MongoDB Data Types

Summary: in this tutorial, you will learn about the most commonly used 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()
}
```

In JavaScript, the `Date` class is used to represent the date type in MongoDB.

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 `ObjectId` uses 12 bytes for storage, where each byte represents 2 hexadecimal digits. In total, an `ObjectId` is 24 hexadecimal digits.

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.

These first 9 bytes of an `ObjectId` guarantee its uniqueness across servers and processes for a single second. The last 3 bytes guarantee uniqueness within a second in a single process.

As a result, these 12-bytes allow for up to 256^3 (16,777,216) unique `ObjectId` s values to be generated per process in a single second.

When you insert a document without specifying a value for the `"_id"` key, MongoDB automatically generates a unique id for the document. 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"
}
```

In this tutorial, you have learned the most commonly used MongoDB data types including null, number, string, array, regular expression, date, and ObjectId.