链滴

mysql45 讲笔记 -- 深入浅出索引（上）

作者：dddygin

原文链接：https://ld246.com/article/1578995932370

来源网站：链滴

深入浅出索引（上）

什么是索引？，索引的出现其实就是为了提高数据查询效率，就像书的目录一样。 一本 500 页的书如果你想快速找到其中的某一个知识点，在不借助目录的情况下，那我估计你可得找一会儿。同样，于数据库的表而言，索引其实就是它的“目录” 。

索引的常见模型

- **哈希表（key-vlalue[value_0->..->vlues_n]）,哈希表这种结构适合只有等值查询的场景。**
- **有序数组（value_0->..->vlues_n）,有序数组在等值和范围查询场景中性能就都非常优秀，但是合静态存储引擎（因为修改耗性能高）。**
- **树，每个节点的左儿子小于父节点，父节点小于右儿子节点，即适合等值查询也适合范围查询。**

树结构中的“二叉树”搜索效率是最高的，但是大多数的数据库存储是采用的是多(N)叉树，原因是引不止在内存中，而且还要写到磁盘中。

假设有一颗100万节点的平衡二叉树，树高20，查询一次可能需要访问20个数据快，假设一个数据需10ms左右寻址,也就是说对于一个100万行的表，如果用二叉树来存储，单独访问一行，可能需要20*0ms=0.2s，速度慢。

其中InnoDB一个整数字段的索引为例，这个N差不多为1200，这样树高4的时候，数据就已经可以存17亿了。

InnoDB的索引模型

在InnoDB中，表都是根据主键顺序以索引的形式存放的，这种存放的方式称之为索引组织表，而且据都是存储在B+树中的。其中**每一个索引在InnoDB里面对应一颗B+树**

下面我们以一个例子来分析一下InnoDB的索引模型

建表语句：

```
mysql> create table T(
      id int primary key,
      k int not null,
      name varchar(16),
      index(k)
    )engine=InnoDB;
```

索引模型：

表中R1~R5的（ID，k）值分别为（100，1）、（200，2），（300，3）、（500，5）、（600，）如下表

从上图中我们可以索引类型分为主键索引和非主键索引。

其中主键索引的叶子节点村的是整行数据。在InnoDB里，主键索引也称之为聚簇索引（clustered index）

而非主键索引的叶子节点内容是主键的值。在InnoDB里，非主键索引也称之为二级索引（secondary ndex）

主键索引和非主键索引的区别

我们以例子来说明它们之间的区别

- 如果查询语句是　select * from T where ID=500,即主键索引，只需要搜索ID这颗B+树；
- 如果查询语句是　select * from T where k=5，即普通索引查询方式，则需要先搜索k索引树，得到D的值为500，在到ID索引树搜索一次，得到数据，这个过程称之为**回表**。

也就是说，**基于非主键索引的查询需要多扫描一颗索引树**。因此我们在应用中应该尽量使用主键查询。

索引维护

B+树为了维护索引的有序性，在插入新值的时候需要做必要的维护。上图为例，如果插入的新行ID为00，则只需要在R5的记录后面插入一个新记录。如果插入的ID是400，这就相对比较麻烦了，需要逻上移动后面的数据，空出位置。

而更加糟糕的是如果所在的数据页已经满了，根据B+树的算法，这个时候需要申请一个新的数据页然后挪动部分数据过去，这个过程称之为**页分列**。在这种情况下，性能自然回受影响。

除了性能外，页分列操作还影响数据页利用率，原本放在一个页的数据，现在分到两个页中，整体上用率降低了50%。

有页分裂就有合并，当相邻的两个页由于数据的删除，利用率很低，会将页合并，这个过程是页分列逆过程。

什么时候需要用自增索引

自增主键是指自增列上定义的主键，在建表语句中一般是这么定义的：NOT NULL PRIMARY KEY AUO_INCREMENT。

也就是说，自增主键的插入数据模式，正符合了我们前面提到的递增插入的场景。每次插入一条新记，都是追加操作，都不涉及到其他挪动其他记录，也不会触发叶子节点的分裂。

有业务逻辑的字段做主键，则往往不容易保证有序插入，这样写数据成本数据相对较高。

除了考虑性能之外，我们还可以从存储空间的角度看。假设你的表中的确有一个唯一字段，比如字符类型的身份证号，那应该用身份证做主键，还是用自增字段做主键呢？

由于每个非主键索引需要在叶子节点存储主键的值，如果用身份证做主键，那么每一个二级索引的叶节点占用20个字节，如果用整形做主键，则只要4个字节，如果是长整形（bingin）则需要8个字节。

显然主键的长度越小越好，这样普通索引占用的空间就越小。所以从存储空间的角度来看，自增主键往往比较合理的选择。

那种情况适合业务字段字节直接做主键？

1. 只有一个索引
2. 该索引必须是唯一索引。

自问自答

1. 什么是B+树？