SERIAL是一种列数据类型，在插入时，从执行插入的节点的时间戳和ID生成64位整数。除极端情况外，此组合可能是全局唯一的（有关详细信息，请参阅此[示例](https://www.cockroachlabs.com/docs/stable/create-table.html#create-a-table-with-auto-generated-unique-row-ids)）此外，由于值生成不需要与其他节点通信，因此它比需要分布式协作的顺序自动递增值要快得多。

在大多数情况下，我们建议使用带有gen\_random\_uuid()函数的UUID数据类型作为默认值，该值生成128位值（远大于SERIAL的64位）并将它们分散到表的所有基础键值范围内 ，确保多个节点共担负载。 有关更多详细信息，请参阅[Create a table with auto-generated unique row IDs](https://www.cockroachlabs.com/docs/stable/uuid.html#create-a-table-with-auto-generated-unique-row-ids)。

别名

SERIAL 类型等效于 INT DEFAULT unique\_rowid()。

在CockroachDB中，以下是 SERIAL的别名:

* SERIAL2
* SERIAL4
* SERIAL8
* SMALLSERIAL
* BIGSERIAL

注意：SERIAL2和SERIAL4与SERIAL相同并存储8字节值，而不是名称可能暗示的2字节或4字节值。

语法

任何INT值都是有效的SERIAL值；特别是不变的SERIAL值可以用数值文本（numeric literals）来代替。

大小

和INT一样。

示例

使用SERIAL自动生成主键

在这个例子中，我们创建一个以SERIAL列为主键的表，这样我们就可以在insert上自动生成唯一ID。

> **CREATE** **TABLE** serial (a SERIAL **PRIMARY** **KEY**, b **STRING**, c BOOL);

SHOW COLUMNS语句显示SERIAL类型只是INT的别名，其中unique\_rowid()作为默认值。

> **SHOW** **COLUMNS** **FROM** serial;

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

| Field | Type | Null | Default |

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

| a | INT | false | unique\_rowid() |

| b | STRING | true | NULL |

| c | BOOL | true | NULL |

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

当我们在列a中插入空值的行并显示新行时，我们会看到每行的列a都有默认生成的唯一值。

> **INSERT** **INTO** serial (b,c) **VALUES** ('red', true), ('yellow', false), ('pink', true);

> **INSERT** **INTO** serial (a,b,c) **VALUES** (123, 'white', false);

> **SELECT** \* **FROM** serial;

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

| a | b | c |

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

| 148656994422095873 | red | true |

| 148656994422161409 | yellow | false |

| 148656994422194177 | pink | true |

| 123 | white | false |

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

自动递增并不总是连续的

一种常见的误解是PostgreSQL和MySQL中的自动递增类型生成严格的顺序值。 实际上，即使未提交插入，每个插入也会将序列增加一个。 这意味着自动递增类型可能会在序列中留下空白。

若想亲自体验一下，请在PostgreSQL中运行以下示例：

1.创建一个带有SERIAL列的表。

> **CREATE** **TABLE** increment (a SERIAL **PRIMARY** **KEY**);

```

2. 运行四个事务来插入行。

``` sql

> **BEGIN**; **INSERT** **INTO** increment **DEFAULT** **VALUES**; **ROLLBACK**;

> **BEGIN**; **INSERT** **INTO** increment **DEFAULT** **VALUES**; **COMMIT**;

> **BEGIN**; **INSERT** **INTO** increment **DEFAULT** **VALUES**; **ROLLBACK**;

> **BEGIN**; **INSERT** **INTO** increment **DEFAULT** **VALUES**; **COMMIT**;

1. 查看创建的行

> **SELECT** \* **from** increment;

+---+

| a |

+---+

| 2 |

| 4 |

+---+

由于每次插入将`a`列中的序列增加1，因此第一个提交的插入值为“2”，第二个提交的插入值为“4”。 如你所见，这些值不是严格顺序的，最后一个值不能准确计算表中的行数。

总而言之，PostgreSQL和CockroachDB中的SERIAL类型以及MySQL中的AUTO\_INCREMENT类型都表现相同，因为它们不会创建严格的序列。 CockroachDB可能会比其他数据库创建更多的空白，但会更快地生成这些值。

支持的转换

SERIAL类型的值可以转换为其他类型，如任何INT值。

See Also

[Data Types](http://doc.cockroachchina.baidu.com/#develop/data-types/overview/)