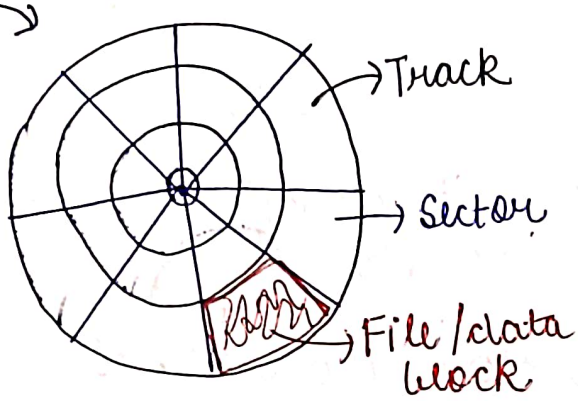
# Ram Vs Hard Disk

- Ram
  - Temporary (Volatile)
  - expensive
  - limited ☹
- Hard Disk
  - Persistence
  - Cheap
  - Take more time (I/O operation)

HDD →




→ Intersection of Track & Sector give us a file block. (tipically 4KB)

- Whenever we need to process any data we need to take the entire file block from Hard disk to RAM. (because C.P.U. can access data stored in RAM).

Time taken to find in 1 million Records. (Rough values)

Let say each record is an entry which takes 400 bytes.

In general

1 Block ⇒ 4KB data (can store)

$\Rightarrow \frac{4000}{400}$ = 10 records/entry (1 block can store)

[Let say any I/O operation in HDD takes 1 milli seconds.]

Now 1 million records,

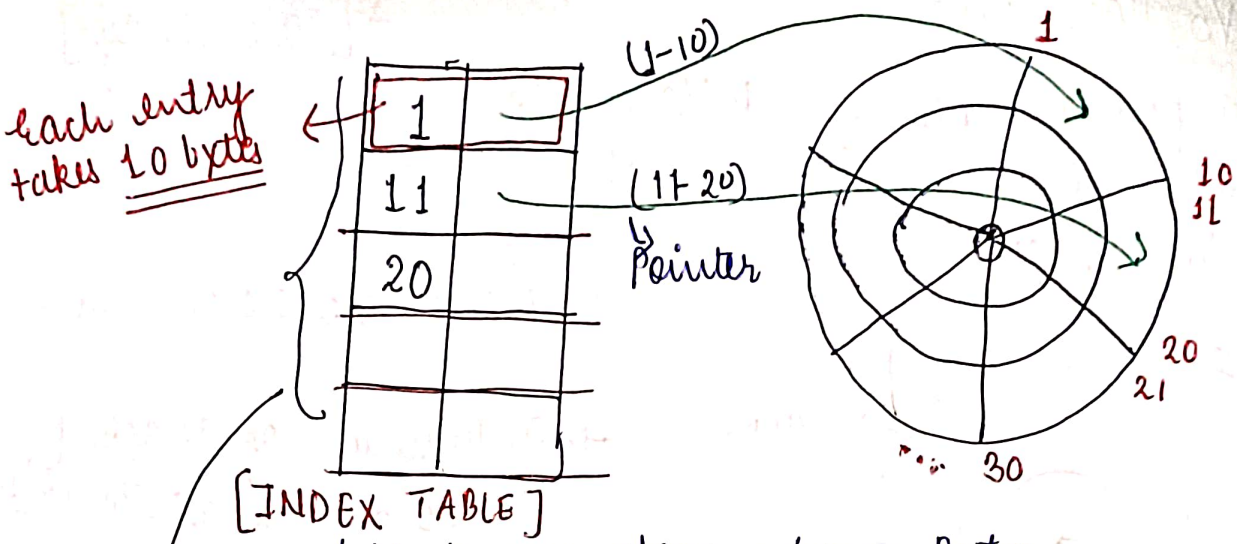
$10^6$ Records ⇒

No. of blocks ⇒ $\frac{10^6}{10} = 10^5$ blocks

$= 10^5$ ms

= 100 sec (Huge)

(We should make it better)




1 block can store = 4000 Bytes

Each entry takes 10 bytes

Each block can store 400 entries of table.

→ 105 entries [ because the table needs to store pointer to each of the data block ]

How many blocks are needed to store this table ?

No. of blocks ⇒ $\frac{105}{400}$

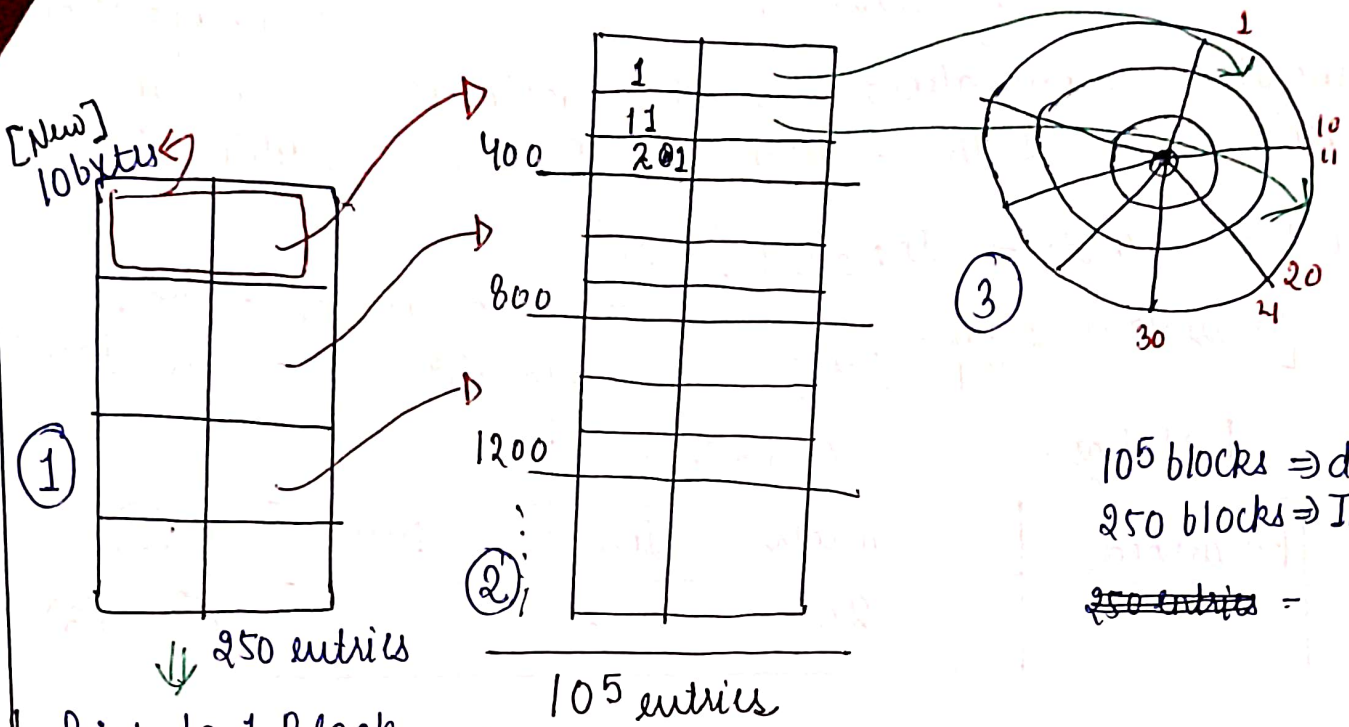
⇒ 250 blocks of disk.

And, another 1 ms to read the corresponding data block (/file block).

⇒ [250ms] (0.25 seconds)

[optimized]

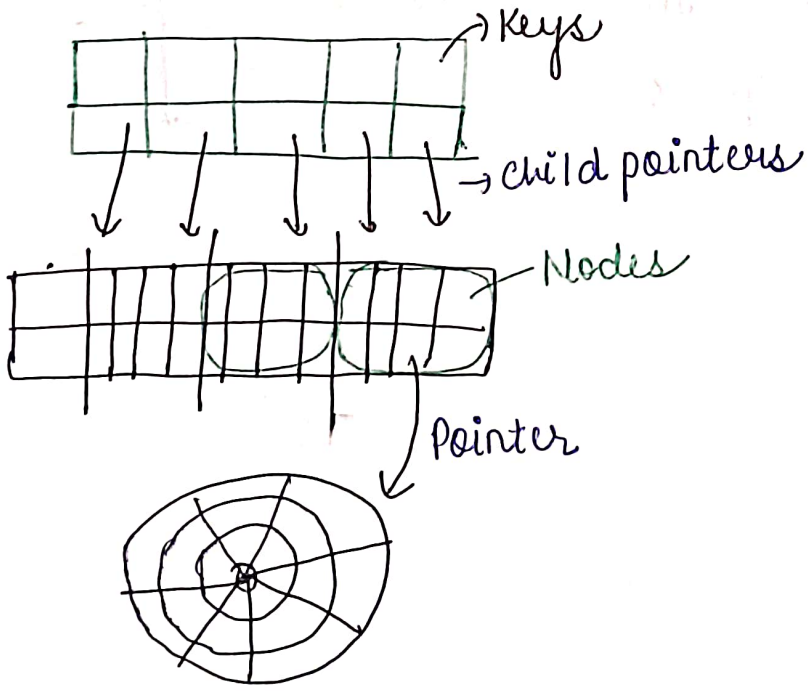
Can we optimized More ? ☺




250 entries X 10 bytes = 2500 Bytes [$< 4$ KB]

3 I/p operation taking 3MS [0.003 seconds] ☺

{MULTI-LEVEL INDEXING}




This is how the B-trees look.

{Search Insert Delete (Most common DB operation)

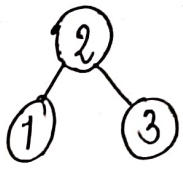
If we dont know about BTree then Balanced Tree is going to come as answer (Logarithmic time to search, delete, insert).

1 million records ⇒ $\log_2(10^6) = 20$

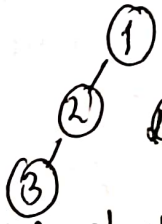
⇒ 20 I/O operations with Hard disk for search, Insert, & delete

NOTE:- [Every I/O operation is expensive.]

| Database | Operations | | |
|---|---|---|---|
| | Search | Insert | Delete |
| Balanced BST | 20 | 20 | 20 |




| | | | |
|---|---|---|---|
| Unbalanced BST | Search | Insert | Delete |
| | $10^6$ | $10^6$ | $10^6$ |




(linearly)

Tricky! → shifting of element

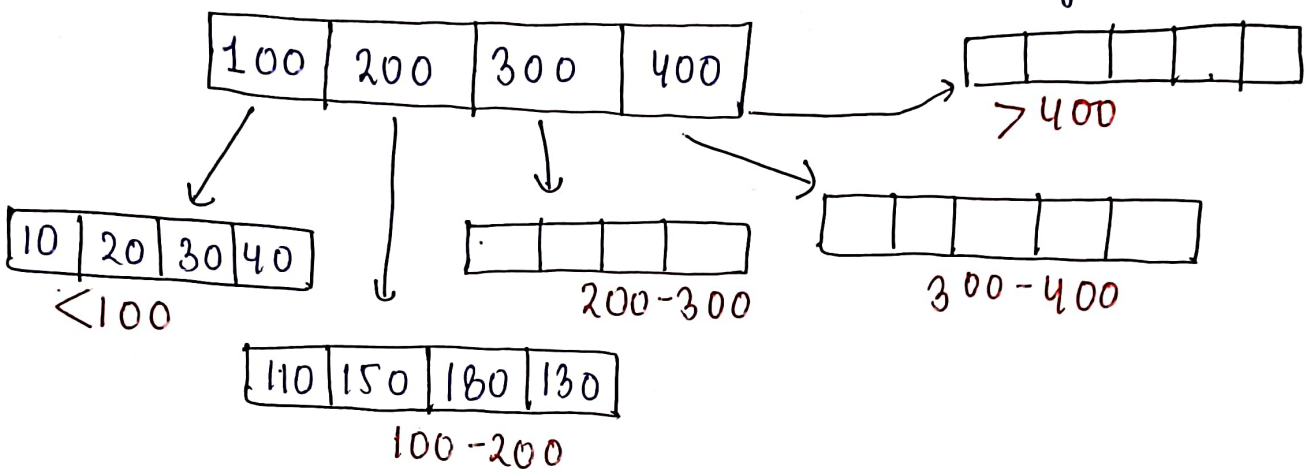
| | | | |
|---|---|---|---|
| Sorted Array | Search (log time) | Insert | Delete |
| | 20 | $10^6$ | $10^6$ |

| | | | |
|---|---|---|---|
| B-Tree (m-way Tree) | Search | Insert | Delete |
| | 3 | 3 | 3 |

Order ⇒ m
max children ⇒ m
⇓
complexity $\log_m(10^6)$
(Let m = 100)
⇒ $\log_{100}(10^6) = 3$

# B-Tree

↳ {A node can have multiple key. }




Even though here we have to do more no. of comparision for every node than (BST).

But, The cost of getting a data from DISK TO RAM [ which is more expensive than the no. of comparision in RAM].

Main problem!

- Major characteristics of B-Tree :-
  - B Tree ⇒ Order m
  - → Max. no of children = m
  - → Max. no of Keys = (m-1)
  - → Min. no of Keys (other than root node) = $\lceil n/2 \rceil$ (Round off to next bigger integer.)
    - ↳ root can have min 1 key.
  - → All leaves are at same level
  - → $[height = \log_m n]$ (n → Keys)
  - → We grow towards Root