**$and**

On this page

* [Examples](https://docs.mongodb.com/manual/reference/operator/query/and/#examples)

$and

*Syntax*: { $and: [ { <expression1> }, { <expression2> } , ... , { <expressionN> } ] }

$and performs a logical AND operation on an array of *two or more* expressions (e.g. <expression1>, <expression2>, etc.) and selects the documents that satisfy *all* the expressions in the array. The $and operator uses *short-circuit evaluation*. If the first expression (e.g. <expression1>) evaluates to false, MongoDB will not evaluate the remaining expressions.

Note

MongoDB provides an implicit AND operation when specifying a comma separated list of expressions. Using an explicit AND with the $and operator is necessary when the same field or operator has to be specified in multiple expressions.

**Examples**

**AND Queries With Multiple Expressions Specifying the Same Field**

Consider the following example:

db.inventory.find( { $and: [ { price: { $ne: 1.99 } }, { price: { $exists: true } } ] } )

This query will select all documents in the inventory collection where:

* the price field value is not equal to 1.99 **and**
* the price field exists.

This query can be also be constructed with an implicit AND operation by combining the operator expressions for the price field. For example, this query can be written as:

db.inventory.find( { price: { $ne: 1.99, $exists: true } } )

**AND Queries With Multiple Expressions Specifying the Same Operator**

Consider the following example:

db.inventory.find( {

$and : [

{ $or : [ { price : 0.99 }, { price : 1.99 } ] },

{ $or : [ { sale : true }, { qty : { $lt : 20 } } ] }

]

} )

This query will select all documents where:

* the price field value equals 0.99 or 1.99, **and**
* the sale field value is equal to true **or** the qty field value is less than 20.

This query cannot be constructed using an implicit AND operation, because it uses the [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#op._S_or) operator more than once.

**$not**

$not

*Syntax*: { field: { $not: { <operator-expression> } } }

$not performs a logical NOT operation on the specified <operator-expression> and selects the documents that do *not* match the <operator-expression>. This includes documents that do not contain the field.

Consider the following query:

db.inventory.find( { price: { $not: { $gt: 1.99 } } } )

This query will select all documents in the inventory collection where:

* the price field value is less than or equal to 1.99 **or**
* the price field does not exist

{ $not: { $gt: 1.99 } } is different from the [$lte](https://docs.mongodb.com/manual/reference/operator/query/lte/#op._S_lte) operator. { $lte: 1.99 } returns *only* the documents where price field exists and its value is less than or equal to 1.99.

Remember that the $not operator only affects *other operators* and cannot check fields and documents independently. So, use the $not operator for logical disjunctions and the [$ne](https://docs.mongodb.com/manual/reference/operator/query/ne/#op._S_ne) operator to test the contents of fields directly.

Consider the following behaviors when using the $not operator:

* The operation of the $not operator is consistent with the behavior of other operators but may yield unexpected results with some data types like arrays.
* The $not operator does **not** support operations with the [$regex](https://docs.mongodb.com/manual/reference/operator/query/regex/#op._S_regex) operator. Instead use // or in your driver interfaces, use your language’s regular expression capability to create regular expression objects.

Consider the following example which uses the pattern match expression //:

db.inventory.find( { item: { $not: /^p.\*/ } } )

The query will select all documents in the inventory collection where the item field value does *not* start with the letter p.

If you are using Python, you can write the above query with the PyMongo driver and Python’s python:re.compile() method to compile a regular expression, as follows:

import re

for noMatch in db.inventory.find( { "item": { "$not": re.compile("^p.\*") } } ):

print noMatch

**$nor**

On this page

* [Examples](https://docs.mongodb.com/manual/reference/operator/query/nor/#examples)

$nor

$nor performs a logical NOR operation on an array of one or more query expression and selects the documents that **fail** all the query expressions in the array. The $nor has the following syntax:

{ $nor: [ { <expression1> }, { <expression2> }, ... { <expressionN> } ] }

See also

[find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find), [update()](https://docs.mongodb.com/manual/reference/method/db.collection.update/#db.collection.update), [$or](https://docs.mongodb.com/manual/reference/operator/query/or/#op._S_or), [$set](https://docs.mongodb.com/manual/reference/operator/update/set/#up._S_set), and [$exists](https://docs.mongodb.com/manual/reference/operator/query/exists/#op._S_exists).

**Examples**

**$nor Query with Two Expressions**

Consider the following query which uses only the $nor operator:

db.inventory.find( { $nor: [ { price: 1.99 }, { sale: true } ] } )

This query will return all documents that:

* contain the price field whose value is *not* equal to 1.99 and contain the sale field whose value *is not* equal to true **or**
* contain the price field whose value is *not* equal to 1.99 *but* do *not* contain the sale field **or**
* do *not* contain the price field *but* contain the sale field whose value *is not* equal to true **or**
* do *not* contain the price field *and* do *not* contain the sale field

**$nor and Additional Comparisons**

Consider the following query:

db.inventory.find( { $nor: [ { price: 1.99 }, { qty: { $lt: 20 } }, { sale: true } ] } )

This query will select all documents in the inventory collection where:

* the price field value does *not* equal 1.99 **and**
* the qty field value is *not* less than 20 **and**
* the sale field value is *not* equal to true

including those documents that do not contain these field(s).

The exception in returning documents that do not contain the field in the $nor expression is when the $nor operator is used with the [$exists](https://docs.mongodb.com/manual/reference/operator/query/exists/#op._S_exists) operator.

**$nor and $exists**

Compare that with the following query which uses the $nor operator with the [$exists](https://docs.mongodb.com/manual/reference/operator/query/exists/#op._S_exists) operator:

db.inventory.find( { $nor: [ { price: 1.99 }, { price: { $exists: false } },

{ sale: true }, { sale: { $exists: false } } ] } )

This query will return all documents that:

* contain the price field whose value is *not* equal to 1.99 and contain the sale field whose value *is not* equal to true

**$or**

On this page

* [Behaviors](https://docs.mongodb.com/manual/reference/operator/query/or/#behaviors)

$or

The $or operator performs a logical OR operation on an array of *two or more* <expressions> and selects the documents that satisfy *at least* one of the <expressions>. The $or has the following syntax:

{ $or: [ { <expression1> }, { <expression2> }, ... , { <expressionN> } ] }

Consider the following example:

db.inventory.find( { $or: [ { quantity: { $lt: 20 } }, { price: 10 } ] } )

This query will select all documents in the inventory collection where either the quantity field value is less than 20 **or** the price field value equals 10.

**Behaviors**

**$or Clauses and Indexes**

When evaluating the clauses in the $or expression, MongoDB either performs a collection scan or, if all the clauses are supported by indexes, MongoDB performs index scans. That is, for MongoDB to use indexes to evaluate an $or expression, all the clauses in the $or expression must be supported by indexes. Otherwise, MongoDB will perform a collection scan.

When using indexes with $or queries, each clause of an $or can use its own index. Consider the following query:

db.inventory.find( { $or: [ { quantity: { $lt: 20 } }, { price: 10 } ] } )

To support this query, rather than a compound index, you would create one index on quantity and another index on price:

db.inventory.createIndex( { quantity: 1 } )

db.inventory.createIndex( { price: 1 } )

MongoDB can use all but the [geoHaystack](https://docs.mongodb.com/manual/core/geohaystack/) index to support $or clauses.

**$or and text Queries**

Changed in version 2.6.

If $or includes a [$text](https://docs.mongodb.com/manual/reference/operator/query/text/#op._S_text) query, all clauses in the $or array must be supported by an index. This is because a [$text](https://docs.mongodb.com/manual/reference/operator/query/text/#op._S_text) query *must* use an index, and $or can only use indexes if all its clauses are supported by indexes. If the [$text](https://docs.mongodb.com/manual/reference/operator/query/text/#op._S_text) query cannot use an index, the query will return an error.

**$or and GeoSpatial Queries**

Changed in version 2.6.

$or supports [geospatial clauses](https://docs.mongodb.com/manual/reference/operator/query-geospatial/) with the following exception for the near clause (near clause includes [$nearSphere](https://docs.mongodb.com/manual/reference/operator/query/nearSphere/#op._S_nearSphere) and [$near](https://docs.mongodb.com/manual/reference/operator/query/near/#op._S_near)). $or cannot contain a near clause with any other clause.

**$or and Sort Operations**

Changed in version 2.6.

When executing $or queries with a [sort()](https://docs.mongodb.com/manual/reference/method/cursor.sort/#cursor.sort), MongoDB can now use indexes that support the $or clauses. Previous versions did not use the indexes.

**$or versus $in**

When using $or with <expressions> that are equality checks for the value of the same field, use the [$in](https://docs.mongodb.com/manual/reference/operator/query/in/#op._S_in) operator instead of the $or operator.

For example, to select all documents in the inventory collection where the quantity field value equals either 20 *or* 50, use the [$in](https://docs.mongodb.com/manual/reference/operator/query/in/#op._S_in) operator:

db.inventory.find ( { quantity: { $in: [20, 50] } } )

**Nested $or Clauses**

You may nest $or operations.