UNIQUE to Validate Data

by Sophia

WHAT'S COVERED

This tutorial explores the use of the UNIQUE constraint to make sure that the data in a column or columns are unique across all of the rows in two parts:

1. The UNIQUE Constraint
2. Using the ALTER TABLE Statement

1. The UNIQUE Constraint

We've explored a bit of the UNIQUE constraint in the tutorial on the CREATE TABLE statement using the primary key. However, we can extend the UNIQUE constraint to other columns or groups of columns to ensure that they are unique within a table. If the UNIQUE constraint is in place in a table, when we insert a new row in the table, it will check if the value already exists in the table. If it does, the database should reject the change and issue an error to the user. The same is the case if we try to update a row in a table where the update violates the constraint.

Note that when we add a UNIQUE constraint to a column or a group of columns, the database does add a unique index on the selection of columns or groups of columns. A simple example would be to look at the contact table we've created in a prior tutorial, but have it such that the username that is entered must be unique.

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

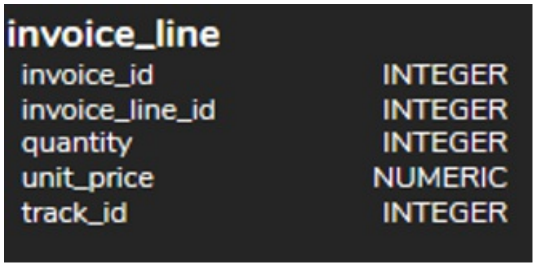
This will create a UNIQUE constraint on the username column. If we inserted a row or updated a row into the table that had the same username as an existing row, we should see an error similar to the following:

```
Query Results
Query failed because of: error: duplicate key value violates unique constraint "contact_username_key"
```

We can also change this to set the UNIQUE constraint as a table constraint by doing the following:

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

This is an important factor as we could pass multiple column names in the table constraint. A perfect example of this could be in our invoice_line table:



The invoice_line has two foreign keys: the invoice_id that references the invoice_id in the invoice table, and then the track_id in the track table. For a given invoice, the track_id should exist only once, because if a customer purchased more than one track in the same order, the quantity would be incremented. In this case, the invoice_id and the track_id together should be unique. We could do this by adding the following in the CREATE TABLE statement as a table constraint:

```
UNIQUE(invoice_id, track_id)
```

This will ensure that for any given invoice_line row, the combination of the invoice_id and track_id must be unique in the entire table.

---

2. Using the ALTER TABLE Statement

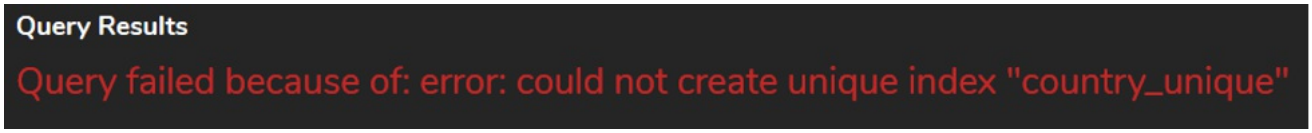
Note that we can also add the UNIQUE constraint on an existing table through the ALTER TABLE statement. The statement looks like the following:

```
ALTER TABLE <tablename> ADD CONSTRAINT <constraintname> UNIQUE (<column>);
```

The statement is very similar to the CREATE TABLE statement option but one thing we'll be doing is specifying the constraint's name, the table name, and the column or column list of the UNIQUE identifier.

Note that the ALTER TABLE statement must be completed on a column that has UNIQUE data. For example, in looking at the customer table, if we tried to create a unique constraint on the column country, we would get the following:

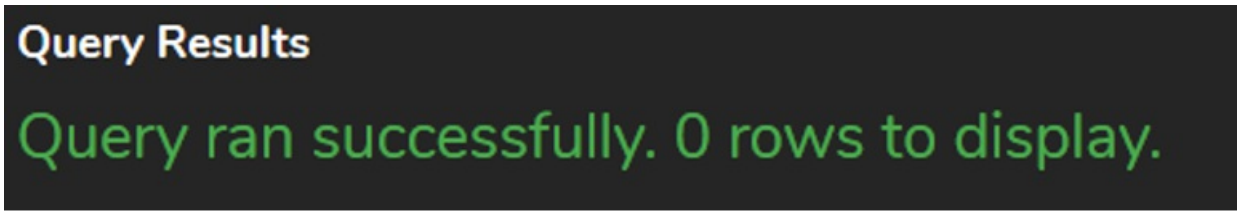
```
ALTER TABLE customer ADD CONSTRAINT country_unique UNIQUE (country);
```



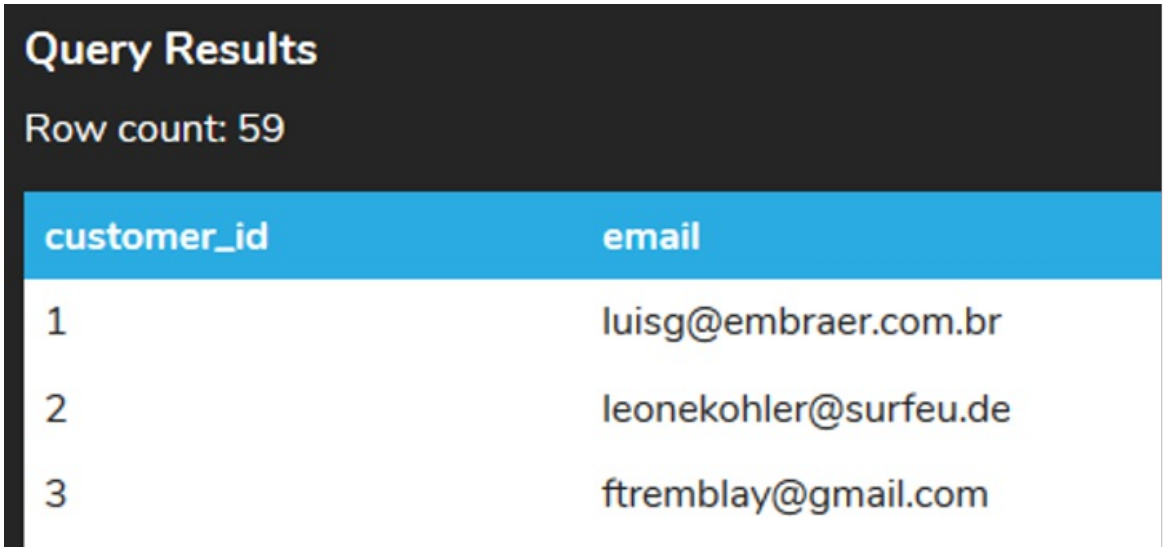
The constraint could not be added, because the country may be repeated for different customers. However,

we could add a constraint on the customer's email, which is unique:

```
ALTER TABLE customer ADD CONSTRAINT email_unique UNIQUE (email);
```

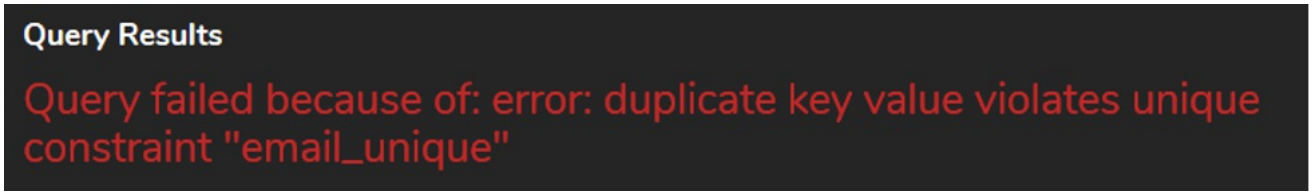


Let us query the email list to test this:



If we tried to set the customer with the customer_id equal to 1 to have the same email address as what customer_id equal to 3 has, we can see the following result:

```
UPDATE customer
SET email = 'ftremblay@gmail.com'
WHERE customer_id = 1;
```



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

The UNIQUE constraint allows us to enforce values stored in a column or a group of columns so that they are distinct from one another.

Source: Authored by Vincent Tran