

MYSQL Regular Expressions (REGEXP) with Syntax & Examples

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 criteria.

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.
- **"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 Caribbean 4	Rob Marshall	2011	1
2	Forgetting Sarah Marshall	Nicholas Stoller	2008	2
3	X-Men		2008	
9	Honey mooners	John Schultz	2005	8
16	67% Guilty		2012	
17	The Great Dictator	Charlie 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 are 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	<i>SELECT * FROM movies WHERE title REGEXP 'da*';</i> 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.	<i>SELECT * FROM `movies` WHERE `title` REGEXP 'mon+';</i> will give all movies containing characters "mon" .For Example, Angels and Demons.
?	The question(?) metacharacter is used to match zero (0) or one instance of the strings preceding it.	<i>SELECT * FROM `categories` WHERE `category_name` REGEXP 'com?';</i> 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.	<i>SELECT * FROM movies WHERE `year_released` REGEXP '200.';</i> 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.	<i>SELECT * FROM `movies` WHERE `title` REGEXP '[vwxyz]';</i> 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.	<i>SELECT * FROM `movies` WHERE `title` REGEXP '^[^vwxyz]';</i> will give all the movies containing characters other than the ones in "vwxyz".
[A-Z]	The [A-Z] is used to match any uppercase letter.	<i>SELECT * FROM `members` WHERE `postal_address` REGEXP '[A-Z]';</i> will give all the members that have a 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 lowercase letter	<i>SELECT * FROM `members` WHERE `postal_address` REGEXP '[a-z]';</i> 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.	<i>SELECT * FROM `members` WHERE `contact_number` REGEXP '[0-9]'</i> will give all the members who have submitted contact numbers containing characters "[0-9]" .For Example, Robert Phil.
^	The caret (^) is used to start the match at the beginning.	<i>SELECT * FROM `movies` WHERE `title` REGEXP '^[cd]'</i> ; 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.	<i>SELECT * FROM `movies` WHERE `title` REGEXP '^[cd] [u]'</i> ; 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.	<i>SELECT * FROM `movies` WHERE `title` REGEXP '[:<:]for'</i> ; gives all the movies with titles starting with the characters. For Example: Forgetting Sarah Marshall.
[:>:]	The [:>:] matches the end of words.	<i>SELECT * FROM `movies` WHERE `title` REGEXP 'ack[:>:]'</i> ; 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. <i>[[:alpha:]]</i> to match letters, <i>[[:space:]]</i> to match white space, <i>[[:punct:]]</i> is match punctuations and <i>[[:upper:]]</i> for upper class letters.	<i>SELECT * FROM `movies` WHERE `title` REGEXP '[:alpha:]';</i> gives all the movies with titles containing letters only .For Example, Forgetting Sarah Marshall, X-Men etc. Movies like Pirates of the Caribbean 4 will be omitted by this query.
--------------------	--	--

The backslash (\) is used 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 are used.
- Regular expressions are not case sensitive.