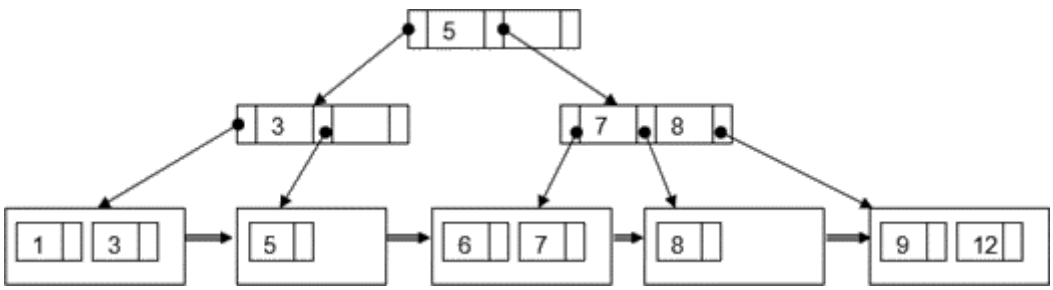
B + Trees: (Computer Science) Examples and Information for Competitive Exams

- Most implementations use the B-tree variation, the B +-tree.
  - In the B-tree, every value of the search field appears once at some level in the tree, along with the data pointer to the record, or block where the record is stored.
  - In a B + tree, data pointers are stored only at the leaf nodes, therefore the structure of the leaf nodes vary from the structure of the internal (non leaf) nodes.
  - If the search field is a key field, the leaf nodes have a value for every value of the search field, along with the data pointer to the record or block.
  - If the search field is a non key field, the pointer points to a block containing pointers to the data file records, creating an extra level of indirection (similar to option 3 for the secondary indexes)
  - The leaf nodes of the B + Trees are linked to provide ordered access on the search field to the record. The first level is similar to the base level of an index.
  - Some search field values in the leaf nodes are repeated in the internal nodes of the B + trees, in order to guide the search.

B + Tree Example



B + Tree Internal Node Structure

Each internal node is of the form $< P1, K1, P2, K2, \ldots, \text{Pq} - 1, \text{Kq} - 1, \text{Pq} >$, where $q <= p$ and each P_i is a tree pointer.

1. Within each internal node, $K1 < K2 < \ldots < \text{Kq} - 1$.
2. For all search field values X in the subtree pointed at by Pi, we have:
3. $\text{Ki} - 1 < X <= \text{Ki}$ for $1 < i < q$;
4. $X <= K$ for $i = 1$;
5. and $\text{Ki} - 1 < X$ for $i = q$.
6. Each internal node has at most, p tree pointers.
7. Each internal node, except the root, has at least $\left\lceil \frac{p}{2} \right\rceil$ tree pointers. The root node has at least two tree pointers if it is an internal node.
8. An internal node with q pointers, $q <= p$, has q-1 search field values.

B + Tree Leaf Node Structure

1. Each leaf node is of the form,
   $\ll K1, \text{Pr}\,1 >, < K2, \text{Pr}\,2 >, \ldots, < \text{Kq} - 1, \text{Prq} - 1 >, \text{Pnext} >$ where $q <= p$, each Pr_i is a data pointer, and P_{next} points to the next leaf node of the B + tree.
2. Within each leaf node, $K1 < K2 < \ldots < \text{Kq} - 1, q <= p$
3. Each Pr_i is a data pointer that points to the record whose search field value is Ki, or to a file block containing the record (or a block of pointers if the search field is not a key field)
4. Each leaf node has at least $\left\lceil \frac{p}{2} \right\rceil$ values.
5. All leaf nodes are at the same level.

B + Tree Information

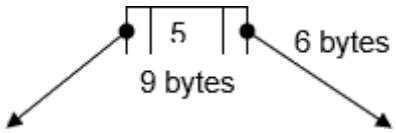
- By starting at the leftmost block, it is possible to traverse leaf nodes as a linked list using the P_{next} pointers. This provides ordered access to the data records on the indexing field.
  - Entries in internal nodes of a B + tree include search values and tree pointers, without any data pointers, more entries can be stored into an internal node of a B + tree, than for a B-tree.
  - Therefore the order p will be larger for a B + tree, which leads to fewer B + tree levels, improving the search time.

- The order p can be different for the internal and leaf nodes, because of the structural differences of the nodes.

Example 6 from Text

To calculate the order p of a B + Tree. suppose the search key field is V = 9 bytes long, the block size is B = 512 bytes, a record pointer is Pr = 7 bytes and a block pointer is P = 6 bytes. An internal node of the B + trees can have up to p tree pointers and p – 1 search field values, which must fit into a single block.

Calculate the value of p for an internal node:



$$p \times P + (p - 1) \times V <= 512$$

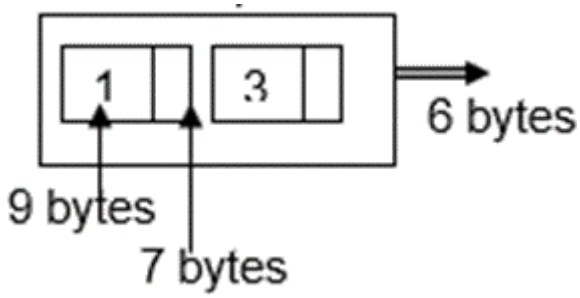
$$p \times 6 + (p - 1) \times 9 <= 512$$

$$6p + 9p - 9 <= 512$$

$$15p <= 522$$

$p = 34$ which means that each internal node can hold up to 34 tree pointers, and 33 search key values.

Calculate the value of p for a leaf node:



$$(\text{pleaf}) \times ((\text{Pr} + V)) + P \leqslant 512$$

$$16\text{pleaf} + 6 <= 512$$

$$\text{pleaf} <= \frac{506}{16}$$

pleaf = 31 which means each leaf node can hold up to p_{leaf} = 31 value/data pointer combinations, assuming data pointers are record pointers.

Example 7 from Text

Suppose that we construct a B + tree on the field of Example 6. To calculate the approximate number of entries of the B + tree we assume that each node is 69 percent full. On average, each internal node will have 34×0.69 or approximately 23 pointers, and hence 22 values. Each leaf node, on the average will hold $0.69 \times \text{pleaf} = 0.69 \times 31$ or approximately 21 data record pointers. A B + tree will have the following average number of entries at each level.

Root: 1 node 22 entries 23 pointers
Level 1: 23 nodes 506 entries 529 pointers
Level 2: 529 nodes 11,638 entries 12,167 pointers
Leaf Level: 12,167 nodes 255,507 record pointers
Count of Nodes

When we compare this result with the previous B-tree example (Example 5) , we can see that the B + tree can hold up to 255,507 record pointers, whereas a corresponding B-tree can only hold 65,535 entries.

Insertion and Deletion with B +-trees.

The following example has $p = 3$, and $\text{pleaf} = 2$

Points to Note:

- Every key value must exist at the leaf level, because all data pointers are at the leaf level,
  - Every value appearing in an internal node, also appears as the rightmost value in the leaf level of the subtree pointed at by the tree pointer to the left of the value.
  - When a leaf node is full, and a new entry is inserted there, the node overflows and must be split. The first j = (p_{leaf} + 1) /2 entries (in the example 2 entries) in the original node are kept there, and the remaining entries are moved to the new leaf node. The entry at position j is **copied/replicated** and moved to the parent node.
  - When an internal node is full, and a new entry is to be inserted, the node overflows and must be split into 2 nodes. The entry at position j is **moved** to the parent node. The first j-1 entries are kept in the original node, and the last j + 1 entries are moved to the new node.

To practice B + Tree insertion, complete Exercise 14.15 in Chapter 14 of the course text.

Developed by: Mindsprite Solutions