https://mp.weixin.qq.com/s/kPWvgvYsz1b7cpaFUeSCAg

# **索引的本质**

在没有索引的情况下，我们查找数据只能按照从头到尾的顺序逐行查找，每查找一行数据，意味着我们要到到磁盘相应的位置去读取一条数据。

如果把查询一条数据分为到磁盘中查询和比对查询条件两步的话，到磁盘中查询的时间会远远大于比对查询条件的时间，这意味着在一次查询中，磁盘io占用了大部分的时间。更进一步地说，一次查询的效率取绝于磁盘io的次数，如果我们能够在一次查询中尽可能地降低磁盘io的次数，那么我们就能加快查询的速度。

在知道了减少磁盘io能加快查询速度后，我们就要聚焦于如何减少磁盘io。如果按照原表逐行查询的话，n条数据就要查询n次，也就是O(N)的时间复杂度，为了减少磁盘io的次数，我们必须用一种查询时间复杂度更低的数据结构来保存数据。

这种查询时间复杂度低的数据结构，我们称之为索引。所以通俗来说，索引其实就是某种数据结构，能充当索引的数据结构是多种多样的。

# **索引的选择**

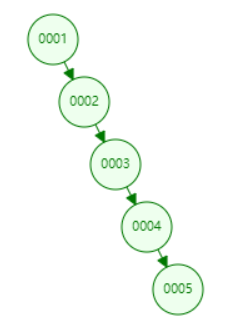
既然索引是一种便于查询的数据结构，如果大家对数据结构有一定了解的话，大概率会首选树型结构。毕竟树型结构普遍有着O(logN)的查询时间复杂度，而且插入删除数据的性能也比较平均。（可能你会说数组，哈希表的查询速度也很高啊，这个后面也会分析）



虽然我们都已经知道Mysql中最常用的引擎像InnoDB和MyISAM，最终都选择了B+树作为索引，但是这里我还是打算从最常见的二叉树开始讲起，推导一下为什么最终选择了B+树作为索引，并比较一下几种树型结构在充当索引时的优劣。

## **二叉树**

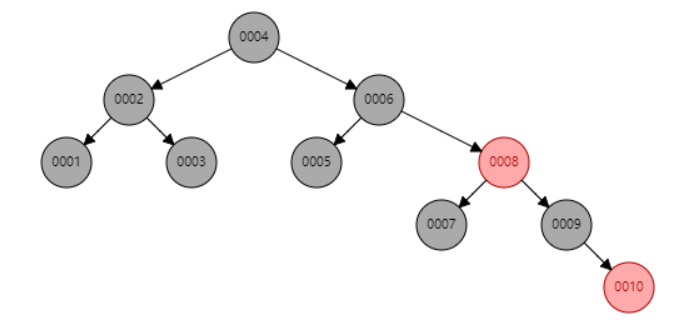
最普通的二叉树的问题在于他不能保证O(logN)的查询时间复杂度，我们看下面的图：



由于插入的元素逐渐增大，元素始终在右边进行插入，好好的一棵二叉树最终变成了一条“链表”。在这种极端的情况下，二叉树的查询时间复杂度不再是O(logN)，而是退化为O(N)，这样显然不符合索引的要求。

## **平衡二叉树（红黑树）**

像红黑树这样的平衡二叉树，无论如何插入元素，他都可以通过一些旋转的方法调整树的高度，使得整棵树的查询效率维持在O(logN)，如下图所示：

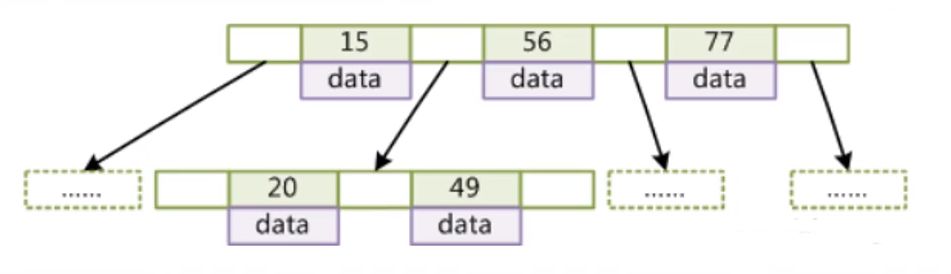


这么来说他已经符合了成为索引的必备条件，但是最终没有选择他作为索引说明还有不足的地方。仔细看看可以发现平衡二叉树的每个节点只有两个孩子节点，如果一张表的数据量特别大，整棵树的高度也会随之上升。一个千万级别的表如果用平衡二叉树作为索引的话，树高将会达到二十多层。这也就意味着做一次查询需要二十多次磁盘io，这是一个不小的开销。

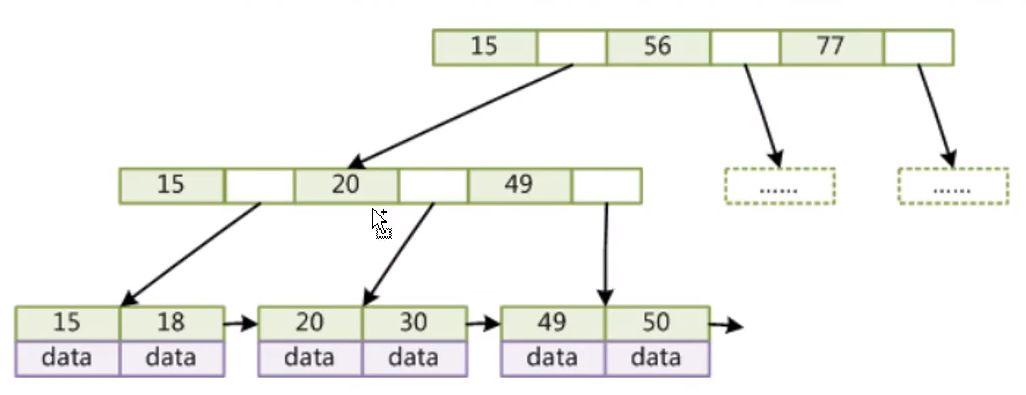
那么有没有能在大数据量的情况下，还能保持一个较小树高的树型结构呢？

## **B树和B+树**

答案就是B树。上面我们说到了平衡二叉树的瓶颈在于一个节点只有两个孩子节点，而B树一个节点可以存放N个孩子节点，这就完美解决了树高的问题，我们可以把B树称为平衡多叉树，B树作为索引如下图所示：



但是以B树的结构作为索引仍有可以优化的地方，我们先看看最终的B+树，再仔细分析B+树在B树的基础上作了哪些改进，为什么B+树最终能够胜任索引的工作：



从图片中可以看到B+树同样是一棵多差平衡树，和B树一样很好地解决了树高的问题。

### **改进点一：**

但仔细看可以发现，B树的节点中既存储索引，也存储表对应的数据；而B+树的非叶子节点是不存储数据的，只存储索引，数据全部存储在叶子节点上。

为什么要做这样的改进？我们做一次算术就知道了。

假设树高为2，主键ID为bigint类型，长度为8字节，节点指针为6字节，一行数据记录的大小为1k，一次io操作能获得一页16k的数据。

在索引为B+树的情况下，根节点能存储：16k / （6 + 8） = 1170 条索引指针；到了第一层，一共能指向 1170 \* 1170 = 1368900 条索引指针；到了最底一层叶子节点，一个节点能存储16k / 1k = 16 条记录，一共能存储 1170 \* 1170 \* 16 = 21902400 条记录

在B树的情况下，由于非叶子节点使用了大量空间存储数据，存放的索引指针肯定就少，最终整棵树如果想要存储和B+树一样多的数据就必须要增加树高，这样一来就增加了磁盘io，所以说B+树作为索引的性能比B树高。

### **改进点二：**

叶子节点之间使用指针连接，提高区间访问效率。如果我们要进行范围查询，可以轻松通过B+树叶子节点之间的指针进行遍历，减少了不必要的磁盘io。

# **总结**

看到这里，相信大家对为什么Mysql的常用引擎都默认使用B+树作为索引已经有了初步的认知。我们只要牢记一点：索引是为了减少磁盘io提高查询性能而存在的。

最后回应一下为什么不常用哈希表和数组作为索引

哈希表虽然单一个值的查询效率很高，但是撑不住范围查询，哪个公司的业务还没个范围查询呢？

而数组虽然查询的效率高，但是增加和删除的效率低，由于记录在增加和删除的时候索引也得跟着维护，这会导致大数据量的情况下，增加或删除一条记录效率较低。