CHECK to Validate Data

by Sophia

WHAT'S COVERED

In this lesson, you will explore using the CHECK constraint to keep the values in a column limited to a set of criteria, in three parts. Specifically, this lesson will cover:

1. Introducing the CHECK Constraint
2. Example Table
3. Error Messages

1. Introducing the CHECK Constraint

CHECK is a unique constraint that verifies that the values being added to a column meet a specific requirement that we define. The CHECK constraint uses a '**Boolean expression** to evaluate whether the value is valid when inserted into or updated in the column. If the check is valid, the database will insert or update those values in the column. However, if the check is not valid, the database will reject the changes and raise an error.

TERMS TO KNOW

CHECK

A constraint that uses a Boolean expression to evaluate whether the value is valid when inserted into or updated in the column.

Boolean Expression

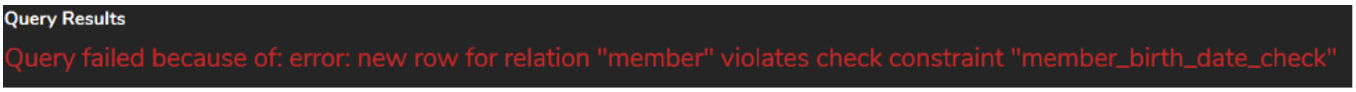
An expression that can be evaluated as either true or false.

2. Example Table

The CHECK constraint is generally set up when we create the table. Let us look at an example of a CREATE TABLE statement with various CHECK constraints.

```
CREATE TABLE member (
member_id SERIAL PRIMARY KEY,
first_name VARCHAR (50),
last_name VARCHAR (50),
birth_date DATE CHECK (birth_date > '1900-01-01'),
joined_date DATE CHECK (joined_date > birth_date),
opt_in CHAR(1) CHECK (opt_in IN ('Y','N')),
membership_fee numeric CHECK(membership_fee > 0)
);
```

Above, we have a table created with four CHECK constraints added with various data types. The first one is set based on the birth_date. It checks that the birth_date is after January 1, 1900. If we enter a birth date before 01/01/1900, we will get an error message:



The second CHECK constraint is based on the joined_date. It verifies that the member's joined_date is later than the birth_date. It wouldn't make sense to have a member joining before being born.

The third CHECK constraint is based on an opt_in column that is one character. It allows only a Y or N character. It could be argued that this could simply be set up as a Boolean. However, if the data is being sent in from a form, the value may need to be a specific character. We could also add other characters to be checked if required.

KEY CONCEPT

In a previous lesson, you used IN and BETWEEN operators as part of SELECT statements. You can also use them as part of CHECK constraints.

As you can see in the above example, you can use IN as part of a constraint to specify a list of values that the entry must match.

You can also use BETWEEN to constrain entries to within a range of values. You did this in the previous lesson by using two Boolean operators like this:

```
CHECK quantity (>=0 and <=10)
```

Using BETWEEN, you could write this same constraint as:

```
CHECK quantity BETWEEN 0 AND 10.
```

The last CHECK constraint verifies the membership_fee is greater than zero.

KEY CONCEPT

You can also include math operators in the CHECK constraint's comparison operators. For example, suppose you wanted the value in the order_qty column to be at least 2 less than the value in the qty_on_hand column:

CHECK (order_qty < qty_on_hand - 2)

3. Error Messages

You may have noticed in the error message that the constraint name was defined even though we did not set a name. In PostgreSQL, the constraint names are automatically created using the table name, column name, and type of constraint separated by underscores. If we wanted to have a specific name, we could replace the CHECK within the CREATE TABLE statement with a new line.

In this case, we name the CHECK constraint as positive_fee instead of the default member_membership_fee_check name. But what if you decide to add a CHECK constraint to a table that already exists? In this case, you would use the ALTER TABLE statement along with the ADD CONSTRAINT clause. For example, let's say you had forgotten to add the CHECK constraint to the birth_date field when you created the member table from the earlier example. You could add it later like this:

```
ALTER TABLE member
ADD CONSTRAINT birth_date_check
CHECK (birth_date > '1900-01-01');
```

In this example, birth_date_check is the constraint name, and birth_date > '1900-01-01' is the condition.

You will learn more about the ALTER TABLE statement later in the course.

In an upcoming lesson, we will explore how to add constraints after the table is created. In doing so, we can add more complex constraints over a series of columns.

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

During this lesson, you learned that the **CHECK constraint** in PostgreSQL is a mechanism that enables you to define custom rules and conditions for data validation. A CHECK constraint lets you specify criteria that values in a column must meet to be considered valid. You also learned that this constraint is

particularly useful for limiting valid values or ensuring data integrity by applying complex logical expressions. A CHECK constraint can be added to an existing table or applied during table creation. It maintains data consistency and accuracy by preventing records that do not meet the defined conditions from being inserted or modified. You examined an **example** of a CREATE TABLE statement with various CHECK constraints and learned that messages like the **error message** communicate specific things about the code in a table.

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

Boolean Expression

An expression that can be evaluated as either true or false.

CHECK

A constraint that uses a Boolean expression to evaluate whether the value is valid when inserted into or updated in the column.