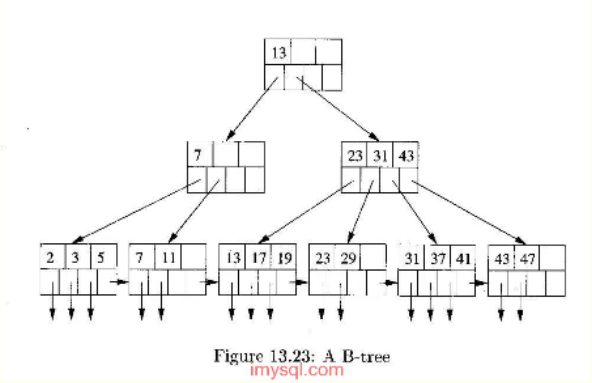
Mysql里常用的索引数据结构有B+树索引和哈希索引，两种索引数据结构的区别及其不同的应用建议

备注：先说下，**在MySQL文档里，实际上是把B+树索引写成了BTREE**，例如像下面这样的写法：

```
CREATE TABLE t(
aid int unsigned not null auto_increment,
userid int unsigned not null default 0,
username varchar(20) not null default '',
detail varchar(255) not null default '',
primary key(aid),
unique key(uid) USING BTREE,
key (username(12)) USING BTREE — 此处 uname 列只创建了最左12个字符长度的部分索引
)engine=InnoDB;
```

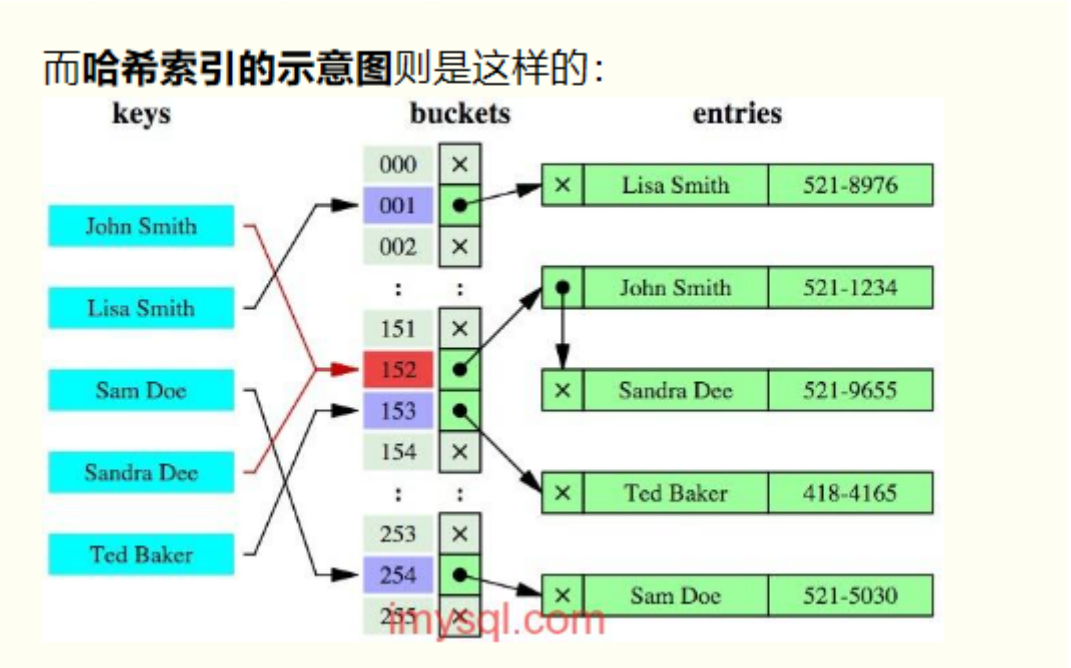
一个经典的**B+树索引数据结构**见下图：



(图片源自网络)

B+树是一个平衡的多叉树，从根节点到每个叶子节点的高度差值不超过1，而且同层级的节点间有指针相互链接。

在B+树上的常规搜索，从根节点到叶子节点的搜索效率基本相当，不会出现大幅度的波动，而且基于索引的顺序扫描时，也可以利用双向指针快速左右移动，效率非常高。因此B+树索引被广泛用于数据库、文件系统等场景。



简单的说，哈希索引就是采用一定的hash算法，把键值换成新的hash值，检索时不需要类似B+树那样从根节点到叶子节点逐级查找，只需一次hash算法即可立刻定位到相应的位置，速度非常之快。

从上图来看，B+树索引和hash索引的明显区别就是：

1.如果是等值查询，那么hash索引明显有绝对优势，因为只需要经过一次算法即可找到相应的键值；当然了，这个前提是，键值是唯一的。如果键值不是唯一的，就需要先找到键值所在位置，然后再根据链表往后扫描，直到找到相应的数据。

2.从示意图中也可以看到，如果是范围查询检索，这时候哈希索引就毫无用武之地了，因为原先是有序的键值，经过哈希算法后，有可能变成不连续的了，这就没有办法再利用索引完成范围查询检索。

3.同时，哈希算法也没有办法利用索引完成排序，模糊查询也不行。

4.B+树索引的关键字检索效率比较平均，在有大量重复键值的情况下，哈希索引的效率也是很低的，因为存在所谓的哈希碰撞问题。