Dynamic Multilevel Indexes Using B-Trees And B^+ Trees

Prof. Neeraj Bhargava
Pooja Dixit
Department of Computer Science
School of Engineering & System Science
MDS, University Ajmer, Rajasthan, India

B–TREE

- It is a self-balancing search tree
- Data is accessed from main memory in case of self balanced search tree like AVL Tree
- Huge amount of data will be accessed from disks
- During accessing of data, disk access time will be very high compared to main memory access time
- The main idea of using B-Trees is to reduce the number of disk accesses

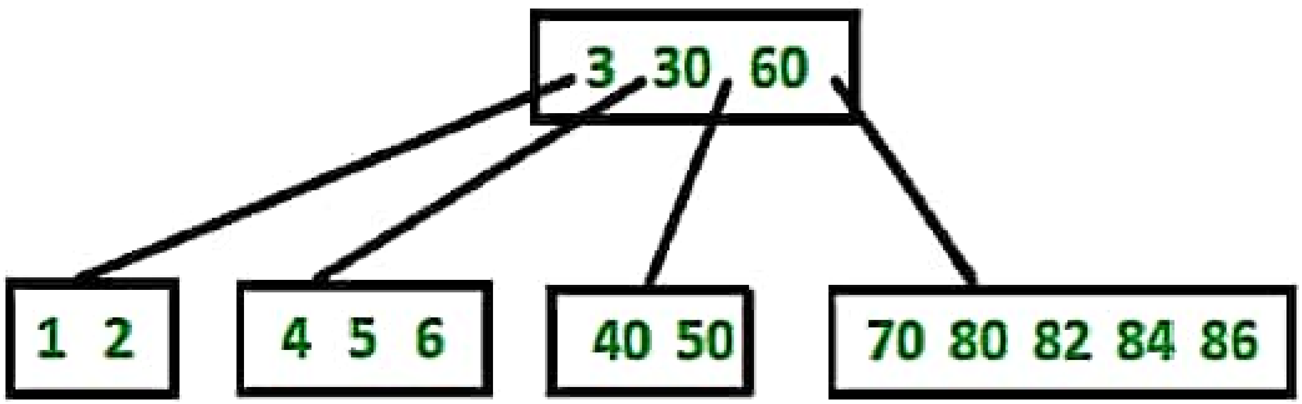
B–TREE

- Most of the tree operations (search, insert, delete etc) require O(h) disk accesses where h is the height of the tree
- B-tree is a fat tree
- Height is kept low by putting maximum possible keys in a B-Tree node
- Generally, a B-Tree node size is kept equal to the disk block size
- Since h is low for B-Tree, total disk accesses for most of the operations are reduced

Properties of B–Tree

- All leaves are at same level
- A B-Tree is defined by the term minimum degree 't' and the value of t depends upon disk block size
- Every node except root must contain at least t-1 keys
- Root may contain minimum 1 key
- All nodes (including root) may contain at most 2t – 1 keys
- Number of children of a node is equal to the number of keys in it plus 1
- All keys of a node are sorted in increasing order
- Time complexity is O(Log n)

B-Tree of minimum degree 3



Drawbacks of B–Tree

- Data pointer corresponding to a particular key value is stored along with that key value in the node of a B-tree. This reduces number of entries that is packed into a node of the tree
- Due to storing of Data Pointer the number of levels in a tree and search time of a record increases

B+ tree

- Eliminates the drawback of the B-Tree
- Stores data pointers only at the leaf nodes of the tree
- Structure of leaf nodes of a B+ tree is different from the structure of internal nodes of the B+ tree

B+ tree

- Leaf nodes store all the key values along with their corresponding data pointers to the disk file block, in order to access them
- Leaf nodes form the first level of index
- Internal nodes form the other levels of a multilevel index

Structure of the internal nodes of a B+ tree

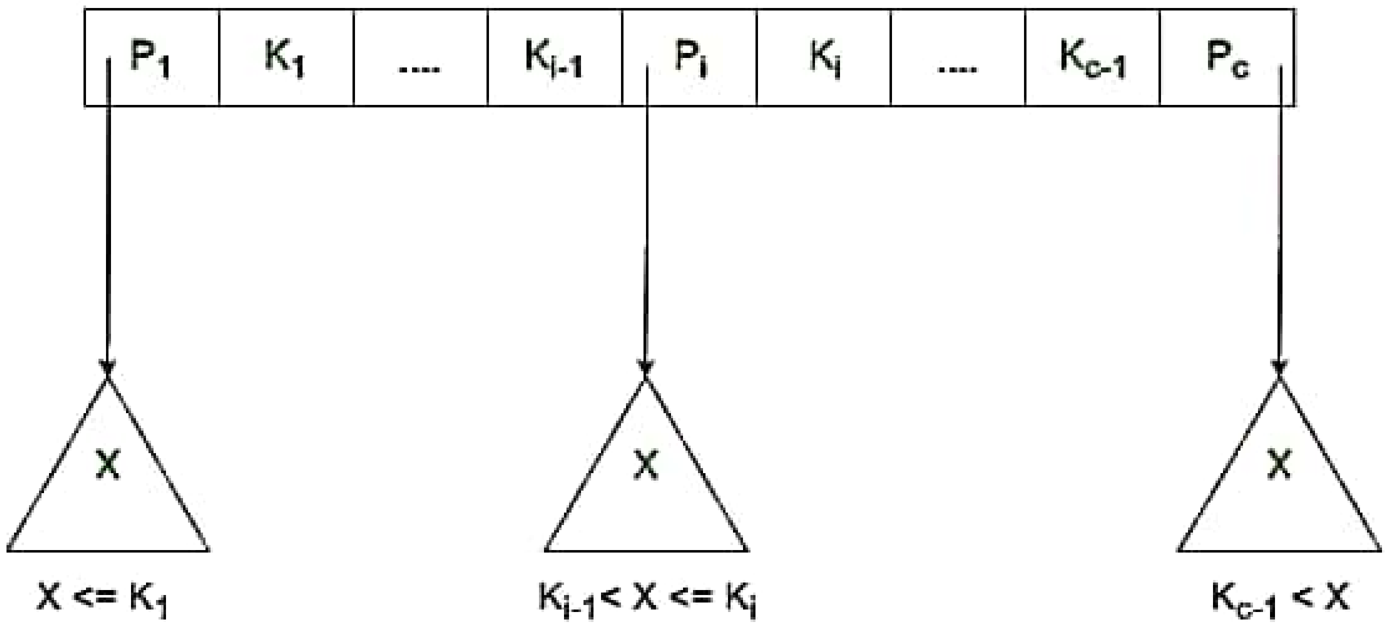
1) Each internal node is of the form :-

   <P_1, K_1, P_2, K_2, ..., P_{c-1}, K_{c-1}, P_c>

   where c <= a and each P_i is a tree pointer and, each K_i is a key value
2) Every internal node has : $K_1 < K_2 < .... < K_{c-1}$
3) For each search field values 'X' in the sub-tree pointed at by P_i, the following condition holds :-

   $K_{i-1} < X <= K_i$, for $1 < i < c$ and,

   $K_{i-1} < X$, for $i = c$
4) Each internal nodes has at most 'a' tree pointers.
5) The root node has, at least two tree pointers, while the other internal nodes have at least \ceil(a/2) tree pointers each.
6) If any internal node has 'c' pointers, c <= a, then it has 'c – 1' key values.



(Internal Node in B+ Tree)

structure of the leaf nodes of a B+ tree

1) Each leaf node is of the form :

   <<K1, D1>, <K2, D2>, ….., <Kc-1, Dc-1>, Pnext>

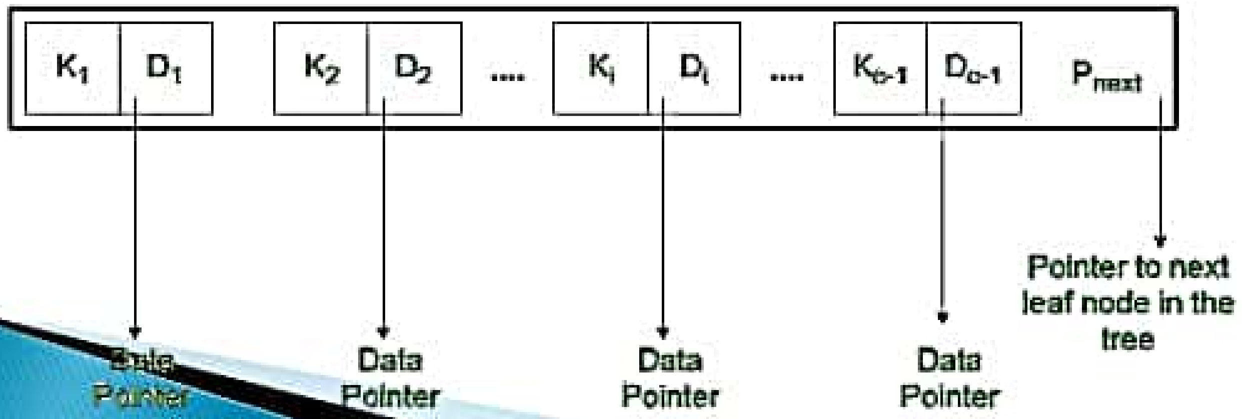
   where c <= b and each Di is a data pointer and, each Ki is a key value and

   Pnext points to next leaf node in the B+ tree

2) Every leaf node has : K1 < K2 < …. < Kc-1, c <= b

3) Each leaf node has at least \ceil(b/2) values.

4) All leaf nodes are at same level.



Example of B+ tree