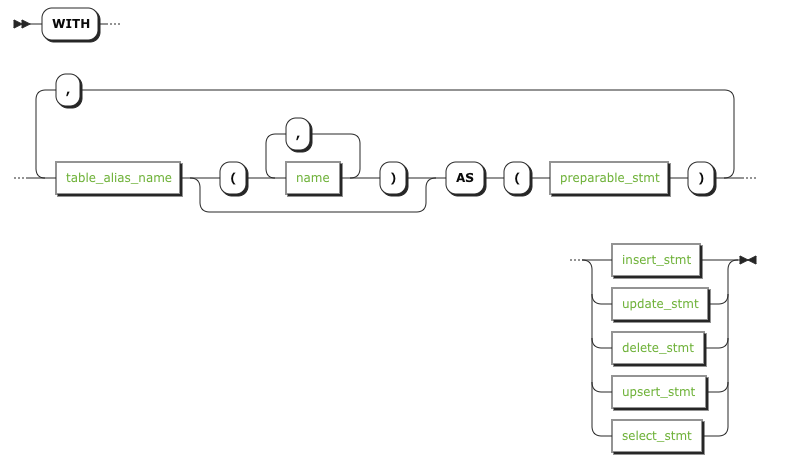
New in v2.0:通用table表达式或CTEs，用于为在较大的查询上下文之前的复杂的子查询提供简写名称。 这提高了SQL代码的可读性。

CTEs可以与SELECT子句和INSERT，DELETE，UPDATE和UPSERT语句结合使用。

概要



参数

| **Parameter** | **Description** |
| --- | --- |
| table\_alias\_name | The name to use to refer to the common table expression from the accompanying query or statement.用于引用附随的查询或语句中的通用table表达式的名称。 |
| name | 新定义的通用table表达式中其中一列的名称。 |
| preparable\_stmt | 要用作通用table表达式的语句或子查询。 |

概览

形式为WITH x AS y IN z的查询或语句为子查询y的结果创建临时表名称x，以便在查询z的上下文中重用。

举个例子:

> WITH o AS (**SELECT** \* **FROM** orders **WHERE** id **IN** (33, 542, 112))

**SELECT** \*

**FROM** customers **AS** c, o

**WHERE** o.customer\_id = c.id;

在此示例中，WITH子句为order上的子查询定义临时名称o，该名称成为有效的表名，以便在后续SELECT子句的任何table表达式中使用。

这个查询相当于以下例子，但阅读起来更简单：

> **SELECT** \*

**FROM** customers **AS** c, (**SELECT** \* **FROM** orders **WHERE** id **IN** (33, 542, 112)) **AS** o

**WHERE** o.customer\_id = c.id;

也可以使用单个WITH子句同时定义多个通用table表达式，用逗号分隔。 后面的子查询可以按名称引用早前的子查询。 例如，以下查询等同于上面的两个示例：

> WITH o AS (**SELECT** \* **FROM** orders **WHERE** id **IN** (33, 542, 112)),

results **AS** (**SELECT** \* **FROM** customers **AS** c, o **WHERE** o.customer\_id = c.id)

**SELECT** \* **FROM** results;

在这个例子中，第二个CTEresults指的是名字的第一个CTEo。 最后的查询是指CTEresults.

嵌套的WITH条款

可以在子查询中使用WITH子句，或者在另一个WITH子句中使用WITH子句， 举个例子:

> WITH a AS (**SELECT** \* **FROM** (**WITH** b **AS** (**SELECT** \* **FROM** c)

**SELECT** \* **FROM** b))

**SELECT** \* **FROM** a;

在分析提及CTEs名称的table表达式时，CockroachDB将选择最接近table表达式的CTE定义。 例如： 举个例子:

> WITH a AS (TABLE x),

b AS (WITH a AS (TABLE y)

**SELECT** \* **FROM** a)

**SELECT** \* **FROM** b;

在这个例子中，内部子查询SELECT \* FROM a将从表y（最接近WITH子句）中选择，而不是从表x中选择。

数据修改语句

可以使用数据修改语句（INSERT，DELETE等）作为通用table表达式。

举个例子:

> WITH v AS (**INSERT** **INTO** t(x) **VALUES** (1), (2), (3) RETURNING x)

**SELECT** x+1 **FROM** v

但是，以下限制适用于：只有SQL语句顶层的WITH子句可以包含数据修改语句。 上面的示例是有效的，但以下例子不是：

> **SELECT** x+1 **FROM**

(**WITH** v **AS** (**INSERT** **INTO** t(x) **VALUES** (1), (2), (3) RETURNING x)

**SELECT** \* **FROM** v);

这是无效的，因为定义INSERT通用table表达式的WITH子句不在查询的顶层。

如果通用table表达式包含数据修改语句（INSERT，DELETE等），即使仅使用部分结果（例如，使用LIMIT），也会完全执行修改。 有关详细信息，请参阅子查询中的[数据写入](http://doc.cockroachchina.baidu.com/" \l "develop/sql-syntax/subqueries/#data-writes-in-subqueries)。

已知限制

在未来的CockroachDB版本中可能会解除以下限制。

多次引用CTE

目前无法通过名称多次引用通用table表达式。 例如，以下查询无效，因为CTEa被引用两次：

**WITH** a **AS** (VALUES (1), (2), (3))

**SELECT** \* **FROM** a, a;

将CTE与数据修改语句一起使用

如果顶层查询未直接或间接引用包含数据修改语句的通用table表达式，则根本不会执行数据修改语句。

例如，以下查询不插入任何行，因为未使用CTEa：

WITH a AS (INSERT INTO t(x) VALUES (1), (2), (3))

SELECT \* FROM b*;*

此外，即使使用CTEa，以下查询也不会插入任何行，因为使用a的其他CTE本身未使用：

WITH a AS (**INSERT** **INTO** t(x) **VALUES** (1), (2), (3)),

b **AS** (**SELECT** \* **FROM** a)

**SELECT** \* **FROM** c;

要确定是否有效进行修改，请使用EXPLAIN并检查所需的数据修改是否是整个查询的最终计划的一部分。

使用带有视图的CTE

在用于定义视图的选择查询中，尚不可能使用通用table表达式。

使用带有VALUES子句的CTE

在VALUES子句内的子查询中，还不可能在VALUES子句外部使用通用table表达式，例如：

**WITH** a **AS** (...) VALUES ((**SELECT** \* **FROM** a));

将CTE与Set Operations一起使用

现在还不能在set运算符的右操作数中使用在set表达式之外定义的通用table表达式，例如：

WITH a AS (**SELECT** 1)

**SELECT** \* **FROM** users **UNION** **SELECT** \* **FROM** a; *-- "a" used on the right, not yet supported.*

对于UNION，你可以通过交换操作数来解决此限制。 对于其他集合运算符，你可以在右侧操作数内部定义CTE。

See also

* [Subqueries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/subqueries/)
* [Selection Queries](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/selection-queries/)
* [Table Expressions](http://doc.cockroachchina.baidu.com/#develop/sql-syntax/table-expressions/)
* [EXPLAIN](http://doc.cockroachchina.baidu.com/#develop/sql-statements/EXPLAIN/)