B树:就是B-树.

此处为二叉搜索树

1.所有的非叶子节点之多拥有两个儿子,

2.所有节点存储一个关键字

3.非叶子节点的左指针指向小于其关键字的子树,右指针指向大于其关键字的子树.

查询的关键字与子树相同,命中子树,如果小,进入左子节点;如果大,进入右子节点;如果为空,报错.

如果BS树的所有非叶子节点的左右子树的节点数目差不多,搜索性能逼近二分法查找,二分法查找需要连续的内存空间, 插入或删除节点,不需要移动打断的内存数据,通常是常数消耗.

BS树经过多次增删后,有可能导致不同结构,使用BS树尽可能要保证平衡的问题.平衡算法是在BS树种插入和删除节点的策略.

B-树:并不是二叉树

是一种多路搜索树

1.定义任意非叶子节点最多只有M(>2)个节点

2.根节点的儿子数为[2,M]

3.除根节点的非叶子节点的儿子数[M/2,M]

4.每个节点存放至少m/2-1个和之多M-1个关键字,至少两个

5.非叶子节点的关键字个数=指向儿子的指针个数-1

6.非叶子节点的关键字

7.非叶子节点的指针

8.所有叶子节点位于同一层

查找:从根节点开始,对应节点的关键字有序进行二分查找,命中,否则进入查询关键所属范围的儿子节点,直至对应的儿子指针为空,或已经是叶子节点,

B-树的特性:

1.关键字集合分布在整棵树中

2.任何一个关键字出现只出现在一个节点中

3.搜索有可能在非叶子节点结束

4.搜索性能等价于在关键字全集内做一次二分查找

5.自动层次控制

B+树:

是B-树的变体,也是一种多路搜索树.

1.定义与B-树相同

2.非叶子节点的指针和关键字相同

3.非叶子节点\_\_\_

4.B-树是开区间

5.为所有叶子节点曾建一个连指针

6.所有的关键字都会在叶子节点出现

查找:B+树和B-树相同,B+树只有达到叶子节点才命中,其性能也等价于对关键字合集做一次二分查找

特性:所有的关键字都出现在叶子节点的链表中,链表中的关键字恰好是有序的.

2.不可能在非叶子节点命中

3.非叶子节点相当于叶子节点的索引,叶子节点相当与关键字的数据的数据层,

4.适合文件索引系统.

B\*树:

是B+ 树的变体,在B+树的非根和非叶子节点增加执行兄弟的指针.

最低使用率2/3

B+树的分裂:B+树,节点如果满时,将源节点1/2数据复制到新节点,在父节点中增加新节点的指针,

B\*的分裂:一个节点满时,下一个兄弟节点未满,那么将一部分数据移动到兄弟节点中,在源节点中插入关键字,最后修改父节点中兄弟节点的关键字.

如果兄弟节点中也满了,在源节点与兄弟节点中增加新节点,并各复制1/3的数据到新节点,最后在父节点中增加新节点的指针.

所以,B\*树分配新节点的概率要比B+树低,空间使用率更高.

BST:二叉搜索树,每个节点只存储一个关键字,等于命中,小左,大右

B树就是B-树:多路搜索树,,所有关键字只出现一次,非叶子节点可以命中

B+树:为叶子节点增加了链表指针,所有关键字在叶子节点中中出现,非叶子节点作为叶子节点的索引,总是在叶子节点才会命中

B\*树:在B+树的基础上,为非叶子节点增加了链表指针,节点的利用率从1/2提高到2/3.