How do you use the OR ($or) operator in MongoDB queries?

Authored by

stats writer

June 30, 2024

The OR ($or) operator in MongoDB queries is used to specify multiple query conditions that must be met in order for a document to be returned in the query results. This allows for more flexibility in querying data, as it allows for documents to be retrieved based on any of the specified conditions being met. The OR operator is typically used in combination with other query operators, such as $eq, $ne, $gt, etc., to create complex and specific query conditions. By using the OR operator, MongoDB queries can be tailored to retrieve data that meets a variety of different criteria, making it a powerful tool for data analysis and retrieval.

MongoDB: Use the OR ($or) Operator in Queries

You can use the $or operator in MongoDB to query for documents that meet one of multiple criteria.

This operator uses the following basic syntax:

```
db.myCollection.find({
"$or":
})
```

This particular example finds all documents in the collection titled myCollection where field1 is equal to "hello" *or*field2 has a value greater than or equal to 10.

The following examples show how to use this syntax in practice with a collection teams with the following documents:

```
db.teams.insertOne({team: "Mavs", points: 30,
```

```
rebounds: 8})
db.teams.insertOne({team: "Mavs", points: 35,
rebounds: 12})
db.teams.insertOne({team: "Spurs", points: 20,
rebounds: 7})
db.teams.insertOne({team: "Spurs", points: 25,
rebounds: 5})
db.teams.insertOne({team: "Spurs", points: 23,
rebounds: 9})
```

Example 1: Use OR Operator with Two Fields

The following code shows how to find all documents in the teams collection where the "team" field is equal to "Spurs" *or* the value in the "points" field is greater than or equal to 31:

```
db.teams.find({
"$or":
})
```

This query returns the following documents:

```
{ _id: ObjectId("62018750fd435937399d6b6f"),
team: 'Mavs',
```

points: 35,
rebounds: 12 }
{ _id: ObjectId("62018750fd435937399d6b70"),
team: 'Spurs',
points: 20,
rebounds: 7 }
{ _id: ObjectId("62018750fd435937399d6b71"),
team: 'Spurs',
points: 25,
rebounds: 5 }
{ _id: ObjectId("62018750fd435937399d6b72"),
team: 'Spurs',
points: 23,
rebounds: 9 }

Notice that each document in the output contains "Spurs" in the team field *or* a value greater than or equal to 31 in the points field.

Example 2: Use OR Operator with More Than Two Fields

The following code shows how to find all documents in the teams collection where the "team" field is equal to "Mavs" *or* the value in the "points" field is greater than or equal to 25 *or* the value in the "rebounds" field is

less than 8:

```
db.teams.find({
"$or":
})
```

This query returns the following document:

```
{ _id: ObjectId("62018750fd435937399d6b6e"),
team: 'Mavs',
points: 30,
rebounds: 8 }
{ _id: ObjectId("62018750fd435937399d6b6f"),
team: 'Mavs',
points: 35,
rebounds: 12 }
{ _id: ObjectId("62018750fd435937399d6b70"),
team: 'Spurs',
points: 20,
rebounds: 7 }
{ _id: ObjectId("62018750fd435937399d6b71"),
team: 'Spurs',
points: 25,
rebounds: 5 }
```

The "team" field is equal to "Mavs"The "points" field has a value greater than or equal to 25The "rebounds" field has a value less than 8

Note: You can find the complete documentation for the $or function .

Additional Resources

The following tutorials explain how to perform other common operations in MongoDB:

MongoDB: How to Use the AND Operator in Queries

MongoDB: How to Query for "not null" in Specific Field