# MySQL优化

所有的优化其实都是一个工程问题，方案没有优劣，只有适合。

不要为了优化而优化。

当性能问题出现了，或者根据经验、根据测算在某个时间将会出现性能问题，再去优化。

一次只执行一个优化方案，看结果之后再决定是否执行下一个优化方案。不要一上来就把所有优化方案都执行了，到时候谁知道是哪个方案起作用。

## 优化

**大原则就是“让索引生效”。如无特别说明，下面所有的优化方案大家都记住“尽量”二字。**

1. 在WHERE、ORDER BY的字段上建索引。
2. LIKE “**%**ABC%”，最左边的**%**不要有，这会导致直接全文检索。”abc%”这种没问题
3. 字段尽量不要有NULL，可以设置默认值，哪怕这个默认值代表的意义是NULL。
4. NOT、<>、!=不使用索引，而<，<=，=，>，>=，BETWEEN，IN使用索引。

另外IN要多说一句，很多资料上说它会使用索引，但你用不好，比如在里面加了好多条件，会导致全表扫描。具体什么时候用索引，什么时候不用，我毕竟不是DBA，还真没发现规律，有精通数据库的朋友可以教教我。可以试试用exists替代IN。

1. WHERE后的字段不要用OR，如果某一个字段没有索引，那么整个查询都不会用索引。

但是，如果OR的字段都有**独立索引**，查询的时候会使用索引。

1. WHERE后的索引字段不要做运算操作、表达式操作、函数操作，索引会失效。

比如：字段总价（totalprice）做了索引。

WHERE totalprice/10<=1000不会用索引。

WHERE totalprice<=10000会用索引。

又比如：入表时间（intime）做了索引。

WHERE to\_char(intime,’yyyy-MM-dd’)=’2017-11-20’不使用索引。

WHERE intime=to\_date(‘2017-11-20’,’yyyy-MM-dd’)会使用索引。

1. 用UNION ALL，尽量别用UNION。
2. 别用SELECT \*。
3. UPDATE的时候，需要更新几个字段就更新几个字段。

比如：UPDATE SET name=’new name’ WHERE id=1;就很好

UPDATE SET name=’new name’,**address=’old address’,dept=’old dept’** WHERE id=1;

address和dept就没必要了。

1. 当INSERT很多很多数据的时候，要拆分，阶段性处理。
2. 复合索引，WHERE的顺序要一致。

比如：create index index1 on tabA(col1,col2,col3);

其实创建的索引是（col1,col2,col3）、（col1,col2）、（col1）。

WHERE col1=1 and col2=2 and col3=3

WHERE col1=1 and col2=2;

WHERE col1=1;

索引起作用。

WHERE col2=2 and col3=3;

WHERE col2=2;

WHERE col3=3;

索引不起作用。

1. MySQL查询的数据量**超过30%**的时候，不会用到索引。
2. 对于性别、状态这种只有1、2个内容的字段，不用建索引。
3. 尽量避免耗时过长的事务。
4. 开启慢查询日志。

打开mysql的配置文件：

#开启慢查询日志

**slow\_query\_log = ON**

#慢查询日志文件

**slow\_query\_log\_file = 绝对路径/slowsql.log**

#查询耗时超过1秒就记录

**long\_query\_time = 1**

1. 用好EXPLAIN。

各属性的含义（不太重要的我就不列了）：

**select\_type：**查询的类型。

simple，最普通的查询。

primary，如果有子查询，primary代表的是最外面的那个SELECT。

union，联合查询。

**table：**用的哪个表。

**type：**连接所用的类型，这一列对于优化来说很重要。从好到查，依次为：

system，表仅有一行，很少见。

const，表只有一行是匹配的。主要是用在主键索引或唯一索引上。

eq\_ref，对于每个来自于前面的表的行组合，从该表中读取一行。

ref，对于每个来自于前面的表的行组合，所有有匹配索引值的行将从这张表中读取。

ref\_or\_null，同ref，添加了可以专门搜索NULL值的行。

range，只检索给定范围的行，key列显示用了哪个索引。

index，索引树被扫描。

all，全表扫描。

**key：**用了哪个索引。

**rows：**查询的行数，越大越不好。

**extra：**解决查询的详细信息，这个很重要。

using index，查询的信息从索引上得来的，没有查表。

where used，用了where的限制。

impossible where，没用上where。

using filesort，在排序的时候需要额外的步骤，要对排序进行优化。

using temporary，使用了临时表。

## MySQL主从同步延时

### 主从的原理

主库将DDL、DML操作写入binlog。

从库slave\_IO\_running线程读取binlog，slave\_SQL\_running线程负责执行。

### 主从同步延时的原因

1. 主库有某个很耗时的DDL操作，或者有很多DDL操作，主库可以走并发，但从库的slave\_SQL\_running是单线程，很可能就超过处理能力，卡在某个操作上了。
2. 从库有个很耗时的查询，把表锁了。