$text

On this page

Definition

`$text`

`$text` performs a text search on the content of the fields indexed with a text index. A `$text` expression has the following syntax:

Changed in version 3.2.

```
{
  $text:
    {
      $search: <string>,
      $language: <string>,
      $caseSensitive: <boolean>,
      $diacriticSensitive: <boolean>
    }
}
```

The `$text` operator accepts a text query document with the following fields:

Field	Type	Description
`$search`	string	A string of terms that MongoDB parses and uses to query the text index. MongoDB performs a logical `OR` search of the terms unless specified as a phrase. See Behavior for more information on the field.
`$language`	string	Optional. The language that determines the list of stop words for the search and the rules for the stemmer and tokenizer. If not specified, the search uses the default language of the index. For supported languages, see Text Search Languages. If you specify a language value of `"none"`, then the text search uses simple tokenization with no list of stop words and no stemming.
`$caseSensitive`	boolean	Optional. A boolean flag to enable or disable case sensitive search. Defaults to `false`; i.e. the search defers to the case insensitivity of the text index. For more information, see Case Insensitivity. *New in version 3.2.*
`$diacriticSensitive`	boolean	Optional. A boolean flag to enable or disable diacritic sensitive search against version 3 text indexes. Defaults to `false`; i.e. the search defers to the diacritic insensitivity of the text index.

Text searches against earlier versions of the text index are inherently diacritic sensitive and cannot be diacritic insensitive. As such, the `$diacriticSensitive` option has no effect with earlier versions of the text index.

For more information, see Diacritic Insensitivity.

New in version 3.2.

The `$text` operator, by default, does *not* return results sorted in terms of the results' scores. For more information on sorting by the text search scores, see the Text Score documentation.

Behavior

Restrictions

- A query can specify, at most, one `$text` expression.
- The `$text` query can not appear in `$nor` expressions.
- To use a `$text` query in an `$or` expression, all clauses in the `$or` array must be indexed.
- You cannot use `hint()` if the query includes a `$text` query expression.
- You cannot specify `$natural` sort order if the query includes a `$text` expression.
- You cannot combine the `$text` expression, which requires a special text index, with a query operator that requires a different type of special index. For example you cannot combine `$text` expression with the `$near` operator.

If using the `$text` operator in aggregation, the following restrictions also apply.

- The `$match` stage that includes a `$text` must be the **first** stage in the pipeline.
- A `text` operator can only occur once in the stage.
- The `text` operator expression cannot appear in `$or` or `$not` expressions.
- The text search, by default, does not return the matching documents in order of matching scores. Use the `$meta` aggregation expression in the `$sort` stage.

`$search` Field

In the `$search` field, specify a string of words that the `text` operator parses and uses to query the text index.

The `text` operator treats most punctuation in the string as delimiters, except a hyphen-minus (`-`) that negates term or an escaped double quotes `\"` that specifies a phrase.

Phrases

To match on a phrase, as opposed to individual terms, enclose the phrase in escaped double quotes (`\"`), as in:

```
"\"ssl certificate\""
```

If the `$search` string includes a phrase and individual terms, text search will only match the documents that include the phrase. More specifically, the search performs a logical `AND` of the phrase with the individual terms in the search string.

For example, passed a `$search` string:

```
"\"ssl certificate\" authority key"
```

The `$text` operator searches for the phrase `"ssl certificate"` **and** (`"authority"` **or** `"key"` **or** `"ssl"` **or** `"certificate"`).

Negations
Prefixing a word with a hyphen-minus (`-`) negates a word:

- The negated word excludes documents that contain the negated word from the result set.
- When passed a search string that only contains negated words, text search will not match any documents.
- A hyphenated word, such as `pre-market`, is not a negation. The `$text` operator treats the hyphen-minus (`-`) as a delimiter.

The `$text` operator adds all negations to the query with the logical `AND` operator.

Match Operation

Stop Words
The `$text` operator ignores language-specific stop words, such as `the` and `and` in English.

Stemmed Words
For case insensitive and diacritic insensitive text searches, the `$text` operator matches on the complete *stemmed* word. So if a document field contains the word `blueberry`, a search on the term `blue` will not match. However, `blueberry` or `blueberries` will match.

Case Sensitive Search and Stemmed Words

For case sensitive search (i.e. `$caseSensitive: true`), if the suffix stem contains uppercase letters, the `$text` operator matches on the exact word.

Diacritic Sensitive Search and Stemmed Words

For diacritic sensitive search (i.e. `$diacriticSensitive: true`), if the suffix stem contains the diacritic mark or marks, the `$text` operator matches on the exact word.

Case Insensitivity

Changed in version 3.2.

The `$text` operator defaults to the case insensitivity of the text index:

- The version 3 text index is case insensitive for Latin characters with or without diacritics and characters from non-Latin alphabets, such as the Cyrillic alphabet. See text index for details.
- Earlier versions of the `text` index are case insensitive for Latin characters without diacritic marks; i.e. for `[A-z]`.

`$caseSensitive` Option
To support case sensitive search where the `text` index is case insensitive, specify `$caseSensitive: true`.

Case Sensitive Search Process

When performing a case sensitive search (`$caseSensitive: true`) where the `text` index is case insensitive, the `$text` operator:

- First searches the `text` index for case insensitive and diacritic matches.
- Then, to return just the documents that match the case of the search terms, the `$text` query operation includes an additional stage to filter out the documents that do not match the specified case.

For case sensitive search (i.e. `$caseSensitive: true`), if the suffix stem contains uppercase letters, the `$text` operator matches on the exact word.

Specifying `$caseSensitive: true` may impact performance.

SEE ALSO:
Stemmed Words

Diacritic Insensitivity

Changed in version 3.2.

The `$text` operator defaults to the diacritic insensitivity of the text index:

- The version 3 text index is diacritic insensitive. That is, the index does not distinguish between characters that contain diacritical marks and their non-marked counterpart, such as `é`, `ê`, and `e`.
- Earlier versions of the `text` index are diacritic sensitive.

`$diacriticSensitive` Option

To support diacritic sensitive text search against the version 3 `text` index, specify `$diacriticSensitive: true`.

Text searches against earlier versions of the `text` index are inherently diacritic sensitive and cannot be diacritic insensitive. As such, the `$diacriticSensitive` option for the `$text` operator has no effect with earlier versions of the `text` index.

Diacritic Sensitive Search Process

To perform a diacritic sensitive text search (`$diacriticSensitive: true`) against a version 3 `text` index, the `$text` operator:

- First searches the `text` index, which is diacritic insensitive.
- Then, to return just the documents that match the diacritic marked characters of the search terms, the `$text` query operation includes an additional stage to filter out the documents that do not match.

Specifying `$diacriticSensitive: true` may impact performance.

To perform a diacritic sensitive search against an earlier version of the `text` index, the `$text` operator searches the `text` index which is diacritic sensitive.

For diacritic sensitive search, if the suffix stem contains the diacritic mark or marks, the `$text` operator matches on the exact word.

SEE ALSO:
Stemmed Words

Text Score

The `$text` operator assigns a score to each document that contains the search term in the indexed fields. The score represents the relevance of a document to a given text search query. The score can be part of a `sort()` method specification as well as part of the projection expression. The `{ $meta: "textScore" }` expression provides information on the processing of the `$text` operation. See `$meta` projection operator for details on accessing the score for projection or sort.

Examples

The following examples assume a collection `articles` that has a version 3 text index on the field `subject`:

```
db.articles.createIndex( { subject: "text" } )
```

Populate the collection with the following documents:

```
db.articles.insert(
   [
     { _id: 1, subject: "coffee", author: "xyz", views: 50 },
     { _id: 2, subject: "Coffee Shopping", author: "efg", views: 5 },
     { _id: 3, subject: "Baking a cake", author: "abc", views: 90  },
     { _id: 4, subject: "baking", author: "xyz", views: 100 },
     { _id: 5, subject: "Café Con Leche", author: "abc", views: 200 },
     { _id: 6, subject: "Сырники", author: "jkl", views: 80 },
     { _id: 7, subject: "coffee and cream", author: "efg", views: 10 },
     { _id: 8, subject: "Cafe con Leche", author: "xyz", views: 10 }
   ]
)
```

Search for a Single Word

The following query specifies a `$search` string of `coffee`:

```
db.articles.find( { $text: { $search: "coffee" } } )
```

This query returns the documents that contain the term `coffee` in the indexed `subject` field, or more precisely, the stemmed version of the word:

```
{ "_id" : 2, "subject" : "Coffee Shopping", "author" : "efg", "views" : 5 }
{ "_id" : 7, "subject" : "coffee and cream", "author" : "efg", "views" : 10 }
{ "_id" : 1, "subject" : "coffee", "author" : "xyz", "views" : 50 }
```

SEE ALSO:

Case Insensitivity, Stemmed Words

Match Any of the Search Terms

If the search string is a space-delimited string, `$text` operator performs a logical `OR` search on each term and returns documents that contains any of the terms.

The following query specifies a `$search` string of three terms delimited by space, `"bake coffee cake"`:

```
db.articles.find( { $text: { $search: "bake coffee cake" } } )
```

This query returns documents that contain either `bake` or `coffee` or `cake` in the indexed `subject` field, or more precisely, the stemmed version of these words:

```
{ "_id" : 2, "subject" : "Coffee Shopping", "author" : "efg", "views" : 5 }
{ "_id" : 7, "subject" : "coffee and cream", "author" : "efg", "views" : 10 }
{ "_id" : 1, "subject" : "coffee", "author" : "xyz", "views" : 50 }
{ "_id" : 3, "subject" : "Baking a cake", "author" : "abc", "views" : 90 }
{ "_id" : 4, "subject" : "baking", "author" : "xyz", "views" : 100 }
```

SEE ALSO:

Case Insensitivity, Stemmed Words

Search for a Phrase

To match the exact phrase as a single term, escape the quotes.

The following query searches for the phrase `coffee shop`:

```
db.articles.find( { $text: { $search: "\"coffee shop\"" } } )
```

This query returns documents that contain the phrase `coffee shop`:

```
{ "_id" : 2, "subject" : "Coffee Shopping", "author" : "efg", "views" : 5 }
```

SEE ALSO:

Phrases

Exclude Documents That Contain a Term

A *negated* term is a term that is prefixed by a minus sign `-`. If you negate a term, the `$text` operator will exclude the documents that contain those terms from the results.

The following example searches for documents that contain the words `coffee` but do **not** contain the term `shop`, or more precisely the stemmed version of the words:

```
db.articles.find( { $text: { $search: "coffee -shop" } } )
```

The query returns the following documents:

```
{ "_id" : 7, "subject" : "coffee and cream", "author" : "efg", "views" : 10 }
{ "_id" : 1, "subject" : "coffee", "author" : "xyz", "views" : 50 }
```

SEE ALSO:

Negations, Stemmed Words

Search a Different Language

Use the optional `$language` field in the `$text` expression to specify a language that determines the list of stop words and the rules for the stemmer and tokenizer for the search string.

If you specify a language value of `"none"`, then the text search uses simple tokenization with no list of stop words and no stemming.

The following query specifies `es`, i.e. Spanish, as the language that determines the tokenization, stemming, and stop words:

```
db.articles.find(
   { $text: { $search: "leche", $language: "es" } }
)
```

The query returns the following documents:

```
{ "_id" : 5, "subject" : "Café Con Leche", "author" : "abc", "views" : 200 }
{ "_id" : 8, "subject" : "Cafe con Leche", "author" : "xyz", "views" : 10 }
```

The `$text` expression can also accept the language by name, `spanish`. See Text Search Languages for the supported languages.

SEE ALSO:
Case Insensitivity

Case and Diacritic Insensitive Search

Changed in version 3.2.

The `$text` operator defers to the case and diacritic insensitivity of the `text` index. The version 3 `text` index is diacritic insensitive and expands its case insensitivity to include the Cyrillic alphabet as well as characters with diacritics. For details, see text Index Case Insensitivity and text Index Diacritic Insensitivity.

The following query performs a case and diacritic insensitive text search for the terms `сы́рники` or `CAFÉS`:

```
db.articles.find( { $text: { $search: "сы́рники CAFÉS" } } )
```

Using the version 3 `text` index, the query matches the following documents.

```
{ "_id" : 6, "subject" : "Сырники", "author" : "jkl", "views" : 80 }
{ "_id" : 5, "subject" : "Café Con Leche", "author" : "abc", "views" : 200 }
{ "_id" : 8, "subject" : "Cafe con Leche", "author" : "xyz", "views" : 10 }
```

With the previous versions of the `text` index, the query would not match any document.

SEE ALSO:
Case Insensitivity, Diacritic Insensitivity, Stemmed Words, Text Indexes

Perform Case Sensitive Search

Changed in version 3.2.

To enable case sensitive search, specify `$caseSensitive: true`. Specifying `$caseSensitive: true` may impact performance.

Case Sensitive Search for a Term

The following query performs a case sensitive search for the term `Coffee`:

```
db.articles.find( { $text: { $search: "Coffee", $caseSensitive: true } } )
```

The search matches just the document:

```
{ "_id" : 2, "subject" : "Coffee Shopping", "author" : "efg", "views" : 5 }
```

SEE ALSO:

Case Insensitivity, Case Sensitive Search and Stemmed Words

Case Sensitive Search for a Phrase

The following query performs a case sensitive search for the phrase `Café Con Leche`:

```
db.articles.find( {
   $text: { $search: "\"Café Con Leche\"", $caseSensitive: true }
} )
```

The search matches just the document:

```
{ "_id" : 5, "subject" : "Café Con Leche", "author" : "abc", "views" : 200 }
```

SEE ALSO:

Case Sensitive Search and Stemmed Words, Case Insensitivity

Case Sensitivity with Negated Term

A *negated* term is a term that is prefixed by a minus sign `-`. If you negate a term, the `$text` operator will exclude the documents that contain those terms from the results. You can also specify case sensitivity for negated terms.

The following example performs a case sensitive search for documents that contain the word `Coffee` but do **not** contain the lower-case term `shop`, or more precisely the stemmed version of the words:

```
db.articles.find( { $text: { $search: "Coffee -shop", $caseSensitive: true } } )
```

The query matches the following document:

```
{ "_id" : 2, "subject" : "Coffee Shopping", "author" : "efg" }
```

SEE ALSO:

Case Sensitive Search and Stemmed Words, Negations

Diacritic Sensitive Search

Changed in version 3.2.

To enable diacritic sensitive search against a version 3 text index, specify `$diacriticSensitive: true`. Specifying `$diacriticSensitive: true` may impact performance.

Diacritic Sensitive Search for a Term

The following query performs a diacritic sensitive text search on the term `CAFÉ`, or more precisely the stemmed version of the word:

```
db.articles.find( { $text: { $search: "CAFÉ", $diacriticSensitive: true } } )
```

The query only matches the following document:

```
{ "_id" : 5, "subject" : "Café Con Leche", "author" : "abc" }
```

SEE ALSO:
Diacritic Sensitive Search and Stemmed Words, Diacritic Insensitivity, Case Insensitivity

Diacritic Sensitivity with Negated Term

The `$diacriticSensitive` option applies also to negated terms. A negated term is a term that is prefixed by a minus sign `-`. If you negate a term, the `$text` operator will exclude the documents that contain those terms from the results.

The following query performs a diacritic sensitive text search for document that contains the term `leches` but not the term `cafés`, or more precisely the stemmed version of the words:

```
db.articles.find(
  { $text: { $search: "leches -cafés", $diacriticSensitive: true } }
)
```

The query matches the following document:

```
{ "_id" : 8, "subject" : "Cafe con Leche", "author" : "xyz" }
```

SEE ALSO:
Diacritic Sensitive Search and Stemmed Words, Diacritic Insensitivity, Case Insensitivity

Return the Text Search Score

The following query searches for the term `cake` and returns the score assigned to each matching document:

```
db.articles.find(
   { $text: { $search: "cake" } },
   { score: { $meta: "textScore" } }
)
```

The returned document includes an *additional* field `score` that contains the document's score associated with the text search. [1]

SEE ALSO:

Text Score

Sort by Text Search Score

To sort by the text score, include the **same** `$meta` expression in **both** the projection document and the sort expression. [1] The following query searches for the term `coffee` and sorts the results by the descending score:

```
db.articles.find(
   { $text: { $search: "coffee" } },
   { score: { $meta: "textScore" } }
).sort( { score: { $meta: "textScore" } } )
```

The query returns the matching documents sorted by descending score.

SEE ALSO:

Text Score

Return Top 2 Matching Documents

Use the `limit()` method in conjunction with a `sort()` to return the top `n` matching documents.

The following query searches for the term `coffee` and sorts the results by the descending score, limiting the results to the top two matching documents:

```
db.articles.find(
   { $text: { $search: "coffee" } },
   { score: { $meta: "textScore" } }
).sort( { score: { $meta: "textScore" } } ).limit(2)
```

SEE ALSO:

Text Score

Text Search with Additional Query and Sort Expressions

The following query searches for documents where the `author` equals `"xyz"` and the indexed field `subject` contains the terms `coffee` or `bake`. The operation also specifies a sort order of ascending `_id`, then descending text search score:

```
db.articles.find(
   { author: "xyz", $text: { $search: "coffee bake" } },
   { score: { $meta: "textScore" } }
).sort( { date: 1, score: { $meta: "textScore" } } )
```

SEE ALSO:

Text Search in the Aggregation Pipeline

[1] *(1, 2)* The behavior and requirements of the `$meta` operator differs from that of the `$meta` aggregation operator. See the `$meta` aggregation operator for details.