How to Use LIKE in SQL: SQL Pattern Matching

Dorota Wdzięczna
Dorota is an IT engineer and works as a Data Science Writer for Vertabelo. She has experience as a Java programmer, webmaster, teacher, lecturer, IT specialist, and coordinator of IT systems. In her free time, she loves working in the garden, taking photos of nature, especially macro photos of insects, and visiting beautiful locations in Poland.

Tags: LIKE MATCHING PATTERNS IN SQL SQL OPERATOR

SQL pattern matching is a very important and useful ability. In this article, we look at how you can perform it using LIKE in SQL.

> The best way to learn SQL is through practice. Try out our interactive SQL Basics course.

SQL Pattern matching is a very simple concept. It allows you to search strings and substrings and find certain characters or groups of characters. Apart from SQL, this operation can be performed in many other programming languages.

In this article, we'll examine **how you can use `LIKE` in SQL to search substrings**. We'll also make the distinction between SQL exact match and SQL partial match by explaining **how you can expand your search by using wildcards**. Finally, we'll clarify when you should use something other than `LIKE` to find a match.

How to Use LIKE in SQL?

Suppose you have to retrieve some records based on whether a column contains a certain group of characters. As you know, in SQL the `WHERE` clause filters `SELECT` results. By itself, `WHERE` finds exact matches. But what if you need to find something using a partial match?

In that case, you can use `LIKE` in SQL. This operator searches strings or substrings for specific characters and returns any records that match that pattern. (Hence the *SQL pattern matching*.) Below is the syntax of the **LIKE** operator in a `SELECT` statement:

```
SELECT   [ column_list |  *  ]
FROM  table_name
WHERE  column or expression  LIKE  pattern;
```

Notice that the column name or the expression to be searched comes before **LIKE** in SQL. After the operator is the pattern to match. This pattern can be pure text or text mixed with one or more wildcards. We'll explain the use of wildcards next.

SQL Partial Match: Using LIKE with

Wildcards

If you don't know the exact pattern you're searching for, you can use wildcards to help you find it. **Wildcards** are text symbols that denote how many characters will be in a certain place within the string. The **SQL** ANSI standard uses two wildcards, percent (%) and underscore (_), which are used in different ways. When using wildcards, you perform a SQL **partial match** instead of a SQL exact match as you don't include an exact string in your query.

wildcard	description
%	zero, one, or many characters, including spaces
_	a single character

Look at the complete `animal` table which will be used in our SQL queries:

id	name
1	frog
2	dog
3	bear
4	fox
5	jaguar
6	puma
7	panda
8	lion
9	leopard

id	name
10	sheep
11	camel
12	monkey
13	lemur
14	rabbit
15	hedgehog
16	elephant
17	elephant.. .
18	langur
19	hog
20	gerenuk
21	
22	null

Note: .. . denotes two spaces.

SQL Partial Match: the Percent Wildcard

As you can see in the above table, the **percent wildcard** can be used when you're not sure how many characters will be part of your match. In the example below, notice what happens when you use only this wildcard with `LIKE` in SQL:

```
SELECT
  id,
  name
FROM animal
```

```
WHERE name LIKE '%' ;
```

Result:

id	name
1	frog
2	dog
3	bear
4	fox
5	jaguar
6	puma
7	panda
8	lion
9	leopard
10	sheep
11	camel
12	monkey
13	lemur
14	rabbit
15	hedgehog
16	elephant
17	elephant.. .
18	langur
19	hog
20	gerenuk

id	name
21	

Note: .. . denotes two spaces.

This use of the SQL partial match returns all the names from the `animal` table, even the ones without any characters at all in the name column. This is because the percent wildcard denotes *any character or no characters*. Even when there is a null value in the name column, an empty string is returned.

But if you would like to return only the animal names that start with a "**g**", you should write the query using a "**g**" in front of the percent wildcard:

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE 'g%' ;
```

The result of this SQL partial match operation is the following:

id	name
20	gerenuk

Similarly, if you would like to select the animal names that end with a "**g**", you'd put the percent wildcard first, as shown in this SQL partial match query:

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE '%g';
```

Result:

id	name
1	frog
2	dog
15	hedgehog
19	hog

The following query returns all animals whose name contains a “**g**”. To do this, use two percent wildcards and a “**g**” character, as shown below.

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE '%g%';
```

Result:

id	name
1	frog
2	dog
5	jaguar
15	hedgehog
18	langur
19	hog
20	gerenuk

All these animals have a name that contains a “g” somewhere – at the beginning, in the middle, or at the end.

Now, let's move on to the underscore wildcard.

SQL Partial Match: the Underscore Wildcard

The **underscore wildcard** represents a single character for each underscore. In this SQL partial match, it can replace any character at all, but each underscore is limited to one character. Look at the example below:

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE '_';
```

Result:

id	name

0 rows

This query didn't return any records because there are no single-character animal names in the table.

The next example displays all names that contain exactly five characters. To represent this, we must use five underscores:

```
SELECT
  id,
  name
FROM animal
```

```
WHERE name LIKE '_____';
```

Result:

id	name
7	panda
10	sheep
11	camel
13	lemur

If you use the underscore wildcard at the end of your SQL partial match string, the query will return every record that matches *the given text plus one more character*. Below we see an example:

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE 'lio_';
```

Result:

id	name
8	lion

What is returned when the query has an underscore wildcard in the middle of the string?

```
SELECT
  id,
  name
FROM animal
```

```
WHERE name LIKE 'p_ma';
```

Result:

id	name
6	puma

It is all animals whose names start with “**p**” and end with “**ma**”, with only one character in between.

SQL Partial Match: Combining Wildcards

You can also use a combination of underscore and percent wildcards for your SQL pattern matching. Look at the following example:

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE '%ho_';
```

Result:

id	name
15	hedgehog
19	hog

As you can see, this query returned names that combined "**ho**" with any number of characters in front and only one character following.

UsingLIKE in SQL with Text

Now we will discuss how to use `LIKE` in SQL with text-only strings and no wildcards. In some circumstances, you may find that there are better options than using `LIKE` in SQL pattern matching. But for now, let's see how this works. We'll start by looking at the complete table of animal names and ID numbers, as shown below:

id	name
1	frog
2	dog
3	bear
4	fox
5	jaguar
6	puma
7	panda
8	lion
9	leopard
10	sheep
11	camel

id	name
12	monkey
13	lemur
14	rabbit
15	hedgehog
16	elephant
17	elephant. ..
18	langur
19	hog
20	gerenuk
21	
22	null

Note: . .. denotes two spaces.

Note that the record where `id`=21 has an empty string (without any characters). The last record has a NULL value in the `name` column.

Now, say we want to retrieve the records where the animal's name is *"elephant"*. That's pretty simple, as the example below shows:

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE 'elephant';
```

Result:

id	name
16	elephant

In the table, there are actually **two** records containing “elephant”. However, the second record has an additional two spaces at the end of the word, so it isn’t returned.

Let’s try another text pattern that includes these two spaces.

```
SELECT
  id,
  name
FROM animal
WHERE name LIKE 'elephant  ';
```

Result:

id	name
17	elephant. ..

Note: . .. denotes two spaces.

Again, there is only one record: “elephant” with two spaces.

Next, suppose we use a concrete text string and an equals operator (=), like this:

```
SELECT
  id,
  name
FROM animal
WHERE name = 'elephant '  ;
```

Result:

id	name
16	elephant

If you want to check if a text string is the same as the value of a column, you're looking for a SQL exact match rather than a SQL partial match. In that case, use an equals operator rather than `LIKE`.

Combining NOT and LIKE Operators

You can also test for strings that do not match a pattern. To do this, we combine the `LIKE` and `NOT` operators. It is another way of performing the SQL pattern matching.

In the example below, we want to find all animal names that don't have an "**a**" character:

```
SELECT
  id,
  name
FROM animal
WHERE name NOT LIKE '%a%';
```

Result:

id	name
1	frog

id	name
2	dog
4	fox
8	lion
10	sheep
12	monkey
13	lemur
15	hedgehog
19	hog
20	gerenuk
21	camel

Using LIKE in SQL with Other Operators

The `WHERE` clause can include more than one condition. Therefore, `LIKE` and `NOT LIKE` can be used with other operators. Let's look at another example:

```
SELECT
```

```
  id,
  name
FROM animal
WHERE name LIKE '%g'  OR name LIKE 's%'  ;
```

Result:

id	name
1	frog
2	dog
10	sheep
15	hedgehog
19	hog

It returned all the animal names that start with an “**s**” character or end with a “**g**” character.

Using LIKE in SQL in Other Statements

So far, we’ve discussed using `LIKE` in SQL only in `SELECT` statements. But this operator can be used in other statements, such as `UPDATE` or `DELETE`. As you can see, the syntax is quite similar:

```
UPDATE table
SET column1 = newValue
```

```
WHERE  column2  LIKE  pattern ;
```

```
DELETE  FROM  table
WHERE  column  LIKE  pattern ;
```

Let's see how we can use `LIKE` to change some animal names. Ready?

```
UPDATE  animal
SET name='tiger'
WHERE name LIKE '%key%' ;
```

There is only one record that matches the `LIKE %key%` condition: mon**key**. After this update, "tiger" will replace all instances of "monkey".

Here's the result after we update and then select all records from the `animal` table.

```
SELECT *
FROM  animal ;
```

id	name
1	frog
2	dog
3	bear
4	fox
5	jaguar
6	puma
7	panda
8	lion
9	leopard

id	name
10	sheep
11	camel
12	tiger
13	lemur
14	rabbit
15	hedgehog
16	elephant
17	elephant.. .
18	langur
19	hog
20	gerenuk
21	
22	null

Note: . .. denotes two spaces.

Next, we'll delete any records where the animal name starts with a "**t**":

```
DELETE FROM animal
WHERE name LIKE 't%'  ;
```

To Learn More About SQL Pattern Matching

SQL pattern matching is very useful for searching text substrings. `LIKE` and its close relative `NOT LIKE` make this quite easy to do. If you are interested in learning more about *pattern matching* and the `LIKE` operator, check out the SQL BASICS COURSE. It will show you how to build queries from scratch, but it will also introduce practical skills **like pattern matching matching**.

If you have a basic knowledge of SQL, you can refresh it with the SQL PRACTICE SET of 88 exercises, ranging from simple tasks with `SELECT FROM` statements to more advanced problems involving multiple subqueries. Try both of the courses now for free!