## Tree

|  |
| --- |
|  |
|  |

## B+Tree(B-Tree的变种)

**MySQL底层数据结构是B+Tree**

|  |
| --- |
|  |

## HASH表(短板是不支持范围查找和排序)

MySQL的索引也可以设置成hash表

如果都是等值查找且不用排序，HASH表的效率是最高的

原因：hash表会根据hash算法算出来一个hash值，然后直接用算出来的值到hash表中找到这个数据，找到并取到这条数据，也就是说，和表的数据量没有关系，只需要一次IO交互，并且由于MySQL对hash碰撞进行了处理，所以发生hash碰撞的概率也非常小。

## 存储引擎

### myisam

|  |
| --- |
| 在数据库的data文件夹中，会有三个和表同名不同格式的文件夹，其中，  .frm文件存储结构，定义文件  .MYD文件存储表中所有的数据行  .MYI文件存储的是表的索引  例如： |

### InnoDB

|  |
| --- |
|  |
|  |
| .frm：表定义文件  .ibd文件：存储的是索引+数据 |
|  |
|  |

### 面试题

|  |
| --- |
| ***为啥InnoDB为啥必须有主键？***  表数据本身就是用B+树存储，如果没有主键索引，无法阻止data中的元素。  自己建表没有建主键也能成功，是因为mysql在表中找到一行唯一数据作为了索引，如果没有找到的话，会在表最后建一列Rowid作为主键索引。  也就是说，设计如此。  ***为啥推荐使用整型的自增主键？***  **整型：**因为B+树查找的过程中会涉及到大量的比较大小的操作，还有，字符串在比较大小的的时候会先转化成ASCII码再比较，也就是说，数字的比较效率会远远大于字符串的比较的效率。  **自增：**因为如果新加一行元素的话，自增的时候只需要在最后加一条就可以了，B+树很好处理，但是如果非自增或者是字符串，可能会打乱之前的排序，尤其当改索引的节点放满之后，很可能会完全打乱之前的排序，不利于运维而且效率比较低。 |
| ***聚集索引、聚簇索引：***  索引和数据列都放到一起存储，比如InnoDB。  ***非聚集索引(稀疏索引)：***  向MyISAM索引的方式，索引和数据是分开存储的。 |

### 非主键索引(除主键之外的)：

|  |
| --- |
| 在叶子节点，存储的是数据的主键。  过程：先根据非主键索引找到数据的主键，然后根据主键再查找该数据，也就是需要遍历两棵树，所以效率相对于主键索引稍微差一点。  为啥只存主键？  保证数据的一致性以及节省存储空间  如果 |

### 联合索引：

|  |
| --- |
| 先比较第一个索引字段，然后比较第二个，然后第三个......根据这个排序。 |
|  |

## Explain工具

|  |
| --- |
|  |