日常业务开发中，最核心的其实就是写SQL命令，但是你写的SQL真的用到索引了吗？

### 索引实现原理

索引实现是在存储引擎中完成的，主要以InnoDB存储引擎为例。采用B+Tree数据结构。

### 索引失效

失效的场景：

1. 少用or，用它来连接时索引会失效。
2. like查询以%开头。
3. 字符串不加单引号索引失效。
4. 如果mysql估计使用全表扫描要比索引快，则不使用索引。
5. 在mysql使用不等于（<,>,!=）的时候无法使用索引，会导致索引失效。
6. is null或者is not null也会导致无法使用索引。
7. 不在索引列上做任何操作（计算，函数（自动或手动）类型转换），会导致索引失效而导致全表扫描。
8. 如果索引了多列，要遵守最左前缀法则。

### 索引优化原则

#### 给需要的字段加索引：

一般需要加索引的都是where中经常使用的字段，但是像性别这种属性字段，加索引的意义不大，因为性别这类字段选择性太低（基础/总数）

如下所示，如果userinfo表中的数据90%以上，都是性别为男的用户，这个时候给sex字段加索引，收效甚微。

select sum(sex=‘男’) as sexCount,sum(1) as countNum from userinfo;

#### 错误的为每个列创建独立索引：

有些人可能会把where条件后面所有的字段都加上索引，但是后面却会发现这样做起到的效果收效甚微，这个时候我们应该优先使用联合索引。

如下所示：虽然我给name和age都设置了索引，但是真正使用到的只有name索引。如果用户表中存在大量名叫张三的用户，那age就需要进行二次筛选，这就会耗费相当的资源。

select \* from userinfo where name= '张三' and age=21;

索引使用情况：

SIMPLE userinfo ref name,age name 768 const 1 33.33 Using where

如果给name+age设置联合索引，就可以快速检索到我们需要的数据，不需要进行二次筛选。

SIMPLE userinfo ref name\_age name\_age 773 const,const 1 100

#### 正确使用联合索引：

联合索引一定要注意索引顺序，一般放在前面的都是选择性比较高的索引字段。

例如一个用户表，需要查询手机号为138+性别为女的数据，如果是性别+手机号这样的顺序作为联合索引，就会导致第一次索引的数据很大，再用手机接着索引效率就会特别低，如果索引顺序反过来就可以避免这类问题。

#### 避免检索全部：

一定要避免select \* from的出现，这样不仅不会用到覆盖索引，而且数据传输也会耗费很长的时间。如果select出来的字段都在where条件之内，这种情况就叫做覆盖索引，覆盖索引检索出来的数据可以直接返回给前端，不需要通过id去查询其它字段，所以效率更高。

#### order by排序字段注意重排序：

使用索引字段来做排序，如果order by排序的字段不在索引中，则会导致数据重排，如果数据量较大会很耗费cpu性能。

如下所示，pwd字段不在索引里面，就会导致Using filesort的出现。

EXPLAIN

select \* from userinfo where name= '张三' and age=21 ORDER BY pwd;

索引使用情况：

SIMPLE userinfo ref name\_age name\_age 773 const,const 1 100 Using

index condition; Using filesort

#### 去除冗余索引：

去除不必要的冗余索引，在添加索引的时候，可能有人不注意会设置重复的索引，像用户名称+年龄的联合索引和用户名称的单列索引，就是典型的冗余索引字段。

#### SQL索引检查：

我们写完SQL，要如何确定这条SQL有没有用到索引，用到的索引具体是哪一个，可以用explain关键字来进行SQL索引分析。



### 总结