Atitit 范围索引btree的查找原理

应该是一个树节点搜索，，遍历

BTree的机构下，就可以使用二分查找的查找方式，查找复杂度为h\*log(n)，一般来说树的高度是很小的，一般为3左右，因此BTree是一个非常高效的查找结构。

带顺序索引的B+TREE很多存储引擎在B+Tree的基础上进行了优化，**添加了指向相邻叶节点的指针**，形成了带有顺序访问指针的B+Tree，这样做是为了提高区间查找的效率，只要找到第一个值那么就可以顺序的查找后面的值。

B+树

在B树中，我们在查找的时候往返于每个结点之间就意味着，我们必须要在硬盘的页面之间进行多次访问。这样就非常糟糕。有没有可能让遍历时每个元素只访问一次呢？

我们在原有B树的结构基础上，加上了新的元素组织方式，这就是B+树。

B+树是应文件系统所需而出的一种B树的变形树。在B树中，每一个元素在该树中只出现一次，有可能在叶子结点上，也有可能在分支节点上。而在B+树中，出现在分支结点中的元素会被当作它们在该分支结点位置的中序后继者（叶子结点）中再次出现。而且，每一个叶子结点都会保存一个指向后一个叶子结点的指针。

BTree结点的定义

template<class K,class V,size\_t M>

//非类型的模板参数在推演时变成常量

//M阶树的孩子个数为 [M/2,M] 个孩子

//M阶树的关键字个数为 [M/2-1,M-1] 个关键字,且为升序排列

struct BTreeNode

{

//将数组的大小预留出一位是为了便于分裂

pair<K, V> \_kvs[M]; //kvs数组,存放kv

BTreeNode<K, V, M>\* \_parent; //指向父结点的指针，便于分裂

BTreeNode<K, V, M>\* \_subs[M + 1]; //指向孩子结点的指针数组

size\_t \_size; //kvs数量,有效数据个数

BTreeNode()

:\_parent(NULL)

, \_size(0)

{

for (size\_t i = 0; i < M + 1; i++)

{

\_subs[i] = NULL;

}

}

————————————————

（1-1kw） （1kw-2kw） .....(9kw--10kw)

(1-100w)(100w-200w)

(1-10w)....

1-1w

1-1k

分裂过程

一千内

1-1000

二千内--一万

1-10000

（1-1000）（1001-1999）

2w1000

1-10w

(1-10000) (10001-19999)....

（1-1000）（1001-1999）..

1500

BTreeNode

dataRagin( 1-10w)

subNodes[ sn1-1w,sn2-3w.....]

dataNode(nodenum,priKeyid)

15005 find node(1w,5k

插入就是找到节点，放入子内容

查找就是查询子内容

————————————————

BTree的插入和查找算法分析 - sky\_Mata的博客 - CSDN博客.html

BTree —— 多路查找树 - ling\_hun\_pang\_zi的博客 - CSDN博客.html