# mysql 技巧

## 常见的SQL实践

1,负向条件查询不能使用索引

select \* from order where status !=0 and status!=1;

not in / not exists 都不是好习惯

可以优化为in查询：

select \* from order where status in(2,3);

2,前导模糊查询不能使用索引

select \* from order where desc like ‘%XX’;

而非前导模糊查询则可以使用索引

select \* from order where desc like ‘XX%’;

3,数据度区分度不大的字段不宜使用索引

select \* from user where sex =1;

原因：性别只有男、女，每次过滤掉的数据很少不宜使用索引

经验上，能过滤80%数据时可以使用索引，对于订单状态，如果状态值很少，不宜使用索引，如果状态值很多，能够过滤大量数据，则应该使用索引

4,在属性上进行计算不能命中索引

select \* from order where YEAR(date) < ‘2017’;

即使date上建立了索引，也会全表扫描，可优化为值计算

select \* from order where date <= CURDATE();

或者 select \* from order date <= ‘2017-01-01’;

5,如果业务大部分是单条查询，使用HASH索引性能更好，例如用户中心

select \* from user where uid = ?;

select \* from user where login\_name=?;

原因：B-Tree索引的时间复杂度是O(log(n));Hash索引的时间复杂度为O(1);

6,允许为null的列，查询有潜在的大坑；

单列索引不存null值，复合索引不存全为null的值，如果允许为null，可能会得到不符合预期的结果集

select \* from user where name != ‘shenjian’;

如果name允许为null，索引不存储null值，结果集中不会包含这些记录；所以，使用not null约束以及默认值

7,复合索引最左前缀，并不是SQL语句的where顺序要和复合索引一致

用户中心建立了(login\_name,passwd)的复合索引

select \* from user where login\_name = ? and passwd = ?;

select \* from user where passwd = ? and login\_name = ?;

都能命中索引

select \* from user where login\_name = ?;也能命中索引，满足复合索引最左前缀

select \* from user where passwd = ?;不能命中索引，不满足复合索引最左前缀

8,使用ENUM而不是字符串

ENUM保存的是TINYINT,别在枚举中搞一些’中国’’北京’’技术部’这样的字符串，字符串空间又大，效率又低

9,如果明确知道只有一条结果返回，limit 1 能够提高效率

select \* from user where login\_name = ?;

可以优化为 select \* from user where login\_name = ? limit 1;

原因：你知道只有一条结果，但数据库不知道，明确告诉，让它主动停止游标移动

10,把计算放到业务层而不是数据层，除了节省数据库的CPU，还有意想不到的查询缓存优化效果

select \* from order where date <= CURDATE();

不是个好的sql实践，应该优化为：

$curDate = date(‘Y-m-d’);

$res = mysql\_query(‘select \* from order where order <= $curDate’);

原因：释放了数据库的CPU，多次调用，传入的SQL相同，才可以利用查询缓存

11,强制类型转换会全表扫描

select \* from user where phone = 15013800920;

需优化为

select \* from user where phone = ‘15013800920’;