2018-07-17T22:43:49+01:00
Dorota Wdzięczna

Using CASE with Data Modifying Statements

What happens when you combine CASE with SQL's data modifying statements? Find out in this article.

The `CASE` expression is a very useful part of SQL and one that you'll employ frequently. We've already covered what the CASE expression does, how to format it, and how to use it in a `SELECT` statement in "Using CASE to Add Logic to a SELECT". Another article, "How to Sort Records with the ORDER BY Clause" demonstrated how to use `CASE` in an `ORDER BY` clause.

But we have not exhausted all the possibilities of the `CASE` expression. In this post, we'll discuss the use of `CASE` with SQL's data modifying statements.

What Are SQL's Data Modifying Statements?

There are three data modifying statements in SQL. They are called data modifying because they make changes to the data that is stored within a database table. These statements are quite familiar, even to novice SQL users. They are:

- `INSERT` – Adds records to a table
- `UPDATE` – Changes values in table columns
- `DELETE` – Removes records from the table

Finish your CRUD learning! With this course, you'll learn How to INSERT, UPDATE, and DELETE Data in SQL.

We will see how each of these statements works with a `CASE` expression. However, before we begin to write queries, let's look at the **`car`** table, shown below. We will use it in all our examples:

id	brand	model	info
1	Ford	Fiesta	NULL
2	Ford	Mondeo	NULL
3	Opel	Astra	NULL
4	Ford	Galaxy	NULL
5	Toyota	Corolla	NULL
6	Toyota	RAV4	NULL

As you can see, this table stores details about cars: their internal ID number, the brand of the car, the model name, and an optional field for additional information.

We'll start our investigation of using data modifying statements with the `CASE` expression with the `INSERT` statement.

Inserting Records Using CASE

You can use an `INSERT` statement to enter new records into a table. If you want to move records from one table and into another, you use the `SELECT INTO` statement.

Our first example will demonstrate adding information into a new table, **`tab`**. This new table will be created with the query. Notice that we are actually using a `SELECT INTO` statement to do this. Here is how it works:

```
SELECT brand, model,
          CASE brand
                WHEN 'Ford' THEN 'Founder was Henry Ford'
                WHEN 'Opel' THEN 'Founder was Adam Opel'
                WHEN 'Toyota' THEN 'Founder was Kiichiro Toyoda'
                 ELSE '-'
        END AS info
INTO tab
FROM car;
```

This `SELECT INTO` statement starts by taking data from the `brand` and `model` columns in the **`car`** table and moving it into the newly-created **`tab`** table. The third column in **`tab`** is the `info` column. The value in the `info` column depends on the value in the `car.brand` column. If the value in the `car.brand` column is "Ford", the "Founder was Henry Ford" string will be displayed in the `tab.info` column. If the brand is "Opel", then the "Founder was Adam Opel" string will be displayed; for "Toyota", the string "Founder was Kiichiro Toyoda" will be displayed. In any remaining cases, the returned value will be "- ".

Here is the resulting **`tab`** table and its content:

brand	model	info
Ford	Fiesta	Founder was Henry Ford
Ford	Mondeo	Founder was Henry Ford
Opel	Astra	Founder was Adam Opel
Ford	Galaxy	Founder was Henry Ford
Toyota	Corolla	Founder was Kiichiro Toyoda
Toyota	RAV4	Founder was Kiichiro Toyoda

Using CASE with UPDATE

The `CASE` expression can also be used in an `UPDATE` statement. You are familiar with the `UPDATE` statement; it changes or *updates* existing column values. If you want to update records based on column values, you can do it with the `CASE` expression. Look at the following example:

```
UPDATE car SET info =
CASE brand
          WHEN 'Ford' THEN 'Founder was Henry Ford'
          WHEN 'Toyota' THEN 'Founder was Kiichiro Toyoda'
          ELSE '-'
          END
;
```

We see that the information in the `info` column has been updated. Usually, we would put the new value in the `SET` clause. Instead, this predetermined value has been replaced by the `CASE` expression. The `CASE` tests the value in the `brand` column and uses that to determine what to enter in the `info` column. If anything other than "Ford" or

"Toyota" is stored in the `brand` column, a "-" string will be displayed into the `info` column for that record.

Here is the `car` table after the update:

id	brand	model	info
1	Ford	Fiesta	Founder was Henry Ford
2	Ford	Mondeo	Founder was Henry Ford
3	Opel	Astra	–
4	Ford	Galaxy	Founder was Henry Ford
5	Toyota	Corolla	Founder was Kiichiro Toyoda
6	Toyota	RAV4	Founder was Kiichiro Toyoda

Notice that the record for the Opel Astra doesn't have anything in the `info` column. This is because "Opel" was not specified as a brand in the `WHEN` of the `CASE` expression. The value defined in `ELSE` has been returned.

Deleting Data Using the CASE Expression

Now let's discuss the last of the data modifying statements: `DELETE`. It is used to remove records from a table. Once again, you can use the `CASE` expression in the `WHERE` clause to control your results. Look at this example:

```
DELETE FROM car
WHERE FALSE = CASE
    WHEN info IS NOT NULL THEN TRUE
      ELSE FALSE
END
;
```

This `CASE` tests if the value in the `info` column is NULL. If it is not NULL, then the TRUE value is returned. If the value is NULL, the FALSE value from `ELSE` is returned. If the `CASE` expression returns a FALSE value, that record is deleted.

Below we see the `car` table after rows with NULL values in the `info` column have been deleted:

id	brand	model	info

id	brand	model	info
1	Ford	Fiesta	Founder was Henry Ford
2	Ford	Mondeo	Founder was Henry Ford
4	Ford	Galaxy	Founder was Henry Ford
5	Toyota	Corolla	Founder was Kiichiro Toyoda
6	Toyota	RAV4	Founder was Kiichiro Toyoda

Learn More About the SQL CASE Expression

If you are interested in learning more about the `CASE` expression, check out LearnSQL's Standard SQL Functions course. You will learn more about this expression and others. Plus, you can practice your new skills with interactive exercises. Try it today!

Do you need to learn SQL functions but don't know which base you'll be using in the future? Check out our Standard SQL Functions course which works with all SQL databases.
