一. 索引定义

MySQL官方对索引的定义为：索引（Index）是帮助MySQL高效获取数据的数据结构。索引的本质：索引是数据结构。

我们知道，数据库查询是数据库的最主要功能之一。我们都希望查询数据的速度能尽可能的快，因此数据库系统的设计者会从查询算法的角度进行优化。最基本的查询算法当然是顺序查找（linear search），这种复杂度为O(n)的算法在数据量很大时显然是糟糕的，好在计算机科学的发展提供了很多更优秀的查找算法，例如二分查找（binary search）、二叉树查找（binary tree search）等。如果稍微分析一下会发现，每种查找算法都只能应用于特定的数据结构之上，例如二分查找要求被检索数据有序，而二叉树查找只能应用于二叉查找树上，但是数据本身的组织结构不可能完全满足各种数据结构（例如，理论上不可能同时将两列都按顺序进行组织），所以，在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引。

二. B树 & B+树

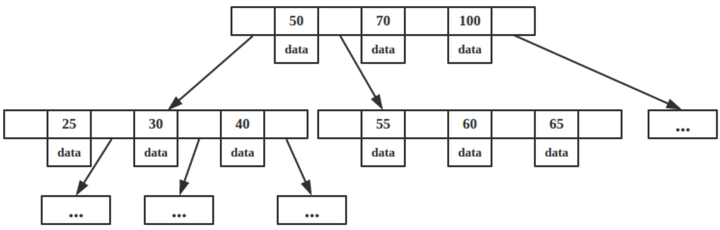
目前大部分数据库系统及文件系统都采用B-Tree或其变种B+Tree作为索引结构。

先看一下为什么会出现B树这种数据结构

传统用来搜索的平衡二叉树有很多，如 AVL 树，红黑树等。这些树在一般情况下查询性能非常好，但当数据非常大的时候它们就无能为力了。原因当数据量非常大时，内存不够用，大部分数据只能存放在磁盘上，只有需要的数据才加载到内存中。一般而言内存访问的时间约为 50 ns，而磁盘在 10 ms 左右。速度相差了近 5 个数量级，磁盘读取时间远远超过了数据在内存中比较的时间。这说明程序大部分时间会阻塞在磁盘 IO 上。那么我们如何提高程序性能？减少磁盘 IO 次数，像 AVL 树，红黑树这类平衡二叉树从设计上无法“迎合”磁盘。

1. **B树**

   B树,这里的 B 表示 balance(平衡的意思),B-树是一种多路自平衡的搜索树，下面是B树简化图



B-树

特点：

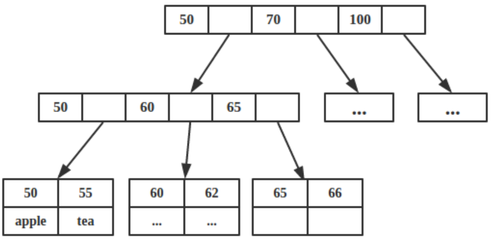
- 每个非叶子节点由n-1个key和n个指针组成，其中d<=n<=2d （d为B树的度）
- 每个叶子节点最少包含一个key和两个指针，最多包含2d-1个key和2d个指针，叶节点的指针均为null
- 所有叶节点具有相同的深度，等于树高h
- key和指针互相间隔，节点两端是指针
- 一个节点中的key从左到右非递减排列
- 每个指针要么为null，要么指向另外一个节点
  - 如果某个指针在节点node最左边且不为null，则其指向节点的所有key小于v(key1)v(key1)，其中v(key1)v(key1)为node的第一个key的值
  - 如果某个指针在节点node最右边且不为null，则其指向节点的所有key大于v(keym)v(keym)，其中v(keym)v(keym)为node的最后一个key的值
  - 如果某个指针在节点node的左右相邻key分别是keyikeyi和keyi+1keyi+1且不为null，则其指向节点的所有key小于v(keyi+1)v(keyi+1)且大于v(keyi)v(keyi)

2. **B+树**

   B+树是B树的变种，它与B树区别如下

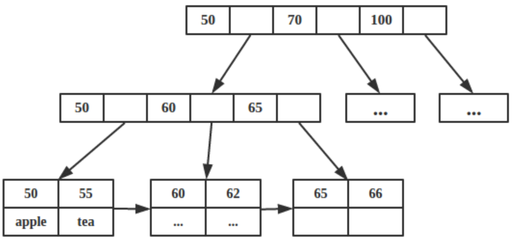
   1). 在B+树中，key 的副本存储在内部节点，真正的 key 和 data 存储在叶子节点上

   2). n 个 key 值的节点指针域为 n 而不是 n+1



B+树

为了增加区间访问效率，会对B+树做一些优化，每个叶子节点添加顺序访问指针



带顺序访问的B+树

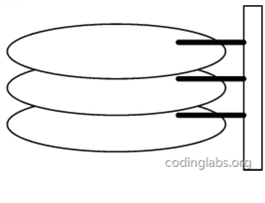
三. 为什么B树&B+树适合做索引结构

一般来说，索引本身也很大，不可能全部存储在内存中，因此索引往往以索引文件的形式存储的磁盘上。这样的话，索引查找过程中就要产生磁盘I/O消耗，相对于内存存取，I/O存取的消耗要高几个数量级，所以评价一个数据结构作为索引的优劣最重要的指标就是在查找过程中磁盘I/O操作次数的渐进复杂度。换句话说，索引的结构组织要尽量减少查找过程中磁盘I/O的存取次数。

1. **磁盘存取原理**

   索引一般以文件形式存储在磁盘上，索引检索需要磁盘I/O操作。与主存不同，磁盘I/O存在机械运动耗费，因此磁盘I/O的时间消耗是巨大的。

   磁盘整体结构图



2. **局部性原理&磁盘预读**

   由于磁盘的存取速度与内存之间鸿沟,为了提高效率,要尽量减少磁盘I/O.磁盘往往不是严格按需读取，而是每次都会预读,磁盘读取完需要的数据,会顺序向后读一定长度的数据放入内存。而这样做的理论依据是计算机科学中著名的局部性原理

   局部性原理：

   1). 当一个数据被用到时，其附近的数据也通常会马上被使用

   2). 程序运行期间所需要的数据通常比较集中

   由于磁盘顺序读取的效率很高（不需要寻道时间，只需很少的旋转时间），因此对于具有局部性的程序来说，预读可以提高I/O效率。预读的长度一般为页(page)的整倍数。

3. **Mysql为什么使用B+树**

   1). B+树更适合外部存储，由于内节点无 data 域，一个结点可以存储更多的内结点，每个节点能索引的范围更大更精确，也意味着 B+树单次磁盘IO的信息量大于B-树，I/O效率更高。

   2). Mysql是一种关系型数据库，区间访问是常见的一种情况，B+树叶节点增加的链指针,加强了区间访问性，可使用在范围区间查询。