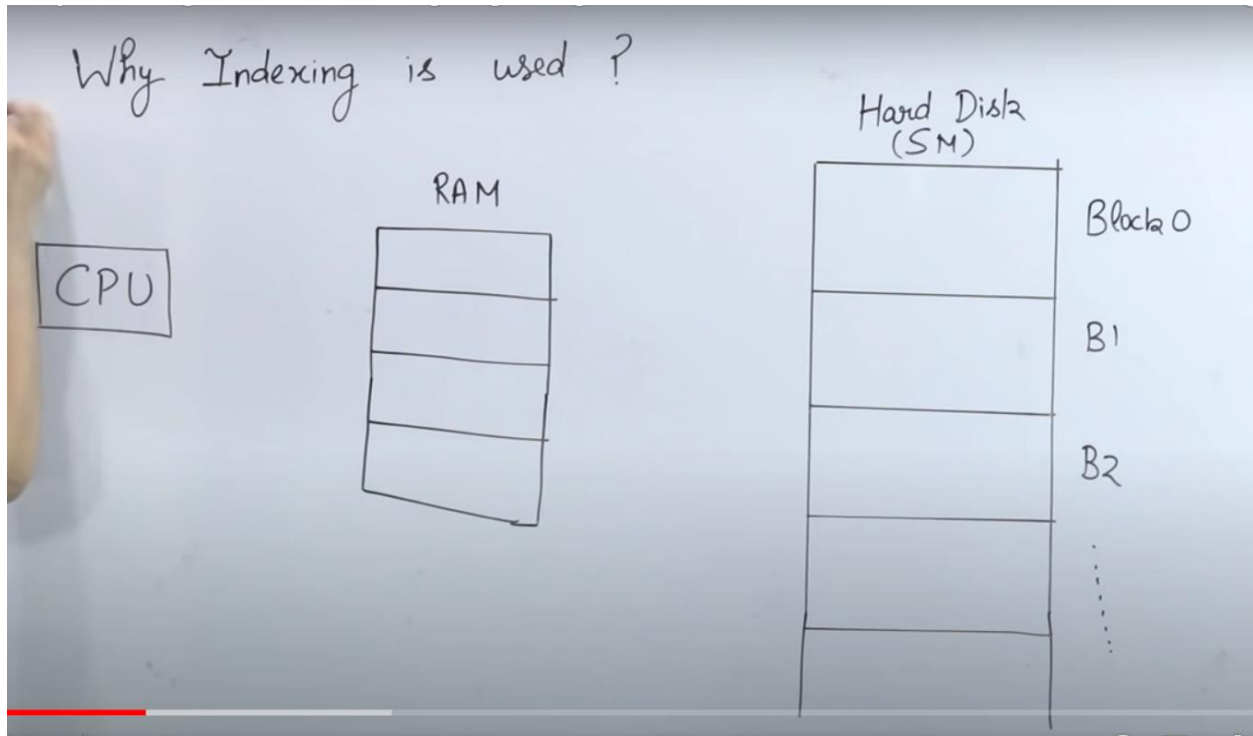


# INDEXING

## GATE SMASHERS

### LEC 93: Why Indexing is used | Indexing Beginning | DBMS



CPU speed is MIPS (Million Instruction Per Second)

Hard Disk is slow, RAM is fast. Hard disk is divided into blocks and each block is loaded into RAM and CPU communicates with RAM. If CPU needs to show the result of below query:  
`select * from somerandomtable where somefield='value';`

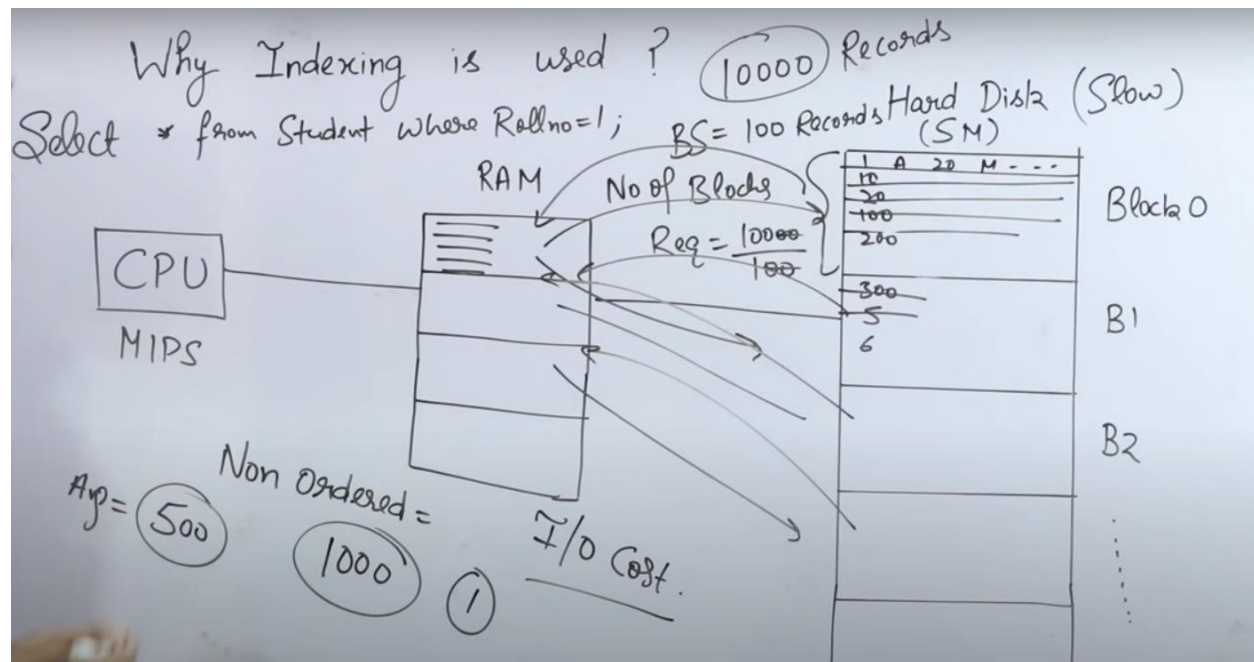
First the instruction would be given to CPU, and it would pass it to RAM and then RAM will take blocks one by one each from Hard Disk and try to find the above query results. The problem lies in the **I/O cost** which is the cost of CPU to bring each block into its own memory.

Lets suppose we have 10000 records in our somerandomtable and saved on the hard disk and block size of each block is 100 means each block can store upto 1000 records. Lets find out the number of blocks required to store 10000 records:

No. of Blocks =  $10000/100 = 100$

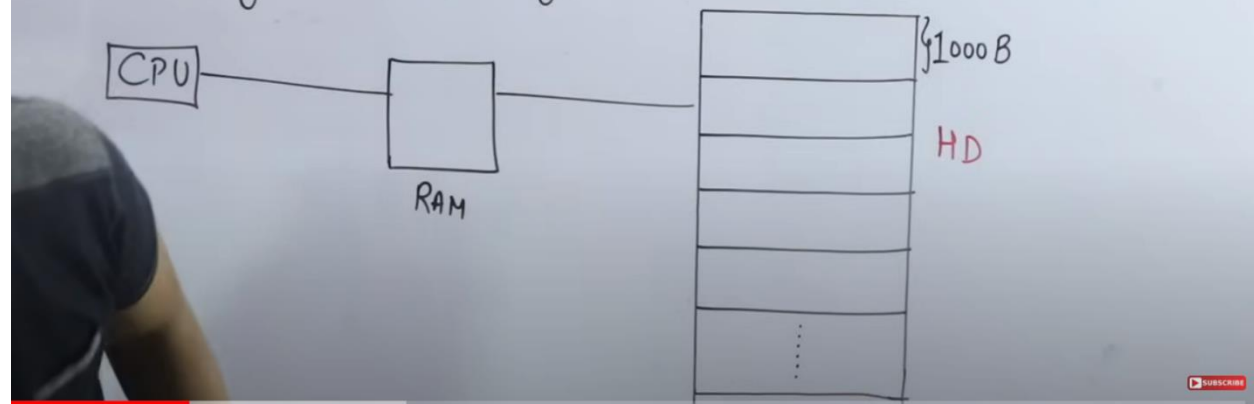
Now lets suppose data is in random order( **unsorted** ) in this case we need to bring 100 (total) blocks in memory one by one in worst case to search a single record.

# INDEXING



## Lec-94: Numerical Example on I/O Cost in Indexing | Part-1 | DBMS

Consider a Hard Disk in which Block Size = 1000 Bytes, each record is of Size = 250 Bytes. If total no of records are 10000 and the data entered in Hard disk without any order (Unordered) what is avg time Complexity to search a record from HD?



No. of Records we can put in each block =  $1000B / 250B = 4$

No. of Blocks required to store the 10000 records =  $10000 / 4 = 25000$

# INDEXING

Time to search a record in hard disk blocks:

Best case = 1 block means we found record in first block

Worst case = 2500 means we found out record in last block = N

Average case =  $2500/2 = 1250 = N/2$

If data is sorted then  $\lg n$ , in our case  $\lg_2 2500 = 12$

Lec-94: Numerical Example on I/O Cost in Indexing | Part-1 | DBMS

Consider a Hard Disk in which Block Size = 1000 Bytes, each record is of Size = 250 Bytes. If total no of records are 10000 and the data entered in Hard disk without any order (Unordered) (Ordered) what is avg time Complexity to search a record from HD?

Select \*  
Student  
Rollno=345  
No of L

CPU  $\log_2 N$   $\log_2 2500 = 12$  RAM  $\log_2 2500 = 12$

every block =  $\frac{1000B}{250B} = 4$   
No of Records we can put =  $\frac{10000}{4} = 2500$

HD  $\log_2 N$   $\log_2 2500 = 12$

I/O Cost =  
Best case = 1  
Worst case = 2500 = N  
Avg =  $\frac{2500}{2} = N/2 = 1250$

The diagram illustrates the I/O cost of searching for a record in a hard disk. It shows a CPU connected to RAM, which is connected to a Hard Disk (HD). The HD is represented as a stack of blocks. The diagram includes calculations for the number of records per block (4) and the total number of blocks (2500). It also shows the I/O cost for best, worst, and average cases.

## Lec-95: Numerical Example on I/O Cost in Indexing | Part 2 | DBMS

# INDEXING

Numerical Example on I/O Cost in Indexing | Part 2 | DBMS

Consider a Hard Disk in which Block Size = 1000 Bytes, each record is of Size = 250 Bytes. If total no of records are 10000 and the data entered in Hard disk without any order (Unordered) (Ordered) what is avg time Complexity to search a record from Index table if Index table entry = 20B (Key + Pointer) 10B 10B

The diagram illustrates a hard disk layout with blocks of 1000 bytes each. A record size of 250 bytes is indicated. An index table is shown with columns for key and pointer, each 10 bytes.

Index table: a table which contains keys and pointers

## Lec-96: Types Of Indexes | Most Important Video on Indexing

Ggfff

Types of Indexes

1) Primary  
2) Clustered  
3) Secondary

Ordered file  
Unordered file

Primary Index	Clustered Index
Secondary Index	Secondary Index
Key	Non Key

Here Key means some unique attribute and non key means no unique attribute.

In Oracle and sql Server if we create a primary key, primary index is automatically created.

Examples

## INDEXING

Ordered file	Primary Index	Clustered Index
Unordered file	Secondary Index	Secondary Index

1 10  
2 9  
3 8  
4 7  
5 6  
...

Ordered and Key

Key

1  
2  
3  
4  
5  
6  
...

Ordered and Non key

Non Key

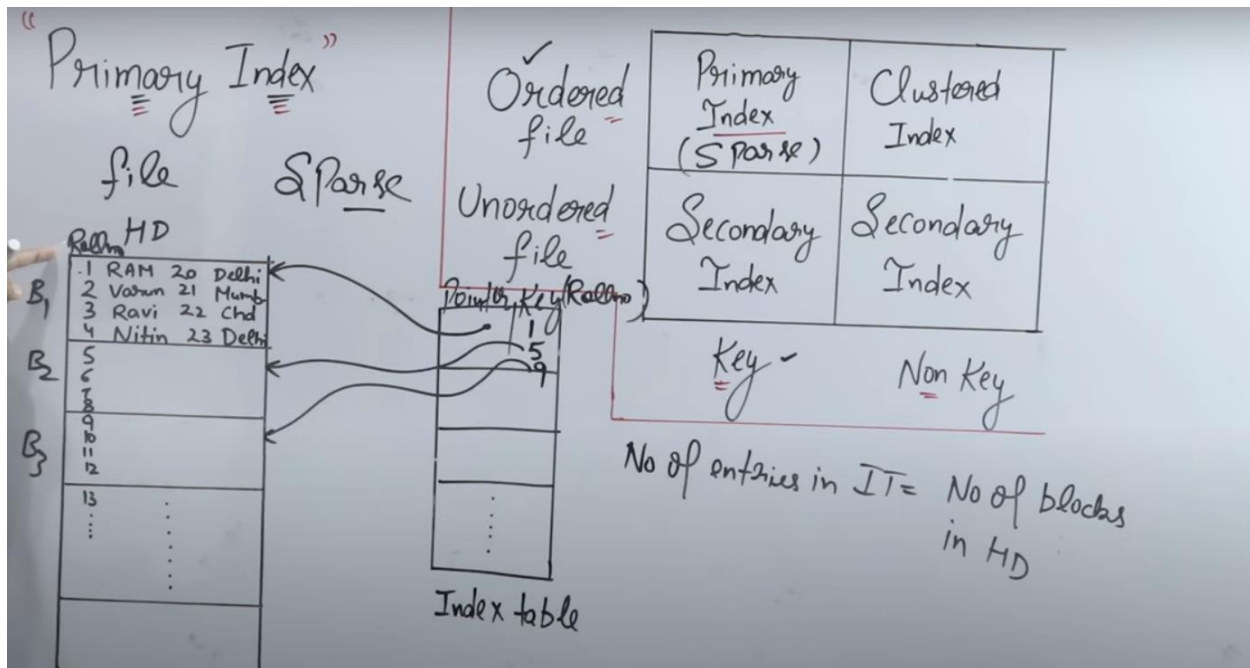
1  
2  
2  
2  
3  
3  
...

Unordered and key

2  
5  
2  
3  
9  
8  
2  
5  
6  
...

Unordered and no key

# INDEXING

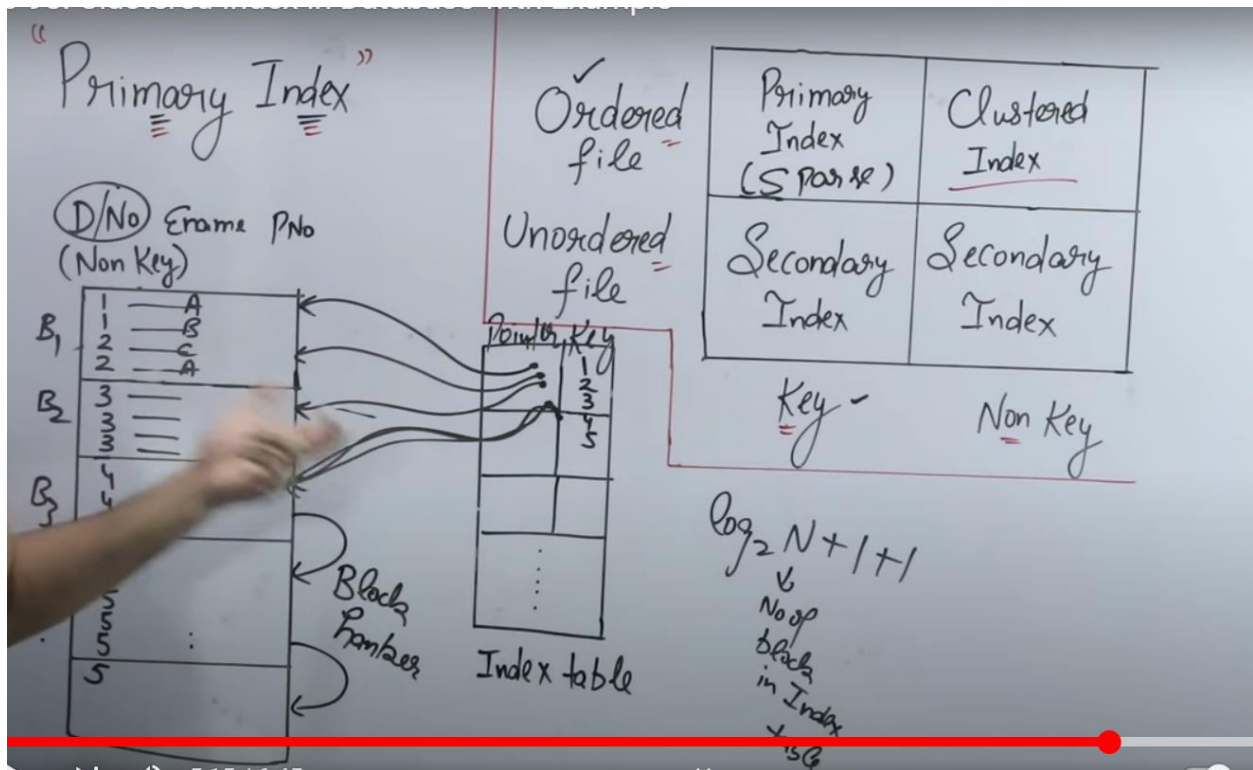


For each Hard Disk block, one pointer is added in the Index table to point to that specific block.

## Lec-98: Clustered Index in Database with Example



# INDEXING



Block Hanger is used to tell that the searched value might be present in the next block.

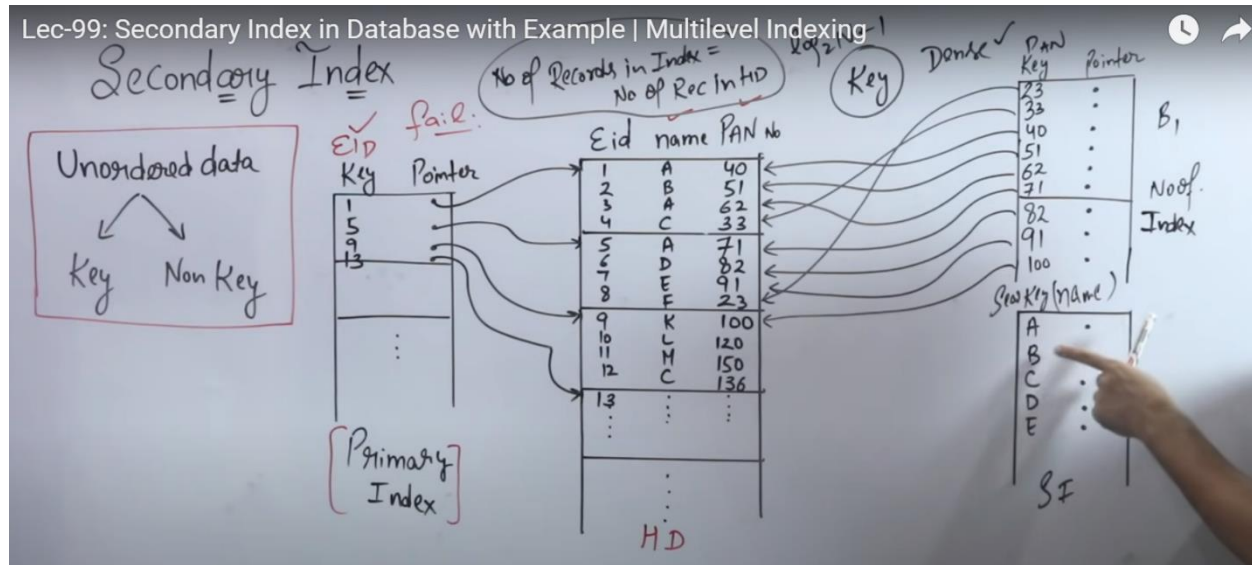
Time complexity to search in clustered index is  $\lg_2 N + 1$  (into the hard disk block) + c (might be present in the next blocks as indicated by Block hanger)

## Lec-99: Secondary Index in Database with Example | Multilevel Indexing

Secondary index is the second index we use although we do have a primary index, but when the user want to access non key values like name in any table which is usually not unique then we use the secondary index.

There are two cases, first one is we have a unique key so we created a primary index, but now we need to search the table based on another candidate key now we can create a index table of all the entries in the original table. We basically will store all the pointers of each entry in index table.

# INDEXING



Second case is when we have attribute which is non key. We will use Block of Record pointers in order to create an intermediate table to store the references of each block values in it. For pointing index table to intermediate table there would be a sparse table and for pointing intermediate table to hard disk, we need a dense table.

