Foreign Key约束指定列的所有值必须与它引用的列中的现有值完全匹配，从而强制引用完整性。

例如，如果在orders.customer上创建一个引用customers.id的外键：

* 每个插入或更新orders.customer的值，都必须在customers.id中存在
* 在orders.customer中引用到的customers.id 的值都不能被删除或者更新。

但是，customers.id的值可以是orders.customer中不存在的。

如果计划在schema中使用外键，请考虑使用[交错表](https://www.cockroachlabs.com/docs/stable/interleave-in-parent.html)，这可以显着提高查询性能。

细节

创建外键的规则

**外键列**

* 外键列必须使用其引用列的类型。
* 每列不能超过1个外键约束。
* 不能是[计算列](http://doc.cockroachchina.baidu.com/#develop/sql-statements/CONPUTED-COLUMNS/).
* 外键列必须索引。这是因为在table中删除或更新列都将会在引用的表中查询是否有对应的值，来判断该值是否会破坏约束条件。实际上，使用这些表的应用程序可能也需要这样的索引，因为查找属于某个实体的所有记录(例如给定客户的所有订单)都很常见。
  + 要在创建新表时满足此要求，可以使用以下几个选择：
    - 使用CREATE TABLE的INDEX子句显式创建索引。
    - 依赖主键或唯一约束创建的索引。
    - CockroachDB自动为您创建外键列的索引。 但是，请务必注意，如果稍后删除外键约束，此自动创建的索引不会被删除。
    - 使用外键列作为索引列的前缀也满足索引的要求。例如，如果创建外键列（A，B），列（（A，B，C）的索引满足索引的要求。
  + 要在已存在表上添加外键约束，如果列并未建立所以，可以使用CREATE INDEX来索引它们，然后使用ADD CONSTRAINT来为列添加外键约束。

**被引用列**

* 被引用列必须仅包含唯一的值集合。 这意味着REFERENCES子句必须使用与引用表上的唯一或主键约束完全相同的列。例如，子句REFERENCES tbl（C，D）要求tbl具有约束UNIQUE（C，D）或PRIMARY KEY（C，D）。
* 在REFERENCES子句中，如果指定表但没有列，则CockroachDB引用表的主键。 在这些情况下，外键约束和引用表的主键必须包含相同数量的列。

\_NULL\_值

单列外键接受\_NULL\_值。

仅在这些情况下，多列外键接受\_NULL\_值：

* 最终引用的行 - 由语句的其他值确定 - 包含NULL作为引用列的值（即从引用完整性的角度看，NULL是有效的）
* write包含所有外键列的NULL值

例如，如果列（A，B）有一个外键约束并尝试插入（1，NULL），则写入将失败，除非存在（A）的值为1和包含（B）为NULL的行，不过插入（NULL，NULL）则是可以的。

然而，在外键或引用列中允许NULL值会降低其引用完整性。为了避免这种情况，你可以在创建表时对两组列都使用Not Null约束。 （Not Null约束无法添加到现有表中。）

外键操作（New in v2.0）

在设置外键约束时，你可以控制当被引用列（外键）被删除或更新时会发生什么。

| **Parameter** | **Description** |
| --- | --- |
| ON DELETE NO ACTION | 默认action。如果任何key的引用正在被删除，该事务将在声明结束时失败。key可以被更新，取决于ON UPDATE操作  Alias: ON DELETE RESTRICT |
| ON UPDATE NO ACTION | 默认操作。如果任何key的引用正在被更新，该事务将在声明结束时失败。key可以被删除，取决于ON DELETE操作  Alias: ON UPDATE RESTRICT |
| ON DELETE RESTRICT / ON UPDATE RESTRICT | RESTRICT和NO ACTION当前是等效的，直到添加了deferring约束检查的选项。 要将现有外键操作设置为“RESTRICT”，必须删除并重新创建外键约束。 |
| ON DELETE CASCADE / ON UPDATE CASCADE | 删除或更新引用的外键时，将分别删除或更新引用该键的所有行。如果对行进行了其他更改，例如“SET NULL”或“SET DEFAULT”，则将优先删除。  请注意，CASCADE不会列出它删除或更新的对象，因此应谨慎使用。 |
| ON DELETE SET NULL / ON UPDATE SET NULL | 当分别删除或更新引用的外键时，引用该键的所有行的列将设置为“NULL”。 该列必须允许“NULL”，否则此更新将失败。 |
| ON DELETE SET DEFAULT / ON UPDATE SET DEFAULT | 分别删除或更新引用的外键时，引用该键的所有行的列将设置为该列的默认值。 如果列的默认值为null，则与ON DELETE SET NULL或ON UPDATE SET NULL具有相同的效果。 默认值仍必须符合所有其他约束，例如UNIQUE。 |

性能

由于外键约束需要对两个表的每一行进行检查，所以涉及外键列的语句执行会花费更长时间。在涉及到外键的批量插入操作中，你将能察觉到这点。

We're currently working to improve the performance of these statements, though. 不过，我们目前正努力改善这些语句的性能。

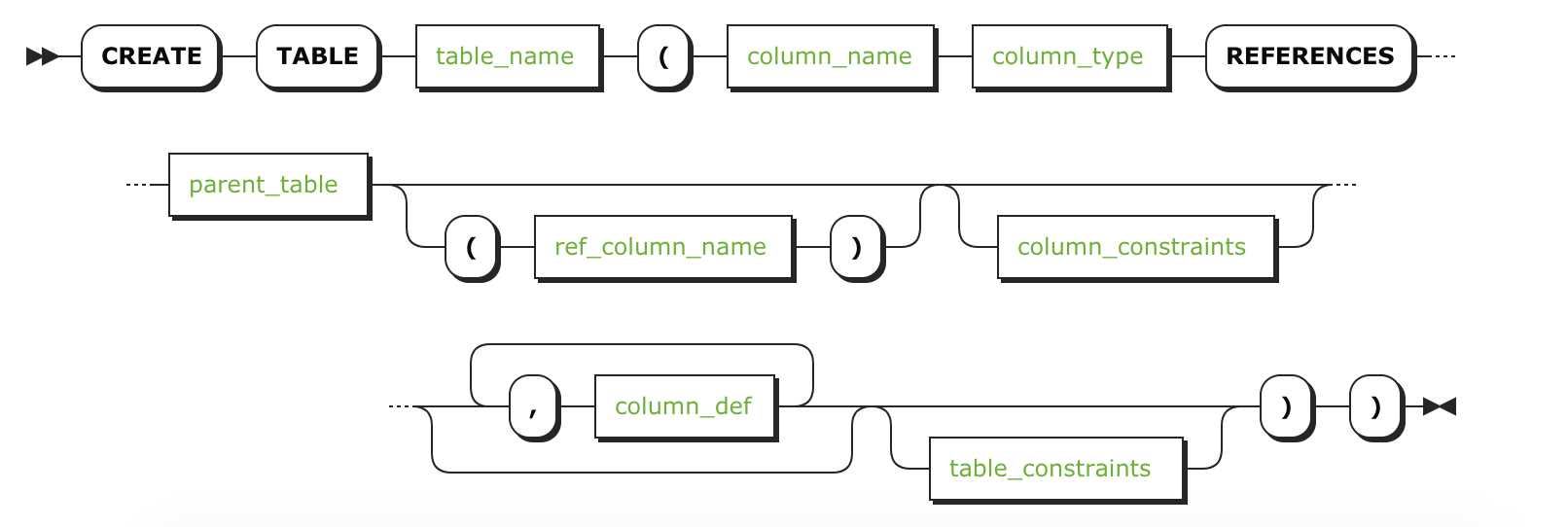
您还可以通过使用INTERLEAVE IN PARENT来提高使用外键的某些语句的性能。

语法

外键约束可以在表级定义。 但是，如果您只希望将约束应用于单个列，则可以在列级别应用该约束。

你还可以通过ADD CONSTRAINT将外键约束添加到现有表。

Column Level



| **Parameter** | **Description** |
| --- | --- |
| table\_name | 表名 |
| column\_name | 外键列名 |
| column\_type | 外键列数据类型 |
| parent\_table | 外键列引用的表名 |
| ref\_column\_name | 外键列引用的列名  如果你不想在parent\_table中引用ref\_column\_name，那么CockroachDB会使用parent\_table主键的第一列。 |
| column\_constraints | 要应用于此列的任何其他列级约束。 |
| column\_def | 表中任何其他列的定义。 |
| table\_constraints | 要应用的任何表级约束。 |

**举例**

> **CREATE** **TABLE** **IF** **NOT** **EXISTS** orders (

id INT **PRIMARY** **KEY**,

customer INT **NOT** NULL **REFERENCES** customers (id) **ON** **DELETE** **CASCADE**,

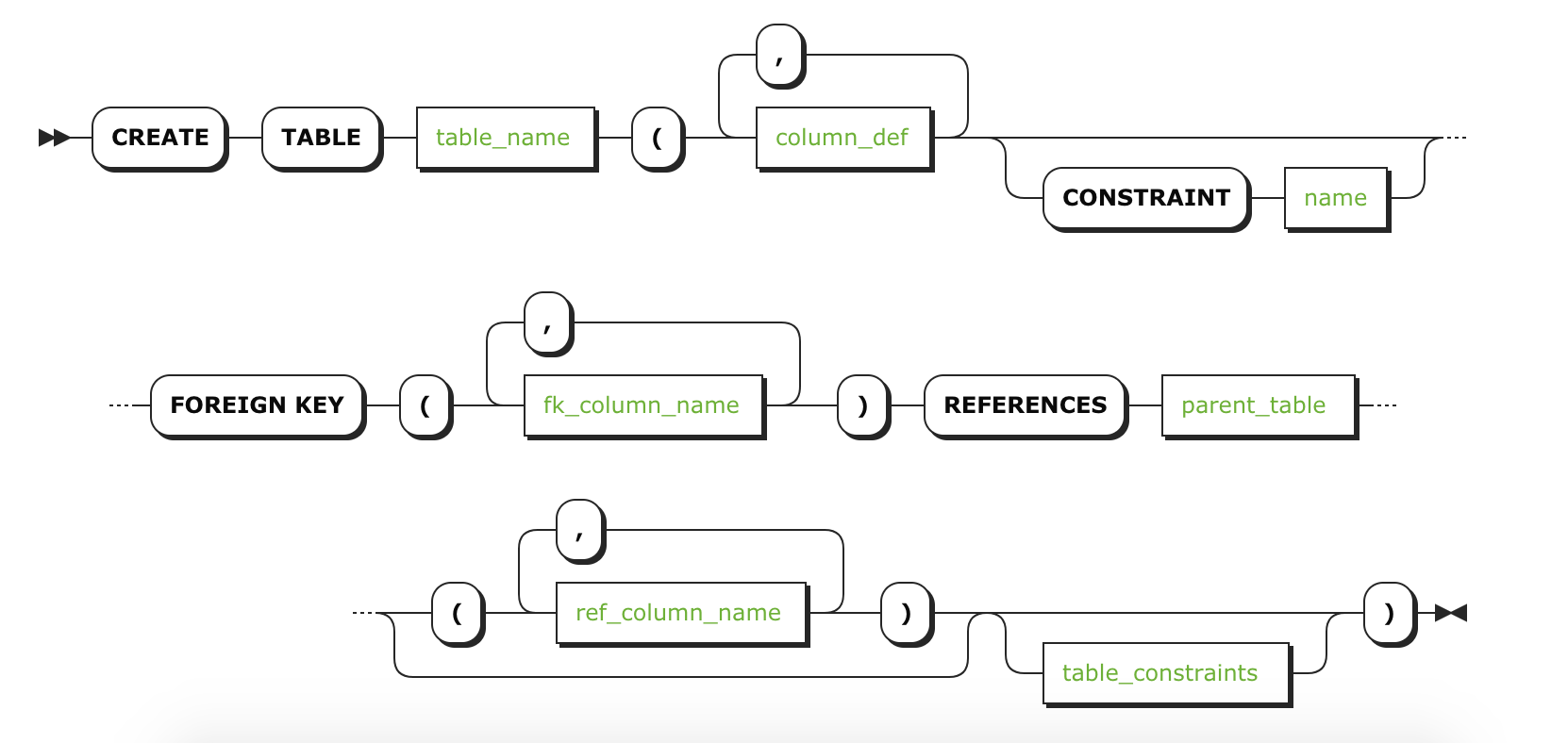
orderTotal DECIMAL(9,2),

**INDEX** (customer)

);

警告：CASCADE不会列出它删除或更新的对象，因此应谨慎使用。

表级别



| **Parameter** | **Description** |
| --- | --- |
| table\_name | 表名 |
| column\_def | 表中任何其他列的定义。 |
| name | 约束名 |
| fk\_column\_name | 外键约束列的名称 |
| parent\_table | 外键约束列引用的表的名称 |
| ref\_column\_name | 外键列引用的列名  如果你不想在parent\_table中引用ref\_column\_name，那么CockroachDB会使用parent\_table主键的第一列。 |
| table\_constraints | 要应用的任何表级约束。 |

**举例**

**CREATE** **TABLE** packages (

customer INT,

"order" INT,

id INT,

address **STRING**(50),

delivered BOOL,

delivery\_date DATE,

**PRIMARY** **KEY** (customer, "order", id),

**CONSTRAINT** fk\_order **FOREIGN** **KEY** (customer, "order") **REFERENCES** orders

) INTERLEAVE **IN** PARENT orders (customer, "order")

;

用法示例

将外键约束与默认操作一起使用

在这个例子中，我们将使用外键约束与默认action来创建表 (ON UPDATE NO ACTION ON DELETE NO ACTION).

首先，创建一个被引用的表customers：

> **CREATE** **TABLE** customers (id INT **PRIMARY** **KEY**, email **STRING** **UNIQUE**);

然后创建表orders：

> **CREATE** **TABLE** **IF** **NOT** **EXISTS** orders (

id INT **PRIMARY** **KEY**,

customer INT **NOT** NULL **REFERENCES** customers (id),

orderTotal DECIMAL(9,2),

**INDEX** (customer)

);

在每个表中插入记录：

> **INSERT** **INTO** customers **VALUES** (1001, 'a@co.tld'), (1234, 'info@cockroachlabs.com');

> **INSERT** **INTO** orders **VALUES** (1, 1002, 29.99);

pq: foreign key violation: value [1002] not found in customers@primary [id]

第二个记录插入返回错误，因为引用的表中不存在客户“1002”。

让我们在引用表中插入一条记录，并尝试更新被引用的表：

> **INSERT** **INTO** orders **VALUES** (1, 1001, 29.99);

> **UPDATE** customers **SET** id = 1002 **WHERE** id = 1001;

pq: foreign key violation: **value**(s) [1001] **in** columns [id] referenced **in** table "orders"

被引用表的更新返回错误，因为 id = 1001被引用了，并且打开了默认外键action (ON UPDATE NO ACTION)。不过，id = 1234可以被更新因为没有被引用。

> **UPDATE** customers **SET** id = 1111 **WHERE** id = 1234;

> **SELECT** \* **FROM** customers;

+------+------------------------+

| id | email |

+------+------------------------+

| 1001 | a@co.tld |

| 1111 | info@cockroachlabs.com |

+------+------------------------+

现在让我们尝试下删除一个引用的值：

> **DELETE** **FROM** customers **WHERE** id = 1001;

pq: foreign key violation: **value**(s) [1001] **in** columns [id] referenced **in** table "orders"

Similarly, the deletion returns an error because id = 1001 is referenced and the default [foreign key action](http://doc.cockroachchina.baidu.com/#develop/constraints/foreign-key/#foreign-key-actions-new-in-v2-0) is enabled (ON DELETE NO ACTION). However, id = 1111 is not referenced and can be deleted: 类似的，该删除操作返回错误，因为 id = 1001被引用了，并且打开了默认外键action (ON DELETE NO ACTION)。然而id = 1111可以被删除因为没有被引用。

> **DELETE** **FROM** customers **WHERE** id = 1111;

> **SELECT** \* **FROM** customers;

+------+----------+

| id | email |

+------+----------+

| 1001 | a@co.tld |

+------+----------+

使用带有CASCADE的外键约束（New in v2.0）

在这个例子中，我们将使用带有ON UPDATE CASCADE and ON DELETE CASCADE的外键约束来创建一个表。 首先，创建一个被引用的表customers\_2：

> **CREATE** **TABLE** customers\_2 (

id INT **PRIMARY** **KEY**

);

然后，创建表orders\_2，使用外键约束：

> **CREATE** **TABLE** orders\_2 (

id INT **PRIMARY** **KEY**,

customer\_id INT **REFERENCES** customers\_2(id) **ON** **UPDATE** **CASCADE** **ON** **DELETE** **CASCADE**

);

插入一些记录到customers\_2：

> **INSERT** **INTO** customers\_2 **VALUES** (1), (2), (3);

插入一些记录到orders\_2：

> **INSERT** **INTO** orders\_2 **VALUES** (100,1), (101,2), (102,3), (103,1);

现在，尝试在被引用的表更新id：

> **UPDATE** customers\_2 **SET** id = 23 **WHERE** id = 1;

> **SELECT** \* **FROM** customers\_2;

+----+

| id |

+----+

| 2 |

| 3 |

| 23 |

+----+

> **SELECT** \* **FROM** orders\_2;

+-----+--------------+

| id | customers\_id |

+-----+--------------+

| 100 | 23 |

| 101 | 2 |

| 102 | 3 |

| 103 | 23 |

+-----+--------------+

当customers\_2中的id = 1更新为id = 23时，更新传播到引用表orders\_2。

同样，删除将级联。 让我们从customers\_2中删除id = 23：

> **DELETE** **FROM** customers\_2 **WHERE** id = 23;

> **SELECT** \* **FROM** customers\_2;

+----+

| id |

+----+

| 2 |

| 3 |

+----+

让我们看看，在 orders\_2中 customers\_id = 23 的行都被删掉了：

> **SELECT** \* **FROM** orders\_2;

+-----+--------------+

| id | customers\_id |

+-----+--------------+

| 101 | 2 |

| 102 | 3 |

+-----+--------------+

使用带有SET NULL的外键约束（New in v2.0）

在这个例子中，我们将使用带有ON UPDATE SET NULL 和ON DELETE SET NULL的外键约束来创建表。

首先，创建被引用的表customers\_3：

> **CREATE** **TABLE** customers\_3 (

id INT **PRIMARY** **KEY**

);

然后，创建使用外键约束的表orders\_3：

> **CREATE** **TABLE** orders\_3 (

id INT **PRIMARY** **KEY**,

customer\_id INT **REFERENCES** customers\_3(id) **ON** **UPDATE** **SET** NULL **ON** **DELETE** **SET** NULL

);

在被被引用表插入一些记录：

> **INSERT** **INTO** customers\_3 **VALUES** (1), (2), (3);

在orders\_3插入一些记录：

> **INSERT** **INTO** orders\_3 **VALUES** (100,1), (101,2), (102,3), (103,1);

> **SELECT** \* **FROM** customers\_3;

+-----+-------------+

| id | customer\_id |

+-----+-------------+

| 100 | 1 |

| 101 | 2 |

| 102 | 3 |

| 103 | 1 |

+-----+-------------+

现在，尝试在被引用表尝试更新id：

> **UPDATE** customers\_3 **SET** id = 23 **WHERE** id = 1;

> **SELECT** \* **FROM** customers\_3;

+----+

| id |

+----+

| 2 |

| 3 |

| 23 |

+----+

> **SELECT** \* **FROM** orders\_3;

+-----+-------------+

| id | customer\_id |

+-----+-------------+

| 100 | NULL |

| 101 | 2 |

| 102 | 3 |

| 103 | NULL |

+-----+-------------+

当customers\_3中的id = 1更新为id = 23时，引用的customer\_id被设置为NULL。

类似地，删除操作会将customer\_id设置为NULL。 让我们从customers\_3中删除id = 2：

> **DELETE** **FROM** customers\_3 **WHERE** id = 2;

> **SELECT** \* **FROM** customers\_3;

+----+

| id |

+----+

| 3 |

| 23 |

+----+

让我们来确认下在orders\_3中customers\_id = 2 被更新为 NULL:

> **SELECT** \* **FROM** orders\_3;

+-----+-------------+

| id | customer\_id |

+-----+-------------+

| 100 | NULL |

| 101 | NULL |

| 102 | 3 |

| 103 | NULL |

+-----+-------------+

使用带有SET DEFAULT的外键约束 （New in v2.0）

在这个例子中，我们将使用带有ON UPDATE SET DEFAULT and ON DELETE SET DEFAULT的外键约束创建表。 首先，创建被引用的表customers\_4：

> **CREATE** **TABLE** customers\_4 (

id INT **PRIMARY** **KEY**

);

然后，创建创建一个customer\_id默认值为9999的表orders\_4：

> **CREATE** **TABLE** orders\_4 (

id INT **PRIMARY** **KEY**,

customer\_id INT **DEFAULT** 9999 **REFERENCES** customers\_4(id) **ON** **UPDATE** **SET** **DEFAULT** **ON** **DELETE** **SET** **DEFAULT**

);

在被引用表插入一些记录：

> **INSERT** **INTO** customers\_4 **VALUES** (1), (2), (3), (9999);

在orders\_4插入一些记录：

> **INSERT** **INTO** orders\_4 **VALUES** (100,1), (101,2), (102,3), (103,1);

+-----+-------------+

| id | customer\_id |

+-----+-------------+

| 100 | 1 |

| 101 | 2 |

| 102 | 3 |

| 103 | 1 |

+-----+-------------+

现在来更新被引用表的id：

> **UPDATE** customers\_4 **SET** id = 23 **WHERE** id = 1;

> **SELECT** \* **FROM** customers\_4;

+------+

| id |

+------+

| 2 |

| 3 |

| 23 |

| 9999 |

+------+

> **SELECT** \* **FROM** orders\_4;

+-----+-------------+

| id | customer\_id |

+-----+-------------+

| 100 | 9999 |

| 101 | 2 |

| 102 | 3 |

| 103 | 9999 |

+-----+-------------+

当customer\_4中的id = 1更新为id = 23时，引用的customer\_id被设置为DEFAULT（即9999）。你可以在orders\_4的第一行和最后一行看到这个，其中id = 100而customer\_id现在是9999。

类似的，删除操作将会把引用表的customer\_id设为默认值，让我们把customers\_4中id = 2 的记录删掉：

> **DELETE** **FROM** customers\_4 **WHERE** id = 2;

> **SELECT** \* **FROM** customers\_4;

+------+

| id |

+------+

| 3 |

| 23 |

| 9999 |

+------+

让我们确认下相应的customer\_id值为id = 101，更新为orders\_4中的DEFAULT值（即9999）：

> **SELECT** \* **FROM** orders\_4;

+-----+-------------+

| id | customer\_id |

+-----+-------------+

| 100 | 9999 |

| 101 | 9999 |

| 102 | 3 |

| 103 | 9999 |

+-----+-------------+

See Also

* [Constraints](http://doc.cockroachchina.baidu.com/#develop/constraints/overview/)
* [DROP CONSTRAINT](http://doc.cockroachchina.baidu.com/#develop/sql-statements/DROP-CONSTRAINT/)
* [ADD CONSTRAINT](http://doc.cockroachchina.baidu.com/#develop/sql-statements/ADD-CONSTRAINT/)
* [Check constraint](http://doc.cockroachchina.baidu.com/#develop/constraints/check/)
* [Default Value constraint](http://doc.cockroachchina.baidu.com/#develop/constraints/default-value/)
* [Not Null constraint](http://doc.cockroachchina.baidu.com/#develop/constraints/not-null/)
* [Primary Key constraint](http://doc.cockroachchina.baidu.com/#develop/constraints/primary-key/)
* [Unique constraint](http://doc.cockroachchina.baidu.com/#develop/constraints/unique/)
* [SHOW CONSTRAINTS](http://doc.cockroachchina.baidu.com/#develop/sql-statements/SHOW-CONSTRAINTS/)