Advanced Data Structures

Tree-based data structures

Today

Part 2 : B-tree Data Structure

- Agenda:
  1. How data is stored on disk?
  2. What is indexing
  3. M-way search trees
  4. What is B-trees?
  5. What is B^+-trees?

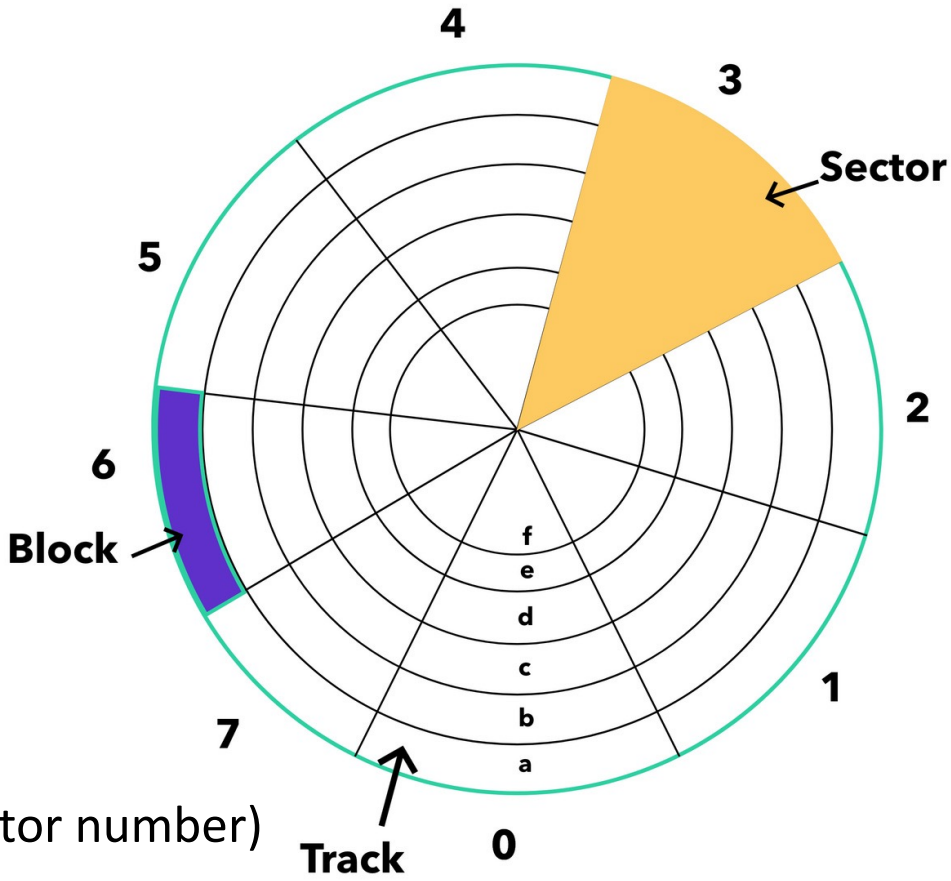
Data Layout on Disk

How data is stored on disk?

A: Track: one ring

B: Sector: one pie-shaped piece.

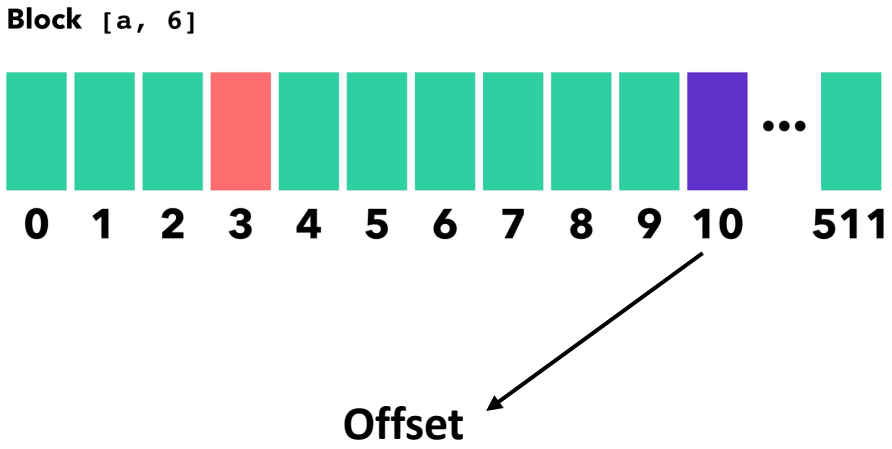
C: Block: intersection of a track and sector.

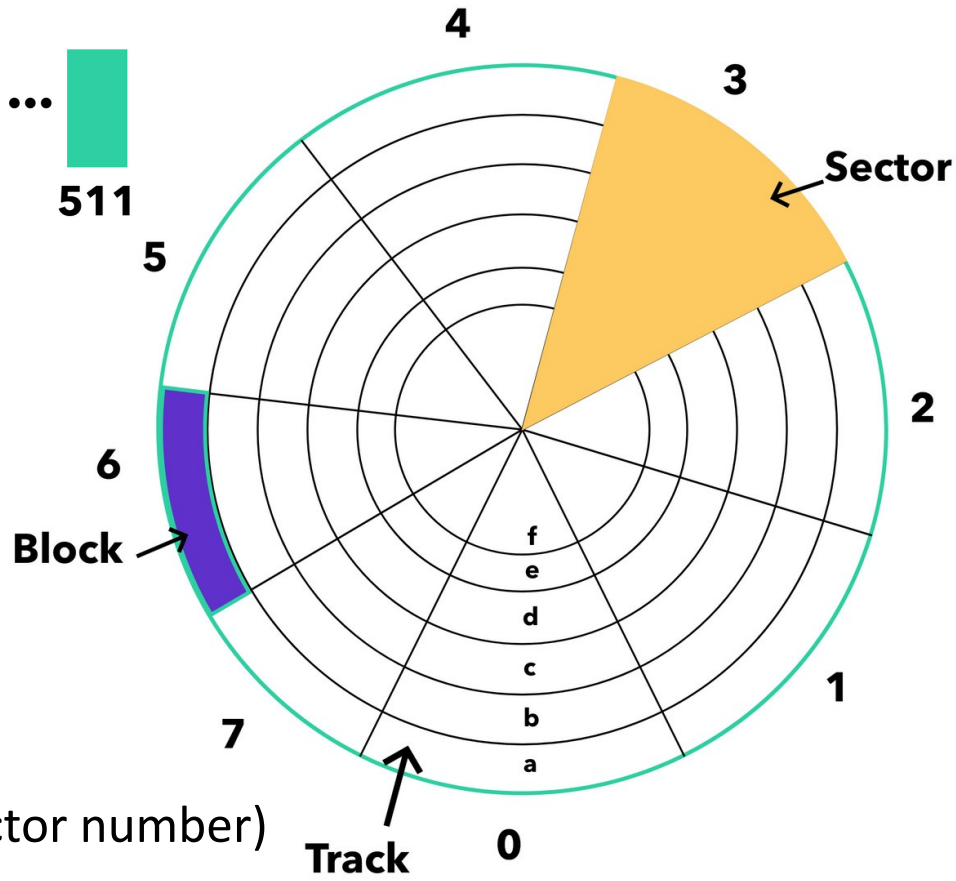


Block address = (Track number, Sector number)

Data Layout on Disk

How data is stored on disk?

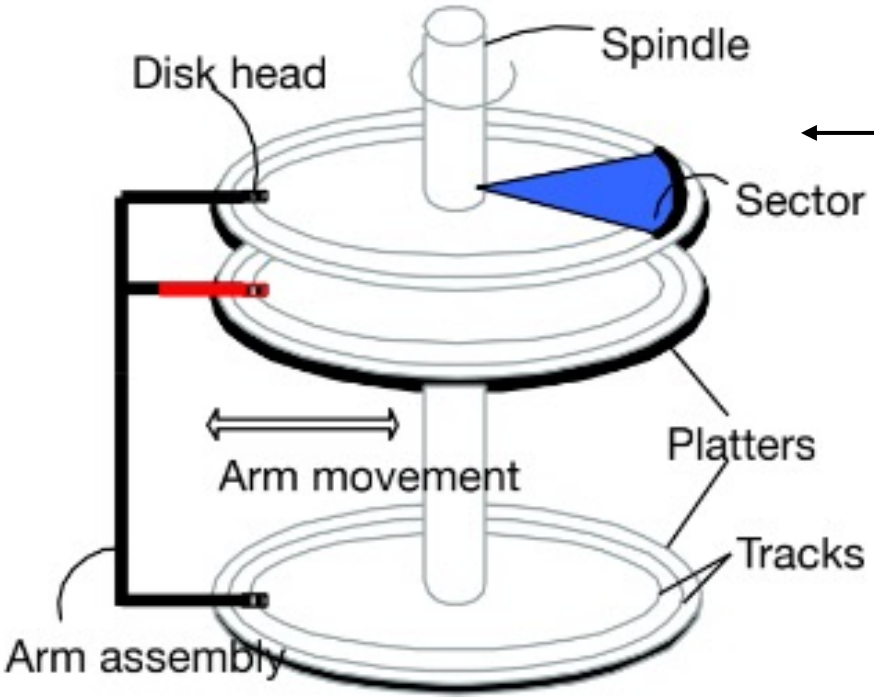


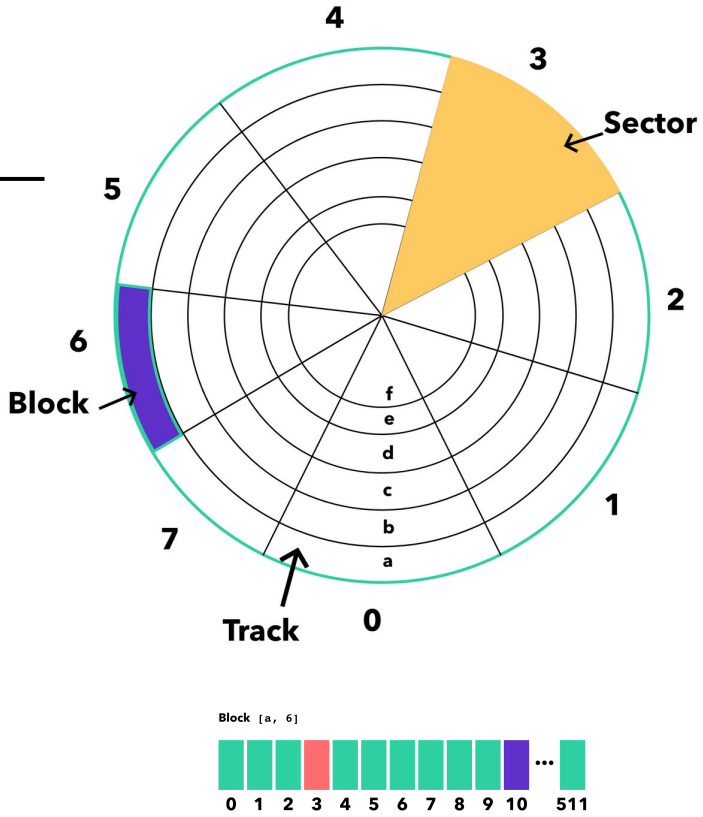


Block address = (Track number, Sector number)

Data Layout on Disk

How data is stored on disk?



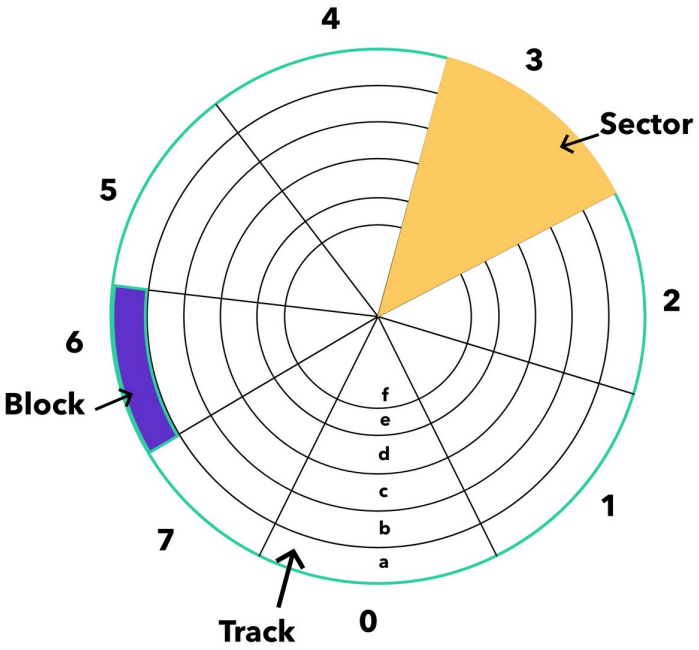


Data Layout on Disk

How data is stored on disk?

Main memory : RAM

Data



Data Layout on Disk

How data is stored on disk?

	A	B	C	D	E	F	G
1	**Employee Database 2019**						
2	Employee ID	First Name	Last Name	Designation	Salary	Gender	
3	D01	William	Smith	CEO	250000	Male	
4	D02	Rezza	Fredd	Sr Manager	100000	Female	
5	D03	Mohan	Kumar	Web Develpoer	50000	male	
6	D04	Cindy	Mac	Senior HR	50000	Female	
7	D05	Vince	Jose	HR Consultant	30000	Male	
8	D06	Rajiv	Kumar	IT Admin	25000	Male	
9	D07	John	Thomas	Sr Technical Lead	45000	Male	
10	D08	Oshane	Mathew	Data Scientist	50000	Male	
11	D09	Chin	Yeu	Data Analyst	20000	Male	
12	D10	Linda	Greedman	Receptionist	15000	Female	
13							

Data Layout on Disk

How data is stored on disk?

Employee :

EID : 10

Name : 50

Destination : 10

Salary :50

Gender:8

_____

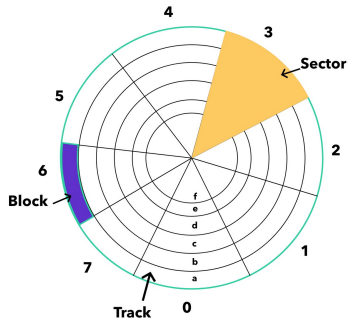
Size = 128 bytes

128 bytes

	Employee ID	First Name
2		
3	D01	William
4	D02	Rezza
5	D03	Mohan
6	D04	Cindy
7	D05	Vince
8	D06	Rajiv
9	D07	John
10	D08	Oshane
11	D09	Chin
12	D10	Linda
13	...	

...

100 records



Block size: 512 bytes

Data Layout on Disk

How data is stored on disk?

Employee :

EID : 10

Name : 50

Destination : 10

Salary :50

Gender:8

_____

Size = 128 bytes

records per block = 512 / 128 = 4

For 100 records ,

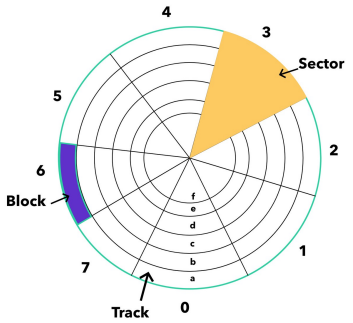
how many blocks? 100/4 = 25 blocks

	Employee ID	First Name
2		
3	D01	William
4	D02	Rezza
5	D03	Mohan
6	D04	Cindy
7	D05	Vince
8	D06	Rajiv
9	D07	John
10	D08	Oshane
11	D09	Chin
12	D10	Linda
13	...	

...

...

100 records



Block size: 512 bytes

Data Layout on Disk

What is indexing?

Employee :

EID : 10

Name : 50

Destination : 10

Salary :50

Gender:8

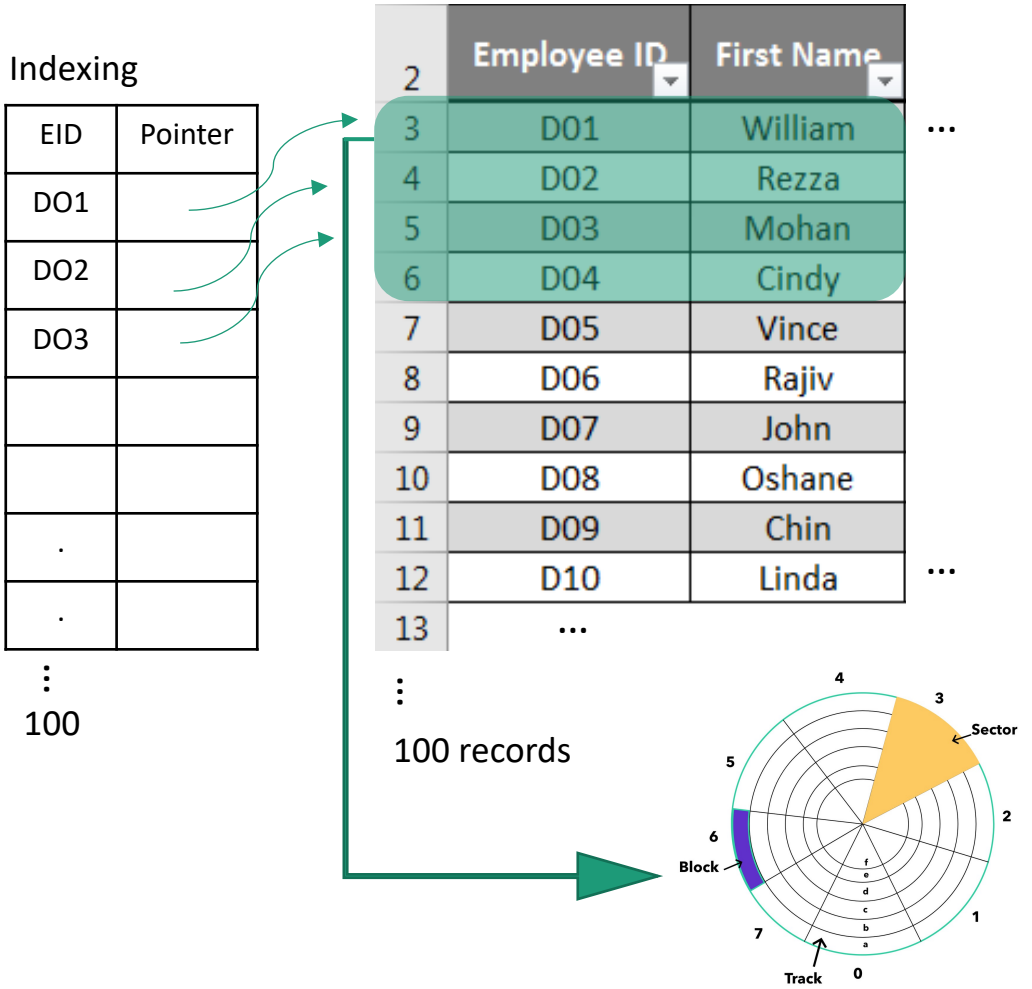
_____

Size = 128 bytes

records per block = 512 / 128 = 4

For 100 records ,

how many blocks? 100/4 = 25 blocks



Block size: 512 bytes

Data Layout on Disk

What is indexing?

Employee :

EID : 10

Name : 50

Destination : 10

Salary :50

Gender:8

Size = 128 bytes

EID : 10

Pointer : 6

Size = 16 bytes

records per block = 512 / 128 = 4

For 100 records ,

how many blocks? 100/4 = 25 blocks

Indexing

EID	Pointer
DO1	
DO2	
DO3	
.	
.	

⋮

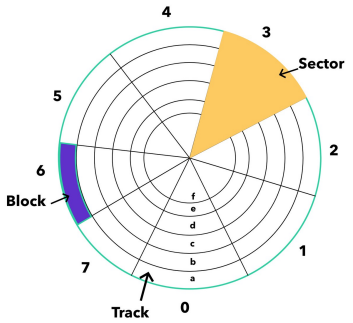
100

	Employee ID	First Name
2		
3	D01	William
4	D02	Rezza
5	D03	Mohan
6	D04	Cindy
7	D05	Vince
8	D06	Rajiv
9	D07	John
10	D08	Oshane
11	D09	Chin
12	D10	Linda
13	...	

...

⋮

100 records



Block size: 512 bytes

So, where to store indexing table ? Blocks !

entries per block = 512 / 16 = 32

→ 100 /32 ≈ 4

Data Layout on Disk

What is indexing?

Employee :

EID : 10

Name : 50

Destination : 10

Salary :50

Gender:8

Size = 128 bytes

EID : 10

Pointer : 6

Size = 16 bytes

records per block = 512 / 128 = 4

For 100 records ,

how many blocks? 100/4 = 25 blocks

Block access :4+1 blocks

Indexing

EID	Pointer
DO1	
DO2	
DO3	
.	
.	

⋮

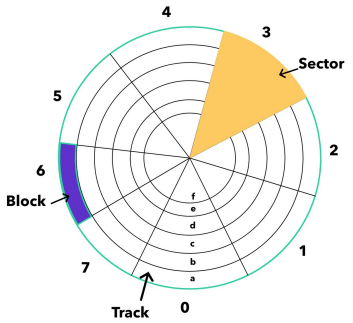
100

	Employee ID	First Name
2		
3	D01	William
4	D02	Rezza
5	D03	Mohan
6	D04	Cindy
7	D05	Vince
8	D06	Rajiv
9	D07	John
10	D08	Oshane
11	D09	Chin
12	D10	Linda
13	...	

...

⋮

100 records



Block size: 512 bytes

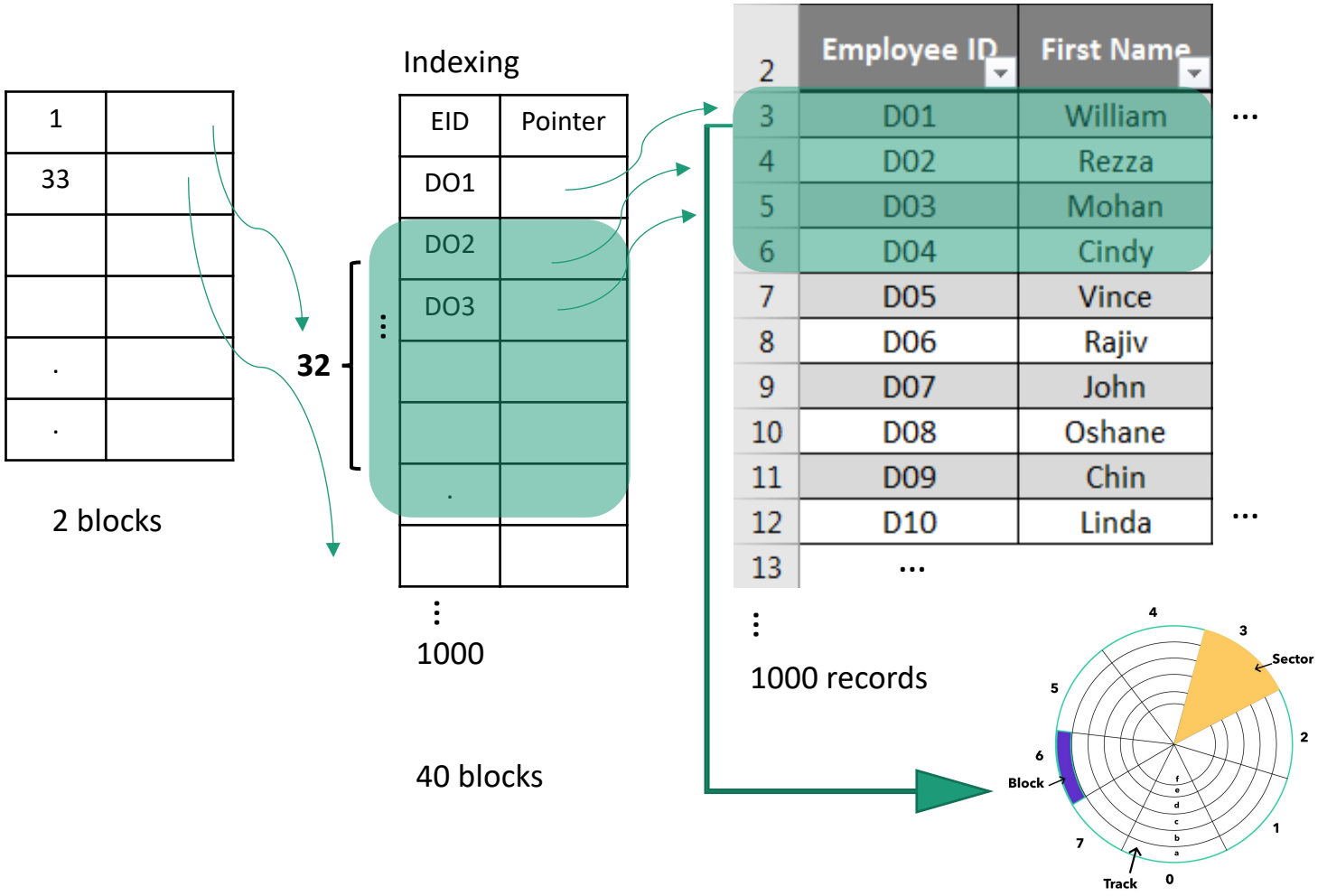
So, where to store indexing table ? Blocks !

entries per block = 512 / 16 = 32

→ 100 /32 ≈ 4

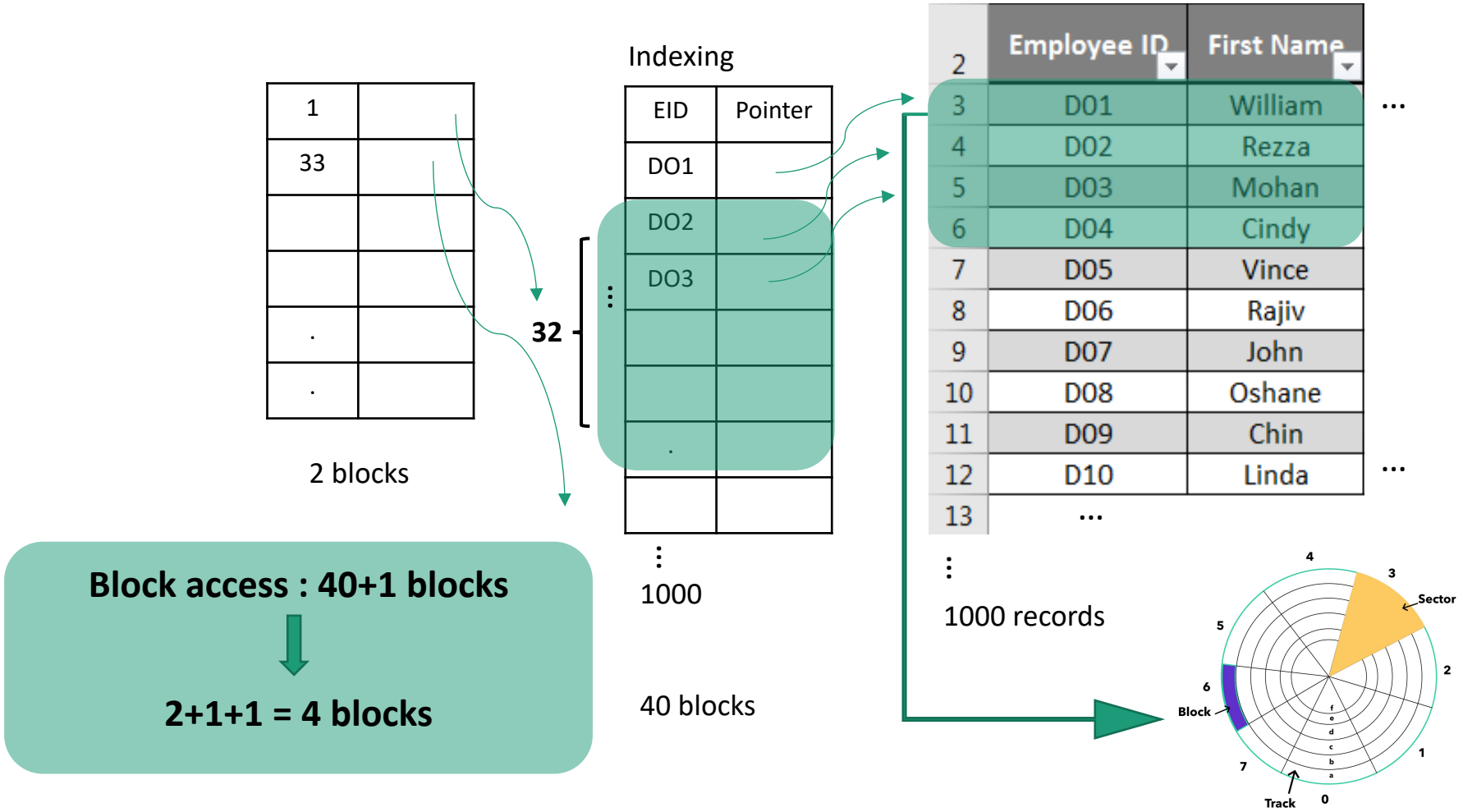
Data Layout on Disk

What is multi-level indexing?



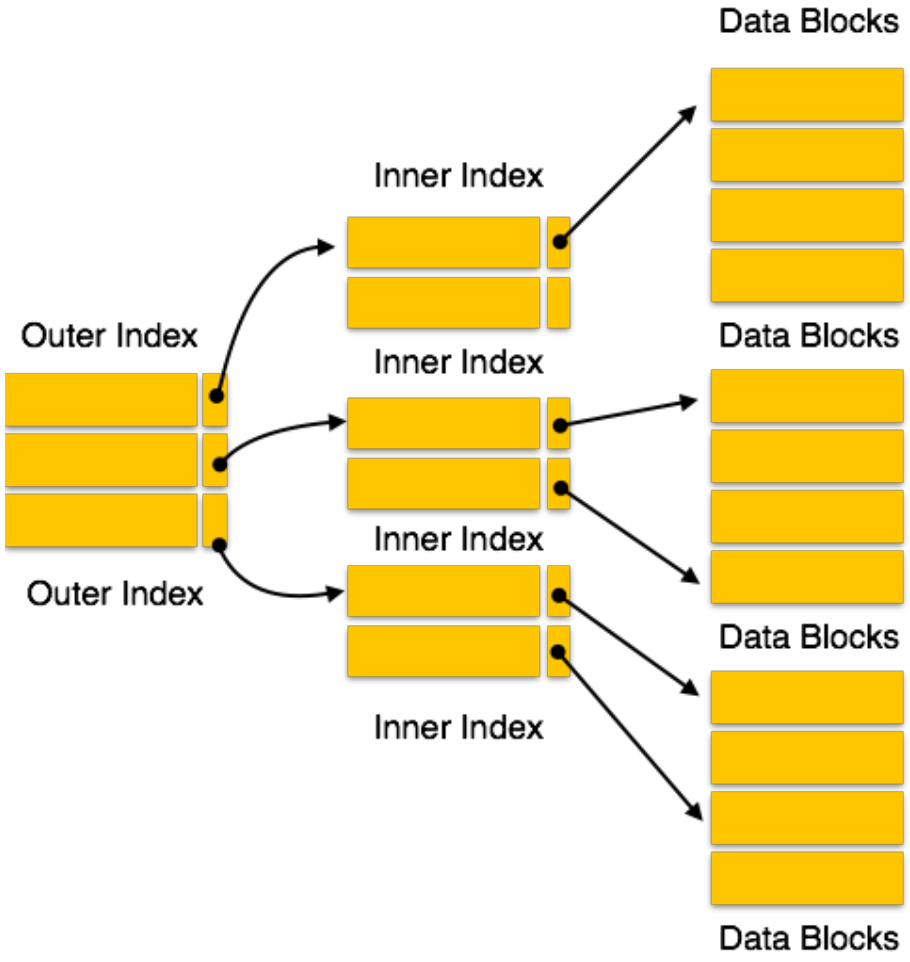
Data Layout on Disk

What is multi-level indexing?



Data Layout on Disk

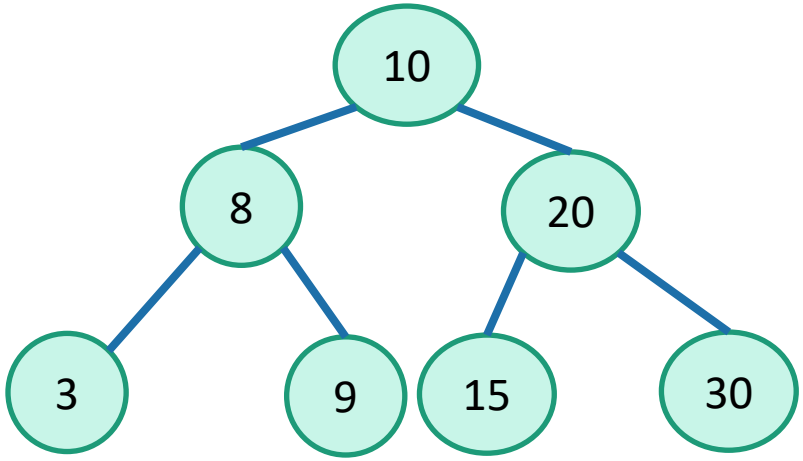
What is multi-level indexing?



M-way search tree

Review : binary search tree

- each tree node has a maximum of two children.
- search tree: search for the presence of a number in O(log(n)) time.



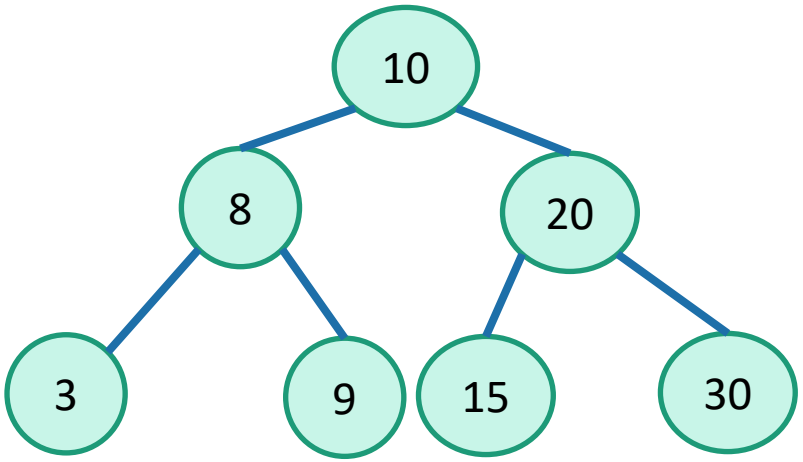
Properties :

1. All nodes of left subtree are less than the root node
2. All nodes of right subtree are more than the root node
3. Both subtrees of each node are also BSTs i.e. they have the above two properties

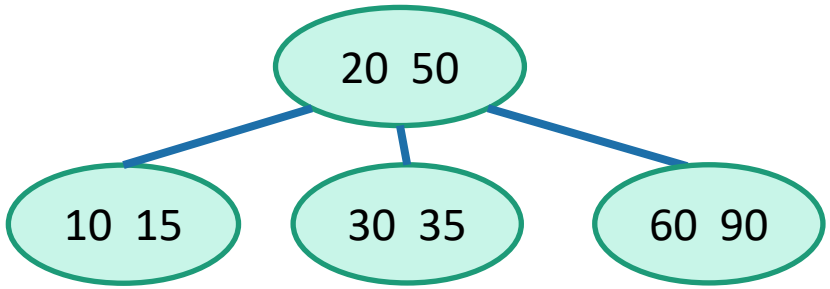
M-way search tree

binary search tree vs m-way search tree

1 key
2 children



2 key
3 children



$k_1 < k_2 < k_3 \ldots$

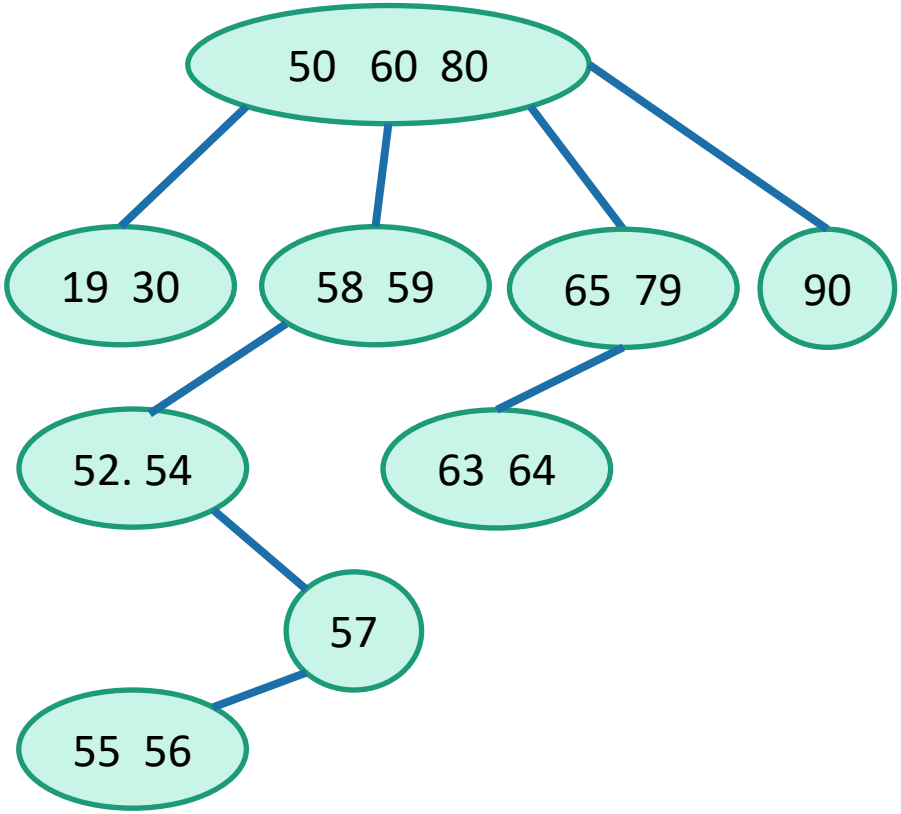
B-tree is a special type of M-way tree.
M-way(multi-way) tree properties:

- Each node in the tree can have at most m children.
- Nodes in the tree have at most **(m-1)** key fields and pointers(references) to the children.

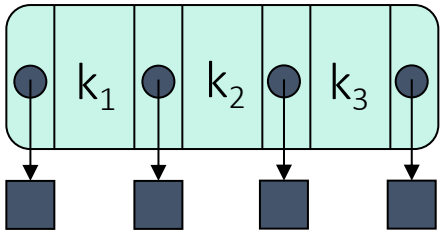
M-way search tree

representation : e.g. 4-way tree

- 4-way search tree : 3 keys



Node structure

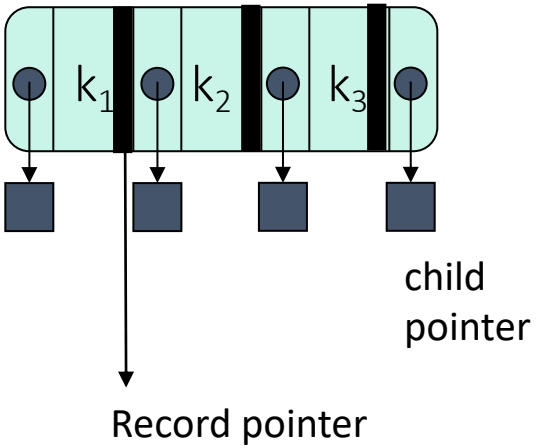


child pointer

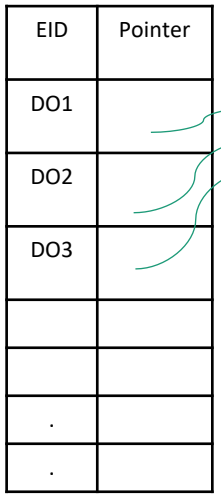
M-way search tree

How can we use m-way search tree for indexing?

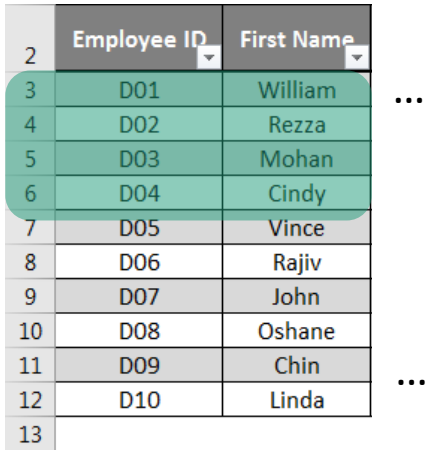
- 4-way search tree : 3 keys
- Node structure :



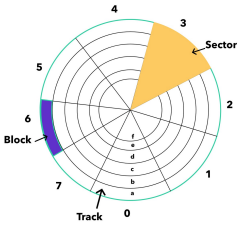
Indexing



1000



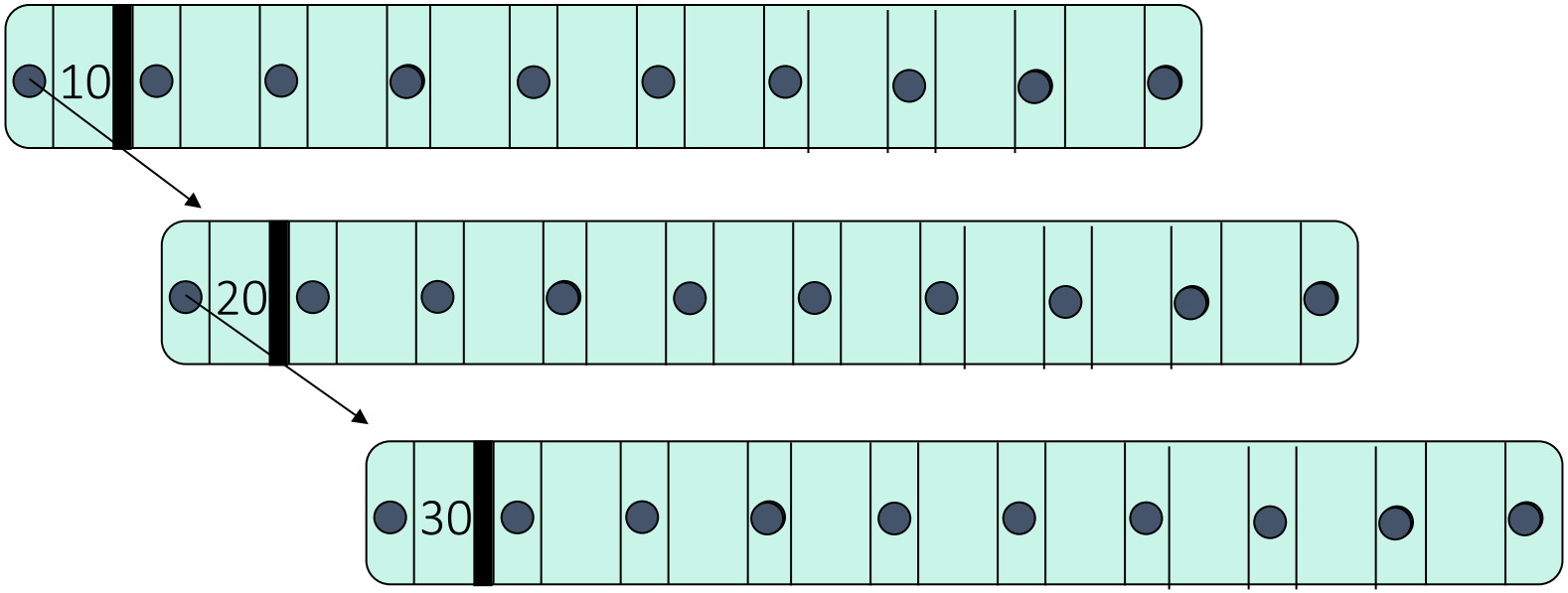
1000 records



M-way search tree

What is the problem with m-way search trees?

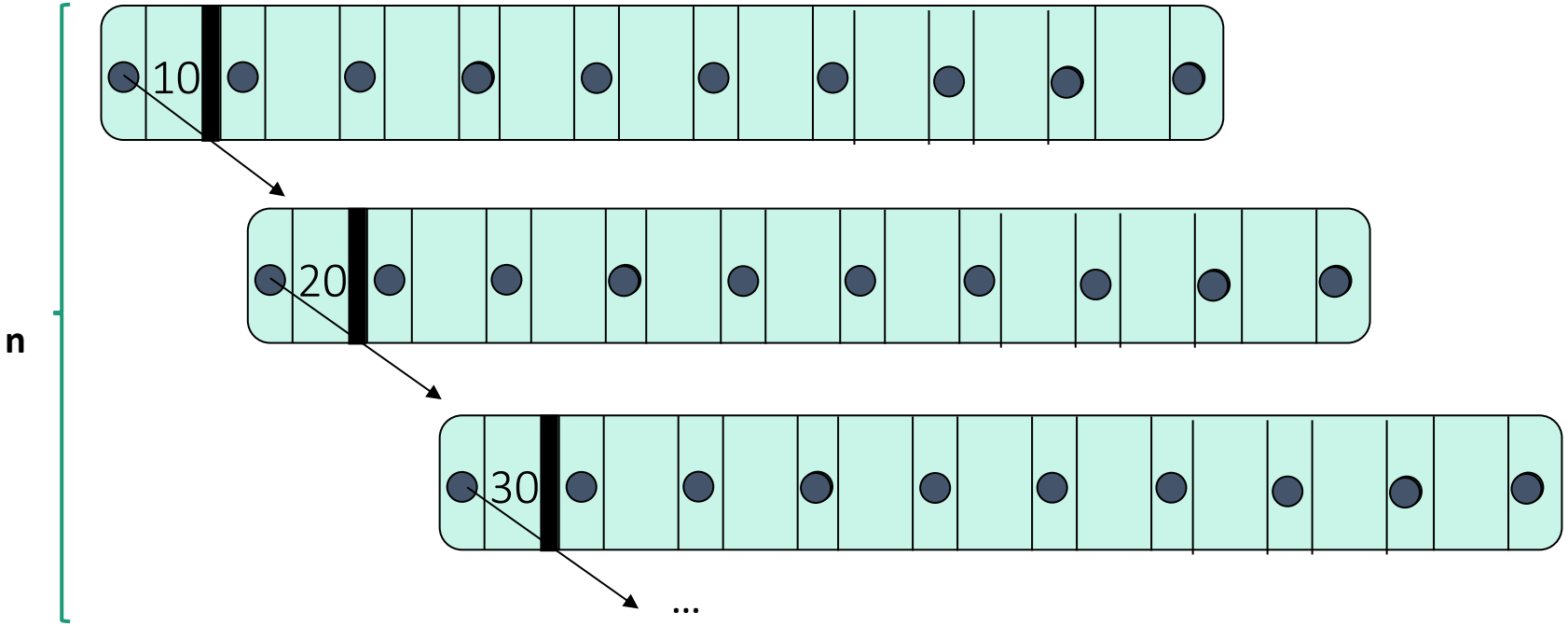
- 10-way search tree : 9 keys
- Insert keys : 10, 20, 30



M-way search tree

What is the problem with m-way search trees?

- What is the height of the tree after **n** insertion?



Time consuming, Same as linear search !

B-trees

A B-tree is an extension of an M-way search tree

Properties :

- All the leaf nodes in a B tree are at the same level.
- All internal nodes must have M/2 children.
- If the root node is a non leaf node, then it must have at least two children.
- All nodes except the root node, must have at least [M/2]-1 keys and at most M-1 keys.

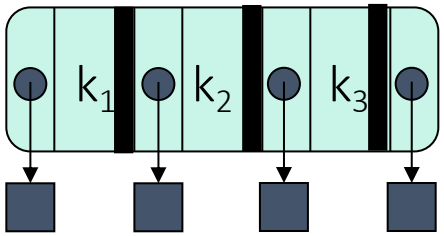
B-trees

Creation

- Let’s build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50



B-trees

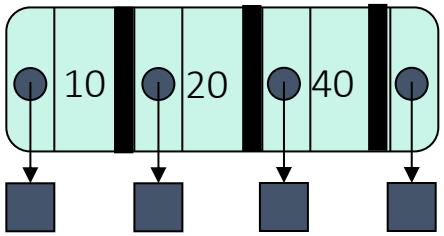
Creation : bottom-up

- Let’s build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  50 ?



B-trees

Creation : bottom-up

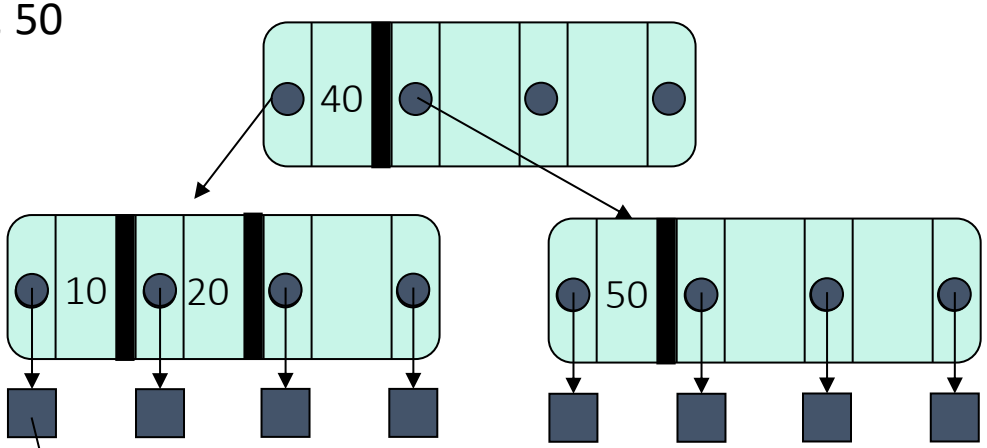
- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  50 ?

  Split : 10, 20, 40, 50



It could have children but now it doesn't have
So it's just an illustration

B-trees

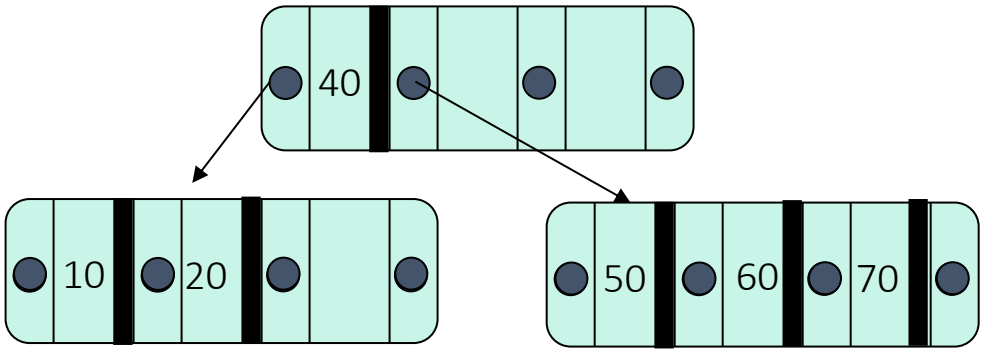
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  Insert more keys : 60, 70 ,80



B-trees

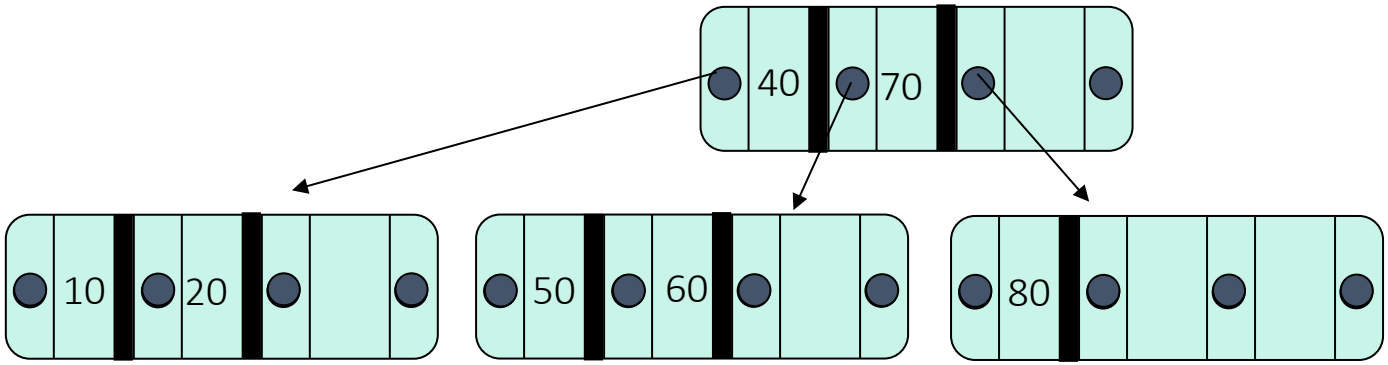
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  Insert more keys : 60, 70 ,80 → no space for 80; split !



B-trees

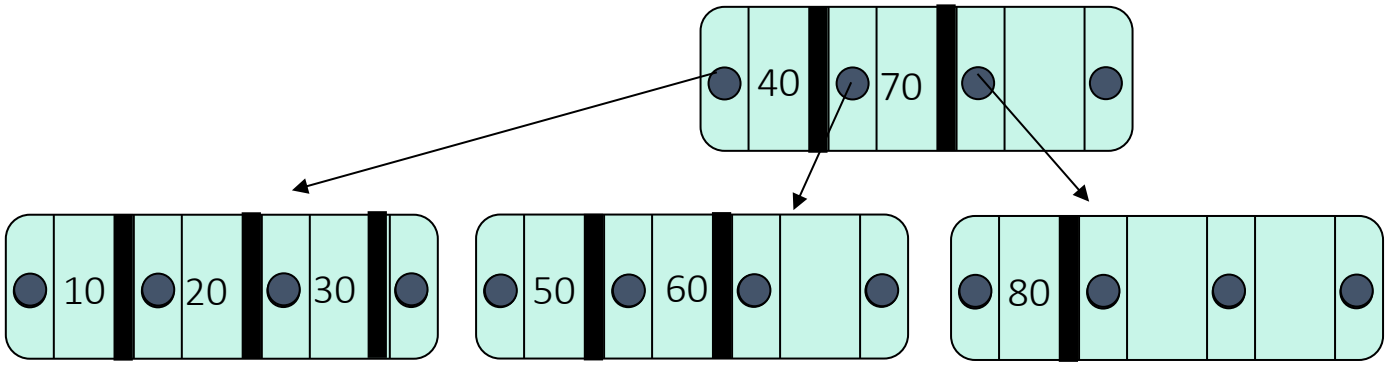
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  Insert more keys : 30,



B-trees

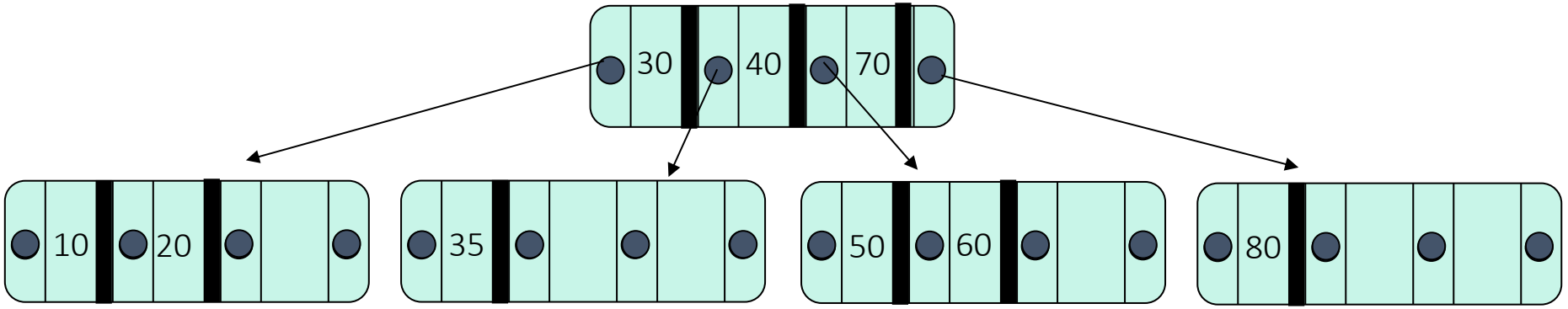
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  Insert more keys : 30, 35 → no space for 35; split ! → 10, 20, 30, 35



B-trees

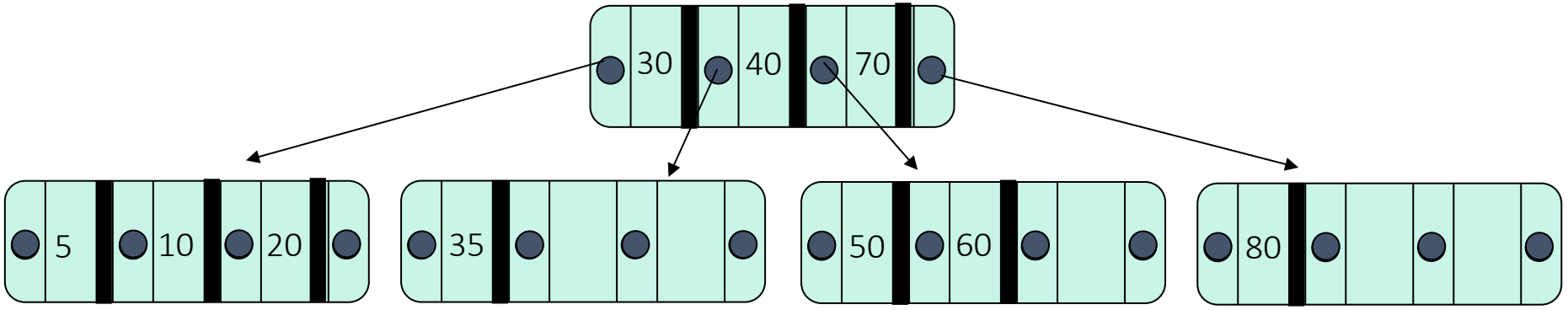
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  Insert more keys : 5, 15 → no space for 15; split ! → 5, 10, 15, 20



B-trees

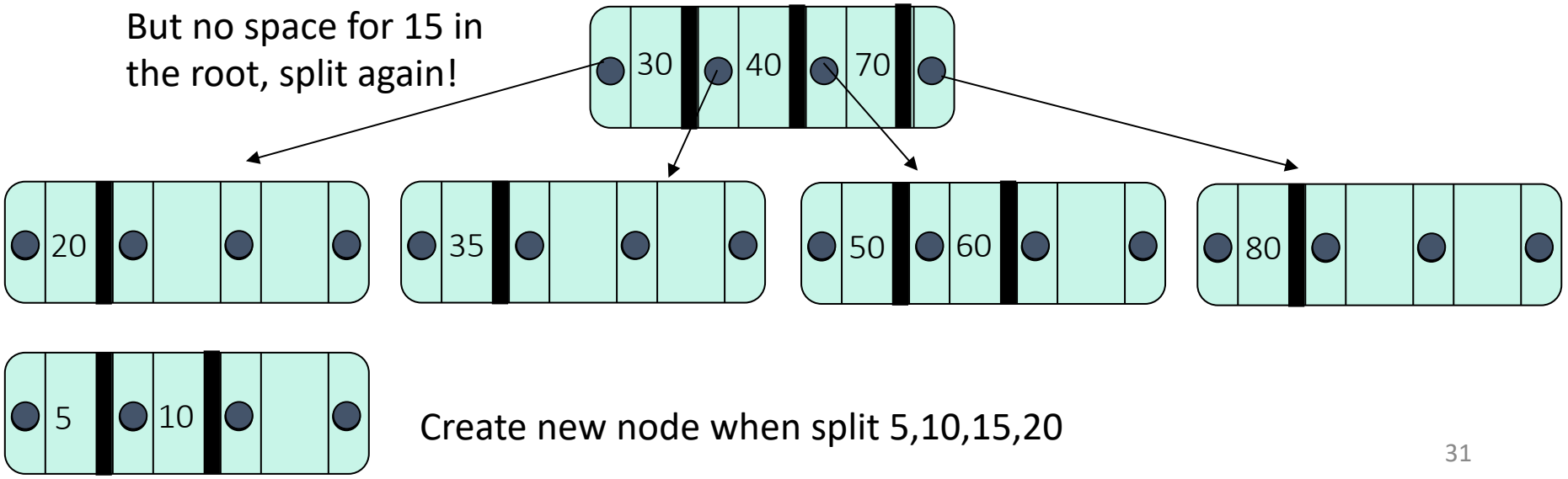
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

Insert more keys : 5, 15 → no space for 15; split ! → 5, 10, 15, 20



Create new node when split 5,10,15,20

B-trees

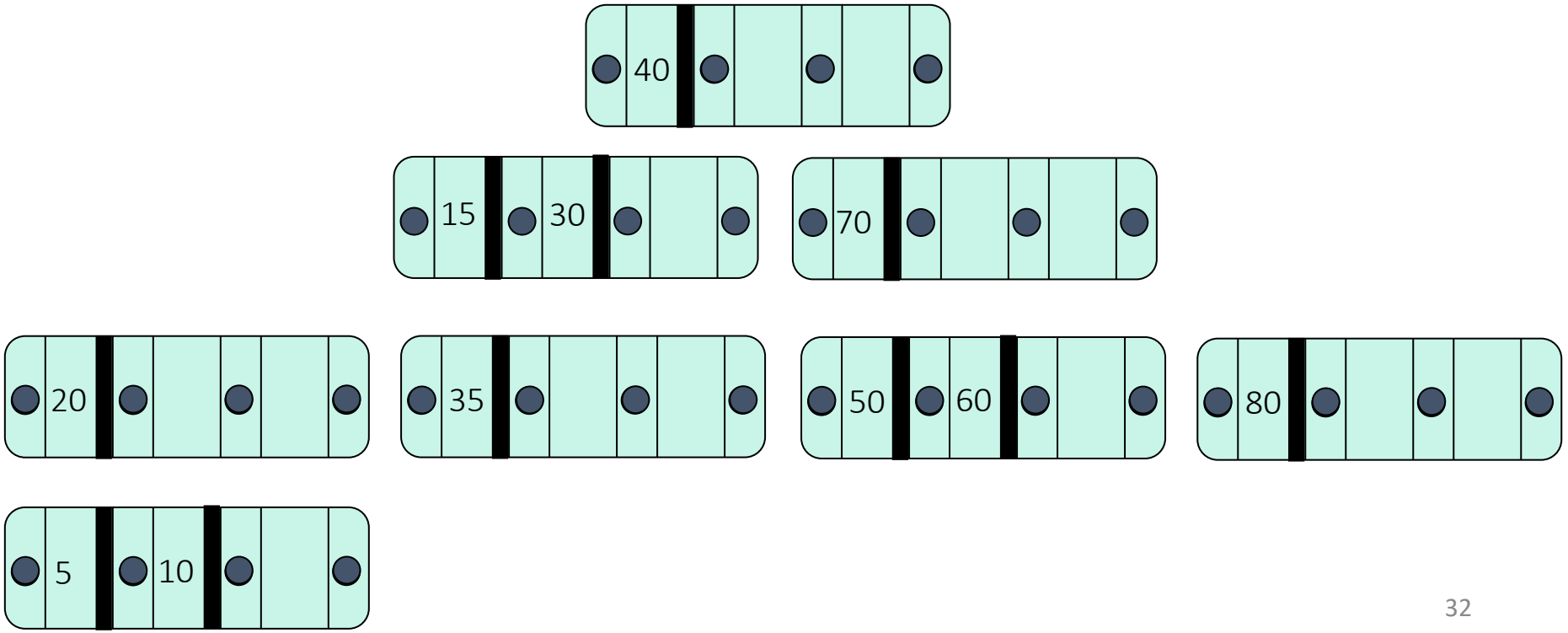
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

  split ! → 15, 30, 40, 70



B-trees

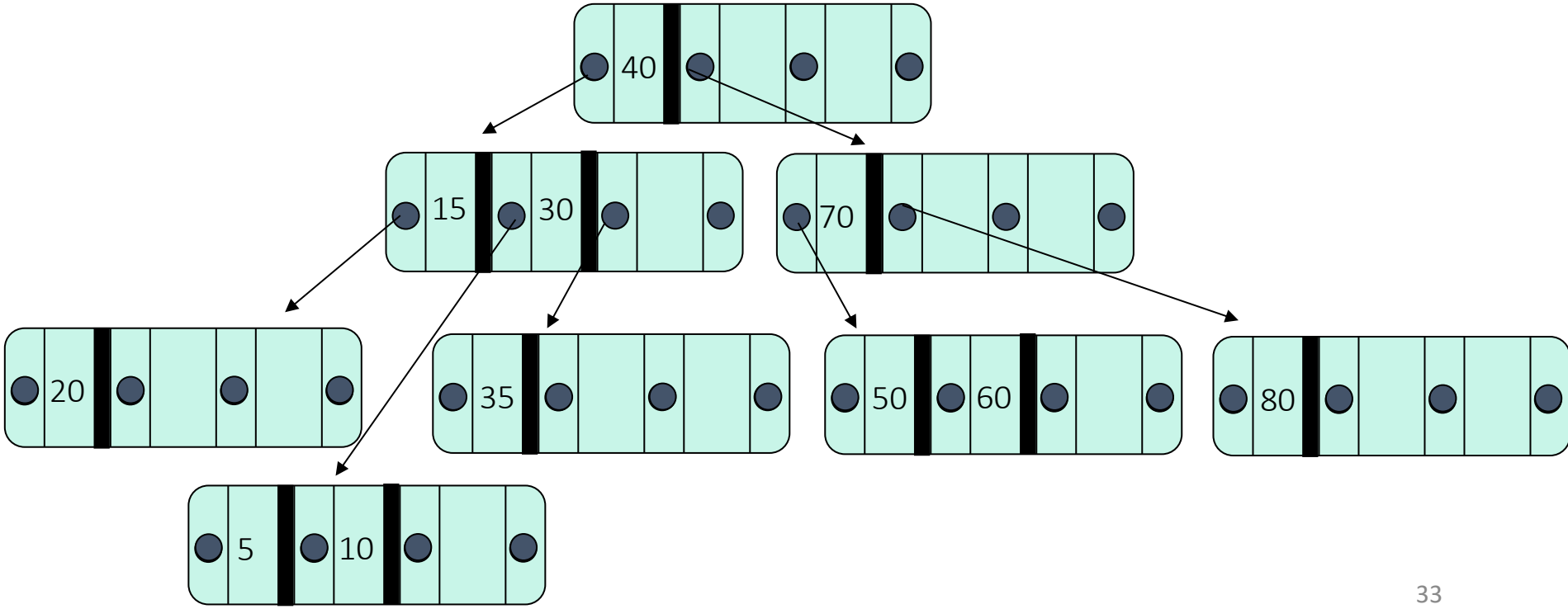
Insertion

- Let's build a b-tree such that :

  m = 4

  Keys : 10, 20, 40, 50

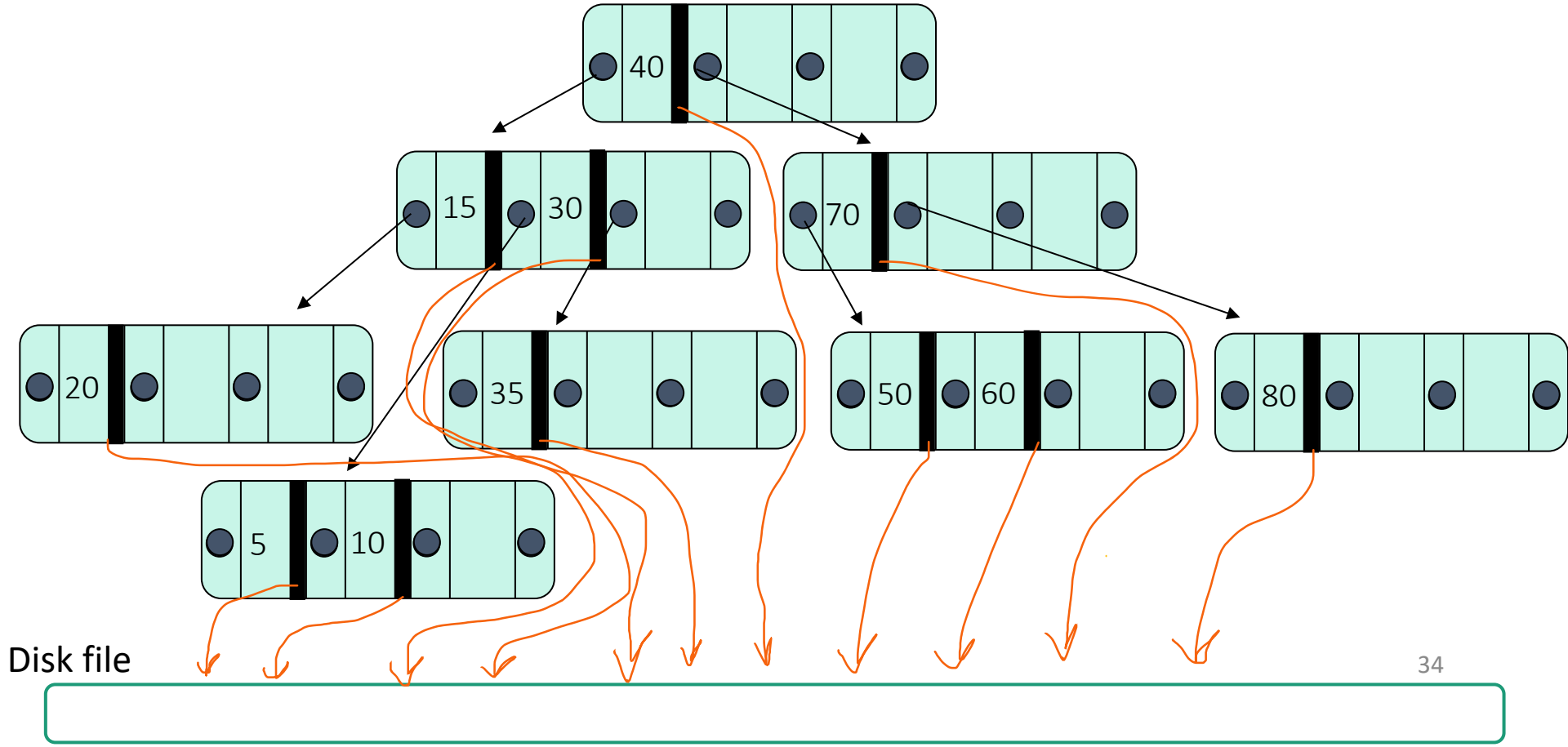
  split ! → 15, 30, 40, 70



B-trees

How Database B-Tree Indexing Works?

- In DBMS, B-tree is an example of multilevel indexing. Leaf nodes and internal nodes will both have record references.



B-trees

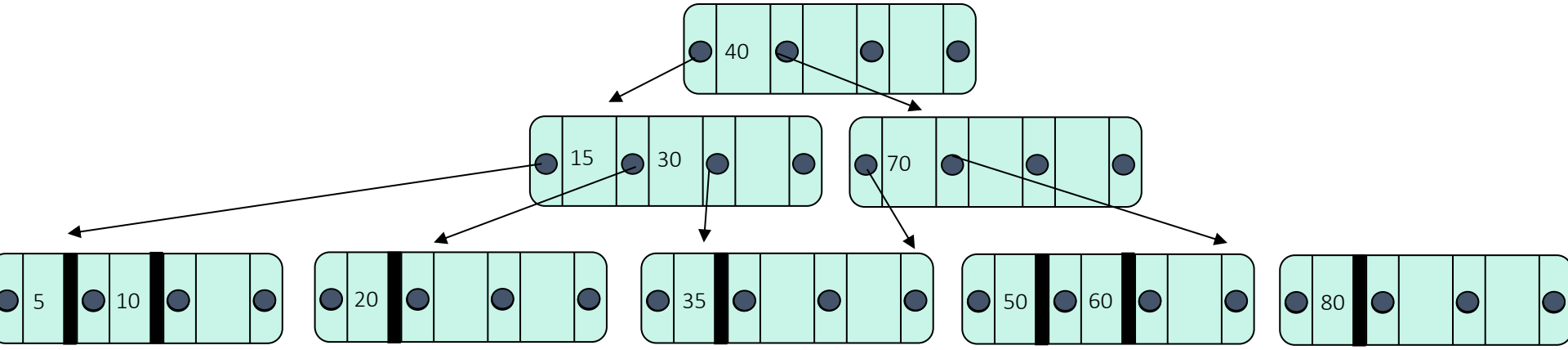
A self-balancing tree data structure

- B-tree in DBMS is an m-way tree which self balances itself. Due to their balanced structure, such trees are frequently used to manage and organize enormous databases and facilitate searches.
- In a B-tree, each node can have a maximum of n child nodes. In DBMS, B-tree is an example of multilevel indexing.

B^+-trees

How Database B^+-Tree Indexing Works?

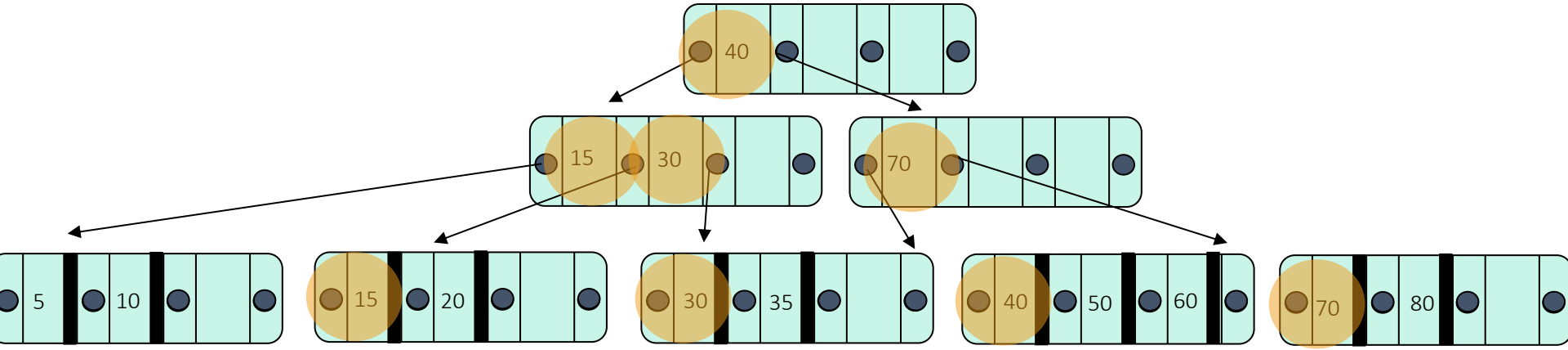
- B^+ tree eliminates the drawback B-tree used for indexing by storing data pointers only at the leaf nodes of the tree, how ?



B^+-trees

How Database B^+-Tree Indexing Works?

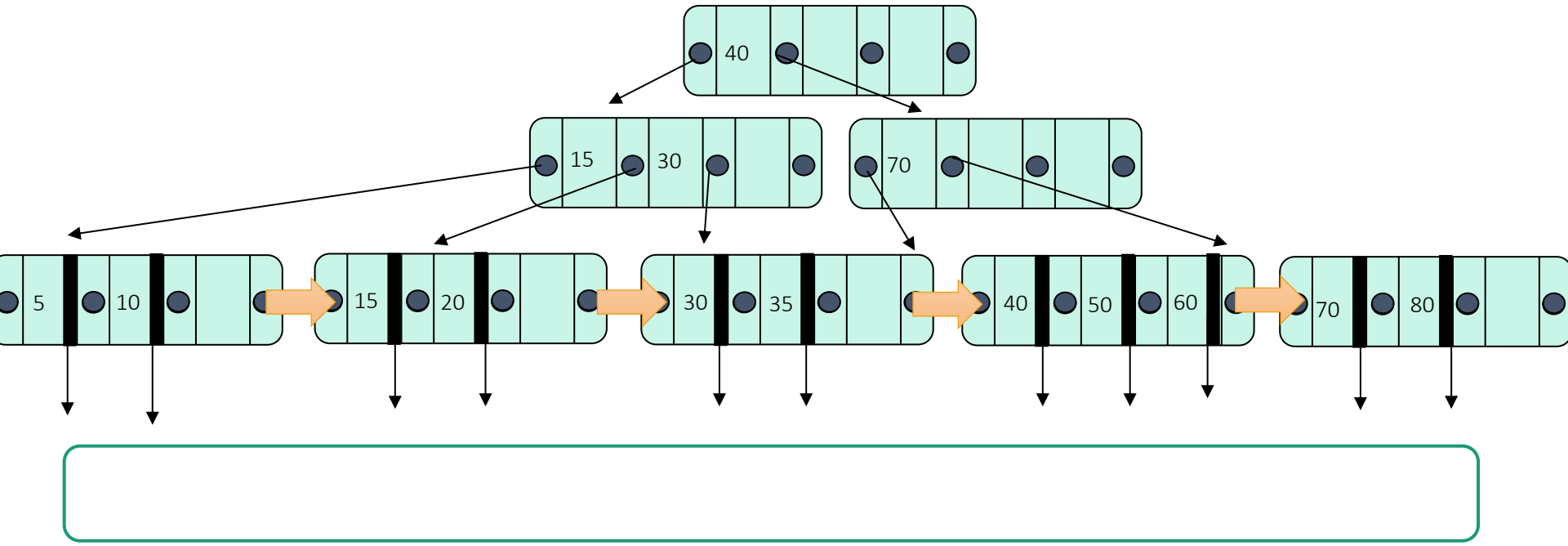
- Since data pointers are present only at the leaf nodes, the leaf nodes must necessarily store all the key values along with their corresponding data pointers to the disk file block, to access them.



B$^+$-trees

How Database B$^+$-Tree Indexing Works?

- the leaf nodes are linked to providing ordered access to the records.



Basis of Comparison	B tree	B+ tree
Pointers	All internal and leaf nodes have data pointers	Only leaf nodes have data pointers
Search	Since all keys are not available at leaf, search often takes more time.	All keys are at leaf nodes, hence search is faster and more accurate.
Redundant Keys	No duplicate of keys is maintained in the tree.	Duplicate of keys are maintained and all nodes are present at the leaf.
Leaf Nodes	Leaf nodes are not stored as structural linked list.	Leaf nodes are stored as structural linked list.
Access	Sequential access to nodes is not possible	Sequential access is possible just like linked list
Height	For a particular number nodes height is larger	Height is lesser than B tree for the same number of nodes
Application	B-Trees used in Databases, Search engines	B+ Trees used in Multilevel Indexing, Database indexing
Number of Nodes	Number of nodes at any intermediary level 'l' is 2^l.	Each intermediary node can have n/2 to n children.