### 作为标量操作数的子查询

以最简单的形式，子查询是返回单个值的标量子查询。标量子查询是一个简单的操作数，几乎在任何一个列值或文字是合法的地方都可以使用，你可以期望它具有所有操作数具有的特性：数据类型，长度，指示是NULL，等等。例如：

CREATE TABLE t1 (s1 INT, s2 CHAR(5) NOT NULL);

INSERT INTO t1 VALUES(100, 'abcde');

SELECT (SELECT s2 FROM t1);

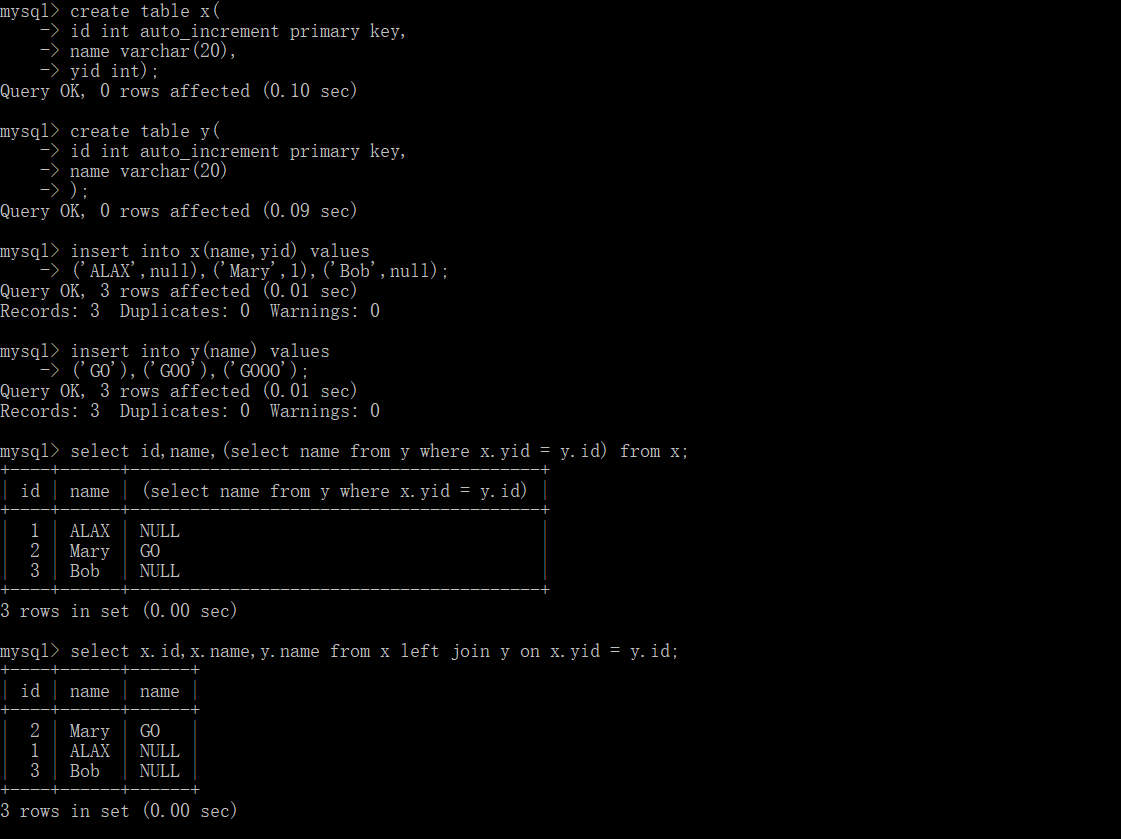
标量子查询选择的值的可空性不被复制，因为如果子查询结果为空，则结果为NULL。对于刚刚显示的子查询，如果t1是空的，结果将是NULL即使 s2是NOT NULL。

**有几个上下文中不能使用标量子查询。如果语句只允许一个字面值，则不能使用子查询。例如，LIMIT需要文字整数参数，并且**[**LOAD DATA INFILE**](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)**需要文字串文件名。您不能使用子查询来提供这些值。**

标量子查询可以是表达式的一部分，但记住括号，即使子查询是为函数提供参数的操作数。例如：

SELECT UPPER((SELECT s1 FROM t1)) FROM t2;

标量子查询与Left Join查询效果相同：



### 使用子查询进行比较

比较时使用的运算符 = > < >= <= <> != <=>

也可以使用like（子查询语句）作比较进行查询

子查询的唯一合法位置在比较的右侧

SELECT \* FROM t1

WHERE column1 = (SELECT MAX(column2) FROM t2);

1

查找t1包含在给定列中出现两次的值的表中的所有行：

SELECT \* FROM t1 AS t

WHERE 2 = (SELECT COUNT(\*) FROM t1 WHERE t1.id = t.id);

为了比较子查询和标量，子查询必须返回一个标量。为了比较子查询和行构造函数，子查询必须是一个行子查询，该行子查询返回的行数与行构造函数的值相同。

### 3.使用ANY，IN或SOME进行子查询

*operand* *comparison\_operator* ANY (*subquery*)

*operand* IN (subquery) // NOT *IN*

*operand* *comparison\_operator* SOME (*subquery*)

***comparison\_operator：***= > < >= <= <> !=

ANY关键字，它必须遵循一个比较操作符

SELECT s1 FROM t1 WHERE s1 > ANY (SELECT s1 FROM t2);

子查询必须返回最多一行但返回多行的语句会发生此错误。考虑下面的例子：

SELECT \* FROM t1 WHERE column1 = (SELECT column1 FROM t2);

如果SELECT column1 FROM t2只返回一行，则前面的查询将起作用。如果子查询返回多个行，则会发生错误1242。在这种情况下，查询应该被重写为：

SELECT \* FROM t1 WHERE column1 = ANY (SELECT column1 FROM t2);

### 4.具有EXISTS或NOT EXISTS的子查询

查询查询一个城市有什么样的商店：

SELECT DISTINCT store\_type FROM stores

WHERE EXISTS (SELECT \* FROM cities\_stores

WHERE cities\_stores.store\_type = stores.store\_type);

查询一个城市没有什么样的商店：

SELECT DISTINCT store\_type FROM stores

WHERE NOT EXISTS (SELECT \* FROM cities\_stores

WHERE cities\_stores.store\_type = stores.store\_type);

查询所有的城市都有什么样的商店：

SELECT DISTINCT store\_type FROM stores s1

WHERE NOT EXISTS (

SELECT \* FROM cities WHERE NOT EXISTS (

SELECT \* FROM cities\_stores

WHERE cities\_stores.city = cities.city

AND cities\_stores.store\_type = stores.store\_type));

最内部的Not Exists 查询的是所有的城市中有没有不存在于商店列表中的商店

再通过外部的not exists 来查询所有的城市都有什么商店。

### 5.派生表（FROM子句中的子查询）

SELECT ... FROM (subquery) [AS] tbl\_name ...

该 子句是强制性的，因为子句中的每个表都 必须有一个名称。选择列表中的任何列必须具有唯一的名称。 [AS] ***tbl\_name***FROM***subquery***

这里是另一个例子：假设你想知道一个分组表中的一组总和的平均值。这不起作用：

SELECT AVG(SUM(column1)) FROM t1 GROUP BY column1; //无效报错

但是，这个查询提供了所需的信息：

SELECT AVG(sum\_column1)

FROM (SELECT SUM(column1) AS sum\_column1

FROM t1 GROUP BY column1) AS t1;

请注意，子查询（sum\_column1）中使用的列名在外部查询中被识别。

派生表可以返回标量，列，行或表。

派生表不能是相关的子查询，或者包含外部引用或对其他表的引用

### 6. 优化子查询

一些可以提高子查询效率的示例：

1. 使用影响子查询中行数或顺序的子查询子句利于优化：

SELECT \* FROM t1 WHERE t1.column1 IN (SELECT column1 FROM t2 ORDER BY column1);

SELECT \* FROM t1 WHERE t1.column1 IN(SELECT DISTINCT column1 FROM t2);

SELECT \* FROM t1 WHERE EXISTS (SELECT \* FROM t2 LIMIT 1);

1. 将联接替换为子查询：

SELECT DISTINCT column1 FROM t1 WHERE t1.column1 IN (

SELECT column1 FROM t2);

而不是这个：

SELECT DISTINCT t1.column1 FROM t1, t2

WHERE t1.column1 = t2.column1;

1. 将子句从外部移动到子查询内部：

SELECT \* FROM t1

WHERE s1 IN (SELECT s1 FROM t1 UNION ALL SELECT s1 FROM t2);

而不是这个查询：

SELECT \* FROM t1

WHERE s1 IN (SELECT s1 FROM t1) OR s1 IN (SELECT s1 FROM t2);

1. 使用行子查询而不是相关的子查询。

SELECT \* FROM t1

WHERE (column1,column2) IN (SELECT column1,column2 FROM t2);

而不是这个查询：

SELECT \* FROM t1

WHERE EXISTS (SELECT \* FROM t2 WHERE t2.column1=t1.column1

AND t2.column2=t1.column2);

1. 对于总是返回一行的不相关的子查询， IN总是慢于 =。