UNION to Combine Results

by Sophia

WHAT'S COVERED

This lesson explores the UNION operator to combine result sets of various queries, in five parts. Specifically, this lesson will cover:

1. Introduction

You can use the **UNION operator** to combine data from multiple tables or queries. This can be useful when you want to temporarily combine multiple tables for some specific purpose.

The syntax of the statement is as follows:

```
SELECT <columnlist>
FROM <table1>
UNION
SELECT <columnlist>
FROM <table2>;
```

The SELECT statements must use the same number of columns in the same order, and the corresponding columns must have the same data types. In the result set, column names are determined by the column names in the first SELECT statement.

KEY CONCEPT

A UNION operation **deduplicates** the result set—that is, it removes any duplicates. If you don't want duplicates to be removed, you can use UNION ALL instead.

TERMS TO KNOW

Union Operator

An operator that combines the results of two or more SELECT statements and removes duplicate rows.

Deduplication

The removal of duplicates.

---

2. Examples

Let's look at an example in which our organization may want to email all users (customers and employees) an email to inform them of an upcoming sale. The same message would be sent to all individuals. Instead of querying the customer and employee tables separately and having two result sets merge together, we can pull all the names and email addresses into a single result set:

```
SELECT first_name, last_name, email
FROM employee
UNION
SELECT first_name, last_name, email
FROM customer;
```

Query Results

Row count: 67

first_name	last_name	email
Manoj	Pareek	manoj.pareek@rediff.com
Alexandre	Rocha	alero@uol.com.br
Hannah	Schneider	hannah.schneider@yahoo.de
Wyatt	Girard	wyatt.girard@yahoo.fr
Helena	Holý	hholy@gmail.com
Julia	Barnett	jubarnett@gmail.com

Note that this will not distinguish which individual is a customer or employee. We could add an extra string in the column list to distinguish the difference:

```
SELECT first_name, last_name, email, 'Employee'
FROM employee
UNION
SELECT first_name, last_name, email, 'Customer'
FROM customer;
```

Query Results

Row count: 67

first_name	last_name	email	Employee
Steve	Johnson	steve@chinookcorp.com	Employee
Laura	Callahan	laura@chinookcorp.com	Employee
Margaret	Park	margaret@chinookcorp.com	Employee
Robert	King	robert@chinookcorp.com	Employee
Andrew	Adams	andrew@chinookcorp.com	Employee
Nancy	Edwards	nancy@chinookcorp.com	Employee
Michael	Mitchell	michael@chinookcorp.com	Employee
Jane	Peacock	jane@chinookcorp.com	Employee
Leonie	Köhler	leonekohler@surfeu.de	Customer
Hugh	O'Reilly	hughoreilly@apple.ie	Customer

To use the UNION operator, the tables we are querying from should have the same attribute characteristics, meaning that the number of columns and data types between the two SELECT statements should match. Here's an example of a situation where they do NOT match:

```
SELECT customer_id
FROM invoice
UNION
SELECT first_name
FROM customer;
```

In the example above, the customer_id is an integer, while the first_name is a character string. As a result, we get the following error:

Query Results

Query failed because of: error: UNION types integer and character varying cannot be matched

If we have a different number of columns in each of the SELECT statements, we would run into another error. For example, in the following query, there is only one column referenced in the first SELECT statement but two columns referenced in the second one.

```
SELECT customer_id
FROM invoice
UNION
SELECT customer_id, first_name
FROM customer;
```

Query Results

Query failed because of: error: each UNION query must have the same number of columns

3. Multiple UNION

We could also create a UNION for more than two queries. For example, we may want to look at all the countries we operate in. We would need to look at the customer table, invoice, and employee table.

```
SELECT billing_country
FROM invoice
UNION
SELECT country
FROM customer
UNION
SELECT country
FROM employee;
```

Notice that the result set only has 24 rows, as it excludes all duplicate values:

4. Retaining Duplicates

If we wanted to retain the duplicate values, we would use UNION ALL instead of UNION:

```
SELECT billing_country
FROM invoice
UNION ALL
SELECT country
FROM customer
UNION ALL
SELECT country
FROM employee;
```

Query Results

Row count: 479

billing_country
Germany
Norway
Belgium
Canada
USA
Germany
Germany

You should see now that the duplicates are included, and we have 479 results returned.

5. Column Names

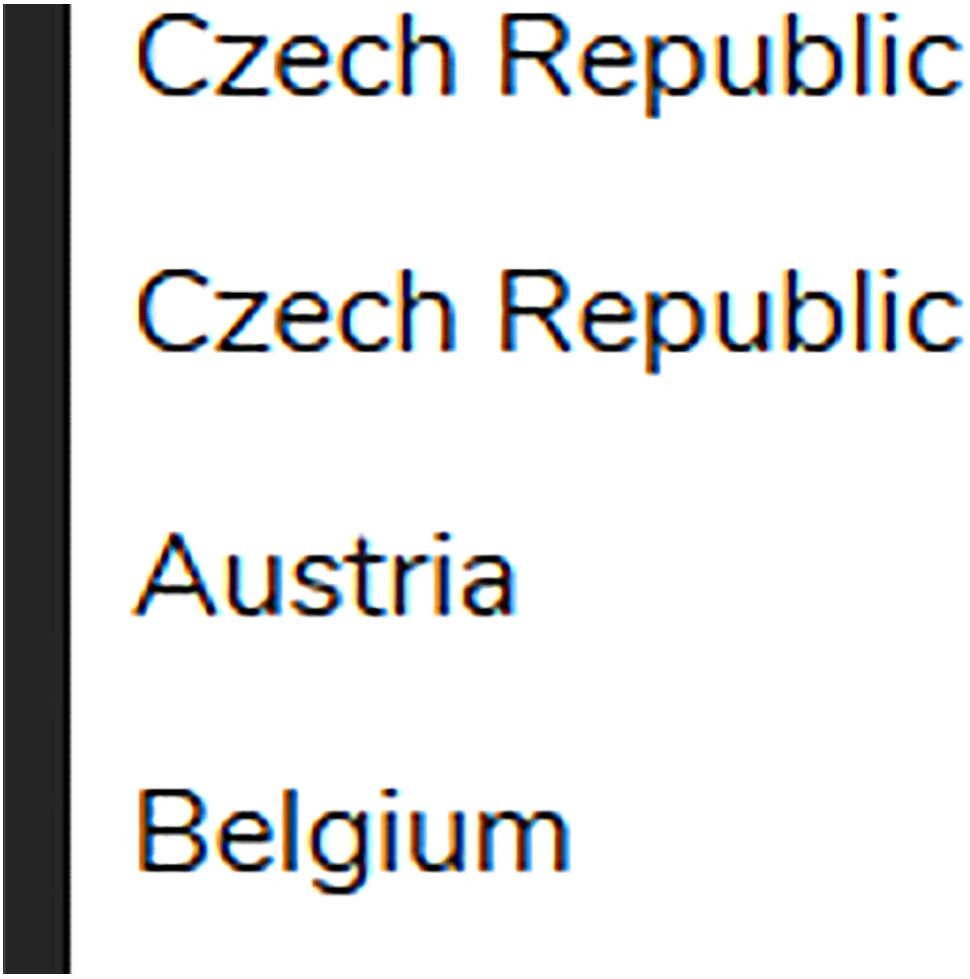
Note as well that the column name that is used as the output reflects the first SELECT statement in the list. If we swapped the first SELECT statement with the second, the country column displays as the output:

```
SELECT country
FROM customer
UNION ALL
SELECT billing_country
FROM invoice
UNION ALL
SELECT country
FROM employee;
```

Query Results

Row count: 479

country
Brazil
Germany
Canada
Norway



As such, if we wanted to use aliases to rename a column, we would only need to do so for the first SELECT statement:

```
SELECT first_name as "First Name", last_name as "Last Name", email as "Email", 'Employee' as "Type"
FROM employee
UNION
SELECT first_name, last_name, email, 'Customer'
FROM customer;
```

Query Results

Row count: 67

First Name	Last Name	Email	Type
Emma	Jones	emma_jones@hotmail.com	Customer
Stanisław	Wójcik	stanisław.wójcik@wp.pl	Customer
Victor	Stevens	vstevens@yahoo.com	Customer
Daan	Peeters	daan_peeters@apple.be	Customer
Alexandre	Rocha	alero@uol.com.br	Customer

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, in the **introduction** you learned that the UNION command is a powerful SQL command that combines the results of multiple SELECT statements. Data is combined efficiently, and duplicate rows are automatically removed, resulting in only unique records in the final result.

You saw several **examples** demonstrating the UNION operator, in which two SELECT statements are run in a single statement, with the word UNION between them. For successful merging, the **column names**, order, and data types must be the same in both SELECT statements. **Multiple UNION** operations are possible in the same statement, including any number of tables. The UNION command deduplicates the result set; if you want to **retain duplicates**, you can use UNION ALL instead.

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

Deduplication
The removal of duplicates.

Union Operator
An operator that combines the results of two or more SELECT statements and removes duplicate rows.