SURVEY ON MONGODB: AN OPEN-SOURCE DOCUMENT DATABASE

Smita Agrawal, Jai Prakash Verma, Brijesh Mahidhariya, Nimesh Patel
CSE Department, Institute of Technology, Nirma University, Ahmedabad, India

Atul Patel
CMPICA, CHARUSAT University, Changa, Gujarat, India

ABSTRACT

MongoDB is open source cross platform database. It is classified as NoSQL (Not Only SQL). It is written in C++ and it follows document oriented data model. It is database model which provides dynamic schema. It uses features like Map/Reduce, Auto-sharding and MongoDump etc. Using these features MongoDB provides high performance, where Map/Reduce is efficient data arrangement, Auto-sharding is storing data on across the different machines, Backup facilities and many more. It has collections as table and each collection can store different kinds of data. It stores data in JSON like structure. Unlike the RDBMS databases it can store unstructured data as well. It can process and handle large amount of data more efficiently than RDBMS. It is ACID system like RDBMS databases. MongoDB mainly used in such application which produces and uses vast amount of data. Like blogs or sites which produces or stores unstructured data. It can be used to in applications which stores structured and semi-structured data as well.

Key words: Mangodb, Auto-Sharding, Mangodump, Mapreduce


http://www.iaeme.com/IJARET/issues.asp?JType=IJARET&VType=6&IType=12

1. INTRODUCTION

MongoDB provides the high performance using binary wire protocol to interact with the server. It doesn't use HTTP or REST request. It has query optimizer which remembers the how to execute the query fastest. It uses memory mapped file storage engine which leaves memory management to operating system rather than handling its own. As part of our study over MongoDB we are going to start with basic database meaning and requirement, types of database models. After defining small
comparison between those database models. Then we will be describing the NoSQL data models types as MongoDB follows document database model. In MongoDB we will be mainly covering its meaning, data types, main features of MongoDB, query structures and execution, specifications about MongoDB, in the end main advantages and disadvantages of MongoDB, application and areas where it is most suitable to use [1,2,4].

**Database Models:** A database model is a logical structure of a database; it includes how the data organized, in which manner data is stored and how data manipulated in the database. There are so many different databases available in a market. Each one is following its own database model. We have following list of common database models [3, 8, 9, 10]:

- Hierarchical Model
- Network Model
- Relational Model
- Document model

**Hierarchical Model:** In Hierarchical Model data is organized in tree-like structure. Data is stored as a record in connected with other using link. So the parent-child relationship is maintained in this model between two data records. Each record is a collection of fields. It was used in early mainframe Database management systems. This model mainly used in file system. An example of Hierarchical Model is windows registry that store configuration settings in Hierarchical form [3, 10].

**Network Model:** Network model is similar to hierarchical model but moreover it allows many-to-many relationship. This means it have multiple Child as well as multiple parent. Network is represented by Graph structure. IDS (integrated database system), IDMS (integrated database management system) uses network model [3, 9].

**Relational Model:** Relational model was introduced by E. F. codd in 1969. In the relational model data is represent as a tuple and grouped into relations. Data is organized in tables. A table is collection of records. Each record contains same fields. Most relational model uses structured query language (sql) to retrieve, store and manipulate the data. Microsoft office access, MySQL, MsSQL, Oracle database are well known and widely used relational database management systems [3, 8].

**Document model:** Document model is also known as semi structured data model. Document model has no separation between data and schema. This type of model is different from above all models. Document model is NoSQL (Not only SQL) category model. NoSQL is simple to design and can scale horizontally. ‘Documents’ is a central concept of document model in which documents word represent vast amount of data. There are so many databases available those uses document model like MongoDB, Couchbase Server, Clusterpoint, ArangoDB, MarkLogic etc.

2. **MONGODB**

MongoDB is an open source document oriented database. It is developed in C++. It was first developed by the software company 10gen in 2007 to use as service for other software. Later on in 2009 the company went in open-source developing area. Since then MongoDB is used as backend software by many websites. It is not the relational database model but as document database model it uses some structures which can be considered equivalent to RDBMS. Each MongoDB database stores table like structure which is known as **Collection**. Each collections stores data in object called **Document**. Document is basically represents single record which is made of multiple
**Key-Value** pair. It stores data in JSON-like format which MongoDB calls BSON format. It is schema free or rather uses dynamic schema which makes possible it to store different kinds of Documents in same collection [1, 2, 5, 6].

**Collection**: Collection is same as table structure. It stores multiple records which either are exact same kind or related but serving same purpose. Every collection has unique field “_id”.

**Documents**: All data is stored in object called Document. It has multiple field and value related to that particular record.

Example,

```
{ 
  field1: value1, 
  field2: value2, 
  field3: value3, 
  … : … , 
  Field N: value N} 
```

**Key-Value**: This is the basic structure the data is stored. Where Key is field name which the data is to be stored and Value is associated value for that field for particular document or record. Following is the Table-1 that represent RDBMS terminology with MongoDB.

<table>
<thead>
<tr>
<th>RDBMS</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Database</td>
</tr>
<tr>
<td>Table</td>
<td>Collection</td>
</tr>
<tr>
<td>Tuple Row</td>
<td>Document</td>
</tr>
<tr>
<td>Column</td>
<td>Field</td>
</tr>
<tr>
<td>Table join</td>
<td>Embedded Documents</td>
</tr>
<tr>
<td>Primary Key</td>
<td>Primary Key(Default key _id provided by mongodb itself)</td>
</tr>
<tr>
<td>Database Server and Client</td>
<td></td>
</tr>
<tr>
<td>Oracle/ Mysql</td>
<td>mongod</td>
</tr>
<tr>
<td>mysql/sqlplus</td>
<td>mongo</td>
</tr>
</tbody>
</table>

Table 1RDBMS terminology with MongoDB

This is basic structure of MongoDB storing structure. Databases can store multiple Collections. Collection has set of documents. Documents are single records which stores field name and its associated values [1.2.7].
Figure 1 Basic Structure of MongoDB

MongoDB can mainly store most of data types which are used in other database models. But also few other data types like:

**Object:** This mainly refer to embedded document. Embedded documents are documents which are stored as value of other document field.

```javascript
{
    _id: "903f1f78kcf86cd799524153",
    "name" : "vijay pvt ltd",
    "address":
    {
        "street" : "104, Murrlidhar ",
        "city" : "Ahmedabad",
        "state" : "GJ",
        "zip" : "373534"
    }
}
```

**Code:** This data type can store the javascript. It does not store whole application logic but can store limited JavaScript function as value of any collection. Example:

```javascript
{
    _id : "AddFunction",
    "name" : "Addition Function",
    "function" : function (x, y){ return x + y; }
}
```
ObjectId: This is data type that is used to store the value of _id field in every document. This field stores unique value which there in every document in every collection. They are like primary key so it is immutable. It consist of 12 byte, which are as follows,

First 4 Bytes of Time Stamp which stores the time of creation of document,
Next 3 bytes stores the machine id,
After that 2 bytes for process id or PID value,
And last 3 bytes used for incrementing.

Example: ObjectId ("807f1f78kcf86cd799439021")

Rules for declaring key in documents: Same document cannot contain duplicate key. Example, First_Name

```json
{
  "name":"Vijay",
  "name":"Karan"
}
```

- Key name must start with the character.
- Key cannot contain ‘0’ null character, it indicates end of key.
- Character ‘.’ and ‘$’ serves special purpose so it can’t be used in name of any key.
- Key starting from ‘_’ considered as reserved key.

3. MAIN FEATURES PROVIDED BY MONGODB

Full Index Support: - Index support for each attribute same as RDBMS. Indexes provide high performance read operations for frequently used queries. Indexes are used in MongoDB for efficient execution of queries. Without using this indexes MongoDB must have to scan all document in the collection to find data that matches query data. Indexes are special structures which store a small portion of the collection’s data set in an easy to search. They store specific or set fields in order of its value. These indexes uses B-tree data structure. All collections have index on _id field. This field is default, which is there in every collection. If application does not provide value for that then MongoDB uses object id as Index on “_id” field.

Types of indexes supported

Single Field: This is very basic index, which is also found in other types of databases like Relational databases. MongoDB provides default index on “_id” field. Any other single field index can also be created on any other field in collection.

Compound Index: Compound index is created on multiple fields which exists in same collection. Compound index sorted basis of field put first in list. Like if a compound index consists of {“userid”: 1, “score”:1}, the index sorts first by userid and then, within each score value, sort by score.

Multikey Index: This index is used on creating index over the fields which have array values. It creates separate index for each element of an array. It automatically creates index as multikey index if defined field contains array as field. It doesn't need to explicitly mention. This here is index created on field “score” which belongs to
user’s collection. This is single field index. It is searched operation illustrate here which basically filter the result which has score less than 30. With use of index on score field it is easy to find all matched records as per query. Without this index field MongoDB would have to check all documents in collection to find matched entries.

**Replications:** MongoDB maintains copy of database on other servers. Its main purpose is to provide high availability. By storing dataset on multiple servers, it saves risk of losing data from any unexpectedly. All the write operations done on only Primary database which is either on main server or local machine. But read operations can be done on all datasets including replica sets. There is only one primary dataset among all copies of these datasets. All replica sets reflects the changes of this primary dataset.

To reflect changes on replica datasets primary dataset maintains logs called **Oplog.** It is special collection which stores the operations which are performed on the primary database. And later secondary database applies those operations on database. All secondary databases also have own copy of Oplog. It is also known as asynchronous process. There is also Initials Syncs are performed when there is new replica set is created. In the following figure-2 three-member replica set, the primary accepts all write operations. Then the secondaries replicate the oplog to apply to their data sets [4].

**Figure 2 Three-Member Replica Set**

**Arbiter Dataset:** MongoDB also can have special replica set called **Arbiter.** This replica set does not stores the data but only exist to vote new primary dataset when needed. If any case there are even numbered replica set are maintained then to have clear election of primary data set an Arbiter dataset can be created.

**Automatic Fail over:** In case of Primary dataset does not respond for more than 10 seconds, the replica set tries to select new primary dataset. Whichever replica set gets majority votes first they are selected as Primary dataset for limited time or onwards the time. In case of the primary dataset is selected for temporarily then Replica Set Roll Back process takes place which only occurs if any write operations are performed on Replica Primary dataset. Using the Oplog of that replica set old Primary dataset can be resurrected. There is only one disadvantage using this replica set for read operation is replica set may not reflect most recent changes performed on primary dataset, which may give old result when read operation is done. Before determining the number of replica set to maintain it is important to have enough replica set that can decide the new Primary dataset on fail of one or more replica
Table 2 Fault tolerance based on number of replica set

<table>
<thead>
<tr>
<th>Number of Members</th>
<th>Majority Required to Elect a New Primary</th>
<th>Fault Tolerance</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Auto Sharding: Sharding is basically scaling out the database over different machines. In this method the large set of data is partitioned into small parts and stored across the different servers. It is also known as Horizontal Scaling. Each of this Shard or small part of this large dataset act as database itself, but logically they form a large database. Below in figure-3 is simple logical representation of the single large collection and its shards divided into equal number of parts.

Auto Sharding in MongoDB

This process is mainly achieved using the Query Routers or mongos and Config Server. MongoDB does automatic sharding and balancing of shards. Shard Clusters maintain all the tasks using Query routers and Config servers.

Query Routers: This is process on application server which mainly decided a query fired needs to be execute exactly on which shard. It processes the query to appropriate shard and returns the result to the user. There can be multiple query routers on single shard cluster. It divides the load of client requested query if required. All the query goes through query routers and they process further this query.

Config Server: Config server is stores the metadata or mapping information of shards. A query routers uses this information to process the query to appropriate shard. To modify or create new sharded cluster there must be a 3 Config server available. If there are less than 3 then shard cluster information becomes read-only. Any moving or splitting shards needs at least 3 Config server to be operable there for single sharded clusters. Sharding also contains process called Balancer. It is background process which keeps managing shards moving shards into another shard. It is also known as migrating of shards. There is also another background process called Splitting which keeps splitting the Chunks of shard when they grow over the...
more than defined size. Shards also maintains index called Shard Index which is either Range based or Hash bases (Hash function RDBMS). All sharded collection must have this index in every document it stores. And it can be single key index only.

**Map/Reduce:** This is feature is consist of two phases mainly. First is mapping phase. This is searching phase where all the documents from mentioned collections are scanned for given key and value or range of value it comes in. And it collects all matched data into one or more document. In second phase it reduces those one or more document data into single result dataset to give result to user. This phase also includes additional feature of **finalize** function. It is not physical function available but it allows to perform final operation on result reduced before returning to client. This final modification may be sorting or any other calculations on populated result of that map/reduce function.

```
Collection
/
db.orders.mapReduce(
  map --> function() { emit( this.cust_id, this.amount ); },
  reduce --> function(key, values) { return Array.sum( values ) },
  query --> { cust_id: "A123", amount: 500, status: "A" },
  output --> out: "order_totals"
)
```

**Figure 1** Map Reduce

Above figure-4 is example where mapping is done on basis of same customer id and reducing phase includes to telling amount of people who have same customer id with query who had status “A”.[4] This is same as grouping feature of RDBMS. This map/reduce functions are mainly defined in JavaScript. They can write the result of this process in collection or return as inline out as well. Map/Reduce functions which writes output to collections can perform operations like merge or replace on same collection as input. While the functions which returns inline results their output documents can't exceed than 16 MB.

**GridFS:** GridFS is specification store file that exceeds 16 MB used by MongoDB. It is divided in 2 parts. GridFS divides file in small chunks and stores across different
document. Each document size is maximum 255k. It maintains the file using two collections. One is files and other is chunks. Files collection stores the metadata of the file. While chunks collection stores information of each part or chunks of file in documents. This is very beneficial when file system does not allow to store more than limited number of files. And also if file is such that when query is made it’s operable without loading whole file into memory.

It uses compound index on chunks collection. This collection have default one unique index called file_id, which stores id of parent document’s _id field. And another field n which stores sequence number of that particular chunk. The first chunk starts with sequence number having n field value as 0.

**Mongodump:** This is basic backup facility provided by the MongoDB. To perform this feature the user must have allowed backup role also with other roles. It can create back up for all data on server or particular collection as well as any single document also. A query result also can be stored as backup if it is performed. If no argument has passed with **Mongodump** command then it creates the backup in current directory on local machine. It can also create backup without running main mongod daemon process. For that physical path to data must be provided.

**MONGODB QUERY SYNTAX**

**Basic Syntax:** DatabaseName.CollectionName.Operation();

**MongoDB CRUD operations:** CRUD operations are basic Create, Read, Update and Delete operations. In which the read operations can be done all the replica set and sharded dataset but the write operations can only done on Primary dataset.

**Create operation:** For basic creation of new database only one command is used which is use. Example:

```bash
use mydb  # To create the new collection in existing database,

db.createCollection(name, options)  # For insert new document in collection,

db.movie.insert({"name":"Happy New Year"})
```

- db: = current database
- movie: = collection name
- insert: =operation

An array of document can also be passed in insert query function to insert multiple documents at same time. Example:

```bash
db.movie.insert([

{"name":"Happy New Year"},
{"name":"Bang Bang"} ])
```

**Read operation:** This operation mainly performed using find(). This read operation can modify using limits, skip and sort. Example:

```bash
db.users.find("age", {$gt: 18}).sort()  # Where $gt is selection operator, which filters the data. This $gt is provides the value greater than 18.
```

Update operation db.users.update(

```bash
{ age: { $gt: 18 } },
{ $set: { status: "Adult" } }, { multi: true } )
```

The **multi: true** allow the multiple documents to be updated with same query. There is also one options upsert, which is combination of update and insert. If any document does not match query passed then it creates new document in the collection and inserts it with given value.
Delete operation: To delete/remove the document from collection `remove()` method is used. Example:

```javascript
db.users.remove(
   {“status”: “D”} )
```

There is also one additional method named `save()`. It can be used for both insert and update operations. This is method used for documents which does not have `_id` field. When this method is called. It creates `_id` field with `ObjectId` value.

4. ADVANTAGES AND DISADVANTAGES

**Advantages of MongoDB:**
- **Sharding:** To store extremely large amount of data, we have to distribute it to across different servers connected to application. One server can’t handle vast amount of data. So to overcome this problem MongoDB provides auto sharding feature.
- **Speed:** If data is modelled in document oriented not relational i.e. it uses embedded document data Model then it can provide maximum speed. Using indexes we can also fetch data fast from the database.
- **Flexibility:** MongoDB doesn’t require same data structure for all documents like table schema in RDBMS.

**Disadvantages of MongoDB**

- **Joins not supported:** There are no joins exists in MongoDB like relational model. When one needs to use join functionality he/she must add it to code level manually. This may results in reduced flexibility and slow execution. MongoDB uses more memory because it stores key names for each value pairs. Also there are no joins so it uses embedded documents and it results in data redundancy and unnecessary usage of memory.
- **No transactions:** Operations are not treated as transactions in MongoDB. To ensure transaction we have to check manually for completion of transaction and from that choose to commit or rollback.

5. APPLICATIONS OF MONGODB

There are following content management system uses MongoDB as back-end

- **Calipso** – This System is built using NodeJS and MongoDB
- **KeyStoneJS** – This system is also built using NodeJS and MongoDB
- **MongoPress** – It uses PHP as front-end
- **Locomotive** – It is an open-source content management system for Rails
- **Forward** – Forward is open-source e-commerce platform. It supports powerful templates
- **Sourcefourge** – it uses MongoDB as back-end database
- **Shutterfly** – It uses MongoDB to store billions of photos
- **Expedia** – it uses MongoDB to automate note taking process and store all the information
- **Ebay** – it uses MongoDB for search suggestions

6. CONCLUSION AND FUTURE WORK

As MongoDB is said to be providing easy scaling, high availability, and high performance like features. Our study over this database indicates that it does provide those features more efficient than other databases. But to provide this features
methods used like Sharding and Replications over different machines and servers gets costly if the amount of data is way too large. As the large data will be the more servers may be needed to store the Shards and also servers that can store replications. But regardless of the amount of the data the performance will be very high compare to the other databases for the same amount the data. So if the amount of the data is very big and keep increases, it includes unstructured data more and high amount of performance and availability required with very large amount of data which can't be handled by single server for those off the applications only MongoDB should be considered as options to use as database.

REFERENCES


