SQL Operators

Sometimes, expressions with a single condition are not enough to satisfy the user's requirement. For that, SQL supports Logical Operators to use multiple conditions at once. The most common logical operators are `AND`, `OR`, and `NOT`.

AND Operator

The `AND` operator takes in two conditions and returns `true` or `false` based on their evaluation:

Code: sql

```
condition1 AND condition2
```

The result of the `AND` operation is `true` if and only if both `condition1` and `condition2` evaluate to `true`:

SQL Operators

```
mysql> SELECT 1 = 1 AND 'test' = 'test';

+---------------------------+
| 1 = 1 AND 'test' = 'test' |
+---------------------------+
|                         1 |
+---------------------------+
1 row in set (0.00 sec)

mysql> SELECT 1 = 1 AND 'test' = 'abc';

+---------------------------+
| 1 = 1 AND 'test' = 'abc' |
+---------------------------+
|                         0 |
+---------------------------+
1 row in set (0.00 sec)
```

In MySQL terms, any `non-zero` value is considered `true`, and it usually returns the value `1` to signify `true`. `0` is considered `false`. As we can see in the example above, the first query returned `true` as both expressions were evaluated as `true`. However, the second query returned `false` as the second condition `'test' = 'abc'` is `false`.

OR Operator

The `OR` operator takes in two expressions as well, and returns `true` when at least one of them evaluates to `true`:

SQL Operators

```
mysql> SELECT 1 = 1 OR 'test' = 'abc';

+--------------------------+
| 1 = 1 OR 'test' = 'abc' |
+--------------------------+
|                        1 |
+--------------------------+
1 row in set (0.00 sec)

mysql> SELECT 1 = 2 OR 'test' = 'abc';

+--------------------------+
| 1 = 2 OR 'test' = 'abc' |
+--------------------------+
|                        0 |
+--------------------------+
1 row in set (0.00 sec)
```

The queries above demonstrate how the `OR` operator works. The first query evaluated to `true` as the condition `1 = 1` is `true`. The second query has two `false` conditions, resulting in `false` output.

NOT Operator

The `NOT` operator simply toggles a `boolean` value 'i.e. `true` is converted to `false` and vice versa':

SQL Operators

```
mysql> SELECT NOT 1 = 1;

+-----------+
| NOT 1 = 1 |
+-----------+
|         0 |
+-----------+
1 row in set (0.00 sec)

mysql> SELECT NOT 1 = 2;

+-----------+
| NOT 1 = 2 |
+-----------+
|         1 |
+-----------+
1 row in set (0.00 sec)
```

As seen in the examples above, the first query resulted in `false` because it is the inverse of the evaluation of `1 = 1`, which is `true`, so its inverse is `false`. On the other hand, the second query returned `true`, as the inverse of `1 = 2` 'which is `false`' is `true`.

Cheat Sheet

Go to Questions

Table of Contents

Introduction

Databases

Intro to Databases

Types of Databases

MySQL

Intro to MySQL

SQL Statements

Query Results

SQL Operators

SQL Injections

Intro to SQL Injections

Subverting Query Logic

Using Comments

Union Clause

Union Injection

Exploitation

Database Enumeration

Reading Files

Writing Files

Mitigations

Mitigating SQL Injection

Closing It Out

Skills Assessment - SQL Injection Fundamentals

My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

Symbol Operators

The `AND`, `OR` and `NOT` operators can also be represented as `&&`, `||` and `!`, respectively. The below are the same previous examples, by using the symbol operators:

SQL Operators

```
mysql> SELECT 1 = 1 && 'test' = 'abc';

+-------------------------+
| 1 = 1 && 'test' = 'abc' |
+-------------------------+
|                       0 |
+-------------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SELECT 1 = 1 || 'test' = 'abc';

+-------------------------+
| 1 = 1 || 'test' = 'abc' |
+-------------------------+
|                       1 |
+-------------------------+
1 row in set, 1 warning (0.00 sec)

mysql> SELECT 1 != 1;

+--------+
| 1 != 1 |
+--------+
|      0 |
+--------+
1 row in set (0.00 sec)
```

Operators in queries

Let us look at how these operators can be used in queries. The following query lists all records where the `username` is NOT `john`:

SQL Operators

```
mysql> SELECT * FROM logins WHERE username != 'john';

+----+---------------+------------+---------------------+
| id | username      | password   | date_of_joining     |
+----+---------------+------------+---------------------+
|  1 | admin         | p@ssw0rd   | 2020-07-02 00:00:00 |
|  2 | administrator | adm1n_p@ss | 2020-07-02 11:30:50 |
|  4 | tom           | tom123!    | 2020-07-02 11:47:16 |
+----+---------------+------------+---------------------+
3 rows in set (0.00 sec)
```

The next query selects users who have their `id` greater than `1` AND `username` NOT equal to `john`:

SQL Operators

```
mysql> SELECT * FROM logins WHERE username != 'john' AND id > 1;

+----+---------------+------------+---------------------+
| id | username      | password   | date_of_joining     |
+----+---------------+------------+---------------------+
|  2 | administrator | adm1n_p@ss | 2020-07-02 11:30:50 |
|  4 | tom           | tom123!    | 2020-07-02 11:47:16 |
+----+---------------+------------+---------------------+
2 rows in set (0.00 sec)
```

Multiple Operator Precedence

SQL supports various other operations such as addition, division as well as bitwise operations. Thus, a query could have multiple expressions with multiple operations at once. The order of these operations is decided through operator precedence.

Here is a list of common operations and their precedence, as seen in the MariaDB Documentation:

- Division (`/`), Multiplication (`*`), and Modulus (`%`)
- Addition (`+`) and subtraction (`-`)
- Comparison (`=`, `>`, `<`, `<=`, `>=`, `!=`, `LIKE`)
- NOT (`!`)
- AND (`&&`)
- OR (`||`)

Operations at the top are evaluated before the ones at the bottom of the list. Let us look at an example:

Code: sql

```sql
SELECT * FROM logins WHERE username != 'tom' AND id > 3 - 2;
```

The query has four operations: `!=`, `AND`, `>`, and `-`. From the operator precedence, we know that subtraction comes first, so it will first evaluate `3 - 2` to `1`:

Code: sql

```sql
SELECT * FROM logins WHERE username != 'tom' AND id > 1;
```

Next, we have two comparison operations, `>` and `!=`. Both of these are of the same precedence and will be evaluated together. So, it will return all records where username is not `tom`, and all records where the `id` is greater than 1, and then apply `AND` to return all records with both of these conditions:

SQL Operators

```
mysql> select * from logins where username != 'tom' AND id > 3 - 2;
```

```
+----+---------------+------------+---------------------+
| id | username      | password   | date_of_joining     |
+----+---------------+------------+---------------------+
|  2 | administrator | adm1n_p@ss | 2020-07-03 12:03:53 |
|  3 | john          | john123!   | 2020-07-03 12:03:57 |
+----+---------------+------------+---------------------+
2 rows in set (0.00 sec)
```

We will see a few other scenarios of operator precedence in the upcoming sections.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 163ms

Terminate Pwnbox to switch location

Start Instance

∞ / 1 spawns left

Waiting to start...

Enable step-by-step solutions for all questions

Questions

Cheat Sheet

Answer the question(s) below to complete this Section and earn cubes!

Target(s): Click here to spawn the target system!

Authenticate to with user "root" and password "password"

+ 1 In the 'titles' table, what is the number of records WHERE the employee number is greater than 10000 OR their title does NOT contain 'engineer'?

654

Submit Hint

← Previous Next →

Mark Complete & Next