Doc 10 Regex in Mongodb

MongoDB Regex

MongoDB provides the functionality to search a pattern in a string during a query by writing a regular expression. A regular expression is a generalized way to match patterns with sequences of characters.

MongoDB uses Perl-compatible regular expressions(PCRE) version 8.42 along with UTF-8 support. In MongoDB we can do pattern matching in two different ways:

With $regex Operator

Without $regex Operator

Pattern Matching Using $regex Operator

This operator provides regular expression capabilities for pattern-matching stings in the queries. I

In other words, this operator is used to search for the given string in the specified collection.

t is helpful when we don't know the exact field value that we are looking in the document.

For example, a collection containing 3 documents i.e.,

{

name: "Tony",

position: "Backend developer"

}

{

name: "Bruce",

position: "frontend developer"

}

{

name: "Nick",

position: "HR Manager"

}

and we are looking for developer information. So, with the help of the $regex operator, we create a pattern(i.e., {position: {$regex: "developer"}}) that will return only those documents that contain developer string.

MongoDB allows querying documents with regular expressions (Regex) for advanced search operations.

Important Points:

You are not allowed to use $regex operator inside $in operator.

If you want to add a regular expression inside a comma-separated list of a query condition, then you have to use the $regex operator.

If you want to use x and s options then you have to use $regex operator expression with $options.

Starting from the latest version of MongoDB(i.e., 4.0.7) you are allowed to use $not operator with $regex operator expression.

For case-sensitive regular expression queries, if the index of the specified field is available, then MongoDB matches the regular expression to the values in the index. It is the easiest way to match rather than scanning all the collections For case-insensitive regular expression queries, they do not utilize index effectively.

If you want to use Perl compatible regular expressions support regular expressions that are not supported in JavaScript, then you must use $regex operator.

Syntax:

{ <field>: { $regex: /pattern/, $options: '<options>' } }

{ <field>: { $regex: 'pattern', $options: '<options>' } }

$options:

In MongoDB, the following <options> are available for use with regular expression:

i: To match both lower case and upper case pattern in the string.

m: To include ^ and $ in the pattern in the match i.e. to specifically search for ^ and $ inside the string. Without this option, these anchors match at the beginning or end of the string.

x: To ignore all white space characters in the $regex pattern.

s: To allow the dot character "." to match all characters including newline characters.

Examples of MongoDB Regex

In the following examples, we are working with:

Database: GeeksforGeeks

Collection: employee

Documents: Six documents that contain the details of the employees in the form of field-value pairs.

Output:

Example 1: Displaying details of employee who are having the word developer in their position field

db.employee.find({position : {$regex : "developer"}}).pretty()

Output:

Here, we are displaying the documents of those employees whose position field contain developer string. So we pass a regular expression using $regex operator(i.e. {$regex : "developer"}) for the position field in the find() method.

Example 2: Displaying details of employee who are a software engineer with case insensitive by using i <options>

db.employee.find({position:{$regex:"software",$options:"i"}}).pretty()

Output:

Here, we are displaying the documents of those employees whose position field contain case-insensitive "software" string. So, we pass a regular expression with option(i.e., {$regex : "software", $options: "$i"}) for the position field in the find() method. In the regular expression, $options:"$i" is used to match both lower case and upper case pattern in the given string(i.e., "software").

Example 3: Displaying details of the employee whose name starts with B

db.employee.find({Name:{$regex:"^B"}}).pretty()

Output:

Here, we are displaying the documents of those employees whose name starts with 'B' letter. So, we pass a regular expression using $regex operator(i.e. {$regex : "^B"}) for the Name field in the find() method.

Example 4: Displaying details of the employee whose name ends with e:

db.employee.find({Name:{$regex:"e$"}}).pretty()

Output:

Here, we are displaying the documents of those employees whose names end with the 'e' letter. So, we pass a regular expression using $regex operator(i.e. {$regex : "e$"}) for the Name field in the find() method.

Pattern matching without using $regex operator

In MongoDB, we can do pattern matching without using the $regex operator. Simply, by using a regular expression object to specify a regular expression. Also, by using regular expression object you are allowed to use regular expression inside $ in operator.

Syntax:

{ <field>: /pattern/<options> }

Here, // means to specify your search criteria in between these delimiters.

Example:

In the following examples, we are working with:

Database: GeeksforGeeks

Collection: employee

Documents: Six documents that contain the details of the employees in the form of field-value pairs.

Output:

Example 1: Displaying details of employee who are having the word "te" in their name by using regular expression object

db.employee.find({Name: /te/}).pretty()

Output:

Displaying-details-of-employee-who-are-having-the-word-TE

Here, we are displaying the documents of those employees whose names contain the "te" string. So, we pass a regular expression(i.e., {Name: /te/}) for the Name field in the find() method. In this regular expression, // means to specify your search criteria in between these delimiters, i.e., /te/.