Syllabus

- Mongo DB Data types, CRUD operations: Mongo DB create operations, Mongo DB read operation, Mongo DB update operations, Mongo DB delete operations, mongo DB bulk write operations

Mongo DB Data Types

- Data types are the types of data used to store data in various formats that is understood by the programming language, or, in our case, the database.
- Normally, the data types can be represented by `string` type, `integer` type, `float` type, etc. The data types give the meaning to the data as well as types of operations that can be performed with these data types

BSON Data Types

- The BSON data types are just like the JSON data types, but in the binary format.
- That's why, they are called Binary JSON. BSON is the binary format and can have simple to complex data types, including the namevalue pairs, such as associative arrays

Data Types in MongoDB

Data Type	Data Type Number	Data Type Allias
Double	1	"double"
String	2	"string"
Object	3	"object"
Array	4	"array"
Binary data	5	"binData"
ObjectId	7	"objectId"
Boolean	8	"bool"
Date	9	"date"
Null	10	"null"
Regular Expression	11	"regex"
JavaScript	13	"javascript"

Data Types in MongoDB

Data Type	Data Type Number	Data Type Allias
JavaScript (with scope)	15	"javascriptWithScope"
32-bit integer	16	"int"
Timestamp	17	"timestamp"
64-bit integer	18	"long"
Decimal128	19	"decimal"
Min key	-1	"minKey"
Max key	127	"maxKey"

Integer Data Types

- Integer data types are used to store numeric values.
- MongoDB supports 32- bit or 64-bit integers which depend on the architecture of the machine on which MongoDB is running. 64-bit integers are also called as `long int` and have alias value as `long` and number value as 18 in the MongoDB data types.
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({
  "Data Type Alias": "int", "Data Type Number":
  "16", "Data Type Value": 7777});
  ```
  ```

String Data types

- String data types are one of the most common data types used in MongoDB. You might be aware that many of the international languages use some special characters.
- In order to save these characters, the MongoDB drivers convert these characters to UTF-8 format during the serialization and de-serialization process, thus, making it possible to store most of these international characters in the BSON strings and validating the international characters to be stored in the database.

String Data types

- **Code 1**
- db.BPBOnlineBooksDataTypesCollection.insert({ "Data Type Alias": "string", "Data Type Number": "2", "Data Type Value": "BPB Publications – The Largest Online Resource for IT Books“});
- **Code 2**

```
db.BPBOnlineBooksDataTypesCollection.find({"Data Type Alias": "string"}).pretty();
```

Double Data Type

- Double data types are used to store the floating point values. Floating point values have decimal points, which are somewhat similar to integer values but they have decimal values with them. For example, 777.27 or 12.10 are decimal type values and they are called double types in MongoDB
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "double", "Data Type Number": "1", "Data Type Value": 777.27});
  ```
- ```
  db.BPBOnlineBooksDataTypesCollection.find({"Data Type Alias": "double"}).pretty();
  ```

Array Data Type

- The array data types are used to store the group of similar data type values which are linked or associated in the form of key and value.
- In our example, we have defined 2 arrays and then we will insert these arrays as a value and the code for the same is as follows:
- `var BPBBookStoresinIndia = ['New Delhi', 'Mumbai', 'Kolkata', 'Chennai'];`
- `var BPBBookStoresinUSA = ['New York', 'Atlanta', 'Arizona', 'New Jersey'];`

Array Data Type

- ```
  db.BPBOnlineBooksDataTypesCollection.insert({
  "Data Type Alias": "array", "Data Type Number":
  "4","Data Type Value 1": BPBBookStoresinIndia,
  "Data Type Value 2": BPBBookStoresinUSA});
  ```
- ```
  db.BPBOnlineBooksDataTypesCollection.find({ "Data Type
  Alias": "array"}).pretty();
  ```

Object Data Type

- The object data types are used to store the embedded documents or the JSON data type which is a kind of JSON object or document and are sometimes called embedded documents
- `var BPBBooksLatestEditions = [{ 'Title': 'Instant Approach to Software Testing: Principles, Applications, Techniques, and Practices', 'Year': '2019', 'ISBN': '9789388511162', 'Pages': 368, 'Weight': '677gm','Dimension': '24x18x2cm'} ];`

Binary Data Type

- The binary data types are used to store the values in binary form such as Base64. Sometimes, there are scenarios where we store some objects like images (in a binary form) in database.
- In our example, we have created a variable which contains a binary data as follows:
- **Code 1**
- ```
  var BPBBooksBinaryData = BinData(1,
  "SGVsbG8gV29ybGQgRnJvbSBCUEIgUHVibGljYXRpb25z");
  ```
- And then, we have inserted this binary data type in our MongoDB collection and the code for the same is shown in the following screenshot:
  ```

Binary Data Type

- **Code 2**
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({ "Data Type Alias": "binData", "Data Type Number": "5", "Data Type Value": BPBBooksBinaryData});
  ```
  ```

Object Id Data Types

- This is a special MongoDB specific data type which stores the unique key ID. In MongoDB, every document in a collection has unique `ObjectId` and every document has the `_id` field.
- **Part name Size(bytes)**
- Timestamp 4
- Machine Id 3
- Process Id 2
- Counter 3

Object Id Data Types

- **Code 1**
- `var BPBBooksObjectId = ObjectId();`
- And then, we have inserted this `ObjectId` data type in our MongoDB
- collection and the code for the same is shown in the following screenshot:
- **Code 2**
- `db.BPBOnlineBooksDataTypesCollection.insert({ "Data Type Alias": "objectid", "Data Type Number": "7", "Data Type Value": BPBBooksObjectId});`

Date data Types

- Date data type stores the date and time. These can be date or date and time combined. There are various methods in MongoDB to generate date and time, as shown in the following table

Date Data Type	Description
Date()	This method returns the current date in the string format
New Date()	This method returns a date object. This method uses the ISODate() wrapper.

Date data Types

- **Code 1**
- `var BPBBooksDate1 = Date();`
- `var BPBBooksDate2 = new Date();`
- And then, we have inserted these date data type variables in our MongoDB
- collection and the code for the same is shown in the following screenshot:
- **Code 2**
- `db.BPBOnlineBooksDataTypesCollection.insert({ "Data Type Alias": "date", "Data Type Number": "9", "Data Type Value 1": BPBBooksDate1, "Date Type Value 2": BPBBooksDate2});`

Null Data Types

- Null data type stores the null values. Null are the data types which has no type or value, so it is called as null. Sometimes, there are scenarios where we have to store something which exists but is yet to be defined and we are not sure what type and value it would have. In this case, we use null.
- In our example, we have created a variable which contains a null type value.
- **Code 1**
- `var BPBBooksNull = null;`
- And then, we have inserted this null data type variable in our MongoDB
- collection and the code for the same is shown in the following screenshot:

Null Data Types

- **Code 2**
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "null", "Data Type Number": "10","Data Type Value": BPBBooksNull});
  ```
- **Code 3**
- ```
  db.BPBOnlineBooksDataTypesCollection.find({"Data Type Alias": "null"}).pretty();
  ```

Regular Expression Data Types

- Regular expression data type stores the regular expression values. Regular expression, or regex, is the sequence of characters which defines the search patterns and is used for find, replace, and validation operations.
- In our example, we have created a variable which contains a regular expression value.

Code 1

```
var BPBBooksRegEx = RegExp("%BPB");
```

Regular Expression Data Types

- **Code 2**
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "regex", "Data Type Number": "11", "Data Type Value": BPBBooksRegEx});
  ```
- **Code 3**
- ```
  db.BPBOnlineBooksDataTypesCollection.find({"Data Type Alias": "regex"}).pretty();
  ```

Java Script data types(without scope)

- We can store JavaScript functions in MongoDB documents and use these functions in our scenarios. In our example, we have created 2 variables where one contains a function definition and the other is a scope variable, but it will be empty and thus, it will be without scope.
- **Code 1**
- `var BPBBooksFunction = "function(){var bpb; bpb=100;}";`
- `var BPBBooksFunctionScope = {};`

Java Script data types(without scope)

- **Code 2**
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "javascript", "Data Type Number": "13", "Data Type Value Function": BPBBooksFunction, "Data Type Value Function Scope": BPBBooksFunctionScope});
  ```
  ```

Java Script data types(with scope)

- In our example, we have created two variables where one contains a function definition and the other one is a scope variable, and now we will define it.
- **Code 1**
- `var BPBBooksFunction = "function(){var bpb; bpb=2000;}";`
- `var BPBBooksFunctionScope = ["object"];`
- And then, we have inserted these variables in our MongoDB collection and the code for the same is shown in the following screenshot

Java Script data types(with scope)

- **Code 2**
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "javascriptWithScope", "Data Type Number": "15","Data Type Value Function": BPBBooksFunction, "Data Type Value Function Scope": BPBBooksFunctionScope});
  ```
  ```

Timestamp Data Types

- Timestamp is the current time of event which is recorded by the computer and it is accurate to milliseconds. We can store timestamps in the MongoDB documents.
- In our example, we have created a variable which contains a timestamp.
- **Code 1**
- `var BPBBooksTimestamp = new Timestamp();`
- And then, we have inserted this variable in our MongoDB collection and the code for the same is shown in the following screenshot:

Timestamp Data Types

- **Code 2**
- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "timestamp", "Data Type Number": "17", "Data Type Value": BPBBooksTimestamp});
  ```
  ```

Boolean data types

- Boolean data type stores the boolean data which is either true or false.
- In our example, we have created two variables which contains a "true value“ and a "false value" respectively.
- **Code 1**
- `var BPBBooksBoolean1 = true;`
- `var BPBBooksBoolean2 = false`

Boolean data types

- ```
  db.BPBOnlineBooksDataTypesCollection.insert({"Data Type Alias": "bool","Data Type Number": "8", "Data Type Value 1": BPBBooksBoolean1,"Data Type Value 2": BPBBooksBoolean2});
  ```
  ```

Min and MAX Key

- In MongoDB, the min and max keys compare a value against the lowest and the highest BSON elements.
- The min key compares the values of the lowest BSON element, whereas the max key compares the values of the highest BSON element

Decimal 128

- The Decimal128 data type has been introduced in the MongoDB version 3.4.
- There are some scenarios where the double data type has limited capacity to store the decimal values which are either very large or very small, and if we store the large decimal value, then the precision will be lost.
- In this case, MongoDB provides the Decimal128 data type so that we can store the decimal values with a lot more precision

Comparison and Sort Order

- While comparing the values of different BSON types, MongoDB uses the following comparison order from the lowest to the highest:
- MinKey
- Null
- Numbers (ints, longs, doubles)
- Symbol, String
- Object

Comparison and Sort Order

- Array
- BinData
- ObjectId
- Boolean
- Date
- Timestamp
- Regular expression
- MaxKey

Comparison and Sort Order

- **Code 1**
- `var BPBBooksVar1 = 100000;`
- `var BPBBooksVar2 = "BPB Publications";`
- `var BPBBooksVar3 = 77.07;`
- `var BPBBooksVar4 = true;`
- `var BPBBooksVar5 = null;`
- `var BPBBooksVar6 = MinKey;`
- `var BPBBooksVar7 = MaxKey;`

Comparison and Sort Order

- **Code 2**
- db.BPBOnlineBooksDataTypesCollectionV2.insert([{ "Data Type Alias": "int", "Data Type Value": BPBBooksVar1
- }, { "Data Type Alias": "string", "Data Type Value": BPBBooksVar2}, { "Data Type Alias": "double", "Data Type Value": BPBBooksVar3}, {"Data Type Alias": "bool","Data Type Value": BPBBooksVar4},{ "Data Type Alias": "null", "Data Type Value": BPBBooksVar5}, { "Data Type Alias": "minKey", "Data Type Value": BPBBooksVar6}, {
- "Data Type Alias": "maxKey","Data Type Value": BPBBooksVar7}]);