Please visit website: http://cxyroad.com

Mysql中使用where 1=1有什么问题吗

昨天偶然看见一篇文章，提到说如果在mysql查询语句中，使用`where 1=1`会有性能问题! 这着实把我吸引了，因为我项目中就有不少同事，包括我自己也有这样写的。为了不给其他人挖坑，赶紧学习一下，这样写sql到底有没有性能问题?

`where 1=1`的使用场景

我们来看下，`where 1=1`实际是怎么使用的， 先来看一段SQL

```
<select id="selectCmsCourseList" parameterType="java.util.Map" resultMap="CourseMap">
    SELECT
    a.id,
    a.category_id,
    a.model_id,
    b.*
    FROM
    cms_content a
    LEFT JOIN cms_course b ON a.id = b.id
    WHERE 1=1
    <if test="courseName != null and courseName != ''">
        AND b.course_name like concat(concat("%",#{courseName}),"%")
    </if>
    <if test="disabled != null">
        AND a.disabled = #{disabled}
    </if>
    <if test="categoryId != null">
        AND a.category_id = #{categoryId}
    </if>
</select>
```

如果用过mybatis的童鞋，看到这段代码应该很熟悉了吧，这里使用`where 1=1`的作用，就是为了方便后面的条件，能通过`if`判断，使用`AND`连接起来，这样即使后面`if`全部是`false`，没有参数，这个`sql`也不会报错。

`where 1=1`的替换方案

其实上面的这个写法，如果用mybatis，完全可以用`where`标签来替换,如：

```
<select id="selectCmsCourseList" parameterType="java.util.Map"
resultMap="CourseMap">
    SELECT
    a.id,
    a.category_id,
    a.model_id,
    b.*
    FROM
    cms_content a
    LEFT JOIN cms_course b ON a.id = b.id
    <where>
        <if test="courseName != null and courseName != ''">
            AND b.course_name like
concat(concat("%",#{courseName}),"%")
        </if>
        <if test="disabled != null">
            AND a.disabled = #{disabled}
        </if>
        <if test="categoryId != null">
            AND a.category_id = #{categoryId}
        </if>
    </where>
</select>
```

如果你使用了这个写法，就不存在`1=1`的问题了，所以也不用纠结，但本着打破砂锅问到底的精神，我们还是得探究一下使用`1=1`到底有没有性能问题

使用`where 1=1`有性能问题吗?

先来问问AI


可以看到，AI给到的答复是，基本没有性能问题，更多的是代码风格和可读性！

亲自检测

为了跟进一步测试，我们来写个SQL看看实际性能是否有差别

1. `explain select * from user;`

2. `explain select * from user where 1=1;`

3. `explain select * from user where id = 8;`

4. `explain select * from user where 1=1 and id = 8;`

可以看到示例1和示例2， 示例3和示例4，这两组写法，都是有个`1=1`的区别，但性能都是一样的，因此我们基本可以判定，这个写法，对性能是没有影响的。

总结

--

如果你在项目中也有使用`where 1=1`写的SQL，如果从可读性或者代码风格考虑，建议优化掉，但如果担心性能问题，则大可不必过多考虑~

原文链接: https://juejin.cn/post/7379263199115968564