MongoDB Basics

Data formats

In MongoDB, you will often deal with JSON and BSON formats. Therefore, it’s important to fully understand them.

JSON

JSON stands for JavaScript Object Notation. JSON syntax is based on a subset of JavaScript ECMA-262 3rd edition.

A JSON document is a collection of fields and values in a structured format. For example:

{

"first\_name": "John",

"last\_name": "Doe",

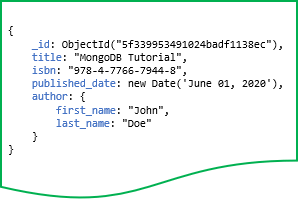
"age": 22,

"skills": ["Programming","Databases", "API"]

}

Documents

MongoDB stores data records as BSON documents, which are simply called documents.



A document is a set of field-and-value pairs with the following structure:

{

field\_name1: value1,

field\_name2: value2,

field\_name3: value3,

...

}

In this syntax, the field names are strings and values can be numbers, strings, objects, arrays, etc. For example:

{

\_id: ObjectId("5f339953491024badf1138ec"),

title: "MongoDB Tutorial",

isbn: "978-4-7766-7944-8",

published\_date: new Date('June 01, 2020'),

author: { first\_name: "John"

, last\_name: "Doe"}

}

This document has the following field-and-value pairs:

* \_id holds an ObjectId
* title holds a string.
* isbn holds a string.
* published\_date holds a value of the Date type.
* author holds an embedded document that contains two fields first\_name and last\_name.

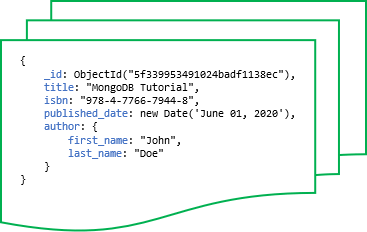
If you are familiar with a relational database management system (RDBMS), you will find that a document is similar to a row in a table, but it is much more expressive.

Field names have the following restrictions:

* MongoDB reserves the field \_id and uses it to uniquely identify the document.
* Field names cannot contain null characters.
* Top-level field names cannot start with the dollar sign ($) character.

Collections

MongoDB stores documents in a collection. A collection is a group of documents.



A collection is analogous to a table in an RDBMS.

|  |  |
| --- | --- |
| **MongoDB** | **RDBMS** |
| Documents | Rows |
| Collections | Tables |

Unlike a table that has a fixed schema, a collection has a dynamic schema.

It means that a collection may contain documents that have any number of different “shapes”. For example, you can store the following documents in the same collection:

{

title: "MongoDB Tutorial",

published\_date: new Date('June 01, 2020')

}

{

title: "MongoDB Basics",

published\_date: new Date('Jan 01, 2021'),

isbn": "978-4-7766-7944-8"

}

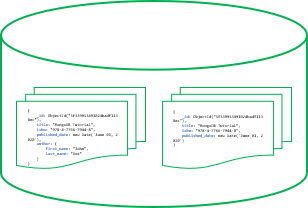
Note that the second document has one more field than the first one. In theory, you can have completely different fields for every document.

A collection has a name e.g., books. The collection name cannot:

* contain the dollar sign ($)
* contain the null character (\0).
* be an empty string.
* begin with the system because MongoDB reserves system\* for internal collection names.

Databases

MongoDB stores collections into a database. A single instance of MongoDB can host multiple databases.



A database can be referenced by a name for example bookdb. The database names cannot:

* Be an empty string ("").
* Contain any of these characters: */*, *\*, *.*, *“*, *\**, *<*, *>*, *:*, *|*, *?*, *$*, (a single space), or *\0* (the null character).
* Exceed the maximum size which is 64 bytes.

MongoDB also has some reserved database names such as admin, local, and config that you cannot use to create new databases.

MongoDB Data Types

Null

The null type is used to represent a null and a field that does not exist. For example:

{

"isbn": null

}

Boolean

The boolean type has two values true and false. For example:

{

"best\_seller": true

}

Number

By default, the mongo shell uses the 64-bit floating-point numbers. For example:

{

"price": 9.95,

"pages": 851

}

The NumberInt and NumberLong classes represent 4-byte and 8-byte integers respectively. For example:

{

"year": NumberInt("2020"),

"words": NumberLong("95403")

}

String

The string type represents any string of UTF-8 characters. For example:

{

"title": "MongDB Data Types"

}

Date

The date type stores dates as 64-bit integers that represents milliseconds since the Unix epoch (January 1, 1970). It does not store the time zone. For example:

{

"updated\_at": new Date()

}

Note that you should always call the new Date(), not just Date() when you create a new Date object because the Date() returns a string representation of the date, not the Date object.

The mongo shell displays dates using local time zone settings. However, MongoDB does not store date with the time zone. To store the time zone, you can use another key e.g., timezone.

Regular Expression

MongoDB allows you to store [JavaScript regular expressions](https://www.javascripttutorial.net/javascript-regex/). For example:

{

"pattern": /\d+/

}

In this example, /\d+/ is a regular expression that matches one or more digits.

Array

The array type allows you to store a list of values of any type. The values do not have to be in the same type, for example:

{

"title": "MongoDB Array",

"reviews": ["John", 3.5, "Jane", 5]

}

The good thing about arrays in the document is that MongoDB understands their structures and allows you to carry operations on their elements.

For example, you can query all documents where 5 is an element of the reviews array. Also, you can create an index on the reviews array to improve the query performance.

Embeded Document

A value of a document can be another document that is often referred to as an embedded document.

The following example shows a book document that contains the author document as an embedded document:

{

"title": "MongoDB Tutorial",

"pages": 945,

"author": {

"first\_name": "John",

"last\_name": "Doe"

}

}

In this example, the author document has its own key/value pairs including first\_name and last\_name.

Object ID

In MongoDB, every document has an "\_id" key. The value of the "\_id" key can be any type. However, it defaults to an ObjectId.

The value of the "\_id" key must be unique within a collection so that MongoDB can identify every document in the collection.

The ObjectId class is the default type for "\_id". It is used to generate unique values globally across servers.

Since MongoDB is designed to be distributed, it is important to ensure the identifiers are unique in the shared environment.

The 12-byte ObjectId value consists of:

* A 4-byte timestamp value that represents the ObjectId‘s generated time measured in seconds since the Unix epoch.
* A 5-byte random value.
* A 3-byte increment counter, initialized to a random value.

For example:

db.books.insertOne({

"title": "MongoDB Basics"

});

Output:

{

"acknowledged" : true,

"insertedId" : ObjectId("5f2fcae09b58c38603442a4f")

}

MongoDB generated the id with the value ObjectId("5f2fcae09b58c38603442a4f"). You can view the inserted document like this:

db.books.find().pretty()

Output:

{

"\_id" : ObjectId("5f2fcae09b58c38603442a4f"),

"title" : "MongoDB Basics"

}

MongoDB insertOne

Introduction to MongoDB insertOne() method

The insertOne() method allows you to insert a single document into a collection.

The insertOne() method has the following syntax:

db.collection.insertOne(

<document>,

{ writeConcern: <document>}

)

The insertOne() method accepts two arguments:

* document is a document that you want to insert into the collection. The document argument is required.
* writeConcern is an optional argument that describes the level of acknowledgment requested from MongoDB for insert operation to a standalone MongoDB server or to shared clusters. We’ll discuss the writeConcern another tutorial.

The insertOne() method returns a document that contains the following fields:

* acknowledged is a boolean value. It is set to true if the insert executed with write concern or false if the write concern was disabled.
* insertedId stores the value of \_id field of the inserted document.

Note that if the collection does not exist, the insertOne() method will also create the collection and insert the document into it.

If you don’t specify the \_id field in the document, MongoDB will add the \_id field and generate a unique ObjectId for it before insert.

If you explicitly specify a value for the \_id field, you need to ensure that it is unique in the collection. Otherwise, you will get a duplicate key error.

To insert multiple documents into a collection, you use the [insertMany()](https://www.mongodbtutorial.org/mongodb-crud/mongodb-insertmany/) method.

MongoDB insertOne() method examples

First, you need to launch the mongo shell and connect it to the bookdb database:

mongosh bookdb

1) Insert a document without an \_id field example

The following example uses the insertOne() method to insert a new document into the books collection:

db.books.insertOne({

title: 'MongoDB insertOne',

isbn: '0-7617-6154-3'

});

Output:

{

"acknowledged" : true,

"insertedId" : ObjectId("5f31cf00902f22de3464ddc4")

}

In this example, we passed a document to the insertOne() method without specifying the \_id field. Therefore, MongoDB automatically added the \_id field and assigned it a unique ObjectId value.

Note that you will see a different ObjectId value from this example because ObjectId values are specific to machine and time when the insertOne() method executes.

To select the document that you have inserted, you can use the [find()](https://www.mongodbtutorial.org/mongodb-crud/mongodb-find/) method like this:

db.books.find()

Output:

[

{

\_id: ObjectId("621489fcf514a446bf1a98ea"),

title: 'MongoDB insertOne',

isbn: '0-7617-6154-3'

}

]

2) Insert a document with an \_id field example

The following example uses the insertOne() method to insert a document that has an \_id field into the books collection:

db.books.insertOne({

\_id: 1,

title: "Mastering Big Data",

isbn: "0-9270-4986-4"

});

Output:

{ "acknowledged" : true, "insertedId" : 1 }

The following example attempts to insert another document whose \_id field already exists into the books collection:

db.books.insertOne({

\_id: 1,

title: "MongoDB for JS Developers",

isbn: "0-4925-3790-9"

});

Since the \_id: 1 already exists, MongoDB threw the following exception:

WriteError({

"index" : 0,

"code" : 11000,

"errmsg" : "E11000 duplicate key error collection: bookstore.books index: \_id\_ dup key: { \_id: 1.0 }",

"op" : {

"\_id" : 1,

"title" : "MongoDB for JS Developers",

"isbn" : "0-4925-3790-9"

}

})

Summary

* Use db.collection.insertOne() method to insert a single document into a collection.
* If you explicitly provide a value for the \_id field, you must ensure that the value is unique within the collection or you will get a duplicate key error.