SQL: AND & OR

The AND and OR are very useful when dealing with several conditions in the filter. We can see those really as the same type of Boolean expression than regular programming language.

A `AND` can be associated to a product whereas the `OR` to a sum or a difference. Hence, similar to basic algebra, we can use those in parentheses and the order of the parantheses has an importance.

Furthermore, the `OR` keyword by its nature, instanciates another filter. For instance

```
SELECT column1, column1, ... FROM table_name where condition1 and condition2;
```

To return something, in this case we need both `condition1` and `condition2` to be true.

```
SELECT column1, column2, ... FROM table_name WHERE condition1 OR condition2;
```

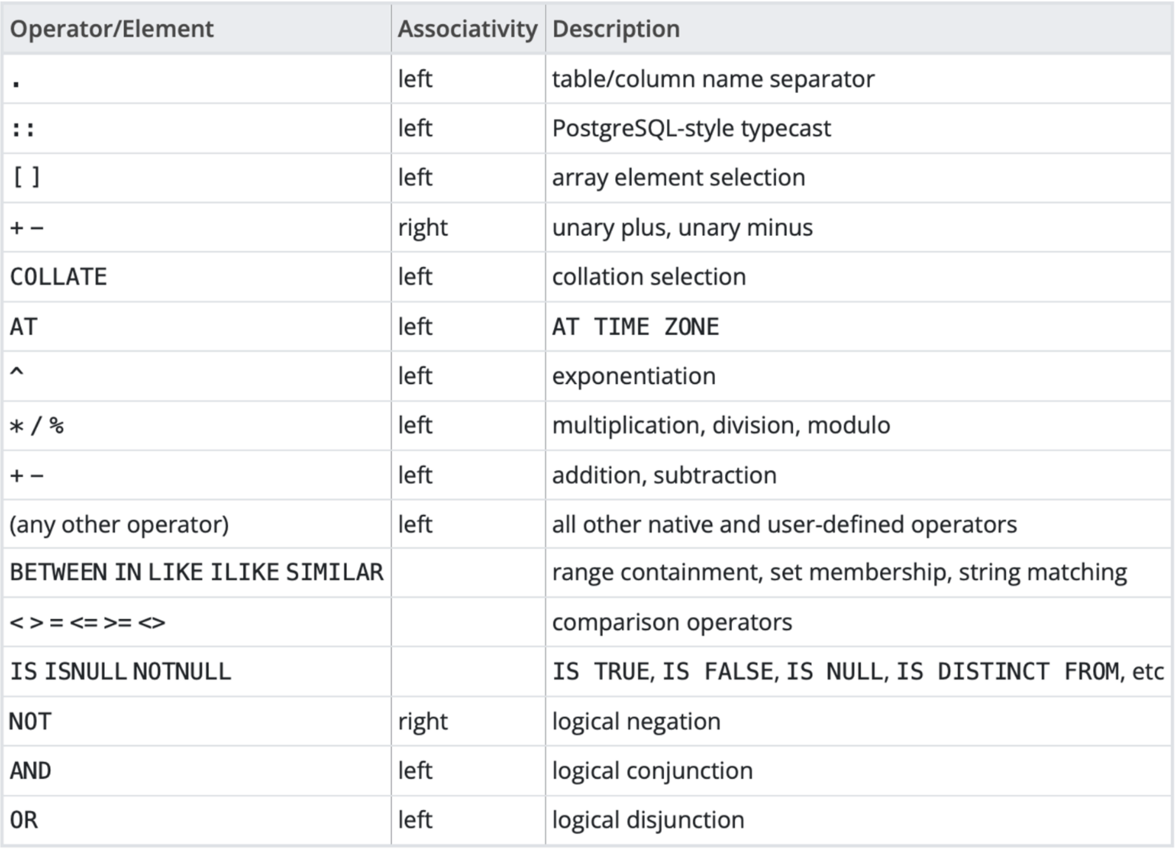
In this case, if either condition is true we will get something in return.

Now as for the parantheses:

```
SELECT column1, column2, ... FROM table_name WHERE (condition1 OR condition2) AND condition3
```

Here we have manage to gather the condition by putting in evidence that `condition3`. But we have several filters due to the presence of the OR.

The rules where we need to be careful with the order of operation is called the operator precedence (the operators that will be run before another one). Here is a list:



Operator/Element	Associativity	Description
.	left	table/column name separator
::	left	PostgreSQL-style typecast
[ ]	left	array element selection
+ -	right	unary plus, unary minus
COLLATE	left	collation selection
AT	left	AT TIME ZONE
^	left	exponentiation
* / %	left	multiplication, division, modulo
+ -	left	addition, subtraction
(any other operator)	left	all other native and user-defined operators
BETWEEN IN LIKE ILIKE SIMILAR		range containment, set membership, string matching
< > = <= >= <>		comparison operators
IS ISNULL NOTNULL		IS TRUE, IS FALSE, IS NULL, IS DISTINCT FROM, etc
NOT	right	logical negation
AND	left	logical conjunction
OR	left	logical disjunction

As an example:

```
-- example 1
SELECT column1 FROM table_name WHERE condition1 AND condition2 OR condition3

-- example 2
SELECT column1 FROM table_name WHERE condition1 AND (condition2 OR condition3)
```

From here we can see that we have two filter in each example. However, due to operator precedence, the AND will be operated only for `condition2`. By putting the `()` we enforce that the AND applies on both conditions.

The operator precedence must be read as we starting from the left or the right.

Examples of Using `AND` and `OR`

Let's go through some examples to see how these operators work in practice.

Example Table: Employees

employee_id	first_name	last_name	department	salary
1	John	Doe	Sales	50000
2	Jane	Smith	Marketing	60000
3	Alice	Johnson	Sales	55000
4	Bob	Brown	IT	70000

1. **Using `AND` to Combine Conditions**:

   Suppose you want to find all employees in the Sales department with a salary greater than 50,000:

   ```
   SELECT first_name, last_name, department, salary
   FROM Employees
   WHERE department = 'Sales' AND salary > 50000;
   ```

   Result:

first_name	last_name	department	salary
Alice	Johnson	Sales	55000

   In this example, the query returns only employees who meet **both** conditions: being in the Sales department **and** having a salary greater than 50,000.

2. **Using `OR` to Combine Conditions**:

   Now, suppose you want to find all employees who are either in the Sales department **or** have a salary above 60,000:

   ```
   SELECT first_name, last_name, department, salary
   FROM Employees
   WHERE department = 'Sales' OR salary > 60000;
   ```

   Result:

first_name	last_name	department	salary
John	Doe	Sales	50000
Alice	Johnson	Sales	55000

Bob	Brown	IT	70000

In this example, the query returns employees who meet **at least one** of the conditions: being in the Sales department **or** having a salary above 60,000.

3. **Combining `AND` and `OR`:**

   You can combine both `AND` and `OR` operators in a single query to create more complex conditions. Suppose you want to find employees who are in the Sales department **and** have a salary above 50,000, **or** are in the IT department:

   ```
   SELECT first_name, last_name, department, salary
   FROM Employees
   WHERE (department = 'Sales' AND salary > 50000) OR department = 'IT';
   ```

   Result:

first_name	last_name	department	salary
Alice	Johnson	Sales	55000
Bob	Brown	IT	70000

   Here, parentheses are used to group the `AND` condition together, ensuring that the query correctly interprets the logic: employees in the Sales department with a salary over 50,000 **or** employees in the IT department.

Key Points

- The `AND` operator requires **all conditions** to be true for a record to be included in the result set.
- The `OR` operator requires **at least one condition** to be true for a record to be included in the result set.
- You can use parentheses `()` to group conditions and control the order of evaluation when combining `AND` and `OR` operators.
- Using `AND` and `OR` allows you to create complex queries that filter data based on multiple criteria.

These operators are essential tools for querying databases, enabling precise data retrieval based on multiple conditions.