B+ Tree Database

Nawrin Sikder, Emily Ramkissoon, Preethi Sridhar, Neelambari Mathivathanan

INTRODUCTION

- B+ trees are a powerful tool for managing large amounts of data in a disk based storage system.
    - Designed to reduce the number of disk accesses.
- B+ trees are commonly used in database management systems as well as file systems.
- This is actually used to share data in a tree structure, where each node in the tree represents a wide range of keys.
- In a B+ tree, the leaf nodes of the tree contain actual data records.

INSERTION

- The insertion process involves finding the correct leaf node where a new key value can be inserted and then added to the leaf node.
- Rules of Insertion
    - Each node except the root can have a maximum of m children and at least m/2 children.
    - The root has at least two children
    - Each node can contain a maximum of m-1 keys and a minimum of (m/2) keys.
    - If the leaf *is not* full, insert the key into the leaf node in increasing order.
    - If the leaf *is* full, insert the key into the leaf node in increasing order and balance the tree in the following way.
    - Break the node at m/2 position.
    - Add m/2 key to the parent node as well.

DELETION

- The deletion process involves finding the node that contains the ‘key to be deleted’ in a particular node. From here, it is removed from the tree.
- A node can have a maximum of m children.
- A node can contain a maximum of m -1 keys.
- A node should have a minimum of (m/2) children.
- A node (excluding the root node) should contain a minimum of {m/2}-1 keys.

SEARCH

- For searching in B+ tree we need to traverse the tree from the root node to a leaf node. The key will be contained here. However, we will need to traverse to the end of the tree if the key is not present.
- When a new value is inserted, the tree is sorted every time.
- We need to compare the search key with the keys sorted in the current node.
- If the search key is *less* than the smallest key in a node, move to the left child.
- If the search key is *greater* than or *equal* to the largest key in a node, move to the right child.
- If the search key falls between two keys in the node, move to the child node whose key range includes the search key.
- Lastly, if the search key is found, return the value. If the search key is not found, return null.

PROPERTIES

- Insertion, deletion and search operations can be performed efficiently, even for very large datasets.
- The leaf nodes in a B+ tree are linked together in a linked list. This enables fast sequential access to the data stored in the tree.
- Each node in a B+ tree can store multiple key-value pairs. This reduces the overall number of nodes in the tree and improves access performance.

ADVANTAGES

- B+ trees offer good performance for insertion and deletion operations
- Requires local rebalancing of the tree rather than a complete tree reorganization.
- Accessing nodes in a B+ tree likely results in fewer cache misses
  - Improves overall performance
- B+ trees are powerful data structures that offer many advantages for handling large amounts of data.
- B+ trees offers fast search performance due to their balanced structure and the use of a binary search within the nodes.

APPLICATIONS

- B+ trees are used in coaching systems to store frequently accessed data in memory.
- B+ trees are used in the file systems to manage large files and directories efficiently.
- B+ trees are commonly used in database management systems to index and retrieve data efficiently as well.
- B+ trees are used in network routers to store routing tables and efficiently route network traffic.

CONCLUSION

- Challenges:
  - Implementing insert and delete functionally.
    - The insert operation would increase the size of the node.
    - With the delete operation we had to ensure that the nodes stayed above a minimum size.
  - Testing the TotalFusions and ParentFusions
- As a team, we also highlighted the differences between a B tree and a B+ tree.
- This project was highly valuable as we now fully understand the way in which a B+ tree functions. Insertion, deletion, searching, and displaying records, as well as knowing how to both maintain balance of a B+ tree as well as the properties of a B+ tree, have cemented their role and importance in computer science.

OUTPUT

DEMO