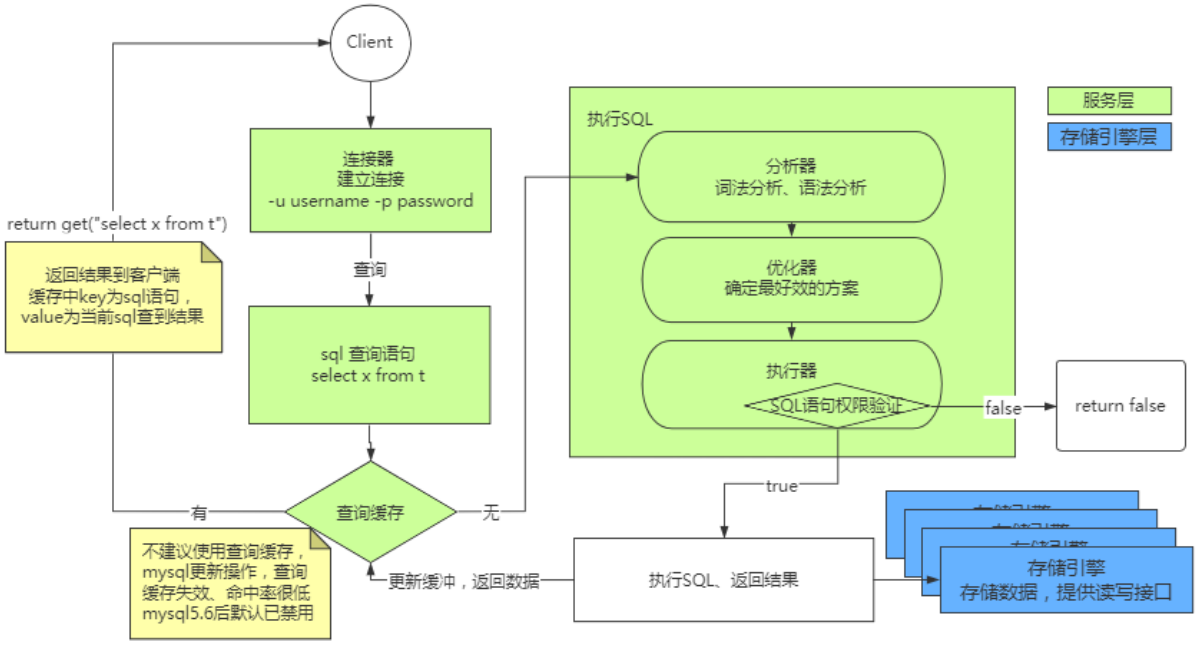
索引本质原理

MySQL服务架构图



mysql 优化之查询缓存(mysql8已经废弃这个功能)

因为数据变动还是比较频繁，所以缓存命中率不是很高，所以可以把有些不是经常变动的数据放在缓存器中

索引

如果建了索引——会根据索引结构去检索数据———效率很高

如果没有索引——会全表扫描

比如：X列是一个索引列，在进行 查找的是后 ，mysql会先判断是否是索引列，如果是索引，则先会从索引结构中定位对应的位置，然后再读取数据

索引为什么要用B+树

为什么不用二叉树做索引结构？

二叉树可能在某一个分支深度会很深（因为是两个叉），类似链表，查询深度很深

为什么不用红黑树？

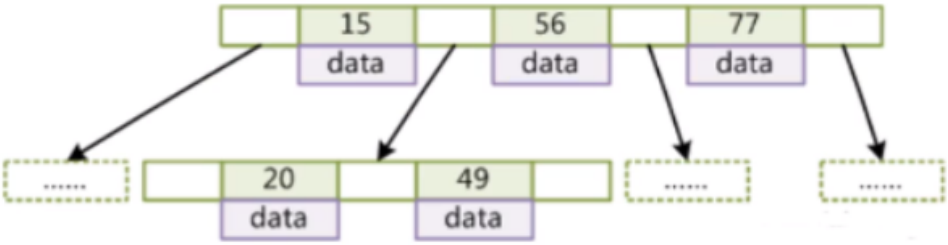
虽然红黑树可以自平衡，但是对于几百万的数据来说深度也很深（两个叉）

为什么不用hash 表？———部分存储引擎索引结构使用的是 hash 一般没人用

因为hash表示索引+链表结构，第一链表深度太深，第二mysql查找数据经常是按照范围查找的，二hash表示等值查找，有确定的数组index才能确定哪一个链表？然后才能遍历，效率低。

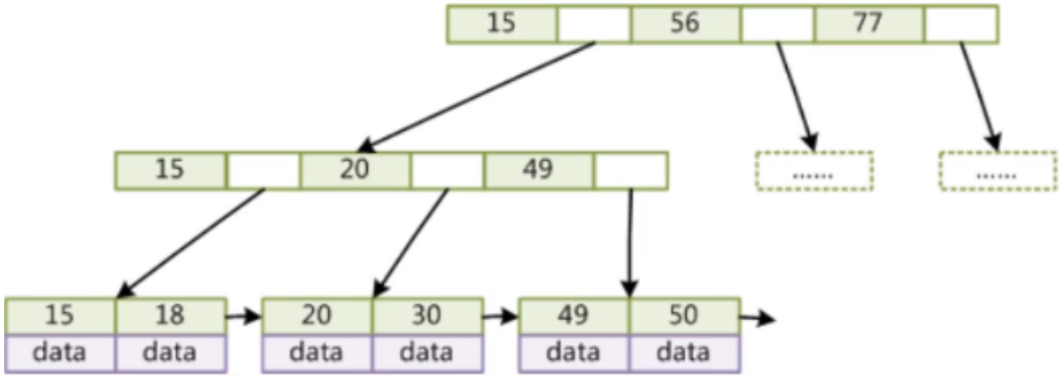
B-Tree

- 叶节点具有相同的深度、叶节点指针为空
- 所有索引元素不重复
- 节点数据索引从左到右递增排列



B+Tree（又叫多叉二叉树）

- 非叶子节点不存储数据，只存储索引，可以放更多的索引（冗余，空间换时间）
- 叶子节点包含所有索引字段
- 叶子节点用指针链接，提高区间访问的性能



B+树是对B树的一个升级改造。

B树：部分非叶子节点存储了一定的数据

B+树：只有叶子节点才能存数据，非叶子节点不可以存取数据

非叶子节点的索引值都是冗余的，方便查找（二分查找）

在叶子节点的索引包括了整张表的所有索引

每一个叶子节点的指针的地址指向主要用于范围查找，即如果查找> 20的list，就可以根据这个指针找到所有的数据

B+树特性

- 每一个节点里面的索引list是从小到大排列的，如 15　20　39
- 叶子节点之间也是从小到大排列的(小二叉树)　如　18<——>20　这个双向指针的目的值为了更好的支持范围查找，B树是不支持这样的范围查找的，它没有这个指针

树的高度

如查找——30

先把根节点一次性load到内存（IO）——Mysql的**根节点**是**常驻内存**的，然后再到内存中做**随机查找**，计算30是属于哪个两个数的范围的时候内存查找，内存查找的效率和IO比起来可以忽略不计，计算到30 是 包含在　15 和 56，而15-56 中间这个元素其实是保存的下一个子节点在磁盘上的地址指针，然后就会根据这个指针再一次load相应的索引节点（又一次IO操作），同理，根据20—49之间的磁盘指针将对应节点的数据都load进内存，这样就会找到 30 对应的数据

InnoDB的缺点：

由于InnoDB的叶子节点中索引和数据是放一起的，这样最后一次的时候可能会load别的数据，可能会将别的无用数据load进内存，然后到内存中再找到确定的值。

而在MyISASM存储引擎中，索引和数据是分开文件保存的，这样在load索引的时候，先根据load到的索引匹配查找需要找的索引，最后根据确定的索引到磁盘中load对应的数据，这样由于在筛选的时候只是筛选索引（占用空间少）——**所以MyISASM存储引擎更适合做大量select操作**

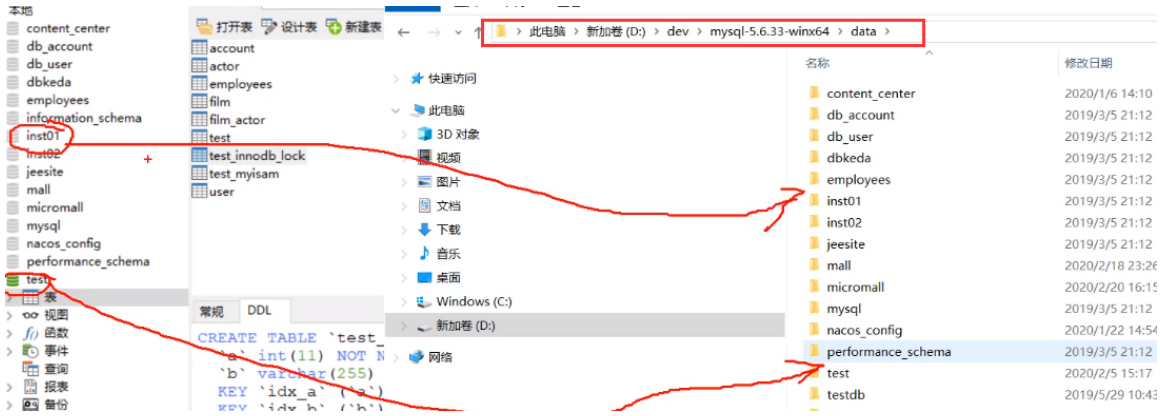
Mysql节点设置

MySQL把一个索引节点默认设置成了16k的大小，允许的设置参数是 4k 8k 16k 32k 64k，如果不是mysql大牛、不要改这个值，

假设以bigint 为索引，即8个字节，而在mysql节点中 索引和索引中间存磁盘指针的大小是6个字节，这样16k的大小可以存放1170 个索引（8字节的索引），理想情况下三四次深度的B+树就可以存放几千万个索引。

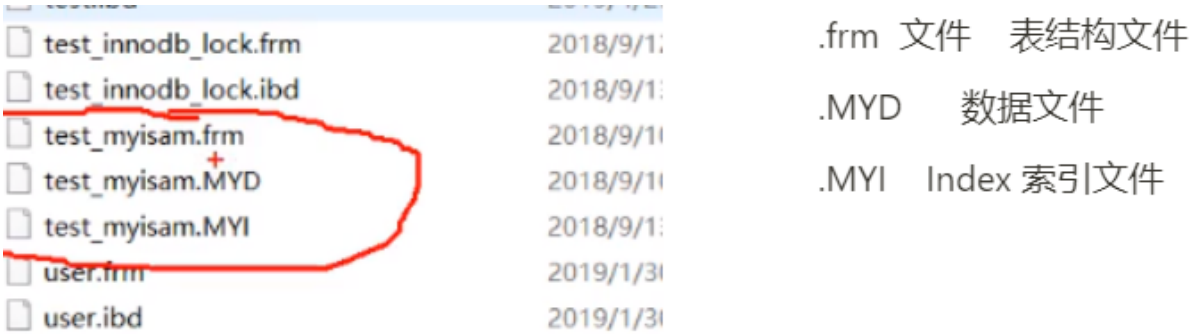
设置成16k的原因是可能还和磁盘的4k对齐有关系（使物理硬盘分区与计算机使用的逻辑分区对齐，保证硬盘读写效率）

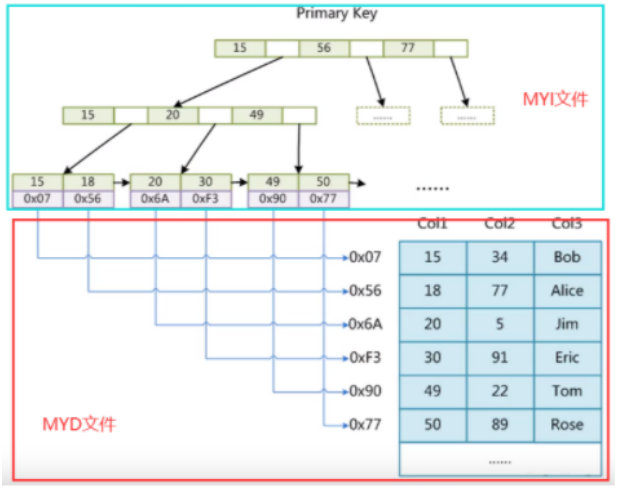
Mysql 数据默认存储位置



MyISAM存储引擎

MyISAM索引文件和数据文件是分离的（非聚集）





where col1 = 30

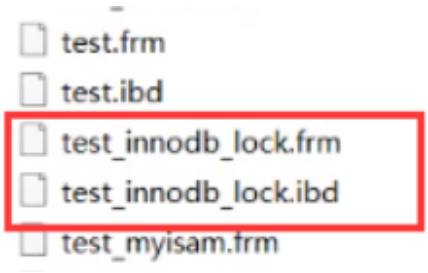
查找的时候会先分析col1是不是索引字段

如果不是，全表扫描

如果是，则从MYI文件中找到对应的磁盘地址

然后根据该磁盘地址找到对应的记录

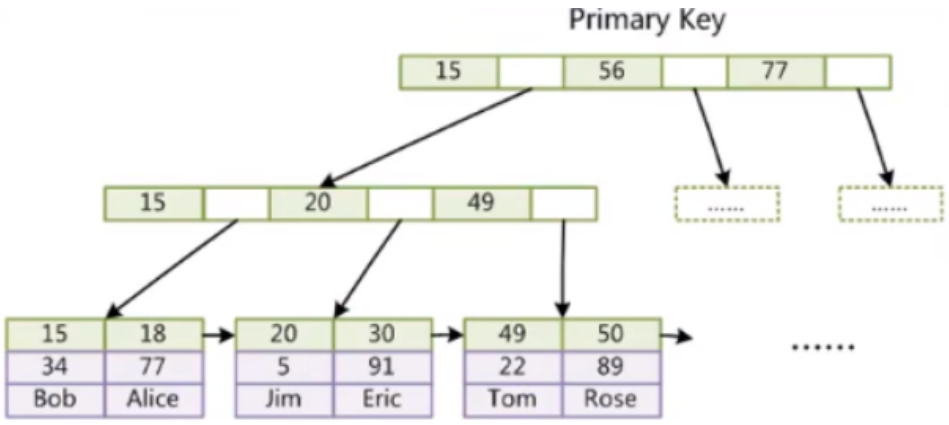
InnoDB存储引擎

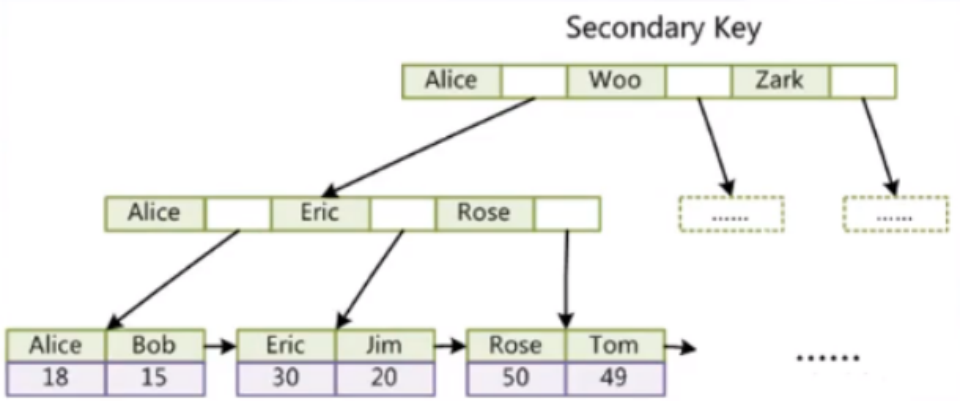


frm 表结构文件

ibd 索引+数据文件

- 表数据文件本身就是按B+Tree组织的一个索引结构文件
- 聚簇索引，叶节点包含了完整的数据记录



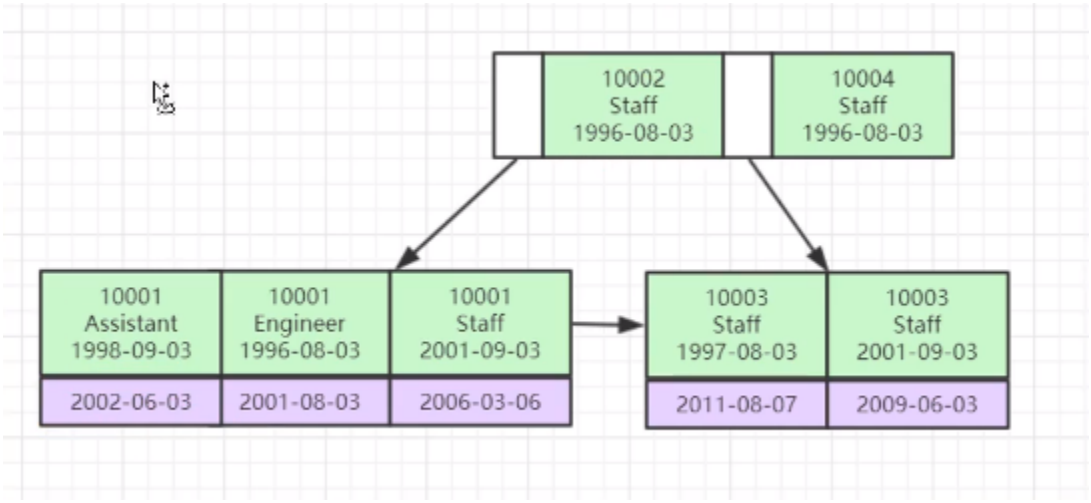


为什么InnoDB表必须有主键，并且推荐使用整形的自增主键？

主键-的目的是组织索引，组织数据，如果不创建主键，mysql会自动找一个数据不重复的字段作为主键索引，如果找不到这样不重复的字段，mysql会自动创建一个默认的隐藏列组织表中每一行数据。

使用自增整形的原因：1、整形在索引节点中比较大小的时候比字符串快，2、字符串占用的空间更大。自增的目的是为了更快的插入到原来已经保存好的节点位置的右面，以适应B+树叶子节点索引从小到大的排列。
如果不是自增的（页分裂），在插入的时候会破会原来节点中元素的位置和额外的平衡树的操作，降低效率，所以自增主键的效率插入的时候更高。

联合索引的底层存储结构长什么样？



联合索引比较大小，首先是和第一个字段比较，即1004 这个字段 大于 1002， 这样就不必要比较后面字段的值了。

越在前面的字段越优先