B = B + 500$$

Write (Balance of C_2);

B

commit;

← Transaction

(various
read and write
together)

Trans (T_2): set bal of C_1, C_2 as 10,000

update account set bal = 10000 where Cid = C_1 ;

update account set bal = 10.000 where Cid = C_2 ;



Trans (T)

write $\text{bal} = 10000 \text{ of } C_1$ A

write $\text{bal} = 10000 \text{ of } C_2$ B

commit

Blind write

(writing back
in DB without
any knowledge
of prev value)

Trans (T_3):

R (A)

R (B)

w (B)

w (C)

w (D)

Commit

Blind write

ACID Properties —

To preserve integrity (correctness) each transaction must satisfy ACID property.

A :	Atomicity	Recovery Management component of DBMS s/w takes care of A and D
D :	Durability	
I :	Isolation	{
C :	consistency	

DBMS S/W } { A : Atomicity
 DBMS S/W } { D : Durability
 DBMS S/W } { I : Isolation

Recovery management component of DBMS s/w takes care of atomicity and durability.

DBA user } { C : Consistency

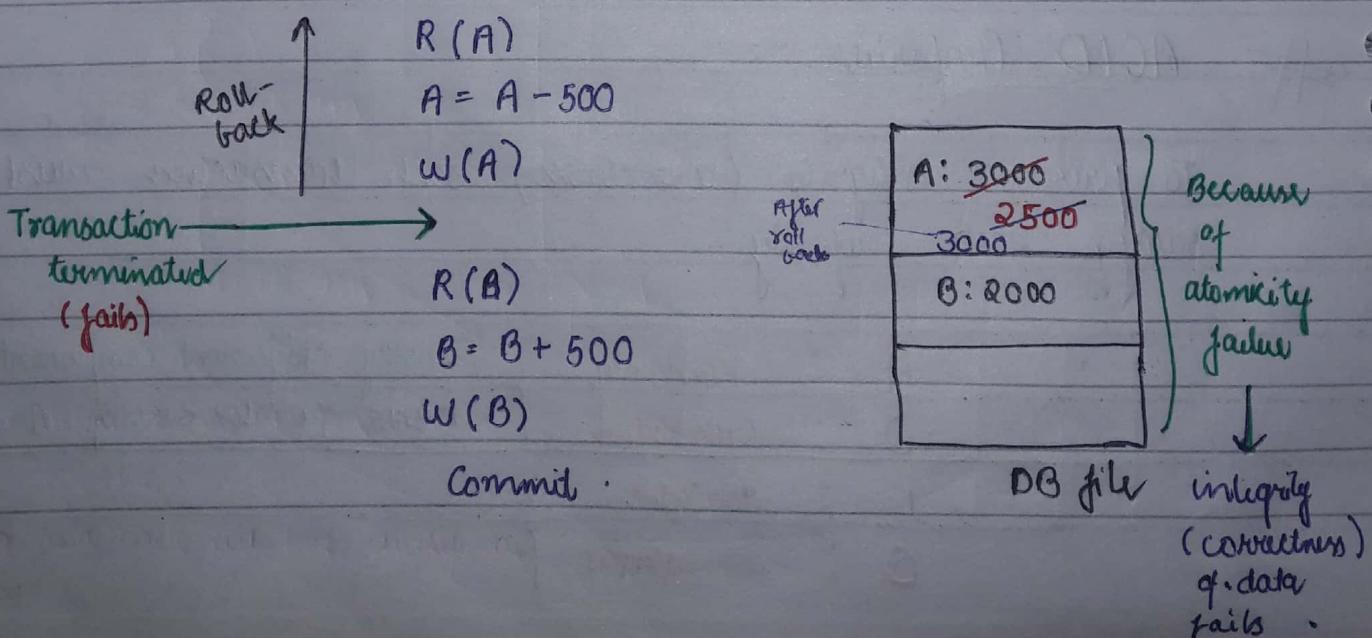
Concurrency control component of DBMS s/w is responsible for isolation
 USER [DBA or DB developer] is responsible for consistency

Atomicity -

(Execute all operations of transaction including commit)
 (OR)

(Execute none of the operation of transaction)
 by the time of transaction termination.

Eg- Trans (T_1) : transfer 500 from A to B



↓
Recovery manager
Roll backs all the
transaction done by T_i

↓
If Rollback (Undo opn) is successful

↓
'Atomicity Preserved'

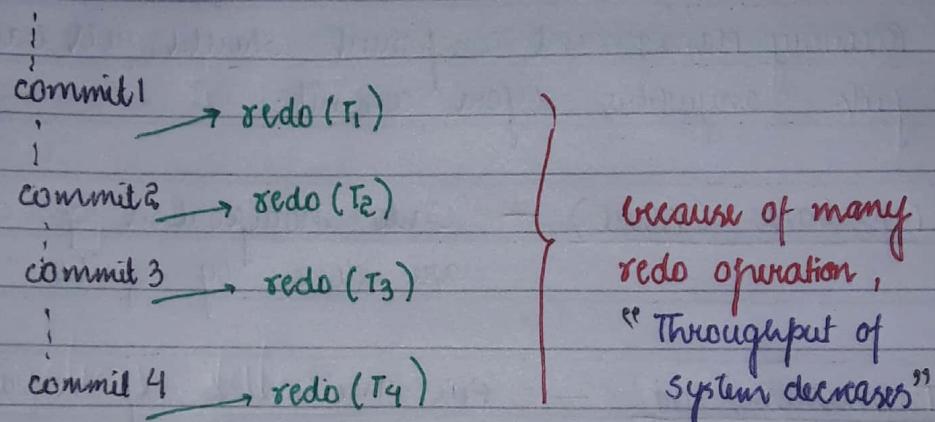
- Recovery Management component should roll back, if transaction fails anywhere before commit.
- Roll back (Abort) — Undo modification of database file which are done by failure transaction.
- Transaction log — File maintained by recovery management component (RMC) in DISK (secondary memory) to record all the activities of transaction.

Transaction (T_i)	log file	DB file	if flag set then dirty block.
$R(A)$	$T_i : \text{begins}$	$B_1 : A: 3000\ 2500$	$\boxed{1}$ Dirty Block
$A = A - 500$	$T_i : \text{Read } A, 3000$		
$W(A)$	$T_i : \text{Write } A, 2500, 2500$	$B_2 : B: 2000\ 2500$	$\boxed{1}$ After change Dirty Block
$R(B)$	$T_i : \text{Read } B, 2000$		$\boxed{2}$ Clean Block
$B = B + 500$	$T_i : \text{Write } B, 2500, 2500$		$\boxed{2}$ Dirty Block
$W(B)$			
Commit			

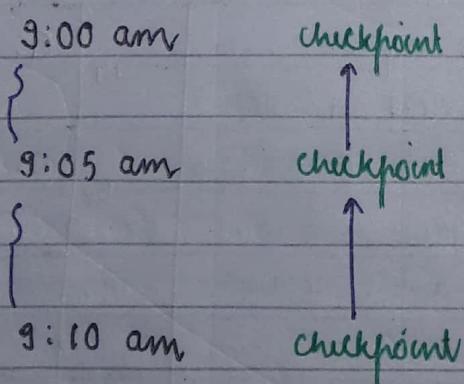
Dirty Block — updated by uncommitted transaction.

- Redo operations —

- Re-performs all the modification of database file because of **transaction commit** (i.e after commit)
- Clean all log entries of committed transaction.
- Redo is not done after every commit, but after checkpoint



↓ To avoid this problem
 'consistency check point' is introduced



"All commit transaction until previous checkpoint performs redo"

FAILURE → 9:14 am

(Need to Recover)

→ Roll back till last checkpoint

~~→~~

- If checkpoint issued, all committed transactions until previous checkpoint should perform redo and clean log entries of committed transaction.
- If system failure happens, required openⁿ to recover are
 - (1) All committed transaction until previous checkpoint will perform Redo and,
 - (2) All uncommitted transaction in entire system will perform Undo.
 - (3) clean log entries.

Ques: Consider the log file entries

	1. T ₁ begin	T ₁
	2. T ₁ w, A, 10, 20	T ₂
	3. T ₂ begin	T ₃
	4. T ₂ w, B, 0, 5	T ₄
	5. T ₂ commit	—→ Committed
	6. T ₃ begin	T ₅
	7. T ₃ w, B, 5, 10	commit
in entire sys whenever no commits go for undo	8. checkpoint	redo of T ₂ done
T ₁ , T ₃ , T ₅	9. T ₄ begin	until prev check point
	10. T ₅ begin	who ever committed will redo
	11. T ₄ w, C, 0, 100	(T ₄)
	12. T ₄ Commit	
	13. failure	

What are required openⁿ performed to recover from failure

a) T₂ T₄ redo and T₁ T₃ T₅ Undo

(b) ✓ T₄ redo, T₅ T₁ T₃ undo

c) T₁ T₂ T₃ T₄ T₅ Undo

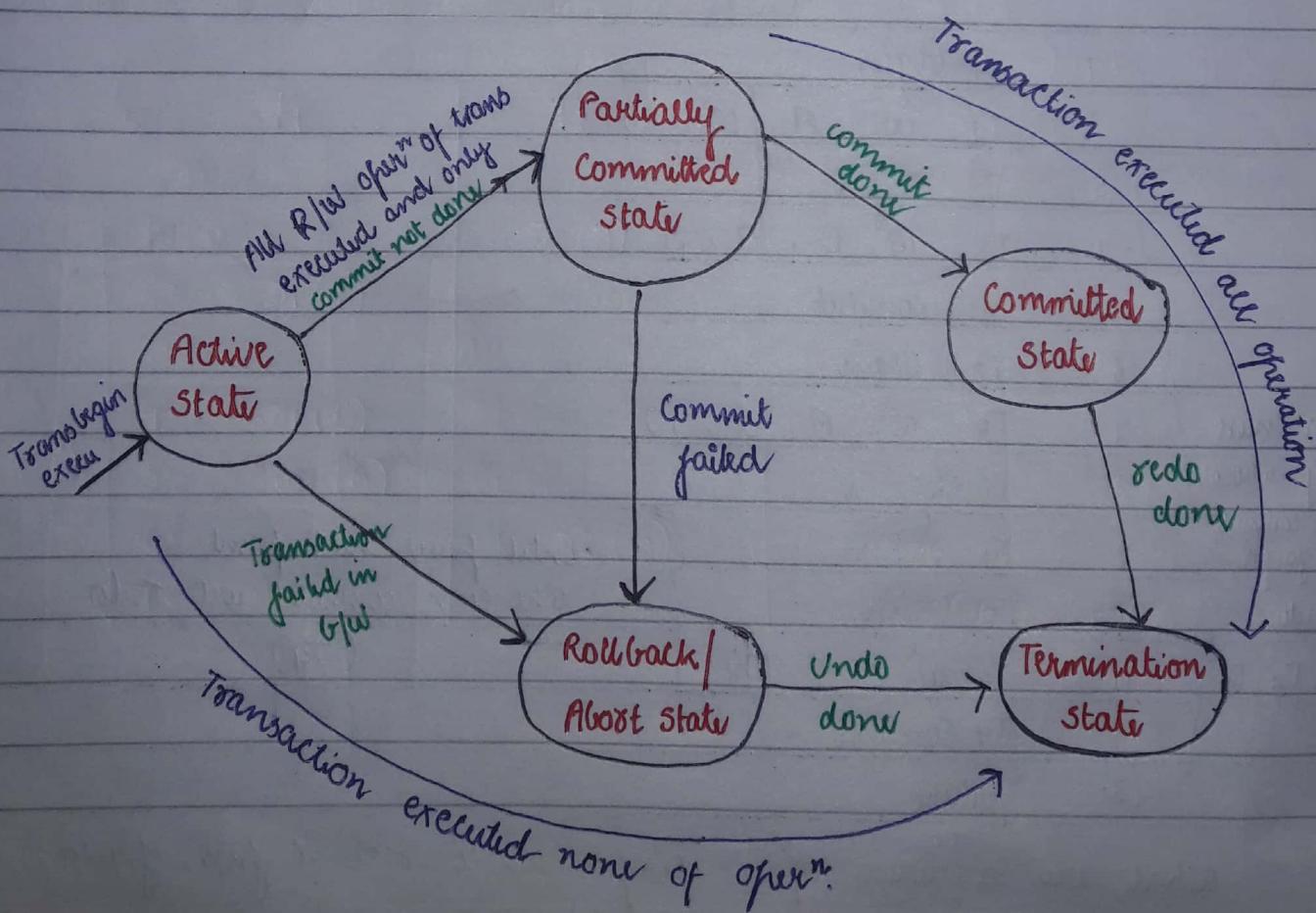
d) T₂ redo and T₁, T₃ T₄ T₅ undo

redo : T₄

undo : T₁, T₃, T₅

Note - Before checkpoint T₂ completed redo i.e commit.

Transaction States -



Durability -

- Transaction should able to recover under any case of failure.
 - Transaction fails because of :
 - ① Power failure
 - ② S/W Crash
 - OS restarted
 - DBMS restarted
 - ③ OS / DBMS concurrency control may kill transaction.
 - ④ H/W crash (DISK crash)
- should
be able to
roll back
'durable'
- if log file and state of
each trans maintained
in DISK
- RAID architecture
of disk design is used



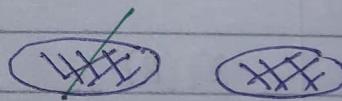
RAID : 0



data in
main disk

Not durable

RAID : 1



Mirror image of Disk

2 disk

Isolation —

(Maintained by Concurrency Control Component)

Isolation means concurrent execution of 2 or more transactions result must be equal to result of some serial schedule.

Schedule : Time order execution sequence of 2 or more transaction.

Eg -

$$\begin{array}{lll} T_1 : & \sigma_1(A) & \sigma_1(B) \\ T_2 : & \sigma_2(A) & \sigma_2(B) \\ T_3 : & w_3(A) & w_3(B) \end{array}$$

↓ concurrent Representation
(Schedule)

Time order diagram

S	T ₁	T ₂	T ₃
1.		$\sigma_2(A)$	
2.	$\sigma_1(A)$		
3.			$w_3(A)$
4.	$\sigma_1(B)$		
5.	$w_1(B)$		
6.		$\sigma_2(B)$	
7.			$w_3(B)$

Types of schedule -

- ① serial schedule
- ②

→ Serial Schedule :

- Transaction should execute one after other.
- After the commit of current executing transaction, allow the beginning (start) of other transaction.

Eg -

T_1 : Transfer 500 from A to B

T_2 : Display total bal of A and B

S_1 :	T_1	T_2
	$\sigma_1(A)$	
	$w_1(A)$	
	$\sigma_1(B)$	
	$w_1(B)$	
	commit	
		$\sigma_2(A)$
		$\sigma_2(B)$
		disp (A+B)
		commit

← only after commit, it can start.

OR

S_2 :	T_1	T_2
		$\sigma_2(A)$
		$\sigma_2(B)$
		dis (A+B)
		commit
	$\sigma_1(A)$	
	$w_1(A)$	
	$\sigma_1(B)$	
	$w_1(B)$	
	commit	

Advantage -

- Serial schedules guarantee produces correct result (integrity guaranteed) as no resource sharing.

Disadvantage -

- It allows transaction to execute serially only
- less degree of concurrency
- Throughput of system is less

→ Concurrent Schedule

Transactions can execute concurrently / simultaneously / interleaved order.

Eg -

S ₃ :	T ₁	T ₂	S ₄ :	T ₁	T ₂
3000	$\gamma_1(A)$		3000	$\gamma_1(A)$	
2500	$w_1(A)$		2500	$w_1(A)$	
		$\gamma_2(A)$ 2500			$\gamma_2(A)$ 2500
		$\gamma_2(B)$ 2000	2000	$\gamma_1(B)$	
		disp(A+B) 4500	2500	$w_1(B)$	
	commit	incorrect	commit		
$\gamma_1(B)$			$\gamma_2(B)$ 2500		
$w_1(B)$			disp(A+B) 5000		
commit			correct		

- Concurrent schedule are both serial / non serial schedule

On executing S₃ —

$$A = 3000$$

$$B = 2000$$

→ Schedule not equal to $T_1 \rightarrow T_2$
serial and $\neq T_2 \rightarrow T_1$ serial
(Isolation fails)

$$\text{disp}(A+B) = 4500$$

(incorrect result)

On executing S₄ —

$$A = 3000$$

$$B = 2000$$

→ Schedule equal to $T_1 \rightarrow T_2$
serial. (S₄)
(Isolation satisfied)

$$\text{disp}(A+B) = 5000$$

(correct result)

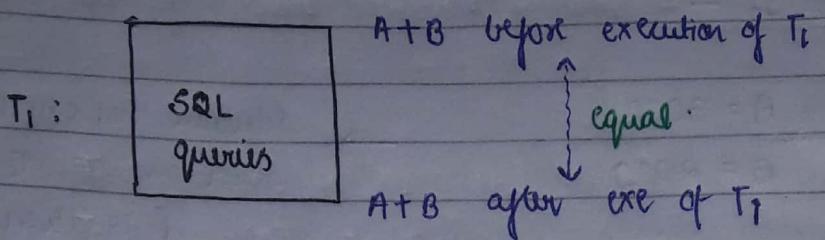
∴ Result of concurrent schedule may be correct or incorrect.

14/11/17
Consistency —

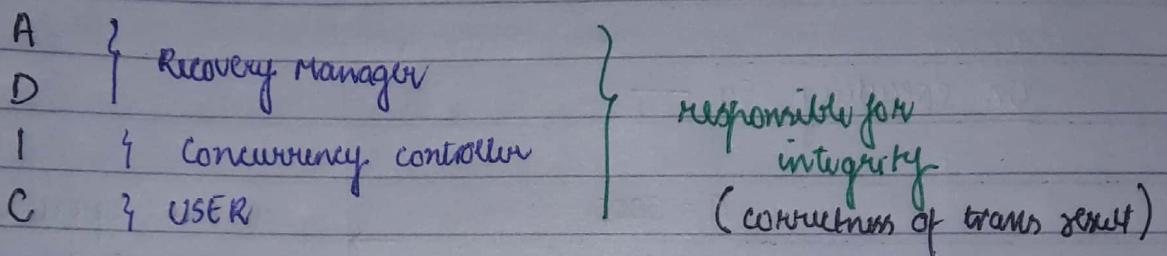
(User responsible for consistency)

- DB operations requested by user [SQL queries / transaction open] must be logically correct
- Consistency depends on transaction to transaction

T_1 : Transfer 500 from A to B



①



Schedule Classification

Serializability
(Isolation must satisfy)

Recoverable schedules
(atomicity and durability must satisfy)

- ④ Conflict serializable schedule
⑤ View serializable schedule

serializability Testing
(isolation testing)

Other Problems of concurrent execution —

even if isolation is preserved, these problems can occur:

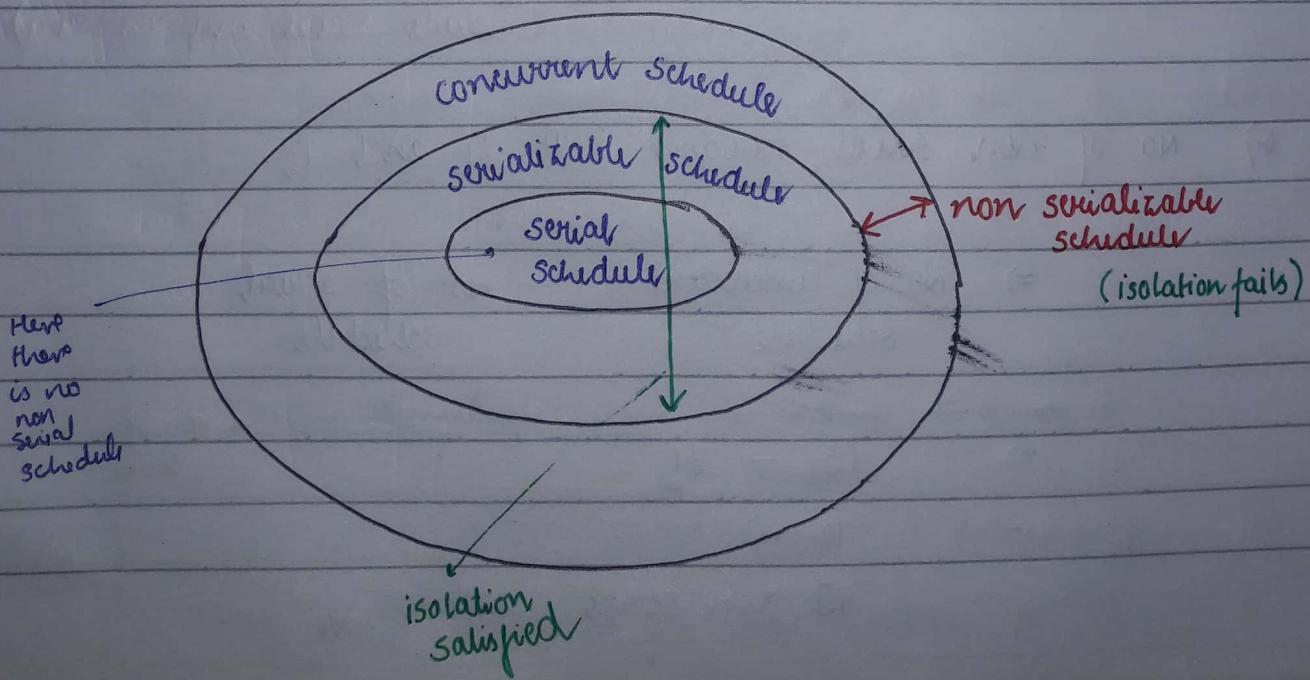
- ① Recoverable schedule problem
- ② Cascading Rollback problem
- ③ Lost update problem.

↓ To avoid above problem soln are —

- ① Recoverable scheduler (for recoverable sch prob)
- ② Cascading Rollback scheduler
(for cascading roll back)
- ③ Strict Recoverable scheduler
(for lost update prob)

Serializable Schedule —

Schedules(s) are serializable iff some serial schedules(s) equal to the schedule(s) i.e. preserve isolation.



Ques: $T_1 T_2 T_3 \dots T_n$ transactions.

No. of serial schedule possible = $n!$

Ques: $T_1 : \sigma_1(A) w_1(A) \sigma_1(B) w_1(B)$ → All should execute in same order
 $T_2 : \sigma_2(A) \sigma_2(B)$ → "

a) No. of concurrent schedule b/w T_1 and T_2

Soluⁿ:

	T_1	T_2
1.		
2.		
3.		
4.		
5.		
6.		

$$= T_1 \downarrow \quad T_2 \downarrow$$

$$= {}^6C_4 \cdot {}^2C_2$$

$$= 15 \times 1$$

= 15 concurrent schedule Ans

(include serial and non serial)

b) NO. of non serial schedule b/w T_1 and T_2

$$\Rightarrow \text{NO. of concurrent schedule} - \text{NO. of serial schedule}$$

$$\Rightarrow 15 - 2! \xrightarrow{\text{No. of trans.}}$$

$$= 15 - 2$$

$$= 13 \text{ non serial schedule Ans}$$

Ques: T_1, T_2 transaction with n, m operation each.
No. of concurrent schedule.

$$\text{For } T_1 \Rightarrow n+m C_n \\ T_2$$

$$\Rightarrow \frac{(n+m)!}{n! \cdot m!}$$

Ans

Ques: T_1, T_2, T_3 transaction with n, m, p operation each.

No. of concurrent schedule?

$$= (n+m+p) C_n \cdot (m+p) C_m \cdot P C_p$$

$$= (n+m+p) C_n \cdot (m+p) C_m$$

$$= \frac{(n+m+p)!}{n! \cdot m! \cdot p!} \quad \underline{\text{Ans}}$$

1) Conflict serializable Schedule —

→ Topological Order: (Greedy Algo)

Graph traversal algorithm applicable only for directed acyclic graph (DAG).

Procedure —

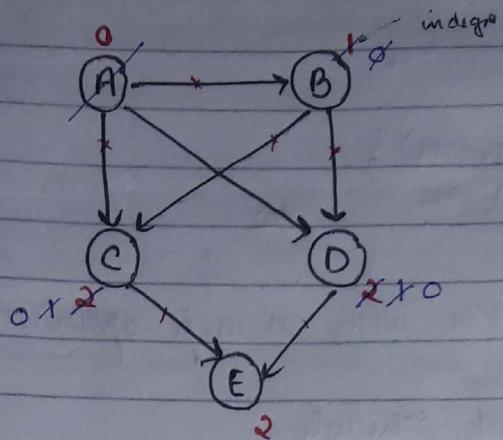
① Visit vertex (v) whose in-degree is '0'
and

delete visited vertex (v) from graph

② Repeat ① for all vertices of graph

Topological Order : Visited order of vertices

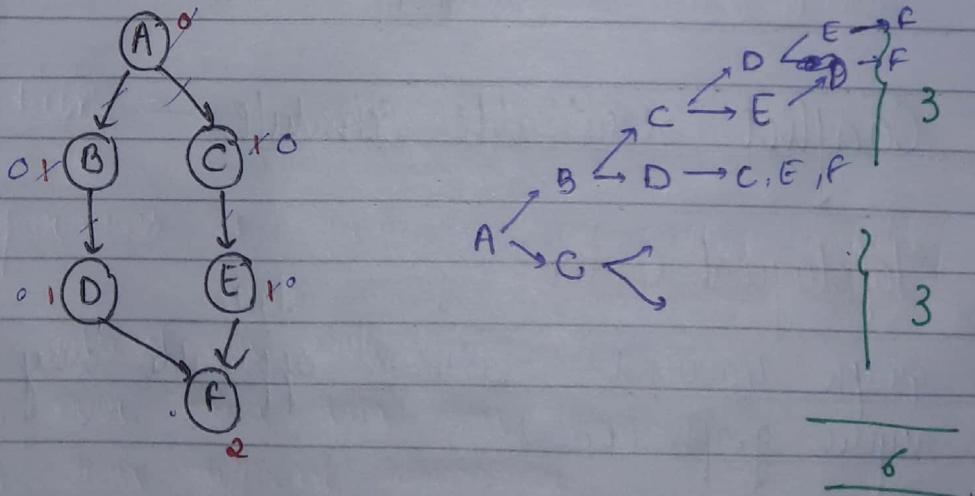
Eg -



visit indegree 0 and delete its outgoing edges; repeat

$A, B \rightarrow$ C, D, E
 $A, B \rightarrow$ D, C, E
 (2 topological order)

Ques: How many no. of topological order possible



A — B — C — D — E — F

$B \rightarrow$ before D
 $C \rightarrow$ before E

$4C_2$ ways

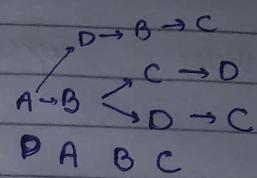
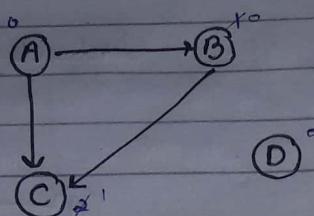
2C_2

$\Rightarrow 4C_2 \cdot {}^2C_2$

$\Rightarrow 6$

Ans

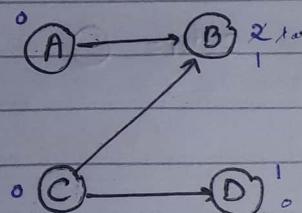
Ques:



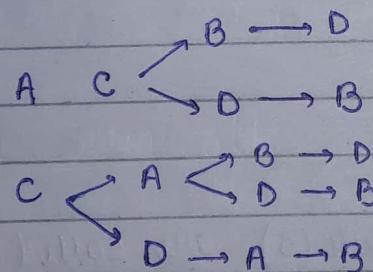
Find no. of topological order.

(4)

Ques:



No. of topological order



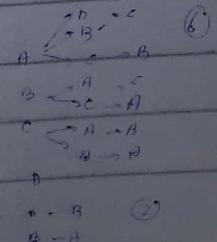
(5)

Ques: No. of Topological order of null graph with n vertices.
(0 edges)

$2 \rightarrow 2$
 $3 \rightarrow 6$

$n!$

$\bullet \bullet \bullet \bullet$

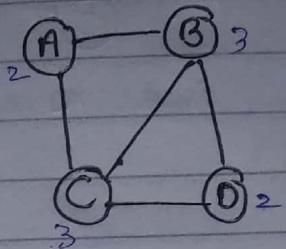
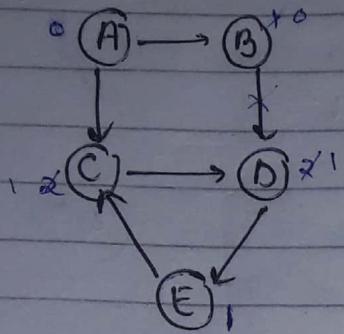


(7)

Ques:

cyclic graph

Undirected graph.



No. of Topolo order?

order = 0

A B order = 0

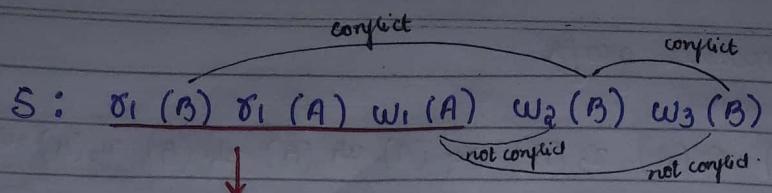
Not Possible.
for cyclic graph

Conflict Pair -

Pair of operations in schedule(s) are called conflict pair iff:

- ① At least one write operation and
- ② over same data item and
- ③ from different transaction

S: $\tau_i(A) \dots w_j(A)$	}	conflict Pair
S: $w_i(A) \dots \tau_j(A)$		
S: $w_i(A) \dots w_j(A)$		
S: $\tau_i(A) \dots \tau_j(A)$	}	Not conflict Pair
S: $\tau_i(A)/w_i(A) \dots \tau_j(B)/w_j(B)$		



opwr^m within transaction
 should execute in same order
 $\therefore \tau_1(B) \ \tau_1(A) \ w_1(A)$

"Neither conflict nor not conflict Pairs"

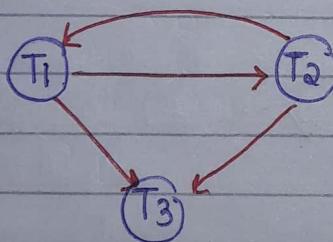
Precedence Graph -

Vertices (V) : Transaction of schedule.

Edges (E) : conflict Pair Precedences

Eg -

$S: \tau_1(A) \ \tau_2(B) \ w_2(A) \ w_1(A) \ w_2(B) \ w_3(B) \ w_3(A)$



1) Highest No. is the no. of transact.

2) Take one and see the conflict
 $\tau_1(A)$

saw A write will be seen if no. is change conflict

Ques: Draw precedence graph

S: $\tau_1(A) \quad \tau_2(A) \quad \tau_3(A) \quad \tau_4(A) \quad w_1(B) \quad w_2(B) \quad w_3(B) \quad w_4(B)$

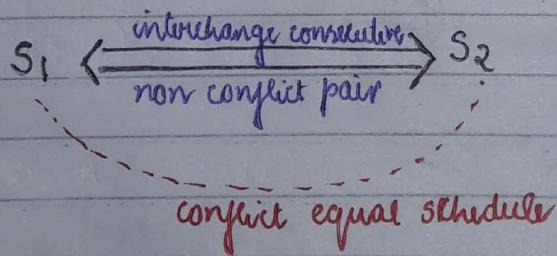
NO conflict for read-read.

```

graph TD
    T1((T1)) --> T2((T2))
    T1 --> T3((T3))
    T1 --> T4((T4))
    T2 --> T4
    T3 --> T4
  
```

Conflict Equal schedules —

s_1, s_2 are conflict equal schedules iff s_2 is derived by interchange of consecutive non conflict of schedule s_1



Eg -

$s_1:$	T_1	T_2	$s_2:$	T_1	T_2	$s_3:$	T_1	T_2
1. $\tau_1(A)$			2. $\tau_1(A)$			3. $\tau_1(A)$		
2. $w_1(A)$			3. $w_1(A)$			4. $w_1(A)$		
3. $\tau_2(A)$			4. $\tau_2(A)$			5. $\tau_2(A)$		
4. $\tau_1(B)$			5. $\tau_1(B)$			6. $\tau_1(B)$		
5. $w_1(B)$			6. $w_1(B)$			7. $w_1(B)$		
6. $\tau_2(B)$			7. $\tau_2(B)$			8. $\tau_2(B)$		

Annotations in the table:

- $w_1(A)$ and $\tau_2(A)$ are swapped in s_2 compared to s_1 , indicated by a red arrow labeled "conflict".
- $\tau_1(B)$ and $w_1(B)$ are swapped in s_3 compared to s_1 , indicated by a red arrow labeled "conflict same as prev".
- $\tau_1(B)$ and $\tau_2(B)$ are swapped in s_3 compared to s_2 , indicated by a red arrow labeled "non conflict".
- $w_1(B)$ is crossed out in s_3 and replaced by $\tau_2(B)$, indicated by a red arrow labeled "conflict not changed".

Eg -

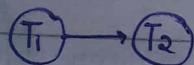
T_1	T_2
$\delta_1(A)$	
$\delta_1(B)$	$w_2(B)$

T_1	T_2
$\delta_1(B)$	
$\delta_1(A)$	$w_2(A)$

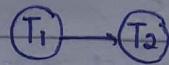
Not conflict equal

Note - The precedence graph of all the conflict equal schedules are same. but not vice versa.

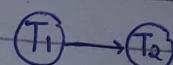
$S_1:$



$S_2:$



$S_3:$



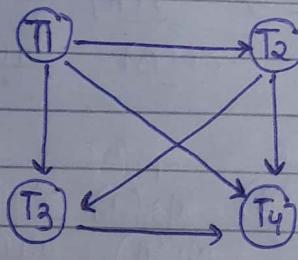
Ans: $S_1: \delta_1(A) \quad \delta_2(A) \quad \delta_3(A) \quad \delta_4(A) \quad w_1(B) \quad w_2(B) \quad w_3(B) \quad w_4(B)$

$S_2: \delta_2(A) \quad \delta_1(A) \quad w_1(B) \quad w_2(B) \quad \delta_3(A) \quad w_3(B) \quad \delta_4(A) \quad w_4(B)$

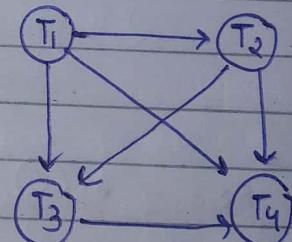
$S_3: \delta_3(A) \quad \delta_4(A) \quad \delta_1(A) \quad w_1(B) \quad \delta_2(A) \quad w_2(B) \quad \delta_4(A) \quad w_4(B) \quad w_3(B)$

what are conflict equal schedule

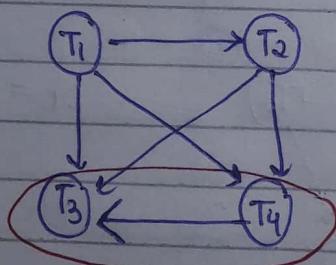
$S_1:$



$S_2:$



$S_3:$



S_1 and S_2 are conflict equal schedule

$$S_1 = S_2 \neq S_3$$

for same precedence graph,
does not mean
conflict equal

s_1 , s_2 are conflict equal iff

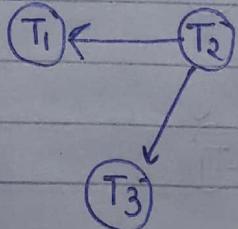
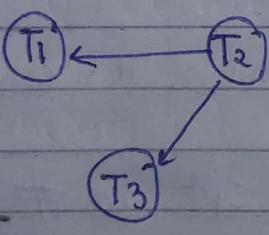
- ① Each transaction of s_1 must be exactly same in s_2
(i.e no. of operations same and sequence same)

$s_1 \dots T_i \dots$	$s_2 \dots T_i \dots$
$\delta_i(A)$	$\delta_i(A)$
$\delta_i(B)$	$\delta_i(B)$
$w_i(B)$	$w_i(B)$

- and ② Every conflict pair precedence of s_1 , s_2 must be same.

Ans ⑦

$s_1:$	$\delta_2(A)$	<u>$w_2(A)$</u>	$\delta_3(C)$	<u>$w_2(B)$</u>	<u>$w_3(A)$</u>	<u>$w_3(C)$</u>	<u>$\delta_1(A)$</u>
					$\delta_1(B)$	<u>$w_1(A)$</u>	<u>$w_1(B)$</u>
$s_2:$	$\delta_3(C)$	<u>$\delta_2(A)$</u>	<u>$w_2(A)$</u>	<u>$w_2(B)$</u>	<u>$w_3(A)$</u>	<u>$\delta_1(A)$</u>	<u>$\delta_1(B)$</u>
					<u>$w_1(A)$</u>	<u>$w_1(B)$</u>	<u>$w_3(C)$</u>
$s_3:$	$\delta_2(A)$	$\delta_3(C)$	<u>$w_3(A)$</u>	<u>$w_2(A)$</u>	<u>$w_2(B)$</u>	<u>$w_3(C)$</u>	<u>$\delta_1(A)$</u>
					$\delta_1(B)$	<u>$w_1(A)$</u>	<u>$w_1(B)$</u>



$$s_1 = s_2 \neq s_3$$

(d)

S

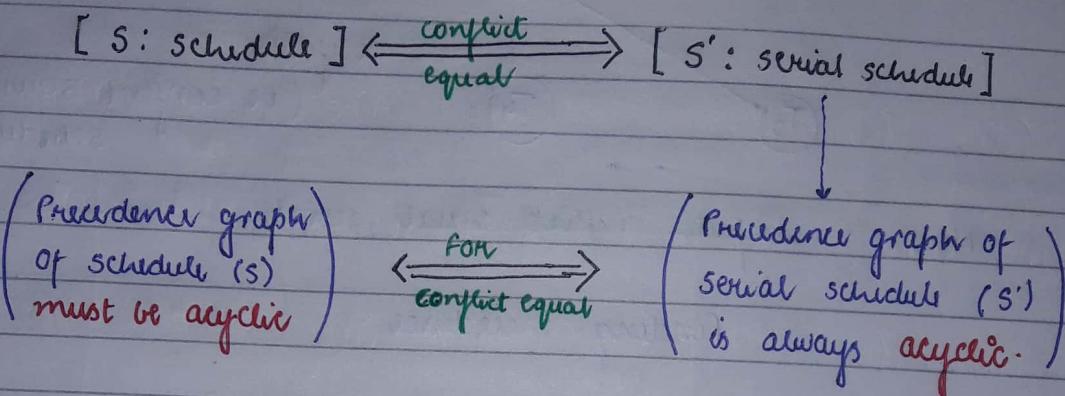
If precedence
same then

s_1, s_2 may
or may not conflict equal

as it does not
show all
conflict pair

Conflict Serializable schedule -

Schedule (s) is conflict serializable schedule iff
some serial schedule (s') conflict equal to schedule s

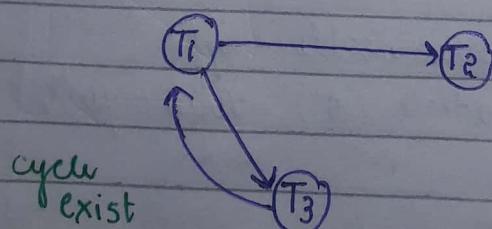


Testing Condition of Conflict Serializable schedule -

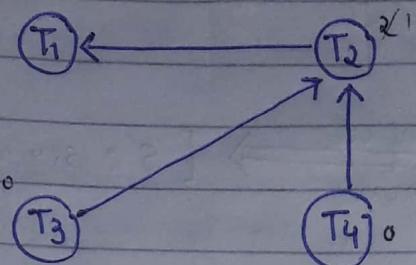
Schedule (s) is conflict serializable schedule iff
precedence graph of schedule (s) is acyclic
and

Conflict equal serial schedule are topological order
of precedence graph.

Eg - $s: r_1(A) w_1(B) r_2(B) r_3(C) w_3(C) w_3(A) w_1(C)$



$\therefore s$ is not conflict serializable schedule
(No serial schedule conflict equal to s)



acyclic
(i.e conflict serializable schedule)

Now, To compute conflict equal serial

Finding Precedence -

2 serial schedule exist { $\begin{matrix} T_4 & T_3 & T_2 & T_1 \\ T_3 & T_4 & T_2 & T_1 \end{matrix}$ order = 2
which is conflict equal to given S

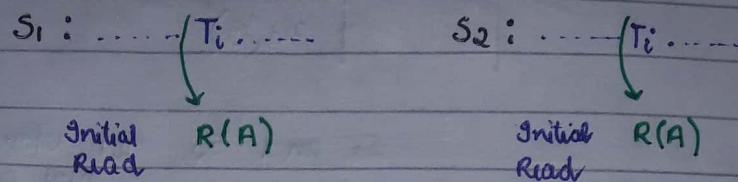
$$\text{No. of serial schedule conflict equal to sch (S)} = \text{No. of topological order of schedule (S) precedence graph}$$

② View Serializable Schedule -

Schedule (S) is view serializable iff
some serial schedule (S') view equal
to schedule (S)

s_1 , s_2 are view equal iff -

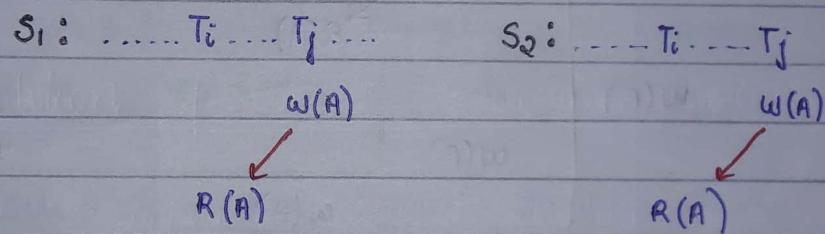
① Initial Read:



Every initial read of s_1 must also be initial read in s_2

Initial read means before read of A , $R(A)$
there is no $w(A)$

and ② Updated Read:



Every updated read of s_1 must be same updated read in s_2 .

and ③ Final write:

$s_1 : \dots T_i \dots$

$s_2 : \dots T_i \dots$

Final write $\rightarrow w(A)$
of A
(last write)

Final write $\rightarrow w(A)$
of A

Every final writes of s_1 must be same in s_2 .

IR - initial read

UR - update read

Eg -

S ₁ :			S ₂ :		
T ₁	T ₂	T ₃	T ₁	T ₂	T ₃
v ₁ (A)			v ₁ (A)		
IR → v ₂ (A)			IR → v ₂ (A)		
IR → v ₂ (B)		w ₃ (B)	X v ₂ (B)	w ₃ (B)	written then read

S₁ and S₂ not view equal
Not conflict equal

Eg -

S ₁ :			S ₂ :		
T ₁	T ₂	T ₃	T ₁	T ₂	T ₃
w(A)			w(A)		
	w(A)			UR by T ₂	
		R(A)			R(A)
w(B)				w(B)	
	w(B)			w(B)	
		w(B)		w(B)	w(B)

∴ Not View Equal, Not conflict equal.

S ₁ :			S ₂ :		
T ₁	T ₂	T ₃	T ₁	T ₂	T ₃
IR v(A)			IR v(A)		
IR v(B)			IR v(B)		
w(A)			w(A)		
w(B)			w(B)		
w(B)	w(A)		w(B)	w(A)	
w(B)		R(A)	w(B)		R(A)
		w(B)			w(B)

IR — ✓
UR — ✓
FR — ✓

∴ View Equal
(Not conflict equal)

$S_1: w_1(A) w_2(A) w_3(A) R_4(A)$
 $S_2: w_2(A) w_1(A) w_3(A) R_4(A)$

Ans:

$S_1:$	T_1	T_2	T_3	T_4
	$w(A)$	$w(A)$	$w(A)$	$R(A)$

$S_2:$	T_1	T_2	T_3	T_4
	$w(A)$	$w(A)$	$w(A)$	$R(A)$

IR — Not Poset

UR — ✓

FR — ✓

∴ View Equal

(Not conflict equal)

Note —

① If $S_1 S_2$ schedules are conflict equal, then $S_1 S_2$ also view equal schedule.

② If $S_1 S_2$ schedules not conflict equal, then $S_1 S_2$ schedules may or may not view equal

③ If schedule S is conflict serializable schedule, then S also view serializable

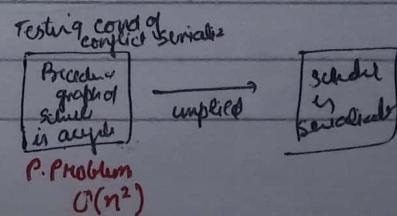
④ If schedule S is not conflict serializable schedule, then Schedule S may or may not view serializable.

⑤ Conflict serializable schedule testing condition is only sufficient but not necessary for serializability testing.

If it fails

We cannot say that it is not serial may or may not

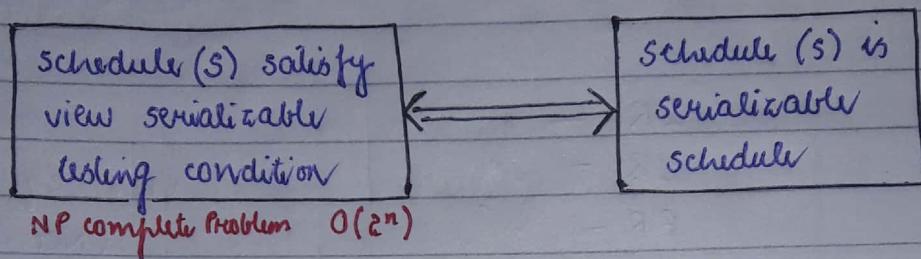
(i.e. only if cond)
→ implied



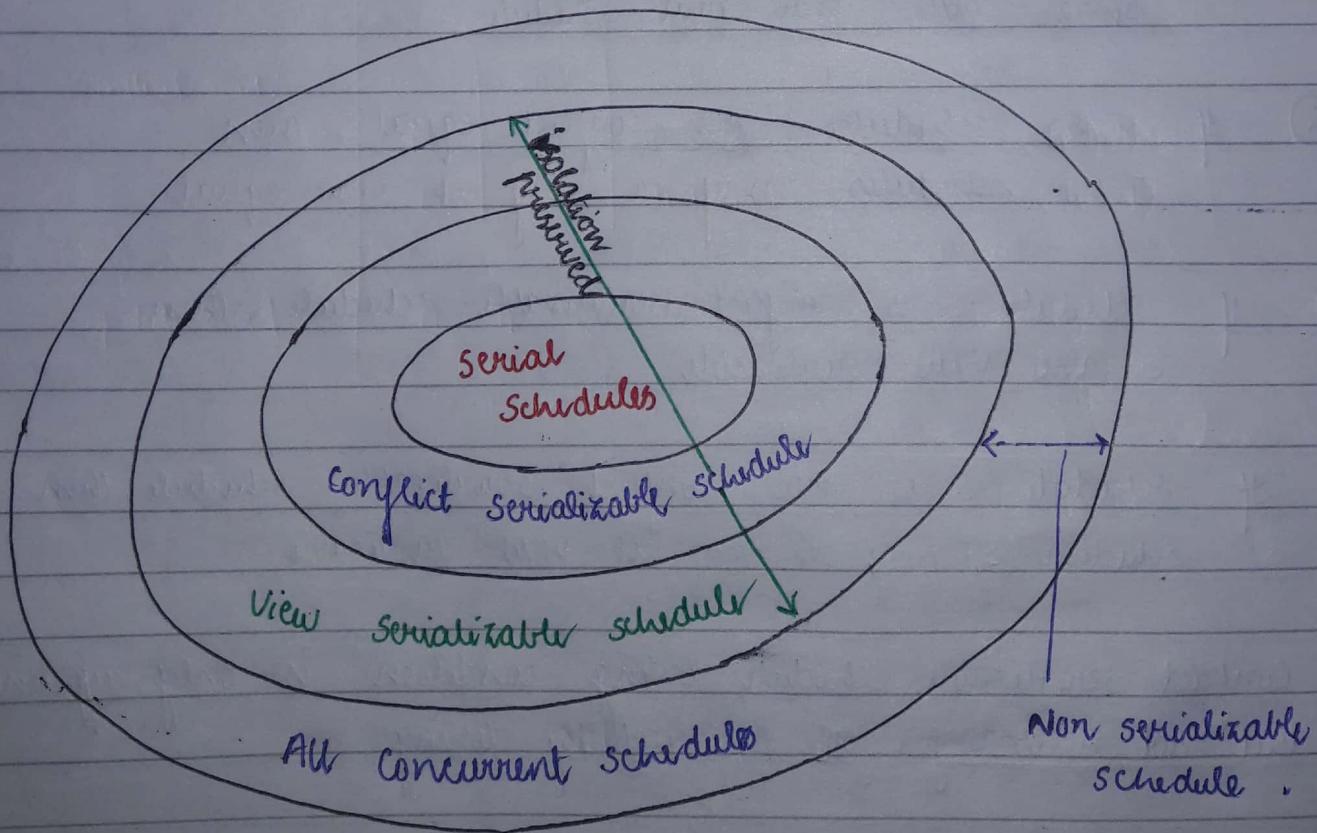
⑥

View serializable schedule testing condition is both sufficient and necessary for serializability testing.

↓
"if and only if" or " \leftrightarrow "



if cond satisfied \rightarrow serializable
cond fails \rightarrow not serializable

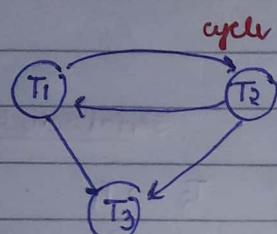


Ques: Test given schedule is view serializable or not

T_1	T_2	T_3
$R_1(A)$		
	$W_2(A)$	
$W_1(A)$		$W_3(A)$

Soluⁿ

a)



∴ Not conflict serializability.

b) For view serializable -

Given schedule (S)		Serial schedule (S')
Final write	$A : \cdot T_2 \cdot T_1 \quad (T_3)$	$(T_1 T_2) \xrightarrow{\text{then}} T_3$ any order
Initial read	data item initial read write	$T_1 \xrightarrow{\text{then}} (T_2 T_3)$

T_1 must be before T_2 and T_3

schedule (S) View serial to

T_1, T_2, T_3
serial schedule

View Serializable

②

T_1	T_2	T_3
	$\delta(A)$	
	$\delta(B)$	
$w(A)$		
$w(B)$	OR	
	$w(B)$	
		$R(A)$
		$w(B)$

Final write :

$$A = T_2 \quad T_1$$

$$B = T_1 \quad T_2 \quad T_3$$

Given Schedule

$$\begin{aligned} T_2 &\rightarrow T_1 \\ (T_1, T_2) &\rightarrow T_3 \end{aligned} \quad \left. \begin{array}{l} T_2 \\ T_1 \\ T_3 \end{array} \right\}$$

Initial read :

data	Initial Read	Write
A	T_2	$T_1 \quad T_2$
B	T_2	$T_1 \quad T_2 \quad T_3$

$$T_2 \rightarrow (T_1, T_2)$$

$$T_2 \rightarrow (T_1, T_3)$$

update read :

$$A : T_1 \rightarrow T_3 \quad \text{ie } w_1(A) \rightarrow R_3(A)$$

(w) and
 (R)
 T_2 other transaction also writes A

$$T_1 \rightarrow T_3$$

and
 $(T_2 \rightarrow T_1 \text{ or } T_3 \rightarrow T_2) \text{ ie } T_1 \xrightarrow{T_2} T_3$

$$T_2 \quad T_1 \quad T_3$$

'View Serializable'

and

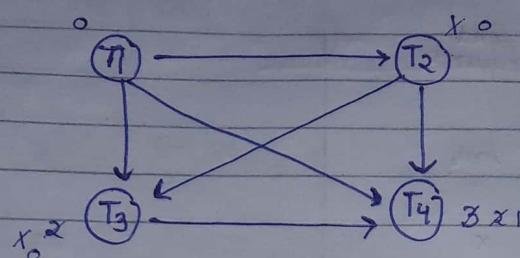
Not conflict serializable.

③ $S : \sigma_1(A) \sigma_2(A) \sigma_3(A) \sigma_4(A) w_1(B) w_2(B) w_3(B) w_4(B)$

- No. of serial schedule conflict equal to sched (S)
- No. of serial schedule view equal to sched (S)
- How many serial schedule view equal to serial S but not conflict equal

$$\Rightarrow 6 - 1 \\ \Rightarrow 5 \text{ Ans}$$

a)



$T_1 \ T_2 \ T_3 \ T_4$

\therefore No. of serial schedule conflict equal = 1 Ans

b)

Given schedule

Final write: B: $T_1 \ T_2 \ T_3 \ (T_4)$

Serial schedule

$(T_1 \ T_2 \ T_3) \rightarrow T_4$

Initial Read:

	data item	Initial R	Wait	
A	T_1	T_1	No one	-

\therefore initial read does not matter

Update read: No one read after write

\therefore

$\underline{\quad} \underline{\quad} \underline{\quad} T_4$

$\Rightarrow 3!$

$\Rightarrow 6 \text{ Ans}$

\therefore 6 serial schedules are view equal.

(T1) $s_3: \tau_1(A) \rightarrow_1 \tau_3(D) \rightarrow_1 w_1(B) \rightarrow_1 w_3(B) \rightarrow_1 \tau_4(B) \rightarrow_1 w_2(C) \rightarrow_1 \tau_5(C) \rightarrow_1 w_4(E) \rightarrow_1 \tau_5(E) \rightarrow_1 w_5(B)$

Given schedule

A = No one

Final write: B = T₁ T₃ T₅

C = T₂

E = T₄ } NO cond required

Initial read:

data	initial R	write
A	T ₁	x
D	T ₃	x

Update read:

write \rightarrow Read

other trans writing data item

all other writings
(T₃, T₅)

w₃(B) \rightarrow $\delta_4(B)$

(T₁, T₅)

w₂(C) \rightarrow $\delta_5(C)$

-

w₄(E) \rightarrow $\delta_5(E)$

-

view Equal Serial sch(s)

(T₁ T₃) \rightarrow T₅

T₁ $\xrightarrow{x(T_3, T_5)}$ T₂ ✓

T₃ $\xrightarrow{x(T_1, T_5)}$ T₄ ✓

T₂ \rightarrow T₅ ✓

T₄ \rightarrow T₅

T₁ T₂ T₃ T₄ T₅

T₃ T₄ T₁ T₂ T₅

Q Am

Classification based on Recoverability -

Concurrent execution may leads:

- ① Irrecoverable schedule
- ② Cascading rollback problem
- ③ Lost update problem

These problems can occur even if schedule is 'Serializable'

Irrecoverable schedule -

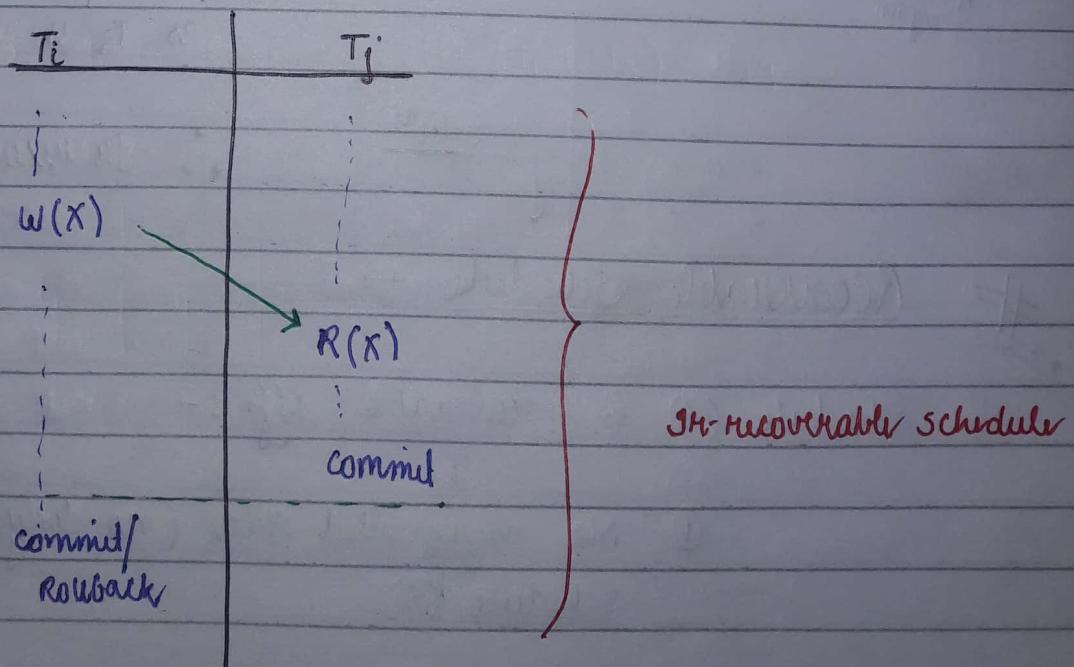


Unable to recover or rollback



(Rollback of committed transaction)

Schedule (s) is irrecoverable iff Transaction T_j reads data item 'X', which is updated by T_i and, commit of T_j before commit / rollback of T_i .

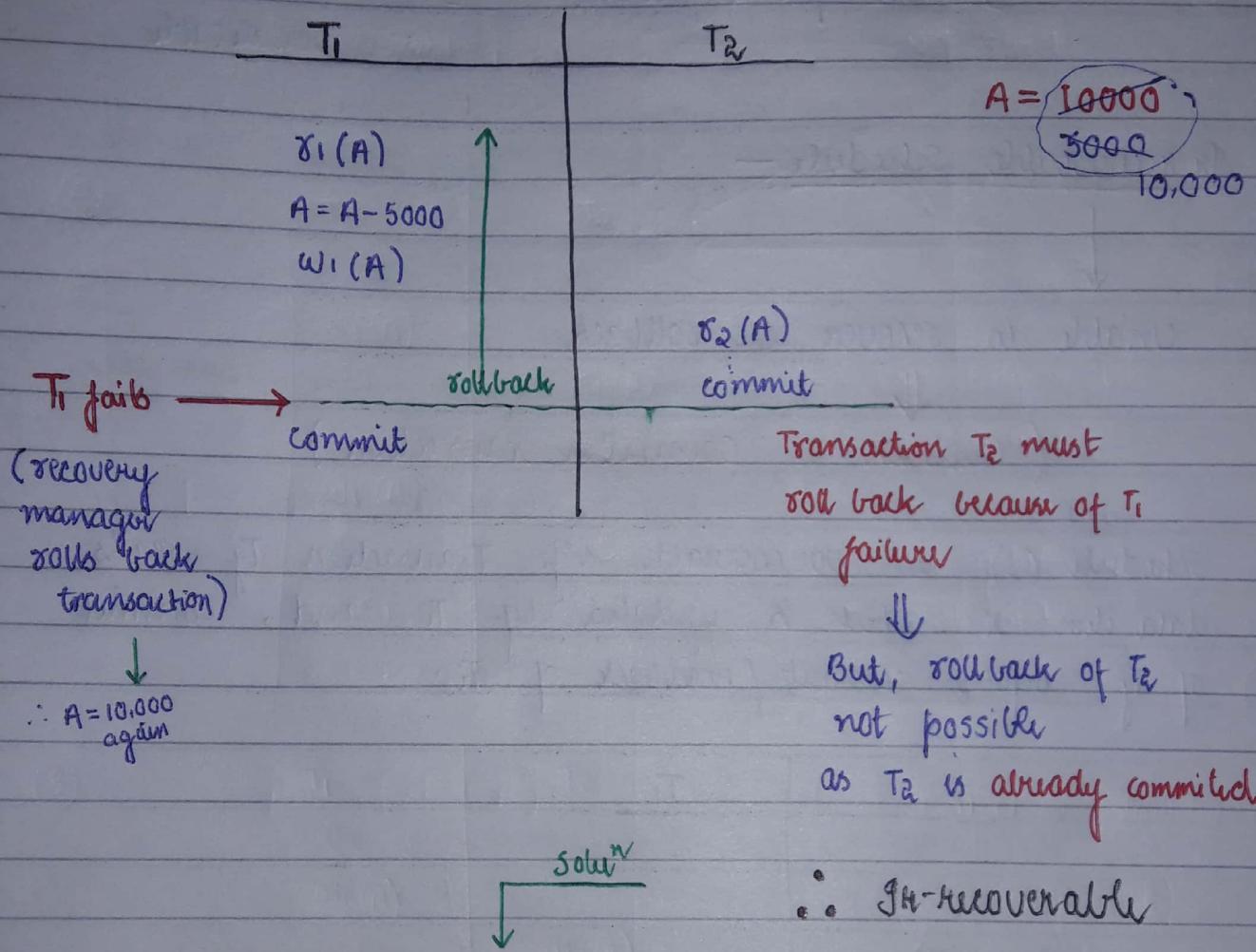


eg -

T_1 : Withdraw 5000 from 'A' [$\delta_1(A)$, $A = A - 5000$, $w_1(A)$, commit]

T_2 : Check balance of 'A' [$\delta_2(A)$, commit]

Let T_1 and T_2 executed concurrently.



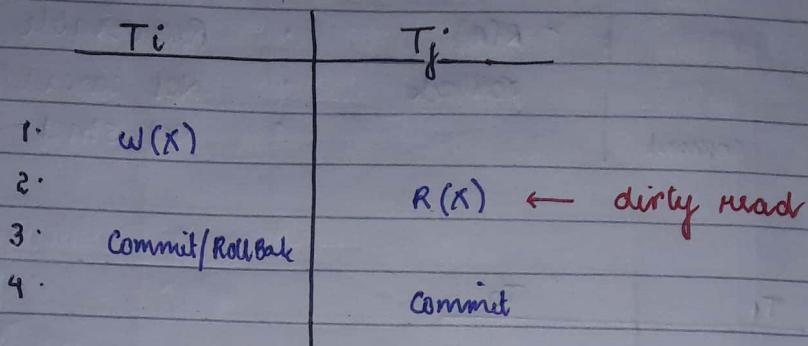
Recoverable Schedule -

Schedule (S) is recoverable iff :-

- ① No uncommitted read (no dirty read) in schedule (S)

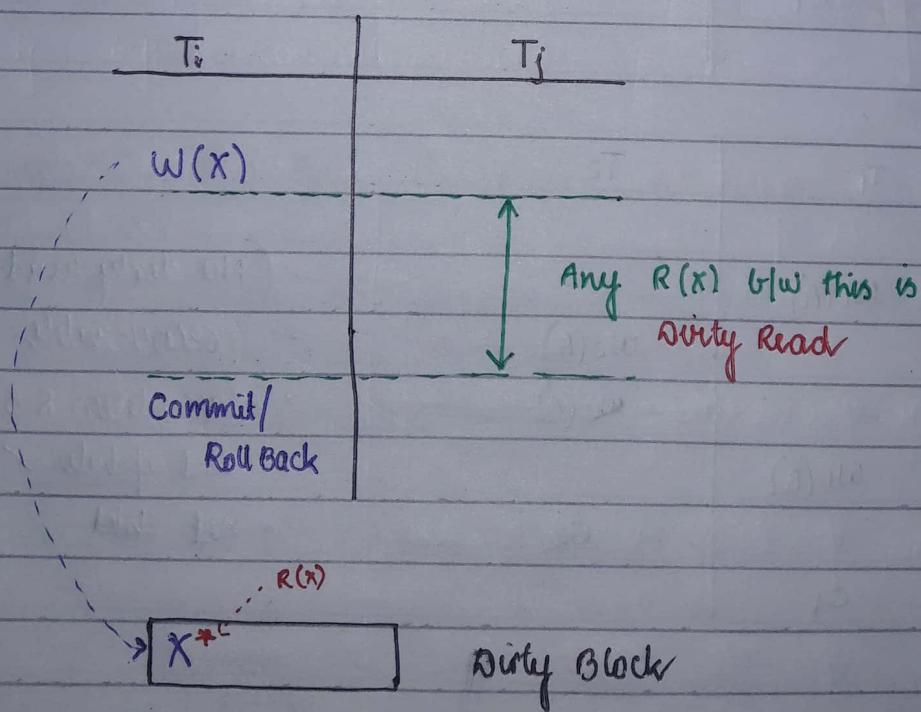
[OR]

(2) If Transaction T_j reads data item 'X' which is updated by transaction T_i , then commit of T_j must be delayed until commit/rollback of T_i .



* Uncommitted read (Dirty Read) —

Transaction T_j reads data item 'X' which is updated by uncommitted transaction T_i



Ques: Test which of the schedule is recoverable -

①

T ₁	T ₂
W(A)	R(A)
	rollback
commit	

- (dirty read exist)
- Recoverable
- NOT cascadeless roll back
- Not strict

②

T ₁	T ₂
$\sigma_1(A)$	$\sigma_2(A)$
	$w_2(A)$
$w_1(A)$	c_2
c_1	

- (NO dirty read)
- Recoverable
- Cascadeless R.B
- lost update \times
- Not strict

③

T ₁	T ₂
$w_1(A)$	$w_2(A)$
	$w_2(B)$
$w_1(B)$	c_2
c_1	

- (NO dirty read)
- Recoverable
- Cascadeless R.B
- lost update \checkmark
- Not strict

(4)

T_1	T_2
$w(A)$	$\rightarrow r(A)$
$w(B)$	$r(B)$
C_1	C_2

- (dirty read exist)

- Recoverable

- (commit of T_2 after C_1)

- Not cascadelless R.B

- lost update X

- Not strict

(5)

T_1	T_2
$w(A)$	$w(A)$ (white done)
C_1	$R(A)$ <small>read this</small>
	C_2

- No dirty read

- ~~Recoverable~~

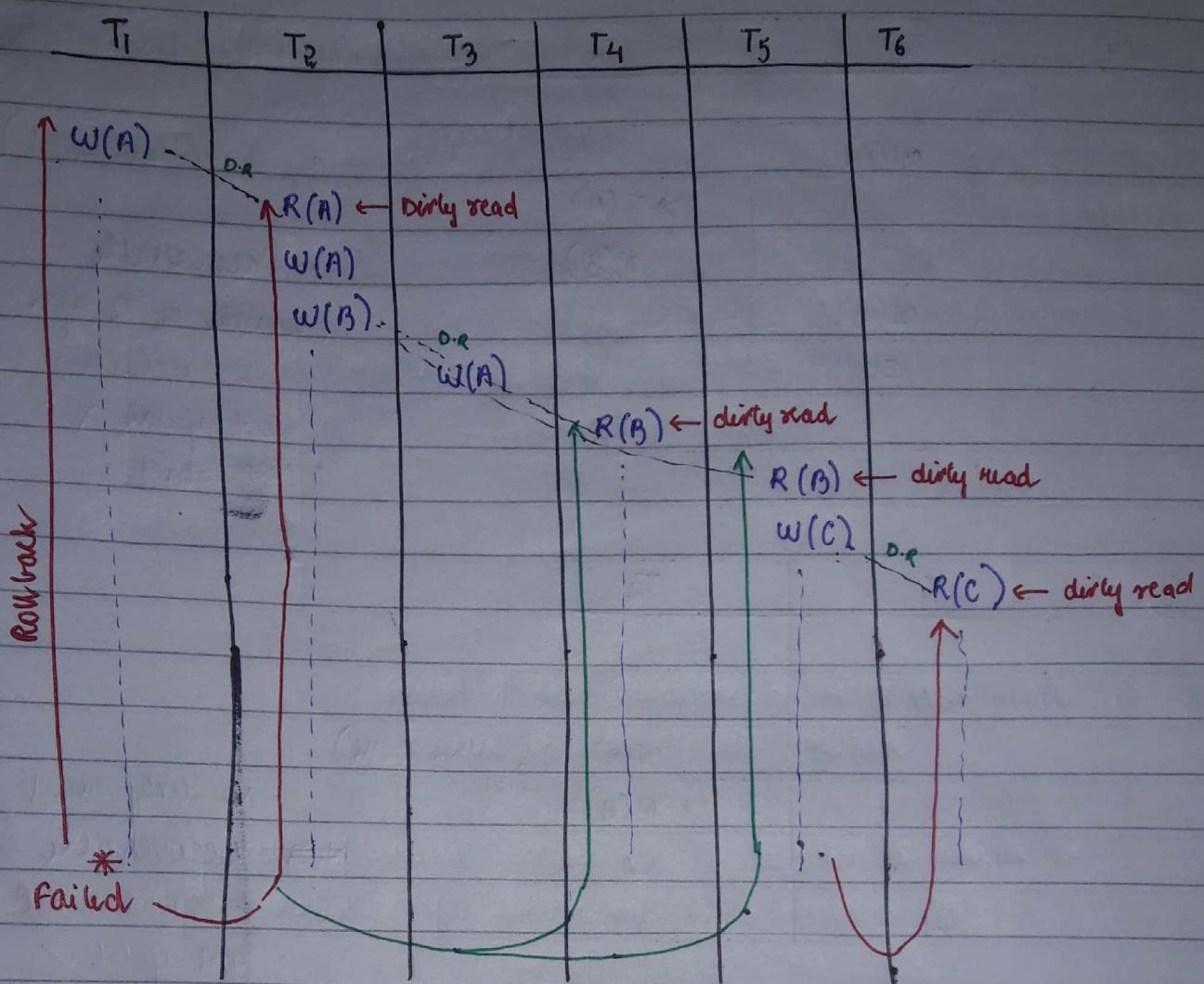
- Cascadelless R.B

- lost update ✓

- Not strict -

Cascading Roll Back Schedule (Problem)

Because of failure of some transaction, many other set of transaction are forced to rollback.



Because of failure of T_1 , forced to rollback T_2, T_4, T_5, T_6

Disadvantage of cascading rollback —

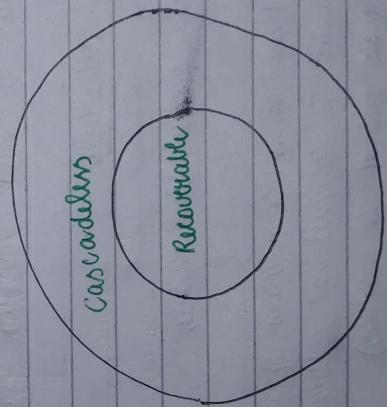
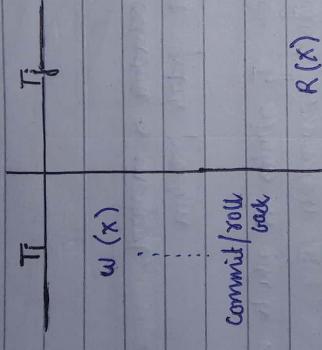
- ① wastage of CPU execution time.
- ② wastage of I/O access cost.

↓ soln

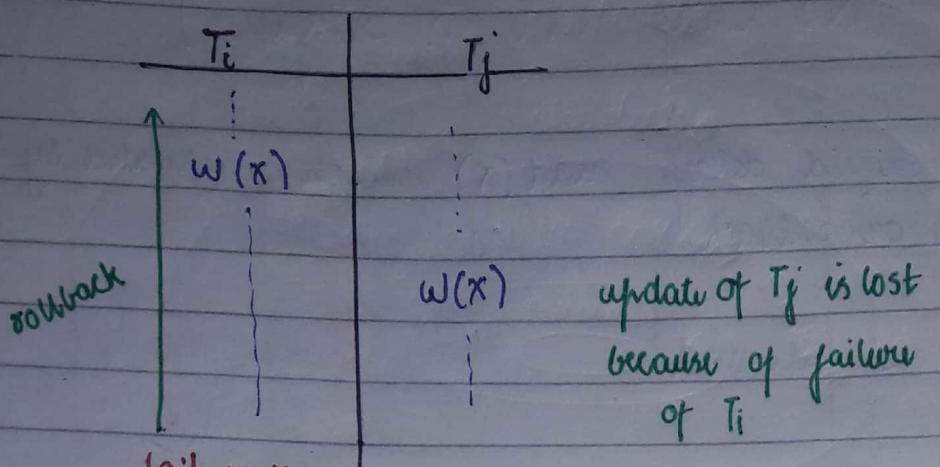
Cascadeless Rollback Recoverable Schedule

Cascadable Rollback Recoverable Schedule — (No Dirty Read in Schedule)

- Transaction T_f should delay $\text{read}(x)$ which is updated by T_i , until T_i commit or rollback.
- Dirty reads are not allowed



Lost Update Problem —



$X : 100$
 $200 \leftarrow$ written by T_i
 $50 \leftarrow$ written by
after failure T_i
rolls back to 100

$w(X)$ update of T_j is lost
because of failure
of T_i

Result of the schedule is incorrect,
because of lost update problem.

- lost update problem occurs if T_j writes ' X ' which is already written by uncommitted transaction T_i

↓
solution is strict Recoverable schedule

Strict Recoverable Schedule —

① Cascadelless rollback recoverable schedule

and ② if T_i writes ' X ' then other transaction T_j write of ' X ' must be delayed until T_i commit or rollback.

(No lost update problem)

T_i	T_j	T_i	T_j
$w(x)$		$w(x)$	
\vdots		\vdots	
commit/rollback		commit/rollback	
	$R(x)$		$w(x)$

and

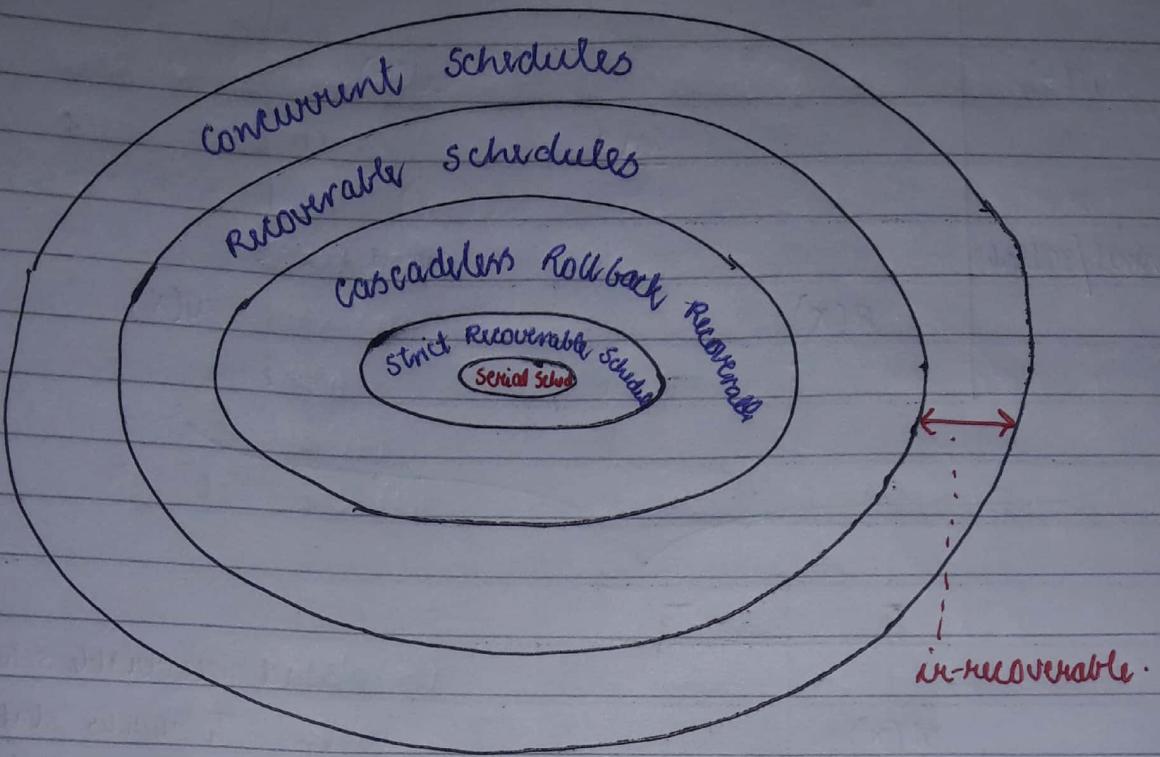
T_i	T_j
$w(x)$	
\vdots	
Commit / Roll	
	$R(x) / w(x)$

Strict recoverable schedule
if T_i writes data item x , then

Read(x) / Write(x) of-
Transaction T_j must
delay until commit/
rollback of T_i

Eg -

T_1	T_2	T_3
$R(A)$		
	$\checkmark \quad w(A)$	
$R(B)$		
		$w(B)$
	$\checkmark \quad Commit$	
$w(A)$		
commit		
		$w(A)$
		Commit



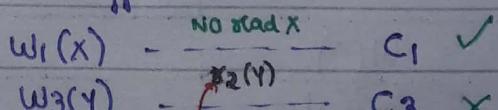
Ques: $S: \tau_1(x) \tau_2(z) \tau_1(z) \tau_3(x) \tau_3(y) w_1(x) w_3(y) \underline{\tau_2(y)}$
 $w_2(z) w_2(y) c_1 c_2 c_3$

which is true?

- a) unrecoverable
- b) recoverable but not cascadelless recover
- c) cascadelless recover (but not strict recov)
- d) strict recov

check for strict rec -

initial read does not affect



dirty read

(also violation for cascadelless)
 \therefore Not cascadelless

Now for recoverable, it should be.

$w_3(y) \dots x_2(y) \dots c_3 c_2$

But

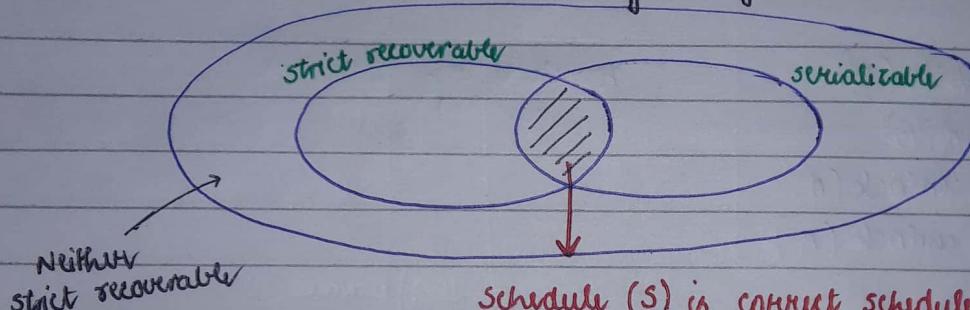
c_2 commit before c_3

∴ Not recoverable

∴ Not recoverable Ans

Note —

- ① strict recoverable schedule may or may not be serializable.
- ② serializable schedule may / may not be recoverable.



schedule (S) is correct schedule iff
 S is strict recoverable and
 S is serializable.

15/11/17

Concurrency Control Protocol -

Concurrency Control Protocol should not allow to execute

- ① Non serializable schedule (violate isolation)
- ② Non strict recoverable schedule (violate Atomity & Durability)

Locking Protocol -

lock is a variable used to identify the status of data item.

Trans (T_1)

lock (A) : granted by concurrency controller.

R(A)

w(A)

lock (B) : denied by concurrency controller.

}

lock (B)

: granted by C.C.

R(B)

unlock (A)

unlock (B)

Types of lock -

- ① Shared lock (S) → read only lock
- ② Exclusive lock (X) → read/write lock.

Eg - Trans (T_1)

S(A) : granted shared lock

R(A) { only read

X(B) : granted exclusive lock.

R(B) { read and write

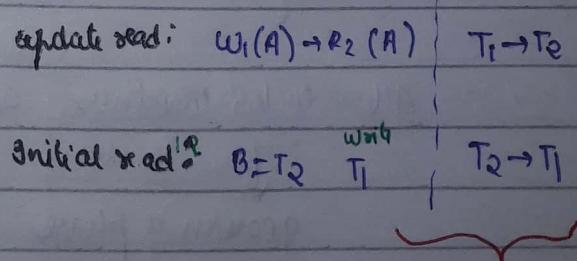
lock compatible Table —

		Held by T_i	
		S	X
Requested by T_j	S	Yes	No
	X	No	No

T_1	T_2
S(A)	S(A) : granted
R(A)	R(A)
S(B)	X(B) : denied
R(B)	

	T_1	T_2
W(A)	R(A)	
	R(B)	
W(B)		
C1 ✓	C2 ✓	

Applying view serializable



serial
schedule
Not possible

∴ Non serializable
Non recoverable

So, we need to place locks in some order.

T_1	T_2
$X(A)$	
$W(A)$	
$V(A)$	
	$S(A)$
	$R(A)$
	$V(A)$
	$S(B)$
	$R(B)$
	$V(B)$
$X(B)$	
$W(B)$	
$V(B)$	

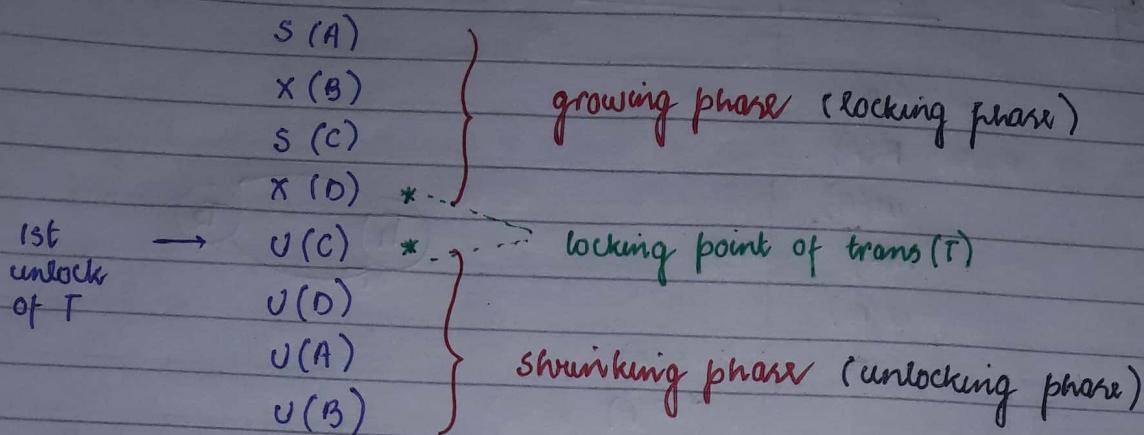
v. unlock

Shared / Exclusive locks not sufficient to avoid
Non serializable and non strict recoverable.

2 Phase Locking Protocol (2PL) — (guaranteed serializability)

- Transaction (T) allowed to request lock on any data item in any mode (S/X) until first unlock of - Trans (T)
- All locks together, ^{i.e 1st} and unlock together ^{i.e later}
 - ↓ growing phase
 - ↓ shrinking phase

Trans (T)



lock point — Position of last lock / 1st unlock of transaction T.

T₁: Transfer 500 from A to B

T₂: display total balance of AB

T₃: set A, B as 2000

Trans (T₁)

$\left\{ \begin{array}{l} X(A) \\ R(A) \\ W(A) \\ X(B) \\ U(A) \\ R(B) \\ W(B) \\ U(B) \end{array} \right.$
 commit

Trans (T₂)

$\left\{ \begin{array}{l} S(A) \\ R(A) \\ S(B) \\ U(A) \\ R(B) \\ U(B) \\ disp(A+B) \\ commit \end{array} \right.$

Trans (T₃)

$\left\{ \begin{array}{l} X(A) \\ W(A) \\ X(B) \\ U(A) \\ W(B) \\ U(B) \\ commit \end{array} \right.$

Satisfy 2PL

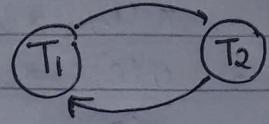
Satisfy 2PL

Satisfy 2PL

∴ Execute concurrently : Concurrent execution of T₁ T₂ T₃
(guaranteed serializable)

Eg -

	T ₁	T ₂
X(A)		
R(A)		
w(A)		
X(B)		
U(A)		
		S(A)
		R(A)
		S(B)
		R(B)
R(B)		
w(B)		
U(B)		
commit		

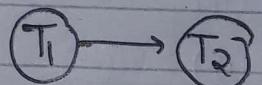


S(B) ← denied till T₁ unlocks B

Not Allowed
by 2PL



	T ₁	T ₂
X(A)		
R(A)		
w(A)		
X(B)		
U(A)		
		S(A)
		R(A)
R(B)		
w(B)		
U(B)		
		S(B)
		R(B)



Conflict serializable S

Allowed by 2PL

Note

- If schedule executed by 2PL, then schedule guaranteed is conflict serializable schedule.

- Conflict equal schedule is based on lock points order of the transaction.

S:	T ₁	T ₂	T ₃
	*	*	*
	*	*	*
	*	*	*

Schedules executed by 2PL

↓
guaranteed conflict
serializable schedule

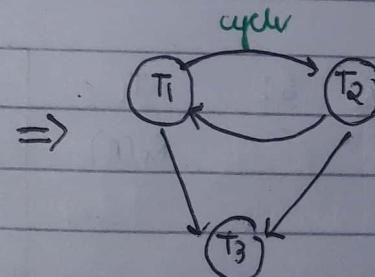
1. Equal serial schedule

T₂ → T₁ → T₃

(order of lock
point)

- If schedule is not conflict serializable schedule (cyclic precedence graph schedule), then schedule not allowed to execute by 2PL

S:	T ₁	T ₂	T ₃
	R(A)		
		w(A)	
	w(A)		w(A)



Not conflict serializable

Checking view serializable,

T₁ → (T₂ T₃)

(T₁ T₂) → T₃

T₁ T₂ T₃

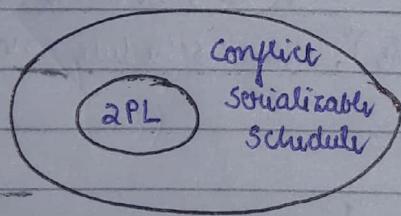
View serializable

Writing using locks -

T_1	T_2	T_3
$X(A)$		
$R(A)$		
	$X(A)$ ← denied	
	$W(A)$	
$W(A)$		
$U(A)$		
		$W(A)$

∴ Above schedule is not conflict serializable but view serializable
 "Not allowed by 2PL"

- Every schedule which is allowed by 2PL is always conflict serializable, but not every conflict serializable schedule is allowed by 2PL



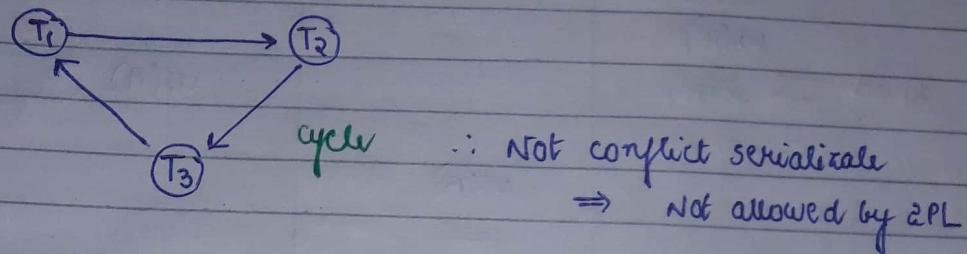
Ques: S:

	T_1	T_2	T_3
	$R_1(A)$		
		$W_2(A)$	
		$W_2(B)$	
			Cascading Recov X
			strict " X
			$R_3(B)$
			$R_3(C)$
			(May or may not recoverable)
	$W_1(C)$		

which is true?

Schedule (s) is

- (a) Conflict SS and allowed by 2PL
- (b) conflict SS but not allowed by 2PL
- (c) not conflict SS and "
- (d) " and view serializable.

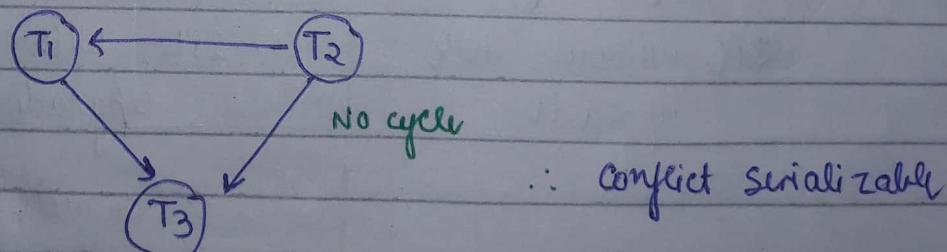


Ques:

T ₁	T ₂	T ₃
$x(A)$ $w(A)$ $x(B)$ $v(A)$ $x(A)$ $w(A)$ $v(B)$	$x(A)$ $w(A)$ $v(A)$ $w(B)$ $v(B)$ $x(A)$	$w(A)$ $w(B)$ $v(B)$

- ✓ Recoverable
- ✓ Cascade Recov
- ✗ Gunit "

which is true, above options.



View

Allowed by 2PL -

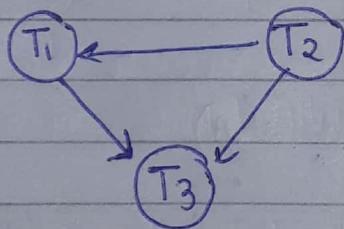
(a)

∴ Equal serial schedule
 $T_2 \rightarrow T_1 \rightarrow T_3$

Answer:

$w_2(A)$ $w_1(A)$ $w_3(A)$ $w_2(B)$ $w_1(B)$ $w_3(B)$

T_1	T_2	T_3
	$x(A)$	
$x(A)$	$w(A)$	
$w(A)$	$x(B)$ $U(A)$	
$x(B)$ $U(A)$	$w(B)$	$x(A) \leftarrow \text{denied}$ $w(A)$
$w(B)$	$U(B)$	$w(B)$



conflict serializable

Not allowed by 2PL

(from above fig)

Lock Upgrading 2PL (vs) without lock upgrade 2PL -

2PL without lock upgrade

2PL with lock upgrade

(default 2PL)

S:	T ₁
X(A)	
R(A)	↑ even if no change in value of A.
:	NO other transaction allowed to use data item 'A' ↓ in shared mode
W(A)	
U(A)	

S:	T ₁
S(A)	
R(A)	
:	↑ Other transaction can use data item 'A' in shared mode
X(A)	
W(A)	

lock upgrade of transaction
must complete in growing
phase.

less degree of concurrency

T ₁	T ₂
X(A)	
R(A)	
	S(A) ← denied
W(A)	R(A)
U(A)	

Not allowed by 2PL without
lock upgrade (even when
it is serializable)

more degree of concurrency

T ₁	T ₂
S(A)	
R(A)	
	S(A) ← grants
W(A)	R(A)
U(A)	

Allowed to execute with
lock upgrade.

Ques: Test the given schedule allowed by 2PL or not?

- ① S: $\sigma_1(A) \ w_1(A) \ \sigma_2(A) \ \tau_2(B) \ \sigma_1(B) \ w_1(B)$

Soluⁿ:

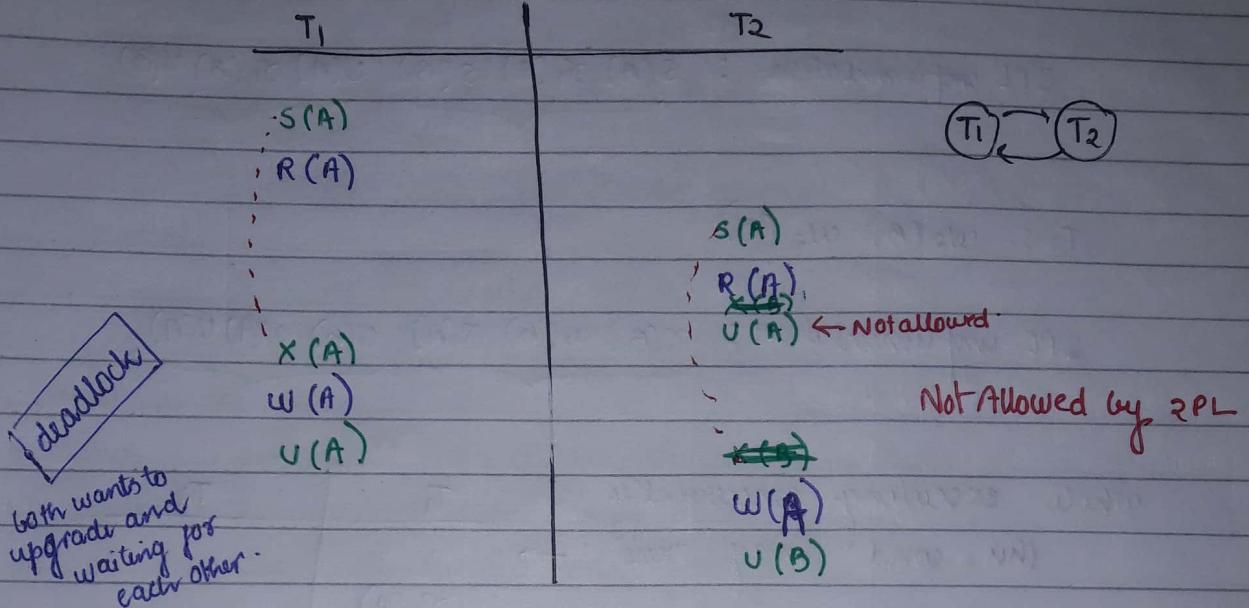
T_1	T_2
$X(A)$	
$R(A)$	
$w(A)$	
$S(A)$	$S(A)$
$U(A)$	$R(A)$
↓	
$R(B)$	$S(B)$
$X(B)$	\leftarrow denied
$w(B)$	$R(B)$
$U(B)$	$U(B)$
↓	
Not allowed during shrinking phase	
Not allowed by 2PL	

$T_1 \rightarrow T_2$
cyclic
Not Conflict
Not 2PL.

- ② S: $\sigma_1(A) \ \sigma_2(A) \ \tau_2(B) \ w_1(B) \ w_1(A)$

T_1	T_2
$S(A)$	$S(A)$
$R(A)$	$R(A)$
↓	
$X(B)$	$S(B)$
$w(B)$	$R(B)$
↓	
$w(A)$	$U(A)$
$U(B)$	$U(B)$
$U(A)$	
↑	
upgrade	
Allowed by 2PL	

③ $S: \tau_1(A) \tau_2(A) w_1(A) w_2(A)$



Limitations of 2PL —

- ① 2PL restriction may lead to deadlock.
- ② 2PL restriction may lead to starvation
- ③ 2PL condition not sufficient to avoid:
 - a) 1R-recoverable schedules
 - b) Cascading rollback problem
 - c) Strict recoverable problem, lost update problem

Eg - 2PL restriction may leads deadlock

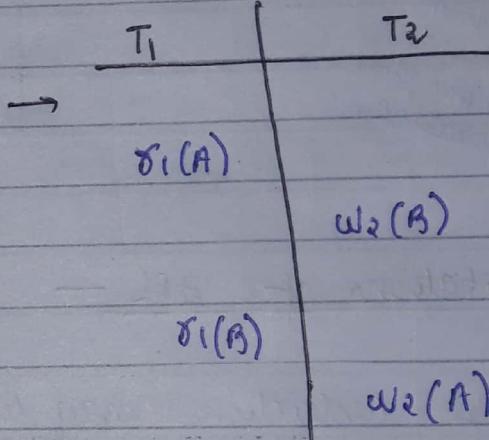
$T_1 : \sigma_1(A) \sigma_1(B)$

2PL implementation : $S(A) \sigma_1(A) S(B) | U(A) \sigma_1(B) U(B)$

$T_2 : w_2(B) w_2(A)$

2PL implementation : $X(B) w_2(B) X(A) | U(B) w_2(A) U(A)$

while executing concurrently
this cond may arise



2PL implementation

T_1	T_2
$S(A)$	
$\sigma_1(A)$	
	$X(B)$
	$w_2(B)$
denied $\rightarrow S(B)$	
$\sigma_1(B)$	
	$X(A) \leftarrow$ denied
	$w_2(A)$

Forms Deadlock

ques: data items A₁ A₂ A₃ ... - A_n

Protocol —

- ① Transaction should lock all required data item in proper sequence of A₁, A₂, ... A_n.
- ② Transaction should perform read / write's
- ③ Transaction unlock's data.

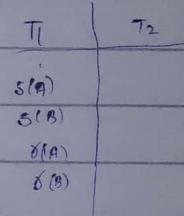
Which is true ?

- a) Protocol guaranteed serializable and No deadlock
- b) Guaranteed serializable but may cause deadlock
- c) Not guaranteed serializable and free from deadlock.
- d) None.

1st step — lock all the data item

2nd step — Perform R/W

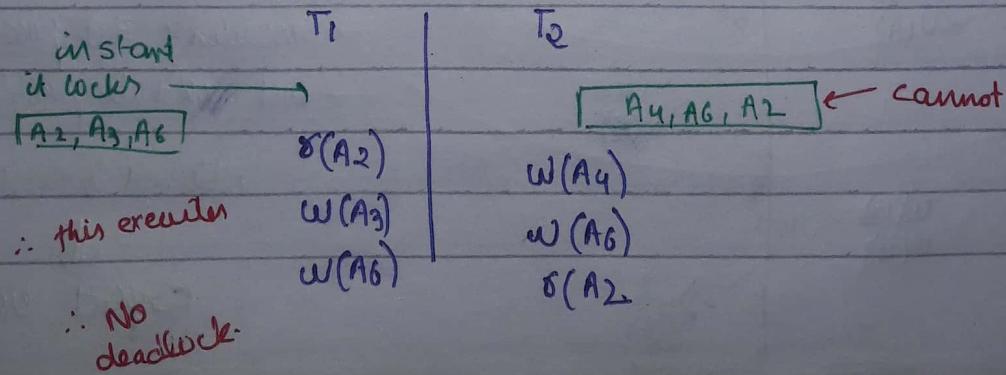
3rd step — unlock all the data item



It is 2PL

∴ guaranteed
serializable.

Since, it lock all the data needed in beginning.



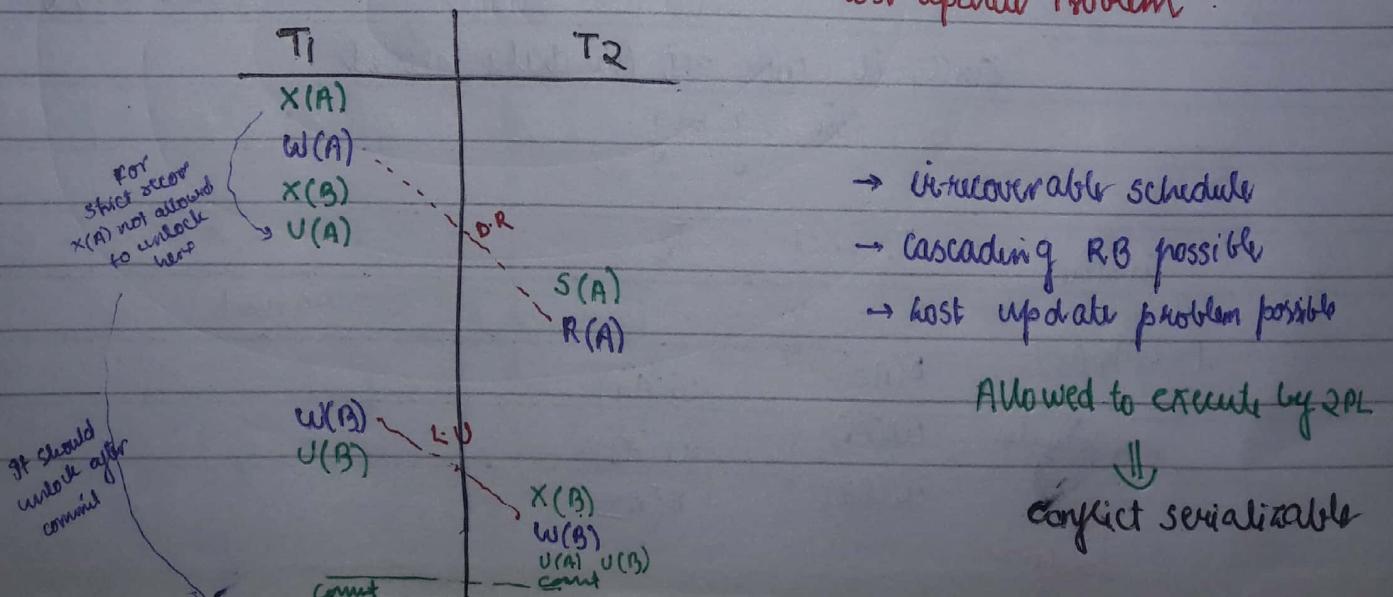
Eq - 2PL may form starvation.

T_1	T_2	T_3	T_4
	$S(A)$			
denied bcoz of $T_2 \rightarrow X(A)$				
		$S(A)$		
		$U(A)$		
denied bcoz of $T_3 \rightarrow X(A)$			$S(A)$	
			$U(A)$	
denied bcoz of $T_4 \rightarrow X(A)$				$S(A)$

Starvation

Eq - 2PL restriction not sufficient to avoid

- Unrecoverable
- Cascading Roll back
- Lost update problem



Strict Recoverable condition using lock —

All exclusive locks of transaction (T) should hold until the commit/ rollback of trans (T)

T_1	T_2
$X(A)$ $w(A)$ unlock only after commit	$S(A) : X(A)$ $R(A) : w(A)$

↓ modified 2PL protocol .

Strict 2PL Protocol —

- Basic 2PL : lock request of trans (T) not allowed in shrinking phase of trans T .

and

- Strict Recoverable : All exclusive locks of transaction T must hold until commit/ rollback of trans T

→ Strict 2PL protocol guarantees —

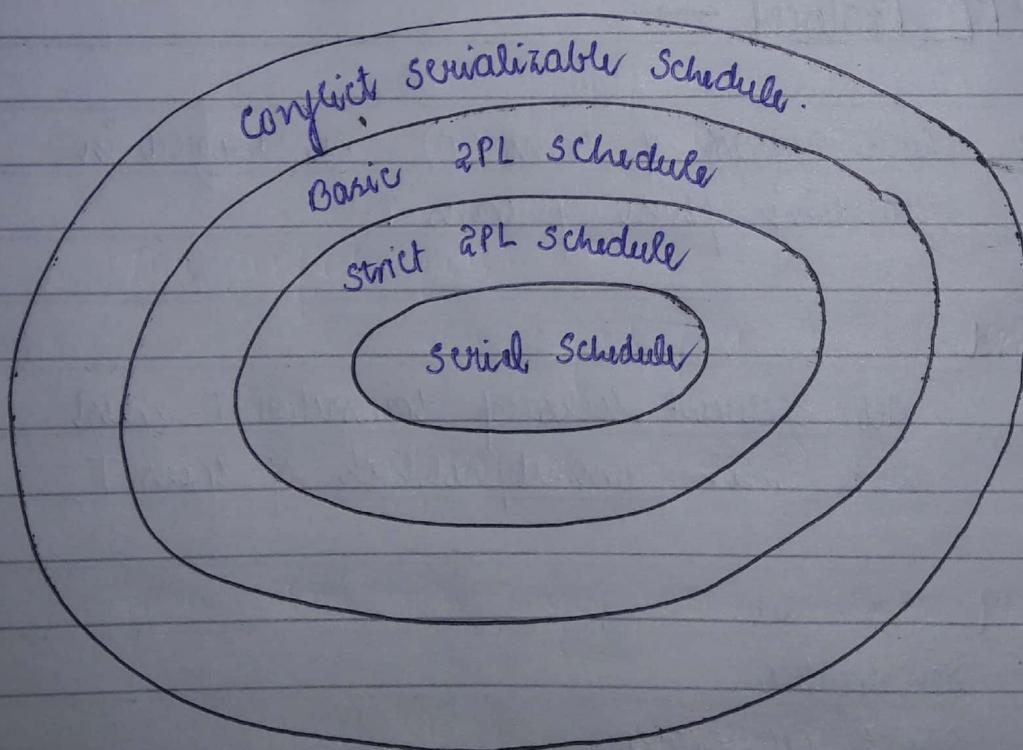
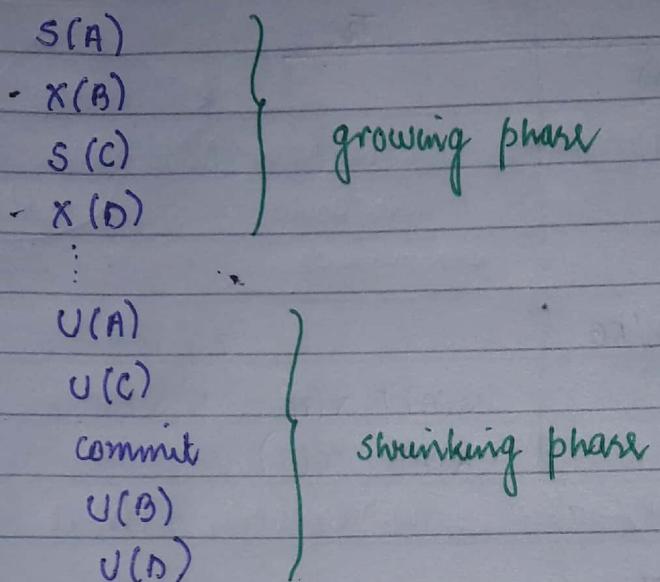
- (1) Serializable
- (2) Strict Recoverable-

- It is also not free from -

- ① Deadlock
- ② Starvation

Trans (T)

strict 2PL transaction



Pg-61

(23) Consider the transaction.

$T_1 : \delta_1(A) \delta_1(B) w_1(B)$

$T_2 : \delta_2(A) \delta_2(B) w_2(B)$

Total concurrent schedule = 6C_3

= 20

How many now serial schedules w/w T_1 and T_2 are serializable.

Sol:

Concurrent execution of T_1, T_2
equal to $T_1 \rightarrow T_2$ serial

Concurrent execution of T_1, T_2
equal to $T_2 \rightarrow T_1$ serial

equal

S:	T_1	T_2
	$\delta_1(A)$	
	$\uparrow \delta_1(B)$	
	$w_1(B)$	
		$\delta_2(A)$
		$\downarrow \delta_2(B)$
		$w_2(B)$

1st possible
last possible T_2 (down)
that should follow
same sequence

$w_1(A) \delta_2(B)$
(Not allowed to interchange)

$S_0 : \delta_1(A) \delta_1(B) w_1(B) \delta_2(B) w_2(B)$

Now before $\delta_2(B)$
we see what is there,
and all possible
of it.

equal

S' :	T_1	T_2
	$\delta_2(A)$	
	$\delta_2(B)$	
	$w_2(B)$	

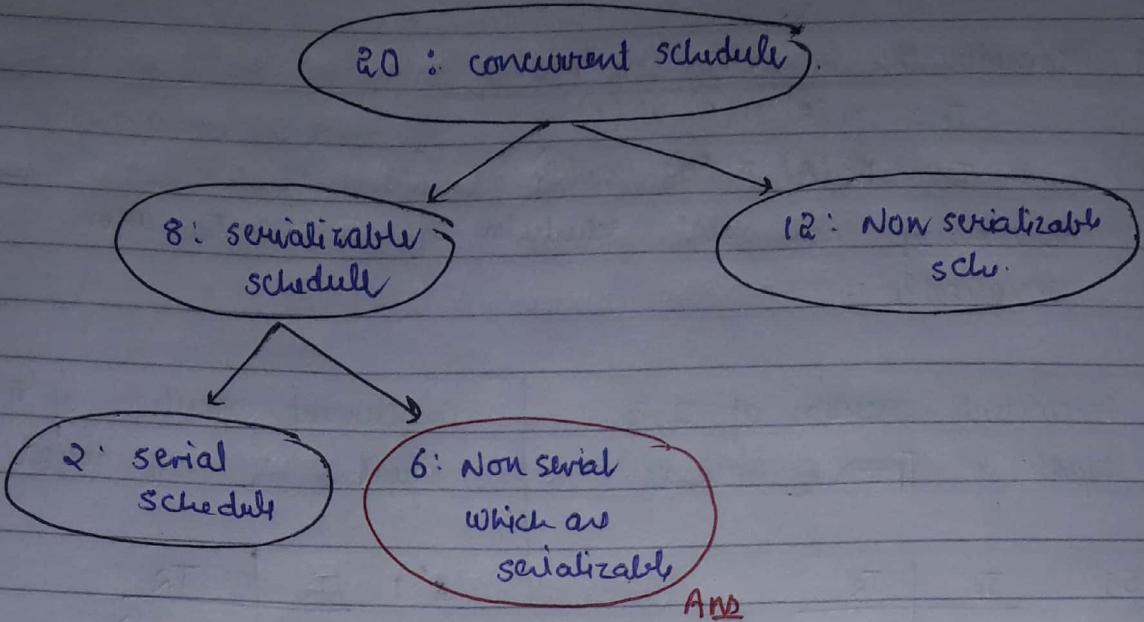
$S' : \delta_2(A) \delta_2(B) w_2(B) \delta_1(B) w_1(B)$

To maintain
 T_2 order
($T_2 \rightarrow T_1$)

$\delta_1(A)$

4 serializable
schedule

\therefore 4 serializable
schedule



Pg 61

(27)

5: $\delta_1(A) w_1(B) \delta_2(A) w_2(B) \delta_3(A) w_3(B)$

conflict pair

conflict pair

a) No. of schedules conflict equal to given schedule 5

so we need to apply permutation
combinations

Topological order

gives only no. of
serial schedule conflict
equal to schedule 5

∴ we can't use this

for conflict equal,

the conflict pair should be in same order -

$w_1(B) \dots w_2(B) \dots w_3(B)$

same order.

Now, start with longest possible relation and place other.

$\delta_1(A) \cdot w_1(B)$

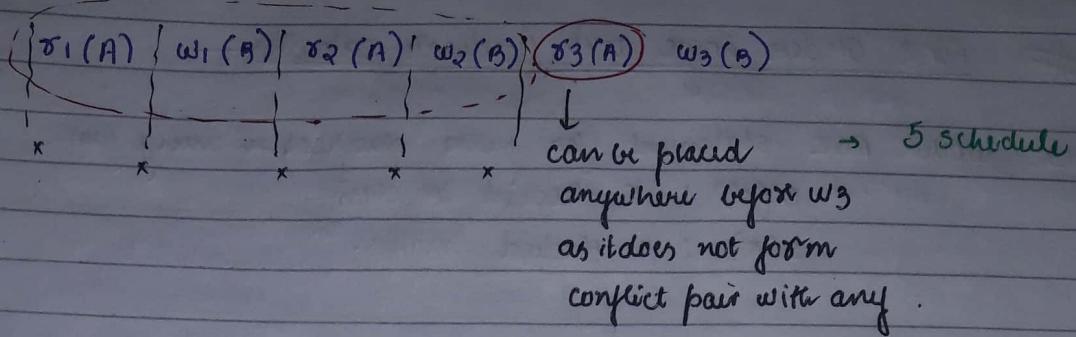
$w_2(B)$

$w_3(B)$

} 3 schedule.

for $\delta_2(A)$

can be placed anywhere $\delta_3(A)$
before $w_3(B)$



$$\therefore \text{Total} = 3 \times 5 \\ = 15 \text{ schedule Ans}$$

b) NO. of schedules ^{view} equal to schedule (S).

S: $T_1(A) \quad w_1(B) \quad T_2(A) \quad w_2(B) \quad T_3(A) \quad w_3(B)$

Here, no violation of initial read, as no one is writing it.
No read update.

Checking for final write.

B: $T_1 \quad T_2 \quad T_3 \quad (T_1 \ T_2) \rightarrow T_3$

i.e. 2 options.

from previous: $w_1(B) \quad w_2(B) \quad w_3(B) \quad \{ \quad 15$

$w_2(B) \quad w_1(B) \quad w_3(B) \quad \{ \quad 15$

30

Ans

H/W Ques: S: $w_1(A) \quad w_1(B) \quad T_2(A) \quad w_2(B) \quad T_3(A) \quad w_3(B)$

a) Find No. of conflict equal

(b) " " " " view Equal.

TRC (Tuple Relational logic)

First Order logic and Predicate calculus -

Ques: write FOL statement for given specification using fun

$c(x)$: x student in class

$s(x)$: x studied maths.

① Some student in class studied maths

$$\exists x (c(x) \wedge s(x))$$

Some 'x' such that

x student in class

and

x studied maths.

② Every student in class studied maths.

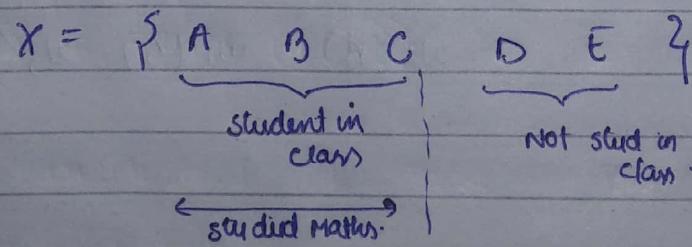
$$\forall x (c(x) \rightarrow s(x))$$

For every x

if x student in class, then
 x must study maths.

Wrong statement $\rightarrow \forall x (c(x) \wedge s(x))$

every x must be
student in class.



$$\forall x (c(x) \rightarrow s(x))$$

T

$$\forall x (c(x) \wedge s(x))$$

F

③ At least 2 student in class studied maths.

2, or more.

$x:$



$y:$



some x student
in class and
studied maths

some y student
in class and
studied maths

\Rightarrow All - Almost 1
 $\exists x (c(x) \wedge s(x)) \wedge$
 $c(x_2) \wedge s(x_2)$
 $\wedge x_1 \neq x_2$

and
 $x \neq y$.

$$\exists x (c(x) \wedge s(x) \wedge \exists y (c(y) \wedge s(y) \wedge (x \neq y)))$$

$$\Rightarrow \exists x \exists y (c(x) \wedge s(x) \wedge c(y) \wedge s(y) \wedge (x \neq y))$$

Ans

④ Only one student in class studied maths.

$$\text{Only one} = \left(\begin{array}{l} \text{At least one student} \\ \text{in class studied maths} \end{array} \right) - \left(\begin{array}{l} \text{at least two stud} \\ \text{in class stud. maths} \end{array} \right)$$

= at least one but not at least 2

$$P \cdot Q \equiv P \wedge \neg Q$$

↑
at least
one
↑
at least
2

$$= P \wedge \neg Q$$

$$\Rightarrow \exists x (c(x) \wedge s(x)) \wedge \neg \left(\exists x \exists y (c(x) \wedge s(x) \wedge c(y) \wedge s(y) \wedge (x \neq y)) \right)$$

OR

Method 2:

{ some x student
in class and and
studied maths } \rightarrow { every y who
are student in
class and
studied maths } \rightarrow $(y = x)$ }

$$\exists x (c(x) \wedge s(x) \wedge \forall y ((c(y) \wedge s(y)) \rightarrow (y = x)))$$

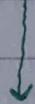
TRC -

TRC query format:

$$\{ T \mid P(T) \}$$

T: Tuple variable

P(T): formula over tuple variable



Results set of tuple (T)

those satisfy formula P(T)

$\exists T_1 \in \text{stud} (P(T_1))$ } $\exists T_2 \in \text{course} (P(T_2))$ }
 T_1, T_2 are
 "Bounded tuple variable"

→ Tuple variable used for result of TRC query must be
free at tuple variable

Ques:

student (sid, age)

course (cid, instructor)

enroll (sid, cid, fee)

i) Retrieve sid's enrolled some course taught by Korth.

Enroll (sid cid ...) Course (cid inst) \equiv Korth

$\exists T_1 \left\{ \begin{array}{l} s_1 \\ c_1 \end{array} \right. \quad \exists T_2 \left\{ \begin{array}{l} c_1 \\ \text{Korth} \end{array} \right. \quad \text{inst} = \text{Korth}$

$\left. \begin{array}{l} T_1 \cdot \text{sid} / \exists T_1 \in \text{Enroll} \exists T_2 \in \text{Course} \\ (T_1 \cdot \text{cid} = T_2 \cdot \text{cid} \wedge T_2 \cdot \text{inst} = \text{Korth} \wedge T_1 \cdot \text{sid} = T_2 \cdot \text{cid}) \end{array} \right\}$

There must be a free variable in result.

ii) Retrieve sid's enrolled some course taught by Korth
(and)

some course taught by Navathe.

in R.A -

$\pi_{\text{Sid}} (\text{E} \bowtie \sigma(c)) \cap \pi_{\text{Sid}} (\text{E} \bowtie \sigma(c))$

$\underbrace{\qquad\qquad\qquad}_{P}$

$\text{inst} = \text{Korth}$

$\underbrace{\qquad\qquad\qquad}_{Q}$

Note -

$P \cap Q \equiv \{x \mid x \in P \wedge x \in Q\}$

$P \cup Q \equiv \{x \mid x \in P \vee x \in Q\}$

$P - Q \equiv \{x \mid x \in P \wedge x \notin Q\}$

$\{ T \cdot sid / \exists T_1 \in Enroll \exists T_2 \in Course (T_1 \cdot cid = T_2 \cdot cid \wedge T_2 \cdot Inst = Koth \wedge T \cdot sid = T_1 \cdot sid) \}$

$\wedge \exists T_1 \in Enroll \exists T_2 \in Course (T_1 \cdot cid = T_2 \cdot cid \wedge T_2 \cdot Inst = Nawal \wedge T \cdot sid = T_1 \cdot sid) \}$

Ques: what is the english statement of -

① $\{ T \cdot sid / \exists T_1 \in Enroll (T \cdot sid = T_1 \cdot sid) \wedge$

$\exists T_1 \in Enroll \exists T_2 \in Enroll$

$(T_1 \cdot sid = T_2 \cdot sid \wedge T_1 \cdot cid \neq T_2 \cdot cid \wedge$

$T \cdot sid = T_1 \cdot sid) \}$

$\Rightarrow P \wedge \neg Q$

$\Rightarrow P - Q$

$= (Sid's enroll \text{ at least one course}) - (Sid's enroll \text{ at least 2 courses})$

All - (sid of different courses)

\therefore Sid's enrolled only one course Ans

② $\{ T \cdot sid / \exists T_1 \in Stud (T \cdot sid = T_1 \cdot sid) \wedge$

$\exists T_1 \in Stud \exists T_2 \in Stud$

$(T_1 \cdot age < T_2 \cdot age \wedge T \cdot sid = T_1 \cdot sid) \}$

\Rightarrow All students sid - sid of student whose age is less than some other student

\Rightarrow Sid's whose age is max Ans

Ques:

- 1) Retrieve Sid's enrolled every course.

In R.A :

$$\pi_{\text{Sid}, \text{Cid}} (\text{enroll}) / \pi_{\text{Cid}} (\text{course})$$

Sid's of enroll (T_1) such that

for every course of course (C)

there exist some record in

enroll (T_2) with $T_2 \cdot \text{Cid} = C \cdot \text{Cid}$ and

$T_2 \cdot \text{Sid} = T_1 \cdot \text{Sid}$

<u>Enroll (sid cid)</u>	<u>course (cid ...)</u>			<u>Enroll (sid cid)</u>
$T_1 \quad S_1$	$\forall C$	C_1	$\exists T_2 \quad S_1 \quad C_1$	
		C_2	$S_1 \quad C_2$	
		C_3	$S_1 \quad C_3$	

TRC:

$$\therefore \{ T \cdot \text{Sid} / \exists T_1 \in \text{Enroll} (T \cdot \text{Sid} = T_1 \cdot \text{Sid}) \} \wedge$$

$\forall C \in \text{course} \exists T_2 \in \text{Enroll} (T_2 \cdot \text{Cid} = C \cdot \text{Cid} \wedge$

$T_2 \cdot \text{Sid} = T \cdot \text{Sid})$

Ans

- 2) Retrieve Sid's enrolled every course taught by Korth.

In R.A -

$$\pi_{\text{Sid}, \text{Cid}} (\text{enroll}) / \pi_{\text{Cid}} (\sigma_{\text{C. Inst} = \text{KORTH}} (\text{course}))$$

We cannot use ques ① result just by adding an additional cond. $C. \text{Inst} = \text{KORTH}$



as it checks for all courses and all courses must be taught by KORTH Sid

C_1	KORTH
C_2	"
C_3	"
C_4	newathu

S_1	C_1
S_1	C_2
S_1	C_3
S_1	C_4

stop Sid in my
ie enrolled
all KORTH course but
 $C_4 \neq \text{KORTH}$ Newathu

$\forall x \ (C(x) \rightarrow S(x))$
 x is Korth course x should be enrolled by T_i. Sid

{ T. sid | $\exists T_i \in \text{Enroll} (T. sid = T_i. sid) \wedge$
 $\forall c \in \text{Enroll}$
 $(c. ginst = \text{KORTH} \rightarrow (T_i. cid = c. cid \wedge T_2. sid = T. sid))$
}

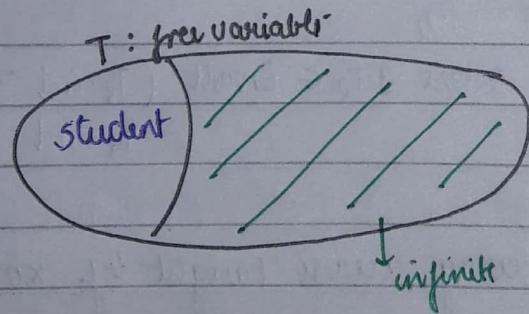
Unsafe TRC Query -

TRC query with infinite set of records in result.

Eg - { T | T ∈ student }

↑
set of tuple (T) which does not belongs to student relation.

(infinite records)



Expressive Power -

(say TRC queries)
 expressive power $\underset{\text{equal to}}{\sim}$ (basic relational algebra
 queries expressive power)

→ Basic R.A queries — R.A queries which can be expressed using { π , σ , \times , \cup , $-$, ρ , η , Σ , \bowtie , \bowtie' , $\bowtie\bowtie'$, $\bowtie\bowtie\bowtie'$ }

Queries failed to be express using basic R.A:

- Sum of attribute val
Avg. of Attribute val
Count of records / count of attr. value
- Ordering of resulted records of query.