链滴

sql优化常见策略

作者：guobing

原文链接：https://ld246.com/article/1472205239102

来源网站：链滴

常见的优化策略
从最核心的开始说起
###1. 选择正确的存储引擎
这个是最核心的了，选错存储引擎那其他的都是白忙活了。那最常用的就是InnoDB了，那如果事务要求不高的话，可以选择myisam了。

###2. 不能用存在null值的列做索引
不能用null作索引，任何包含null值的列都将不会被包含在索引中。即使索引有多列这样的情况下，要这些列中有一列含有null，该列就会从索引中排除。也就是说如果某列存在空值，即使对该列建索也不会提高性能。

###3. where后面的索引不一定100%有效
任何在where子句中使用is null或is not null的语句优化器是不允许使用索引的。

>* 尽量避免null类型的列
使得索引、索引统计、值比较都比较复杂、使用更多的存储空间、null的列为索引时，每个索引记录要额外一个字节、
>* datetime timestamp
timestamp只使用datetime一半的存储空间。并且会根据时区变化。
>* 整数类型
tinyint、smallint、mediumint、int、bigint分别使用8/16/24、32/64位存储空间
unsigned属性表示不允许负值
>* 字符串类型
varchar char的区别
varchar存储可变长的字符串、需要用额外1、2一个字节记录字符串长度信息、适合最大长度比平均度大很多。。列的更新很少，碎片不是问题
char是定长的，适合经常变更的数据、因为不容易产生碎片。也适合非常短的列、比如char(1)存储yn的值，char只需要一个字节，而varchar需要两个字节，额外一个字节存储长度信息

###4. mysql手册里推荐的优化策略
####4.1 MySQL怎样优化WHERE子句
>* 去除不必要的括号：
`((a AND b) AND c OR (((a AND b) AND (c AND d))))`应该写为 -> `(a AND b AND c) OR (a AND b AND c AND d)`
>* 常量重叠：
`(a<b AND b=c) AND a=5`
改为-> `b>5 AND b=c AND a=5`
>* 去除常量条件(由于常量重叠需要)：
`(B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)`
·改为： -> `B=5 OR B=6`
>* 尽量避免在 where 子句中使用!=或<>操作符,容易走全表扫描
>* 首先应考虑在 where 及 order by 涉及的列上建立索引。
>* 尽量避免在 where 子句中对字段进行 null 值判断,否则会走全表扫描,例如：
`select id from t where num is null`会走全表扫描。可以设成默认值0
>* 避免在 where 子句中使用 or 来连接条件.否则会走全表扫描。解决办法是多个条件用union all来接。
>* like查询%不要前置，这样会走全表扫描。例如：
`select id from t where name like  '%cc%'  `,酱紫不好。
>* in 和 not in 也要慎用，否则会导致全表扫描.尽量用between and替换
>* 如果在 where 子句中使用参数，也会导致全表扫描。因为SQL只有在运行时才会解析局部变量，优化程序不能将访问计划的选择推迟到运行时；它必须在编译时进行选择。然 而，如果在编译时建立问计划，变量的值还是未知的，因而无法作为索引选择的输入项。如下面语句将进行全表扫描：
`select id from t where num=@num`
可以改为强制查询使用索引：

`select id from t with(index(索引名)) where num=@num`
>* 应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描
如：`select id from t where num/2=100`应改为:`select id from t where num=100*2`
>* 很多时候用 exists 代替 in 是一个好的选择：`select num from a where num in(select num fro b)`用下面的语句替换：`select num from a where exists(select 1 from b where num=a.num)`
>* 只有低选择性的数据行才有索引，一般根据经验来说，查出的数据量大于表数据量21%以上的话走索引。
>* 根据经验，一个表的索引数最好不要超过6个
>* 尽量使用数字型字段.若只含数值信息的字段尽量不要设计为字符型，这会降低查询和连接的性能并会增加存储开销。这是因为引擎在处理查询和连接时会 逐个比较字符串中每一个字符，而对于数字而言只需要比较一次就够了。
>* 避免使用`select * `

###4.2 复合索引优化
　索引可以包含一个、两个或更多个列。两个或更多个列上的索引被称作复合索引。

复合索引有个重要的原则就是`最左前缀原则`。
复合索引起作用的原则是
1. 必须包含最左边一个索引
2. 索引顺序不能改变
如果 (col1，col2，col3)有一个索引，则起作用的索引是col1、(col1, col2)、(col1, col3);
这就是最左前缀代表的意义。

###4.3 复合索引对排序的优化
　复合索引只对和索引中排序相同或相反的order by 语句优化。

在创建复合索引时，每一列都定义了升序或者是降序。如定义一个复合索引：
`CREATE INDEX idx_example
ON table1 (col1 ASC, col2 DESC, col3 ASC)`
其中 有三列分别是：`col1` 升序，`col2` 降序， `col3` 升序。现在如果我们执行两个查询

1：`Select col1, col2, col3 from table1 order by col1 ASC, col2 DESC, col3 ASC `和索引顺序相同

2：`Select col1, col2, col3 from table1 order by col1 DESC, col2 ASC, col3 DESC `和索引顺序相

查询1，2 都可以别复合索引优化。

如果查询为：

`Select col1, col2, col3 from table1 order by col1 ASC, col2 ASC, col3 ASC `排序结果和索引完全同时，此时的查询不会被复合索引优化。

备注：部分来自《mysql技术内幕》