第四章 Schema与数据类型优化

#### 一．选择优化的数据类型

* 更小的通常更好

占用更少的磁盘、内存和CPU缓存、处理时需要更少 的cpu周期。

* 简单就好

简单类型需要更少的CPU周期。

* 尽量避免null

1. 整数类型

TINYINT：8位

SMALLINT：16位

MEDIUMINT：24位

INT：32位

BIGINT：64位

存储空间是 -2（N-1）~2（ N-1）-1 其中N是存储空间的位数。

另外，整数类型有可选的UNSIGNED属性，表示不允许负值，这可以使正数的上限提高一倍，例如TINYINT从-128`127变成了0~255。

对于存储和计算来说，设置INT(1)和INT(20)是相同的，对存储和计算这样是没有意义的。

1. 实数类型

浮点类型：

浮点计算（float，double），会有精度损失

float：4子节

double：8子节

DECIMAL：

精确计算

1. 字符串类型

* VARCHAR：

用于存储可变长字符串（仅使用必要的空间）

会有额外子节（1个或2个，取决于列的长度是否大于255），来记录字符串长度。

优点：节省存储空间

缺点：行是变长的，所以update会需要额外工作，影响性能。

页面内没有更多存储空间时：

MyISAM会将行拆分成不同片段存储。

InnoDB则需要分裂页来存储。

适用场景：

字符串过长，或者列的更新很少，InnoDB会把过长的VARCHAR存储为BLOB。

* CHAR：

用于存储定长字符串（总是分配指定空间）

在检索的时候，会去掉字符串末尾的空格。

优点：效率更高

缺点：空间利用率限制

问题：VARCHAR(100)与VARCHAR(200)？

对于VARCHAR数据类型来说，硬盘上的存储空间虽然都是根据实际字符长度来分配存储空间的，但是对于内存来说，则不是。其时使用固定大小的内存块来保存值。简单的说，就是使用字符类型中定义的长度，即200个字符空间。显然，这对于排序或者临时表(这些内容都需要通过内存来实现)作业会产生比较大的不利影响。

1. BLOB和TEXT类型

BLOB：二进制类型

无排序规则或字符集。

TEXT：字符类型

有排序规则和字符集。

对这两个列进行排序：只对每个列的最前max\_sort\_length字节而不是整个字符串做排序。只需要排序前面一小部分字符时，可以使用order by substring(column,length)或者减小max\_sort\_length配置。

1. 枚举类型ENUM
2. 日期类型

TIMESTAMP ：

4字节，保存的是从1970年一月一日午夜至今的毫秒数。

DATETIME：

8字节，可以保存大范围的值，从1001到9999，精度为秒。

#### 一．范式化和反范式化

1. 范式的优缺点

优点：

* 更新操作更快
* 数据重复少
* 范式化的表更小
* 很少有多余的数据，很少用到group by

缺点：

需要关联，复杂的查询语句需要过多关联

2. 反范式的优缺点

优点：

可以避免关联

某些情况下，反范式的索引查询比范式更加高效。

缺点：

会有数据的重复