



UNIT V – Physical Storage and MongoDB

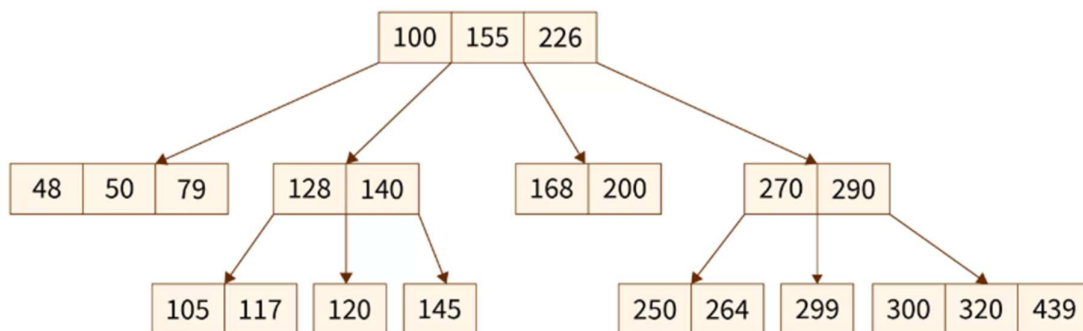
Data Storage and Indexes – RAID- File organization-Indexing and Hashing –Ordered Indices – B+ tree Index Files – B tree Index Files – Static Hashing – Dynamic Hashing. Query Processing Overview-Algorithms for Selection and Sorting Basics of MongoDB, Procedural Language

B Tree Index Files

An m-way tree that self-balances itself is called a "B-tree." 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 m child nodes.

Definition of B-tree

B-tree in DBMS is an m-way tree that 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. Leaf nodes and internal nodes will both have record references. B-Tree is called a Balanced stored tree as all the leaf nodes are at the same levels.



Properties of B-tree

- A non-leaf node's number of keys is one less than the number of its children.
- The number of keys in the root ranges from one to (m-1) maximum. Therefore, the root has a minimum of two and a maximum of m children.

- The keys range from $\min(\lceil m/2 \rceil - 1)$ to $\max(m-1)$ for all nodes (non-leaf nodes) besides the root. Thus, they can have between m and $\lceil m/2 \rceil$ children.
- The level of each leaf node is the same.

Need of B-tree

- For having optimized searching we cannot increase a tree's height. Therefore, we want the tree to be as short as possible in height.
- Use of B-tree in DBMS, which has more branches and hence shorter height, is the solution to this problem. Access time decreases as branching and depth grow.
- Hence, use of B-tree is needed for storing data as searching and accessing time is decreased.
- The cost of accessing the disc is high when searching tables. Therefore, minimizing disc access is our goal.
- So to decrease time and cost, we use B-tree for storing data as it makes the Index Fast.

How Database B-Tree Indexing Works

- When B-tree is used for database indexing, it becomes a little more complex because it has both a key and a value. The value serves as a reference to the particular data record. A payload is the collective term for the key and value.
- For index data to a particular key and value, the database first constructs a unique random index or a primary key for each of the supplied records. The keys and record byte streams are then all stored on a B+ tree. The random index that is generated is used for indexing of the data.
- So this indexing helps to decrease the searching time of data. In a B-tree, all the data is stored on the leaf nodes, now for accessing a particular data index, the database can make use of binary search on the leaf nodes as the data is stored in the sorted order.
- If indexing is not used, the database reads each and every record to locate the requested record and it increases time and cost for searching the records, so B-tree indexing is very efficient.