# Mysql索引及其实现原理

## 索引的本质

索引是帮助mysql高效获取数据的排好序的数据结构

## 索引的数据结构

### 2.1 二叉树：不是mysql索引的数据结构

|  |
| --- |
| 右子节点的值大于父节点的值，左子节点小于父节点的值  1580970579(1) |
| 如果给col1字段设置索引，使用二叉树作为数据结构二叉树将会特别长(类似于数组结构，查询效率低)，如果需要查找col1=6的值，需要查找6次  1580970444(1) |

### 2.2 红黑树：不是mysql索引的数据结构

|  |
| --- |
| 红黑树实质上也是一棵二叉树(平衡二叉树)：他会自动旋转平衡  红黑树的层数太多，树的高度将会特别大，所以mysql索引的数据结构也不会采用红黑树  1580971049(1) |

### 2.3 Hash：mysql索引数据结构，默认是BTREE

Hash会对值进行散列，不支持范围查找

一般使用默认的BTREE数据结构。

Select \* from table where col = 6

如果是这种的话，hash数据结构查询效率比BTREE数据结构快，因为可以使用hash(6)来计算这列数据的散列值

Select \* from table where col > 6 and col < 16

如果是这种的话，hash作为索引的数据结构将不能生效，因为没法对查询条件进行散列计算，而B+Tree可以很好的支持范围查找

### 2.4 B-tree：不是mysql索引的数据结构

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

|  |
| --- |
| 1580972095(1) |

### 2.5 B+Tree(B-Tree的变种)：mysql索引的数据结构

非叶子节点不存储数据，只存储索引(冗余)，可以放更多的索引

叶子节点包含所有索引字段

叶子节点用指针连接，提高区间访问的可能性，很好的支持范围查找(因为叶子节点的末尾和开头会存储一个地址值，双向)

|  |
| --- |
| 1580972251(1) |

## Mysql存储引擎

Mysql存储引擎是形容数据表的，而非形容数据库的

### myIsam(非聚集索引也叫稀疏索引)

|  |
| --- |
| 1580974359(1)  eg：select \* from table where col1 = 49 的查找流程   1. 先查找col1这一列是不是索引，如果是再找49在索引中的位置 2. 找到49对应数据的磁盘文件地址为0\*90 3. 再根据磁盘文件地址去myd数据文件里面查找0\*90的这条数据 |

#### myisam的存储结构

每个表在磁盘上被存储成3个文件：(mysqlserver/data/数据库名称):

每一个文件的名字以表名开头，扩展名指出文件类型。

tableName.frm文件存储表结构

tableName.myi (MYIndex)文件存储索引

tableName.myd (MYData)文件存储数据

#### Myisam的存储空间

myisam可以被压缩，存储空间较小，支持3中不同的存储格式

（1）静态表：默认存储格式，但是注意数据末尾不能有空格，会被去掉

（2）动态表

（3）压缩表

#### 其它

1. 不支持事务。强调的是性能，每次查询具有原子性，执行速度比innodb更快
2. 不支持外键
3. 如果执行大量的select操作（读多写少），myisam是最好的选择(因为没有支持行级锁，只支持表级索)。在增删改时需要锁定整个表格，效率低

### Innodb(聚集索引：叶节点包含了完整的数据记录)

|  |
| --- |
| eg：select \* from table where col1 = 49 的查找流程  （1）先查找col1这一列是不是索引，如果是再找49在索引中的位置  （2）找到49对应的数据  1580975799(1) |

#### innodb的存储结构

每个表在磁盘上被存储成两个文件

每一个文件的名字以表名开头，扩展名指出文件类型。

tableName.frm：存储表结构

tableName.ibd：存储索引和数据

#### innodb的存储空间

Innodb需要更多的内存和存储空间，它会在主内存中建立其专用的缓存池用于高速缓存数据和索引

#### 其它

1. 支持事务：commit（提交）、rollback（回滚）、crash recovery capabilities（崩溃修复能力）的事务安全型表
2. 支持外键
3. 如果执行大量的增删改（读少写多），出于性能方面的考虑，应该选择innodb（支持行级锁，增删改的时候只需要锁定改变的行即可）。但delete from table时，Innodb不会重新建立表，而是一行一行的删除，在innodb上如果要清空保存有大量数据的表，最好使用truncate table 这个命令

#### innodb面试题

##### 为什么innodb表必须有主键，并且推荐使用整型的自增主键

（1）为什么innodb表必须有主键

表数据文件本身就是按B+Tree组织的一个索引结构文件，需要使用主键来维护B+Tree的索引数据结构

如果设计表的时候没有设计主键，mysql底层会选择一列没有重复的数据作为主键，如果不存在这样的数据(即不存在任何一列没有重复数据的列)，mysql会在最后保存一行rowId来维护主键

1. 为什么推荐使用整型主键

因为根据主键去查找索引位置的时候，根据数字查找比较快，select \* from table from col = 49，并且使用整型节约内存，占用空间比较小

而如果主键使用uuid之类的话，根据主键去查找索引的时候字符串不如数字查找快并且字符串比整型占用空间大

1. 为什么推荐使用自增主键

效率高

## 索引

### 组合索引

#### 组合索引(结构)最左前缀法则

|  |
| --- |
| 1581046683(1) |