How can I query MongoDB to retrieve data within a specific date range?

Authored by

stats writer

July 2, 2024

MongoDB is a popular NoSQL database that allows users to store and retrieve data in a flexible and efficient manner. One of its key features is the ability to query data based on specific criteria, such as date ranges. This means that users can retrieve data that falls within a specific date range, allowing them to analyze and manipulate data in a more targeted and precise manner. This can be achieved by using MongoDB's date-related operators and functions, which allow for a variety of date-based queries. By specifying the desired start and end dates, users can retrieve data within a specific date range and perform further analysis and operations on the results. This feature makes MongoDB a powerful tool for managing and analyzing large volumes of data with time-based attributes.

MongoDB: Query with a Date Range

You can use the following basic syntax to perform a query with a date range in MongoDB:

```
db.collection.find({
day: {
$gt: ISODate("2020-01-21"),
$lt: ISODate("2020-01-24")
}
})
```

This particular query will return all documents in the collection where the "day" field is greater than 2020-01-21 and less than 2020-01-24.

Note that $gt indicates "greater than" and $lt indicates "less than."

You can also use $gte for "greater than or equal" and $lte for "less than or equal."

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

```
db.sales.insertOne({day: new Date("2020-01-20"), amount: 40})db.sales.insertOne({day: new Date("2020-01-21"), amount: 32})db.sales.insertOne({day: new Date("2020-01-22"), amount: 19})db.sales.insertOne({day: new Date("2020-01-23"), amount: 29})db.sales.insertOne({day: new Date("2020-01-24"), amount: 35})
```

Example 1: Find Documents Between Two Dates

We can use the following code to find all documents where the "day" field is between two specific dates:

```
db.sales.find({
day: {
$gt: ISODate("2020-01-21"),
$lt: ISODate("2020-01-24")
```

}
})

This query returns the following two documents:

{ _id: ObjectId("618548bc7529c93ea0b41490"),
day: 2020-01-22T00:00:00.000Z,
amount: 19 }

{ _id: ObjectId("618548bc7529c93ea0b41491"),
day: 2020-01-23T00:00:00.000Z,
amount: 29 }

Example 2: Find Documents After Specific Date

We can use the following code to find all documents where the "day" field is after a specific date:

db.sales.find({
day: {
$gt: ISODate("2020-01-22")
}
})

This query returns the following two documents:

{ _id: ObjectId("618548bc7529c93ea0b41491"),
day: 2020-01-23T00:00:00.000Z,
amount: 29 }

{ _id: ObjectId("618548bc7529c93ea0b41492"),
day: 2020-01-24T00:00:00.000Z,
amount: 35 }

Example 3: Find Documents Before Specific Date

We can use the following code to find all documents where the "day" field is before a specific date:

```
db.sales.find({
day: {
$lt: ISODate("2020-01-22")
}
})
```

This query returns the following two documents:

{ _id: ObjectId("618548bc7529c93ea0b4148e"),
day: 2020-01-20T00:00:00.000Z,
amount: 40 }

{ _id: ObjectId("618548bc7529c93ea0b4148f"),

day: 2020-01-21T00:00:00.000Z,
amount: 32 }

Note: You can find the complete documentation for the ISODate() function .

Additional Resources

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