DROP INDEX to Remove Indexes

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores the use of the DROP INDEX command to remove indexes in two parts:

1. Index Creation
2. Dropping an Index

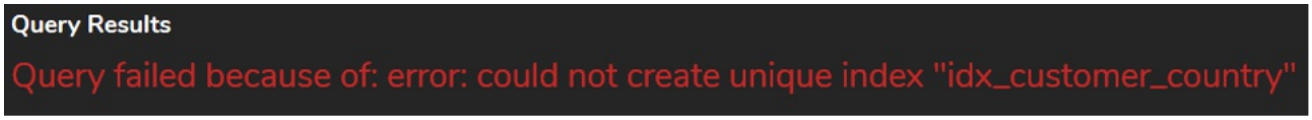
1. Index Creation

We previously looked at the creation of indexes through the use of primary keys and foreign keys. Both of those constraints automatically create the indexes. The CREATE INDEX statement looks like this:

CREATE [UNIQUE] INDEX <indexname>
<tablename> (<columnname>) [USING method];
For example, if we wanted to create a unique constraint and index on the email address in the customer table, we could do:

CREATE UNIQUE INDEX idx_customer_email
ON customer(email);
However, we could not create a UNIQUE constraint and index on the country, as the country does repeat by customer:

CREATE UNIQUE INDEX idx_customer_country
ON customer(country);



You could, however, create an index on the country in general:

CREATE INDEX idx_customer_country
ON customer(country);
We can add the type of index that we want by adding the USING method. By default, the b-tree option is

selected. However, you can pass in a hash, gist, or gin. Note, though, that not all versions of PostgreSQL support this.

```
CREATE INDEX idx_customer_country USING hash
ON customer(country);
```

---

2. Dropping an Index

Once we have created an index, removing the index is quite simple. We simply do the following:

```
DROP INDEX [CONCURRENTLY] [IF EXISTS] <indexname> [CASCADE or RESTRICT];
```

You have options when removing the index, similar to some of the other DROP statements for other objects. Typically, we would want to remove any unused indexes in consideration of the database's performance.

IF EXISTS will only remove an index if it exists. Using IF EXISTS will not throw an error if the index does not exist.

```
DROP INDEX IF EXISTS idx_customer_country;
```

CASCADE will automatically drop any objects that depend on the index that we are dropping.

```
DROP INDEX idx_customer_country CASCADE;
```

RESTRICT is set by default, and will not drop the index if we have any objects that depend on it.

```
DROP INDEX idx_customer_country RESTRICT;
```

When we run the DROP INDEX statement, PostgreSQL will get a lock on the table and block any other access to the table until the dropping of the index is complete. If there is a conflicting transaction that is running on the table, we can add CONCURRENTLY to the command to wait until any conflicting transactions are done before we remove the index. Note that if we do use the DROP INDEX CONCURRENTLY option, we cannot use the CASCADE option.

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

Dropping an index is important to run if you do not use an index, to preserve database performance.

Source: Authored by Vincent Tran