Lecture - 14 (Entity Relationship Diagrams)

→ **High level DB design** :-
↳ { Diagrametic representation of DB design }.

Steps to design database

① Gearthering Requirements
⇩
② Conceptual / Logical data Base design.
(ER diagram) — Convert → RDBMS. Table.

} High level Data Base design.

③ Apply Normalizasation.
↳ To reduce Redundancy.
④ Physical database design (Indexing)
↳ To reduce Access cost
⑤ Application/security design.

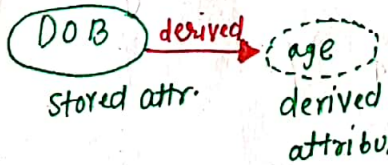
} Low level Database design.

Main component of ER diagram :-

① Attributes ⬭ ③ Relationship sets. ◇
② Entity sets ▭

Attribute ⬭ ✓ multivalued attribute ⬭ ✓
Key attribute ⬭ ✓ derived attribute. ⬭ ✓

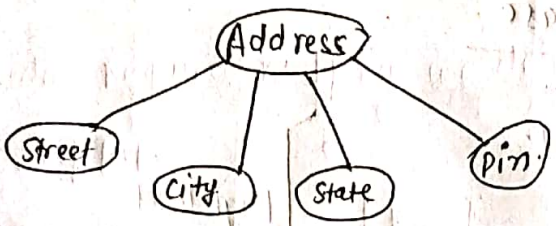
- Attribute Value which can be derived from other stored attribute values.
  ↳ Let's dob is stored attribute ← age of student from dob is derived attribute



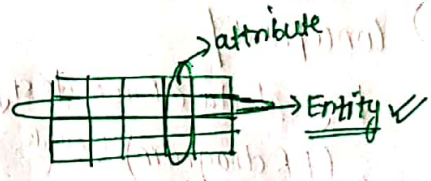
Composite attribute :-

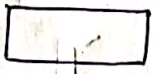
attribute which can represent as two or more attribute.

Ex. Address :→ (street name, city, state, pincode)



Entity set :- Set of similar Entity (tuples)

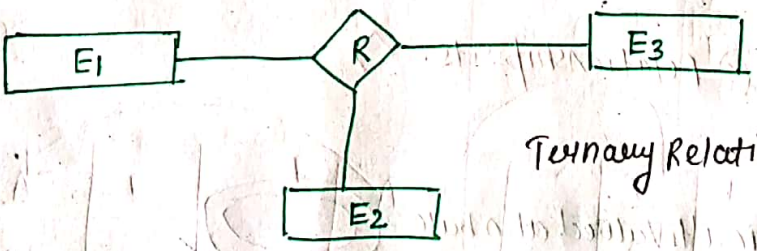


↳ Represented by 

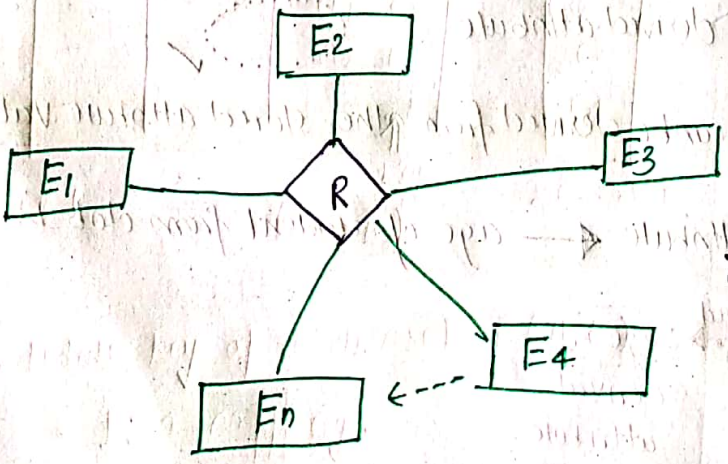
Relationship sets :-

Relate two or more Entity sets.

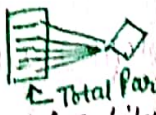
Binary Relation



Ternary Relation.



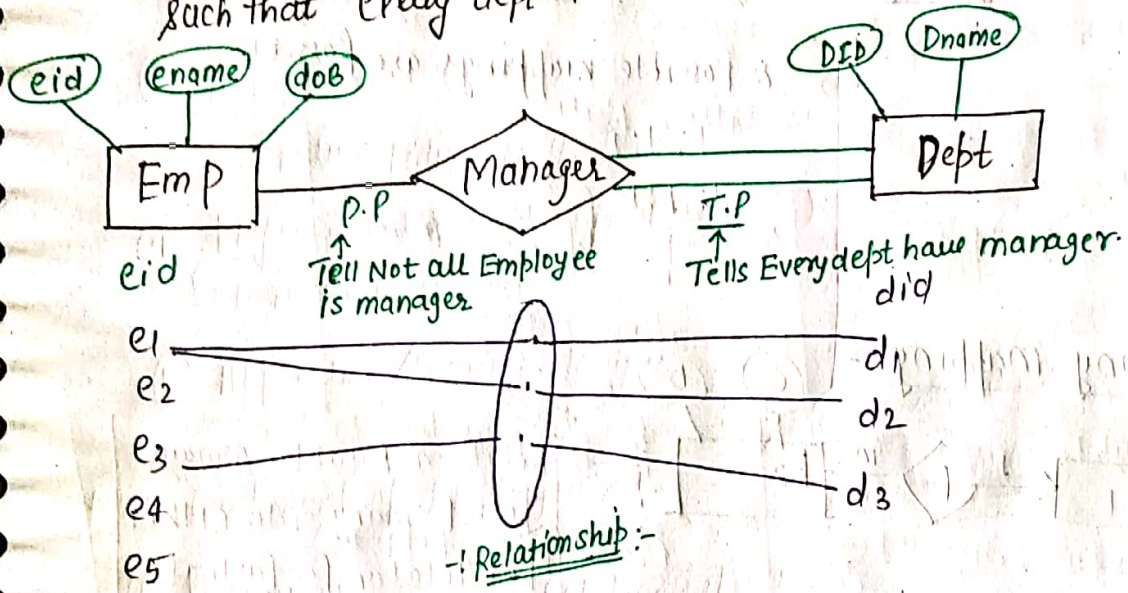
n-ary Relation

① Participation :- 

If every Record (Entity) of entity set must Related to Relationship set.
Such a Participation called Total Participation. (═══)
Otherwise Partial Participation. (——)

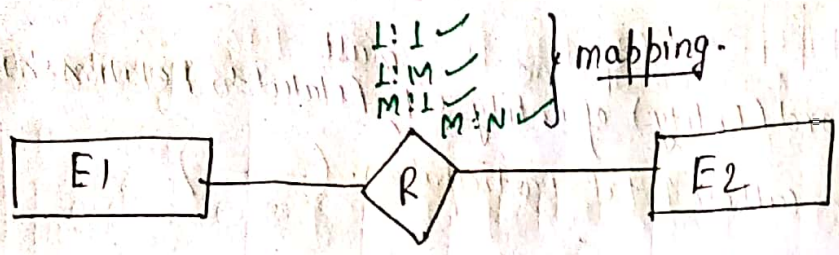
Total Participation :- 100% Participation.
→ may or maynot 100% Participation called Partial Participation.

Ex :- Emp & Dept are entity sets manages is relationship set such that every dept there must be manager.



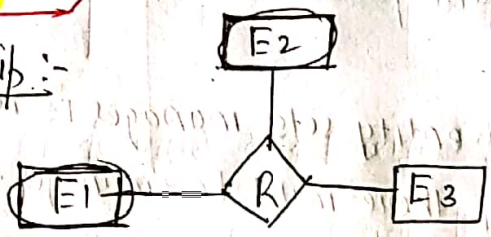
Cardinality :-
↳ one → [Relation will be at most 1]
↳ [0 or 1 Relation] (——)
↳ many
↳ [0 or more*] Relation
[0 ——]

Binary relationship Test :-
① One : One mapping
② One : many mapping.
③ many : one mapping
④ many : many mapping.



↳ Based on Required mapping we design candidate key of Relationship Test.

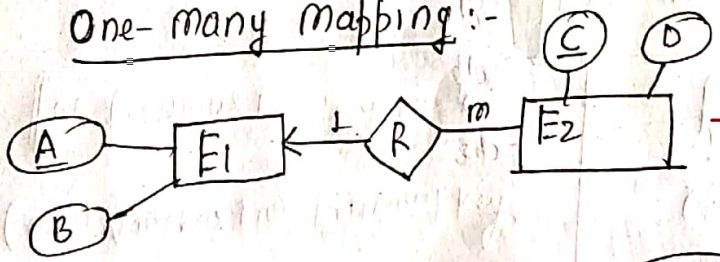
Ternary Relationship :-



↳ 8 Possible mappings are Possible.

1 : 1 : 1
1 : 1 : M
⋮

One- many mapping :-



*** Relation & Table Always Join to many side either For Total or for Partial Participation.

For Partial Partition we Put Null at the Place of Non-Participated element

E1

A	B
a1	—
a2	—
a3	—
a4	—

{A → B}

R

A	C
a1	c1
a1	c2
a2	c3

unique

Key

(C → A)

E2

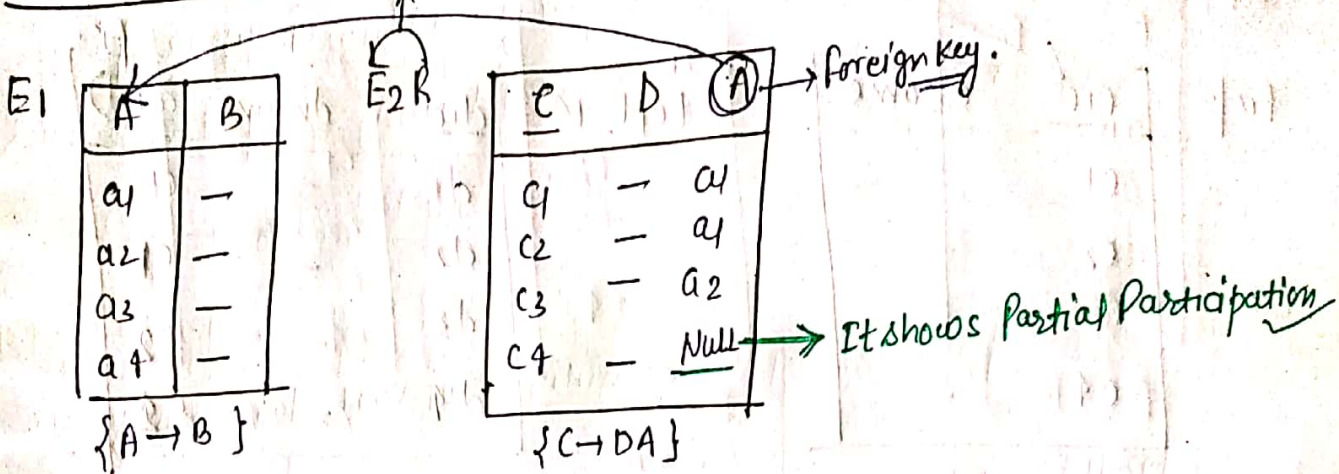
C	D
c1	—
c2	—
c3	—
c4	—

{C → D}

• Candidate key of relationset R(A,C) is {C} for —
1:M mapping.

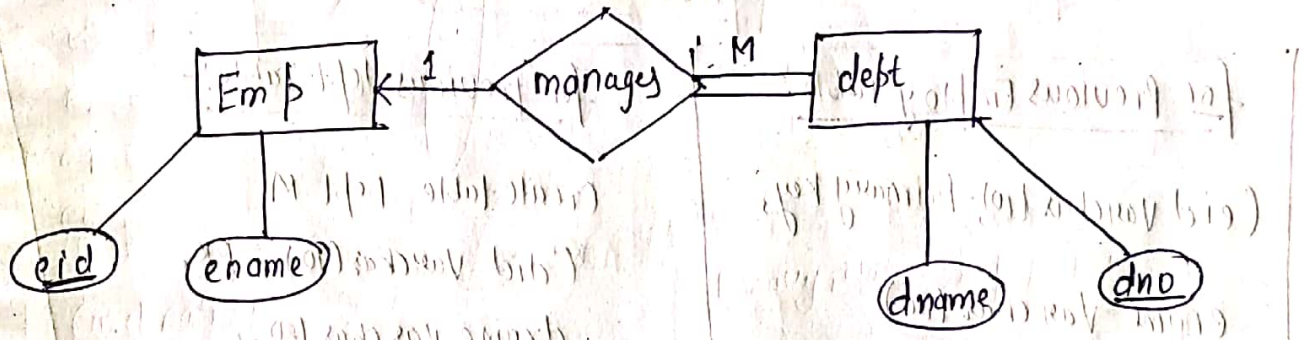
RDBMS design

Rem merge with E_2 due to candidate key of both are same



So 1:m Need → 2 RDBMS Tables && 1 foreign key

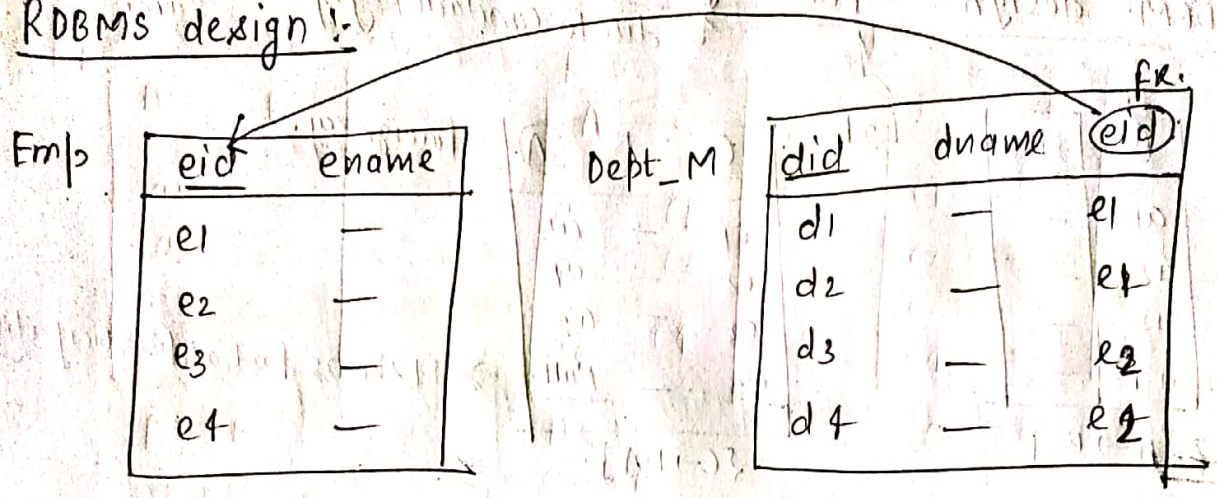
Ques :- Emp && Dept Entity set manages relation set 1:M.
each emp can manage many dept, each dept must manages by one emp.



Emp (eid ename) M (eid did) Dept (dno dname)

e1		e1	d1	d1	—
e2		e1	d2	d2	—
e3		e2	d3	d3	—
e4	—	e2	d4	d4	—

RDBMS design :-



Min 2 RDBMS Tables & 1 Foreign Key Needed

*** If Partial Participation at right side then foreign key which we Added right side Entity set should allow Null.

*** If Total Participation of 1:M Mapping NOT allow Null.

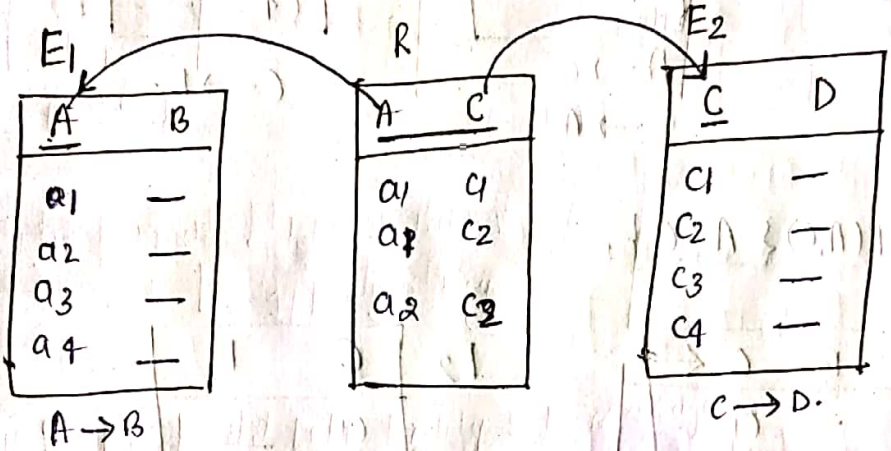
for Previous Employ Table:-

```
(eid varchar(10), Primary key,
 ename Var char(20)
);
```

for Previous dept Table:-

```
Create table Dept M
(did Varchar(10) PK,
 dname varchar(30,
eid Var char (10) NOT NULL,
FK (eid) references emp)
```

Many to many mapping :-



Candidate key for rel set R(A, C) {AC} for M:N

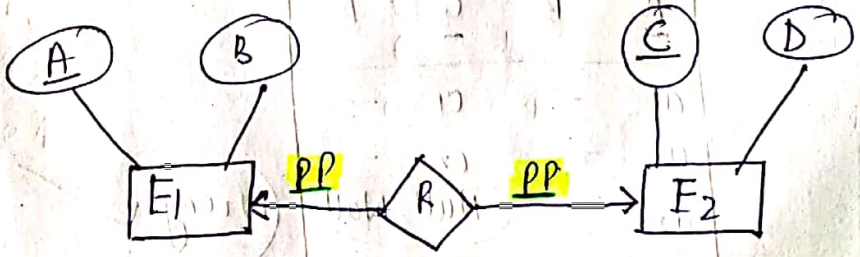
RDBMS design :-

E_1(A B)
E_2(C D)
R (AC)

min 3 rel Tables required

:- One to one mapping :-

(a) 1:1 mapping b/w E_1, E_2 with Partial Participation at both ends.



P.T.O

E1

A	B
a1	—
a2	—
a3	—
a4	—

$A \rightarrow B$

R

A	C
a1	C1
a2	C3
a4	C4

$A \rightarrow C$
$C \rightarrow A$

E2

C	D
C1	—
C2	—
C3	—
C4	—

$C \rightarrow D$.

Cand Keys of R(A,C) = {A, C}.

E1 R

A	B	C
a1	—	C1
a2	—	C3
a3	—	Null
a4	—	C4

P.keys — A

A.Key — C

E2

C	D
C1	—
C2	—
C3	—
C4	—

$A \rightarrow BC$
$C \rightarrow A$
$C \rightarrow D$

& Candade Keys = {A, C}.
A.Keys
P Keys

***

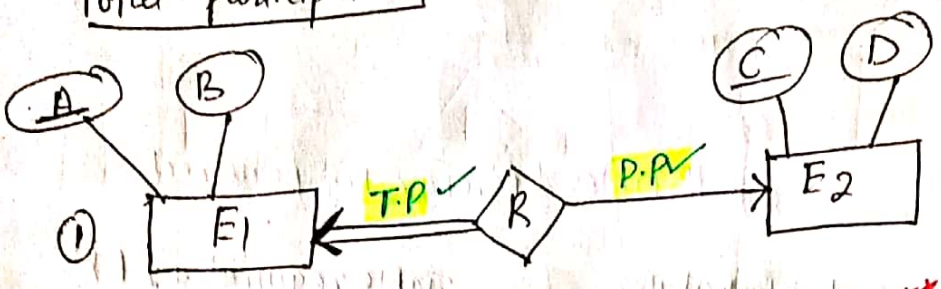
But we cannot merge all R E1, R & E2 Tables into only one Table. Ex.

E1 R E2

A	B	C	D
a1	—	C1	—
a2	—	C3	—
a3	—	Null	Null
a4	—	C4	—
Null	Null	C2	—

Wrong

(b) 1:1 mapping between E_1 & E_2 with at least one end Total participation :-



①

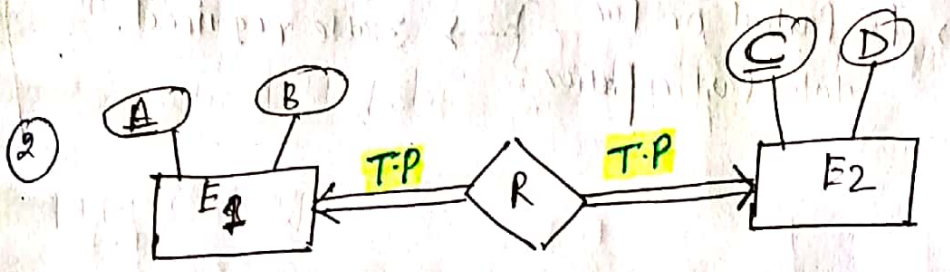
E_1 R E_2

A → BC
C → AD
{A, C}

A	B	C	D

A/C — PK

*** Min. 1 RDBMS Table & 0 foreign key



②

E_1 R E_2

A → BC
C → AD
{A, C}

A	B	C	D

*** Min 1 RDBMS Table & 0 Foreign Key

A — Primary Key ✓
C — UNIQUE Not Null ✓

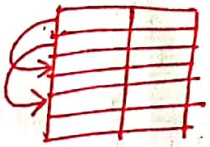
:- Learning from E-R diagram :-

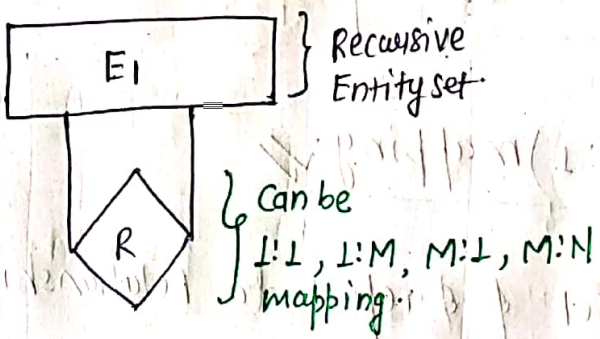
Mapping Type	# Tables & foreign key Required
1 : M M : 1 → for Partial Participation or for Total Participation	→ 2 Table required + 1 FK's
1 : 1 → Partial Participation Both side	→ 2 Table + 1 FK's req
1 : 1 → One side Total Participation (or) both side Total Participation	→ 1 Table + 0 FK's req
M : N → for Partial or for total Participation	→ 3 Table required.

:- Lecture-15 :-

Self Referencial Relationship set :- (Recursive Entity set)

Entities of entity set (E) related to some other entities of same ~~other~~ entity set.

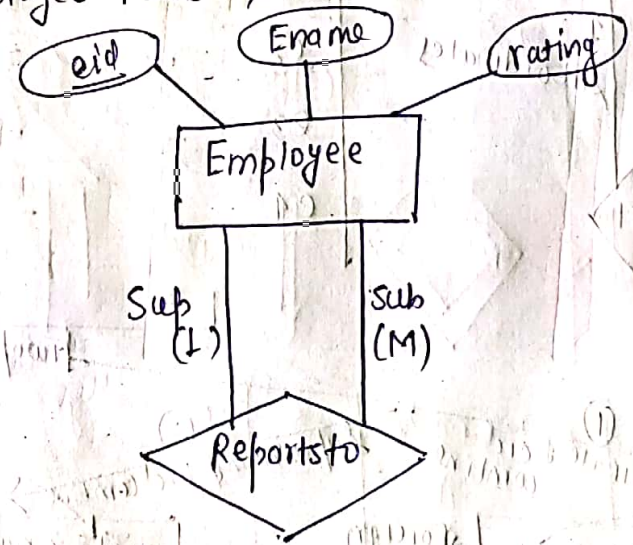




***

Ques Employee Entity set & Reports to Relationship sets related b/w Supervisor && Subordinates. Each supervisor can supervise many sub-ordinates. Each subordinate reports to one supervisor.

Sol^M Here Employee supervisor && subordinates both are Employee so both create Recursive entity set.



Key

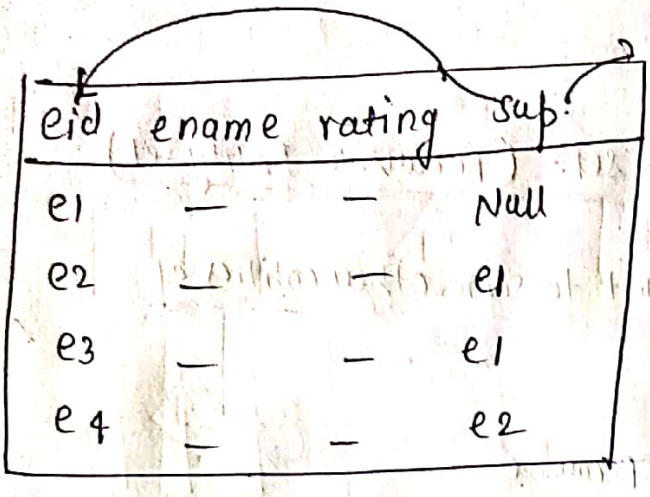
Emp

eid	ename	rating
e1	—	—
e2	—	—
e3	—	—
e4	—	—

Key

Rep

Sup	Sub
e1	e2
e1	e3
e2	e4



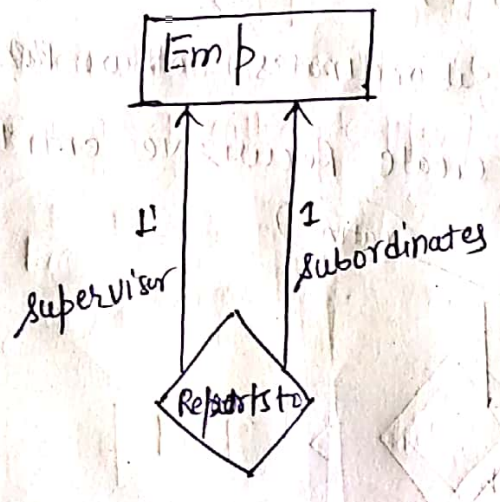
Only one Table required ✓

Self Refeuencial (1:1) mapping:

Ex Emp entity set & & reports to Relationship set related b/co supervisor & subordinates.

Each supervisor can supervise one subordinates & Each subordinate reports to one supervisor.

Sol"



Emp

eid	ename	rating
e1	—	—
e2	—	—
e3	—	—
e4	—	—

Rep

sup	sub
e1	e2
e2	e3
e3	e4

Two possibility

① subordinate & eid combine

② supervisor & Eid combine

eid	ename	rating	sup
e1	—	—	Null
e2	—	—	e1
e3	—	—	e3
e4	—	—	e4

foreign key

eid	ename	rating	subord
e1	—	—	e2
e2	—	—	e3
e3	—	—	e4
e4	—	—	Null

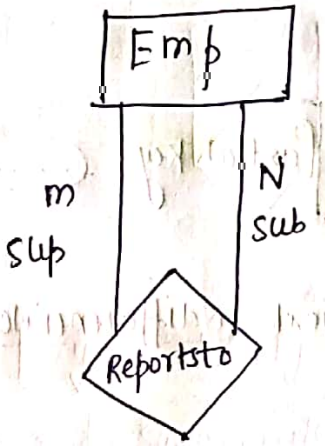
foreign key

Self Referencial (m:n)

Que Emp entity set ₹₹ reports to Relationship set related b/w supervisor & subordinates.

Each supervisor can supervise many sub-ordinates & each subordinates reportsto many supervisor.

Soln:-



Emp

eid	ename	rating
e1	—	—
e2	—	—
e3	—	—
e4	—	—

Rep

Sup	Sub
e1	e2
e1	e3
e4	e3

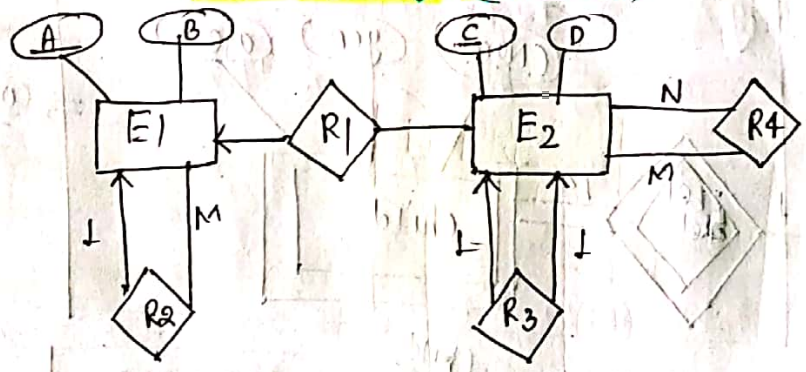
Cannot minimize ✓

minimum 2 Relational sets required ✓

&

2 foreign key (Sup & Sub) ✓

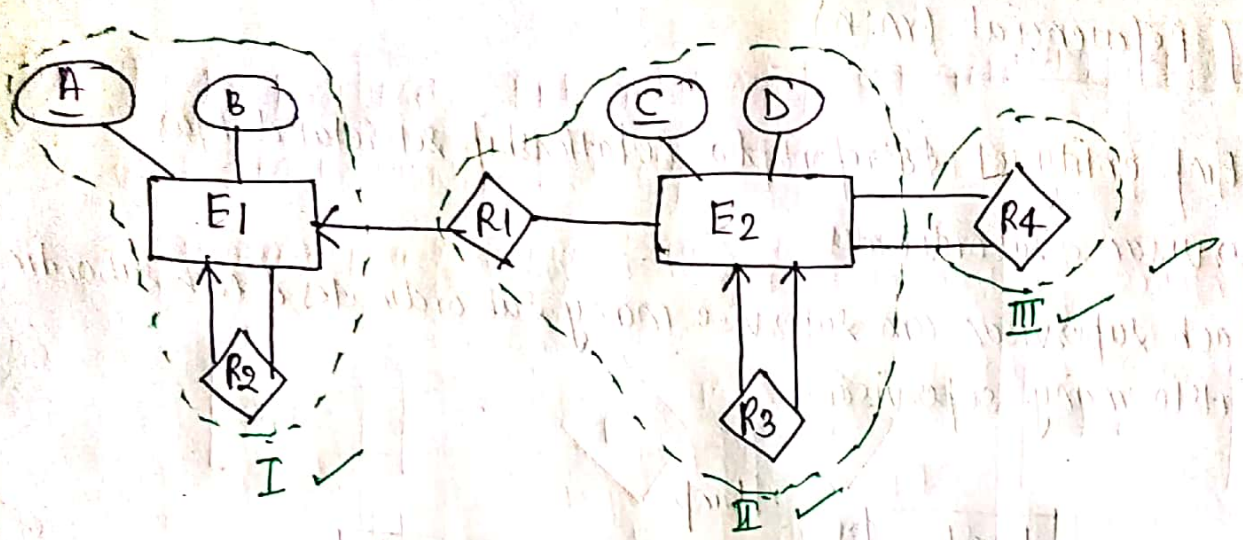
Que



How many minimum relational Tables required.??

Soln ③ Relational Tables required

P.T.O



WEAK ENTITY SET :-

→ Entity set with No Key.

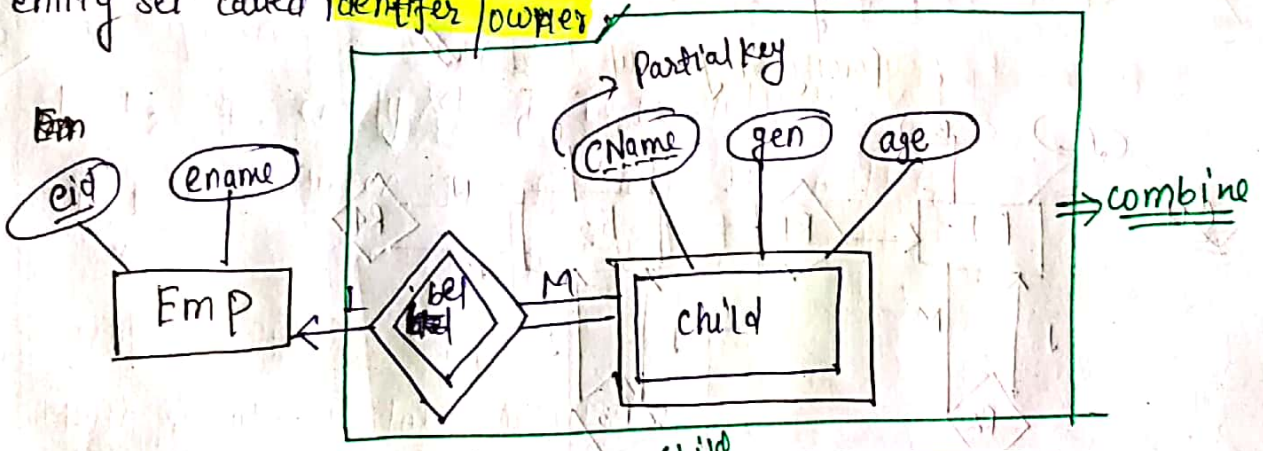
Partial Key

→ Attributes of weak entity set Not sufficient to differenciate Entities or record uniquely.

* Represented by → weak Entity

→ Weak entities are depending entities, which depends on other strong entity set called identifer / owner.

Ex:-



Emp

eid
e1
e2
e3
e4

eid	ename
e1	A
e2	A
e2	B
e3	B

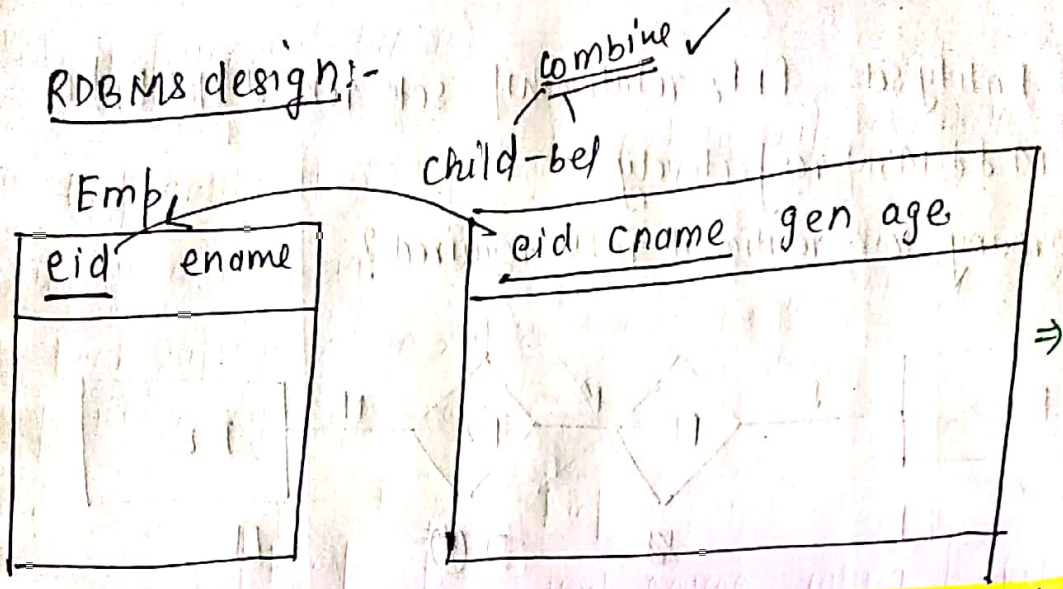
weak

Child

Cname	gen	age
A	male	10
A	male	10
B	fem	15
B	fem	12

depending

RDBMS design:-



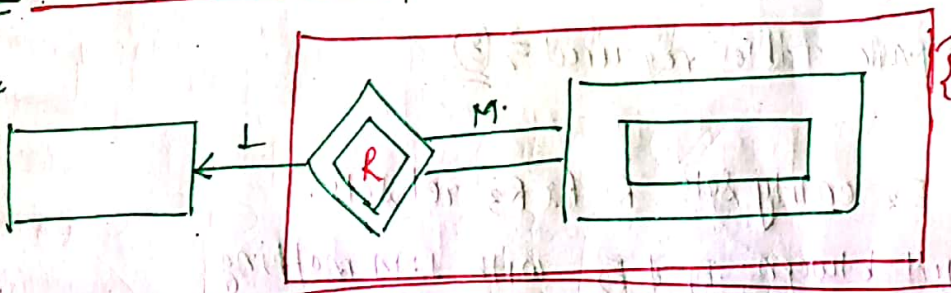
⇒ Min 2 Relation Table Required

⇒ Participation at weak entity set end must be TOTAL

⇒ Mapping b/w strong entity set & weak entity set must be 1:M.

⇒ weak entities uniquely identifies by combination of Partial Key & & Identifier Key

** when weak relation & weak Entity is given. ✓



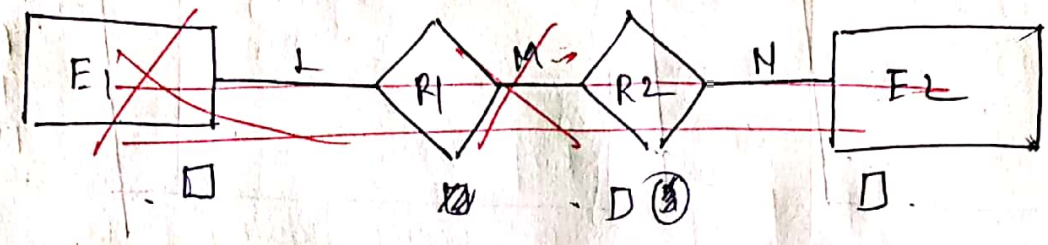
. Multivalued attribute & weak entity set allowed to represent in E-R diagram but Not allowed to represent in RDBMS.

gate Qu

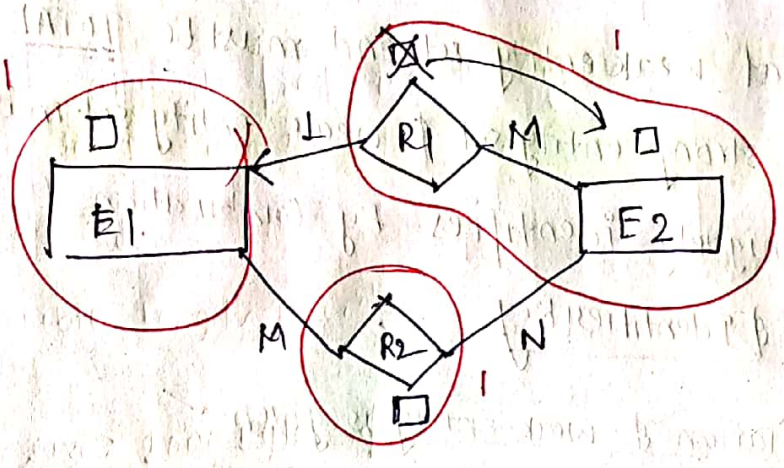
E_1 E_2 Entity sets R_1 R_2 relationship sets related b/w E_1 & E_2 with 1:M && M:N respectively

How many min relational Tables required?

Soln



3 Tables



Min Tables required ⇒ (3)

gate Qu

E_1 E_2 E_3 entity sets R_1 R_2 R_3 rel sets.

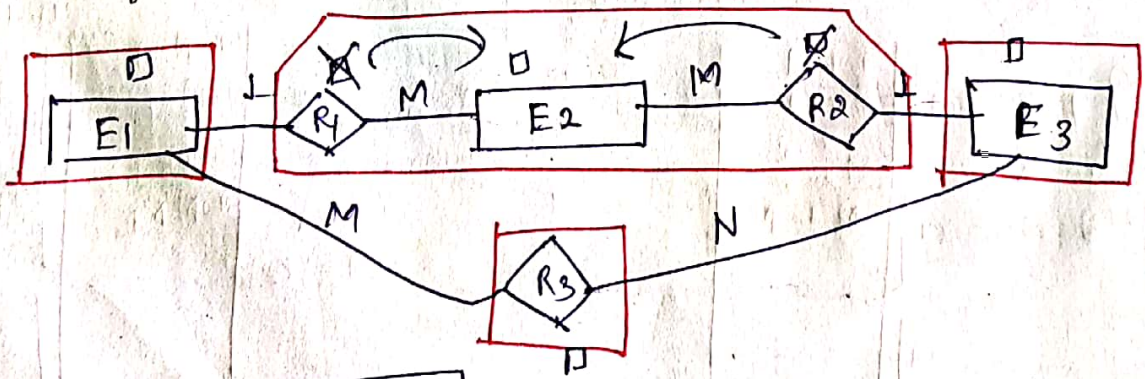
R_1 Related btween E_1 & E_2 with 1:M mapping

R_2 " " E_2 & E_3 " M:1 " "

R_3 " " E_1 & E_3 " M:N " "

Min relational Table required.

Soln



⇒ 4 Tables required

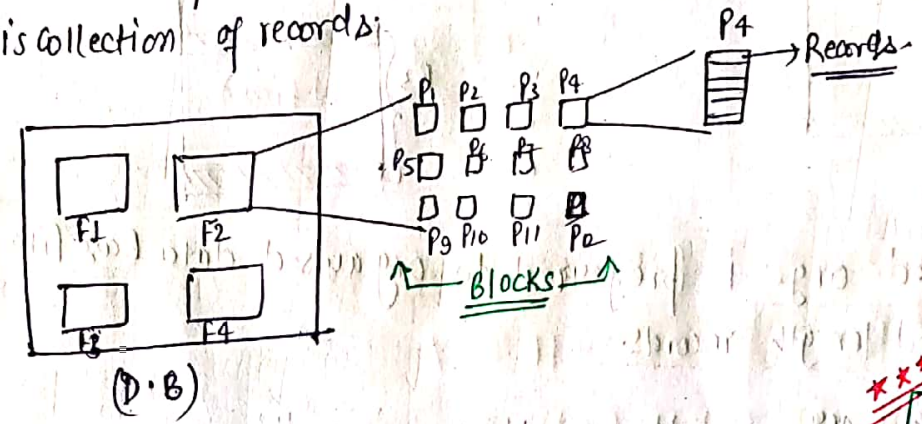
:- Lecture-16 :-

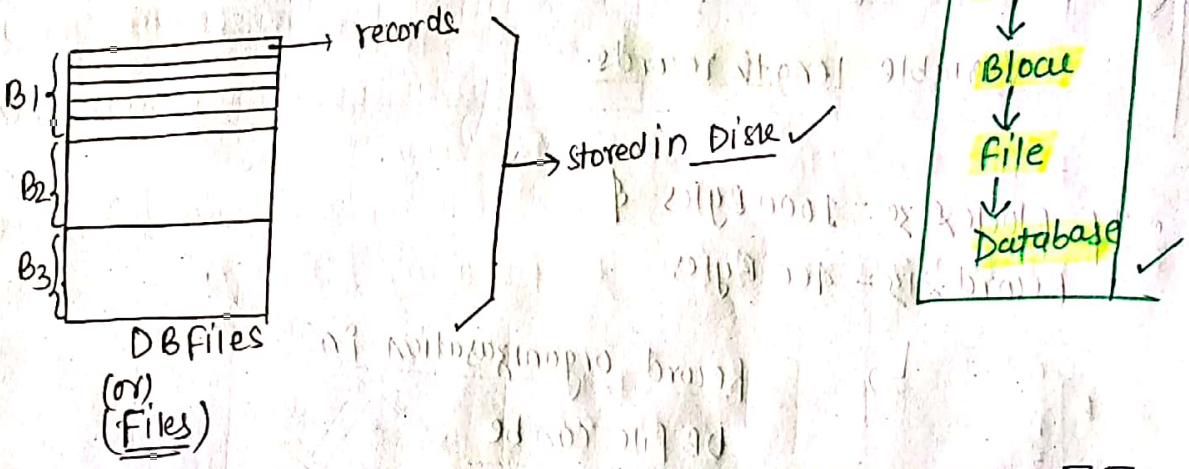
File organizasation & Indexing :-

→ Database is the collection of File (Tables)

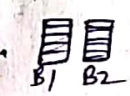
→ File is collection of Blocks (Pages)

→ Block is collection of records:





- Data is Access from disk to m·m is Block By Block.



Records in D·B File

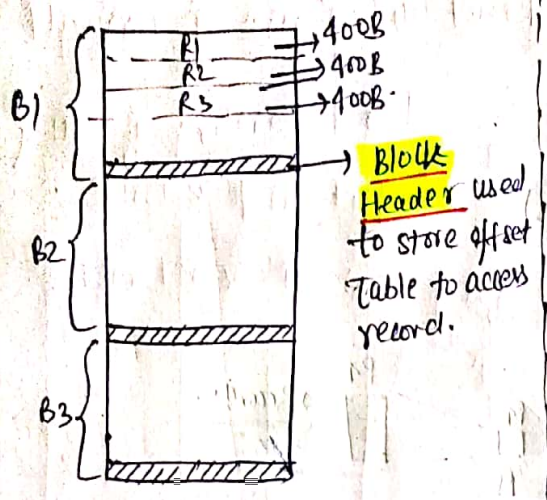
Fixed length Record

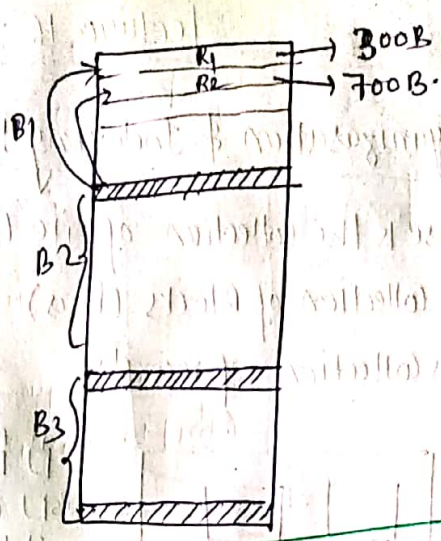
```
Create table (R)
(
  A char(100);
  B char(100);
  C char(200);
)
```

Variable length Records

```
Create table S
(
  D char(200);
  E char(50);
  Ftext;
)
```

→ text is a variable length type record in database.





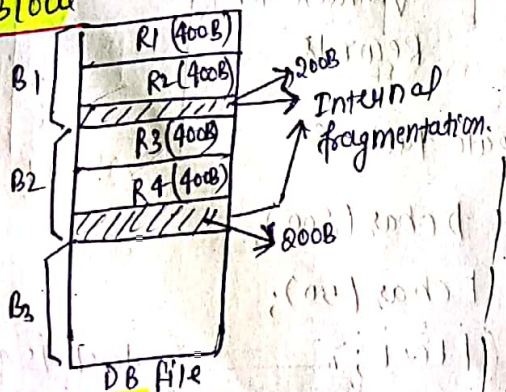
① Unspanned org. is preferred to organised data Base file with fixed length records.

② Spanned org: (फैला हुआ) Variable length records.

- If Block size = 1000 Bytes &
  Record size = 400 Bytes

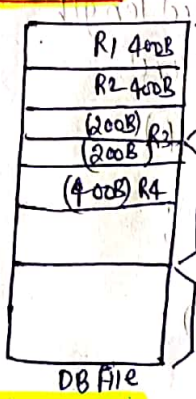
↳ Record organization in DB File can be

- Complete record allowed to be stored in one Block



*** Less access cost
- Preferred for fixed length records DB files.

- record allowed to span more than one Block.



If possible at Last Block only

*** More Access cost.
- Preferred For Variable length record DB File.

Blocle factor of DB File (Apply for Unspanned org.)

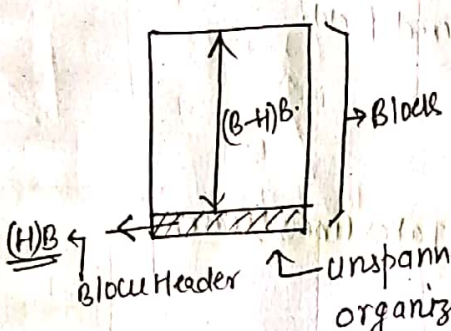
- It tells maximum possible record/Blocks.
- if Blocle size = B Bytes & Record size = R Bytes (fixed length)
- if Blocle Header size: H Bytes

thus

$$\text{Blocle factor of D.B file} = \left\lfloor \frac{B-H}{R} \right\rfloor \text{ records/Blocle}$$

Floor Value

& Remaining size go to Internal frag-mentation

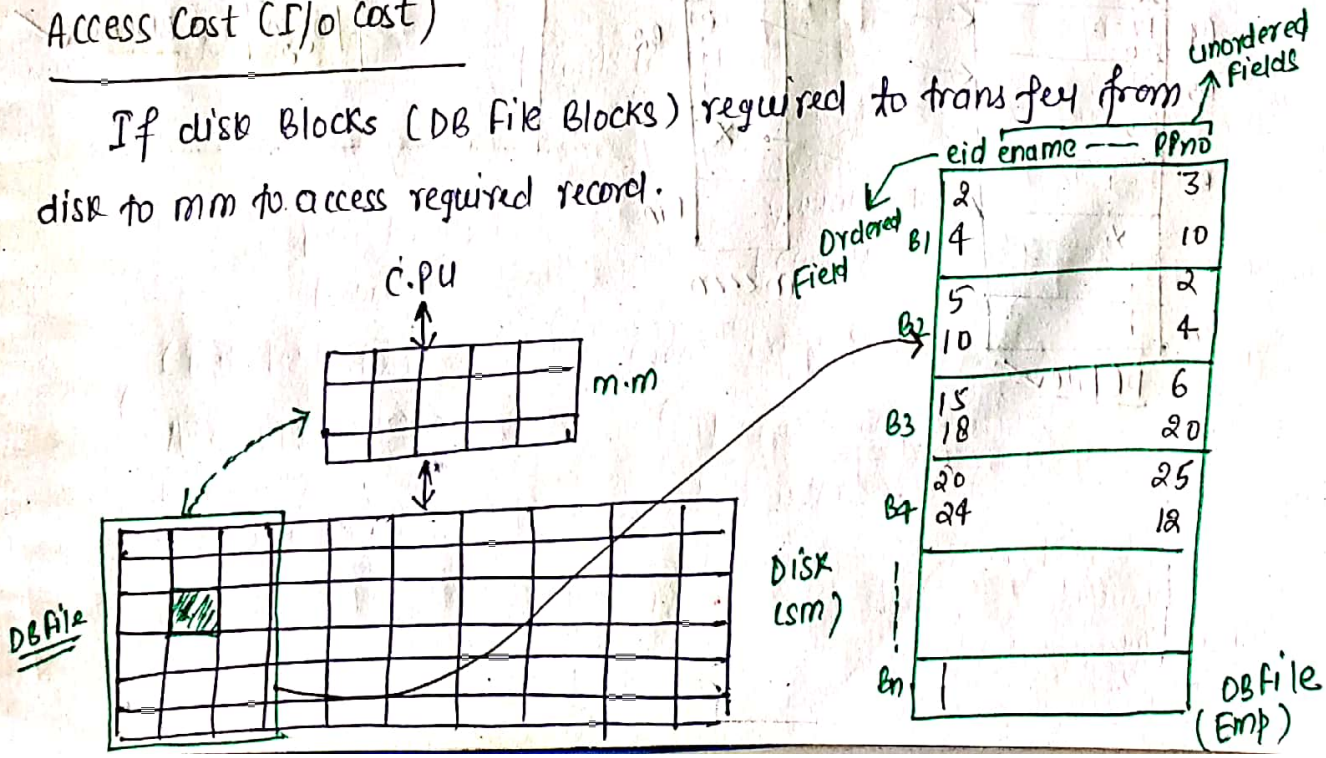


Qu: Blocle ⇒ 1000 B.
Record ⇒ 400 B (fixed length)
Find Record/Blocle

Sol: Blocle factor $= \left\lfloor \frac{B-H}{R} \right\rfloor = \left\lfloor \frac{1000-0}{400} \right\rfloor = 2$ records/Blocle

Access Cost (I/O cost)

If disk Blocks (DB File Blocks) required to transfer from disk to mm to access required record.



- I/O cost to access record from data-Base file

a) using ordered field

ordered field

- Select * from Emp where (eid) = x ;

- Worst case I/O cost $\lceil \log_2 n \rceil$ Blocks

Binary search (for ordered field)

b) using unordered field -

unordered field

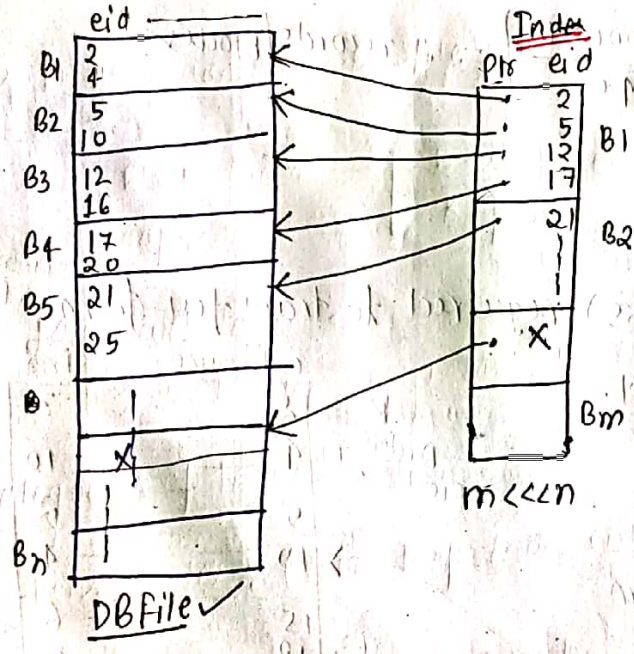
- select * from Emp where (PPNO) = y

- Worst case I/O cost n Blocks

Unorder so, it Takes linear search

To Reduce this I/O cost we use Indexing.

I/O cost to access record from DB file using Indexing :-

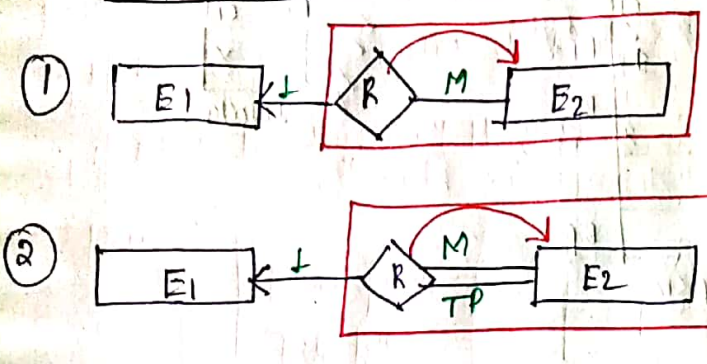


- I/O cost using 1st level index

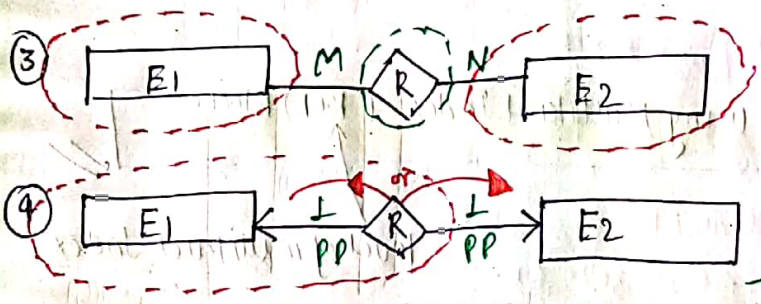
$= \lceil \log_2 m \rceil + 1$ Blocks

Revesion (revising) from E-R diagram:-

Assume No multivalued attribute

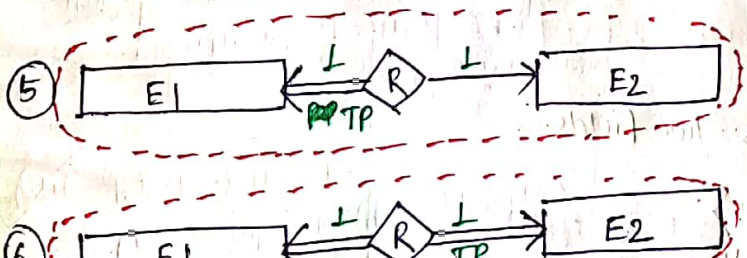


Min 2 RDBMS Tables && 1 FK

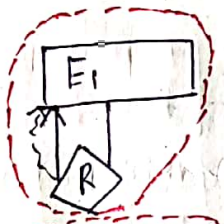


min 3 RDBMS tables && 2 FKs

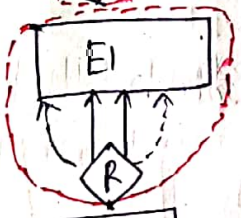
min 2 RDBMS tables && 1 FK's



min 1 RDBMS table && NO FK ← foreign key

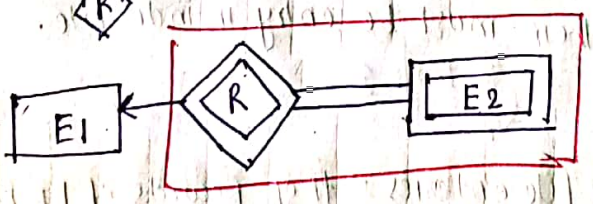


min 1 RDBMS table & 1 FK



min 1 RDBMS table & 1 FK

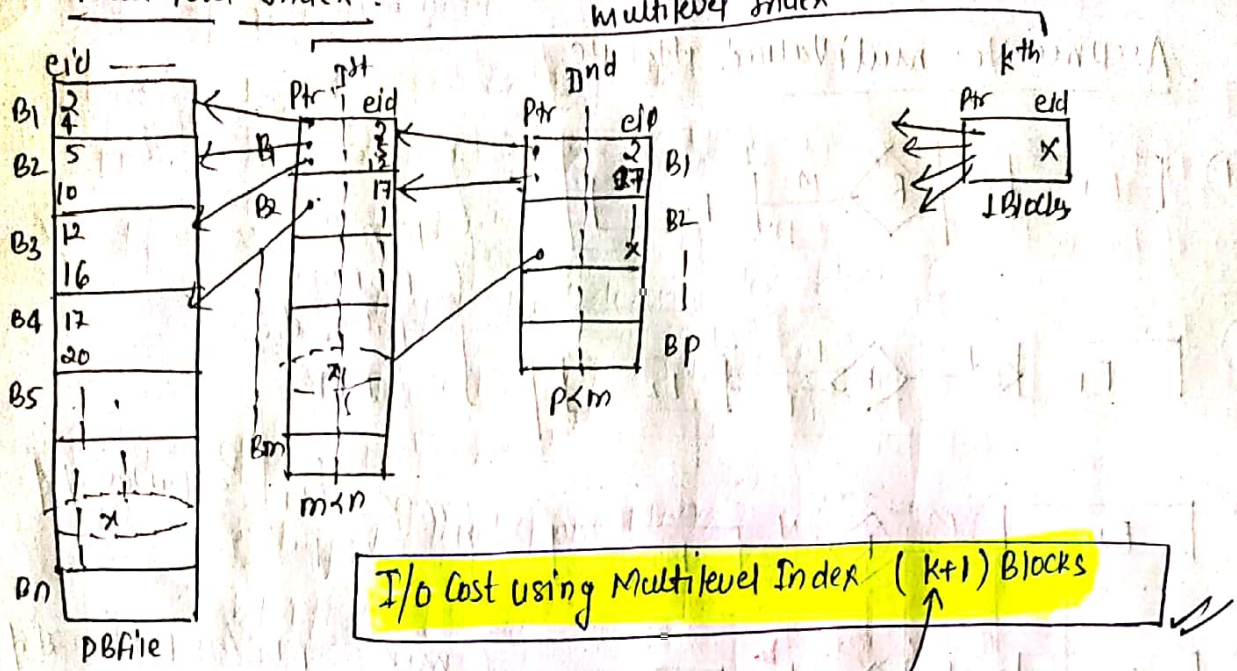
min 2 RDBMS Tables && 2 FK's



min 2 RDBMS Tables Tables & 1 FK.

gave ✓✓

Multilevel Index :-



I/O Cost using Multilevel Index (K+1) Blocks

level of multilevel Index.

Index File :-

⇒ Each Entry of Indexed Files two fields.

⟨ search key , Pointer ⟩
↳ K Bytes. ↳ P Bytes.

⇒ Block factor of Index file max Possible Index enteries per Block.

***

$$\text{Block factor of Index} = \left\lfloor \frac{B-H}{K+P} \right\rfloor \text{ enteries/Block}$$

Multilevel Index :-

Index to Index until last level single Block of Index.

Categories of Index :-

① Dense Index : (More # of Index enteries)

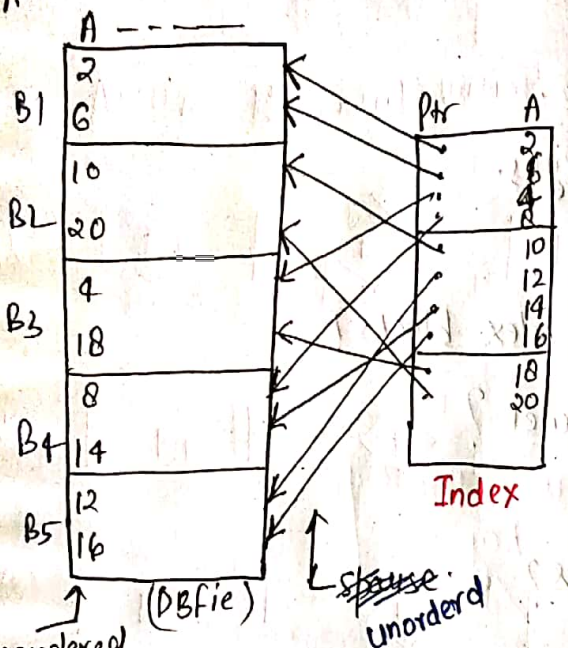
for each record of db File their must be entry in index file.

for dense Index

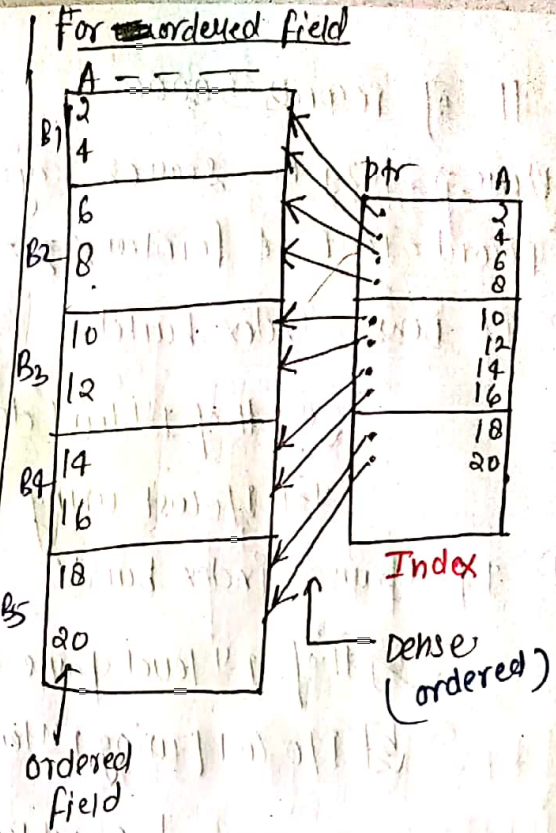
**

# of Index file enteries = # of Records of DB file.

for unordered field

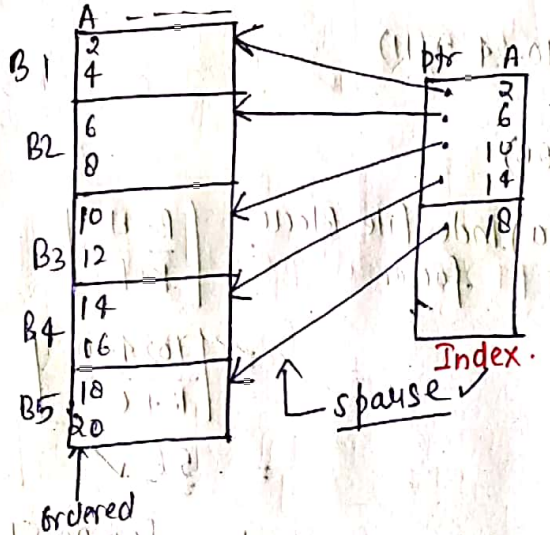


For ordered field



Spause Index :- (less No. of Index enteries)

for set of DB records their exist entry in Index file.



⇒ if sparse index over key & ordered field ✓

***

# of Index file enteries = # of Blocks of DB file

Que:- # of records 50,000

Block ⇒ 1024 B Search key ⇒ 12 B.

Record = 100 B & Pointer size = 6 Bytes.

(I) If Dense Index build

a] # of 1st level dense index blocks?

b] I/O cost using 1st level ?

(II) If sparse Index build

a] # of 1st level sparse Blocks?

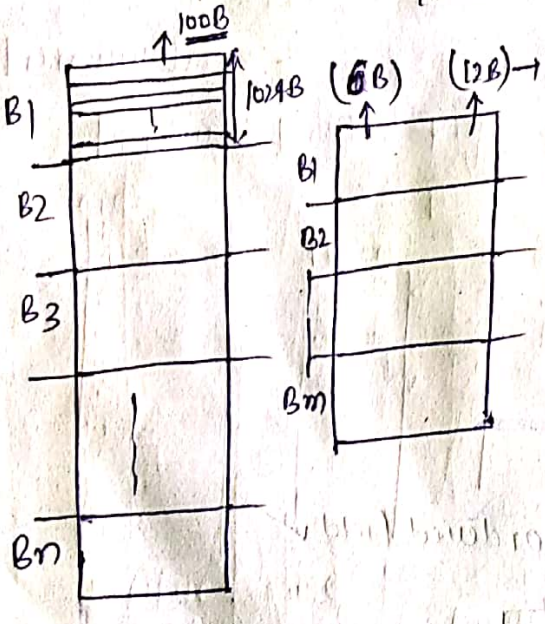
b] I/O Cost [using 1st level].

Soln # records = 50,000
Block size = 1024 B

Search key size = 12 B.
Pointer size = 6 B.
Record size = 100 B.

(I) for dense Index -

Record / Block ⇒ $\frac{1024\ B}{100\ B}$ ⇒ 10.24 ⇒ (11)



(a) Index file Block factor = $\left\lfloor \frac{B-H}{K+P} \right\rfloor$

= $\left\lfloor \frac{1024-0}{12+6} \right\rfloor$

= 56 ✓

Blocks Required for 50,000 records = $\left\lceil \frac{50,000}{56} \right\rceil$

= 893 ✓ Blocks ✓

(b) I/O Cost = $\lceil \log_2(893) \rceil + 1$

= $\lceil 9.80 \rceil + 1$ ⇒ 10 + 1 ⇒ 11 ✓ blocks.

(11)(a) for sparse Index ✓

#Index enteries = #Blocks of DB

Blocke factor of data Base = $\left\lfloor \frac{B-H}{\text{Record size}} \right\rfloor$

$= \left\lfloor \frac{1024-0}{100} \right\rfloor = 10$

$10 * x = 50,000$

$x = 5,000$

↑ #Blocks ✓

$\left(\left\lceil \frac{1024}{18} \right\rceil \right) * p = 5,000$

↳ Blocke factor of Index

$p = \left\lceil \frac{5,000}{56} \right\rceil \Rightarrow 89.2 \Rightarrow 90$ Blocks ✓

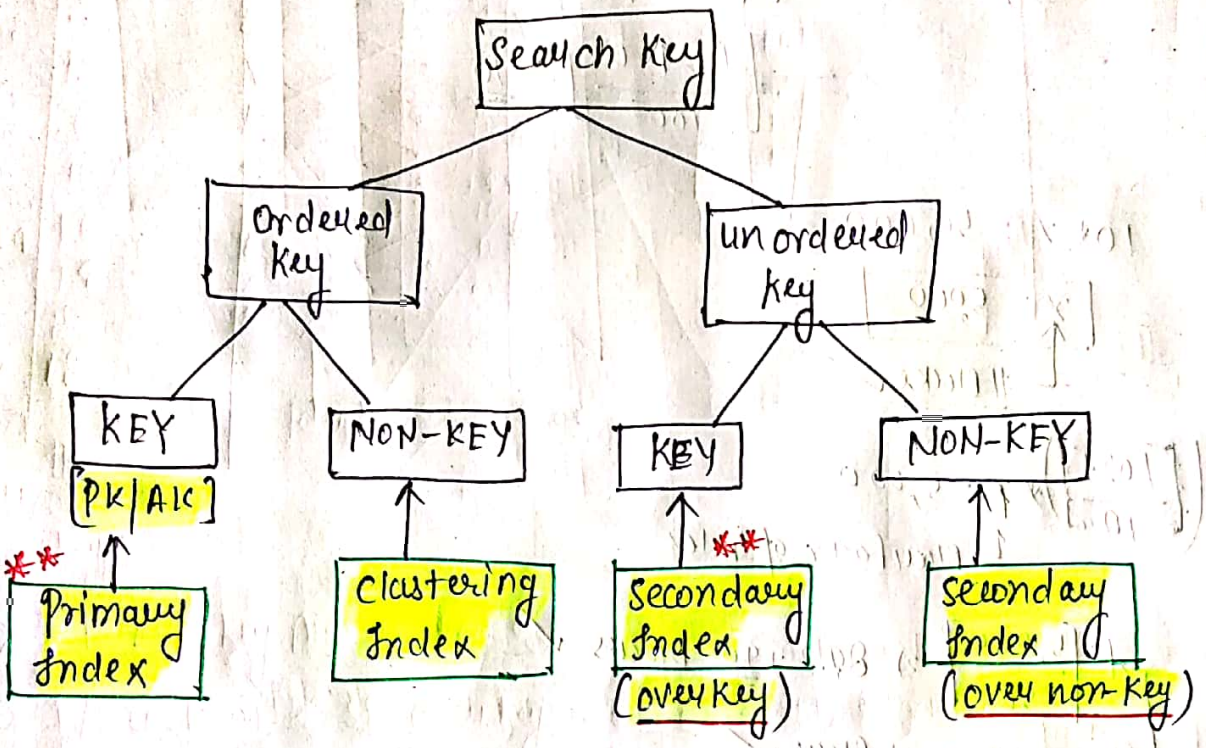
↑ Blocke factor of Indexed Blocke ✓

(b) I/O cost $\Rightarrow \lceil \log_2 90 \rceil + 1$

$= 8$ Blocks ✓

Lecture - 17 :-

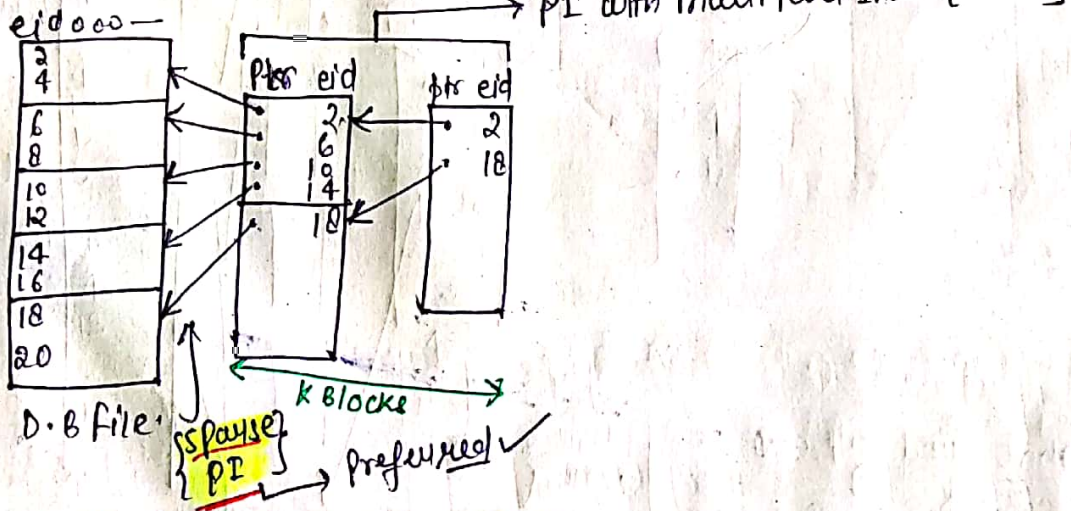
Types of Indexes



Search key: fields used for Indexing ✓

Primary Index :-

[Ordered field & key]



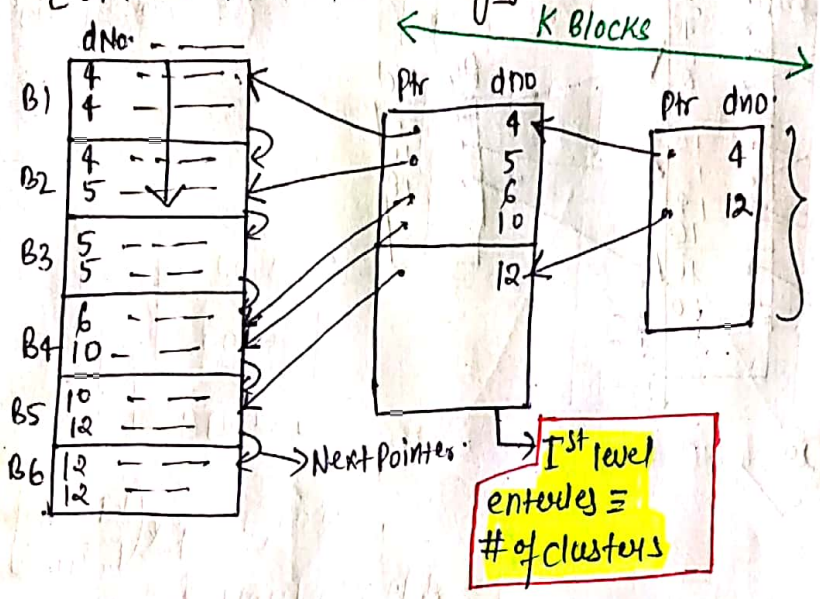
⇒ I/O Cost } (K+1) Blocks. ✓
PI with MLI }

⇒ PI can be dense / sparse
(Sparse primary Indexes prefered)

⇒ Atmost one PI for ~~one~~ Any DB file.

Clustering Index :-

[ordered field & Non key]



⇒ I/O Cost

Cluster Index with Multilevel Index ⇒ { K+ one/more DB Blocks } Blocks

⇒ Cluster Index (~~CI~~) (CI) mostly sparse Index

[Each cluster with one record CI is dense]

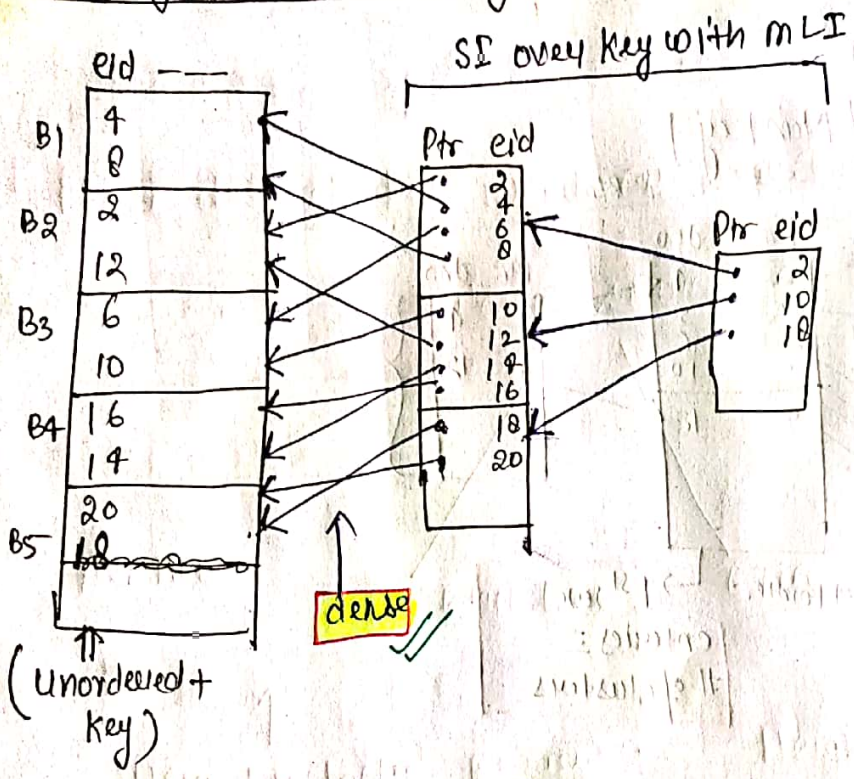
⇒ Atmost one CI for any DB File (Bcoz ordered field)

- for any data base relation Either 1 Primary Index or atmost 1 clustering Index are Possible but Not Both ✓

Secondary Index :- (Indexes for unordered field & (Key / Non key))

⇒ many secondary Indexes are possible for data base File.

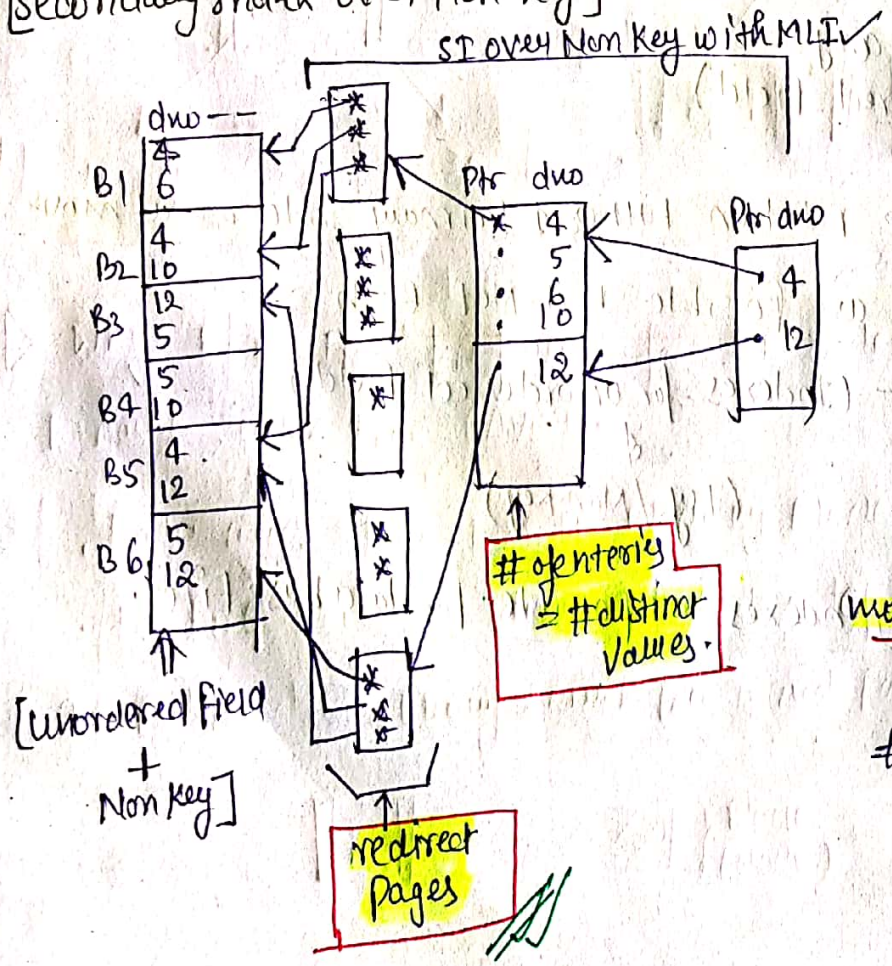
[Secondary Index over key]



⇒ SI over key Always Dense

⇒ I/O Cost with SI over key ∝ MLI (K+1) Blocks.

[Secondary Index over Non Key]



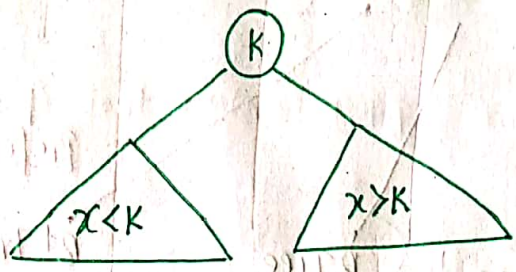
⇒ SI over Non key mostly Sparse Index.

# of Index entries < # of records

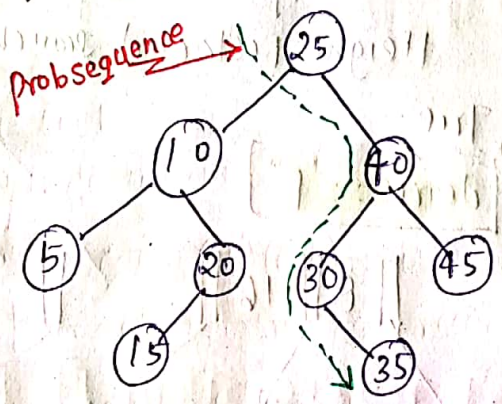
:- Dynamic multilevel Index :-

** Btree, B+tree } Balanced Search tree

Searchtree :- DS organizes Keys



Binary search Tree:-



For Search Key :- 35

Prob Sequence :-

25, 40, 30, 35

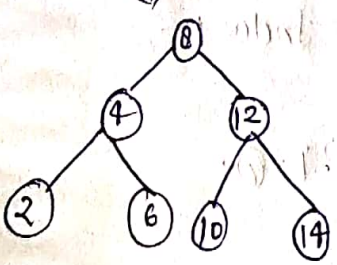
[Prob sequence is One Node access from Each level]

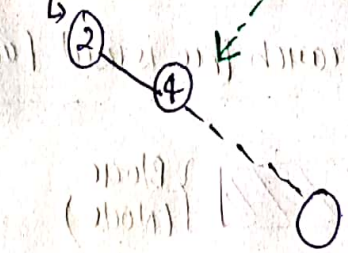
Height of Search Tree

For n distinct keys

Min Height $O(\log_2 n)$

Max Height $O(n)$





In simple Binary Tree

Dis Adv: Worst Case Search Cost to perform search of key $O(n)$

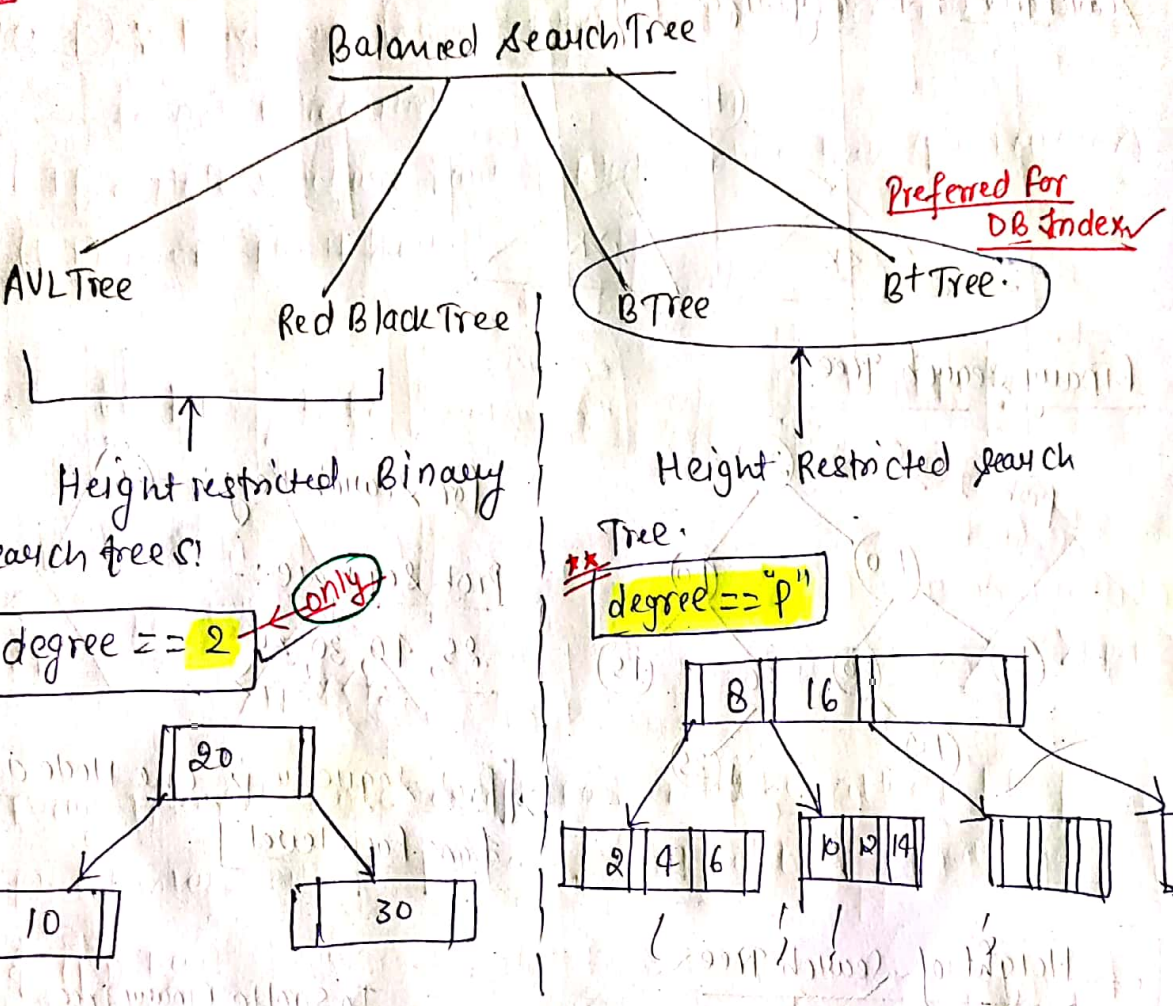
Ex: Skewed Tree

- left skewed ✓
- Right skewed ✓

Balanced Search Tree :- (Height Restricted Search Tree)

For B & B+ tree : Max Height of search Tree should not extend $O(\log n)$

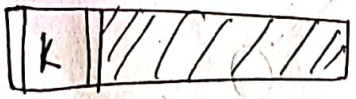
*** For n distinct keys worst case search cost $O(\log n)$



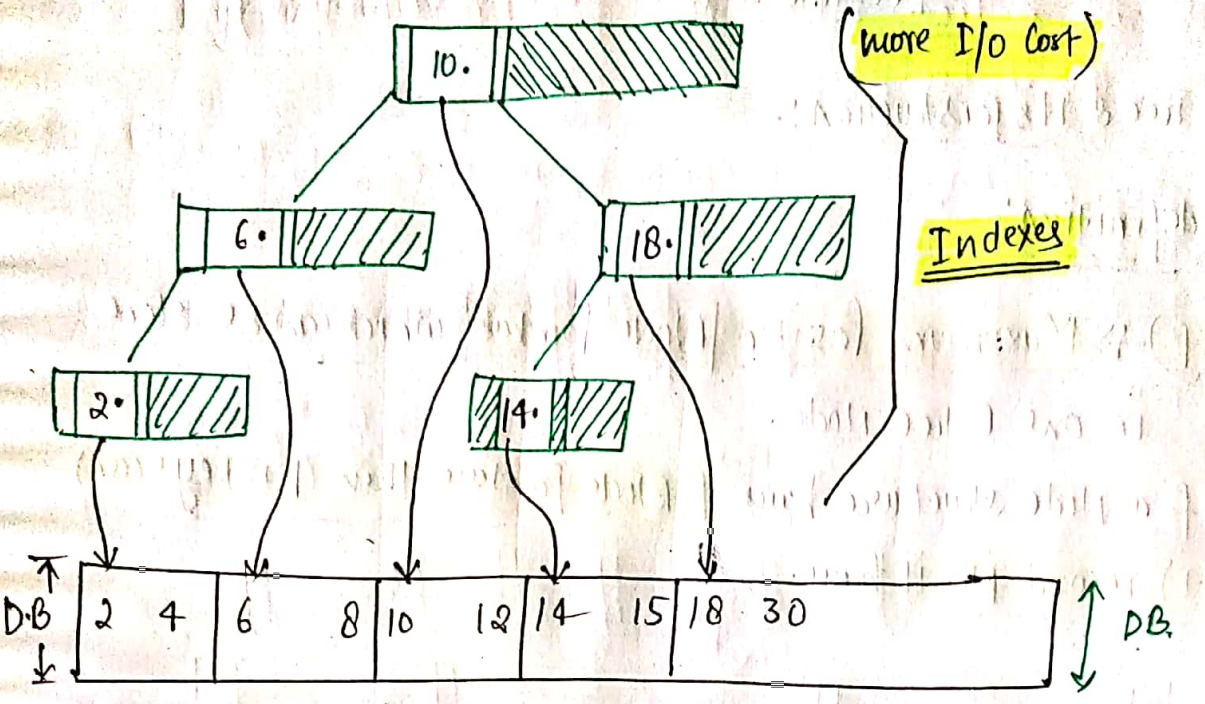
⇒ Why B/B+ Tree preferred for DB Index rather than Balanced Binary Search Tree (AVL Tree) ?

- Data access from Disk is Block By Block.
- DB file stored in disk && index of DB file must be in disk.

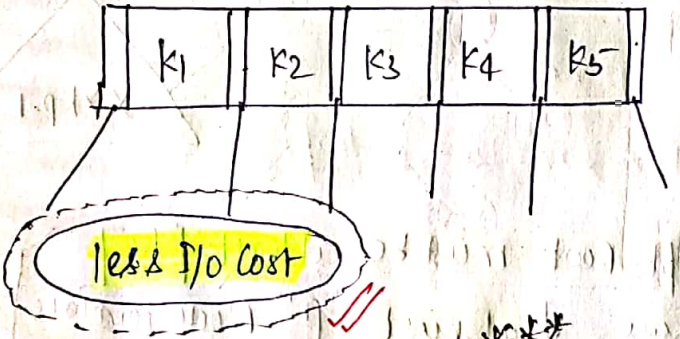
If Balanced ~~B+~~ Binary Search tree is used for DB Index :-



P.T.O.



If B/B+ Tree used for DB Index



★ S.memory is Huge so, it Needs only B+ Tree for storage purpose of Index (Search Tree)

*** Search Tree application (DS used for Search)

DB Index

MM application

[DS stored in mm used for Search]

S.m Application

[DS stored in Disk Disk Blocks should access Based on Search]

① Less # of Keys
n = 100's
AVL Tree is Best

② Huge # of keys
n = Lakhs
B Tree is Best

③ less No. of Keys
n: 100's
B+ Tree is Best

④ Huge # of Blocks
n: Lakhs
B+ Tree is Best

(less I/O cost)

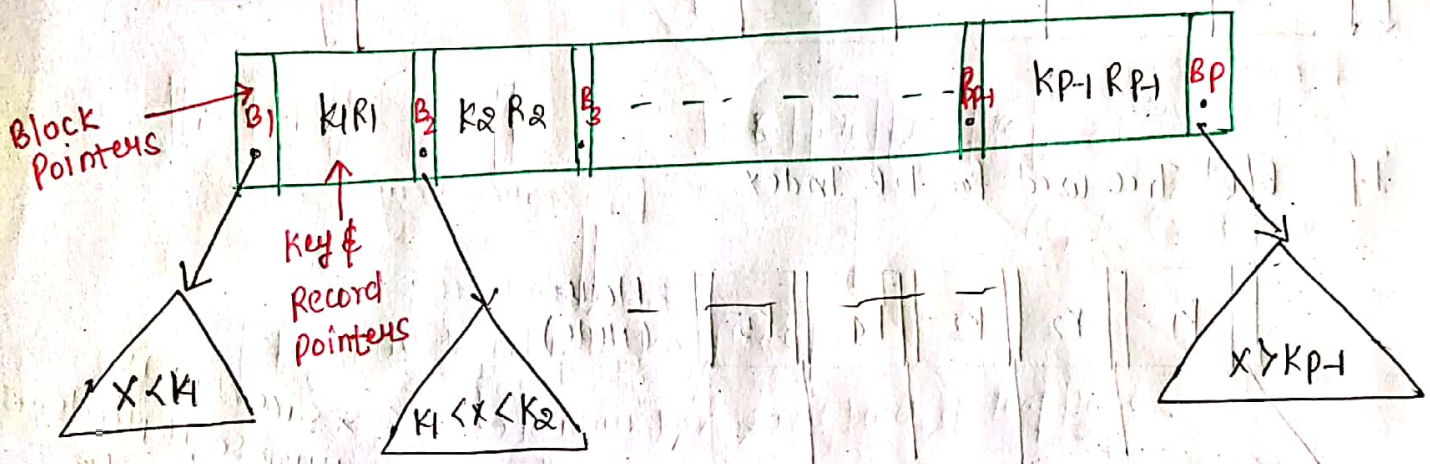
Lecture - 18

B-Tree & It's construction

B-Tree definition:

Order p → Maximum Possible Block Pointers which can be store in one B tree Node.

1] If a Node structure have p Block Pointers then (p-1) keys (or) (p-1) record ptr. it have.



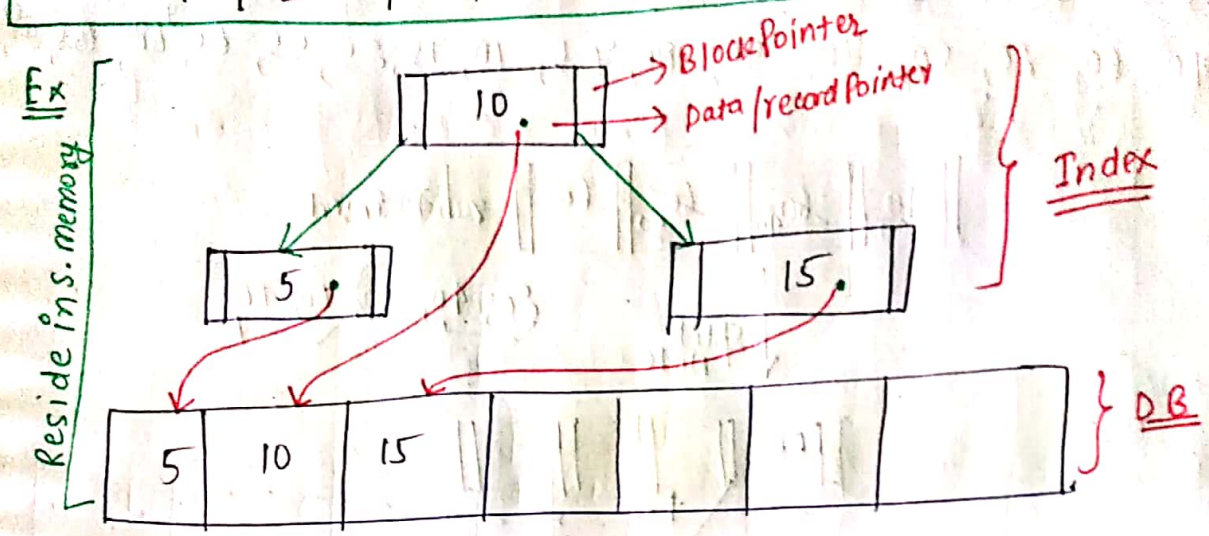
2] Every Internal node except Root must be atleast $\lceil p/2 \rceil$ Block pointers with $(\lceil p/2 \rceil - 1)$ keys & at most 'p' Block Pointers & (p-1) keys.

3] Root Node can be at least 2 bP's with 1 key & at most p block pointers with (p-1) keys.

4] Every leaf Node must be at same level.

Tree Balancing Rule

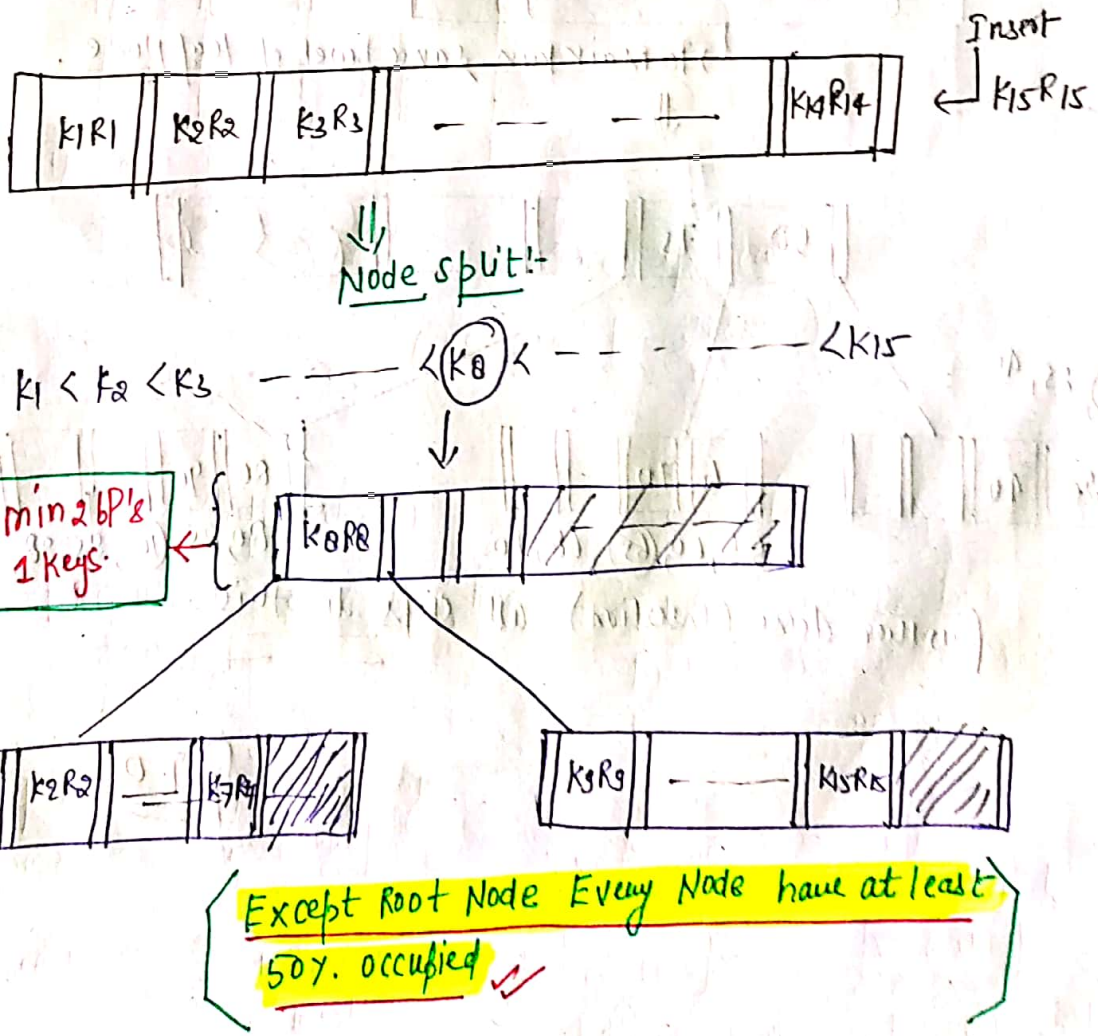
- Block Ptr / Tree ptr pointed to index Node.
- Data Ptr / Record ptr pointed to dB file.



- for order P:15 (It means max is ~~Data~~ Block Pointer's are possible)

if their are More than (15-1) ⇒ 14 Block Ptr then split it into two pieces & mid Part will be Root Node.

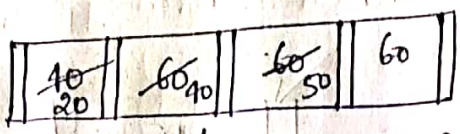
Ex :-



Que Construct B Tree with order P:5 (max Pointers)
& sequence of Keys:-

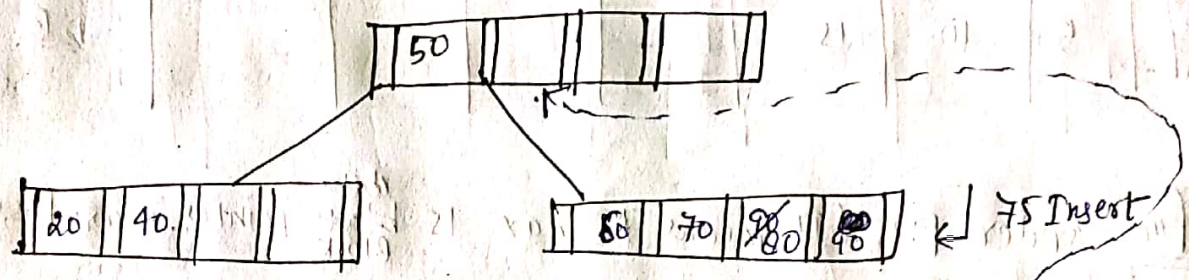
40, 60, 20, 50, 70, 90, 80, 75, 10, 15, 25, 55, 65, 68, 85, 98, 95

Sol^n I →



20, 40, (50), 60, 70

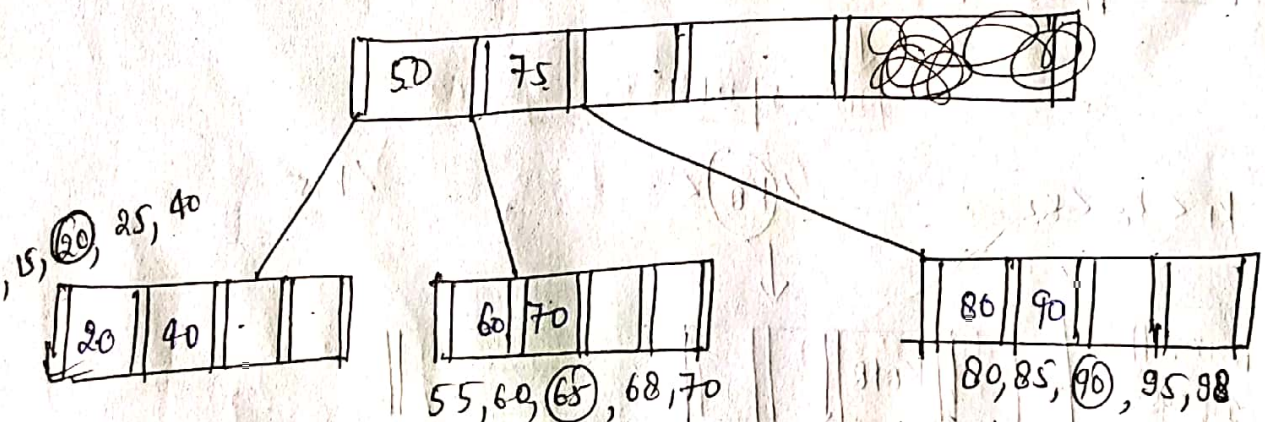
↓ split Node



60, 70, (75), 80, 90

Key must insert into leaf Node

↳ To maintain same level of leaf Node.



15, (20), 25, 40

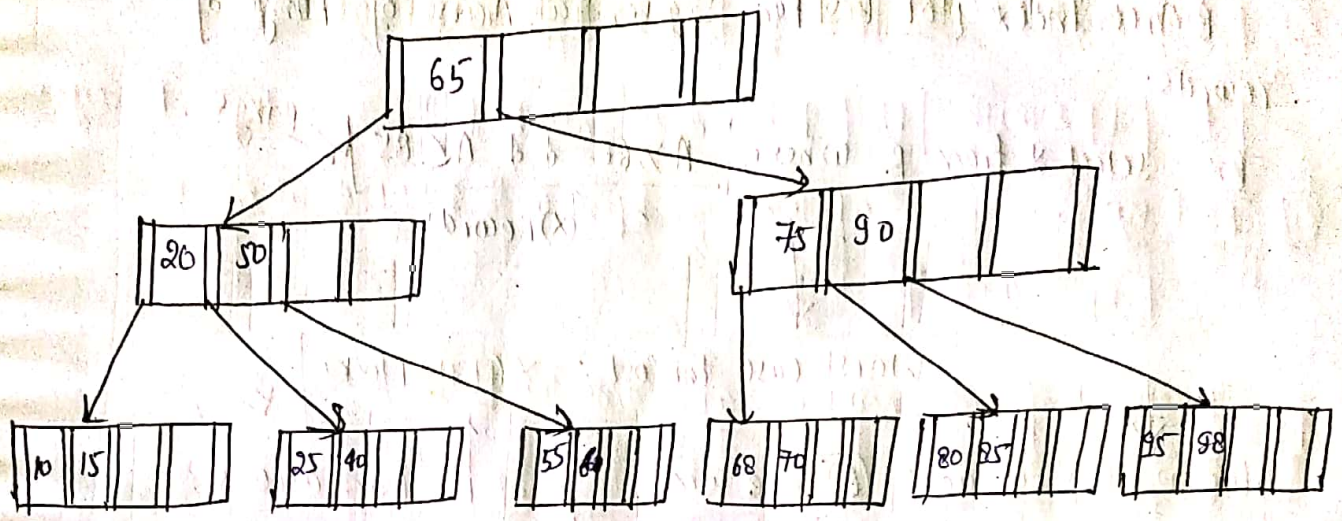
55, 60, (65), 68, 70

80, 85, (90), 95, 98

further done (insertion) all steps like this

P.T.O

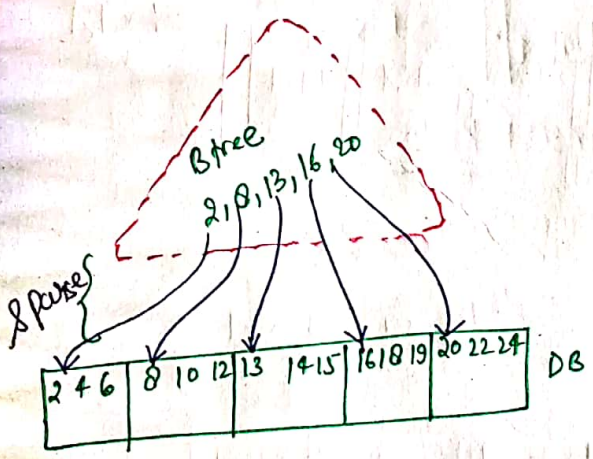
at Last You will get

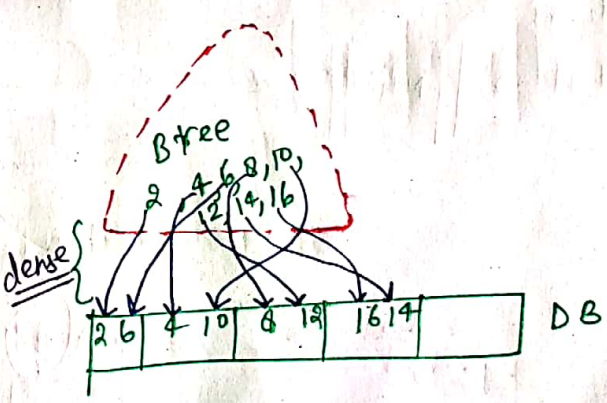


B tree Indexes can be Sparse or dense in Nature ✓

for ordered DB → Sparse Key in BTree

Unordered DB → Dense Key in BTree.





B. Tree Advantage :-

B Tree Index Best Suitable for Random Access of Anyone record.

Ex select * from R where R = 40 ✓ one record.

w.c I/O cost = (K+1) Blocks ✓✓

Dis-adv:-

B Tree Index Not Best for Sequential Access of Range of records.

Ex: select * from R where A >= 60 && A <= 85 → Range.

(x) Records.

Worst case Cost = X(K+1) Blocks

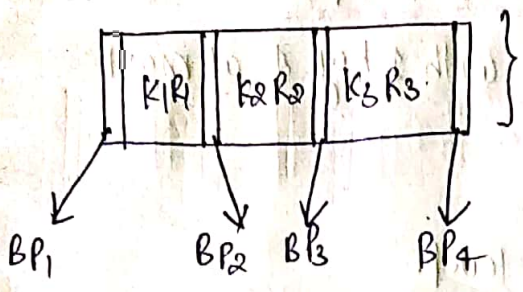
for Each access

due to this dis-adv (B)+ Tree come in Existance

All Indexes & dB all are Reside Into Secondary memory (Disk)

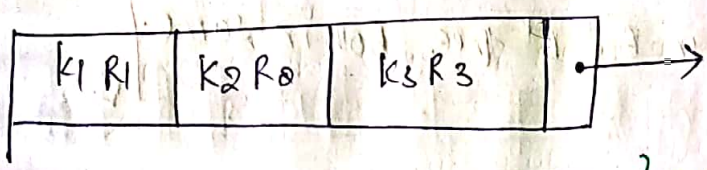
B+ tree :-

Order P: 4 :-

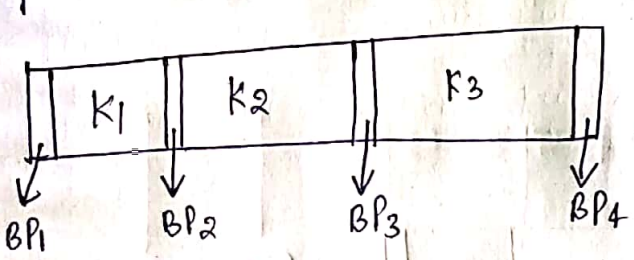


B Tree: Same Node structure for Internal Node & Leaf Node

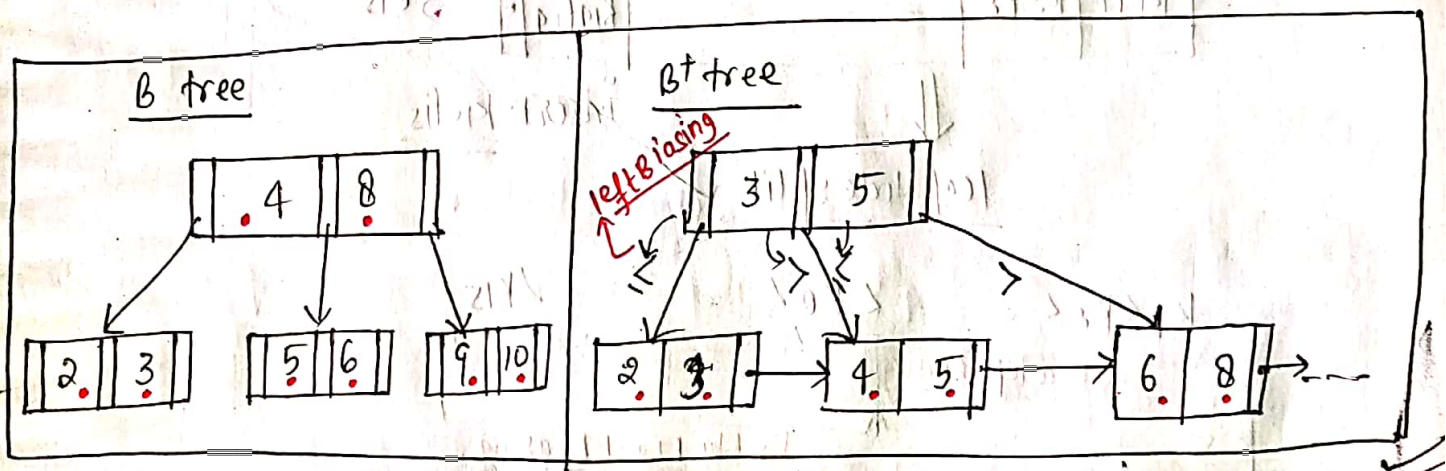
Order P: 4



B+ Tree leaf.



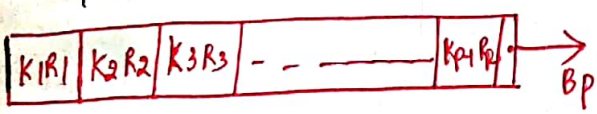
B+ tree Node (Internal)



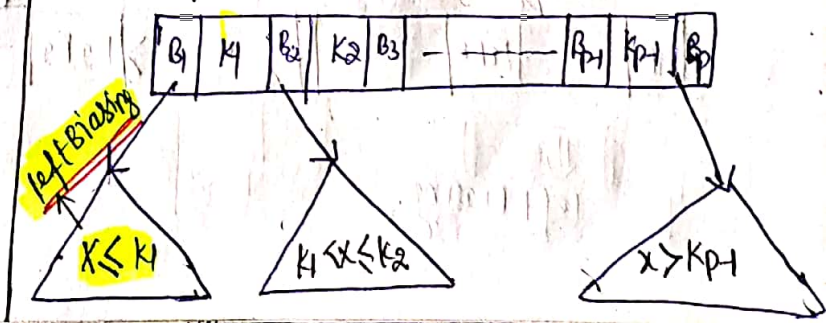
B+ tree definition :- Order P: Max Possible Pointers which can store in B+ Tree Node.

Node structure :-

1) **Leaf Node :-** Set of (Key, Rp) Pairs && one block ptr which Pointed to next leaf:



2) **Internal Node :** only keys && Block pointers :-



2] Every Internal Node Except Root must be at least $\lceil p/2 \rceil$ bp's with $\lceil p/2 \rceil - 1$ key & at most p bp's with (p-1) keys.

3] Root node can be at least 2 bp's with 1 key & at most p bp's with (p-1) keys.

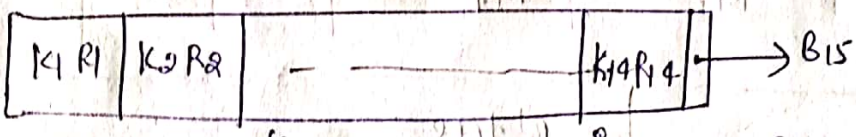
4] Every leaf Node must be at same level.

→ Node Balancing Conditions

Node Balancing Conditions in B & B+ tree Both in same.

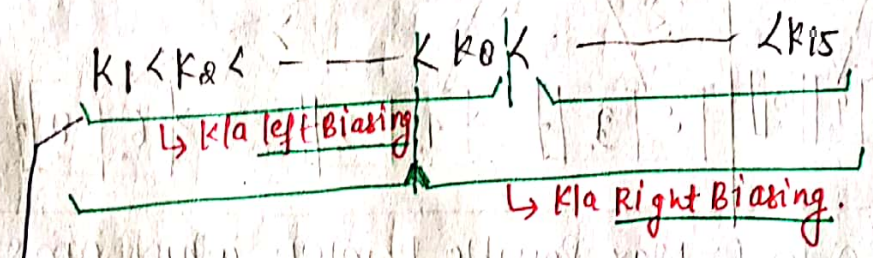
leaf Node split :-

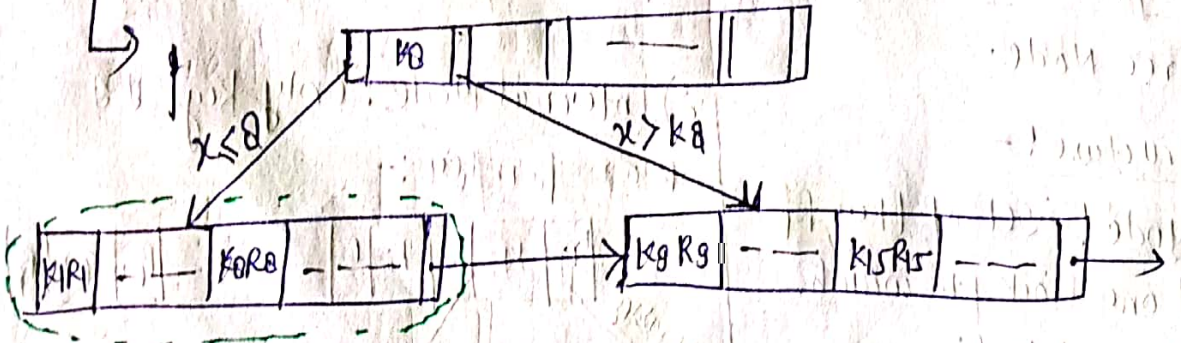
Order p : 15



insert K15 R15

leaf Node split





⇓ left Biasing.

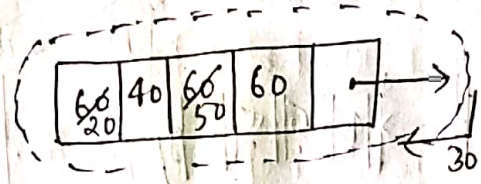
• Internal Node split same as B Tree Node split

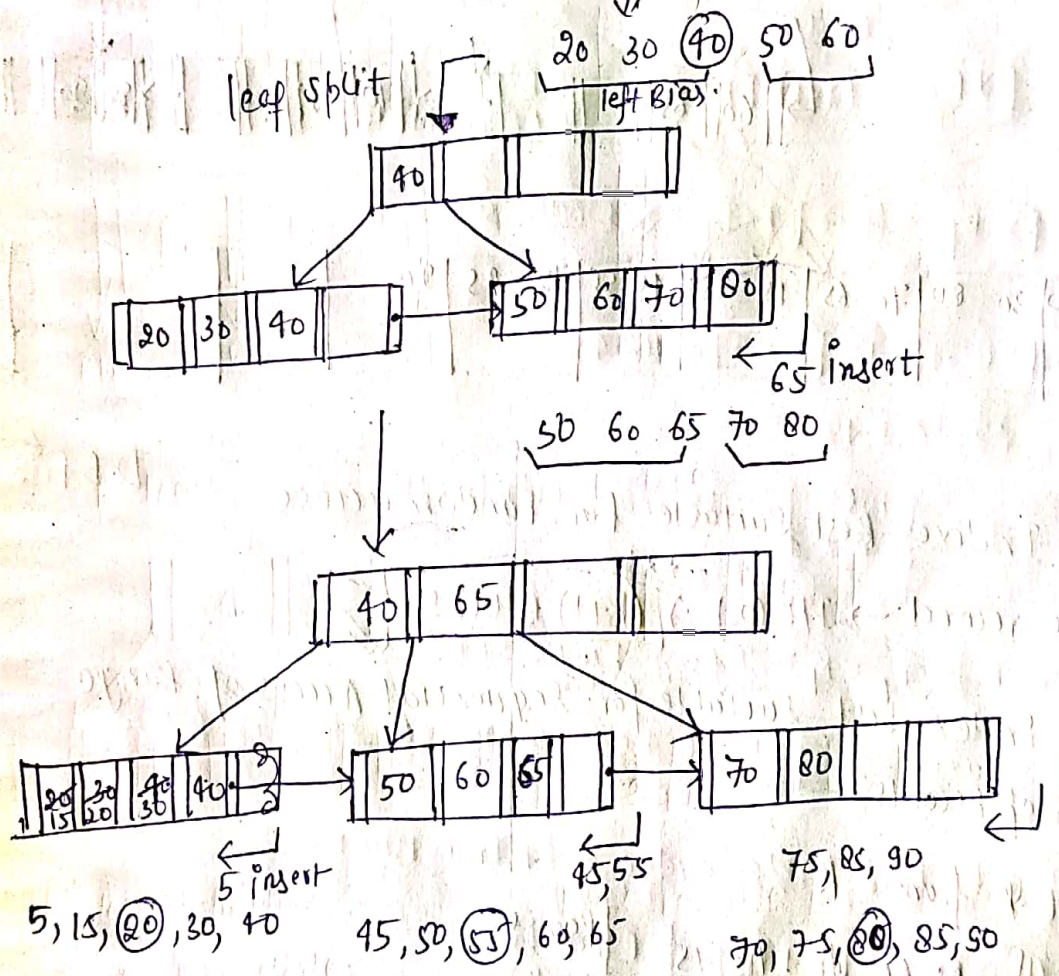
Construct B+Tree :-

Order P : 5 [max Possible Pointers Per node]

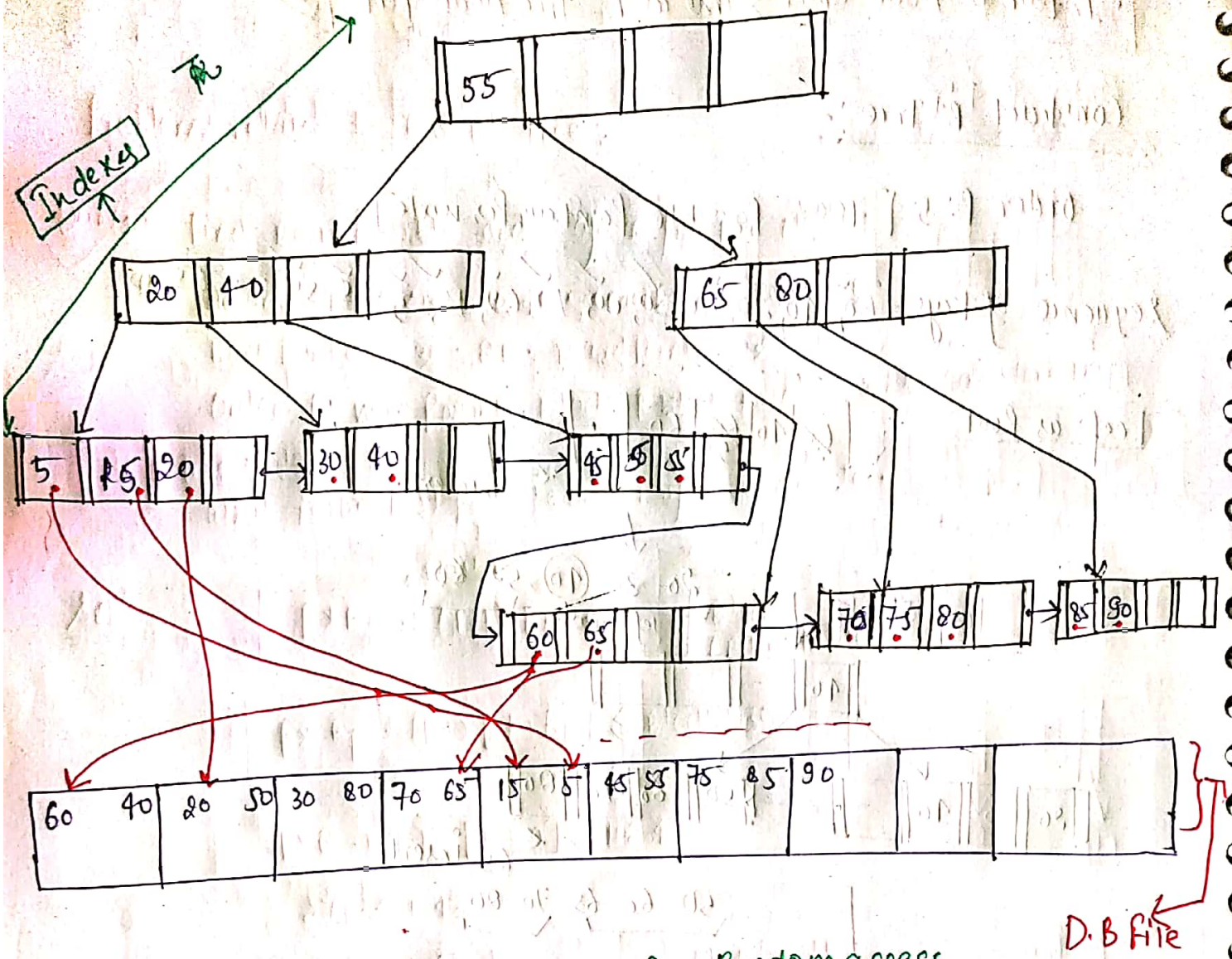
Sequence of keys :- 60, 40, 20, 50, 30, 80, 70, 65, 15, 5, 45, 55, 75, 85, 90

Leaf as Root





P.T.O



① B+ Tree Indexes Best Suitable for Random access of any one record → I/O Cost ⇒ (K+1) Blocks ✓

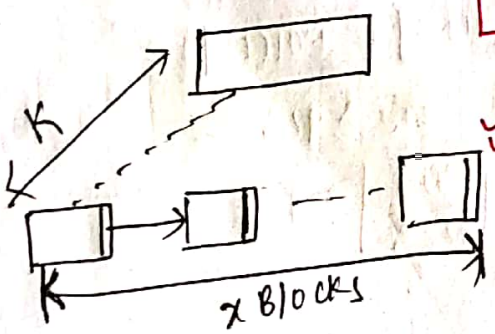
② B+ Tree Index best suitable for Sequencial Access of Range of Records

Select * from R

Where A ≥ 60 & A ≤ 75 Range.

↳ I/O Cost

(K+x) Blocks.



- If left Biasing used ∴ keys which are stored in internal nodes of B+ Tree are max key of each leaf node ~~expect~~ except last ~~Node~~ leaf Node.

*** # of Internal Nodes Key = # of leaf Nodes - 1

- If Right Biasing used keys which are stored in internal nodes of B+ Tree are min key of Each leaf Node except first ~~Node~~ leaf Node.

*** ⇒ Complete Tree must be exclusive left Biasing or Right Biasing

~~Lec~~ Lecture – 19

Problems related to B & B+ Tree :-

Que (1) Find Best Possible order of B / B+ Tree Node.

(a) Assume Block size = 1024B, search key size = 10B.
Block Pointer size = 8B, record Pointer size = 9B.
Order P: max Possible Block Pointers which can store in B tree Node.
What is the Best Possible order of B Tree Node.

Soln

Block

$P * B_p + (P-1)[K + R_p]$ ≤ Block (B Tree Node)

$P * B_p + (P-1)[K + R_p] \leq$ Block size.

$P * 8 + (P-1)[10 + 9] \leq 1024$

$27P \leq 1043$

$P = \left\lfloor \frac{1043}{27} \right\rfloor = 38$ max Block Pointer

Que $B = 512B$, $B_p = R_p = 10B$, Key $(K) = 5B$.

Order P: Maximum possible keys which can store in B Tree node.
What is best Possible order of B Tree node.

Soln

$(P+1)B_p + P[K + R_p]$ ≤ B tree Node (Block)

$(P+1)10 + P[5+10] \leq 512$

$25P \leq 502$

$P = \left\lfloor \frac{502}{25} \right\rfloor = 20$ max Keys.

***

Que-③

Block size = 1024 B | Key Rp = 10 B

Bp = 8 B | Rp = 9 B

Order P: max possible pointers which can store in B+Tree Node.

What is the Best Possible order of B+Tree.

(1) Internal Node (2) leaf Node

(1) for Internal Node

Order P: max pointers [P Bp's]

P* Bp + (P-1) Keys ≤ Block (B+ Tree Node)

Possible order in B+Tree for Internal Node

$P \times Bp + (P-1) Keys \leq Block\ size$

$P*8 + (P-1)\ 10 \leq 1024$

$P = \lfloor \frac{1034}{10} \rfloor = 57$ max bp

(2) for leaf Node :-

order P max Ptr

(P-1) Rp | Lbp

(P-1) [K+Rp] + Bp ≤ Block

$(P-1)[K+Rp] + Bp \leq Block\ size$

$(P-1)[10+9] + 8 \leq 1024$

$19P \leq 1035$

$P = \lfloor \frac{1035}{19} \rfloor = 54$

Possible order in B+ Tree for leaf Node

Imp ***

Note:- If size of Block Pointer (Bp) = size of Record Pointer Rp

Then B+ Tree Internal Node order = B+ Tree leaf Node order

(P) (P)

Que ④ Block size = 512 B

K = 5 B $B_p = R_p = 10$ B

Order p: Max Possible keys which can store in B+ Tree node

What is best possible order of B+ Tree Node?

Sol^{n} Order p: max keys per node

Here $B_p = R_p$ → So, Either we calculate it by Internal Node or by ~~External~~ leaf Node Both Ans are same.

By Internal Node calculation:-

$(p+1)B_p + p*keys \leq Block$

$(p+1)10 + p*5 \leq 512$

$p = 33$

By leaf Node Calculation

$p[K+R_p] + B_p \leq$ Block size

$p = 33$ ✓

Note:

If size of B_p = size of R_p

Then (B+ Internal node order) = (B+ Tree leaf Node order)

Analysis :-

B = 1024 B
K = 10 B, Rp = 9 B, Bp = 8 B

B Tree Node

Order p : 38 ✓
(max bP)

B^+ Tree Node → Preferred for data Base Index

Internal Node order = 57
leaf Node order = 54

Means :- If Block size of B Tree = Block size of B^+ Tree ✓

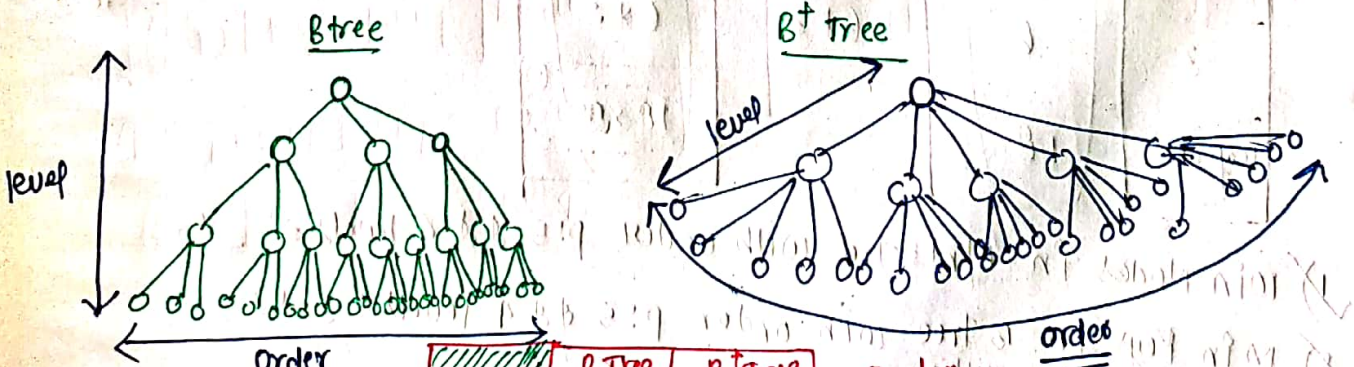
Then ***

(1) (Order p of B Tree Node) < (order p of B^+ Tree Node) → more pointer Node generated in B^+ Tree compare to B Tree ✓

(2) (# of B tree Nodes for n distinct keys) > (# of B^+ tree nodes for n distinct keys)

(3) (# of levels of B Tree index for n distinct keys (I/O Cost)) > (# of levels of B^+ Tree index for n distinct key (I/O cost))

So, for same Numerical data · (same Pictorial representation) ✓



	B Tree	B^+ Tree
Order	low	High
level	High	low
#Nodes	High	low

~~Order~~ ✓ Vinay

** Find min/max Keys in B/B+ Tree for given levels:-

(1) How many min/max Keys & Nodes in B/B+ Tree with order p: 5 & levels 4.

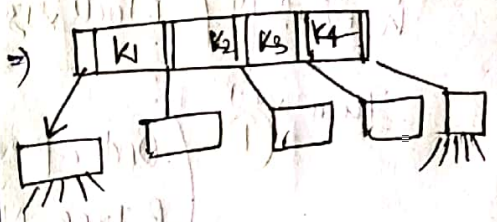
Order p: b/w 2 to p bp's for Root Node.

b/w $\lceil p/2 \rceil$ to p bp's for other Nodes.

b/w $\lceil p/2 \rceil - 1$ to (p-1) Key for left Node.

Soln:-

level	max Nodes	Max Bp's	max Keys
1	① 1	5	4 ②
2	5	5*5	5*4
3	25	25*5	25*4
4	125	—	③ 125*4

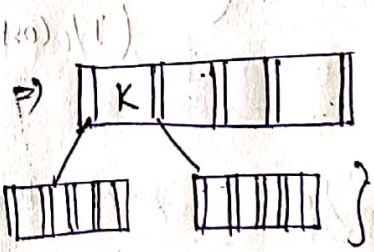
⇒ 

1) max Nodes in B/B+ Tree with order p:5 level 4 ⇒ 156

2) max Keys in B Tree with order p:5 & level 4 ⇒ 624

3) max Keys in B+ Tree with order p:5 & level 4 ⇒ 500

level	Min Nodes	min Bps	min Keys
Root=1	① 1	2	② 1
2	2	2*3	2*2
3	6	6*3	6*2
leaf=4	18	—	④ 18 ×2

⇒ 

1) min Nodes in B/B+ Tree with order p:5 && 4 levels ⇒ 27

2) Min Keys in B Tree with order p:5 && 4 levels ⇒ 53

3) Min Keys in B+ Tree with order p:5 && 4 levels ⇒ 36

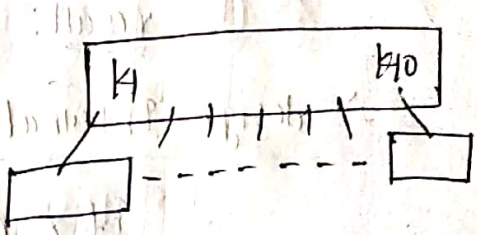
Q) min/max keys in B/B+ Tree for given levels.

a) How many min/max keys & Nodes in B Tree / B+ Tree with Order P: 5 & levels: 4

> Assume order P:- b/w (1) to (2P) keys for Root
> : b/w (P) to (2P) keys for other Nodes

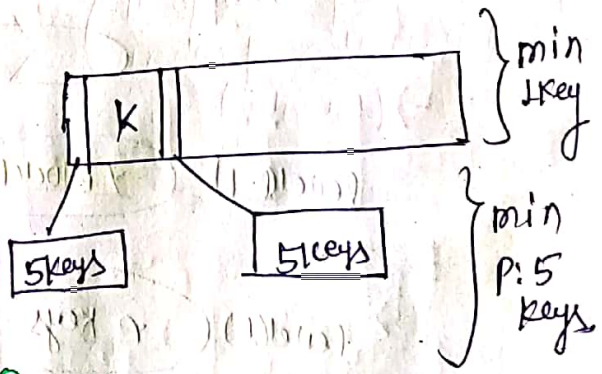
Sol^{n}:-

level	Max Nodes	Max BP's	Max Keys
Root → 1	① 1	11	10 ②
2	11	11*11	11*10
3	121	121*11	121*10
level → 4	133	—	133*10 ③



1) Max Nodes in B/B+ tree with order P=5, level 4: 1464
2) Max Keys in B tree with order P=5, & level 4: 14640
3) max Keys in B+ tree with order P=5, level 4: 13310.

level	Min Nodes	Min BP's	Min Keys
Root = 1	① 1	2	② 1
2	2	2*6	2*5
3	12	12*6	12*5
leaf = 4	72	—	72*5 ③



1) Min Nodes in B/B+ Tree with order P:5 & 4 levels ⇒ 87
2) Min Keys in B Tree with order P:5 & 4 levels. ⇒ 431
3) Min Keys in B+ Tree with order P:5 & 4 levels. ⇒ 360

Bulk loading B+ Tree :-

[construction of B+ Tree in bottom up approach]

leaf to root design.

⇒ **Design leaf Nodes :-**

Distribute keys into leaf Nodes.

result: # of leaf Nodes.

⇒ **design Internal Nodes :-**

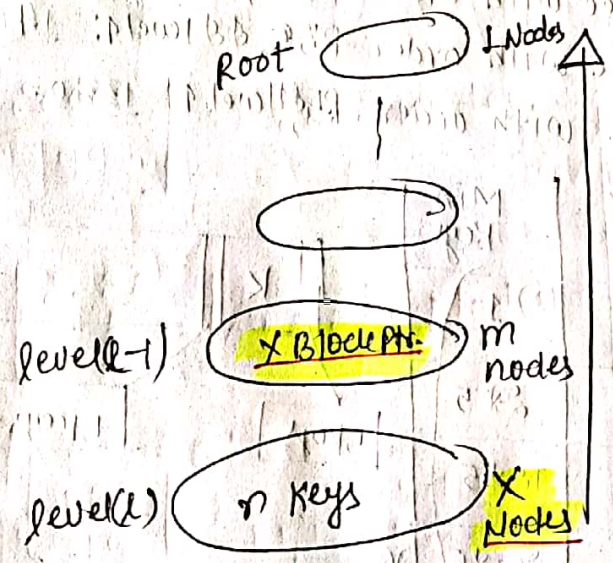
If x nodes at level l then

x Block ptrs at level $(l-1)$

Distributes Block pointers into nodes

result :- # nodes at level $(l-1)$

(repeat until Root design)



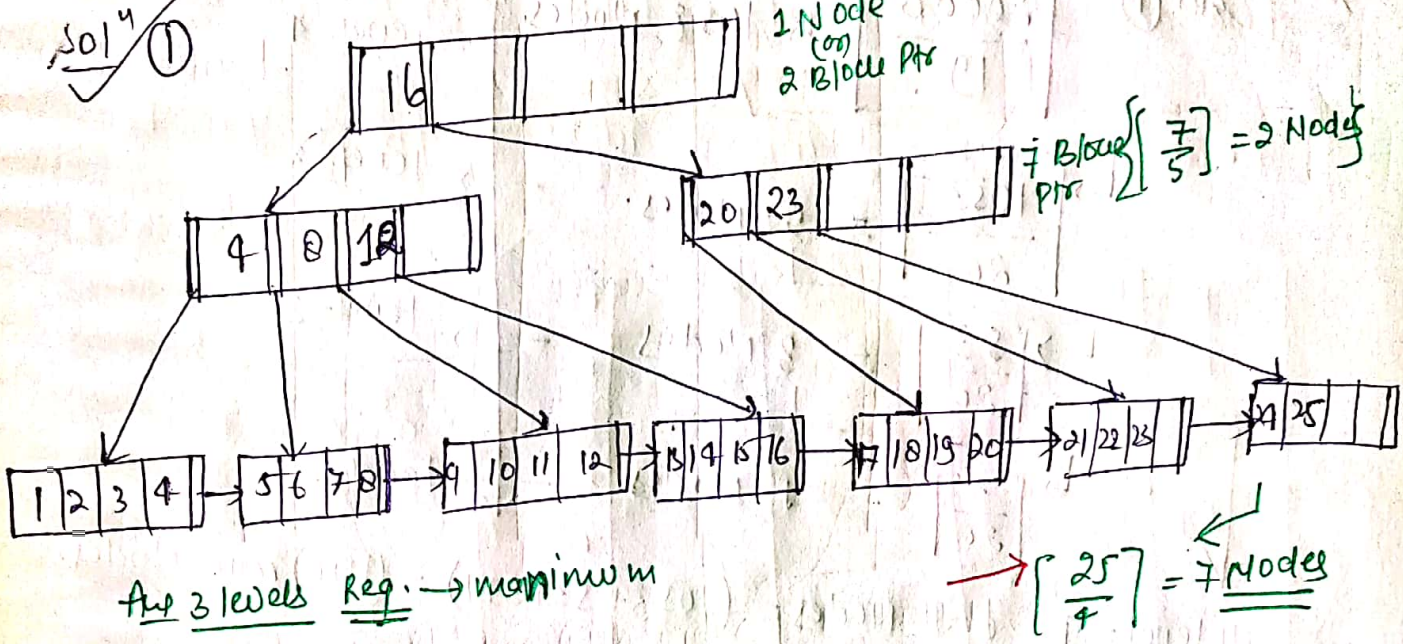
Q) Construct bulk loading B+ tree for.

Keys [1, 2, 3, 4, 5 - - - - - 24, 25] with order P: 5

① How many min levels:-

② How many max levels:-

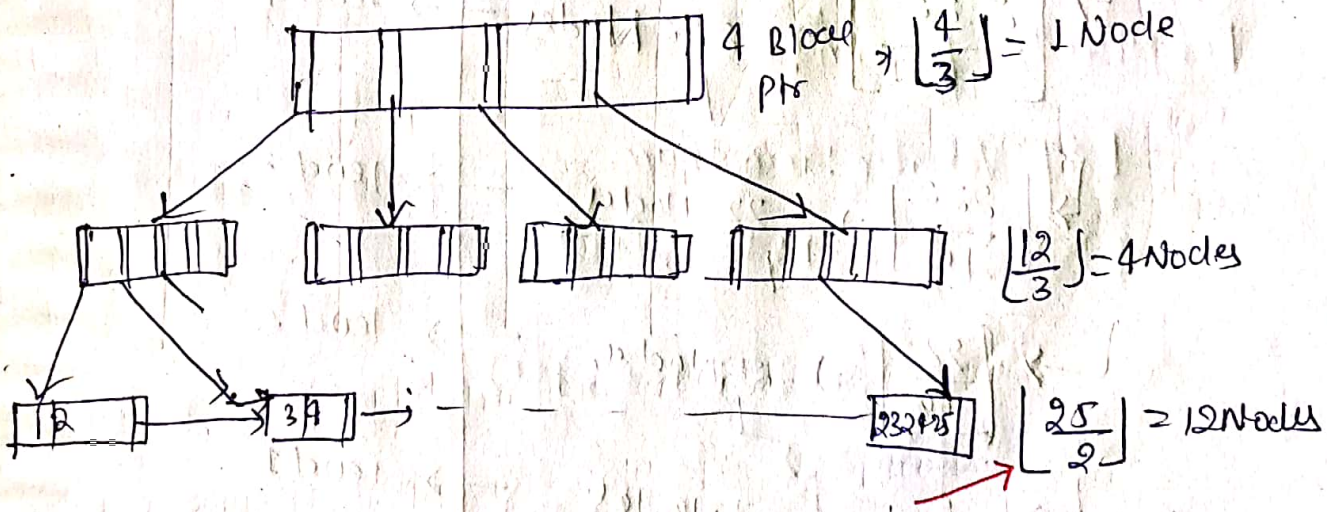
Sol^n ①



Ans 3 levels Req. → minimum

Sol^n ② minimum 3 levels Req

↓ ex



Qn # of keys used for Index: 20000
order p: 20 (max ptr per Node)

① How many minimum levels for minimum Nodes of B+Tree?

② How many maximum levels of max ~~level~~ Nodes of B+Tree?

Soln ① $\left\lceil \frac{20000}{19} \right\rceil$ = 1053 Nodes → level-4

$\left\lceil \frac{1053}{20} \right\rceil$ = 53 Nodes. → level-3

$\left\lceil \frac{53}{20} \right\rceil$ = 3 Nodes → level-2

$\left\lceil \frac{3}{20} \right\rceil$ = 1 Node → level-1

level = 4 & # Nodes = (1053 + 53 + 3 + 1) = (1110) min Nodes.

② $\left\lceil \frac{20000}{9} \right\rceil$ ⇒ 2222 Nodes. → level 5

$\left\lceil \frac{2222}{10} \right\rceil$ ⇒ 222 Nodes → level 4

$\left\lceil \frac{222}{10} \right\rceil$ ⇒ 22 Nodes → level 3

$\left\lceil \frac{22}{10} \right\rceil$ ⇒ 2 Nodes → level 2

$\left\lceil \frac{2}{10} \right\rceil$ ⇒ 1 Node → level 1

↳ # levels = 5, # Nodes = (2222 + 222 + 22 + 2 + 1) ⇒ 2469 Nodes

→ If levels given to find min/max keys of B/B+ Tree

↳ (Top-down) → Root to leaf.

→ If #f keys given to find min/max levels of B+ Trees

↳ (Bottom up) → Bulk Loading leaf to Root.

→ If #f keys given to find min/max levels of B Tree

↳ (Top-down) Root to leaf.

Last type ques ↳ Best Possible order of B/B+ Tree.

Bulk loading is only for B+ Tree → Kyun ki Bulk loading me saare keys leaf Node Pr hi hote hain jise hm saare key ko ek baar me load kr skte hai. usse baad left Biasing ya Right Biasing ke through hm saare Nodes form krskte hain.

↳ To Find # levels
&
• To Find # Nodes

Kyunki B tree me saare keys leaf Pr Nhi hote hai to hm Bulk loading B tree me Nhi kr skte

Lecture -20

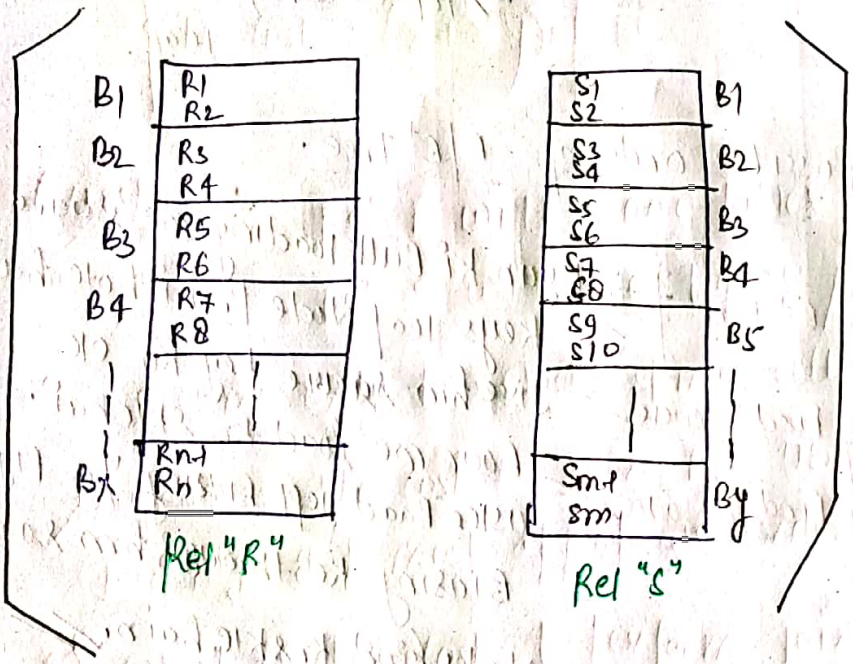
Finding Access cost after joining two tables. ($\bowtie$) → Access cost

Join Algo →
1. Nested loop Join Algorithm
2. Block Nested loop Join Algorithm.

Relational Schema R with n Tuples occupies in x blocks

&

Relational Schema S with m Tuples occupies in y blocks



Stored in Disk (sm)

For Nested loop Join :-

Loop Variable : record num.

$R \bowtie S \Rightarrow$ for (i=1; i≤n; i++)

{
  for (J=1; J≤m; J++)
  {
    Join (R_i, S_J)
  }
}

Dis Adv:-

{ More I/O Cost to perform Join if MM space allocated for Join is limited }

Block Nested loop Join :-

loop Variable:- Block num.

R ⋈ S ⇒ for (i=1 ; i≤x ; i++) // Block Num of R.

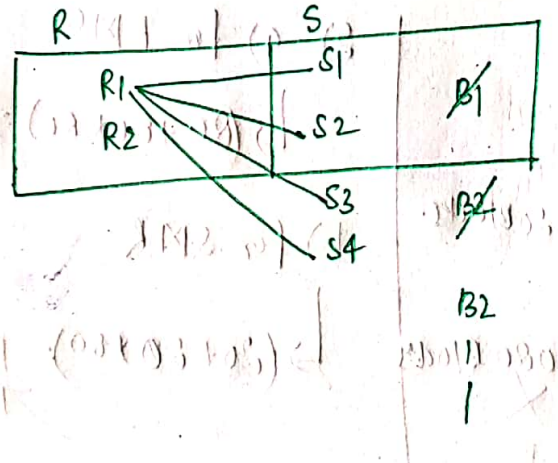
```
{
  for (j=1 ; j≤y ; j++) // Block Num of S.
  {
    ( Join all records of ith ~~Block~~ records of
      ith Block of R, jth Block of S)
  }
}
```

⇒ Assume mm space allocation for Join is only two Blocks.

[at any time one block of R records & one block of S records can store in mm].

a) access cost to perform Join using Nested loop Join (R ⋈ S)

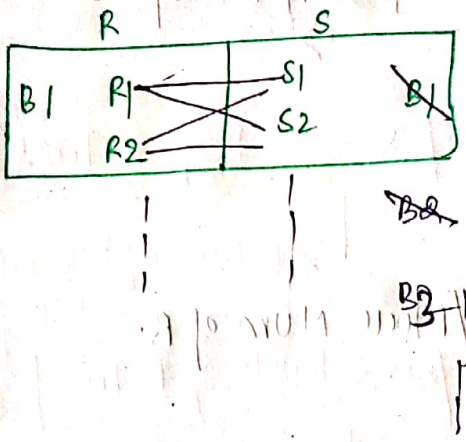
Ex



y Blocks access from S relation For Join one record of R.

So, I/O cost = (x + n*y) Blocks

Case - II BK



I/O
Y Blocks access from S Relation from Join one Block of R Records.

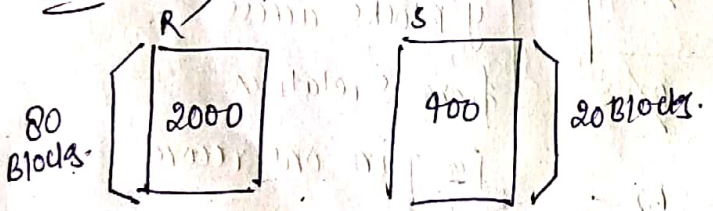
I/O Cost = (X + XY) Blocks

Questions:- Rel R with 2000 tuples / 80 Blocks.
Rel S with 400 tuples / 20 Blocks.
Assume only two Blocks of mm allocated for Join

1) What is I/O Cost to Join R, S using Nested loop Join Algo & & most appropriate order. can perform Join?

2) What is I/O Cost to perform Join of R, S using "Block Nested loop Join" Algo? (Most appropriate order.)

Sol^n 1) a) for R ⋈ S.



{80 + 2000 * 20} ⇒ 40,080 Blocks ✗

(b) for S ⋈ R

↳ (20 + 400 * 80) ⇒ 32020 Blocks

most appropriate Answer

2) a) for R ⋈ S

↳ (80 + 80 * 20) ⇒ 1680 Blocks ✗

b) for S ⋈ R

↳ (20 + 20 * 80) = 1680 Blocks

most appropriate Answer.