

可重复读简介



‘COUNT()’ 不能直接应用在返回多行的子查询上，因为 ‘COUNT()’ 期望一个单列或单值。
那这种情况和COUNT(*)基于多行不是矛盾了吗？



你提到的疑惑是关于 ‘COUNT(*)’ 基于多行来统计时，为什么不能在子查询中处理多行的结果。这涉及到两种不同的 SQL 语句情境，‘COUNT(*)’ 和 **子查询** 的区别，并且它们的行为并不矛盾，下面我详细解释：

1. ‘COUNT(*)’ 的工作方式

- ‘COUNT(*)’ 是 SQL 中的聚合函数，用来统计表中**所有行数**，无论列的具体值是什么。
- 这个操作是对**整个结果集的行**进行统计，它不关心具体哪些列，甚至某些列的值是否为 ‘NULL’。

例子：

```
sql
```

```
SELECT COUNT(*) FROM students;
```

这个查询返回的是表中所有行的总数，假设表中有 4 行，那么结果就是 ‘4’。这个操作没有指定列，也不依赖列的具体内容。

2. 为什么子查询中的 ‘COUNT()’ 无法处理多行返回值

这里的关键在于：‘COUNT()’ 的处理对象是整个查询上下文中的表达式或单列，而不是子查询的多行返回结果。

子查询返回多行的问题：

当你在外部查询中使用一个子查询时，如果这个子查询返回多行（如多行的 ‘sec_id’），那么 ‘COUNT()’ 无法处理这些行，因为它期望的是一个表达式或列，而不是一个子查询的结果集。

错误的示例：

```
sql
```

```
SELECT
  i.ID AS instructor,
  i.name,
  COUNT((SELECT sec_id
          FROM teaches t
```

```
WHERE i.ID = t.ID)) AS number_of_section
FROM instructor i;
```

- **问题：**子查询 `SELECT sec_id FROM teaches t WHERE i.ID = t.ID` 可能返回多个 `sec_id`，但 `COUNT()` 无法对这个多行结果进行处理，因为它期望的是一个单值或列的表达式。
- **根本原因：**在 SQL 中，子查询返回的多行结果在作为聚合函数的参数时是不被允许的，`COUNT()` 函数不能直接作用于子查询的多行结果，必须确保它处理的是单个数值或单列。

3. 对比 `COUNT(*)` 和子查询

- `COUNT(*)` 在它的使用上下文中（即在查询的 FROM 子句之后）统计整个结果集的行数，遍历所有行，因此它是基于**整个结果集**的，而不是某个子查询。

sql

```
SELECT COUNT(*) FROM teaches;
```

这个查询会返回 `teaches` 表中的行数。

- **子查询中的 `COUNT()`：**当我们需要通过子查询统计某个特定范围的行数时，我们可以在**子查询内部**使用 `COUNT()`，而不是返回多行数据给外部查询处理。

sql

```
SELECT
  i.ID AS instructor,
  i.name,
  (SELECT COUNT(t.sec_id)
   FROM teaches t
   WHERE t.ID = i.ID) AS number_of_section
FROM instructor i;
```

这里，子查询 `SELECT COUNT(t.sec_id) ...` 返回的是一个单一的值（某个教师教授课程的数量），而不是多行数据，因此外部查询可以正常处理这个值。

4. 总结：两者行为不矛盾的原因

- `COUNT(*)` 是对整个结果集进行统计，统计的是行数，而不涉及多行数据的问题。
- **子查询中返回多行数据**的问题是，它在作为外部查询的参数时会导致问题，因为外部查询无法处理子查询返回的多行数据。而 `COUNT()` 聚合函数期望的输入是单列或单个值，因此无法接受多行子查询作为参数。
- **解决方法**是在子查询内部进行聚合统计，比如使用 `COUNT()` 返回一个单值结果给外部查询。

总结：

`COUNT(*)` 统计的是表中的行数，这与子查询返回多行数据所引发的错误并不矛盾。关键在于，`COUNT(*)` 基于行数进行操作，而子查询返回的多行数据不能直接作为 `COUNT()` 的参数。通过在子查询内部进行聚合统计（如使用 `COUNT()`），可以避免这个问题。