[聚簇索引和非聚簇索引(通俗易懂 言简意赅)](https://www.cnblogs.com/jiawen010/p/11805241.html)

总结：**InnoDB中，表数据文件本身就是按B+Tree组织的一个索引结构，聚簇索引就是按照每张表的主键构造一颗B+树，同时叶子节点中存放的就是整张表的行记录数据，也将聚集索引的叶子节点称为数据页。这个特性决定了索引组织表中数据也是索引的一部分；**

**一般建表会用一个自增主键做聚簇索引，没有的话MySQL会默认创建，但是这个主键如果更改代价较高，故建表时要考虑自增ID不能频繁update这点。**

**我们日常工作中，根据实际情况自行添加的索引都是辅助索引，辅助索引就是一个为了需找主键索引的二级索引，现在找到主键索引再通过主键索引找数据；**

本文链接：https://blog.csdn.net/lm1060891265/article/details/81482136  
参考博客：http://www.admin10000.com/document/5372.html

聚簇索引并不是一种单独的索引类型，而**是一种数据存储方式**。具体细节依赖于其实现方式。

MySQL数据库中innodb存储引擎，B+树索引可以分为聚簇索引（也称聚集索引，clustered index）和辅助索引（有时也称非聚簇索引或二级索引，secondary index，non-clustered index）。这两种索引内部都是B+树，聚集索引的叶子节点存放着一整行的数据。

Innobd中的主键索引是一种聚簇索引，非聚簇索引都是辅助索引，像复合索引、前缀索引、唯一索引。

Innodb使用的是聚簇索引，MyISam使用的是非聚簇索引

**聚簇索引(聚集索引)**

　　聚簇索引就是按照每张表的主键构造一颗B+树，同时叶子节点中存放的就是整张表的行记录数据，也将聚集索引的叶子节点称为数据页。这个特性决定了索引组织表中数据也是索引的一部分，每张表只能拥有一个聚簇索引。

　　Innodb通过主键聚集数据，如果没有定义主键，innodb会选择非空的唯一索引代替。如果没有这样的索引，innodb会隐式的定义一个主键来作为聚簇索引。

聚簇索引的优缺点

　　优点：

　　　　1.数据访问更快，因为聚簇索引将索引和数据保存在同一个B+树中，因此从聚簇索引中获取数据比非聚簇索引更快

　　　　2.聚簇索引对于主键的排序查找和范围查找速度非常快  
　　缺点：

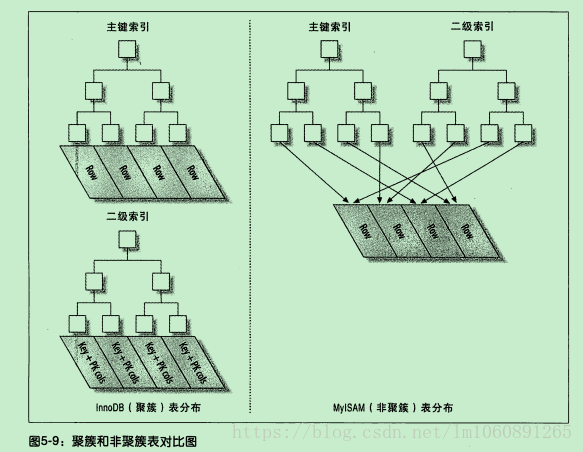
　　　　1.插入速度严重依赖于插入顺序，按照主键的**顺序插入**是最快的方式，否则将会出现页分裂，严重影响性能。因此，对于InnoDB表，我们一般都会定义一个**自增的ID列为主键**  
　　　　2.**更新主键的代价很高**，因为将会导致被更新的行移动。因此，对于InnoDB表，我们一般定义主键为不可更新。  
　　　　3.二级索引访问需要两次索引查找，第一次找到主键值，第二次根据主键值找到行数据。

**辅助索引（非聚簇索引）**

　　在**聚簇索引之上创建的索引称之为辅助索引**，辅助索引访问数据总是需要二次查找。辅助索引叶子节点存储的不再是行的物理位置，而是主键值。通过辅助索引首先找到的是主键值，再通过主键值找到数据行的数据页，再通过数据页中的Page Directory找到数据行。

　　Innodb辅助索引的叶子节点并**不包含行记录的全部数据**，叶子节点除了包含键值外，还包含了相应行数据的聚簇索引键。

　　辅助索引的存在不影响数据在聚簇索引中的组织，所以一张表可以有多个辅助索引。在innodb中有时也称辅助索引为二级索引。



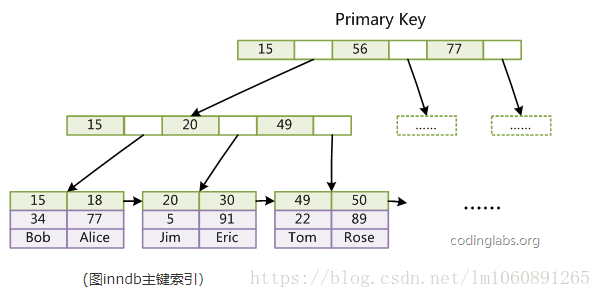
　　Innodb聚簇索引和MyIsam非聚簇索引的比较说明  
　　参考博客：https://www.cnblogs.com/zlcxbb/p/5757245.html

**InnoDB索引实现**

　　InnoDB**也使用B+Tree作为索引结构**，但具体实现方式却与MyISAM截然不同.

　　1）主键索引：

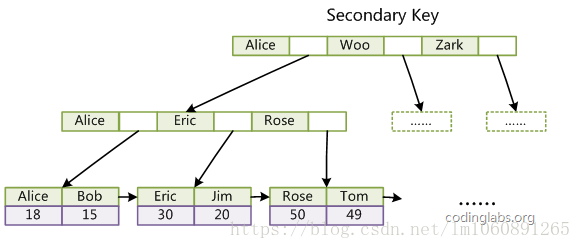
**MyISAM索引文件和数据文件是分离的**，**索引文件仅保存数据记录的地址**。而在**InnoDB中，表数据文件本身就是按B+Tree组织的一个索引结构**，这棵树的叶节点data域保存了完整的数据记录。这个索引的key是数据表的主键，因此InnoDB表数据文件本身就是主索引。



　　(图inndb主键索引）是InnoDB主索引（同时也是数据文件）的示意图，可以看到叶节点包含了完整的数据记录。这种索引叫做聚集索引。因为InnoDB的数据文件本身要按主键聚集，所以InnoDB要求表必须有主键（MyISAM可以没有），如果没有显式指定，则MySQL系统会自动选择一个可以唯一标识数据记录的列作为主键，如果不存在这种列，则**MySQL自动为InnoDB表生成一个隐含字段作为主键，这个字段长度为6个字节，类型为长整形**。

2）InnoDB的辅助索引

   InnoDB的所有辅助索引都引用主键作为data域。例如，下图为定义在Col3上的一个辅助索引：



　　InnoDB 表是基于聚簇索引建立的。因此InnoDB 的索引能提供一种非常快速的主键查找性能。不过，它的辅助索引（Secondary Index， 也就是非主键索引）也会包含主键列，所以，**如果主键定义的比较大，其他索引也将很大**。如果想在表上定义 、很多索引，则争取尽量把主键定义得小一些。InnoDB 不会压缩索引。

　　文字符的ASCII码作为比较准则。聚集索引这种实现方式使得按主键的搜索十分高效，但是辅助索引搜索需要检索两遍索引：首先检索辅助索引获得主键，然后用主键到主索引中检索获得记录。

　　不同存储引擎的索引实现方式对于正确使用和优化索引都非常有帮助，例如知道了InnoDB的索引实现后，就很容易明白：

　　　　1、为什么不建议使用过长的字段作为主键，因为所有辅助索引都引用主索引，过长的主索引会令辅助索引变得过大。再例如，

　　　　2、用非单调的字段作为主键在InnoDB中不是个好主意，因为InnoDB数据文件本身是一颗B+Tree，非单调的主键会造成在插入新记录时数据文件为了维持B+Tree的特性而频繁的分裂调整，十分低效，而使用自增字段作为主键则是一个很好的选择。

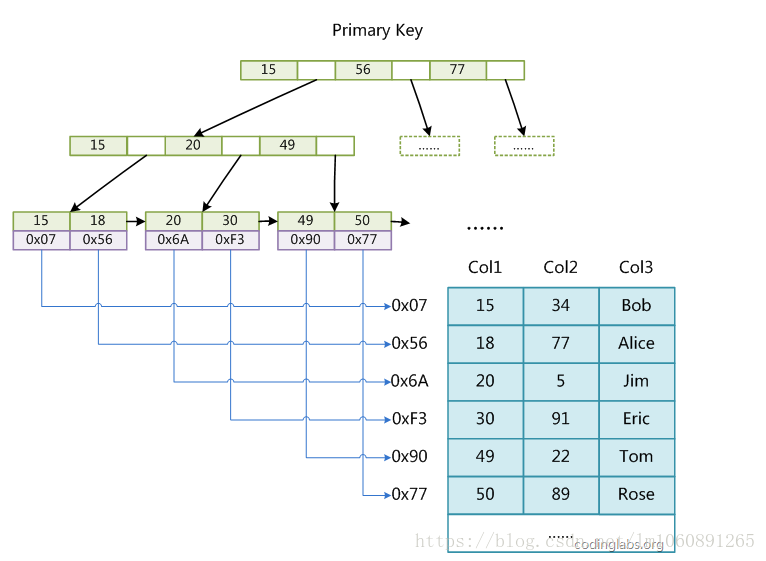
　　InnoDB使用的是聚簇索引，将主键组织到一棵B+树中，而行数据就储存在叶子节点上，若使用"where id = 14"这样的条件查找主键，则按照B+树的检索算法即可查找到对应的叶节点，之后获得行数据。若对Name列进行条件搜索，则需要两个步骤：第一步在辅助索引B+树中检索Name，到达其叶子节点获取对应的主键。第二步使用主键在主索引B+树种再执行一次B+树检索操作，最终到达叶子节点即可获取整行数据。

MyISAM索引实现

MyISAM索引文件和数据文件是分离的，索引文件仅保存数据记录的地址

1）主键索引：

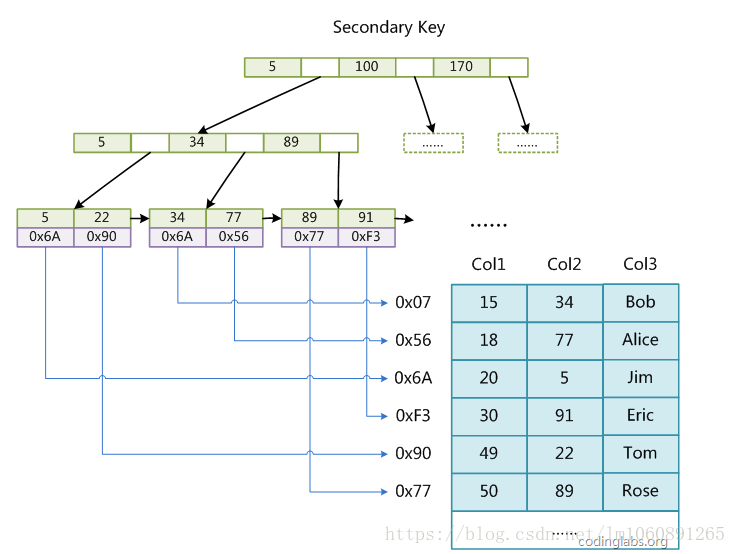
MyISAM引擎使用B+Tree作为索引结构，叶节点的**data域存放的是数据记录的地址**。下图是MyISAM主键索引的原理图：



这里设表一共有三列，假设我们以Col1为主键，图myisam1是一个MyISAM表的主索引（Primary key）示意。可以看出MyISAM的索引文件仅仅保存数据记录的地址。

2）辅助索引（Secondary key）

在MyISAM中，主索引和辅助索引（Secondary key）在结构上没有任何区别，**只是主索引要求key是唯一的，而辅助索引的key可以重复**。如果我们在Col2上建立一个辅助索引，则此索引的结构如下图所示：

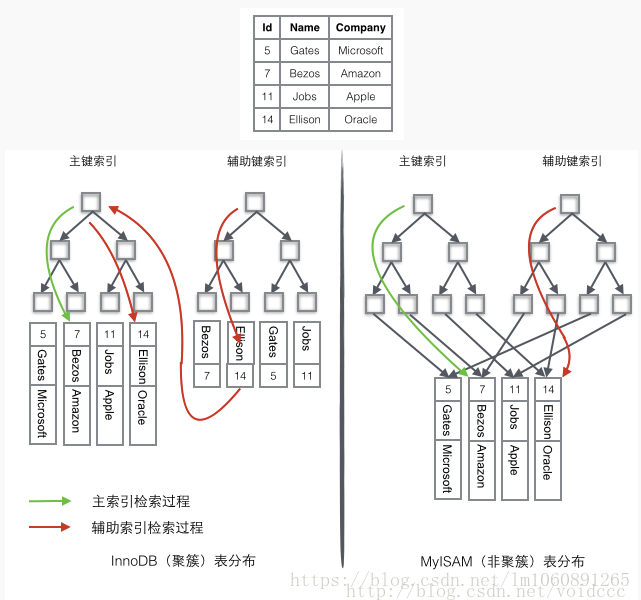


　　同样也是一颗B+Tree，data域保存数据记录的地址。因此，MyISAM中索引检索的算法为首先按照B+Tree搜索算法搜索索引，如果指定的Key存在，则取出其data域的值，然后以data域的值为地址，读取相应数据记录。

MyISAM的索引方式也叫做“非聚集”的，之所以这么称呼是为了与InnoDB的聚集索引区分。

**MyISM使用的是非聚簇索引**，非聚簇索引的两棵B+树看上去没什么不同，节点的结构完全一致只是存储的内容不同而已，主键索引B+树的节点存储了主键，辅助键索引B+树存储了辅助键。表数据存储在独立的地方，这两颗B+树的叶子节点都使用一个地址指向真正的表数据，对于表数据来说，这两个键没有任何差别。由于索引树是独立的，通过辅助键检索无需访问主键的索引树。

为了更形象说明这两种索引的区别，我们假想一个表如下图存储了4行数据。其中Id作为主索引，Name作为辅助索引。图示清晰的显示了聚簇索引和非聚簇索引的差异。



问题：主键索引是聚集索引还是非聚集索引？

在Innodb下主键索引是聚集索引，在Myisam下主键索引是非聚集索引

聚簇索引和非聚簇索引的区别

聚簇索引的叶子节点存放的是主键值和数据行，**支持覆盖索引**；二级索引的叶子节点存放的是主键值或指向数据行的指针。

由于节子节点(数据页)只能按照一颗B+树排序，故**一张表只能有一个聚簇索引**。辅助索引的存在不影响聚簇索引中数据的组织，所以一张表可以有多个辅助索引

注： 覆盖索引（covering index）指一个查询语句的执行只用从索引中就能够取得，不必从数据表中读取。也可以称之为实现了索引覆盖。 当一条查询语句符合覆盖索引条件时，MySQL只需要通过索引就可以返回查询所需要的数据，这样避免了查到索引后再返回表操作，减少I/O提高效率。 如，表covering\_index\_sample中有一个普通索引 idx\_key1\_key2(key1,key2)。当我们通过SQL语句：select key2 from covering\_index\_sample where key1 = ‘keytest’;的时候，就可以通过覆盖索引查询，无需回表。

参考文档：

<https://www.cnblogs.com/jiawen010/p/11805241.html>