交错表通过优化关联密切的表的键值结构来提高查询性能，如果有可能一起读写数据，则尝试将数据保持在相同的键值range内。

交错表不会影响它们在SQL中的行为。

交错表是如何工作的？

当表被交错时，写入一个表（称为**child**）的数据被直接插入到键值存储中的另一个表（称为**父**）中。 这是通过将子表的主键与父表的主键匹配来实现的。

交错前缀

对于具有可匹配主键的交错表，子表必须使用父表的主键作为其自己的主键的前缀 - 这些匹配的列称为**交错前缀**。 最简单的方法是将这些列视为表示相同的数据，通常使用外键实现。

要强制执行每个表的交错前缀列之间的关系，我们建议使用外键约束。

例如，如果要将orders表与customers表进行交错，并且customers的主键是id，则需要创建一个表示customers.id的列作为orders主键中的第一列，例如，使用名为customer的列。 因此，customers.id列包含的数据是交错前缀，它在orders表中作为customer列存在。

Key-Value 结构

将数据写入子表时，在父表的键匹配交错前缀之后，立即将其插入键值存储中。

例如，如果将orders与customers交错，则orders数据直接写在kv存储中的customers表中。 以下是在此结构中key的简单示例：

/customers/1

/customers/1/orders/1000

/customers/1/orders/1002

/customers/2

/customers/2/orders/1001

/customers/2/orders/1003

...

/customers/n/

/customers/n/orders/<order belonging to n>

通过以这种方式写入数据，相关数据更可能保持在相同的键值range内，这可以使读取和写入更快。 使用上面的示例，所有客户1的数据将被写入相同的range，包括它在customers和orders表中的表示。

什么时候使用交错表

在以下情况下，使用交错表中可能会更好：

* 你的表格形成一个层次结构
* 查询最大化了交错的好处
* 查询不会因交错的权衡而受到太大影响

交错层次结构

当表格形成层次结构时，交错表通常最有效。 例如，你可以将表order（作为子项）交错到表customers（作为父级，表示下达订单的人员）。 你可以通过将表invoices（作为子项）和packages（作为子项）交叉到order（作为父项）来作为示例来试验。

这些关系的整个集合被称为交织层次结构，其包含通过交织前缀相关的所有表格。

好处

通常，通过交错前缀相关的值的读取，写入和连接的速度要快得多。 不过，你还可以使用以下任何一项来提高性能：

* 在交错前缀中过滤更多列（从左到右）。

例如，如果packages的交错前缀是（customer，order），对customer的过滤会很快，但对customer 和order的过滤会更快。

* 仅使用交错层次结构中的表。

权衡

* 通常，在交错表中的表值范围（例如，WHERE column> value）上的读取和删除较慢。

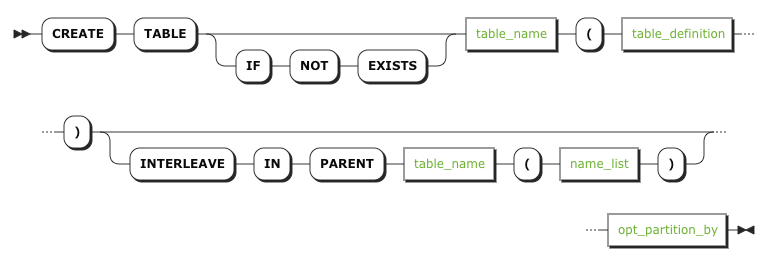
但是，一个例外是对交错层次结构中最大后代的表值范围执行操作，该值对具有常量值的交错前缀的所有列进行过滤。

例如，如果packages的interleave前缀是（customer，order），则在计算另一列上的表值范围时，使用常量值对整个interleave前缀进行过滤，例如WHERE customer = 1 AND order = 1001 AND delivery\_date> DATE'2016-01-25'，仍然很快。

* 如果为根表的任何主键值存储的交错数据量大于键值range的最大大小（默认为64MB），则交错优化将被限制。

例如，如果一个客户有200MB的订单数据，他们的数据很可能分散在多个键值range内，而且尽管交叉存在，但CockroachDB无法快速访问它。

语法



参数

| **Parameter** | **Description** |
| --- | --- |
| CREATE TABLE ... | 使用语法参考[CREATE TABLE](http://doc.cockroachchina.baidu.com/#develop/sql-statements/CREATE-TABLE/). |
| INTERLEAVE IN PARENT table\_name | 要将新子表交织到的父表的名称。 |
| name\_list | 子表的主键中以逗号分隔的列列表，表示父表的主键（即交错前缀）。 |

要求

* 只能在创建子表时交错表。
* 每个子表的主键必须包含其父表的主键作为前缀（称为**交错前缀**）。

例如，如果父表的主键是（（INT，b STRING），则子表的主键可以是（（INT，b STRING，c DECIMAL）。

仅通过确保列使用相同的数据类型来强制执行此要求。 但是，我们建议使用外键约束确保列引用相同的值。

* 交错表不能多于1个父表。 但是，每个父表可以有许多子表。 子表也可以是交错表的父表。

建议

* 当架构形成层次结构时使用交叉表，并且根表的主键（例如，“user ID”或“account ID”）是大多数查询的参数。
* 要强制父表和子表的主键之间的关系，请在子表上使用外键约束。
* 如果不确定交错表是否会提高查询的性能，请测试表在交错时和不交错时的执行表现。

示例

交错表

这个例子在customers，orders和packages之间创建了一个交错的层次结构，以及相应的外键约束。 您可以看到每个子表使用其父表的主键作为其自己的主键的前缀（**交错前缀**）。

> **CREATE** **TABLE** customers (

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

name **STRING**(50)

);

> **CREATE** **TABLE** orders (

customer INT,

id INT,

total DECIMAL(20, 5),

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

**CONSTRAINT** fk\_customer **FOREIGN** **KEY** (customer) **REFERENCES** customers

) INTERLEAVE **IN** PARENT customers (customer)

;

> **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")

;

Key-Value 存储示例

通过查看键值存储中的内容，可以更容易地理解交错表的作用。 例如，使用上面在customers中交错orders的例子，我们可以插入以下值：

> **INSERT** **INTO** customers

(id, name) **VALUES**

(1, 'Ha-Yun'),

(2, 'Emanuela');

> **INSERT** **INTO** orders

(customer, id, total) **VALUES**

(1, 1000, 100.00),

(2, 1001, 90.00),

(1, 1002, 80.00),

(2, 1003, 70.00);

使用键值存储的说明性格式（键以颜色表示;值由- > value表示），数据将如下所示：

/customers/<customers.id = 1> -> 'Ha-Yun'

/customers/<orders.customer = 1>/orders/<orders.id = 1000> -> 100.00

/customers/<orders.customer = 1>/orders/<orders.id = 1002> -> 80.00

/customers/<customers.id = 2> -> 'Emanuela'

/customers/<orders.customer = 2>/orders/<orders.id = 1001> -> 90.00

/customers/<orders.customer = 2>/orders/<orders.id = 1003> -> 70.00

你会注意到customers.id和orders.customer被写入键值存储中的相同位置。 这就是CockroachDB如何将两个表的数据与交错结构联系起来。 通过这种方式存储数据，访问order数据要快得多。

如果我们没有在customers.id和orders.customer之间设置外键约束，并插入orders.customer = 3，那么数据仍然会写入customers表标识符旁边的预期位置的键值，但是SELECT \* FROM customers WHERE id = 3不会返回任何值。

要更好地了解CockroachDB如何写入键值数据，请参阅我们的博客文章[Mapping Table Data to Key-Value Storage](https://www.cockroachlabs.com/blog/sql-in-cockroachdb-mapping-table-data-to-key-value-storage/).

See Also

* [CREATE TABLE](http://doc.cockroachchina.baidu.com/#develop/sql-statements/CREATE-TABLE/)
* [Foreign Keys](http://doc.cockroachchina.baidu.com/#develop/constraints/foreign-key/)
* [Column Families](http://doc.cockroachchina.baidu.com/#develop/performance-optimization/column-families/)