[Advanced DBMS - Class 2]

⊙ Extendible Hashing: (Dynamic Hashing)

- It is a dynamic hashing method where in directories & buckets are used to hash data. It is an agressibly flexible method in which hash function experiences dynamic changes.

Example:

16 - 10000

4 - 00100

6 - 00110

22 - 10110

24 - 11000

10 - 01010

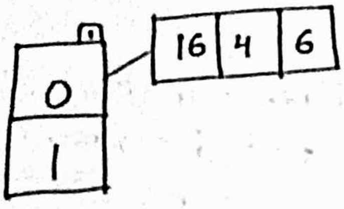
31 - 11111

7 - 00111

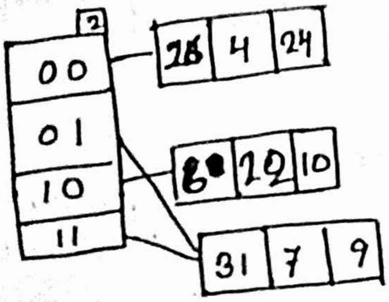
9 - 01001

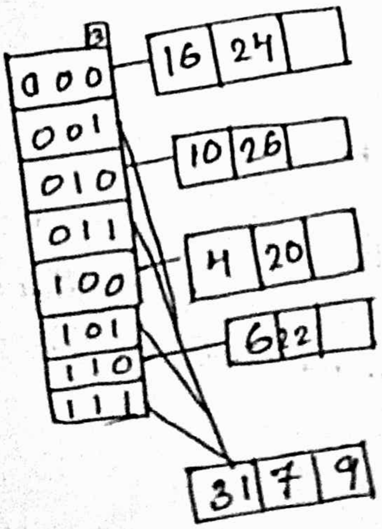
20 - 10100

26 - 11010



LSB





⊙ File Organization:

① Hash file organization

② Heap file organization

③ Sequential file organization

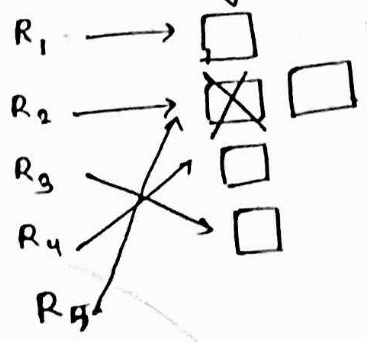
④ Indexed sequential file organization.

① HASH FILE ORGANIZATION:

- It uses the computations of hash function on some fields of the records. The hash function output determines the location of these blocks where records are to be placed. When a record has to be received using hash key then address is generated and ~~Hwd~~whole record is retrieved using that address.

② HEAP FILE ORGANIZATION:

- It is the simplest and most basic type of organization. It works wit data blocks. In this the records are inserted at the file's end. when records are inserted it does not require the sorting and ordering of records.



(Data records)

Insert : R1, R2, R3, R4

Delete : R2

Insert : R5

③ SEQUENTIAL FILE ORGANIZATION:

- It is the easiest method for file organization. In this method files are sorted sequentially. This method can be implemented by two ways:

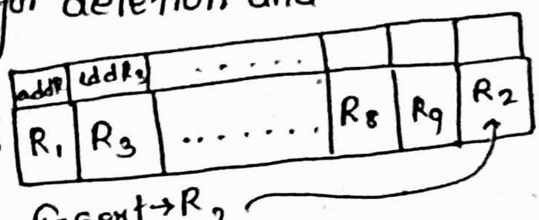
I. Pile-file method

II. Sorted-file method.

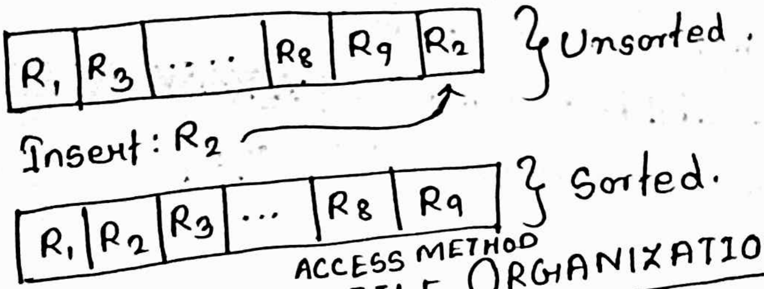
I· Pile-file-method:

→ It is a quite simple method. In this method, we store the record in a sequence, i.e., one after another. Here, the record will be inserted in order in which they are inserted into tables. In case of updating/deleting of any record, the records will be searched in the memory blocks. When it is found, then it will be marked for deletion and the new record may be inserted in that place.

II· Sorted-file method:

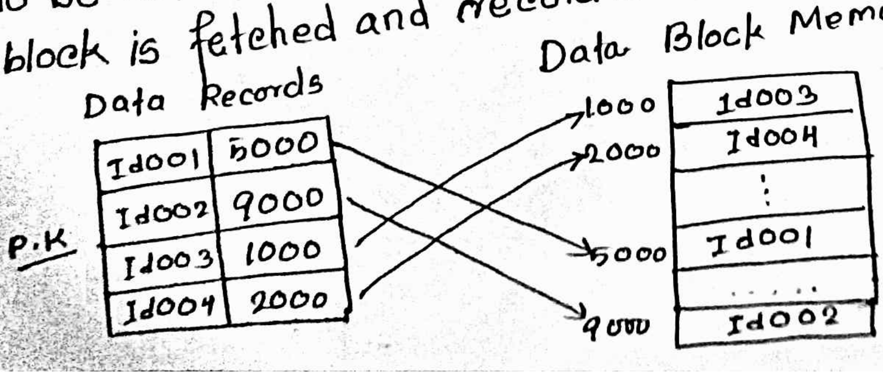


→ In this method the new record is always inserted at the files end and then it will sort in ascending or descending order. In case of modification of any record it will update the record and sort the file and lastly updated record is placed in the right place.



④ INDEXED SEQUENTIAL ACCESS METHOD FILE ORGANIZATION:

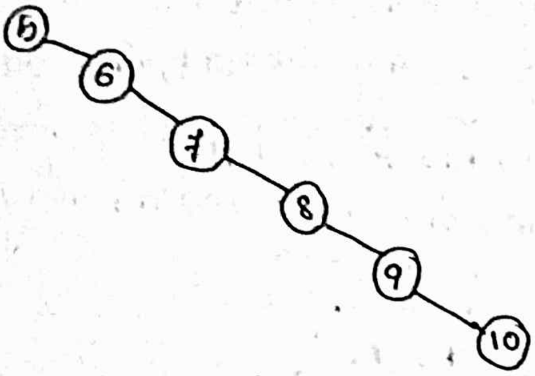
→ ISAM is an advanced sequential file organization. In this, records are stored in the file using the primary key and index value is generated for each primary key and mapped with the record and this index contains address of records in the file. If any record has to be retrieved based on its index value, then address of data block is fetched and record is retrieved from memory.



[Faster than other method.]

⊙ Binary Search Tree

5, 6, 7, 8, 9, 10



⊙ Balanced m-way Tree (B-Tree)

- Generalization of binary search tree in which a node can have more than one key or more than one children.

It maintains sorted data.

All leaf nodes must be at same level.

B-tree of order m has following properties.

(i) Every node has maximum m-children.

(ii) Minimum no. of children will be 0 ~~and maximum will be m/2~~

(ii) Minimum no. of children will be $\lceil m/2 \rceil$

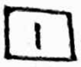
(iii) For internal nodes → minimum will be $\lceil m/2 \rceil$

root → 2

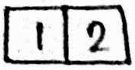
(iv) Maximum no. of children → (m-1)

1) Create a B-Tree of order 3 by inserting values (1-10).

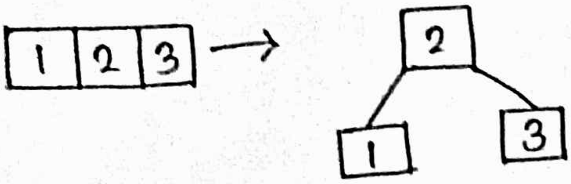
1:



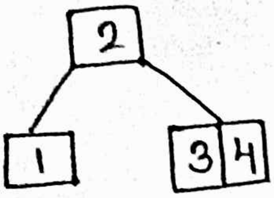
2:



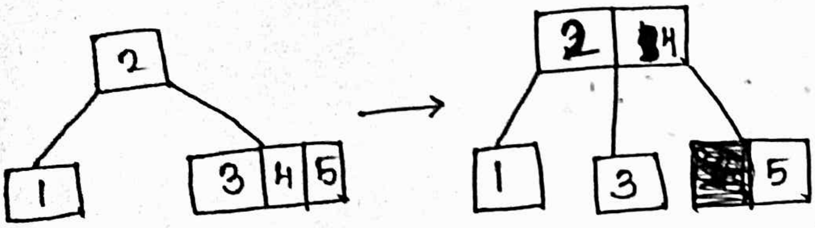
3:



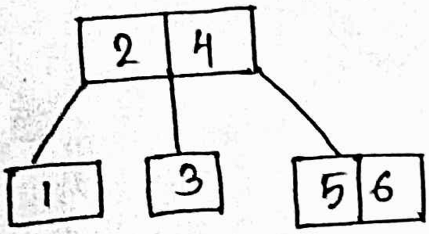
4:



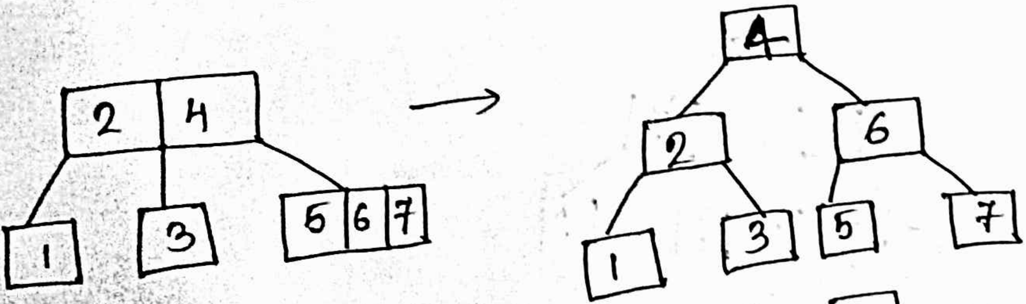
5:



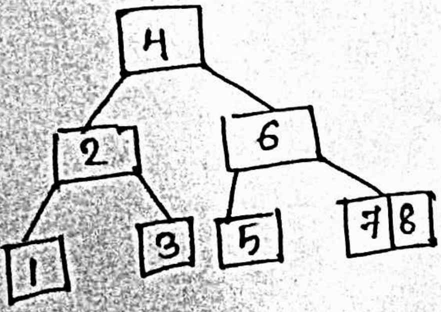
6:



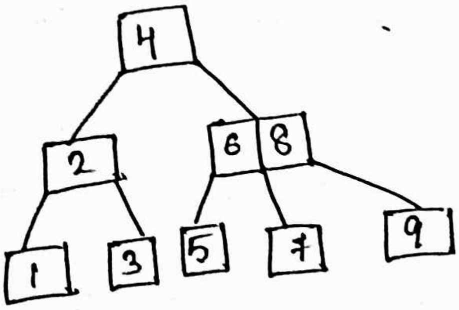
7:



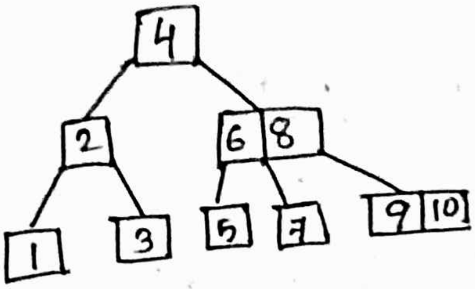
8:



9:



10:

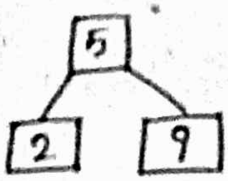


2) 2, 9, 5, 6, 3, 1, 7

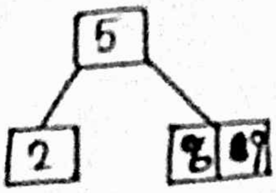
⇒ 2 : $\boxed{2}$

⇒ 9 : $\boxed{2}\boxed{9}$

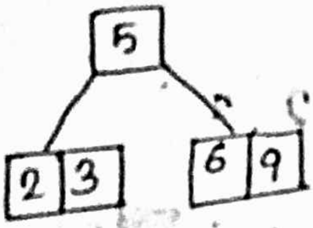
⇒ 5 :



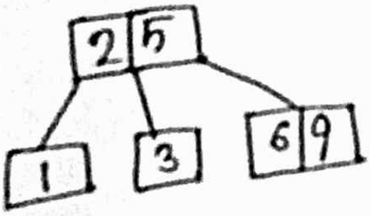
⇒ 6 :



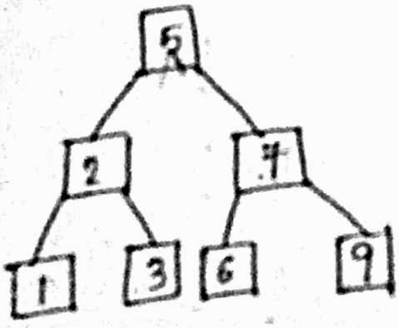
⇒ 3 :



⇒ 1 :



7 :



⊙ B^+ tree

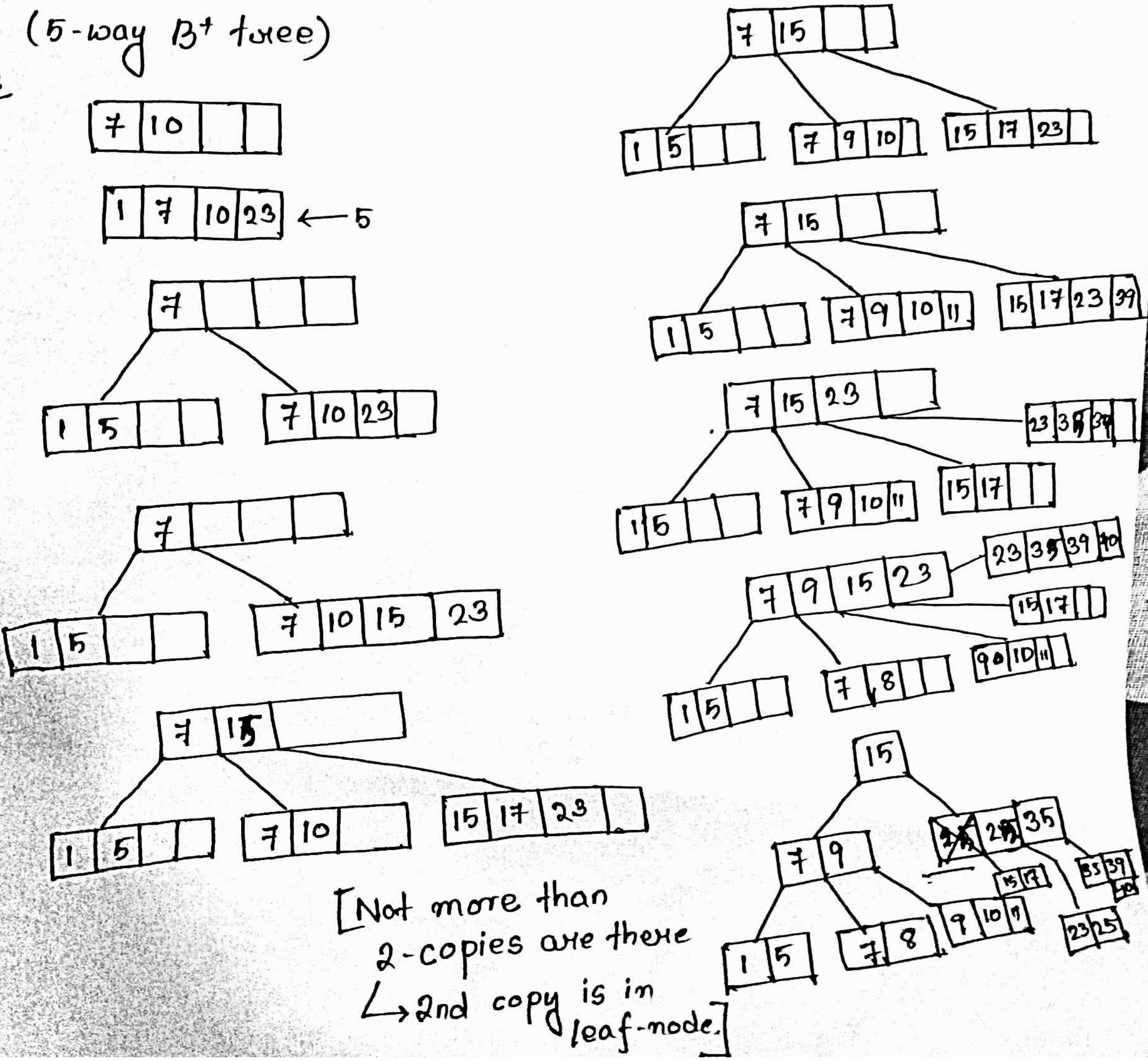
→ It is a variation of B-Tree ds. In a B^+ tree data pointers are stored only at the leaf nodes of the tree. In a B^+ tree structure of leaf node differs from the structure of internal nodes. The leaf nodes have an entry for every value of search field along with a data pointer to the records. The leaf nodes of B-tree are linked together to provide access to the search fields to the record.

Given,

7, 10, 1, 23, 5, 15, 17, 9, 11, 39, 35, 8, 40, 25

(5-way B^+ tree)

Sol:



[Not more than 2-copies are there ↳ 2nd copy is in leaf-node.]

⊙ Difference b/w B-tree and B$^+$ tree.