New in v2.0: CREATE SEQUENCE语句在数据库中创建新序列。 使用序列在表中自动递增整数。

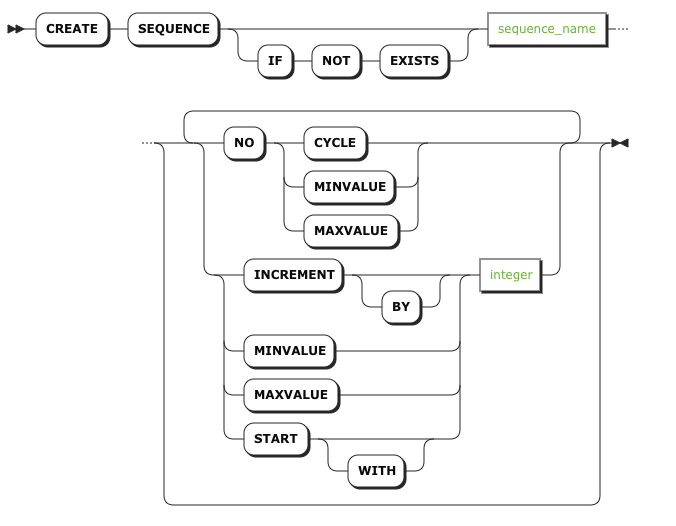
注意事项

* 使用序列比使用UUID，BYTES或SERIAL数据类型自动生成唯一ID要慢。 增加序列需要写入持久存储，而自动生成唯一ID则不需要。 因此，若非必须，否则请使用自动生成的唯一ID。
* 如果事务使序列递增后回滚，则使用序列的列可能在序列值中出现缺口。 序列更新会立即提交，并且不会与其包含的事务一起回滚。 这样做是为了避免阻塞使用相同序列的并发事务。

所需权限

用户必须具有父数据库的CREATE特权。

概要



参数

| **Parameter** | **Description** |
| --- | --- |
| seq\_name | 要创建的序列的名称，该序列在其数据库中必须是唯一的，并遵循标识符规则。 如果未设置默认父数据库，则必须将名称格式设置为database.seq\_name。 |
| INCREMENT | 序列递增的值。 负数会创建降序。 正数表示升序。  **默认:** 1 |
| MINVALUE | 序列的最小值。 如果未指定默认值或输入NO MINVALUE，则应用默认值。  **升序默认值：**1  **降序默认值：**MININT |
| MAXVALUE | 序列的最大值。 如果未指定默认值或输入NO MAXVALUE，则应用默认值。  **升序默认值：**MAXINT  **降序默认值：**-1 |
| START | 序列的第一个值。  **升序的默认值：**1  **降序的默认值：**-1 |
| NO CYCLE | 目前所有序列都设置为NO CYCLE，序列不会回环重用。 |

序列函数

我们支持以下SQL序列函数：

* nextval('seq\_name')

如果nextval()与RETURNING NOTHING语句一起使用，则序列增量可以被重新排序。 更多信息，请参阅[并行语句执行](https://www.cockroachlabs.com/docs/stable/parallel-statement-execution.html)。

* currval('seq\_name')
* lastval()
* setval('seq\_name', value, is\_called)

示例

列出所有序列

> **SELECT** \* **FROM** information\_schema.sequences;

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

| sequence\_catalog | sequence\_schema | sequence\_name | data\_type | numeric\_precision | numeric\_precision\_radix | numeric\_scale | start\_value | minimum\_value | maximum\_value | increment | cycle\_option |

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

| def | db\_2 | test\_4 | INT | 64 | 2 | 0 | 1 | 1 | 9223372036854775807 | 1 | NO |

| def | test\_db | customer\_seq | INT | 64 | 2 | 0 | 101 | 1 | 9223372036854775807 | 2 | NO |

| def | test\_db | desc\_customer\_list | INT | 64 | 2 | 0 | 1000 | -9223372036854775808 | -1 | -2 | NO |

| def | test\_db | test\_sequence3 | INT | 64 | 2 | 0 | 1 | 1 | 9223372036854775807 | 1 | NO |

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

(4 rows)

使用默认设置创建序列

在此示例中，我们使用默认设置创建序列：

> **CREATE** SEQUENCE customer\_seq;

> **SHOW** **CREATE** SEQUENCE customer\_seq;

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

| Sequence | CreateSequence |

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

| customer\_seq | CREATE SEQUENCE customer\_seq MINVALUE 1 MAXVALUE 9223372036854775807 INCREMENT 1 START 1 |

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

使用用户定义的设置创建序列

在这个例子中，我们创建一个从-1开始并以2为增量减少的序列。

> **CREATE** SEQUENCE desc\_customer\_list **START** -1 INCREMENT -2;

> **SHOW** **CREATE** SEQUENCE desc\_customer\_list;

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

| Sequence | CreateSequence |

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

| desc\_customer\_list | CREATE SEQUENCE desc\_customer\_list MINVALUE -9223372036854775808 MAXVALUE -1 INCREMENT -2 START -1 |

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

使用序列创建表

在这个例子中，使用我们在第一个例子中创建的序列作为主键创建一个表：

> **CREATE** **TABLE** customer\_list (

id INT **PRIMARY** **KEY** **DEFAULT** nextval('customer\_seq'),

customer **string**,

address **string**

);

插入几条记录，查看序列：

> **INSERT** **INTO** customer\_list (customer, address)

**VALUES**

('Lauren', '123 Main Street'),

('Jesse', '456 Broad Ave'),

('Amruta', '9876 Green Parkway');

> **SELECT** \* **FROM** customer\_list;

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

| id | customer | address |

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

| 1 | Lauren | 123 Main Street |

| 2 | Jesse | 456 Broad Ave |

| 3 | Amruta | 9876 Green Parkway |

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

查看序列的当前值

要在不递增序列的情况下查看当前值，请使用：

> **SELECT** \* **FROM** customer\_seq;

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

| last\_value | log\_cnt | is\_called |

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

| 3 | 0 | true |

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

返回log\_cnt和is\_called列仅是为了PostgreSQL兼容性; 它们不存储在数据库中。

如果已经从当前会话中的序列获取了一个值，你还可以使用currval('seq\_name')函数来获取最近获得的值：

> **SELECT** currval('customer\_seq');

+---------+

| currval |

+---------+

| 3 |

+---------+

See Also

* [ALTER SEQUENCE](http://doc.cockroachchina.baidu.com/#develop/sql-statements/ALTER-SEQUENCE/)
* [RENAME SEQUENCE](http://doc.cockroachchina.baidu.com/#develop/sql-statements/RENAME-SEQUENCE/)
* [DROP SEQUENCE](http://doc.cockroachchina.baidu.com/#develop/sql-statements/DROP-SEQUENCE/)
* [Functions and Operators](http://doc.cockroachchina.baidu.com/#develop/functions-and-operators/)
* [Other SQL Statements](http://doc.cockroachchina.baidu.com/#develop/sql-statements/overview/)