INDEXING.

⇒ Database
- RAM ✗ ⇒ Volatile.
- HDD ✓ ⇒ Permanent Storage.

Students ⇒ HDD

id	name	batch_id	psp
1	X	3	80
2	Y	1	90
3	A	2	75.6
4	M	2	81.8
5	N	3	90.4
6	B	1	50.7

```
Select *
from students
where batch_id = 3 ;
```

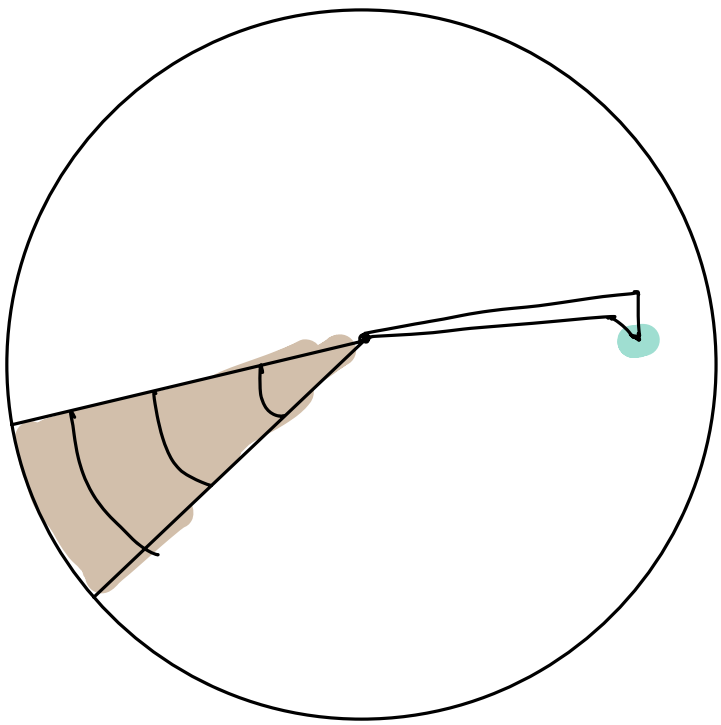
→ CPU can't directly fetch the data from HDD.

→ Data from HDD is brought to the RAM and then CPU will read the data from there.

Students ⇒ HDD

id	name	batch_id	psp	
1	X	3	80	→ B1
2	Y	1	90	
3	A	2	75.6	→ B2
4	M	2	81.8	
5	N	3	90.4	→ B3
6	B	1	50.7	

N rows



⇒ To get all the students from batch_id = 3, We'll have to go through all the blocks one by one & check in the block if there's a student with batch_id = 3 inside that block.

↳ O(N)

```
Select *
from Students
where id = 4;
```

id → Primary Key

↓

Sorted based on PK.

→ Unique.

⇒ Iterate the table row by row & get the student with id = 4, we need not to go to the rows after id = 4.

Students

id	name	batch-id	psp	
1	X	3	80	→ B1
2	Y	1	90	
3	A	2	75.6	→ B2
4	M	2	81.8	
5	N	3	90.4	→ B3
6	B	1	50.7	

```
Select *
from Students
where id = 4;
```

Index table.

Block No	Address
B_1	Ad1
B_2	Ad_2
B_3	Ad_3
.	.
.	.
.	.
.	.
.	.
.	.

PK

Students

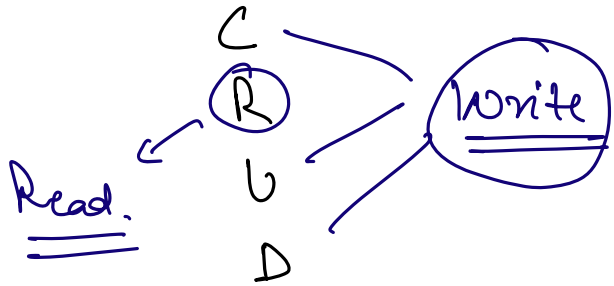
id	name	batch-id	psp
1	X	3	80
2	Y	1	90
3	A	2	75.6
4	M	2	81.8
5	N	3	90.4
6	B	1	50.7
7			

AD6

Index table

1	AD1
2	AD2
3	AD3
4	AD4
.	.
6	AD6
.	

```
Select *
from students
where id = 6;
```



#

```
Select *
from students
where batch_id = 3 ;
```

Index

bid	
1	—
1	—
1	—
1	—
1	—

Indexing

→ Makes our queries faster by reducing the no. of disk accesses.

→ For any Create | Update or Delete operation on original table, Index table needs to be updated.

→ Indexing makes writes (C|U|D) slower.

Note

→ Create index if it is actually required.

→ Don't create index at the time of table creation, create index based on access pattern.

type of queries we are getting

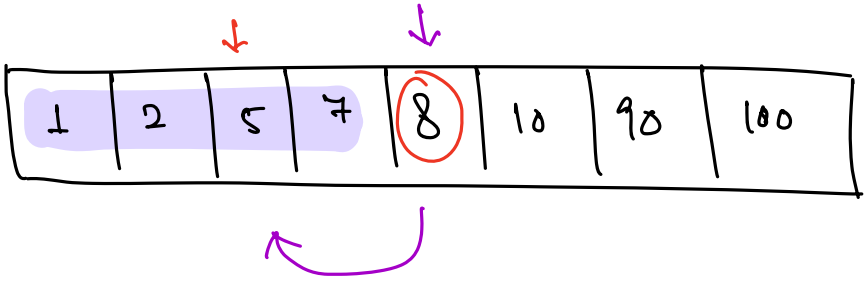
```
show indexes from film;

explain select * from film where length = 105;

create index idx_film_length on film(length); -- creating the index

drop index idx_film_length on film; -- deleting the index.

explain select * from film where film_id = 105;

explain select * from film where rental_duration = 3;
```

Index table is also sorted.

index table

1	—
2	—
3	—
4	—
5	—
6	—

Searching in a non sorted Array ⇒ O(N).



②

$N \rightarrow \frac{N}{2} \rightarrow \frac{N}{4} \rightarrow$ ---- ①

⇒ Binary Search. : O(logN).

Index table : B | B+ Trees.

↳ O(logN)

Indexing on Strings.

Students

id	name	batch_id	psp
1	Sushant	3	80
2	Mukta	1	90
3	Murli	2	75.6
4	Abhigyan	2	81.8
5	Sai	3	90.4
6	Shlok	1	50.7

```
Select *
from Student
where name = 'Abhigyan';
```

⇒ Full Table Scan.

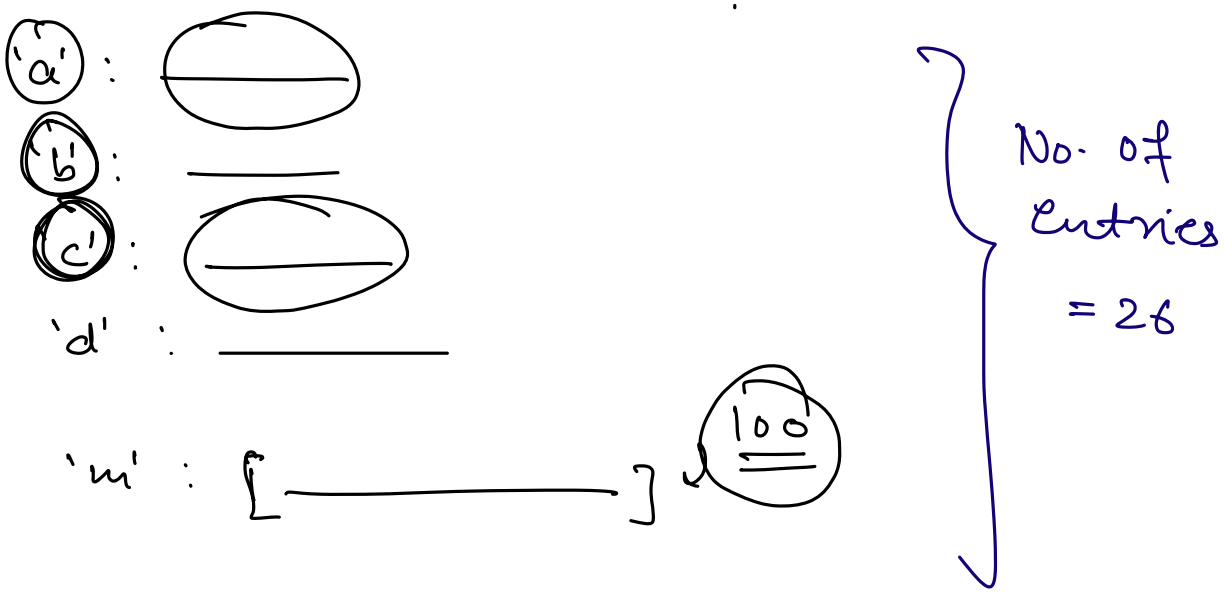
Create index on name column.

Sushant	—
Mukta	—
Murli	—
Deepak	—
Shlok	—

→ Index on name column will create a very big table, which will consume a lot of space

⇒ Index table is persisted permanently inside the HDD & there will be a copy of index table inside the RAM.

Index of first character only =0



Index of first 2 characters

'aa'
'ab'
:
:
'mm' : [- - - - -] 5

No. of Entries = 26 x 26

```
select * from city;

explain select * from city where city = 'Goa';

create index idx_city on city(city(3));

drop index idx_city on city;

show indexes from city;

-- without index -> 600
-- with index on 1 character -> 13
-- with index on 2 characters -> 7
-- with index on 3 characters -> 4
-- with index on 4 characters -> 1
-- with index on complete city column -> 1
```