正在加载一言...

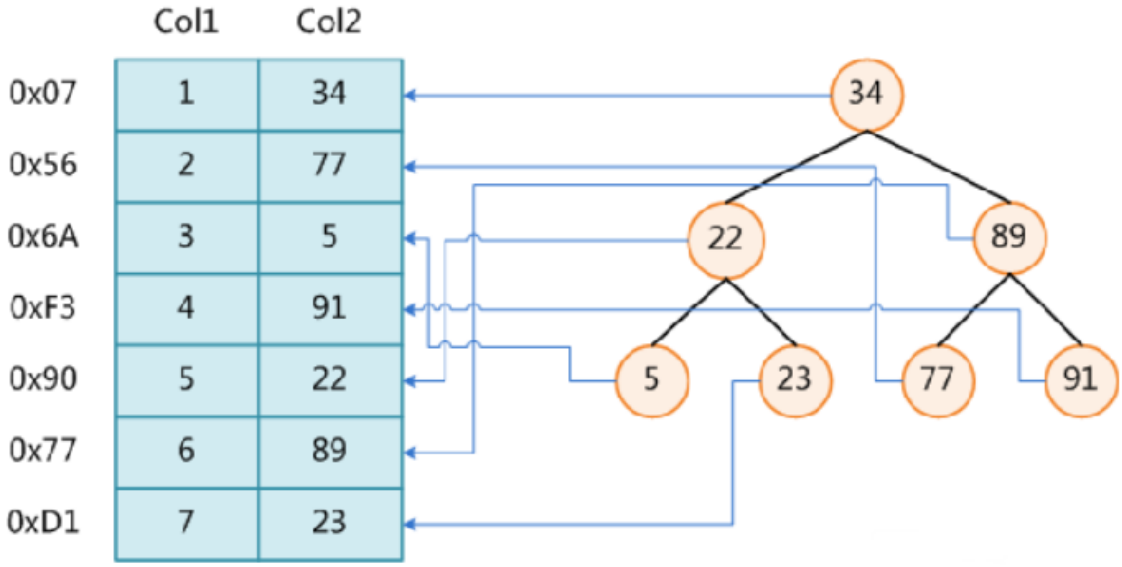
MySQL索引底层数据结构

1、索引的本质

1、索引是帮助MySQL高效获取数据的排好序的数据结构

2、索引的数据结构

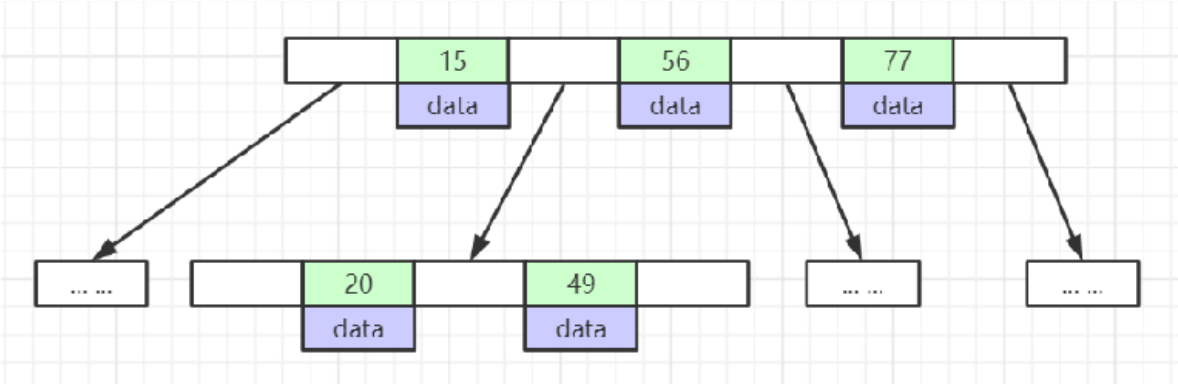
- 二叉树（对于单边增长的列，可能形成链表结构（单边树），导致查询次数很大，效率没有提升，所以不用二叉树做数据结构）
- 红黑树（二叉平衡树，虽然有平衡功能，但是，数据量大时，树的高度会很高，查询次数很大）
- Hash表
- B-Tree



B-Tree：

叶节点具有相同的深度，叶节点的指针为空 所有索引元素不重复 节点中的数据索引从左到右递增排列

一个节点存放多个索引元素，每个元素存放data数据（非叶子节点也存储data数据，获取data磁盘地址）



B+Tree(B-Tree变种)

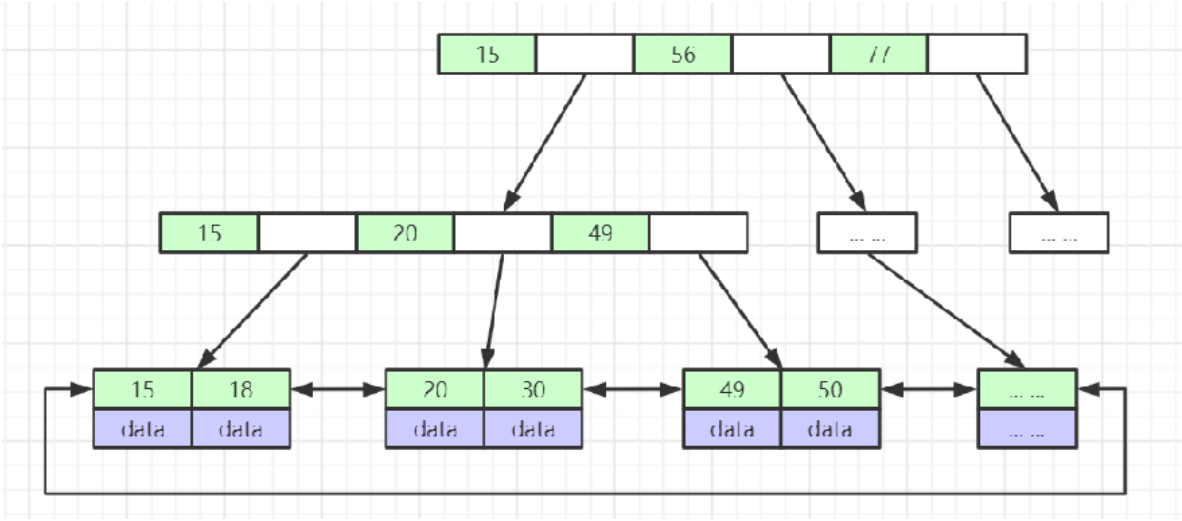
非叶子节点不存储data，只存储索引(冗余)，可以放更多的索引，有效的控制了树的高度 叶子节点包含所有索引字段（非叶子节点为冗余索引，为了构建B+树）　　叶子节点用指针连接，提高区间访问的性能

B+树每个节点大小为一个页（通常为16kb 通过 show global status like 'Innodb_page_size';查询）

相比B Tree改动点：1>非叶子节点不存储数据；2>相邻叶子节点互相指向（其实也算是B+树的变种，B+树本身只是单向指向，方便范围查找）

查找数据过程：

先从磁盘加载根节点到内存（按页取数据，有版本的mysql是直接把根节点或非叶子节点常存在内存中），然后比对定位要查找数据的位置，依次查询下一个节　点，最后定位到叶子节点



Hash

对索引的key进行一次hash计算就可以定位出数据存储的位置 很多时候Hash索引要比B+ 树索引更高效 仅能满足 “=”，“IN”，不支持范围查询 hash冲突问题

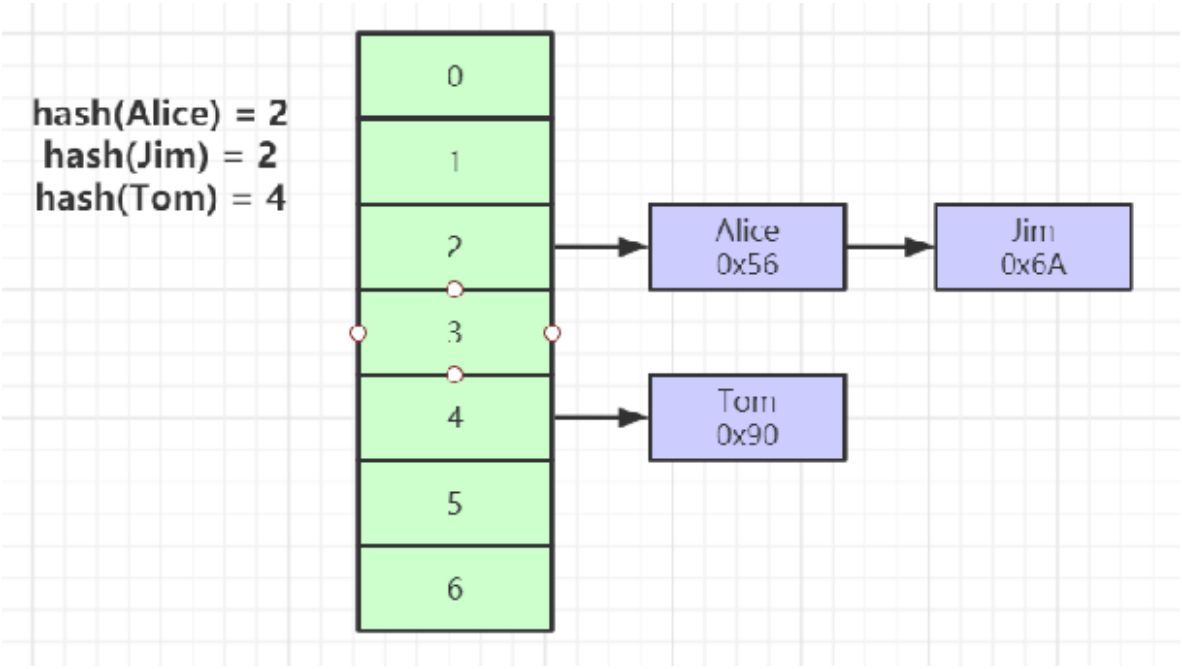
如果是等值查询，哈希索引明显有绝对优势， 前提：键值唯一

哈希索引没办法完成范围查询检索

哈希索引也没办法利用索引完成排序，以及like ‘xxx%’ 这样的部分模糊查询

哈希索引也不支持多列联合索引的

在有大量重复键值情况下，哈希索引的效率也最左前缀原则是极低的，因为存在哈希碰撞问题



3、存储引擎

1. MyISAM存储引擎索引实现

   所谓存储引擎是针对表的而非数据库的

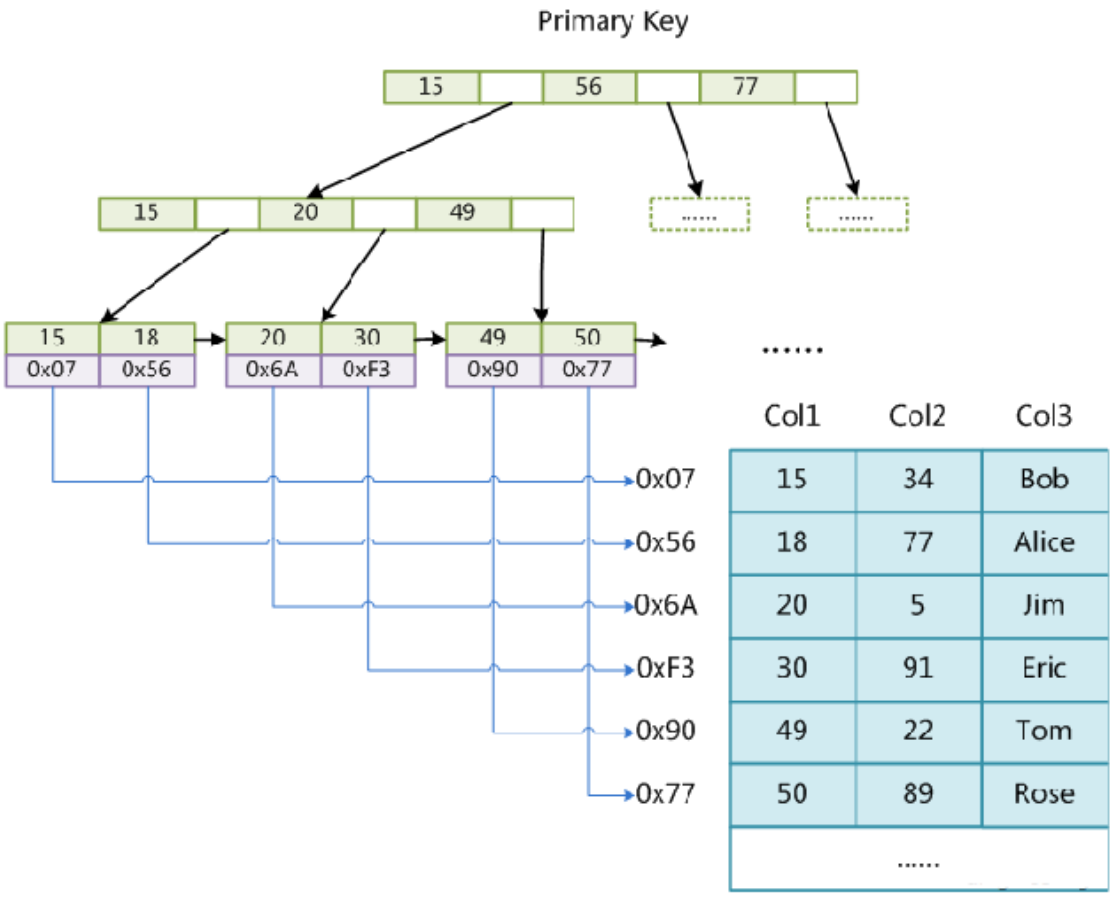
   MyISAM类型引擎的表包含三个文件.frm(框架文件，数据表结构相关的)、.MYD（存储数据的文件）、.MYI（存放索引的文件）

   .MYI文件存储索引数据，MyISAM查找数据时先从索引文件拿到查找值的磁盘地址，然后去.MYD文件去检索数据

   索引和数据不在同一个文件中，因此MyISAM索引被称为非聚集索引

   MyISAM 索引叶子节点data 存储的是索引所在行的磁盘文件地址

   MyISAM索引文件和数据文件是分离的(非聚集)



2. InnoDB存储引擎索引实现

使用InnoDB引擎的表包含两个文件：.frm文件为框架文件，存储数据结构相关信息的、.ibd文件存储索引和数据（索引和数据在同一个文件中，因此InnoDB索引被称为聚集索引）

InnoDB 索引叶子节点data 存储的是索引所在行的是其它列的数据（对于主键索引来说）

InnoDB索引实现(聚集)　表数据文件本身就是按B+Tree组织的一个索引结构文件　聚集索引-叶节点包含了完整的数据记录（针对主键索引来说，二级索引通过回表取数）

3. 问题

问：为什么InnoDB表必须有主键。并且推荐使用整型的自增主键：（整型：索引查找过程中涉及很多次比较，整型比较比字符串比较大小效率高；自增：）

1、如果设置了主键，那么InnoDB会选择主键作为聚集索引、如果没有显式定义主键，则InnoDB会选择第一个不包含有NULL值的唯一索引作为主键索引、如果也没有这样的唯一索引，则InnoDB会选择内置6字节长的ROWID作为隐含的聚集索引(ROWID随着行记录的写入而主键递增)。

2、如果表使用自增主键

那么每次插入新的记录，记录就会顺序添加到当前索引节点的后续位置，主键的顺序按照数据记录的插入顺序排列，自动有序。当一页写满，就会自动开辟一个新的页

3、如果使用非自增主键（如果身份证号或学号等）

由于每次插入主键的值近似于随机，因此每次新纪录都要被插到现有索引页得中间某个位置，此时MySQL不得不为了将新记录插到合适位置而移动数据，甚至目标页面可能已经被回写到磁盘上而从缓存中清掉，此时又要从磁盘上读回来，这增加了很多开销，同时频繁的移动、分页操作造成了大量的碎片，得到了不够紧凑的索引结构，后续不得不通过OPTIMIZE TABLE来重建表并优化填充页面。

问：为什么非主键索引结构叶子结点存储的是主键值：(数据一致性和存储空间)

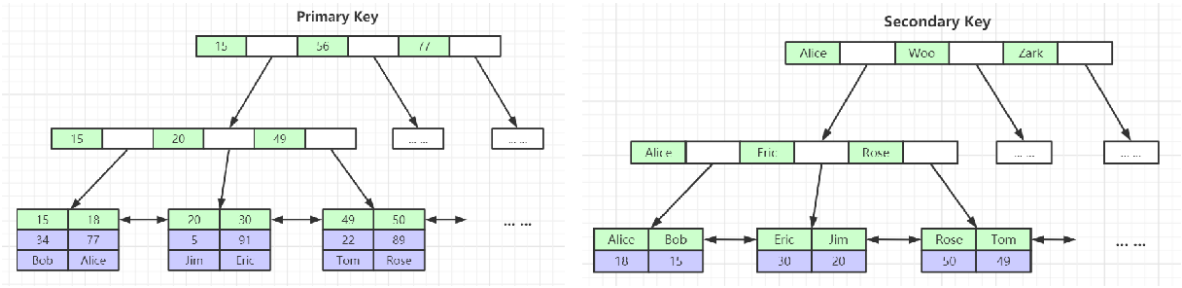
减少了出现行移动或者数据页分裂时二级索引的维护工作（当数据需要更新的时候，二级索引不需要修改，只需要修改聚簇索引，一个表只能有一个聚簇索引，其他的都是二级索引，这样只需要修改聚簇索引就可以了，不需要重新构建二级索引）

聚簇索引也称为主键索引，其索引树的叶子节点中存的是整行数据，表中行的物理顺序与键值的逻辑（索引）顺序相同。一个表只能包含一个聚集索引。因为索引（目录）只能按照一种方法进行排序。

非聚簇索引（普通索引）的叶子节点内容是主键的值。

在 InnoDB 里，非主键索引也被称为二级索引（secondary index）。

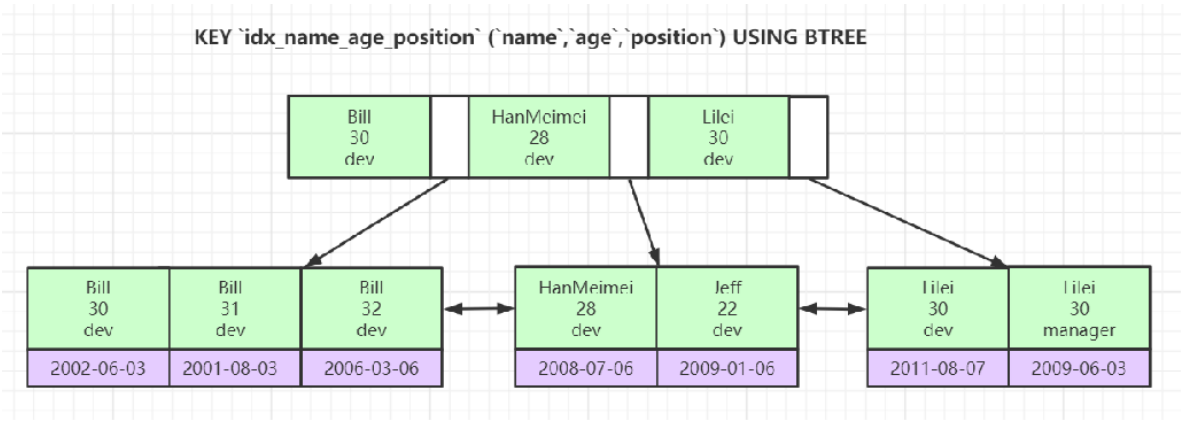
但从索引角度讲：聚集索引比非聚集索引要快，聚集索引不需要跨文件取数据，效率更高



4、索引最左前缀原理

开发中用的最多的是联合索引，通过建立几个联合索引，可以满足90%作用的查询需求，不建议建立太多的单值索引

联合索引的底层存储结构长什么样?以下是组合主键所构成的索引结构



底层数据结构还是B+树

排序时，逐个比较name age position字段；第一个字段name排好后，第二个age相对name是有序的

5、二级索引查找数据过程

先从二级索引中查出主键，在去主键所在索引查询数据（回表），好处：数据一致性、节省存储空间