In today's data-driven world, the ability to efficiently store and manage large amounts of data is crucial. MongoDB, a powerful **NoSQL** database, has become a go-to choice for developers looking for flexibility, scalability, and performance. Unlike traditional relational databases, MongoDB uses a document-oriented data model, making it easier to handle complex data structures and scale out horizontally.

In this tutorial, we'll guide you through the basics of MongoDB, from setting up your environment to performing **CRUD** operations and optimizing performance. Whether you're new to databases or looking to expand your skills, this step by step guide to MongoDB will help you harness the full potential of MongoDB. Let's get started and explore how MongoDB can revolutionize the way you handle data. But before this let's have a quick look in MongoDB.

What is MongoDB?

As we already explain that, MongoDB is a **document-oriented** NoSQL database system that provides high scalability, flexibility, and performance. Unlike standard relational databases, MongoDB stores data in a JSON document structure form. This makes it easy to operate with dynamic and unstructured data and MongoDB is an open-source and cross-platform database System.

Database

- Database is a container for collections.
- Each database gets its own set of files.
- A single MongoDB server can has multiple databases.

Collection

- Collection is a group of documents.
- Collection is equivalent to RDBMS table.
- A collection consist inside a single database.
- Collections do not enforce a schema.
- A Collection can have different fields within a Documents.

Why Use MongoDB?

Document Oriented Storage − Data is stored in the form of JSON documents.

- **Index on any attribute**: Indexing in MongoDB allows for faster data retrieval by creating a searchable structure on selected attributes, optimizing query performance.

- **Replication and high availability**: MongoDB's replica sets ensure data redundancy by maintaining multiple copies of the data, providing fault tolerance and continuous availability even in case of server failures.
- **Auto-Sharding**: Auto-sharding in MongoDB automatically distributes data across multiple servers, enabling horizontal scaling and efficient handling of large datasets.
- **Big Data and Real-time Application**: When dealing with massive datasets or applications requiring real-time data updates, MongoDB's flexibility and scalability prove advantageous.
- **Rich queries**: MongoDB supports complex queries with a variety of operators, allowing you to retrieve, filter, and manipulate data in a flexible and powerful manner.
- **Fast in-place updates**: MongoDB efficiently updates documents directly in their place, minimizing data movement and reducing write overhead.
- **Professional support by MongoDB**: MongoDB offers expert technical support and resources to help users with any issues or challenges they may encounter during their database operations.
- **Internet of Things (IoT) Applications:** Storing and analyzing sensor data with its diverse formats often aligns well with MongoDB's document structure.

Where to Use MongoDB?

- Mobile and Social Infrastructure
- Data Hub
- Previous Pag
- Big Data
- User Data Management
- Content Management and Delivery

Prerequisites for the MongoDB

Before you go to study MongoDB, it is suitable if you have some prior knowledge of Databases, Frontend development, Text editor and execution of programs, etc. It will be beneficial if you have a basic understanding of database fundamentals because we'll be developing high-performance databases (RDBMS).

What is ObjectId in MongoDB?

In **MongoDB**, every document within a collection contains an “_id” field that uniquely identifies it, serving as the primary key. The default format for this field is the ObjectId, a **12-byte BSON** type that ensures uniqueness and embeds useful metadata, such as the creation timestamp.

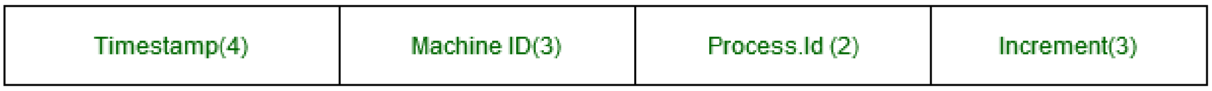
In this article, We will use **MongoDB ObjectId** in detail by understanding their examples in detail.

MongoDB ObjectId

Every document in the **collection** has an “_id” field that is used to uniquely identify the document in a particular collection it acts as the primary key for the documents in the collection. The “**_id**” field can be used in any format and the default format is the **ObjectId** of the document.

An ObjectID is a 12-byte Field Of BSON type

- The first **4** bytes represent the Unix Timestamp of the document.
- The next **3** bytes are the machine **ID** on which the MongoDB server is running.
- The next **2** bytes are of the process ID.
- The last Field is 3 bytes used for incrementing the objectid.



Timestamp(4)	Machine ID(3)	Process.Id (2)	Increment(3)

Format of ObjectId:

ObjectId(<hexadecimal>)

ObjectId accepts one parameter which is optional Hexadecimal ObjectId in String.

We can give our own ObjectId to the document but it must be unique.

*db.<collectionname>.insertOne({"_id":"231231"})

Key Characteristics of MongoDB ObjectId:

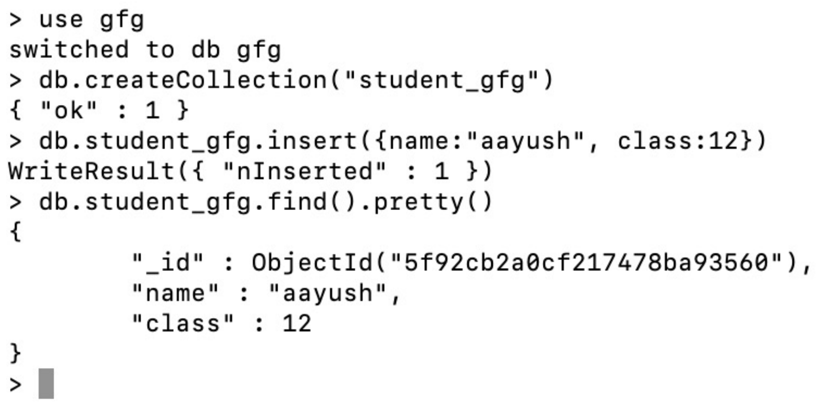
- **Uniqueness**: Ensures each document has a unique identifier within a collection.
- **Timestamp**: Embeds a timestamp, allowing you to extract the creation time of the document.
- **Efficiency**: Provides a compact, efficient way to generate unique identifiers without requiring coordination across servers.

Example of ObjectId in MongoDB

***Database** : gfg*

***Collection:** student_gfg*

Output:



```
> use gfg
switched to db gfg
> db.createCollection("student_gfg")
{ "ok" : 1 }
> db.student_gfg.insert({name:"aayush", class:12})
WriteResult({ "nInserted" : 1 })
> db.student_gfg.find().pretty()
{
        "_id" : ObjectId("5f92cb2a0cf217478ba93560"),
        "name" : "aayush",
        "class" : 12
}
> 
```

Methods of MongoDB ObjectId

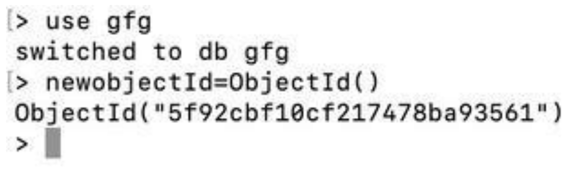
1. **str:** Returns the hexadecimal string format of the **ObjectId**.
2. **ObjectId.getTimestamp()** : It returns the timestamp portion of the object as a Date.
3. **ObjectId.valueOf():** It return the **hexadecimal** format of a given String Literal.
4. **ObjectId.toString():** This method returns ObjectId in String format in javascript representation.

1. Creating ObjectId: To generate new ObjectId of particular document.

newObjectId = ObjectId()

Output:

ObjectId("5f92cbf10cf217478ba93561")



```
> use gfg
switched to db gfg
> newobjectId=ObjectId()
ObjectId("5f92cbf10cf217478ba93561")
> 
```

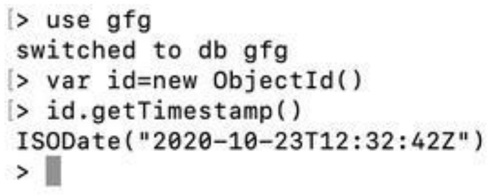
2. Timestamp of the ObjectID: It returns the timestamp information of the object as a Date in ISO format.

var id =new ObjectId();

id.getTimestamp()

Output:

ISODate(“2020-10-23T12:32:42Z”)



```
> use gfg
switched to db gfg
> var id=new ObjectId()
> id.getTimestamp()
ISODate("2020-10-23T12:32:42Z")
>
```

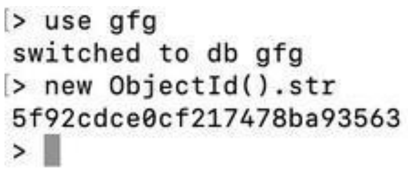
3. Converting ObjectId to string: ObjectId can be converted into string format.

new ObjectId().str

Output:

5f92cdce0cf217478ba93563

Output:



```
> use gfg
switched to db gfg
> new ObjectId().str
5f92cdce0cf217478ba93563
>
```

Conclusion

ObjectId in MongoDB is a fundamental component that ensures each document within a collection has a unique identifier. By embedding metadata like the creation timestamp and maintaining efficiency in generating unique IDs, ObjectId plays a crucial role in the effective management and retrieval of documents in MongoDB.

MongoDB ObjectId

Can I specify my own value for the “_id” field in MongoDB?

Yes, you can specify your own value for the “_id” field as long as it is unique within the collection.

What is the purpose of the timestamp in ObjectId?

The timestamp in ObjectId allows you to determine the creation time of the document, which can be useful for time-based queries and sorting.

How does MongoDB ensure the uniqueness of ObjectId?

MongoDB ensures the uniqueness of ObjectId by combining a timestamp, machine ID, process ID, and a counter, making it nearly impossible to generate the same ObjectId within the same collection.

Can I convert an ObjectId to a different format?

Yes, you can convert an ObjectId to its hexadecimal string format using the str method, and you can also retrieve the timestamp or get its string representation in JavaScript.