**索引优化，可以说是[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/zhangliangzi/article/details/_blank)相关优化、理解尤其是查询优化中最常用的优化手段之一。所以，只有深入索引的实现原理、存储方式、不同索引间区别，才能设计或使用最优的索引，最大幅度的提升查询效率！**

Mysql目前主要有以下几种索引类型：FULLTEXT，HASH，BTREE，RTREE

**一、BTree索引**

**1、概述**

**[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "http://blog.csdn.net/zhangliangzi/article/details/_blank)**数据库中使用最频繁的索引类型，**基本所有存储引擎都支持BTree索引**。正是其优异的检索表现，才使其有这样的地位。

**2、存储结构**

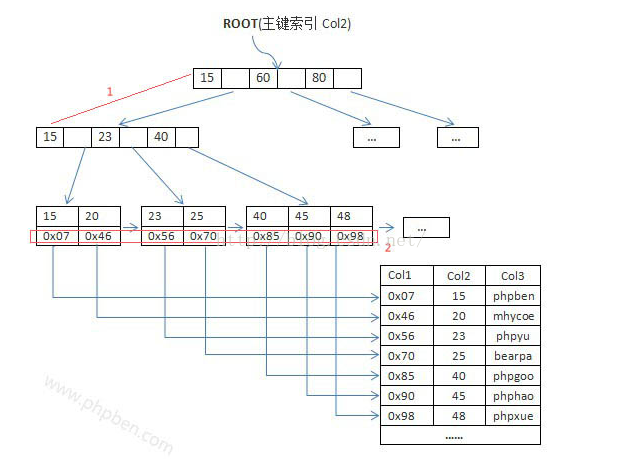
正如其名，这类索引的物理文件大多就是以BTree结构来存储的，但会有不同的存储引擎在使用BTree索引时，对存储结构稍作修改。比如MyISAM存储引擎，使用B+Tree的**[数据结构](http://lib.csdn.net/base/datastructure" \o "算法与数据结构知识库" \t "http://blog.csdn.net/zhangliangzi/article/details/_blank)**，它相对与BTree结构，所有的数据都存放在叶子节点上，且把叶子节点通过指针连接到一起，形成了一条数据链表，以加快相邻数据的检索效率。

另外，对于innoDB存储引擎，虽然同样使用B+Tree作为索引的存储结构，但具体实现却与MyISAM截然不同，这也是作为**MyISAM与InnoDB存储引擎的一个重要区别**反复被面试官问到。

MyISAM的**BTree索引：**

（1）MyISAM引擎索引结构的 叶子节点的数据域，存放的并不是实际的数据记录，而是数据记录的地址。索引文件与数据文件分离，这样的索引称为“**非聚簇索引**”。MyISAM的主索引与辅助索引区别并不大，只是主键索引不能有重复的关键字。

如下图所示为**非聚簇索引**的主键索引：

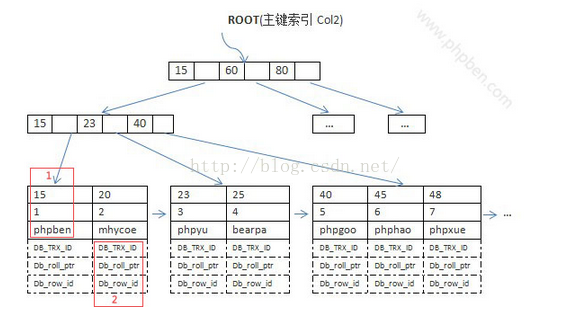


其检索算法：先按照B+Tree的检索算法检索，找到指定关键字，则取出对应数据域的值，作为地址取出数据记录。

InnoDB的**BTree索引：**

（2）InnoDB引擎索引结构的**叶子节点的数据域**，存放的就是**实际的数据记录**（对于主索引，此处会存放表中所有的数据记录；对于辅助索引此处会引用主键，检索的时候通过主键到主键索引中找到对应数据行），或者说，InnoDB的数据文件本身就是主键索引文件，这样的索引被称为“**聚簇索引**”，一个表只能有一个聚簇索引。

如下图所示为聚簇索引的主键索引：



**二、Hash索引**

**1、概述及存储结构**

主要就是通过Hash算法（常见的Hash算法有直接定址法、平方取中法、折叠法、除数取余法、随机数法），将数据库字段数据转换成定长的Hash值，与这条数据的行指针一并存入Hash表的对应位置；如果发生Hash碰撞（两个不同关键字的Hash值相同），则在对应Hash键下以链表形式存储。

检索算法：在检索查询时，就再次对待查关键字再次执行相同的Hash算法，得到Hash值，到对应Hash表对应位置取出数据即可，如果发生Hash碰撞，则需要在取值时进行筛选。目前使用Hash索引的数据库并不多，主要有Memory等。

**2、Hash索引的弊端**

一般来说，**Hash索引**的检索效率非常高，可以一次定位，不像B-Tree索引需要进行从根节点到叶节点的多次IO操作。有利必有弊，Hash算法在索引的应用也有很多弊端。

a、Hash索引仅仅能满足等值的查询，范围查询不保证结果正确。因为数据在经过Hash算法后，其大小关系就可能发生变化。

b、Hash索引不能被排序。同样是因为数据经过Hash算法后，大小关系就可能发生变化，排序是没有意义的。

c、Hash索引不能避免表数据的扫描。因为发生Hash碰撞时，仅仅比较Hash值是不够的，需要比较实际的值以判定是否符合要求。

d、Hash索引在发生大量Hash值相同的情况时性能不一定比B-Tree索引高。因为碰撞情况会导致多次的表数据的扫描，造成整体性能的低下，可以通过采用合适的Hash算法一定程度解决这个问题。

e、Hash索引不能使用部分索引键查询。因为当使用组合索引情况时，是把多个数据库列数据合并后再计算Hash值，所以对单独列数据计算Hash值是没有意义的。

**三、Full-Text索引**

**1、概述**

全文索引，目前**MySQL中只有MyISAM存储引擎支持**，并且只有CHAR、VARCHAR、TEXT类型支持。它用于替代效率较低的LIKE模糊匹配操作，而且可以通过多字段组合的全文索引一次性全模糊匹配多个字段。

**2、存储结构**

同样使用B-Tree存放索引数据，但使用的是特定的算法，将字段数据分割后再进行索引（一般每4个字节一次分割），索引文件存储的是分割前的索引字符串集合，与分割后的索引信息，对应Btree结构的节点存储的是分割后的词信息以及它在分割前的索引字符串集合中的位置。

**四、索引利弊**

**1、索引的好处**

a、提高数据检索的效率，降低检索过程中必须要读取得数据量，降低数据库IO成本。

b、降低数据库的排序成本。因为索引就是对字段数据进行排序后存储的，如果待排序的字段与索引键字段一致，就在取出数据后不用再次排序了，因为通过索引取得的数据已满足排序要求。另外，分组操作是先排序后分组，所以索引同样可以省略分组的排序操作，降低内存与CPU资源的消耗。

**2、索引的弊端**

a、索引会增加 增、删、改操作所带来的IO量与调整索引的计算量。

b、索引要占用空间，随着数据量的不断增大，索引还会带来存储空间的消耗。

**五、判断是否应该建索引的条件**

1、**较频繁的作为查询条件的字段**应该创建索引

2、唯一性太差的字段不适合单独创建索引，即使频繁作为查询条件

3、**增、删、改**操作较多的数据库字段不适合建索引