# 一、索引是什么

在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（或称指向）数据。这样就可以在这些数据结构上实现高级查找算法，这种数据结构就是索引（index）。

索引是帮助数据高效获取数据的数据结构。所以索引的本质就是数据结构。也可以理解为”排好序的快速查找数据结构”。

索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录。

# 二、索引的优劣势

## 1、优势

索引的建立对于数据库的高效运行是很重要的，索引可以大大提高数据库的检索（即查询）效率，降低数据库的IO成本。

通过索引对数据进行排序，降低数据排序的成本，降低了CPU的消耗。

## 2、劣势

上面说到索引也是一张表，该表保存了主键与索引字段，并指向实体表的记录，所以索引列也是要占空间的。

　　虽然索引大大提高了查询速度，但过多的使用索引将会造成滥用，同时确会降低更新表的速度（如：增、删、改）。 因为更新表时，数据库不仅要保存数据，还要保存一下索引文件每次更新添加了索引列的字段都会调整因为更新所带来的键值变化后的索引信息。

建立索引会占用磁盘空间的索引文件。

# 三、索引的分类

主键索引、唯一索引、普通索引、全文索引、组合索引

# 四、基本语法

## 1、普通索引（INDEX）

语法：ALTER TABLE 'table' ADD INDEX index\_name('col')

最基本的索引，没有任何限制

## 2、唯一索引（UNIQUE）

语法：ALTER TABLE 'table' ADD UNIQUE('col')

与“普通索引”类似，不同的就是：索引列的值必须唯一，但允许有空值。

## 3、主键索引（PRIMARY KEY）

语法：ALTER TABLE 'table' ADD PRIMARY KEY('col')

是一种特殊的唯一索引，不允许有空值。

## 4、全文索引（FULLTEXT）

语法：ALTER TABLE 'table' ADD FULLTEXT('col')

仅可用于MyISAM和InoDB，针对较大的数据，生成全文索引很耗时耗空间

## 5、组合索引

语法：ALTER TABLE 'table' ADD INDEX index\_name('col1','col2','col3')

为了更多的提高mysql效率可建立组合索引，遵循“最左前缀”原则。创建复合索引应该将最常用（频率）做限制条件的列放在最左边，依次递减。组合索引最左字段用in是可以用到索引的。相当于建立了col1,col1col2,col1col2col3三个索引。

## 6、修改索引名称（以mysql为例）

对于MySQL 5.7及更新版本,可以执行以下命令：

ALTER TABLE tbl\_name RENAME INDEX old\_index\_name TO new\_index\_name

对于MySQL 5.7以前的版本，可以执行下面两个命令：

ALTER TABLE tbl\_name DROP INDEX old\_index\_name

ALTER TABLE tbl\_name ADD INDEX new\_index\_name(column\_name)

# 五、索引的使用原则（怎样设计索引）

## 1、哪些情况需要创建索引

①主键自动建立唯一索引

②频繁作为查询条件的字段应该创建索引

③查询中与其他表关联的字段，外键关系建立索引

④频繁更新的字段不适合建立索引，因为每次更新不单单是更新了记录还会更新索引

⑤WHERE条件里用不到的字段不创建索引

⑥单键/组合索引的选择问题，who?(在高并发下倾向创建组合索引)

⑦查询中排序的字段，排序的字段若通过索引去访问将大大提高排序速度

⑧查询中统计或者分组字段

## 2、哪些情况不要创建索引

①表记录太少

②经常增删改的表

③数据重复且分布平均的表字段，只为最经常查询和最经常排序的数据建立索引。

④某个数据列包含许多重复的内容，为它建立索引就没有太大的实际效果。

## 3、索引使用注意事项

### （1）不要滥用索引

①索引提高查询速度，却会降低更新表的速度，因为更新表时，数据库不仅要更新数据，保存数据，还要更新索引，保存索引

②索引会占用磁盘空间

### （2）索引不会包括含有NULL值的列

复合索引只要有一列含有NULL值，那么这一列对于此复合索引就是无效的，因此我们在设计数据库设计时不要让字段的默认值为NULL。

### （3）MySQL查询只适用一个索引

如果where字句中使用了索引的话，那么order by中的列是不会使用索引的

### （4）like

like '%aaa%'不会使用索引而like "aaa%"可以使用索引

# 六、索引失效的原因

### 1、索引列不独立

是指被索引的这列不能是表达式的一部分，不能是函数的参数，比如下面的这种情况

**select** id,name,age **from** user **where** age **+** 1 **=** 10;

### 2、使用了左模糊

like '%aaa%'会使索引失效，而like "aaa%"不会使索引失效

### 3、使用 or 查询部分字段没有使用索引

select id,name,age from user where name = '小明' or age >10;

这种情况，可以为 name 和 age 都建立索引，否则会走全表扫描，不使用索引。

### 4、字符串条件没有使用引号

select id,name,age from user where name=xiaoming

上面的这条sql，name字段类型是字符串类型的，但是没有使用引号, SQL就全表扫描了，而没有使用索引。

### 5、不符合最左前缀原则的查询

例如有这样一个组合索引 index(a,b,c)

（1）这两条 SQL 都是无法走索引执行的

select \* from table where b='1' and c='2';

select \* from table where c='2';

最左原则，就是要最左边的优先存在，我不在的话，你们自己就玩不动了，除非你自己单独创立一个索引。

（2）下面这几条SQL就可以走索引执行

select \* from table where a = '0' and b='1'and c='2'

select \* from table where a = '0' and b='1'

（3）下面这条部分走索引。

select \* from table where a = '1' and c='2'

### 6、索引字段没有添加 not null 约束

select \* from user where name is null;

// 这条sql就无法走索引执行了，is null 条件不能使用索引，只能全表扫描了

// mysql 官方建议是把字段设置为 not null所以针对这个情况，在mysql 创建表字段的时候，可以将需要索引的字符串设置为 not null default '' 默认空字符串即可

### 7、隐式转换导致索引失效

关联表的两个字段类型不一致会发生隐式转换，

例1：

select \* from table1 t1 left join table2 t2 on t1.id=t2.id;

上面这条语句里，如果 t1 表的id 类型和 t2 表的tid 类型不一致的时候，就无法按索引执行了。

例2：a字段为varchar类型

select \* from table where a = '1';会走索引

select \* from table where a = 1;不会走索引

// 解决方式就是统一设置字段类型。