### 简介

在MySQL中，索引（index）也叫做“键（key）”，它是存储引擎用于快速找到记录的一种数据结构。索引对于良好的性能非常关键，尤其是当表中的数据量越来越大时，索引对性能的影响就愈发重要。索引优化应该是对查询性能优化最有效的手段，创建一个真正最优的索引经常需要重写SQL查询语句。

### **类型**

**常规索引**，也叫普通索引（index或key），它可以常规地提高查询效率。一张数据表中可以有多个常规索引。常规索引是使用最普遍的索引类型，如果没有明确指明索引的类型，我们所说的索引都是指常规索引。

**主键索引**（Primary Key），也简称主键。它可以提高查询效率，并提供唯一性约束。一张表中只能有一个主键。被标志为自动增长的字段一定是主键，但主键不一定是自动增长。一般把主键定义在无意义的字段上（如：编号），主键的数据类型最好是数值。

**唯一索引**（Unique Key），可以提高查询效率，并提供唯一性约束。一张表中可以有多个唯一索引。

**全文索引**（Full Text），可以提高全文搜索的查询效率，一般使用Sphinx替代。但Sphinx不支持中文检索，Coreseek是支持中文的全文检索引擎，也称作具有中文分词功能的Sphinx。实际项目中，我们用到的是Coreseek。(仅MyISAM支持)

**创建**：

· CREATE TABLE article (

·     id INT UNSIGNED AUTO\_INCREMENT NOT NULL PRIMARY KEY,

·     title VARCHAR(200),

·     content TEXT,

·     FULLTEXT (title, content) --在title和content列上创建全文索引

· );

**使用**：SELECT \* FROM article WHERE MATCH(title, content) AGAINST('查询字符串')

**外键索引**（Foreign Key），简称外键，它可以提高查询效率，外键会自动和对应的其他表的主键关联。外键的主要作用是保证记录的一致性和完整性（仅InnoDB支持，但不推荐使用）。

外键字段如果没有指定索引名称，会自动生成。

如果要删除父表（如分类表）中的记录，必须先删除子表（带外键的表，如文章表）中的相应记录，否则会出错。

创建表的时候，可以给字段设置外键，如 foreign key(cate\_id) references cms\_cate(id。

### 索引方法

**B-Tree 索引**

不同的存储引擎以不同的方式使用B-Tree索引，性能也各不相同。

MyISAM：前缀压缩技术使索引更小，通过数据的物理位置引用被索引的行。

InnoDB：按照原始的数据格式存储索引，根据主键引用被索引的行。

1. Tree 对索引列是顺序存储的，因此很适合查找范围数据。它能够加快访问数据的速度，因为存储引擎不再需要进行全表扫描来获取需要的数据。 如果一个索引中包括多个字段（列）的值，那它就是一个复合索引。复合索引对多个字段值进行排序的依据是创建索引时列的顺序。

B-Tree 索引适用于全键值、键值范围或键前缀查找，其中键前缀查找只适用于根据最左前缀查找。

**全值匹配**

全值匹配指的是和索引中的所有列进行匹配。例如：查找姓Allen、名Cuba、出生日期为1960-01-01的人。

SQL语句为：

|  |  |
| --- | --- |
|  | select id,last\_name,first\_name,birthday from people where last\_name='Allen' and first\_name='Cuba' and birthday='1960-01-01'; |

**匹配最左前缀**

比如只使用索引的第一列，查找所有姓为Allen的人。SQL语句为：

|  |  |
| --- | --- |
|  | select id,last\_name,first\_name,birthday from people where last\_name='Allen'; |

**匹配列前缀**

比如只匹配索引的第一列的值的开头部分，查找所有姓氏以A开头的人。SQL语句为：

|  |  |
| --- | --- |
|  | select id,last\_name,first\_name,birthday from people where last\_name like ‘A%'; |

**匹配范围值**

比如范围匹配姓氏在Allen和Clinton之间的人。SQL语句为：

|  |  |
| --- | --- |
|  | select id,last\_name,first\_name,birthday from people where last\_name BETWEEN ‘Allen' And ‘Clinton'; |

这里也只使用了索引的第一列。

**精确匹配第一列并范围匹配后面的列**

比如查找姓Allen，并且名字以字母C开头的人。即全匹配复合索引的第一列，范围匹配第二列。SQL语句为：

|  |  |
| --- | --- |
|  | select id,last\_name,first\_name,birthday from people where last\_name = ‘Allen' and first\_name like'C%'; |

**只访问索引的查询**

B-Tree 通常可以支持“只访问索引的查询”，即查询只需要访问索引，而无需访问数据行。这和“覆盖索引”的优化相关，后面再讲。