Primary Key and Auto-increment

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores using the use of an auto-incrementing column as a primary key in two parts:

1. Sequences
2. Using the Data Type SERIAL

1. Sequences

All databases will have the option of an auto-incrementing column to use as the primary key column in a table. In PostgreSQL, this is done with a SEQUENCE, which is a special type of object that creates a sequence of integers. It is important to know the order of the numbers, as the sequences of 1, 2, 3, 4, 5 and 5, 4, 3, 2, 1 would be different sequences.

Sequences can have a few common 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.

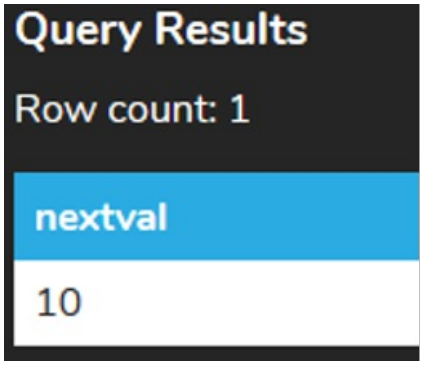
The structure of creating a sequence looks like the following:

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:

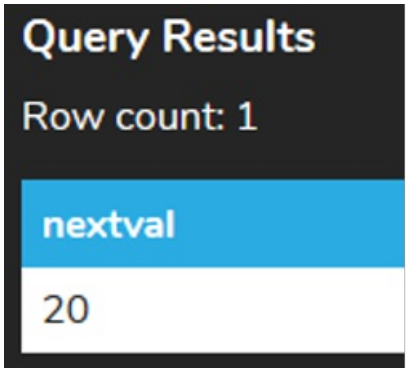
CREATE SEQUENCE mysequence
START 10
INCREMENT 10;
If we wanted to get the next value from the sequence, we can use the nextval function like this:

SELECT nextval('mysequence');

© 2022 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.



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



2. Using the Data Type SERIAL

A sequence can be automatically added to a table using the data type SERIAL. It is an important pseudo-type as it simplifies much of the complexity of creating the sequence and incrementing it. By assigning the SERIAL pseudo-type to a table, the database will do the following:

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

In this following statement:

CREATE TABLE contact(contact_id SERIAL, username VARCHAR(50), password VARCHAR(50));
Behind the scenes, it is the same as the database doing this:

```
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)
);
```

© 2022 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.

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

This looks complex and has some extra commands we haven't reviewed 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.

It is important to note that using SERIAL does not create an index on the column or make the column a primary key. To do so, we have to define the PRIMARY KEY constraint on the serial column:

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

We will explore inserting into a table with the SERIAL primary key in a later tutorial.

Video Transcription

[MUSIC PLAYING] The primary key is quite important being that it uniquely identifies every single role within the table. There are instances where you might not have a value to make it such that it's going to be unique across the board, and you have to generate it. Therefore, you can use sequences or in Postgres you can use Acero, or it's also called an auto increment, or auto number depending on the database that you're utilizing.

In this case here, go ahead and create this table. And now if we insert it into this table, we're only going to insert into the username itself with the values of fun, coding, and SQL into the username. If we go ahead and run this statement, don't worry we'll get into the Insert statements later on, but if we go ahead now and select from the table.

[MUSIC PLAYING]

You should see that even though we didn't insert anything into the contact ID, it automatically sequences it with one, two, or three sequences work in the same way the next time that we insert into the table it'll be 4, and so forth.

[MUSIC PLAYING]

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.

© 2022 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.

SUMMARY

We can use an auto-increment column called a SERIAL in PostgreSQL that acts as a sequence for primary keys.

Source: Authored by Vincent Tran

© 2022 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.