B-Trees

March 4, 2017

1 B-Trees

B-Trees are useful when the data doesn't fit in memory. B-Trees is a data structure that minimizes I/O operations. Let's see how B-Trees does it. Many database systems use B-Trees of variant of B-Trees to store information.

1.1 Definition:

We store keys in B-Trees. We also store pointers to the actual data pages besides the key. For this discussion we are going to assume that the data is the keys stored in a node of a B-Tree. A non empty B-Tree has at least 1 key in the root.

If T is the B-Tree then T.root is it's root. Let's assume that x is one of the nodes in the B-Tree:

1. x.n is the number of keys stored in node x
2. $x.key_1$, $x.key_2$, $x.key_3$, ..., $x.key_{x.n}$ are the keys stored in node x
3. x.leaf is a boolean that tells whether x is a leaf.
4. x also contains (x.n+1) pointers to it's children say $x.c_1$, $x.c_2$, ..., $x.c_{x.n+1}$
5. All leaves have the same depth, which is the height of the tree.

With the exception of the root any node x has minimum t-1 keys or maximum 2t-1 keys where t is the degree of the B-Tree and t>=2. t is called the **minimum degree** of the B-Tree and the lower and upper bound on the number of keys that can be stored in a B-Tree is mentioned in terms of t.

We say a node x is full when it exactly has 2t-1 keys

If the B-Tree is non-empty then root of the B-Tree must have at lease 1 key, no matter what value t has

2 Worst case height of a B Tree

A B-Tree of height h has **at least** 2 nodes at depth 1.

has atleast (t-1) keys in any node besides the root. Root has at least 1 key.

So, at depth 2, there are 2t nodes

at depth 3, there are $2t^2$ nodes

at depth h, there are $2t^{h-1}$ nodes

if n is the total number of keys stored in the tree, that means:

$n \leq 1 + (t-1)\sum_{i=1}^{h} 2^{i-1}$

$\leq 1 + 2(t-1)\dfrac{(t^h - 1)}{t-1}$

$$\leq 2^h - 1$$

$$h \geq \log_t \frac{n+1}{2}$$