Indexing

TABLE OF CONTENTS

24th Hard day challenge :

1. Assignments + Revision (MCQs)
2. Backlog (Assignments of prev. session)
3. Additional Questions

Intro to Indexing

- **What is the expected TC for the following queries?**

Query-1 :

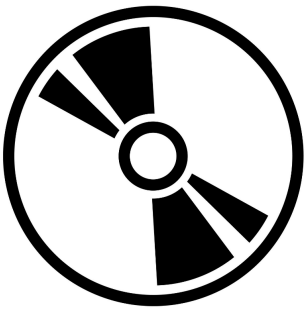
```
SELECT  *
FROM    students;
```

Query-2 :

```
SELECT  *
FROM    students
WHERE   id  =  100;
```

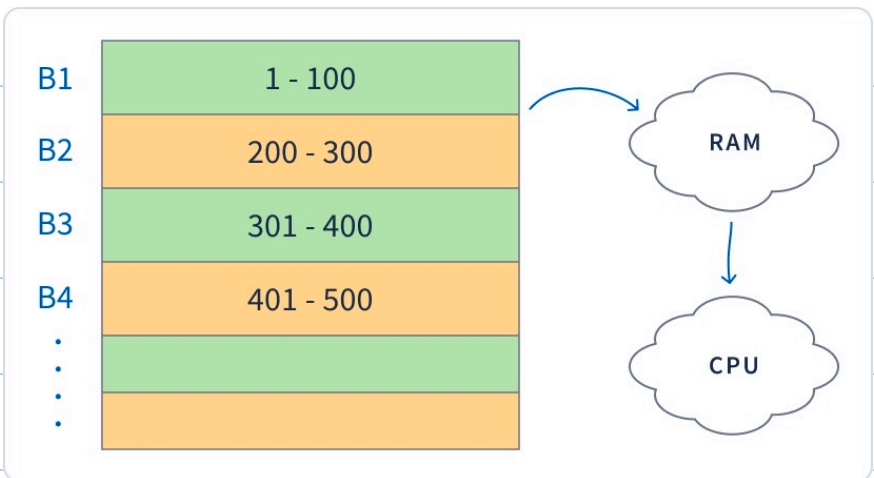
TC = O(N)

- **Where is all the data getting stored?**



Query-3 : SELECT *
FROM students
WHERE id = 500;

- **The data of id = 500 is being stored in disk in a memory block as shown below :**



- **Hash Map**

id	block_address
1	B1
2	B1
.	.
.	.
.	.
500	B4
.	.
1050	B10

What is the TC here?

→ We can store indexes using Hashmap which takes O(1) to search.

- **How do we create indexing for following query cases?**

Query :

```
SELECT  *
FROM    students
WHERE   psp = 80;
```

psp	memory block
1	[B1, B5...]
.	.
.	.
80	[B2, B4..]

- Indexes makes query lightning fast

- Using indexes we can increase performance of our DB

Range Queries

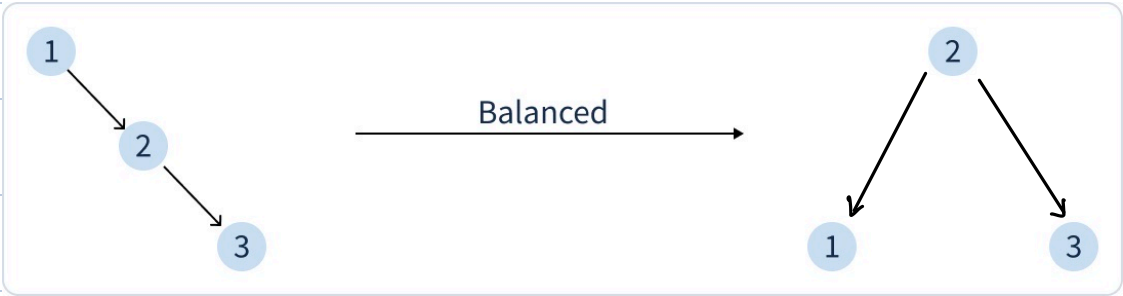
```
SELECT  *
FROM    students
WHERE   psp between 40.2 and 60.5;
```

many many psp in this range

' *Can hash map work on range queries like above one?*

- If we use hash map to store indexing then we might need to exclusively check for every possible values in a range.

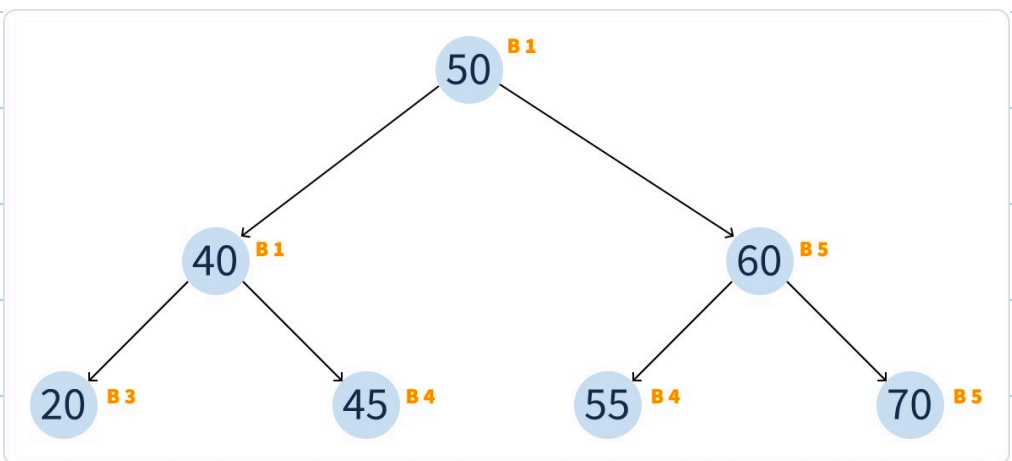
Tree Map (Self Balancing) (BBST)



' What is TC here? '

$TC = O(\log N)$

Example

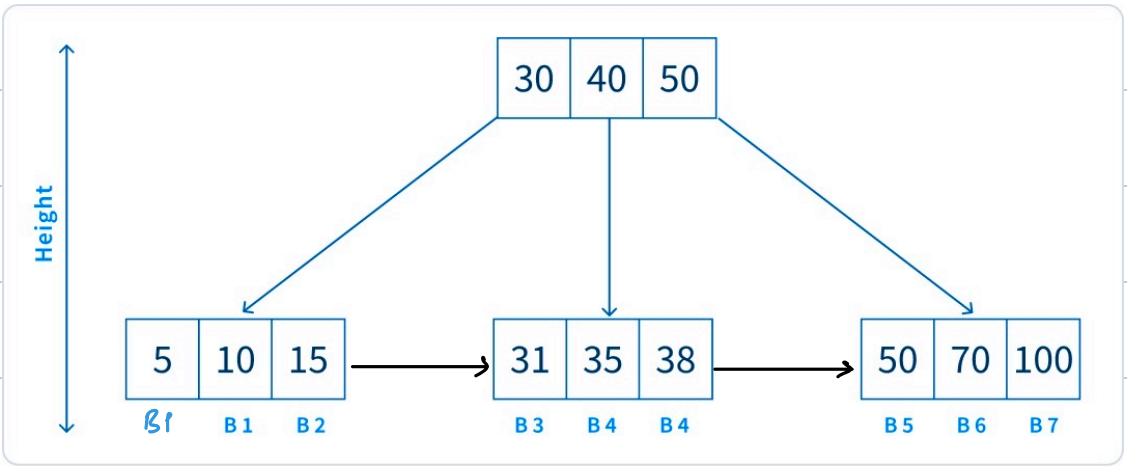


- Using tree maps we can use range queries easily.

B Trees, B+ Trees

- We can have multiple children for every node.

0 - 100



TC = O (lg N)

- This is actual DS used by SQL.
- since we can have multiple children for every node, the overall height of B+ Tees < Tree maps.
- TC = O (H) < Tree map.

→ By default indexing on PK : Clustered index

→ Custom index on any other column : Non - clustered index

Cons of Indexing

' Where are these indexes getting stored? '

- Extra space required to store them.
- Write write / update / delete operations will be slower.

→ Cardinality : No. of unique values

→ Selectivity : $\frac{\text{No. of unique values}}{\text{Total no. of values}}$ (0 - 1)

1 Billion rows

id	Name	Drinks
		Tea
		Coffee
		Tea
		⋮

⇒ $\frac{2}{100\ldots}$ ⇒ ~ 0

Indexing on multiple column

| id | name | psp | attendance |

Indexing on Strings

- Let's say at Scaler we do a lot of queries email_id column like :

```
SELECT    *
FROM      students
WHERE     email  =  "Naveen@gmail.com"
```

' Do we create index on full email ? '

* Whenever we create indexing, we create it on a part of string.
  → 4-6 chr.

"abc@gmail.com"

"Rahul@scaler.com"

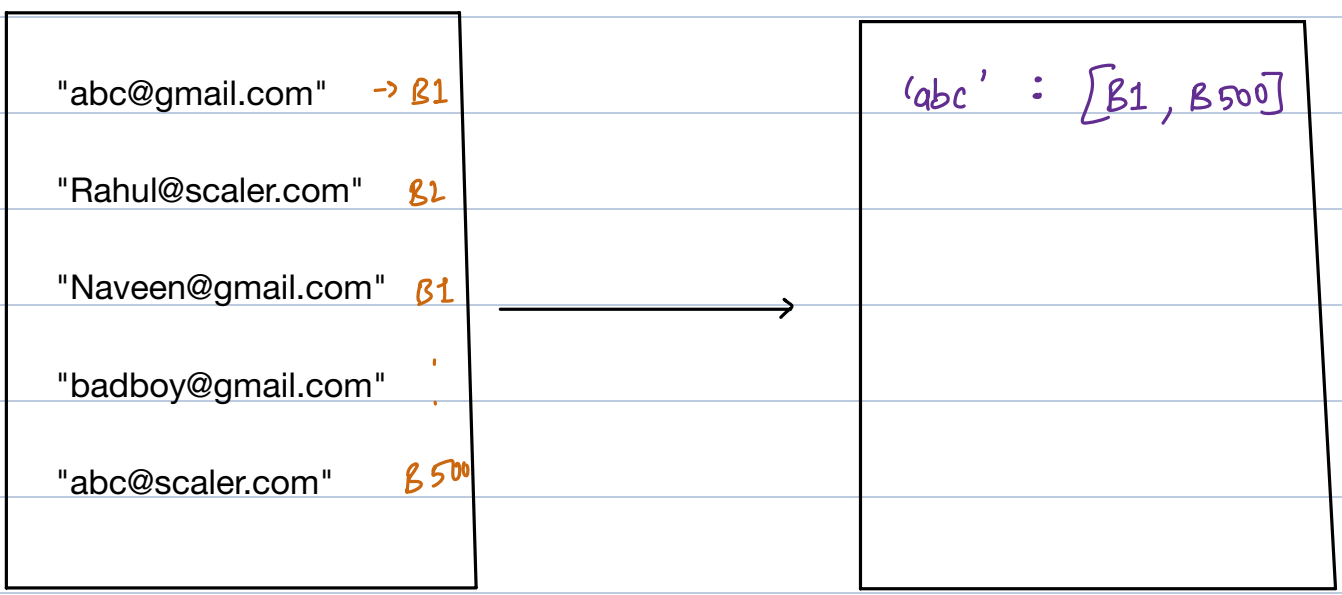
"Naveen@gmail.com"

"badboy@gmail.com"

"abc@scaler.com"

' Can we create index on some part of this string ? '

- Creating index on part of a string will reduce space required



* full text Indexing → HW

1. Clustered Indexing

- Table have default indexing on PK.
- Clustered.

2. Non - clustered Indexing

- Indexing created by us.