DataTypes in MongoDB

KapilChhipa

In MongoDB, the documents are stores in BSON, which is the binary encoded format of JSON and using BSON we can make remote procedure calls in MongoDB. BSON data format supports various data-types. Below are the enlisted MongoDB data types:

1. String: This is the most commonly used data type in MongoDB to store data, BSON strings are of UTF-8. So, the drivers for each programming language convert from the string format of the language to UTF-8 while serializing and de-serializing BSON. The string must be a valid UTF-8.

Example: In the following example we are storing the name of the student in the student collection:

```
Command Prompt - mongo
switched to db gfg
> db.student.insertMany([{name:"Akshay"},{name:"Vikash"}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("601af2dd6fd54aa34c9c6df3"),
                ObjectId("601af2dd6fd54aa34c9c6df4")
        ]
}
> db.student.find().pretty()
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df3"), "name" : "Akshay" }
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df4"), "name" : "Vikash" }
>
```

Here, the data type of the value of the name field is a string.

2. Integer: In MongoDB, the integer data type is used to store an integer value. We can store integer data type in two forms 32 -bit signed integer and 64 – bit signed integer.

Example: In the following example we are storing the age of the student in the student collection:

3. Double: The double data type is used to store the floating-point values.

Example: In the following example we are storing the marks of the student in the student collection:

```
> db.student.insertOne({name:"Sagar",age:20,marks:546.43})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601af4126fd54aa34c9c6df6")
}
> db.student.find().pretty()
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df3"), "name" : "Akshay" }
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df4"), "name" : "Vikash" }
{
        "_id" : ObjectId("601af3456fd54aa34c9c6df5"),
        "name" : "Akash",
        "age" : 19
}
{
        "_id" : ObjectId("601af4126fd54aa34c9c6df6"),
        "name" : "Sagar",
        "age" : 20,
        "marks" : 546.43
}
>
```

4. Boolean: The boolean data type is used to store either true or false.

Example: In the following example we are storing the final result of the student as pass or fail in boolean values.

```
> db.student.insertOne({name:"vishal",pass:true})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601af47d6fd54aa34c9c6df7")
}
> db.student.find().pretty()
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df3"), "name" : "Akshay" }
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df4"), "name" : "Vikash" }
{
        "_id" : ObjectId("601af3456fd54aa34c9c6df5"),
        "name" : "Akash",
        "age" : 19
}
{
        "_id" : ObjectId("601af4126fd54aa34c9c6df6"),
        "name" : "Sagar",
        "age" : 20,
        "marks" : 546.43
}
{
        "_id" : ObjectId("601af47d6fd54aa34c9c6df7"),
        "name" : "vishal",
        "pass" : true
}
>
```

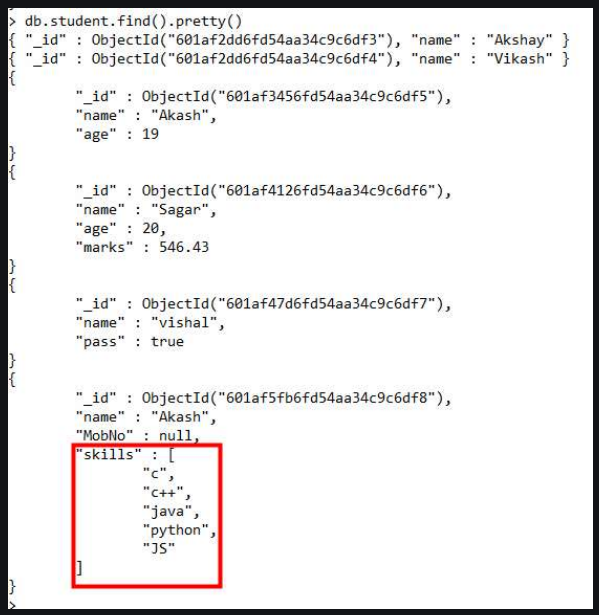
5. Null: The null data type is used to store the null value.

Example: In the following example, the student does not have a mobile number so the number field contains the value null.

```
> db.student.insertOne({name:"Akash",MobNo:null,skills:["c","c++","java","python","JS"]})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601af5fb6fd54aa34c9c6df8")
}
> db.student.find().pretty()
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df3"), "name" : "Akshay" }
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df4"), "name" : "Vikash" }
{
        "_id" : ObjectId("601af3456fd54aa34c9c6df5"),
        "name" : "Akash",
        "age" : 19
}
{
        "_id" : ObjectId("601af4126fd54aa34c9c6df6"),
        "name" : "Sagar",
        "age" : 20,
        "marks" : 546.43
}
{
        "_id" : ObjectId("601af47d6fd54aa34c9c6df7"),
        "name" : "vishal",
        "pass" : true
}
{
        "_id" : ObjectId("601af5fb6fd54aa34c9c6df8"),
        "name" : "Akash",
        "MobNo" : null,
        "skills" : [
                "c",
                "c++",
                "java",
                "python",
                "JS"
        ]
}
```

6. Array: The Array is the set of values. It can store the same or different data types values in it. In MongoDB, the array is created using square brackets([]).

Example: In the following example, we are storing the technical skills of the student as an array.



```
> db.student.find().pretty()
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df3"), "name" : "Akshay" }
{ "_id" : ObjectId("601af2dd6fd54aa34c9c6df4"), "name" : "Vikash" }
{
        "_id" : ObjectId("601af3456fd54aa34c9c6df5"),
        "name" : "Akash",
        "age" : 19
}
{
        "_id" : ObjectId("601af4126fd54aa34c9c6df6"),
        "name" : "Sagar",
        "age" : 20,
        "marks" : 546.43
}
{
        "_id" : ObjectId("601af47d6fd54aa34c9c6df7"),
        "name" : "vishal",
        "pass" : true
}
{
        "_id" : ObjectId("601af5fb6fd54aa34c9c6df8"),
        "name" : "Akash",
        "MobNo" : null,
        "skills" : [
                "c",
                "c++",
                "java",
                "python",
                "JS"
        ]
}
```

7. Object: Object data type stores embedded documents. Embedded documents are also known as nested documents. Embedded document or nested documents are those types of documents which contain a document inside another document.

Example: In the following example, we are storing all the information about a book in an embedded document.

8. Object Id: Whenever we create a new document in the collection MongoDB automatically creates a unique object id for that document(if the document does not have it). There is an _id field in MongoDB for each document. The data which is stored in Id is of hexadecimal format and the length of the id is 12 bytes which consist:

- 4-bytes for Timestamp value.
- 5-bytes for Random values. i.e., 3-bytes for machine Id and 2-bytes for process Id.
- 3- bytes for Counter

You can also create your own id field, but make sure that the value of that id field must be unique.

Example: In the following example, when we insert a new document it creates a new unique object id for it.

```
> db.book.insertOne({Book:{name:"C in depth",writer:"Aaksh"}})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601af71f6fd54aa34c9c6df9")
}
> db.bool.find().pretty()
> db.book.find().pretty()
{
        "_id" : ObjectId("601af71f6fd54aa34c9c6df9"),
        "Book" : {
                "name" : "C in depth",
                "writer" : "Aaksh"
        }
}
>
```

9. Undefined: This data type stores the undefined values.

Example: In the following example the type of the duration of the project is undefined.

```
        }
}
> db.project.insertOne({name:"FauG",duration:undefined,binaryValue:"110011001"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601af8516fd54aa34c9c6dfa")
}
> db.project.find().pretty()
{
        "_id" : ObjectId("601af8516fd54aa34c9c6dfa"),
        "name" : "FauG",
        "duration" : undefined,
        "binaryValue" : "110011001"
}
>
```

10. Binary Data: This datatype is used to store binary data.

Example: In the following example the value stored in the binaryValue field is of binary type.

```
> db.project.insertOne({name:"FauG",duration:undefined,binaryValue:"110011001"})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601af8516fd54aa34c9c6dfa")
}
> db.project.find().pretty()
{
        "_id" : ObjectId("601af8516fd54aa34c9c6dfa"),
        "name" : "FauG",
        "duration" : undefined,
        "binaryValue" : "110011001"
}
>
```

11. Date: Date data type stores date. It is a 64-bit integer which represents the number of milliseconds. BSON data type generally supports UTC datetime and it is signed. If the value of the date data type is negative then it represents the dates before 1970. There are various methods to return date, it can be returned either as a string or as a date object. Some method for the date:

- **Date():** It returns the current date in string format.
- **new Date():** Returns a date object. Uses the ISODate() wrapper.
- **new ISODate():** It also returns a date object. Uses the ISODate() wrapper.

Example: In the following example we are using all the above method of the date:

```
>
> var date1  = Date()
> var date2 = new Date()
> var date3 = new ISODate()
> db.date.insertOne({Date1:date1,Date2:date2,Date3:date3})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("601afa326fd54aa34c9c6dfc")
}
> db.date.find().pretty()
{
        "_id" : ObjectId("601afa326fd54aa34c9c6dfc"),
        "Date1" : "Thu Feb 04 2021 01:00:34 GMT+0530 (India Standard Time)",
        "Date2" : ISODate("2021-02-03T19:30:40.014Z"),
        "Date3" : ISODate("2021-02-03T19:30:56.454Z")
}
>
```