# POSTGRESQL - OPERATORS

 

## What is an Operator in PostgreSQL?

An operator is a reserved word or a character used primarily in a PostgreSQL statement's WHERE clause to perform operation*s*, such as comparisons and arithmetic operations.

Operators are used to specify conditions in a PostgreSQL statement and to serve as conjunctions for multiple conditions in a statement.

- Arithmetic operators
- Comparison operators
- Logical operators
- Bitwise operators

## PostgreSQL Arithmetic Operators:

Assume variable **a** holds 2 and variable **b** holds 3, then:

Show Examples

| Operator | Description | Example |
|---|---|---|
| + | Addition - Adds values on either side of the operator | a + b will give 5 |
| - | Subtraction - Subtracts right hand operand from left hand operand | a - b will give -1 |
| * | Multiplication - Multiplies values on either side of the operator | a * b will give 6 |
| / | Division - Divides left hand operand by right hand operand | b / a will give 1 |
| % | Modulus - Divides left hand operand by right hand operand and returns remainder | b % a will give 1 |
| ^ | Exponentiation - This gives the exponent value of the right hand operand | a ^ b will give 8 |
| \|/ | square root | \|/ 25.0 will give 5 |
| \|\|/ | Cube root | \|\|/ 27.0 will give 3 |
| !/ | factorial | 5 ! will give 120 |
| !! | factorial *prefixoperator* | !! 5 will give 120 |

## PostgreSQL Comparison Operators:

Assume variable a holds 10 and variable b holds 20, then:

Show Examples

| Operator | Description | Example |
|---|---|---|

| | | |
|---|---|---|
| = | Checks if the values of two operands are equal or not, if yes then condition becomes true. | $a = b$ is not true. |
| != | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | $a != b$ is true. |
| <> | Checks if the values of two operands are equal or not, if values are not equal then condition becomes true. | $a <> b$ is true. |
| > | Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true. | $a > b$ is not true. |
| < | Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true. | $a < b$ is true. |
| >= | Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true. | $a >= b$ is not true. |
| <= | Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true. | $a <= b$ is true. |

## PostgreSQL Logical Operators:

Here is a list of all the logical operators available in PostgresSQL.

Show Examples

| Operator | Description |
|---|---|
| AND | The AND operator allows the existence of multiple conditions in a PostgresSQL statement's WHERE clause. |
| NOT | The NOT operator reverses the meaning of the logical operator with which it is used. Eg. NOT EXISTS, NOT BETWEEN, NOT IN etc. **This is negate operator.** |
| OR | The OR operator is used to combine multiple conditions in a PostgresSQL statement's WHERE clause. |

## PostgreSQL Bit String Operators:

Bitwise operator works on bits and perform bit by bit operation. The truth table for & and | is as follows:

| p | q | p & q | p \| q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

1 0 0 1

Assume if A = 60; and B = 13; now in binary format they will be as follows:

A = 0011 1100

B = 0000 1101

---

A&B = 0000 1100

A|B = 0011 1101

~A = 1100 0011

The Bitwise operators supported by PostgreSQL are listed in the following table. Assume variable A holds 60 and variable B holds 13 then:

Show Examples

| Operator | Description | Example |
|---|---|---|
| & | Binary AND Operator copies a bit to the result if it exists in both operands. | A & B will give 12 which is 0000 1100 |
| \| | Binary OR Operator copies a bit if it exists in either operand. | $A \mid B$ will give 61 which is 0011 1101 |
| ~ | Binary Ones Complement Operator is unary and has the effect of 'flipping' bits. | $A$ will give -61 which is 1100 0011 in 2's complement form due to a signed binary number. |
| << | Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand. | A << 2 will give 240 which is 1111 0000 |
| >> | Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand. | A >> 2 will give 15 which is 0000 1111 |
| # | bitwise XOR. | A # B will give 49 which is 0100 1001 |