Introduction to MongoDB

Prashanth B S[1]

[1]Department of Information Science & Engineering, Nitte Meenakshi Institute of Technology, Yelahanka - 560064, Bengaluru

April 8, 2022

1 Introduction to MongoDB

MongoDB is an open-source NoSQL based database management system designed for the unstructured databases. MongoDB is written using C++[1]. Some of the basic terminologies are as follows,

1. A *document* is the basic unit of data for MongoDB, roughly equivalent to a row in a relational database.
2. A *Collection* can be visualized as scheme free equivalent of the table
3. MongoDB groups collections into databases
4. MongoDB includes a simple but *powerful javascript shell* which is useful for administration of MongoDB instances and data manipulation
5. A single instance of mongodb can invoke many independent databases
6. Every document has a special key "_id" that is unique across the document

2 MongoDB v/s RDBMS

Some of the notable features of the MongoDB w.r.t to RDBMS is discussed below,

- Schema less: MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one document to another
- Ease of scale-out : MongoDB is easy to scale
- Does not support complex joins
- Powerful C++ engine, faster queries
- Supports sharding of data
- Supports Replication and redundancy
- Non Complaint to ACID property

[1]https://www.tutorialspoint.com/mongodb/index.htm

2.1 Documents

The core of the MongoDB is the concept of document which is in the form of key-value pair. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.A sample document is as shown below,

```
{
        user:'prashanth', // user -> key, 'prashanth' -> value
        message: 'First comment', // each field is seperarted by ","
        dateCreated: new Date(2022,4,4,7,45),
        like: 5
}
```

2.2 JSON vs BSON

The key-value pairs in the documents are ordered. The concept of documents are taken from JSON and MongoDB uses BSON format to store the database values. JavaScript Object Notation, more commonly known as JSON, was defined as part of the JavaScript language that is used as standard notation for data exchange over internet. BSON simply stands for "Binary JSON," and that's exactly what it was invented to be. BSON's binary structure encodes type and length information, which allows it to be parsed much more quickly [2].MongoDB stores data in BSON format both internally, and over the network. A sample representation is as shown below,

```
{"hello": "world"} = \x16\x00\x00\x00      // total document size
             \x02                      // 0x02 = type String
             hello\x00                 // field name
             \x06\x00\x00\x00world\x00 // field value
             \x00                      // 0x00 = type EOO ('end of object')
```

2.3 JSON/BSON types

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array. An object is an unordered set of name/value pairs. An object begins with left brace and ends with right brace. Each name is followed by: colon and the name/value pairs are separated by , comma.

- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.An array is an ordered collection of values. An array begins with [left bracket and ends with]right bracket. Values are separated by , comma.

- A *value* can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested. A *string* is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A *character* is represented as a single character string. A string is very much like a C or Java string.A *number* is very much like a C or Java number, except that the octal and hexadecimal formats are not used

Examples for the above is shown below,

[2] https://www.mongodb.com/json-and-bson

```
The { (curly brace) represents the JSON object.
{
    "employee": {
        "name":      "sonoo",
        "salary":     56000,
        "married":   true
    }
}
The [ (square bracket) represents the JSON array. A JSON array can have values and
    objects.
[
    {"name":"Ram", "email":"Ram@gmail.com"},
    {"name":"Bob", "email":"bob32@gmail.com"}
]

// Example 1
{"employees":[
    {"name":"Shyam", "email":"shyamjaiswal@gmail.com"},
    {"name":"Bob", "email":"bob32@gmail.com"},
    {"name":"Jai", "email":"jai87@gmail.com"}
]}

// The XML representation of the above JSON example is given below.
<employees>
    <employee>
        <name>Shyam</name>
        <email>shyamjaiswal@gmail.com</email>
    </employee>
    <employee>
        <name>Bob</name>
        <email>bob32@gmail.com</email>
    </employee>
    <employee>
        <name>Jai</name>
        <email>jai87@gmail.com</email>
    </employee>
</employees>

// Example 2
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Save", "onclick": "SaveDoc()"}
    ]
  }
}}
// The XML representation of above JSON example is given below.
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Save" onclick="SaveDoc()" />
  </popup>
</menu>
```

3 Databases

A group of independant collections constitutes a Database. MongoDB provides default databases to support smooth access by the user, These databases are as follows,

1. *admin*: Root database, used for authentication. A new user gains access to all the database if the user is added to the admin database
2. *local*: These databases are never replicated and are used to store the local databases in a single server
3. *config*: the config databases hold the information of shards in the distributed network where the shard is a chunk of database spread across many servers over the network

4 Setting up the environment in Ubuntu

The following are the steps to install MongoDB Community Edition using the apt package manager.

1. Import the public key used by the package management system. From a terminal, issue the following command to import the MongoDB public GPG Key from https://www.mongodb.org/static/pgp/server-5.0.asc.

   ```
   $ wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add -
   ```

2. Create the list file /etc/apt/sources.list.d/mongodb-org-5.0.list for your version of Ubuntu

   ```
   $ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu
       focal/mongodb-org/5.0 multiverse" | sudo tee
       /etc/apt/sources.list.d/mongodb-org-5.0.list
   ```

3. Reload local package database

   ```
   $ sudo apt update
   ```

4. Install the MongoDB packages

   ```
   $ sudo apt-get install -y mongodb-org
   ```

5. Configure the MongoDB to start at the boot time by default,

   ```
   $ sudo apt-get install -y mongodb-org
   ```

   ```
   $ sudo systemctl enable mongod
   ```

6. To stop the mongodb server :

```
$ sudo systemctl stop mongod
```

7. To check the status of the mongodb server :

```
$ sudo systemctl status mongod
```

8. To restart the mongodb server :

```
$ sudo systemctl restart mongod
```