索引

# . 索引（index）是什么

## MySQL官方对索引的定义

索引是帮助MySQL高效获取数据的数据结构。

可以得到索引的本质，**一种数据结构。**

描述：

索引的目的在于提高查询效率，可以类比字典

如果要查mysql这个单词，我们肯定需要定位到m字母，然后往下找到y字母，在找剩下的sql。

如果没有索引，那么你可能需要a---z，如果我想找到java开头的单词呢？

## 我们的定义

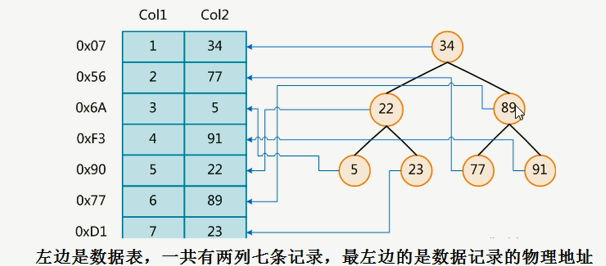
**排好序的快速查找结构。★**

所以，索引有两个作用，排好序，和查找快，分别对应两个部分的字段，**where后面的查找，和order by后面的排序**

### 详解：

**在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种凡是引用（指向）数据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引（B树索引）**

下图为一个索引案例：



为了加快Col2的查找，可以维护一个右边所示的二叉查找树，每个节点分别包含索引键值和一个指向对应数据记录物理地址的指针，，这样就可以运用二叉查找在一定的复杂度内获取相应数据，从而加快的检索出符合条件的记录。

### 索引的问题

1. 大量的删除数据，会使索引指向的内容为空，还会打乱索引的结构，通常把数据置为不可用即可，避免删除数据。
2. 大量的修改数据，索引也需要跟着修改，修改数据的速度会下降。

总结：索引适合建在查找频繁的表中，不适合建在删改频繁的表中。

## 常见索引类型

我们平时所说的索引，如果没有特别指明，都是指B树 BTREE（多路搜索树，并不一定是二叉树）结构组织的索引。其中聚集索引，次要索引，覆盖索引，复合索引，前缀索引，唯一索引默认都是使用B+树索引，统称索引。当然，除了B+树这种类型的索引之外，还有哈希索引（hash index）等。

## 文件形式

一般来说索引本身也很大，不可能全部存储在内存中，因此索引往往以索引文件的形式储存到磁盘上。

# .索引的优劣势

## 优势

类似大学图书馆书目索引，提高数据检索的效率，降低数据库的IO成本

通过索引列对数据进行排序，降低数据排序的成本，降低CPU的消耗

## 劣势

1. 实际上索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录，所以索引列也是要占用空间的。**一般一张表最多建5个索引。**
2. 虽然索引大大提高了查询速度，同时却会**降低更新表的速度**，如对表进行insert,update,delete都会调整因为更新所带来的键值变化后的索引信息。
3. 索引只是提高效率的一个因素，如果你的mysql有大量数据的表，就需要花时间研究和建立最优秀的索引，或优化查询语句。**索引不是一天建成的。**

# .索引的分类

## 单值索引

即一个索引只包含单个列，一个表可以有多个单列索引。

## 唯一索引

索引列的值必须唯一，但允许有空值。

## 复合索引

# .基本语法

[UNIQUE]表示是唯一索引

## 创建索引

CREATE [UNIQUE] INDEX 索引名 ON mytable(列名(长度));

ALTER 表名 ADD [UNIQUE] INDEX [索引名] ON (列名(长度))

## 删除

DROP INDEX [索引名] ON 表明

## 查看

SHOW INDEX FROM 表明

## 四种方式添加索引

ALTER TABLE 表名 PRIMARY KEY(列名)：该语句添加一个主键，这意味着索引值必须是唯一的，且不能为NULL。

ALTER TABLE 表名 ADD UNIQUE 索引名(列名)：该语句创建索引的值必须是唯一的（除了NULL外，NULL可能会出现多次）。

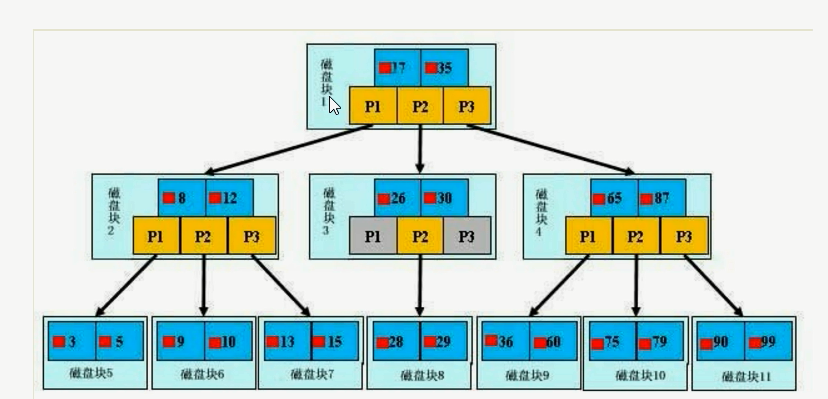
ALTER TABLE 表名 ADD INDEX 索引名(列名)：添加普通索引，索引值可出现多次。

ALTER TABLE 表名 ADD FULLTEXT 索引名(列名)：该语句指定了索引为FULLTEXT，用于全文索引。

# .索引分类和检索原理

## BTree索引

Java工程师必备，其他为数据库管理师需要学的。



### 初始化介绍

一颗B+树，浅蓝色的块为磁盘块，可以看到每个磁盘块包含几个数据项（深蓝所示）和指针（黄色所示），如磁盘块1包含数据项17和35，包含指针P1、P2、P3。

P1表示小于17的磁盘块，P2表示在17和35之间的磁盘块，P3表示大于35的磁盘块。

**真实的数据存在于叶子节点**即3、5、9、10、13、15、 28、 29、 36、 60、 75、 79、 90、 99。

**非叶子节点不存储真实数据，只储存指引搜索方向的数据项，**如17、 35并不真实存在于数据表中。

### 查找过程

查找数据项29，

1. 首先会把磁盘块1有磁盘加载到内存，此时**发生一次IO**，在内存中用二分查找确定29在17和35之间，锁定磁盘块1的P2指针，内存时间因为非常短（相比磁盘的IO）可以忽略不计。
2. 通过磁盘块1的P2指针的磁盘地址把磁盘块3有磁盘加载到内存，**发生第二次IO**，29在26和30之间，锁定磁盘块3的P2指针。
3. 通过指针加载磁盘块8到内存，**发生第三次IO**，同时内存中做二分查找找到29。
4. 结束查询，总计三个IO。

### 总结

在真实的生产中，3层的B+树可以表示上百万的数据，如果上百万的数据查找只需要3此IO，性能提高将是巨大的，如果没有索引，每个数据项都要发生一次IO，那么总共需要上百万次IO，显然成本非常非常高。

## Hash索引

了解

## Full-text全文索引

了解

## R-Tree索引

了解

# .创建索引的时机

## 哪些情况可以建索引

1. 主键自动创建唯一索引
2. 频繁作为查询条件的字段应该创建索引
3. 查询中与其它表关联的字段，外键关系建立索引
4. 单键/组合索引的选择问题，在高并发下倾向创建组合索引
5. 查询中排序的字段，排序字段若通过索引去访问将大大提高排序速度
6. 查询中统计或者分组字段

## 哪些情况不能建索引

1. 表记录太少
2. 经常增删改的字段或表

因为索引提高了查询速度，同时却会降低更新表的速度，如对表进行insert、update和delete。更新表时，mysql不仅要保存数据，还要保存一下索引文件。

1. Where条件里用不到的字段不创建索引
2. 数据重复且分布平均的表字段

如果某个数据列包括许多重复的内容，为它建立索引没有太大实际效果。

