## **Symbolic Logic and Truth Tables**

We have now seen examples of how to filter on a single column with some filtering criteria. But what about filtering on multiple columns within the same query? For this we can employ a subset of what SQL calls logical operators: the AND operator and the OR operator.

The query syntax used when employing these two logical operators is fairly simple. Instead of ending our query with a single filtering statement after the WHERE clause, we can add one of the logical operators and include a second filtering criteria. For example, if we wanted to return all rows from the HumanResources.vEmployee view where the employee’s first name is either Chris or Steve, we would execute:

SELECT \*

FROM HumanResources.vEmployee

WHERE FirstName = 'Chris' OR FirstName = 'Steve'

Notice that the second criteria in the WHERE clause contains all three components of the WHERE clause: the column name to be filtered on, the comparison operator, and then the filtering criteria. The only subtle difference is that you do not need to add “WHERE” for a second time.

Let’s now look at an example of the AND logical operator being used. Suppose we wanted to return all rows from the Production.Product table where the ListPrice value is greater than 100 and the Color column has a value of “Red”, we would execute the query:

SELECT \*

FROM Production.Product

WHERE ListPrice > 100 AND Color = 'Red'

The WHERE clause is similar to the previous example; the only difference is the logical operator that we have used and the filtering criteria.

As you may have guessed, we can string together more than two logical operators in a single WHERE clause. For example, let’s suppose we wanted to find all rows in the Production.Product table that have a ListPrice greater than 100, a color equal to “Red”, SafetyStockLevel equal to 500, and a Size greater than 50. To complete this query we execute:

SELECT \*

FROM Production.Product

WHERE ListPrice > 100 AND Color = 'Red' AND SafetyStockLevel = 500 AND Size > 50

Nothing complicated has happened here; we simply add another logical operator and then another filtering statement. Complexity increases when we start to mix the two different logical operators we have been using in the WHERE clause. The SQL parser will evaluate the WHERE clause from left to right – that is the order of operations does not give priority to either the AND or the OR operator. We will look at a few examples of this and show how we can explicitly ensure we get the results we would like.

Let’s modify the previous example and try to retrieve only those rows from the Production.Product table where the ListPrice is greater than 100, the color is “Red” **OR** the StandardCost is greater than 30. To do this, we would execute the query:

SELECT \*

FROM Production.Product

WHERE ListPrice > 100 AND Color = 'Red' OR StandardCost > 30

Examine the results for a brief moment. You might notice that you have many rows returned where the color is not equal to “Red”. You even find rows with a value in the ListPrice column that is less than 100. This is not a mistake – in fact this is the 100% correct data set returned by SQL. To explain why this is the case, we need to take a step back and understand the concept of truth tables and properly evaluating Boolean expressions.

Determining whether or not a Boolean expression returns a TRUE or FALSE value is an exact activity. That is, there are very clear rules to be followed – subjectivity is not a factor in determining the truth of a Boolean expression. In most computer science or mathematics curriculums in college, a course in symbolic logic is often required for graduation and often a prerequisite for more advanced courses. Set theory in mathematics and much of programming relies heavily on the concepts learned in a symbolic logic course. Truth tables are one of the core concepts in this course, and they help new students of the subject determine whether or not a statement is true or false based on the truth of the individual components that make up the Boolean expression.

A truth table breaks down a Boolean expression into its simple components. Taking one of the filtering criteria from one of our earlier examples, let’s look at the query:

SELECT \*

FROM Production.Product

WHERE ListPrice > 100 AND Color = 'Red'

There are two filtering components here: ListPrice is greater than 100, and Color is equal to “Red”. Symbolic logic courses, and truth tables, would give each of these components a letter as a symbol: let’s use A and B to give it basic. So operator A is the filtering component that the ListPrice is greater than 50, and operator B is the filtering component that the color is equal to “Red”.

In a truth table, we would take these two components and list all possible combinations that these two components could be true or false (T implies TRUE and F implies FALSE). A row’s ListPrice column could be greater than 100 or less than or equal to 100. Therefore the expression “ListPrice > 100” has two possible outcomes: true or false. The same goes for the second filtering criteria “Color = “Red”. The Color column’s value could be “Red” or it might not be “Red”, thus returned either a TRUE or FALSE value for that Boolean expression. The truth table below summarizes the combination of these possible outcomes:

|  |  |
| --- | --- |
| **A** | **B** |
| T | T |
| T | F |
| F | T |
| F | F |

There are four different outcomes for each of these Boolean expressions: A is TRUE and B is TRUE, A is TRUE and B is FALSE, A is FALSE and B is TRUE, and A is FALSE and B is FALSE. To further clarify this with examples, if the ListPrice of a row was greater than 100 and the Color was “Red”, both A and B (the two filtering criteria symbolized) would be TRUE and the first row would represent the evaluation of the two expressions. If the ListPrice was not greater than 100, but the Color was “Red”, then A would be FALSE and B would be TRUE, implying the third row symbolizes the truth of each individual component.

Identifying a Boolean value for each component individually is fairly simple: the ListPrice is either greater than 100 or it is not (i.e. it is either TRUE or FALSE). Where truth tables demonstrate their importance is when we apply an operator to the two individual components. In the WHERE clause of the example we have been using, AND is the logical operator connecting the two filtering criteria. Truth tables will then utilize a third column (in our example) so determine the truth of the two individual components when combined together by the logical operator in use:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A and B** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

Adding a third column to determine the Boolean evaluation of the expression “component A **AND** component B”, symbolic logic defines truth for the expression. When evaluation an expression of two components joined together by the “AND” operator, the only time “A and B” evaluates as TRUE is if both individual components evaluate to TRUE. This makes sense if we look at it in the context of the SQL query. If the ListPrice is greater than 100, but the Color column does not equal to “Red” then both criteria are not true. It then follows that any row where both criteria are not true, in this example, would not evaluate as TRUE and be returned by SQL. When we use the “AND” operator, we explicitly are telling the SQL parser that we only want the rows where **BOTH** filtering criteria are met. The above truth table simply visually represents the total possible outcomes for the expression given the combinations of truth for each individual components.

Let’s take the same query but modify the comparison operator from AND to OR:

SELECT \*

FROM Production.Product

WHERE ListPrice > 100 OR Color = 'Red'

Take a look at some of the results from this query. If either the ListPrice is greater than 100 or the Color is equal to “Red” then the row is returned to the results panel. Below is the truth table for two components joined with the “OR” operator:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A or B** |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

Notice how all rows except the last row in the truth table evaluate to TRUE when evaluating the Boolean expression “A or B”. With the OR operator, as long as one of the components yields a TRUE value, then the entire expression returns as TRUE. This is why some rows in the result set for the previous query appear in the panel when the ListPrice is greater than 100 but the Color is not “Red”.

Now that we have covered the basics of the truth tables for the “OR” and “AND” operators for two components, what happens when we combine them? If you remember, this discussion began when evaluating the query:

SELECT \*

FROM Production.Product

WHERE ListPrice > 100 AND Color = 'Red' OR StandardCost > 30

So, let’s break this WHERE clause down into the three separate filtering components and create a truth table. Component A will be “ListPrice > 100”, Component B will be “Color = ‘Red’”, and Component C will be “Standard Cost > 30”. Keep in mind that the criteria will be evaluated left to right with no priority given to either the “OR” or the “AND” operators. Because the database engine evaluates this criteria left to right (since no parentheses indicate any other order), the first Boolean expression to be evaluated will be “ListPrice > 100 and Color = ‘Red’” or “A and B”. The truth table for this is:

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A and B** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

Once that has been evaluated, then the database engine evaluates the next Boolean expression. This next expression is the result of “A and B” OR component C. We could visualize this as “(A and B) or C”. Now, the truth table for this final Boolean expression is:

|  |  |  |
| --- | --- | --- |
| **A and B** | **C** | **(A and B) or C** |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

So, regardless of whether or not (A and B) evaluates as FALSE, as long as component C is TRUE, then entire expression evaluates as TRUE. This is why we had what seemed to be strange results the first time that we evaluated this query. In fact, as long as one of the components of the OR expression are true, the entire expression is TRUE. So, if component C is FALSE but component (A and B) evaluates to TRUE, then (A and B) or C evaluates as TRUE. In terms of our SELECT statement, if the ListPrice was greater than 100 and the Color was “Red” but the StandardCost was not greater than 30, that particular row would still appear as part of our results.

This was just a very basic introduction to the concepts of symbolic logic and truth tables. The lab exercises will help to improve and reinforce the concepts learned.