Primary Key and Auto-Increment

by Sophia

WHAT'S COVERED

In this lesson, you will learn about using an auto-incrementing column as a primary key, in two parts. Specifically, this lesson will cover:

1. Sequences

All database systems provide a way of auto-incrementing a primary key field in a table. In PostgreSQL, this is done with SEQUENCE, a special type of object that creates a sequence of integers. The sequence can be incrementing (1, 2, 3) or decrementing (3, 2, 1).

Sequences can have a few standard parameters:

- START: The value that the sequence starts with. The default is to start with 1.
- INCREMENT: The value that should be added to the current sequence value to create a new value.
- MINVALUE: The minimum value that is set to a sequence. The default is 1.
- MAXVALUE: The maximum value that is set to a sequence. The default maximum value is the maximum value of the data type of the sequence. For example, if the data type is set to SMALLINT, the maximum value is 32,767.

The structure of creating a sequence looks like the following:

```
CREATE SEQUENCE <sequence_name>
[parameters];
```

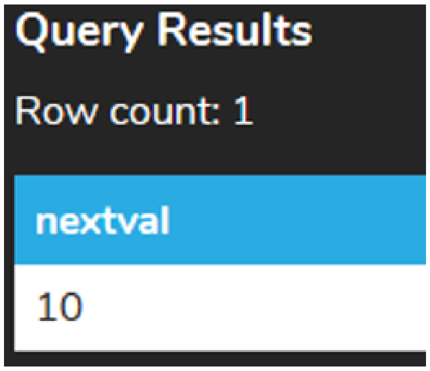
For example, if we wanted to create a sequence named mysequence and have it start with 10, and increment it by 10, we would do the following:

```
CREATE SEQUENCE mysequence
```

```
START 10
INCREMENT 10;
```

If we wanted to get the next value from the sequence, we could use the nextval function like this:

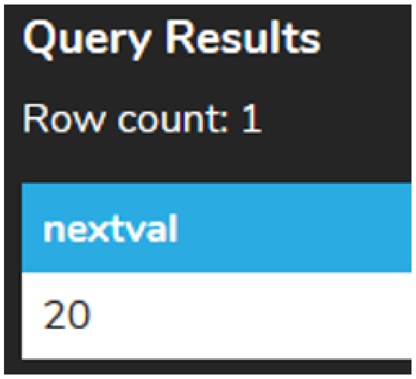
```
SELECT nextval('mysequence');
```


Query Results

Row count: 1

nextval
10

You'll notice that the first value is set to 10. If we rerun the same statement, it'll increment the value by 10:


Query Results

Row count: 1

nextval
20

2. Using the SERIAL Data Type

A sequence can be automatically added to a table using the SERIAL data type. SERIAL is not a standard data type in PostgreSQL; it is a pseudo-type. A **pseudo-type** is a special data type that represents a result set or a value that does not correspond directly to a stored column in a table. Pseudo-types are not associated with any actual storage in the database but are used to represent specific constructs or results in the context of queries or functions.

The SERIAL data type simplifies the complexity of creating and incrementing a sequence. When you assign the SERIAL pseudo-type to a table, the database does the following:

- Creates a sequence object and sets the sequence's next value as the column's default value .
- Adds a NOT NULL constraint to the column since the sequence should always generate an integer that is always not null.
- Assigns the sequence owner to the column in the table, so if the column or table containing it is dropped (removed), the sequence is also removed.

For example, consider the following statement:

```
CREATE TABLE contact(
contact_id SERIAL,
username VARCHAR(50),
password VARCHAR(50) );
```

Behind the scenes, here's a rough equivalent of what's happening:

```
CREATE SEQUENCE contact_contact_id_seq;

CREATE TABLE contact(
contact_id integer NOT NULL DEFAULT nextval(contact_contact_id_seq),
username VARCHAR(50),
password VARCHAR(50)
);

ALTER SEQUENCE contact_contact_id_seq
OWNED BY contact.contact_id;
```

This looks complex and has some extra commands you haven't learned about yet, but it follows the structure as we described above:

- Creates a sequence with a unique specific name behind the scenes.
- Creates the table with the contact_id set as an integer with the NOT NULL constraint. The default value is set to the next value of the sequence.
- Alters the sequence to set the owner to the contact_id column in the contact table.

As you can see, SERIAL makes the process much easier and simpler.
It is important to note that using SERIAL does not create an index on the column or make the column a primary key. Here's how you would create the table if you were doing that:

```
CREATE TABLE contact(
contact_id SERIAL PRIMARY KEY,
username VARCHAR(50),
password VARCHAR(50)
);
```

In a later lesson, we will explore inserting into a table with the SERIAL primary key.

WATCH

TRY IT

Your turn! Open the SQL tool by clicking on the LAUNCH DATABASE button below. Then, enter in one of the examples above and see how it works. Next, try your own choices for which columns you want the query to provide.

TERM TO KNOW

Pseudo-Type
A data type that is used on data that is not directly stored in a table, such as data in a result set.

SUMMARY

In this lesson, you learned how to **use the SERIAL data type** when creating a table to specify that a certain field should automatically generate a unique, sequential integer value for each record. By default, PostgreSQL creates a **sequence** object for each SERIAL column you define and sets it as the column's default value. This will ensure that each new record added to the table has a unique, incrementing value. With the SERIAL data type, unique identifiers, such as primary keys, can be managed and generated more efficiently in tables, enabling auto-incrementing integers to be handled consistently.

TERMS TO KNOW

Pseudo-Type
A data type that is used on data that is not directly stored in a table, such as data in a result set.