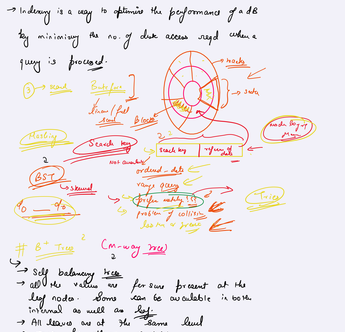
DBMS-4
Thursday, 30 July 2020 8:03 PM

Database Indexes

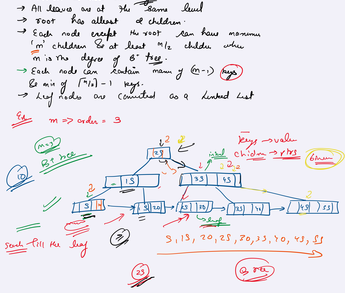
→ Indexing is a way to optimize the performance of a DB by minimizing the no. of disk access req'd when a query is processed.



B^+ Tree (m-way tree)

→ Self balancing tree
→ All the values are found at the leaf nodes. Stores can be available in both internal as well as leaf.
→ All leaves are at the same level
→ Root has atleast 2 children.
→ Each node except the root can have maximum 'm' children & at least m/2 children where m is the degree of B^+ tree.
→ Each node can contain max of (m-1) keys & min of $\lceil m/2 \rceil - 1$ keys.
→ Leaf nodes are connected as a linked list

Ex m => order = 3



Insertions →

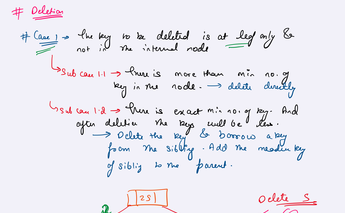
→ Every element is inserted in leaf node.

Case I → If the leaf is not full.
→ Search for the appropriate leaf node for yourself.
→ Insert the data in ordered form.

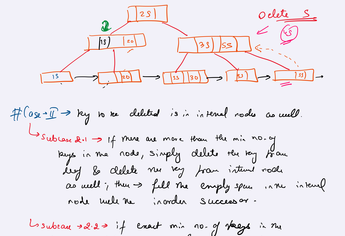
Case II → If the leaf is full.
1. → Insert the key in ordered form in the leaf node and balance in the following way
2. → Break the node at $(m/2)^{th}$ position
3. → Add the $(m/2)^{th}$ key to the parent node as well
4. If the parent node is already full repeat 2 & 3.

Deletion

Case I → the key to be deleted is at leaf only & not in the internal node
→ sub case I.1 → there is more than min no. of keys in the node. → delete directly
→ sub case I.2 → there is exact min no. of keys. And after deletion the keys will be less.
→ Delete the key & borrow a key from the sibling. Add the median key of sibling to the parent.



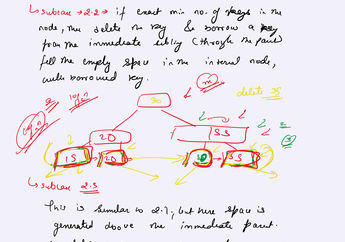
Case II → key to be deleted is in internal nodes as well
→ subcase 2.1 → If there are more than the min no. of keys in the node, simply delete the key from leaf & delete the key from internal node as well; then → fill the empty space in the internal node with the inorder successor.
→ subcase 2.2 → if exact min no. of keys in the node, then delete the key & borrow a key from the immediate sibling (through the parent) fill the empty space in the internal node, with borrowed key.

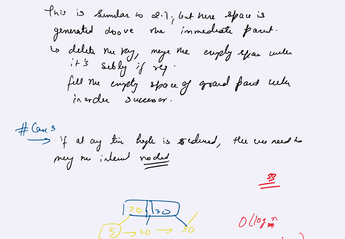


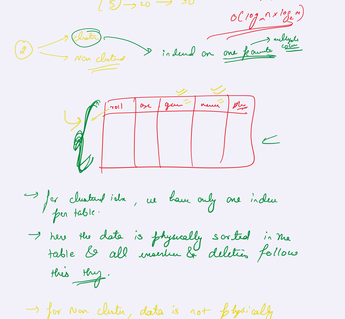
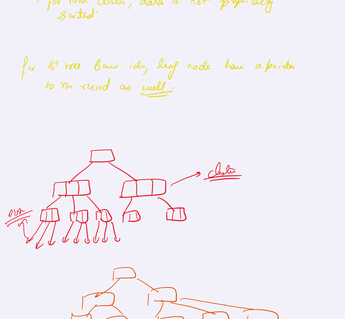
→ subcase 2.3

This is similar to 2.1, but here space is generated above the immediate parent.
→ delete the key, merge the empty space with its sibling if req.
fill the empty space of grand parent with inorder successor.

Case 3
→ If all any two keys is deleted, then we need to merge the internal nodes





→ For clustered index, we have only one index per table.
→ Here the data is physically sorted in the table & all insertion & deletion follow this key.
→ For non clustered, data is not physically sorted.

For B^+ tree Base index, leaf nodes have a pointer to the record as well.