Department of UIC

Database Management Systems

Pawandeep Sharma
Assistant Prof., UIC

B+ Tree in DBMS

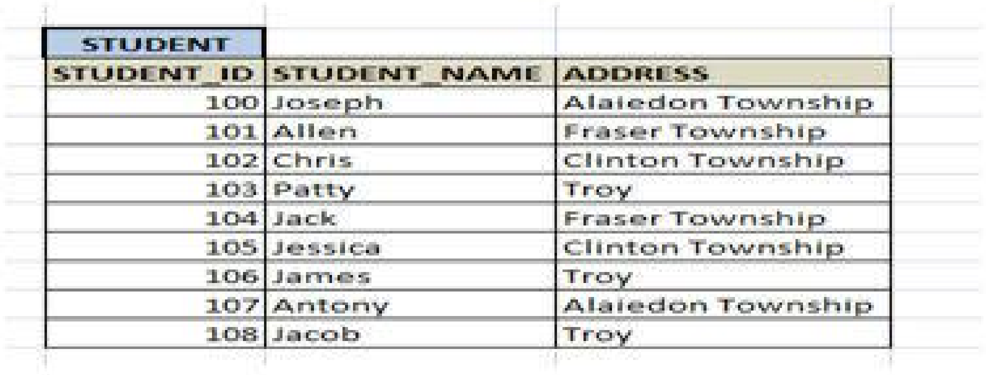
The B+ Tree

- B+ tree is a (key, value) storage method in a tree like structure.
- B+ tree has one root, any number of intermediary nodes (usually one) and a leaf node.
- All leaf nodes will have the actual records stored.
- Intermediary nodes will have only pointers to the leaf nodes; it not has any data.
- Any node will have only two leaves. This is the basic of any B+ tree.

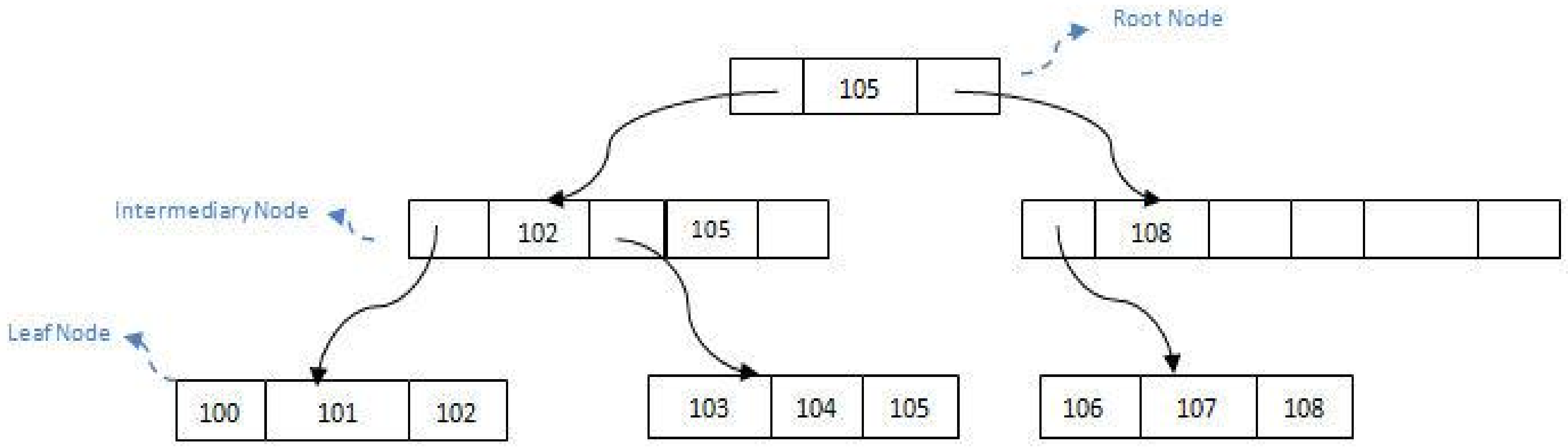
Basic Concepts

- Consider the STUDENT table that can be stored in B+ tree structure as shown below.
- We can observe here that it divides the records into two and splits into left node and right node.
- Left node will have all the values less than or equal to root node and the right node will have values greater than root node.
- The intermediary nodes at level 2 will have only the pointers to the leaf nodes.
- The values shown in the intermediary nodes are only the pointers to next level.
- All the leaf nodes will have the actual records in a sorted order.

The STUDENT table

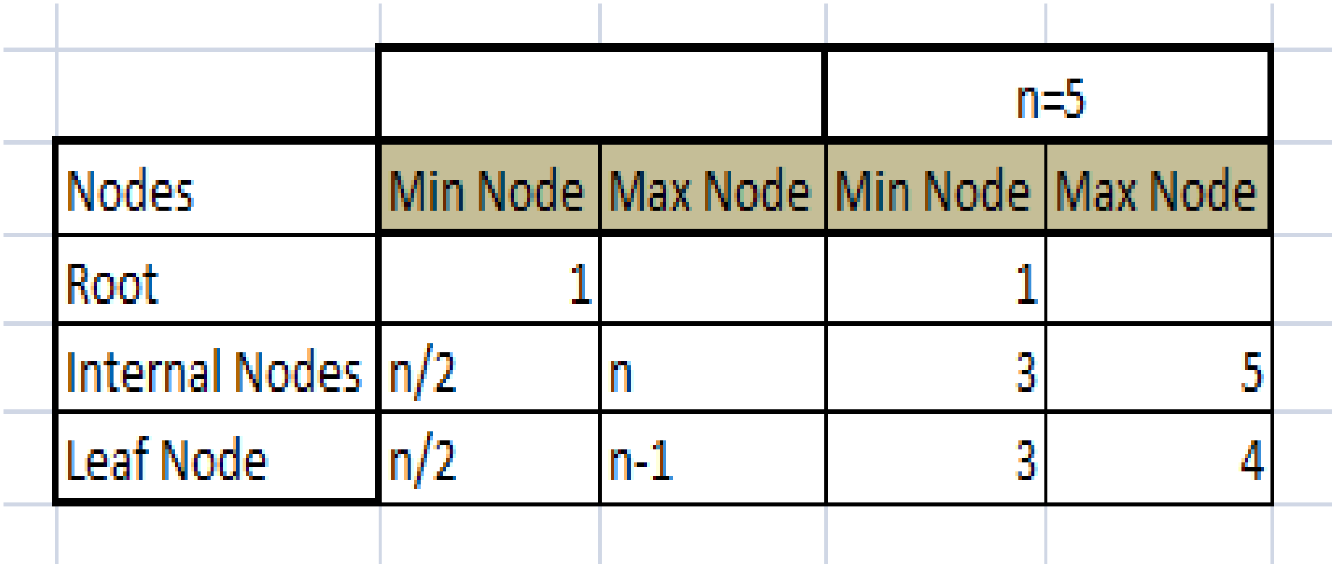


STUDENT		
STUDENT_ID	STUDENT_NAME	ADDRESS
100	Joseph	Alaiedon Township
101	Allen	Fraser Township
102	Chris	Clinton Township
103	Patty	Troy
104	Jack	Fraser Township
105	Jessica	Clinton Township
106	James	Troy
107	Antony	Alaiedon Township
108	Jacob	Troy



Basic Concepts

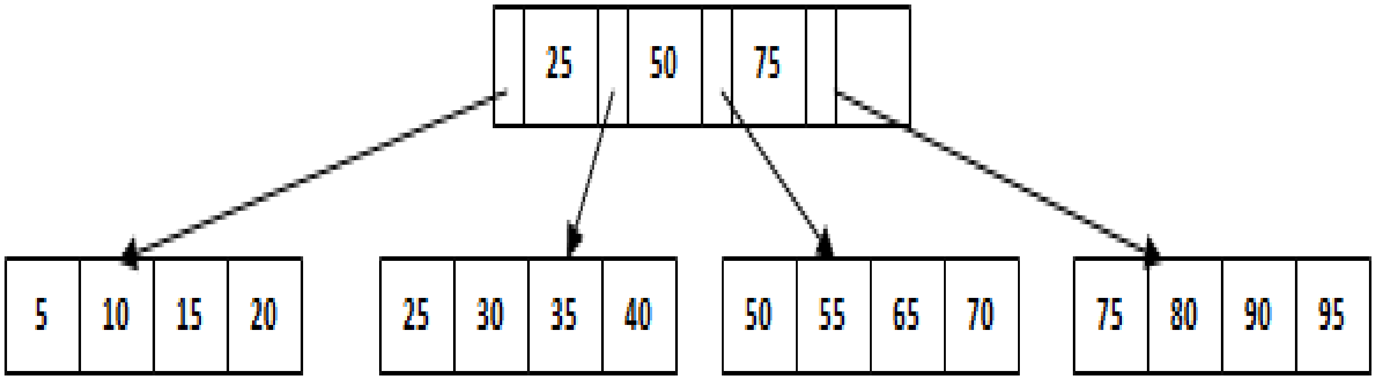
- Suppose a B+ tree has an order of n and then it can have n/2 to n intermediary nodes and n/2 to n-1 leaf nodes.
- In our example above, n= 5 i.e.; it has 5 branches from root. Then it can have intermediary nodes ranging from 3 to 5. And it can have leaf nodes from 3 to 4.



			n=5	
Nodes	Min Node	Max Node	Min Node	Max Node
Root	1		1	
Internal Nodes	n/2	n	3	5
Leaf Node	n/2	n-1	3	4

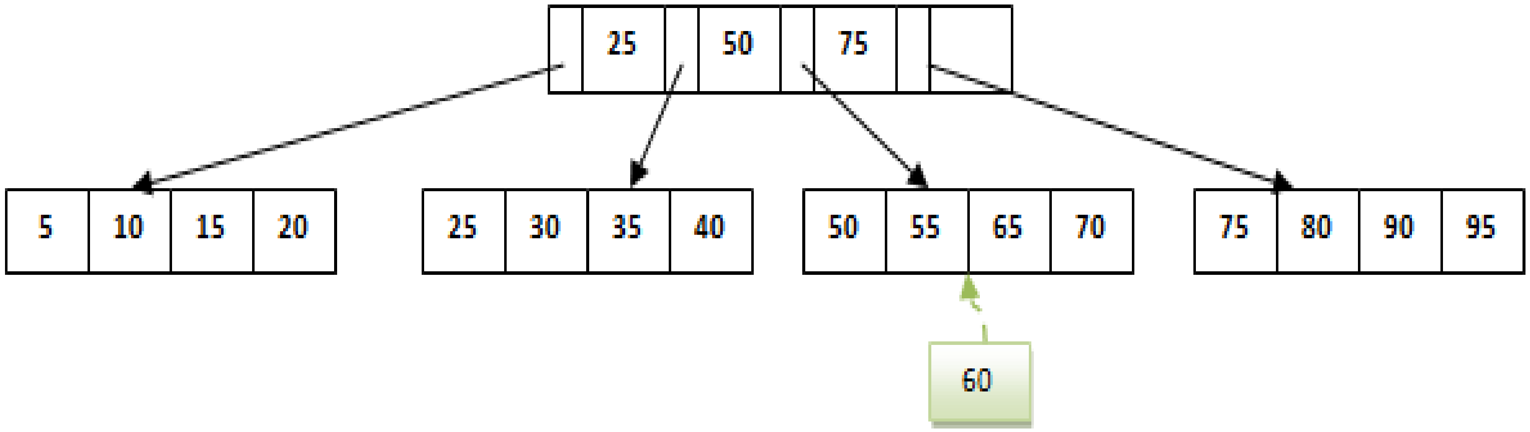
Searching a record in B+ Tree

- Suppose we want to search 65 in the below B+ tree structure. First we will fetch for the intermediary node which will direct to the leaf node that can contain record for 65. So we find branch between 50 and 75 nodes in the intermediary node. Then we will be redirected to the third leaf node at the end. Here DBMS will perform sequential search to find 65.



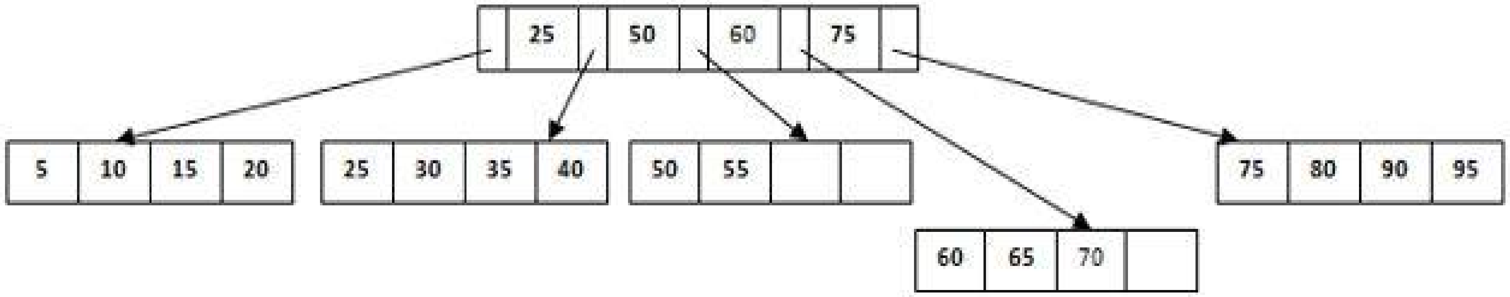
Inserting a record in B+ Tree

- Suppose we have to insert a record 60 in below structure. It will go to 3rd leaf node after 55. Since it is a balanced tree and that leaf node is already full, we cannot insert the record there. But it should be inserted there without affecting the fill factor, balance and order. So the only option here is to split the leaf node. But how do we split the nodes?



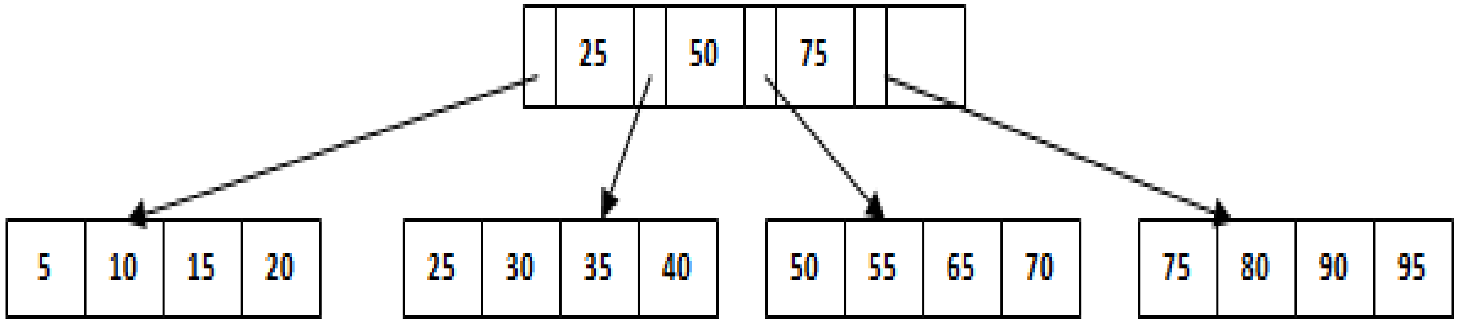
Inserting a record in B+ Tree

- The 3rd leaf node should have values (50, 55, 60, 65, 70) and its current root node is 50. We will split the leaf node in the middle so that its balance is not altered. So we can group (50, 55) and (60, 65, 70) into 2 leaf nodes. If these two has to be leaf nodes, the intermediary node cannot branch from 50. It should have 60 added to it and then we can have pointers to new leaf node.



Deleting a record in B+ Tree

- Suppose we have to delete 60 from the above example. What will happen in this case? We have to remove 60 from 4th leaf node as well as from the intermediary node too. If we remove it from intermediary node, the tree will not satisfy B+ tree rules. So we need to modify it have a balanced tree. After deleting 60 from above B+ tree and re-arranging nodes, it will appear as below.



B+ Tree index files

- The concept of B+ tree is used to store the records in the secondary memory.
- If the records are stored using this concept, then those files are called as B+ tree index files.
- Since this tree is balanced and sorted, all the nodes will be at same distance and only leaf node has the actual value, makes searching for any record easy and quick in B+ tree index files.
- Even insertion/deletion in B+ tree does not take much time. Hence B+ tree forms an efficient method to store the records.

QUERIES ??

THANK YOU