# 如何给MySQL的字符串字段加好索引？

原创 T Ti 笔记 2025年03月11日 16:57 广东

**点击蓝字 关注我们**

在实际的数据库优化中，字符串字段的索引设计往往是开发者和DBA的“痛点”。过长的索引会浪费存储空间、降低写入性能；而索引设计不当又会导致查询效率低下。本文将结合**前缀索引**、**索引区分度**等，谈谈如何才能给字符串字段加好索引。

## 一、为什么字符串字段需要特殊对待？

假设我们有一个用户表，其中包含邮箱字段（`email VARCHAR(255)`）。当我们需要按邮箱查询用户时，通常会给该字段加索引：

```
ALTER TABLE users ADD INDEX idx_email(email);
```

但此时会遇到两个问题：

1. **存储空间浪费**：每个邮箱可能长达255字符，但实际有效部分（如 @ 符号前的内容）可能只有20-30字符。
2. **查询性能瓶颈**：索引长度越长，B+树的层级可能越深，范围查询时IO次数增加。

## 二、全字段索引 vs 前缀索引

### 1. 全字段索引的困境

为整个字符串字段建立索引虽然能保证100%的区分度，但存在明显缺陷：

- **索引文件膨胀**：假设有100万条记录，每个索引条目多占用50字节，整体将增加约50MB空间。
- **写操作成本高**：每次INSERT/UPDATE都需要维护更大的索引树。

### 2. 前缀索引的救赎

前缀索引仅对字段的前N个字符建立索引：

```
ALTER TABLE users ADD INDEX idx_email_prefix(email(10));
```

但关键问题是：**如何选择合适的前缀长度？**

## 三、黄金法则：索引区分度的计算

**区分度（Selectivity）** 是衡量索引有效性的核心指标，计算公式为：

```
区分度 = COUNT(DISTINCT column) / COUNT(*)
```

**操作步骤：**

1. 统计不同前缀长度的区分度：

```
SELECT
  COUNT(DISTINCT LEFT(email, 5)) / COUNT(*) AS selectivity5,
  COUNT(DISTINCT LEFT(email, 6)) / COUNT(*) AS selectivity6,
  COUNT(DISTINCT LEFT(email, 7)) / COUNT(*) AS selectivity7
FROM users;
```

2. 找到区分度增长趋于平缓的拐点（通常达到90%以上即可）。

**案例：**
假设测试结果如下：

- 前缀5位：区分度82%
- 前缀6位：区分度95%
- 前缀7位：区分度96%

此时选择6位前缀能在空间和性能间取得最佳平衡。

## 四、前缀索引的副作用与应对策略

### 1. 覆盖索引失效

使用前缀索引后，以下查询无法利用覆盖索引，必须回表：

```
SELECT id, email FROM users WHERE email = 'xxx@domain.com';
```

**解决方案：**

- 若必须使用覆盖索引，只能采用全字段索引。
- 权衡回表成本与索引空间开销。

### 2. 前缀冲突风险

当数据分布不均匀时（如大量相同前缀的数据），可能引发额外扫描。

**解决方法：**

- 结合业务特征调整前缀长度。

## 五、其他优化方案

### 1. 倒序存储 + 前缀索引

适用于尾部区分度高的场景（如手机号后四位）：

```
ALTER TABLE users
ADD reversed_phone VARCHAR(15) AS (REVERSE(phone)),
ADD INDEX idx_reversed_phone(reversed_phone(4));
```

### 2. Hash字段法

增加一个哈希值的整型字段：

```
ALTER TABLE users
ADD email_crc INT UNSIGNED AS (CRC32(email)),
ADD INDEX idx_email_crc(email_crc);

-- 查询时需校验原字段
SELECT * FROM users
WHERE email_crc = CRC32('xxx@domain.com')
  AND email = 'xxx@domain.com';
```

### 3. 字段类型优化

- 将部分字符串字段转换为ENUM类型（如性别、状态）。
- 使用更紧凑的字符集（如将utf8mb4改为latin1，需确保兼容性）。

## 六、总结

1. **优先评估区分度**：使用 COUNT(DISTINCT LEFT(col, N)) 找到性价比最高的前缀长度。
2. **警惕字符集陷阱**：utf8mb4下每个字符占4字节，需在计算索引长度时注意。
3. **监控索引使用率**：定期执行 SHOW INDEX FROM table 观察 Cardinality 值。
4. **权衡写入与查询**：高频写入场景需更严格控制索引长度。
5. **联合索引优化**：将字符串字段与其他字段组成联合索引时，优先将区分度高的列放在前面。

## 往期推荐

▲ 数据库优化器是什么？


▲ 享元模式：高效复用对象的内存优化利器

▲ MySQL索引创建，用普通索引还是唯一索引？

END

别忘了

点赞、分享、爱心

↓↓↓

MySQL 17 数据库 17

MySQL · 目录

上一篇
数据库优化器是什么？

下一篇
MySQL表数据已经删了，为什么空间还是没释放？