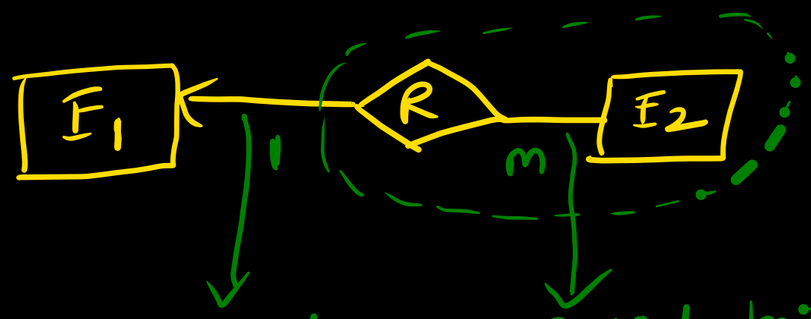
## All gate model questions:

1)



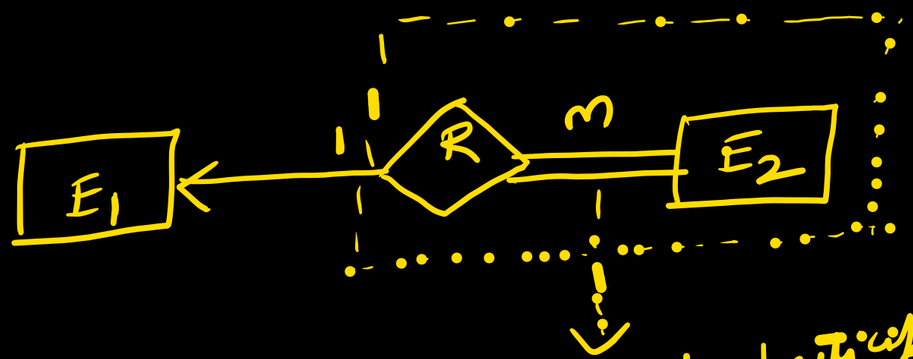

mandise ✓

Partial participation

Partial participation

min Rdbms Tables 2 and 1 FK is required.

(2)

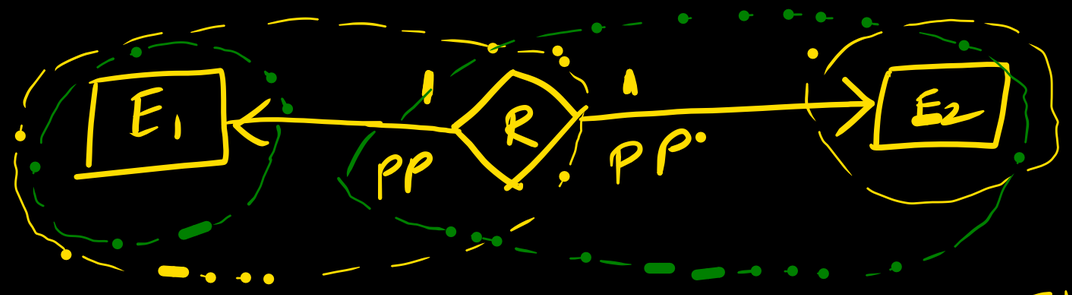



In RDBMS min 2 tables $E_1$, $RE_2$ and 1 FK

(3)

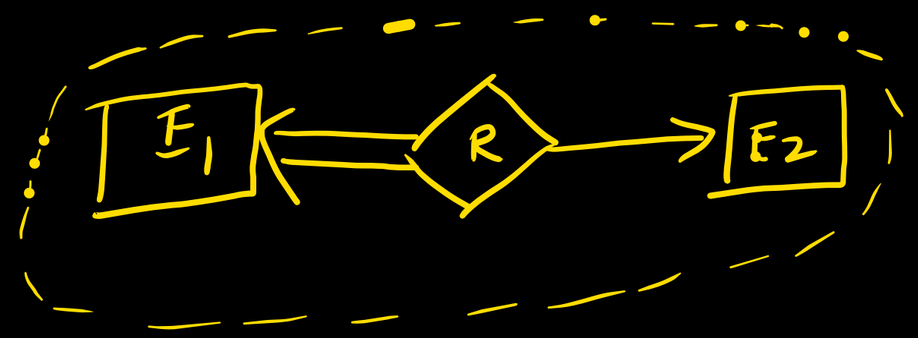

min RDBMS tables required are = 3 and 2 FKs are required.

4)



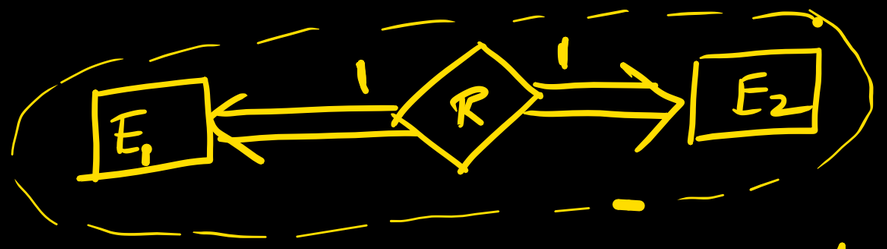

min two tables and 1 FK.

5.)



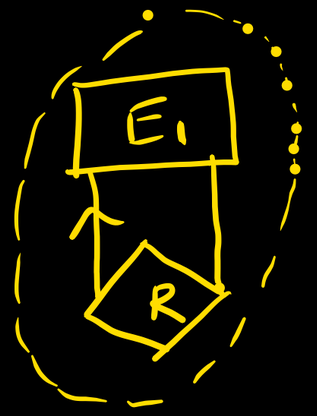

One RDBMS table && NO FK

6)




one RDBMS table no FK.

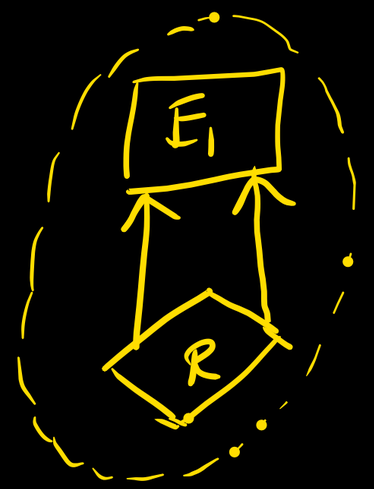
7)




min 1 ROOMS

& 1 FK

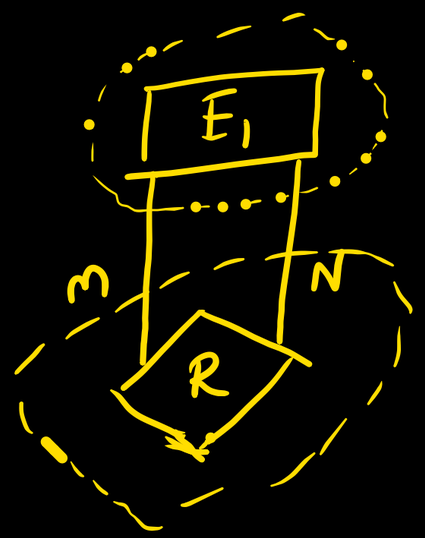
6)




0
min 1 ROBMS

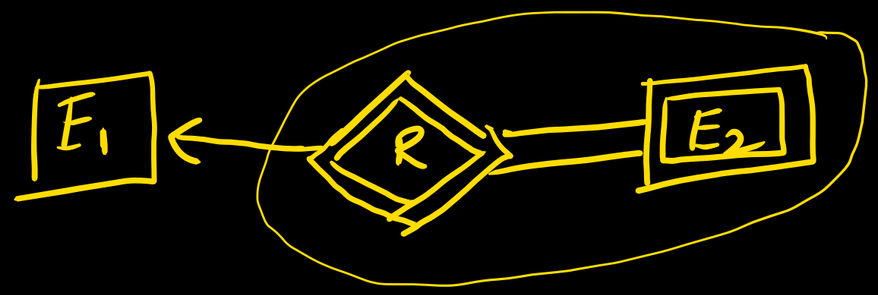
&& 1 FK

9)




2 ROBMS table ll 2 FKS

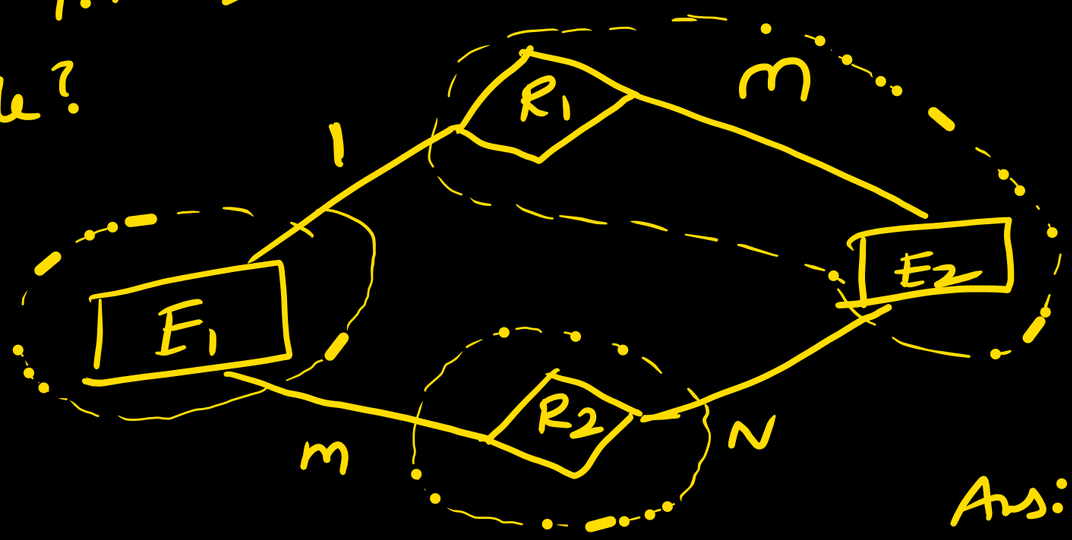
10)




min 2 tables $E_1$, $RE_2$
and 1 FK.

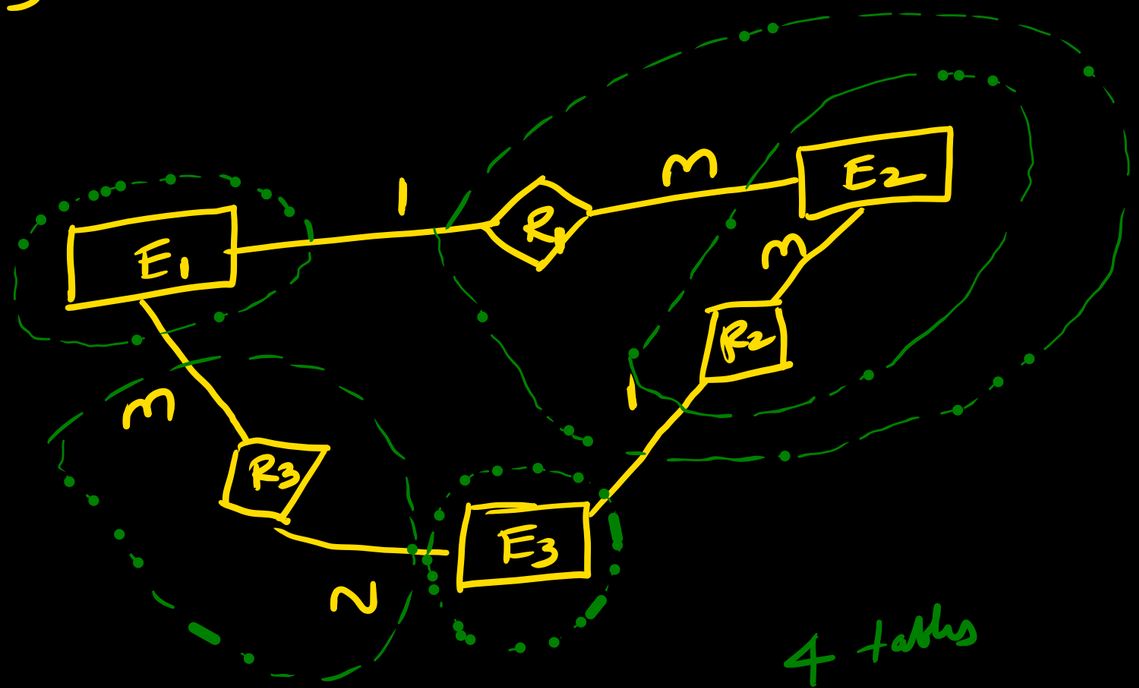
Q) Only 2 diff questions.

$E_1, E_2$ Entity sets, $R_1$ $R_2$ are relationships related b/w $E_1$ and $E_2$ with 1:m & m:N respectively. How many min rel (tables) are possible?




Ans: 3.

Q) $E_1$ $E_2$ $E_3$ entity sets. $R_1$ $R_2$ $R_3$ relationship sets. $R_1$ is related b/w $E_1$ and $E_2$ with 1:m. $R_2$ is related b/w $E_2$ and $E_3$ with m:1. $R_3$ is related b/w $E_1$ & $E_3$ with m:N.




4 tables

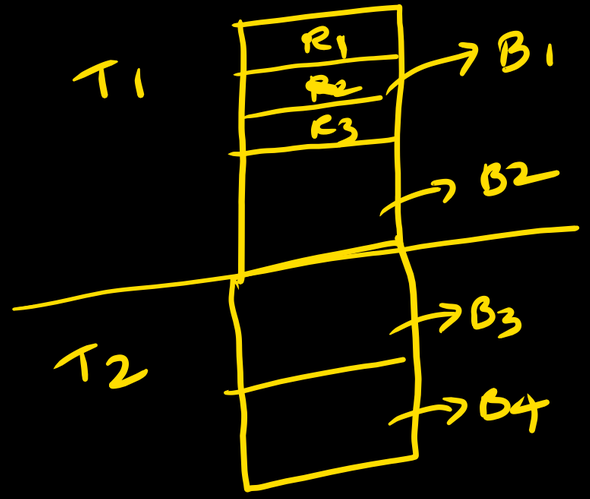
$E_1$, $E_2$ $R_1$ $R_2$, $E_3$, $R_3$

# File organisation and indexing.

DB is a collection of files (tables) ✓

File is a collection of Blocks (pages) ✓

Block is a collection of Records. ✓



→ We move the contents from Disk to mm in block by block manner

→ Data access from Disk to mm is block by block

Records in DB file

Fixed length Records

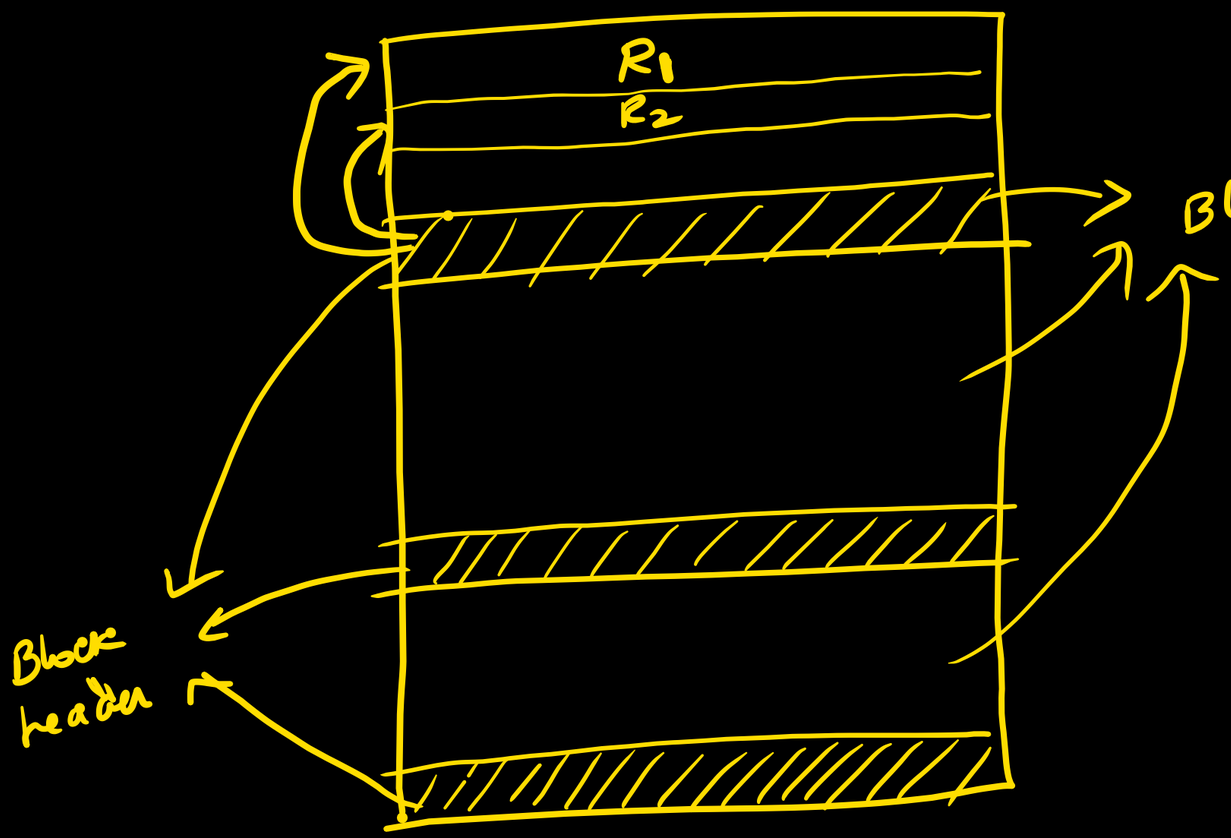
Create table R
( A char (100)
  B char (100)
  C char (200))

400 char is max size of record

Variable length Records

Create table S
( D char (20)
  E char (30)
  F text )

variable length



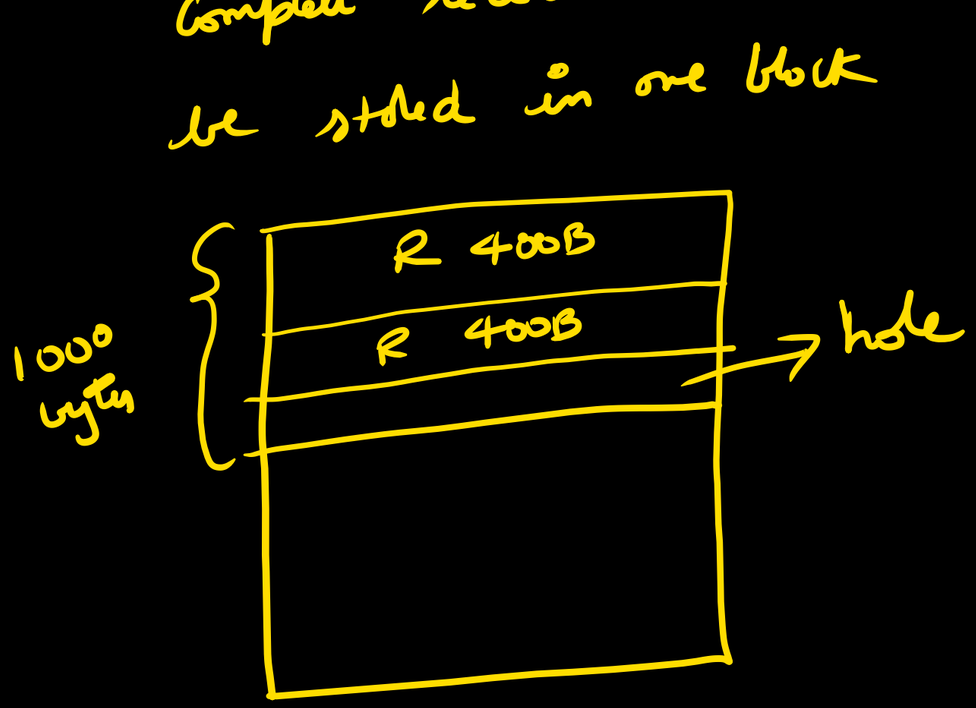

Block header used to store offset table to access records.

In exam, if they dont mention about Block header, consider it as '0'

# Record organisation in DB file:

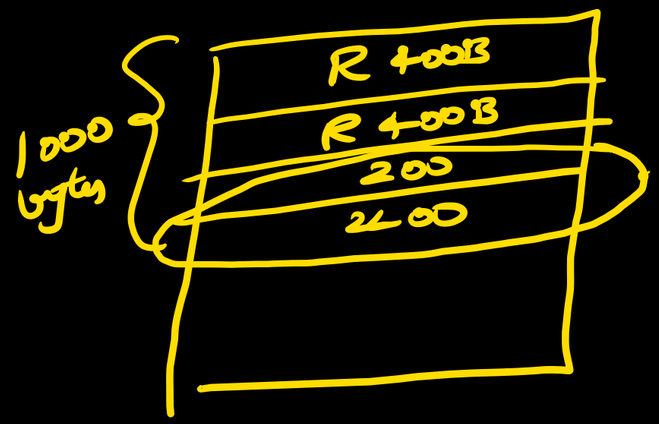
## Un Spanned

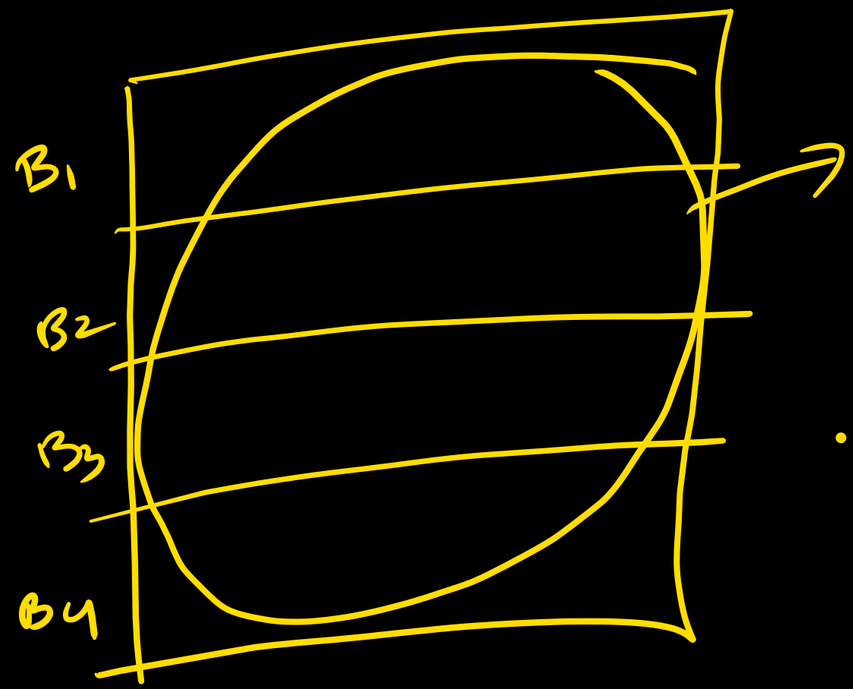
Complete record must be stored in one block




## Spanned

Record can span more than one block.






To access record R1 we need 4 blocks.

∴ In spanned organisation, more disk access are required.

Fixed length records uses unspanned organisation

Var length Records uses spanned organisation

## Block factor of DB file:

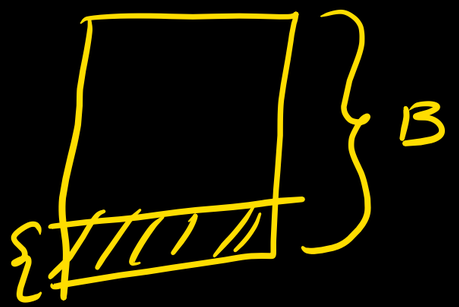
max possible records / Block

Block size : B bytes

Record size : R bytes

Block header size : H bytes

$$BF = \left\lfloor \frac{B - H}{R} \right\rfloor$$




Note: If block header size is not given, then take it as '0'.

**Problem:**

If Block size = 1000 B, (Rec = 400 B), BF of DB?

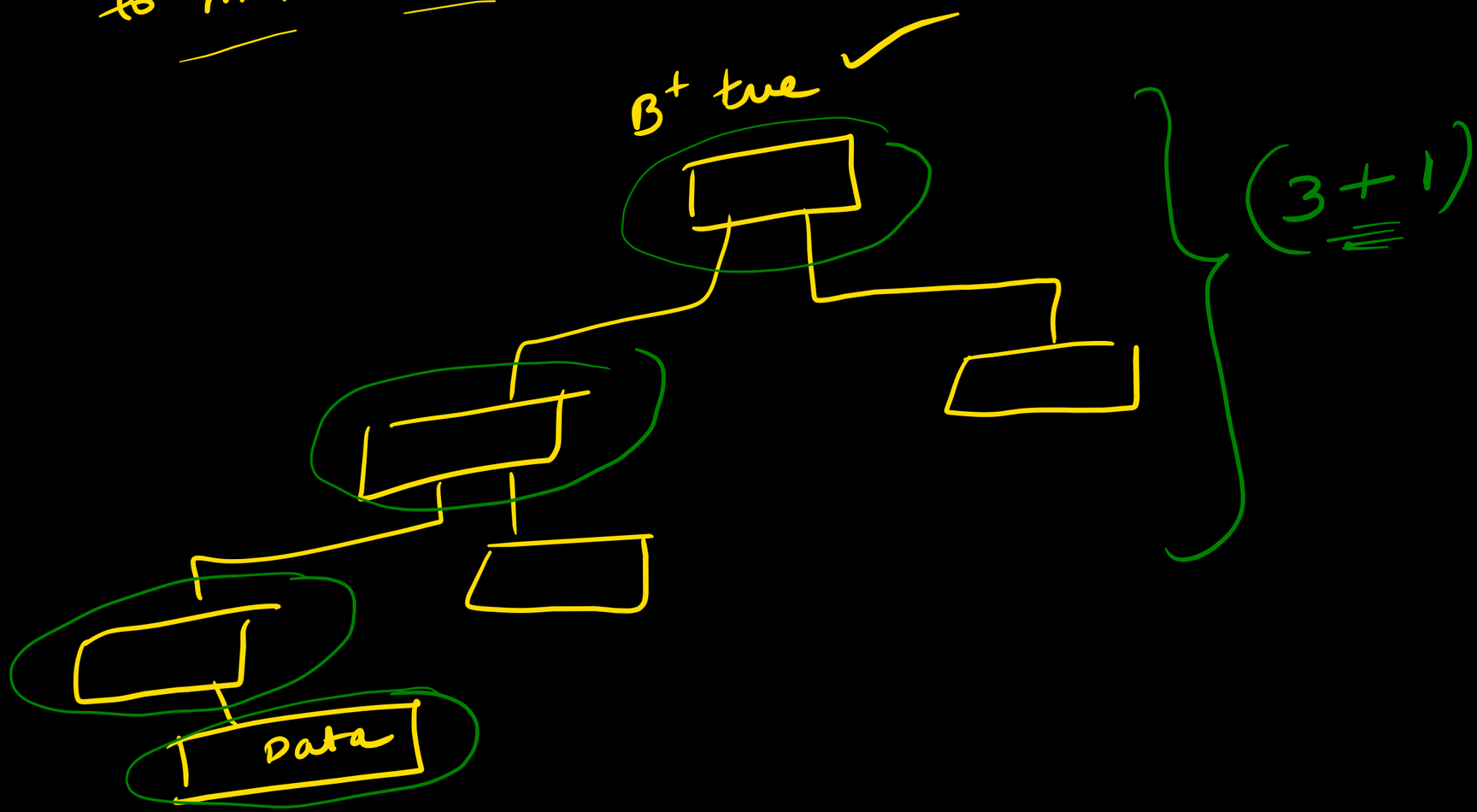
$$\left\lfloor \frac{1000-0}{400} \right\rfloor = \lfloor 2.5 \rfloor = 2$$

Only in case of fixed length records, we go for BF.

In variable length records, BF doesn't apply.

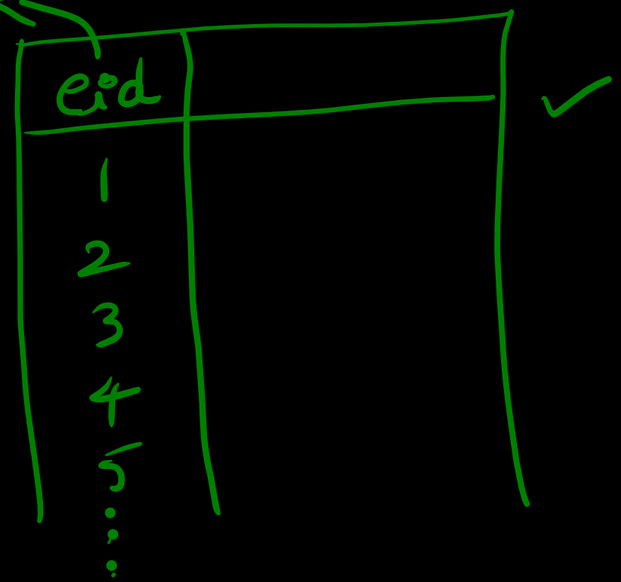
I/O cost:

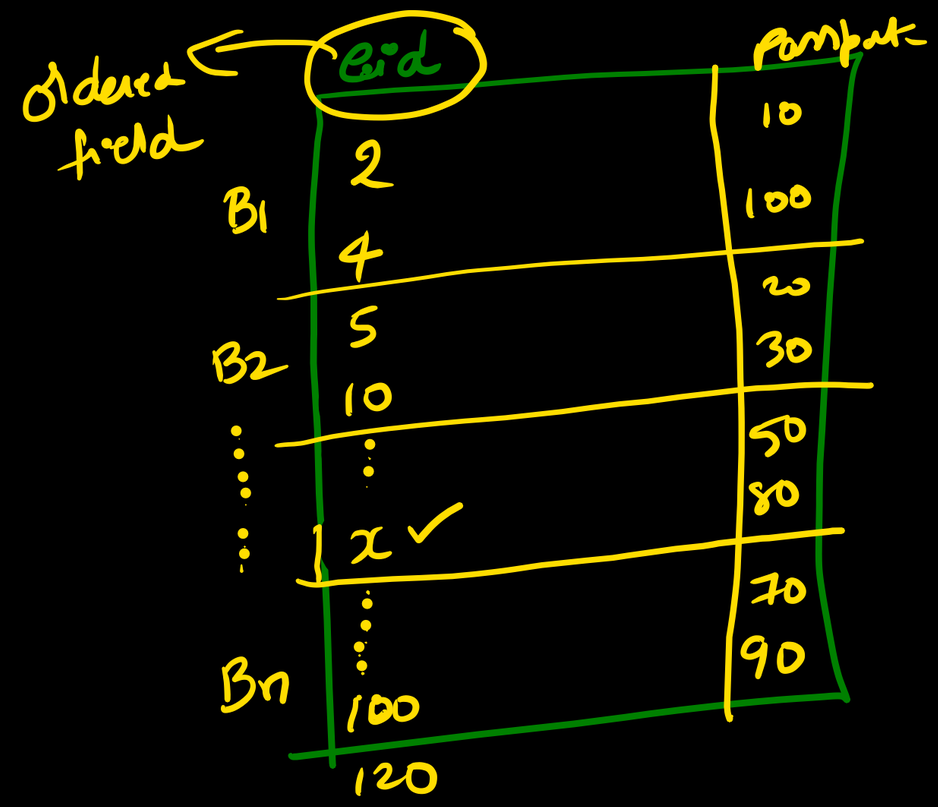
No of disk blocks (PB File blocks) required to transfer from DISK to mm to access required record.



In a database table, generally DBMS will organise it in a physical order based on a particular field. That field is called ordered field of the file.

ordered field






I/O cost to access record from DB file without index

a) using ordered field — Binary search.

```
Select *
From emp
where eid = x;
```

ordered field.

$\lceil \log n \rceil$ blocks

b) using unordered field

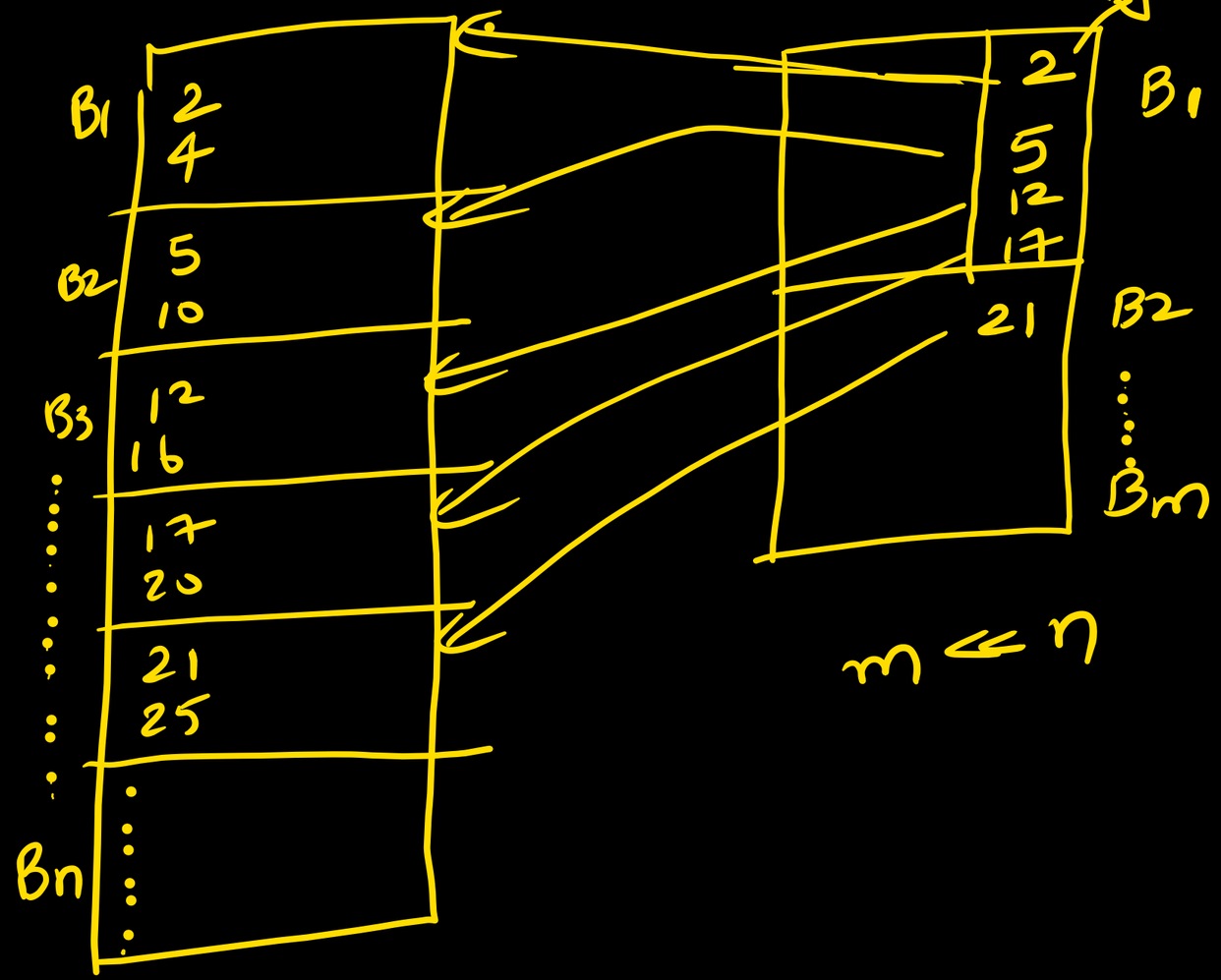
```
Select *
from emp
where passport = X;
```

unordered.

worst case I/O cost = n blocks.

→ If we use indexing I/O cost will reduce

→ I/O cost to access record from DB file using Index ordered.



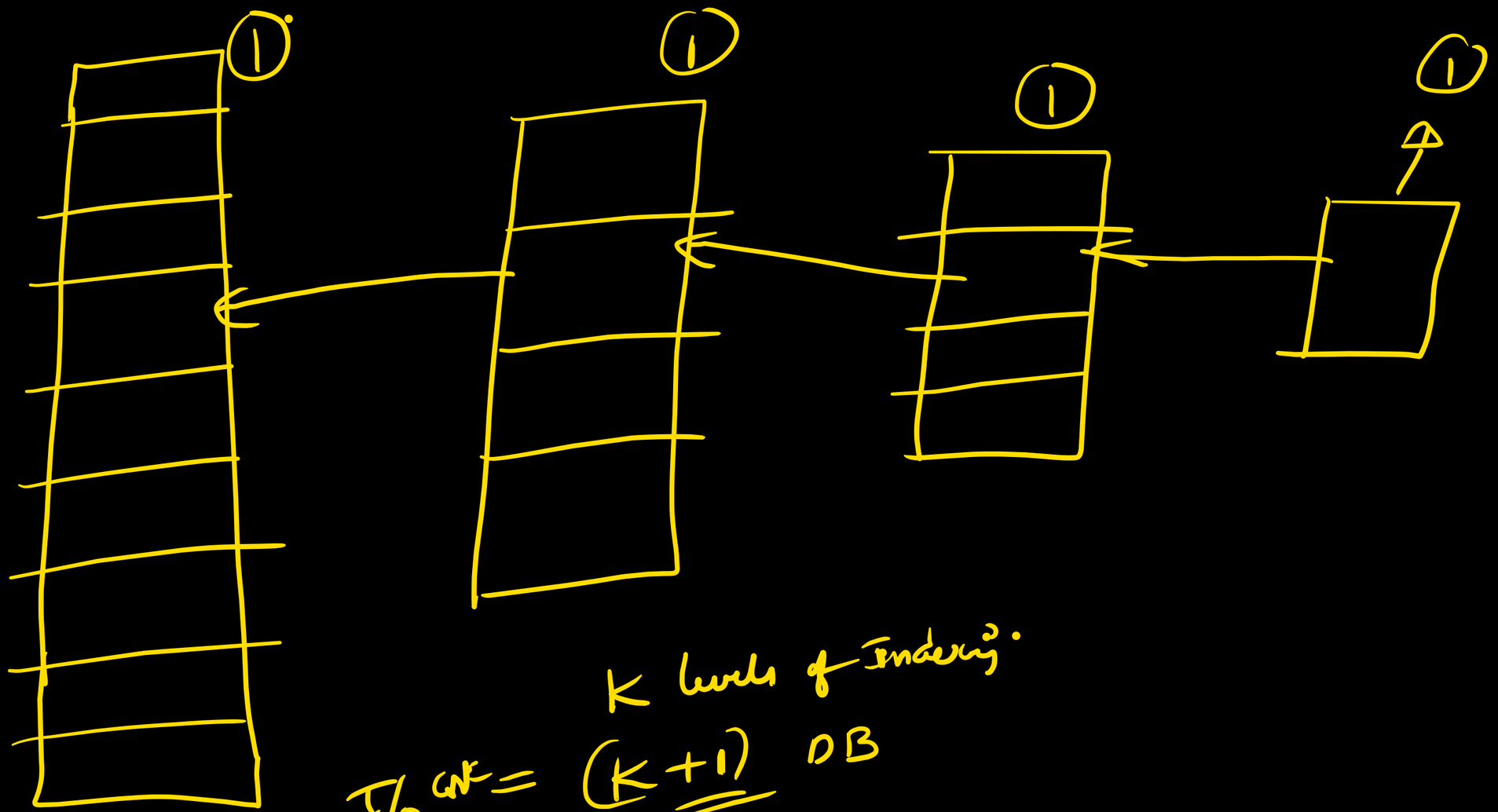

$m \ll n$

I/O cost in this case

$\lceil \log_2 m \rceil + 1$ block

"Index is always ordered"

To reduce the I/O cost we can go with multi level index



K levels of indexing.

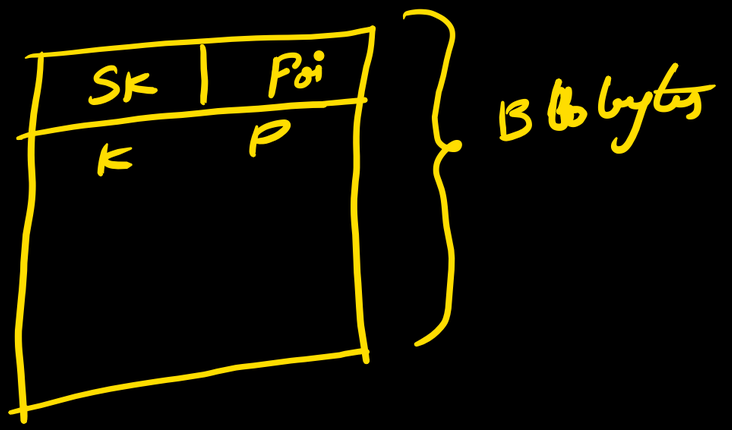
I/O cost = (K+1) DB

## Index file:

Each entry of index file will have two fields

<Search key, pointer>

K bytes  P bytes.




Block size B bytes

How many ~~indices~~ index records are there in one block (BF)

$$BF = \left\lfloor \frac{B - H}{K + P} \right\rfloor \text{ entries per block}$$

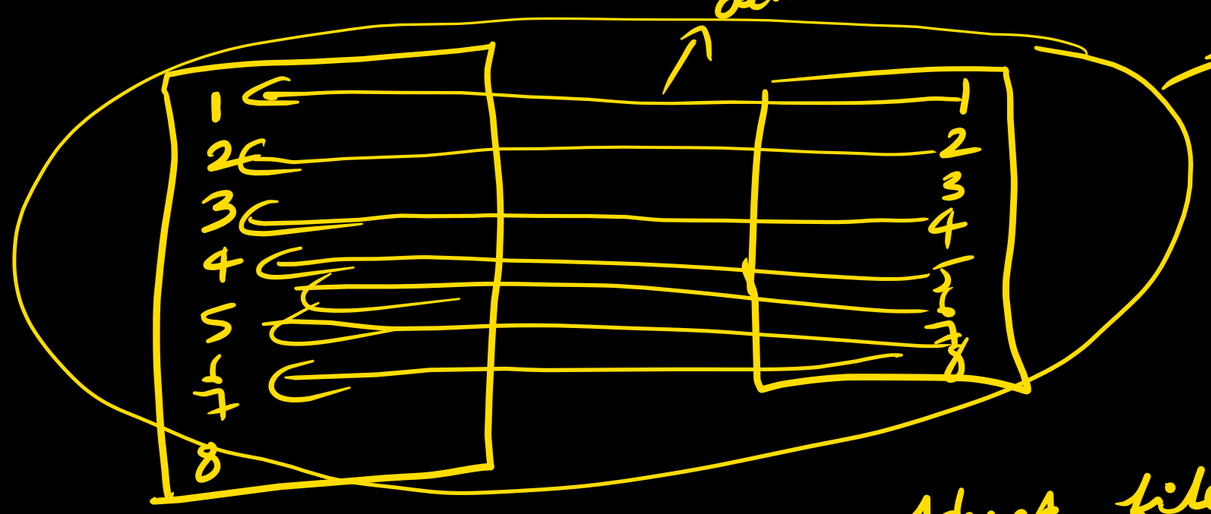
multi level index:

Index to index until last level contains single block of index

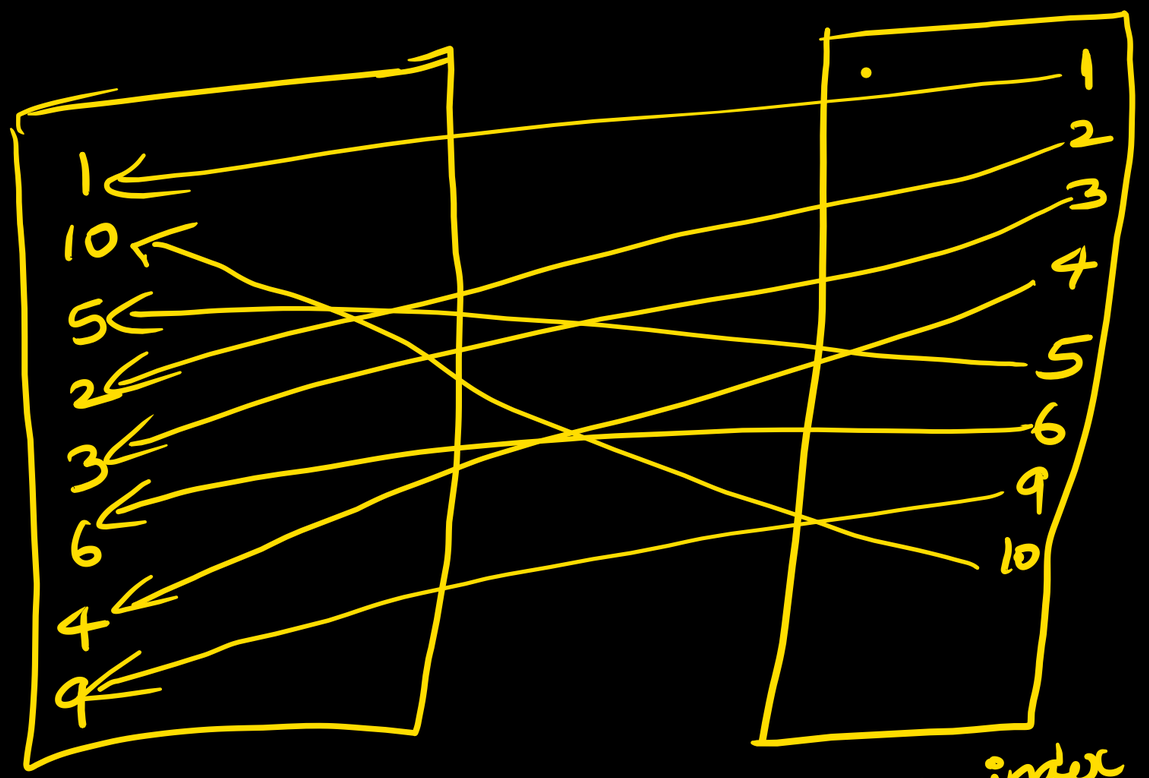
## Types of indices:-

Dense index (node no of index entries)

For each record in DB file there must be entry in index file.




Dense index can be build on ordered file as well as unordered file.

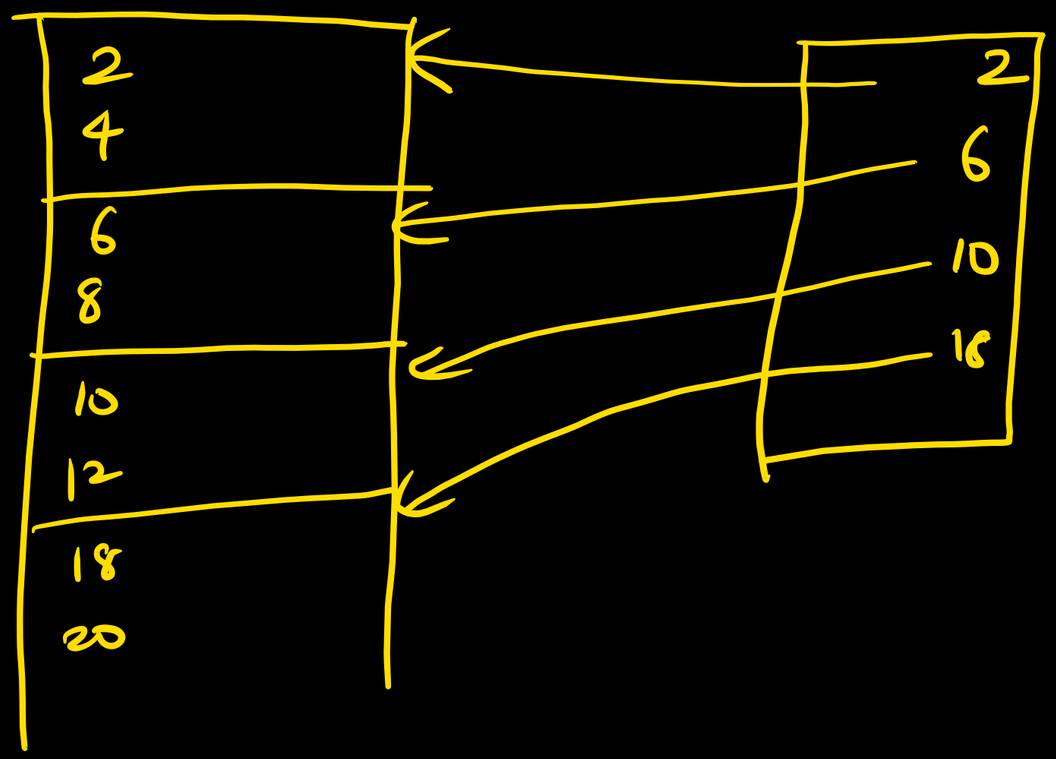



Dense index on unordered field.

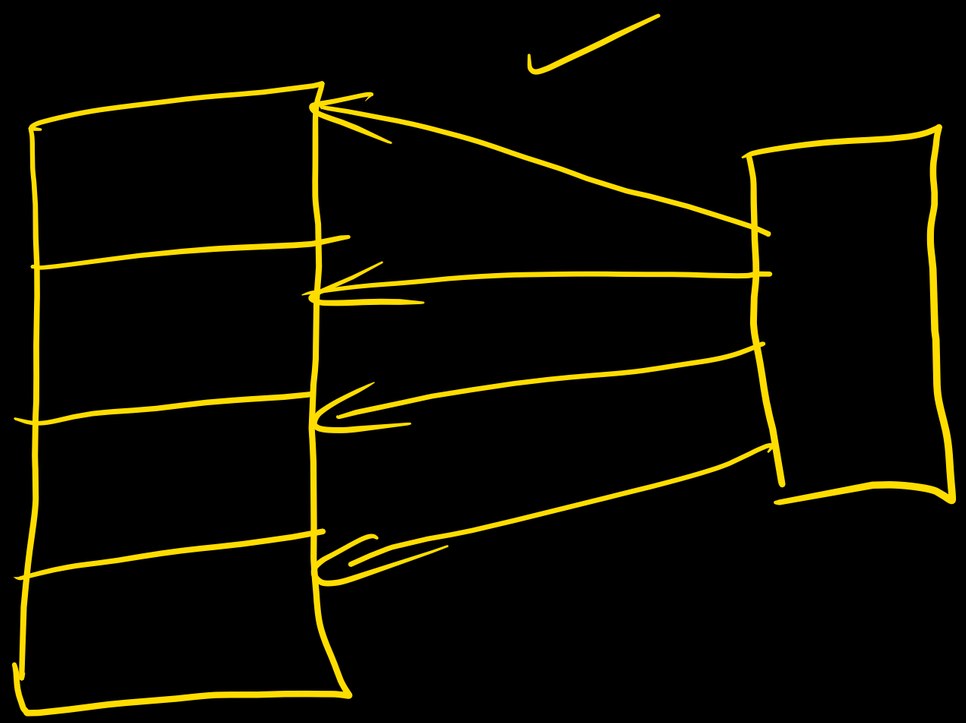
## Sparse Index:

Fd a set of DB records there exists an entry in index file

Sparse index is possible only on ordered fields.

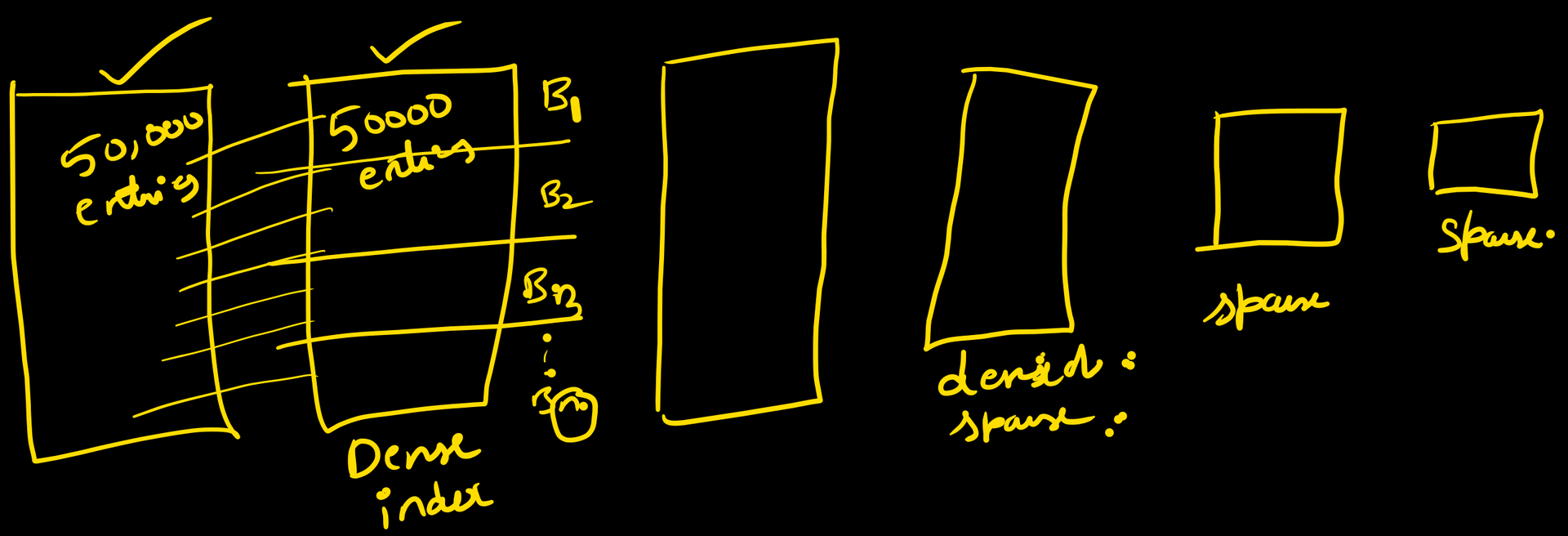


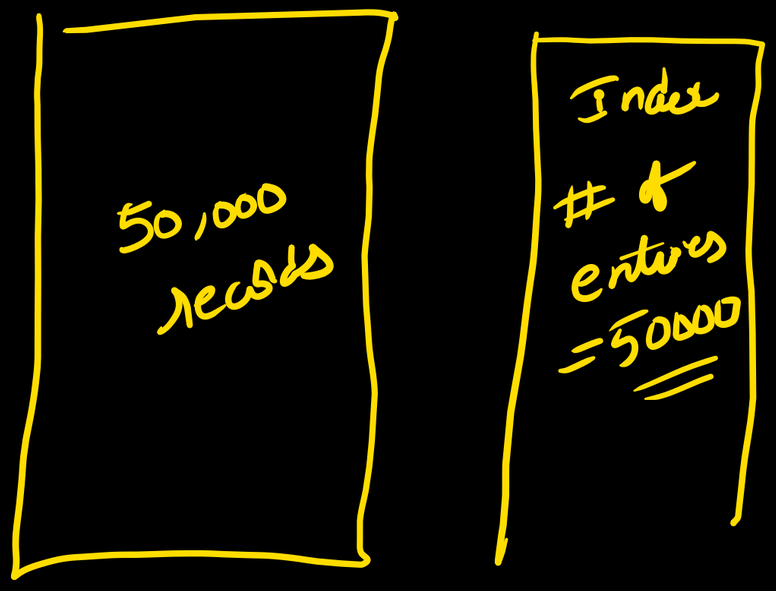
If sparse index is build on ordered field which is a key (Candidate key) then # no of index file entries = # blocks of DB file.



→ No of records in database table are 50000; block = 1024 B,
Search key = 12 bytes, Record = 100 bytes, pointer size = 6 bytes :

1) If dense index is built :
   a) ~~No~~ No of I level dense index blocks / level indexes $\lceil \log_2 \rceil$
   b) ~~No of~~ I/O cost using 1






Index blocks per file $= \frac{B-H}{K+P} = \left\lfloor \frac{1024}{12+6} \right\rfloor$

$= 56$ entries

Each block → 56 entries

1 entry → $\frac{1}{56}$ block

50000 entries → $\left\lceil \frac{50000}{56} \right\rceil$ blocks

$= 893$ blocks

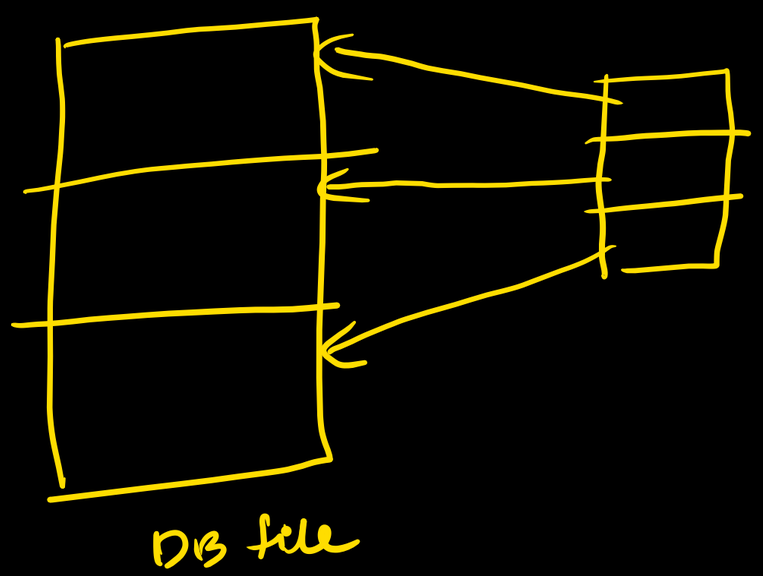
I/O cost using 1 level index $= \lceil \log_2 893 \rceil + 1$

→ No of records in database table are 50000, block = 1024 B, search key = 12 B, record = 100 B, pointer size = 6 bytes

If sparse index is built

a) No of 1st level sparse blocks?

b) I/O cost using one level indexing




Sparse index there will one entry per block

BF of DB table

$$\left\lfloor \frac{B-H}{R} \right\rfloor$$

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

10 Records per blocks

1 B → 10 Records

10 Rec → 1 B

1 Rec → $\frac{1}{10}$ B.

50000 → $\frac{50000}{10}$ Blocks

$= \underbrace{5000}$ Blocks.

∴ we need $\underbrace{5000}$ entries in sparse index.

BF index file = $\left\lfloor \frac{B-H}{k+p} \right\rfloor = \left\lfloor \frac{1024}{18} \right\rfloor = \underbrace{56}$ ✓

∴ no of blocks = $\left\lceil \frac{5000}{56} \right\rceil = 90$ blocks.

I/o cost $\lceil \log_2 90 \rceil + 1$.