## 一 Mysql索引

### 1.1 什么是索引

索引是数据库中用来提高性能的最常用工具。所有 MySQL 列类型都可以被索引，对相关列使用索引是提高 SELECT 操作性能的最佳途径。根据存储引擎可以定义每个表的最大索引数和最大索引长度，每种存储引擎（如 MyISAM、InnoDB、BDB、MEMORY 等）对每个表至少支持 16 个索引，总索引长度至少为 256 字节。大多数存储引擎有更高的限制。MyISAM 和 InnoDB 存储引擎的表默认创建的都是 BTREE 索引

### 1.2 索引的使用

#### 1.2.1 创建索引

alter table table\_name add index index\_name (column\_list) ;  
 alter table table\_name add unique (column\_list) ;  
 alter table table\_name add primary key (column\_list) ;

#### 1.2.2 删除索引

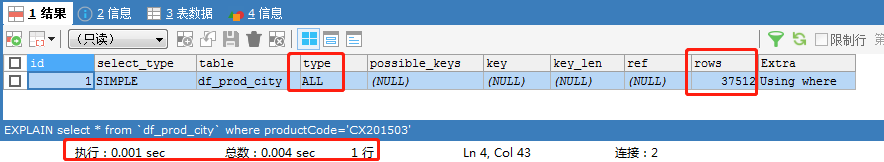
drop index index\_name on table\_name ;  
alter table table\_name drop index index\_name ;  
alter table table\_name drop primary key ;

#### 1.3.3 索引实例

(1) 步奏一 无索引查询

EXPLAIN SELECT \* FROM `df\_prod\_city` WHERE productCode='CX201503';

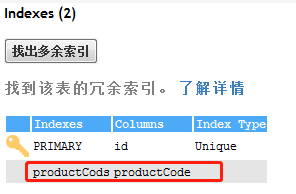
在没有添加索引的时候



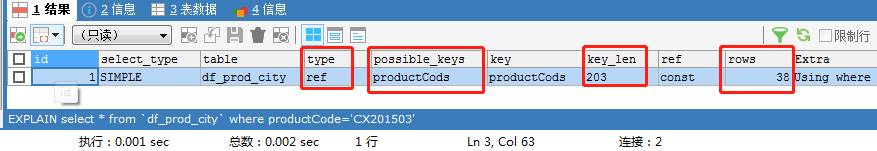
如上所示的sql语句，在没有使用索引的情况下，做了一次全表扫描，查询了37512个数据行。

1. 步奏二 添加索引

ALTER TABLE df\_prod\_city ADD INDEX productCods (productCode) ;



1. 步奏三 查询



此时再次查询的时候，你会发现查询性能得到了提升。

### 1.3 索引原则

(1)最适合索引的列是出现在 WHERE 子句中的列或连接子句中的列。

1. 索引的列的基数越大，索引的效果越好。例如，存放出生日期的列具有不同值，很容易区分各行。而用来记录性别的列，只含有“ M”和“F”，则对此列进行索引没有多大用处，因为不管搜索哪个值，都会得出大约一半的行。
2. 指定一个前缀长度，只要有可能就应该这样做。例如，如果有一个 CHAR(200)列，如果在前 10 个或 20 个字符内，多数值是惟一的，那么就不要对整个列进行索引。对前10个或20个字符进行索引能够节省大量索引空间，也可能会使查询更快。较小的索引涉及的磁盘 IO 较少，较短的值比较起来更快。更为重要的是，对于较短的键值，索引高速缓存中的块能容纳更多的键值，因此，MySQL 也可以在内存中容纳更多的值。这样就增加了找到行而不用读取索引中较多块的可能性。

(4) 利用最左前缀。在创建一个 n 列的索引时，实际是创建了 MySQL 可利用的 n 个索引。

多列索引可起几个索引的作用，因为可利用索引中最左边的列集来匹配行。这样的列集称为

最左前缀。

(5)不要过度索引。不要以为索引“越多越好”，什么东西都用索引是错误的。每个额

外的索引都要占用额外的磁盘空间，并降低写操作的性能。在修改表的内容时，索引必须进

行更新，有时可能需要重构，因此，索引越多，所花的时间越长。如果有一个索引很少利用

或从不使用，那么会不必要地减缓表的修改速度。此外，MySQL 在生成一个执行计划时，

要考虑各个索引，这也要花费时间。创建多余的索引给查询优化带来了更多的工作。索引太

多，也可能会使 MySQL 选择不到所要使用的最好索引。只保持所需的索引有利于查询优化。

(6) 对于 InnoDB 存储引擎的表，记录默认会按照一定的顺序保存，如果有明确定义的主

键，则按照主键顺序保存。如果没有主键，但是有唯一索引，那么就是按照唯一索引的顺序

保存。如果既没有主键又没有唯一索引，那么表中会自动生成一个内部列，按照这个列的顺

序保存。按照主键或者内部列进行的访问是最快的，所以 InnoDB 表尽量自己指定主键，当

表中同时有几个列都是唯一的，都可以作为主键的时候，要选择最常作为访问条件的列作为

主键，提高查询的效率。另外，还需要注意，InnoDB 表的普通索引都会保存主键的键值，

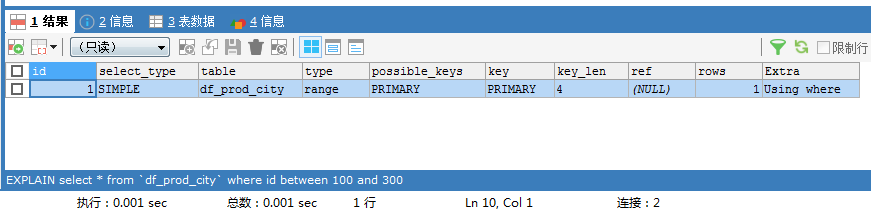
所以主键要尽可能选择较短的数据类型，可以有效地减少索引的磁盘占用，提高索引的缓存

效果。

### 1.4 Btree索引与hash索引

(1)b-tree索引查看

EXPLAIN SELECT \* FROM `df\_prod\_city` WHERE id BETWEEN 100 AND 300;



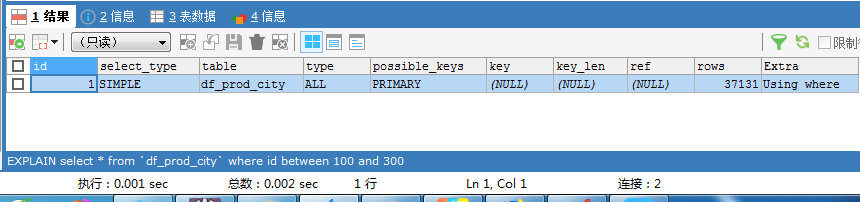
这就是使用了BTree索引来完成的搜索。由上面的执行计划可以看出，是通过了btree索引了没有进行全表扫描。

(2)修改存储引擎

ALTER TABLE df\_prod\_city ENGINE=MEMORY

1. Hash索引查看

EXPLAIN SELECT \* FROM `df\_prod\_city` WHERE id BETWEEN 100 AND 300;



由上可以看出hash索引跟b-tree 索引的区别，使用memory存储引擎的时候，对范围检索的效果不太好。所以我们在识别搜索引擎的时候只要通过区间去检索，再查询sql执行计划就能知道是用了什么存储引擎了。