# SQL优化（系列）-SQL优化策略

**一、优化策略**

    SQL优化策略适用于数据量较大的场景下，如果数据量较小，没必要以此为准。

**1.1、尽量避免在字段开头模糊查询。**

    尽量避免在字段开头模糊查询，会导致数据库引擎放弃索引进行全表扫描。如下：

SELECT \* FROM t WHERE username LIKE '%陈%'

优化方式：尽量在字段后面使用模糊查询。如下：

SELECT \* FROM t WHERE username LIKE '陈%'

    如果需求是要在前面使用模糊查询，使用MySQL内置函数INSTR(str,substr) 来匹配，作用类似于java中的indexOf()，查询字符串出现的角标位置。

    数据量较大的情况，建议引用ElasticSearch、solr，亿级数据量检索速度秒级；当表数据量较少（几千条儿那种），别整花里胡哨的，直接用like '%xx%'。

**1.2、尽量避免使用in 和not in，会导致引擎走全表扫描**

SELECT \* FROM t WHERE id IN (2,3);

优化方式：如果是连续数值，可以用between代替。如下：

SELECT \* FROM t WHERE id BETWEEN 2 AND 3;

**1.3、尽量避免使用 or关键词**

SELECT \* FROM t WHERE id = 1 OR id = 3

优化方式：可以用union代替or。如下：

SELECT \* FROM t WHERE id = 1UNIONSELECT \* FROM t WHERE id = 3

**1.4、尽量避免进行null值的判断**

SELECT \* FROM t WHERE score IS NULL

优化方式：可以给字段添加默认值0，对0值进行判断。如下：

SELECT \* FROM t WHERE score = 0

**1.5、尽量避免在where条件中等号的左侧进行表达式、函数操作**

可以将表达式、函数操作移动到等号右侧。如下：

-- 全表扫描SELECT \* FROM T WHERE score/10 = 9-- 走索引SELECT \* FROM T WHERE score = 10\*9

**1.6、避免使用where 1=1的条件**

    当数据量大时，避免使用where 1=1的条件。通常为了方便拼装查询条件，我们会默认使用该条件，数据库引擎会放弃索引进行全表扫描。如下：

SELECT username, age, sex FROM T WHERE 1=1

优化方式：

    用代码拼装sql时进行判断，没 where 条件就去掉 where，有where条件就加 and。

**1.7、查询条件不能用 <> 或者 !=**

    使用索引列作为条件进行查询时，需要避免使用<>或者!=等判断条件。如确实业务需要，使用到不等于符号，需要在重新评估索引建立，避免在此字段上建立索引，改由查询条件中其他索引字段代替。

**1.8、where条件仅包含复合索引非前置列**

    复合（联合）索引包含key\_part1，key\_part2，key\_part3三列，但SQL语句没有包含索引前置列"key\_part1"，按照MySQL联合索引的最左匹配原则，不会走联合索引。

select col1 from table where key\_part2=1 and key\_part3=2

**1.9、隐式类型转换造成不使用索引**

    SQL语句由于索引对列类型为varchar，但给定的值为数值，涉及隐式类型转换，造成不能正确走索引。

select col1 from table where col\_varchar=123;

1.10、**order by 条件要与where中条件一致，否则order by不会利用索引进行排序**

*-- 不走age索引*SELECT \* FROM t order by age; *-- 走age索引*SELECT \* FROM t where age > 0 order by age;