MySQL | Regular expressions (Regexp)

- Difficulty Level : Medium
- Last Updated : 11 Oct, 2019

MySQL supports another type of pattern matching operation based on the regular expressions and the REGEXP operator.

1. It provide a powerful and flexible pattern match that can help us implement power search utilities for our database systems.
2. REGEXP is the operator used when performing regular expression pattern matches. RLIKE is the synonym.
3. It also supports a number of metacharacters which allow more flexibility and control when performing pattern matching.
4. The backslash is used as an escape character. It’s only considered in the pattern match if double backslashes have used.
5. Not case sensitive.

Pattern	What the Pattern matches
*	Zero or more instances of string preceding it
+	One or more instances of strings preceding it
.	Any single character
?	Match zero or one instances of the strings preceding it.
^	caret(^) matches Beginning of string
$	End of string
[abc]	Any character listed between the square brackets
[^abc]	Any character not listed between the square brackets
[A-Z]	match any upper case letter.
[a-z]	match any lower case letter
[0-9]	match any digit from 0 through to 9.
[[:<:]]	matches the beginning of words.
[[:>:]]	matches the end of words.
[:class:]	matches a character class i.e. [:alpha:] to match letters, [:space:] to match white space, [:punct:] is match punctuations and [:upper:] for upper class letters.
p1|p2|p3	Alternation; matches any of the patterns p1, p2, or p3
{n}	n instances of preceding element
{m,n}	m through n instances of preceding element

Examples with explanation :

Attention reader! Don’t stop learning now. Learn SQL for interviews using **SQL Course** by GeeksforGeeks.

- **Match beginning of string(^):**
  Gives all the names starting with ‘sa’.Example- sam,samarth.

  ```
  SELECT name FROM student_tbl WHERE name REGEXP '^sa';
  ```

- **Match the end of a string($):**
  Gives all the names ending with ‘on’.Example – norton,merton.

  ```
  SELECT name FROM student_tbl WHERE name REGEXP 'on$';
  ```

- **Match zero or one instance of the strings preceding it(?):**
  Gives all the titles containing ‘com’.Example – comedy , romantic comedy.

  ```
  SELECT title FROM movies_tbl WHERE title REGEXP 'com?';
  ```

- **matches any of the patterns p1, p2, or p3(p1|p2|p3):**
  Gives all the names containing ‘be’ or ‘ae’.Example – Abel, Baer.

  ```
  SELECT name FROM student_tbl WHERE name REGEXP 'be|ae' ;
  ```

- **Matches any character listed between the square brackets([abc]):**
  Gives all the names containing ‘j’ or ‘z’.Example – Lorentz, Rajs.

  ```
  SELECT name FROM student_tbl WHERE name REGEXP '[jz]' ;
  ```

- **Matches any lower case letter between ‘a’ to ‘z’- ([a-z]) ([a-z] and (.)):**
  Retrieve all names that contain a letter in the range of ‘b’ and ‘g’, followed by any character, followed by the letter ‘a’.Example – Tobias, sewall.

  Matches any single character(.)

  ```
  SELECT name FROM student_tbl WHERE name REGEXP '[b-g].[a]' ;
  ```

- **Matches any character not listed between the square brackets.([^abc]):**
  Gives all the names not containing ‘j’ or ‘z’. Example – nerton, sewall.

  ```
  SELECT name FROM student_tbl WHERE name REGEXP '[^jz]' ;
  ```

- **Matches the end of words[[:>:]]:**
  Gives all the titles ending with character “ack”. Example – Black.

  ```
  SELECT title FROM movies_tbl WHERE REGEXP 'ack[[:>:]]';
  ```

- **Matches the beginning of words[[:<:]]:**
  Gives all the titles starting with character “for”. Example – Forgetting Sarah Marshal.

  ```
  SELECT title FROM movies_tbl WHERE title REGEXP '[[:<:]]for';
  ```

- **Matches a character class[:class:]:**
  i.e [:lower:]- lowercase character ,[:digit:] – digit characters etc.
  Gives all the titles containing alphabetic character only. Example – stranger things, Avengers.

  ```
  SELECT title FROM movies_tbl WHERE REGEXP '[:alpha:]' ;
  ```

https://www.geeksforgeeks.org/mysql-regular-expressions-regexp/

MySQL REGEXP: Search Based On Regular Expressions

Summary*:* in this tutorial, you will learn how to use the MySQL REGEXP operator to perform complex searches based on **regular expressions.**

Introduction to regular expressions

A regular expression is a special string that describes a search pattern. It is a powerful tool that gives you a concise and flexible way to identify strings of text e.g., characters, and words, based on patterns.

For example, you can use regular expressions to search for email, IP address, phone number, social security number, or anything that has a specific pattern.

A regular expression uses its own syntax that can be interpreted by a regular expression processor. A regular expression is widely used in almost platforms from programming languages to databases including MySQL.

The advantage of using regular expression is that you are not limited to search for a string based on a fixed pattern with the percent sign (%) and underscore (_) in the `LIKE` operator. The regular expressions have more meta-characters to construct flexible patterns.

The disadvantage of using regular expression is that it is quite difficult to understand and maintain such a complicated pattern. Therefore, you should describe the meaning of the regular expression in the comment of the SQL statement. In addition, the speed of data retrieval, in some cases, is decreased if you use complex patterns in a regular expression.

The abbreviation of regular expressions is regex or regexp.

MySQL REGEXP operator

MySQL adapts the regular expression implemented by Henry Spencer. MySQL allows you to match pattern right in the SQL statements by using `REGEXP` operator.

The following illustrates the syntax of the `REGEXP` operator in the `WHERE` clause:

```
SELECT
    column_list
FROM
    table_name
WHERE
    string_column REGEXP pattern;
Code language: SQL (Structured Query Language) (sql)
```

This statement performs a pattern match of a `string_column` against a `pattern`.

If a value in the `string_column` matches the `pattern`, the expression in the `WHERE` clause returns true, otherwise it returns false.

If either `string_column` or `pattern` is `NULL`, the result is `NULL`.

In addition to the `REGEXP` operator, you can use the `RLIKE` operator, which is the synonym of the `REGEXP` operator.

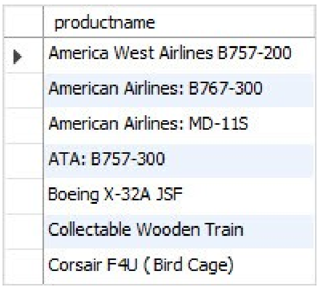
The negation form of the `REGEXP` operator is `NOT REGEXP`.

MySQL REGEXP examples

Suppose you want to find all products whose last names start with character A, B or C. You can use a regular expression in the following `SELECT` statement:

```
SELECT
    productname
FROM
    products
WHERE
    productname REGEXP '^(A|B|C)'
ORDER BY productname;
Code language: SQL (Structured Query Language) (sql)
```

Try It Out



	productname
▶	America West Airlines B757-200
	American Airlines: B767-300
	American Airlines: MD-11S
	ATA: B757-300
	Boeing X-32A JSF
	Collectable Wooden Train
	Corsair F4U (Bird Cage)

The pattern allows you to find the product whose name begins with A, B, or C.

- The character ^ means to match from the beginning of the string.
- The character | means to search for alternatives if one fails to match.

The following table illustrates some commonly used metacharacters and constructs in a regular expression.

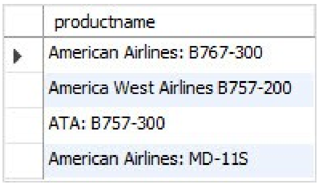
Metacharacter	Behavior
^	matches the position at the beginning of the searched string
$	matches the position at the end of the searched string
.	matches any single character
[…]	matches any character specified inside the square brackets
[^…]	matches any character not specified inside the square brackets
p1|p2	matches any of the patterns p1 or p2
*	matches the preceding character zero or more times
+	matches preceding character one or more times
{n}	matches n number of instances of the preceding character
{m,n}	matches from m to n number of instances of the preceding character

To find products whose names start with the character `a`, you use the metacharacter `'^'` to match at the beginning of the name:

```
SELECT
```

```
    productname
FROM
    products
WHERE
    productname REGEXP '^a';
Code language: SQL (Structured Query Language) (sql)
```

Try It Out



productname
American Airlines: B767-300
America West Airlines B757-200
ATA: B757-300
American Airlines: MD-11S

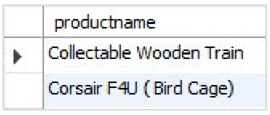
If you want the `REGEXP` operator to compare strings in case-sensitive fashion, you can use the `BINARY` operator to cast a string to a binary string.

Because MySQL compares binary strings byte by byte rather than character by character. This allows the string comparison to be case sensitive.

For example, the following statement matches only uppercase `"C"` at the beginning of the product name.

```
SELECT
    productname
FROM
    products
WHERE
    productname REGEXP BINARY '^C';
Code language: SQL (Structured Query Language) (sql)
```

Try It Out

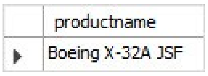


productname
Collectable Wooden Train
Corsair F4U (Bird Cage)

To find the product whose name ends with f, you use `'f$'` to match the end of a string.

```
SELECT
    productname
FROM
    products
WHERE
    productname REGEXP 'f$'
Code language: SQL (Structured Query Language) (sql)
```

Try It Out



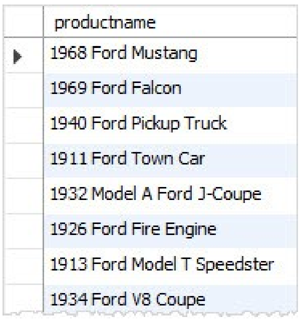
productname
Boeing X-32A JSF

To find the product whose name contains the word `"ford"`, you use the following query:

```
SELECT
    productname
FROM
    products
WHERE
```

```
    productname REGEXP 'ford';
Code language: SQL (Structured Query Language) (sql)
```

Try It Out

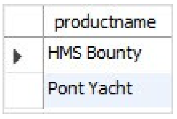


	productname
▶	1968 Ford Mustang
	1969 Ford Falcon
	1940 Ford Pickup Truck
	1911 Ford Town Car
	1932 Model A Ford J-Coupe
	1926 Ford Fire Engine
	1913 Ford Model T Speedster
	1934 Ford V8 Coupe

To find the product whose name contains exactly 10 characters, you use '`^'` and '`$` to match the beginning and end of the product name, and repeat `{10}` times of any character '`.'` in between as shown in the following query:

```
SELECT
    productname
FROM
    products
WHERE
    productname REGEXP '^.{10}$';
Code language: SQL (Structured Query Language) (sql)
```

Try It Out



	productname
▶	HMS Bounty
	Pont Yacht

In this tutorial, you have learned how to query data using the MySQL REGEXP operator with regular expressions.

https://www.mysqltutorial.org/mysql-regular-expression-regexp.aspx

MYSQL Regular Expressions (REGEXP) with Syntax & Examples

ByRichard Peterson
Updated

What are regular expressions?

Regular Expressions help search data matching complex criteria. We looked at wildcards in the previous tutorial. If you have worked with wildcards before, you may be asking why learn regular expressions when you can get similar results using the wildcards. Because, compared to wildcards, regular expressions allow us to search data matching even more complex criterion.

Basic syntax

The basic syntax for a regular expression is as follows

```
SELECT statements... WHERE fieldname REGEXP 'pattern';
```

HERE –

- **"SELECT statements…"** is the standard SELECT statement
- **"WHERE fieldname"** is the name of the column on which the regular expression is to be performed on.
- **"REGEXP 'pattern'"** REGEXP is the regular expression operator and 'pattern' represents the pattern to be matched by REGEXP. **RLIKE** is the **synonym for REGEXP** and achieves the same results as REGEXP. To avoid confusing it with the LIKE operator, it **better to use REGEXP** instead.

Let's now look at a practical example-

```
SELECT * FROM `movies` WHERE `title` REGEXP 'code';
```

The above query searches for all the movie titles that have the word code in them. It does not matter whether the "code" is at the beginning, middle or end of the title. As long as it is contained in the title then it will be considered.

Let's suppose that we want to search for movies that start with a, b, c or d , followed by any number of other characters, how would we go about to achieve that. We can use a regular expression together with the metacharacters to achieve our desired results.

```
SELECT * FROM `movies` WHERE `title` REGEXP '^[abcd]';
```

Executing the above script in MySQL workbench against the myflixdb gives us the following results.

movie_id	title	director	year_released	category_id
4	Code Name Black	Edgar Jimz	2010	NULL
5	Daddy's Little Girls	NULL	2007	8
6	Angels and Demons	NULL	2007	6
7	Davinci Code	NULL	2007	6

Let’s now take a close look at our regular expression responsible for the above result.

‘^[abcd]’ the caret (^) means that the pattern match should be applied at the beginning and the charlist [abcd] means that only movie titles that start with a, b, c or d are returned in our result set.

Let’s modify our above script and use the NOT charlist and see what results we will get after executing our query.

```
SELECT * FROM `movies` WHERE `title` REGEXP '^[^abcd]';
```

Executing the above script in MySQL workbench against the myflixdb gives us the following results.

movie_id	title	director	year_released	category_id
1	Pirates of the Caribean 4	Rob Marshall	2011	1
2	Forgetting Sarah Marshal	Nicholas Stoller	2008	2
3	X-Men		2008	
9	Honey mooners	John Schultz	2005	8
16	67% Guilty		2012	
17	The Great Dictator	Chalie Chaplie	1920	7
18	sample movie	Anonymous		8
19	movie 3	John Brown	1920	8

Let’s now take a close look at our regular expression responsible for the above results.

‘^[^abcd]’ the caret (^) means that the pattern match should be applied at the beginning and the charlist [^abcd] means that the movie titles starting with any of the enclosed characters is excluded from the result set.

Regular expression metacharacters

What we looked at in the above example is the simplest form of a regular expression. Let’s now look at more advanced regular expression pattern matches. Suppose we want to search for movie titles that start with the pattern “code” only using a regular expression, how would we go about it? The answer is metacharacters. They allow us to fine tune our pattern search results using regular expressions.

Char	Description	Example
*	The **asterisk (*)** metacharacter is used to match zero (0) or more instances of the strings preceding it	*SELECT * FROM movies WHERE title REGEXP ‘da*’;* will give all movies containing characters “da” .For Example, Da Vinci Code , Daddy’s Little Girls.
+	The **plus (+)** metacharacter is used to match one or more instances of strings preceding it.	*SELECT * FROM `movies` WHERE `title` REGEXP ‘mon+’;* will give all movies containing characters “mon” .For Example, Angels and Demons.
?	The question**(?)** metacharacter is used to match zero (0) or one instances of the strings preceding it.	*SELECT * FROM `categories` WHERE `category_name` REGEXP ‘com?’;* will give all the categories containing string com .For Example, comedy , romantic comedy .
.	The **dot (.)** metacharacter is used to match any single character in exception of a new line.	*SELECT * FROM movies WHERE `year_released` REGEXP ‘200.’;* will give all the movies released in the years starting with characters “200” followed by any single character .For Example, 2005,2007,2008 etc.
[abc]	The **charlist [abc]** is used to match any of the enclosed characters.	*SELECT * FROM `movies` WHERE `title` REGEXP ‘[vwxyz]’;* will give all the movies containing any single character in “vwxyz” .For Example, X-Men, Da Vinci Code, etc.
[^abc]	The **charlist [^abc]** is used to match any characters excluding the ones enclosed.	*SELECT * FROM `movies` WHERE `title` REGEXP ‘^[^vwxyz]’;* will give all the movies containing characters other than the ones in “vwxyz”.
[A-Z]	The **[A-Z]** is used to match any upper case letter.	*SELECT * FROM `members` WHERE `postal_address` REGEXP ‘[A-Z]’;* will give all the members that have postal address containing any character from A to Z. .For Example, Janet Jones with membership number 1.
[a-z]	The **[a-z]** is used to match any lower case letter	*SELECT * FROM `members` WHERE `postal_address` REGEXP ‘[a-z]’;* will give all the members that have postal addresses containing any character from a to z. .For Example, Janet Jones with membership number 1.
[0-9]	The **[0-9]** is used to match any digit from 0 through to 9.	*SELECT * FROM `members` WHERE `contact_number` REGEXP ‘[0-9]’* will give all the members have submitted contact numbers containing

		characters “[0-9]” .For Example, Robert Phil.
^	The **caret (^)** is used to start the match at beginning.	_SELECT * FROM `movies` WHERE `title` REGEXP ‘^[cd]’;_ gives all the movies with the title starting with any of the characters in “cd” .For Example, Code Name Black, Daddy’s Little Girls and Da Vinci Code.
|	The **vertical bar (|)** is used to isolate alternatives.	_SELECT * FROM `movies` WHERE `title` REGEXP ‘^[cd]|^[u]’;_ gives all the movies with the title starting with any of the characters in “cd” or “u” .For Example, Code Name Black, Daddy’s Little Girl, Da Vinci Code and Underworld – Awakening.
[[:<:]]	The**[[:<:]]** matches the beginning of words.	_SELECT * FROM `movies` WHERE `title` REGEXP ‘[[:<:]]for’;_ gives all the movies with titles starting with the characters. For Example: Forgetting Sarah Marshal.
[[:>:]]	The **[[:>:]]** matches the end of words.	_SELECT * FROM `movies` WHERE `title` REGEXP ‘ack[[:>:]]’;_ gives all the movies with titles ending with the characters “ack” .For Example, Code Name Black.
[:class:]	The **[:class:]** matches a character class i.e. _[:alpha:]_ to match letters, _[:space:]_ to match white space, _[:punct:]_ is match punctuations and [:upper:] for upper class letters.	_SELECT * FROM `movies` WHERE `title` REGEXP ‘[:alpha:]’;_ gives all the movies with titles contain letters only .For Example, Forgetting Sarah Marshal, X-Men etc. Movie like Pirates of the Caribbean 4 will be omitted by this query.

The backslash (\) is used to as an escape character. If we want to use it as part of the pattern in a regular expression, we should use double backslashes (\\)

Summary

- Regular expressions provide a powerful and flexible pattern match that can help us implement power search utilities for our database systems.
- REGEXP is the operator used when performing regular expression pattern matches. RLIKE is the synonym
- Regular expressions support a number of metacharacters which allow for more flexibility and control when performing pattern matches.
- The backslash is used as an escape character in regular expressions. It’s only considered in the pattern match if double backslashes have used.
- Regular expressions are not case sensitive.

https://www.guru99.com/regular-expressions.html