



## INTRODUCTION

MongoDB is a document database which is often referred to as a non-relational database. This does not mean that relational data cannot be stored in document databases. It means that relational data is stored differently. A better way to refer to it is as a non-tabular database.

MongoDB stores data in flexible documents. Instead of having multiple tables you can simply keep all of your related data together. This makes reading your data very fast.

You can still have multiple groups of data too. In MongoDB, instead of tables these are called collections.

## DATABASE

MongoDB groups collections into databases.

A single instance of MongoDB can host several databases, each grouping together zero or more collections.

A database has its own permissions, and each database is stored in separate files on disk.

A good rule of thumb is to store all data for a single application in the same database.

## SQL

SQL databases are considered relational databases. They store related data in separate tables. When data is needed, it is queried from multiple tables to join the data back together

## NO SQL

A NoSQL database has a dynamic schema for unstructured data. Data is stored in many ways which means it can be document-oriented, column-oriented, graph-based, or organized as a key-value store. This flexibility means that documents can be created without having a defined structure first. Also, each document can have its own unique structure. The syntax varies from database to database, and you can add fields as you go.

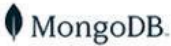
SQL	NoSQL
<b>Stands for Structured Query Language</b>	Stands for Not only SQL
<b>Relational database management system (RDBMS)</b>	Non-relational database management system
<b>Data is stored in tables with columns and rows</b>	Data is stored in collection or documents
<b>Supports JOIN and complex queries</b>	Does not supports JOIN and complex queries
<b>Vertically scalable</b>	Horizontally scalable
<b>Ex :</b> MySQL, PostgreSQL, Oracle, etc	Ex: MongoDB, HBase, Neo4j, etc

## Install MongoDB Compass

The very first step is to install the MongoDB compass into your system.

Go to the official site of [MongoDB compass](#) and click the download now button.

[←](#) [→](#) [↻](#) [mongodb.com/products/tools/compass](#) [🔗](#) [★](#) [🔒](#) [🔍](#) [🗖](#) [👤](#) [⋮](#)

 [Products](#) [Resources](#) [Solutions](#) [Company](#) [Pricing](#) [🔍](#) [⋮](#)

# Compass. The GUI for MongoDB.

Compass is a free interactive tool for querying, optimizing, and analyzing your MongoDB data. Get key insights, drag and drop to build pipelines, and more.

[Download Now](#) [Read the docs](#) [→](#)

<https://www.mongodb.com/try/download/compass>

[Version](#)  
1.40.4 (Stable) [▼](#)

[Platform](#)  
Windows 64-bit (10+) [▼](#)

[Package](#)  
exe [▼](#)

[Download](#) [⬇](#) [📄](#) [Copy link](#) [More Options](#) [⋮](#)

# Installing MongoDB Shell (mongosh)

There are many ways to connect to your MongoDB database. We will start by using the MongoDB Shell, **mongosh**. Use the [official instructions](#) to install **mongosh** on your operating system. To verify that it has been installed properly, open your terminal and type: **mongosh --version**. You should see that the latest version is installed.

The version used in this tutorial is v1.3.1.

## Create Database using mongosh

After connecting to your database using **mongosh**, you can see which database you are using by typing **db** in your terminal.

If you have used the connection string provided from the MongoDB Atlas dashboard, you should be connected to the **myFirstDatabase** database.

## Show all databases

To see all available databases, in your terminal type **show dbs**.

Notice that **myFirstDatabase** is not listed. This is because the database is empty. An empty database is essentially non-existent.

## Change or Create a Database

You can change or create a new database by typing **use** then the name of the database.

### Example

Create a new database called "blog":

```
use blog
```

# Create Collection using **mongosh**

There are 2 ways to create a collection.

## Method 1

You can create a collection using the **createCollection()** database method.

### Example

```
db.createCollection("posts")
```

## Method 2

You can also create a collection during the **insert** process.

### Example

We are here assuming **object** is a valid JavaScript object containing post data:

```
db.posts.insertOne(object)
```

# Insert Documents

There are 2 methods to insert documents into a MongoDB database.

## **insertOne()**

To insert a single document, use the **insertOne()** method.

This method inserts a single object into the database.

## **insertMany()**

To insert multiple documents at once, use the **insertMany()** method.

This method inserts an array of objects into the database.

# Find Data

There are 2 methods to find and select data from a MongoDB collection, `find()` and `findOne()`.

## find()

To select data from a collection in MongoDB, we can use the `find()` method.

This method accepts a query object. If left empty, all documents will be returned.

### Example

```
db.posts.find()  
Try it Yourself »
```

## findOne()

To select only one document, we can use the `findOne()` method.

This method accepts a query object. If left empty, it will return the first document it finds.

### Example

```
db.posts.findOne()
```

Command	Expected Output	Notes
show dbs	admin 40.00 KiB config 72.00 KiB db 128.00 KiB local 40.00 KiB	All Databases are shown

use db	switched to db db	Connect and use db
show collections	Students	Show all tables
db.foo.insert({"bar" : "baz"})		Insert a record to collection. Create Collection if not exists

## Querying Data

To query, or filter, data we can include a query in our `find()` or `findOne()` methods.

### Example

```
db.posts.find( {category: "News"} )
```

## Update Document

To update an existing document we can use the `updateOne()` or `updateMany()` methods.

The first parameter is a query object to define which document or documents should be updated.

The second parameter is an object defining the updated data.

### updateOne()

The `updateOne()` method will update the first document that is found matching the provided query.

Let's see what the "like" count for the post with the title of "Post Title 1":

## Example

```
db.posts.find( { title: "Post Title 1" } )
```

# Delete Documents

We can delete documents by using the methods `deleteOne()` or `deleteMany()`.

These methods accept a query object. The matching documents will be deleted.

## deleteOne()

The `deleteOne()` method will delete the first document that matches the query provided.

## Example

```
db.posts.deleteOne({ title: "Post Title 5" })
```

[Try it Yourself »](#)

## deleteMany()

The `deleteMany()` method will delete all documents that match the query provided.

## Example

```
db.posts.deleteMany({ category: "Technology" })
```

# MongoDB Query Operators

There are many query operators that can be used to compare and reference document fields.

## Comparison

The following operators can be used in queries to compare values:



- **\$eq**: Values are equal
- **\$ne**: Values are not equal
- **\$gt**: Value is greater than another value
- **\$gte**: Value is greater than or equal to another value
- **\$lt**: Value is less than another value
- **\$lte**: Value is less than or equal to another value
- **\$in**: Value is matched within an array

## Logical

The following operators can logically compare multiple queries.

- **\$and**: Returns documents where both queries match
- **\$or**: Returns documents where either query matches
- **\$nor**: Returns documents where both queries fail to match
- **\$not**: Returns documents where the query does not match

## Evaluation

The following operators assist in evaluating documents.

- **\$regex**: Allows the use of regular expressions when evaluating field values
- **\$text**: Performs a text search
- **\$where**: Uses a JavaScript expression to match documents

# • MongoDB Update Operators

- There are many update operators that can be used during document updates.

## • Fields

- The following operators can be used to update fields:
- **\$currentDate**: Sets the field value to the current date
- **\$inc**: Increments the field value
- **\$rename**: Renames the field
- **\$set**: Sets the value of a field
- **\$unset**: Removes the field from the document

•

•

•



Try it Yourself »

