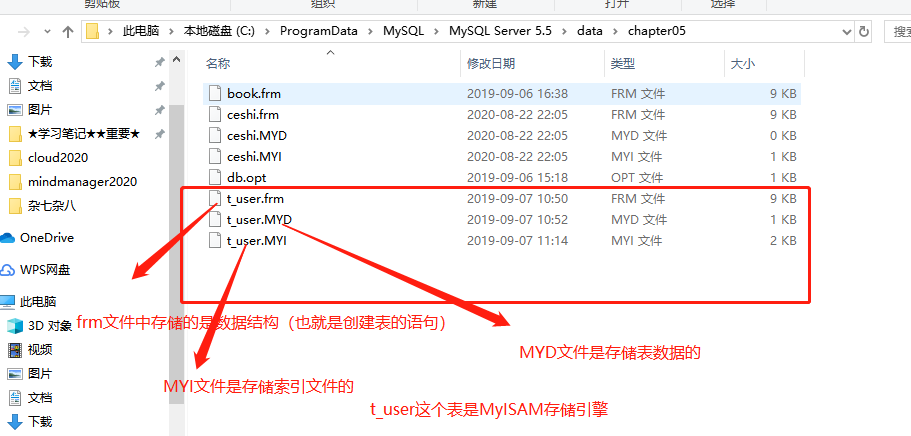
# 数据库学习笔记

## mySQL学习笔记

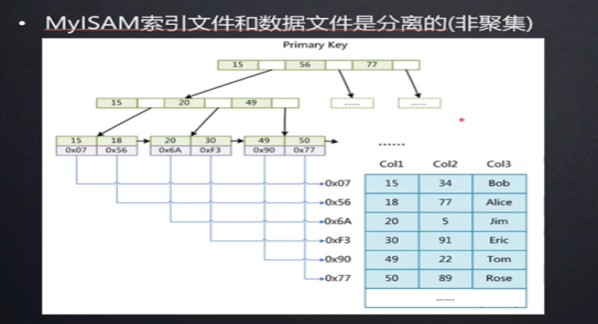
### MyISAM存储引擎索引实现

MyISAM索引文件和数据文件是分离的（非聚集）



### 什么叫做集聚什么叫做非聚集？

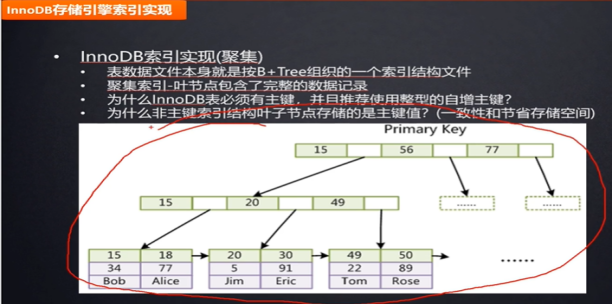
索引与数据分开存储叫为非聚集（MYISAM为非聚集），索引与数据一起存储为聚集（innoDB为聚集）



非聚集索引需要查找两次，一次过滤MYI文件，一次过滤MYD文件，而聚集索引只需要过滤一次IDB文件(因为他们数据和索引存储在一个IDB文件中)

### InnoDB存储引擎索引实现

InnoDB索引文件和数据文件是在一起的（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（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组件？

为什么飞主键索引结构叶子节点存储的是主键值？（一致性和节省存储空间）

### 为什么用自增列作为主键

如果我们定义了主键(PRIMARY KEY)，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，这个ROWID不像ORACLE的ROWID那样可引用，是隐含的)。

### MYSQL联合索引

1、联合索引是两个或更多个列上的索引。

对于联合索引:Mysql从左到右的使用索引中的字段，一个查询可以只使用索引中的一部份，但只能是最左侧部分。

例如索引是key index (a,b,c). 可以支持a 、 a,b 、 a,b,c 3种组合进行查找，但不支持 b,c进行查找 .当最左侧字段是常量引用时，索引就十分有效。

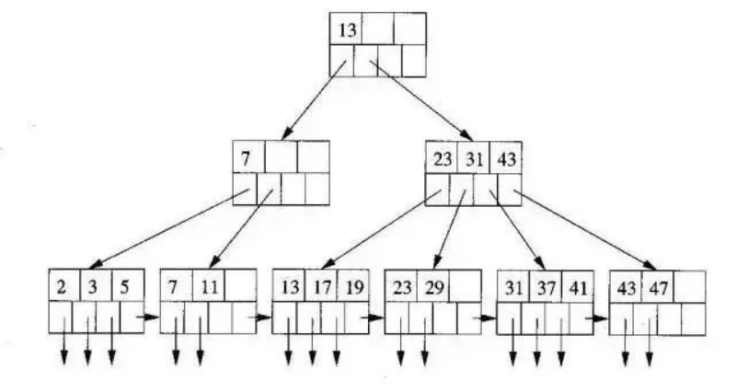
2、利用索引中的附加列，您可以缩小搜索的范围，但使用一个具有两列的索引不同于使用两个单独的索引。

复合索引的结构与电话簿类似，人名由姓和名构成，电话簿首先按姓氏对进行排序，然后按名字对有相同姓氏的人进行排序。

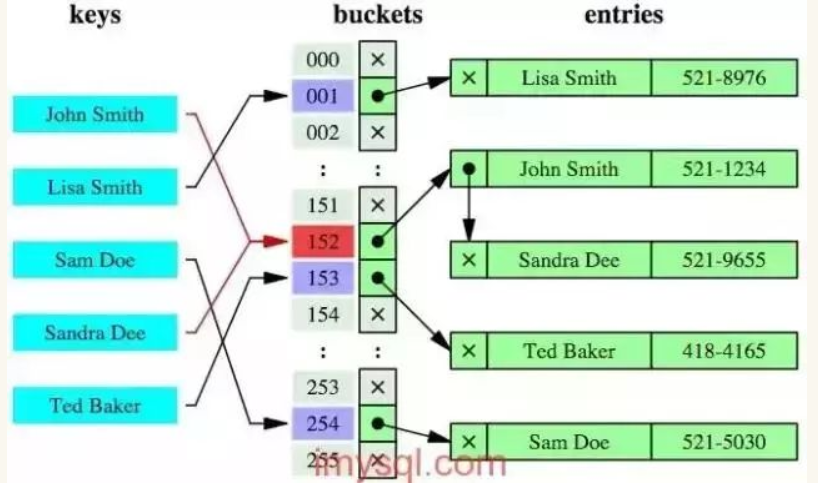
如果您知道姓，电话簿将非常有用；如果您知道姓和名，电话簿则更为有用，但如果您只知道名不知道姓，电话簿将没有用处。

### B+树索引和哈希索引的区别

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，是有序的，如下图：



哈希索引就是采用一定的哈希算法，把键值换算成新的哈希值，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，只需一次哈希算法即可,是无序的，如下图所示：



### 哈希索引的优势：

等值查询，哈希索引具有绝对优势（前提是：没有大量重复键值，如果大量重复键值时，哈希索引的效率很低，因为存在所谓的哈希碰撞问题。）

### 哈希索引不适用的场景：

不支持范围查询

不支持索引完成排序

不支持联合索引的最左前缀匹配规则

通常，B+树索引结构适用于绝大多数场景，像下面这种场景用哈希索引才更有优势：

在HEAP表中，如果存储的数据重复度很低（也就是说基数很大），对该列数据以等值查询为主，没有范围查询、没有排序的时候，特别适合采用哈希索引，例如这种SQL：

# 仅等值查询

select id, name from table where name='李明';

而常用的 InnoDB 引擎中默认使用的是B+树索引，它会实时监控表上索引的使用情况。

如果认为建立哈希索引可以提高查询效率，则自动在内存中的“自适应哈希索引缓冲区”建立哈希索引（在InnoDB中默认开启自适应哈希索引）。

通过观察搜索模式，MySQL会利用index key的前缀建立哈希索引，如果一个表几乎大部分都在缓冲池中，那么建立一个哈希索引能够加快等值查询。

注意：在某些工作负载下，通过哈希索引查找带来的性能提升远大于额外的监控索引搜索情况和保持这个哈希表结构所带来的开销。

但某些时候，在负载高的情况下，自适应哈希索引中添加的read/write锁也会带来竞争，比如高并发的join操作。like操作和%的通配符操作也不适用于自适应哈希索引，可能要关闭自适应哈希索引。

## 2.Oracle学习笔记