ALTER TABLE to Change Columns: Data Type

by Sophia

WHAT'S COVERED

In this lesson, you will explore using the ALTER TABLE statement to change the data type of a column in a table. Specifically, this lesson will cover:

1. Changing Columns

Although it is rare to do so once data has been inserted into a column, there are times when we may want to change the data type of a column. When we change the data type, PostgreSQL will change the values to the new data type, which is called **casting**. If casting the data type to the new one fails, the database will issue an error.

KEY CONCEPT

Not all data types can be changed. It is easy to change from INT to VARCHAR but impossible to change from VARCHAR to INT. VARCHAR allows for numbers and letters, while INT is just numbers. It also matters if it is BIG INT or TINY INT; you can't go from BIG to TINY, but you can go from TINY to BIG.

BIG IDEA

Just because the database system permits a certain data type conversion does not mean it is necessarily a good idea. For example, some allowed data type conversions have the effect of truncating or modifying the data. For example, if you were to change the unit_price column in the invoice_line table in our sample database to INT, the 0.99 prices would round up to a value of 1.

Another example of a change that is allowed but might cause problems is changing numeric fields to text fields. Doing so prevents the database from performing calculations on those values. For example, if you

changed the unit_price column to VARCHAR(10), you would no longer be able to calculate the line item's expense by multiplying unit_price by quantity.

The structure of the command looks like the following:

```
ALTER TABLE <tablename>
ALTER COLUMN <columnname>
TYPE <newdatatype>;
```

Let's start with a new contact table that we created to handle requests for an application:

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

Let's add some sample data to the table. We will cover these statements in a future lesson:

```
INSERT INTO contact VALUES(1, 'bob798', 'mypass1',1);
INSERT INTO contact VALUES(2, 'jen558', 'mypass2',1);
INSERT INTO contact VALUES(3, 'rand058', 'mypass3',1);
```

TERM TO KNOW

Casting

To change the data type for all entries in a column when that column's data type changes.

2. Changing the Opt_In Column

Suppose that the company decided that instead of a number 0 or 1, it wanted to use Y or N in the column. We need to pick a data type that enables us to keep 0 or 1 and use Y or N in the column. We could use either `VARCHAR(1) or CHAR(1)`; either will allow for both values, Y/N and 0/1, in the table. Let's use `CHAR(1)` for this example. We are deliberately using `CHAR(1)` because of the expectation that the opt_in column will contain only a single character. It will error if a two-character value is entered, such as a number greater than 9.

```
ALTER TABLE contact
ALTER COLUMN opt_in
TYPE CHAR(1);
```

Changing from an integer to a character here does not create an error because integers are allowed character types in CHAR. Let's add more sample data into the table using the updated data type. Notice that we are now using Y instead of 1 in the opt_in column:

```
INSERT INTO contact VALUES(4, 'jeff768', 'mypass4',' Y');
INSERT INTO contact VALUES(5, 'sandra547', 'mypass5',' Y');
INSERT INTO contact VALUES(6, 'roberto9128', 'mypass6',' Y');
```

However, if we tried to convert opt_in back to an int, we should get an error:

```
ALTER TABLE contact
ALTER COLUMN opt_in
TYPE int;
```

Query Results

Query failed because of: error: column "opt_in" cannot be cast automatically to type integer

This is because we have the character Y in the opt_in table in the last three records:

Query Results

Row count: 6

contact_id	username	password	opt_in
1	bob798	mypass1	1
2	jen558	mypass2	1
3	rand058	mypass3	1
4	jeff768	mypass4	Y
5	sandra547	mypass5	Y
6	roberto9128	mypass6	Y

This is an issue to consider when you change the data type, as all numerals are characters, but not all characters are numerals. You might run into a similar problem if the opt_in column had a value made up of two or more characters, because the new CHAR(1) data type allows only one character. When we change the data type, it is important to consider both the data that the table already contains, and what it should be changed to. If there isn't any data at all in the table yet, it won't be an issue to make those data conversions using the `ALTER TABLE` command. If there is data in the table already, it is important to choose the correct data types that allow for current and new data to be inserted into the table.

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.

SUMMARY

In this lesson, you learned how to use the ALTER TABLE's TYPE option to **change a column's** data type. You saw an example of doing this to **change the opt_in column** in a sample table from the INT type to the CHAR type, and to enforce a one-character size limit on the data entered. You also learned about some limitations in changing the data type of a column that already contains data.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR **TERMS OF USE**.

TERMS TO KNOW

Casting

To change the data type for all entries in a column when that column's data type changes.