MongoDB – Indexing Fields for Sub-Documents

Let us consider a situation where you wish to look for documents depending on the city and country fields (from the above example).

As all the fields are part of the location sub-document field, you will have to generate an index on every field of sub-document.

MongoDB – Indexing Fields for Sub-Documents

To create an index on each of the two fields of your sub-document, you have to make use of the below-mentioned code:

```
db.users.ensureIndex( {"location.city":1,"location.country":1}
)
```

After creating an index, the search operation in any sub-document field can be performed using the index:

```
db.users.find({"location.city":"Los Angeles"}
```

MongoDB – HOW THE CONCEPT OF RELATION HELPS

The concept of replication offers redundancy or duplication of data, which accordingly augments data availability as numerous copies of data will be accessible from different database servers.

Replication also helps in protecting a database from the loss of a particular server.

MongoDB – HOW THE CONCEPT OF RELATION HELPS

Data can be recovered in case there are a hardware failure or service interruptions through this concept and approach.

As there are additional copies of the same data on various servers, a single server can prove worthy in case of disaster recovery, reporting, or act as backup.

MongoDB – ADVANTAGES, NEEDS OF DATA REPLICATION:

Here are some of the essential points that you can keep in mind before implementing the concept of data replication in MongoDB:

Helps in disaster recovery and backup of data.

24 by 7 (24 x 7) availability of every data.

Data can be kept safe through this redundant backup approach.

Minimizes downtime for maintenance.

MongoDB – DISADVANTAGES, NEEDS OF DATA REPLICATION:

Disadvantages of Data Replication:

- More space required.
- Redundant data is stored, so more space and server processing required.

MongoDB – How Replication Works

MongoDB makes use of a replica set to achieve replication.

Replica sets are collections of mongod instances which targets to host the identical dataset.

There is only one primary node associated with the replica set.

To perform this, a minimum of three nodes are required.

MongoDB – How Replication Works

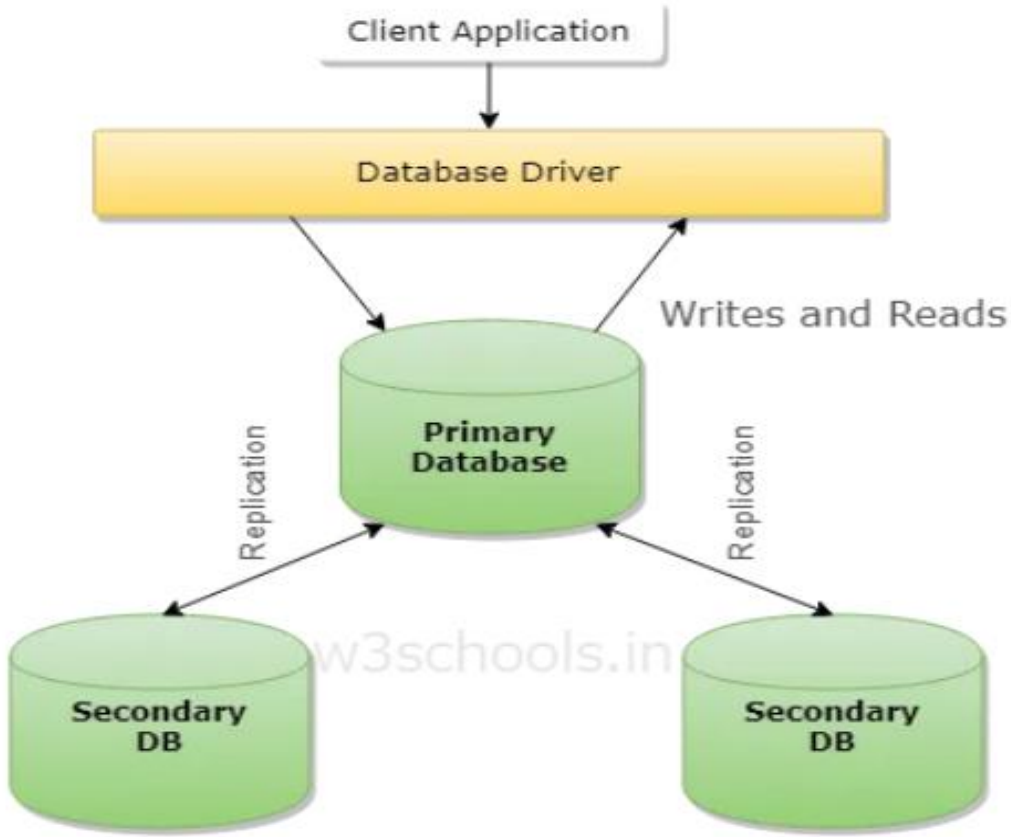
In this operation of replication, MongoDB assumes one node of replica set as the primary node and the remaining are secondary nodes.

From within the primary node, data gets replicated to secondary nodes.

New primary nodes get elected in case there is automatic maintenance or failover

MongoDB – How Replication Works

Here is a block diagram of how the replication in MongoDB takes place:



The Replication in MongoDB

MongoDB – How Replication Works

Now, you will convert a standalone MongoDB instance to a replica set. For this, you have to take the following measures first:

Stop your currently executing MongoDB server.

Restart the MongoDB server by indicating 'replSet' option. The necessary syntax is --replSet

MongoDB – How Replication Works

Syntax:

```
mongod --port "PORT" --dbpath "your_db_complete_path" --replSet "instanceName_of_Replica_set"
```

Example:

```
mongod --port 27017 --dbpath " C:\Program Files\MongoDB\Server\4.0\data" --replSet rs0
```

So the above command will:

- Initiate a mongod instance having the name rs0 and is running on port 27017.
- Initiate the command prompt and hook up to your mongod instance.
- Be executing the command **rs.initiate()** will help start a new replica set.
- Ensure the configuration of a replica set and execute the command **rs.conf()**. Also, employ the method/command **rs.status()** for checking the status of the replica set.

Include Members to Replica Set

MongoDB – How Replication Works

Start your mongod instances on different machines.

Then, initiate your mongo client application and use the method/command rs.add() which is used to include members. The syntax of this method is:

rs.add (hostname : PORT)

Here is an example of how to Implement the same

rs.add ("mongod1.net : 27017")

MongoDB – Concept of Shrding

A single set of data can be stored in multiple machines.

MongoDB supports such an essential feature concerning to database.

MongoDB fulfills the approach to meet the demand for data growth.

In this chapter, we learn about this MongoDB feature name - sharding.

MongoDB – What is Sharding

Sharding is an approach of distributing data across different machines. In other words, it can be said that the sharding concept is used for splitting large data sets into undersized data sets across several MongoDB instances.

This concept is employed for supporting deployments of data having huge data sets that perseveres high throughput operations.

MongoDB – What is Sharding

What are Shards

There are database situations where the data sets in MongoDB will grow so massive, that MongoDB statements, operations, and queries against such large data sets can cause a tremendous amount of CPU utilization on that particular server.

Such situations can be tacked in MongoDB using this concept of "sharding", where the data sets are split across numerous instances of MongoDB.

MongoDB – What is Sharding

That extensive collection of data sets can be actually split across several small-sized collections called "Shards".

But, logically, all the shards perform the work as a single collection.

Ways of Addressing System Growth

There is a parallel concept that is implemented for understanding sharding. System growth or scaling (which is used to increase the efficiency and working power of a system) can be addressed in 2 different ways. These are -

1. **Vertical Scaling** engages escalating the ability of a single server by making use of a more powerful CPU, where additional RAMs are incorporated, as well as the amount of storage has also increased. Restrictions
2. **Horizontal Scaling** engages segregating the data set of a single system as well as its workload over several servers, where similar servers are interconnected for increasing the capacity as per requirement. Here each server may not have high speed or capacity, but all of them together can handle the workload and provide efficient work than that of a single high-speed server.

Hence, sharding uses the concept of horizontal scaling to support the processing of large data sets.

Why the Sharding Concept Needs to Be Adopted

- Data replication can be done where master node absorbs and stores all the new data
- Simple queries can perform the task efficiently since data sets are segmented into small size
- A particular replica set has a restriction of 12 nodes only
- Memory cannot be outsized enough in case your dataset is large
- It becomes too expensive to have a vertical scaling
- Local disk won't be large enough and so multiple servers can be an alternative

Implementing the Concept of Sharding

Implementing the concept of sharding can be done with the use of clusters (which can be defined as a collection of MongoDB instances). Various shard components include:

- **The shard** - which is a MongoDB instance that embraces a part or subset of the data.
- **Config server** - which is a MongoDB instance holding the metadata regarding the cluster. It holds various MongoDB instances that are associated with shard data.
- **A router** - is mainly accountable for re-directing the instructions sent from the client to the authoritative servers.

Example of Sharding Cluster

1. Construct another database for your config server using the command:

```
mkdir /data/configdb
```

2. As configuration mode initiates the MongoDB instance. Let suppose; the server name is ServG:

```
mongod -configdb ServG: 27019
```

3. Initiate your mongos instance by identifying configuration server:

```
mongos -configdb ServG: 27019
```

4. Connect to the instance of Mongo from mongo shell:

```
mongo -host ServG -port 27017
```

5. When your other servers (Server K and Server R) are ready to be added with the cluster, type the command:

```
sh.addShard("ServK:27017")
sh.addShard("ServR:27017")
```

6. Allow the sharding for your database. If you want to shard the database techwriterDb the command will be:

```
sh.enableSharding(techwriterDb)
```

7. You can also allow sharding for your collection like this:

```
sh.shardCollection("db.techwriter" , { "writerID" : 1 , "writername" : "James Gosling"})
```

MongoDB – Export

- The data that has been prepared, i.e., inserted and updated might require to be pushed outside the database as a backup or for other similar purposes.
- MongoDB allows its users to perform that. In this chapter, you will learn about mongoexport command, which is used in MongoDB to export the data from the database.

MongoDB – Export

- What does export data mean?
- Data export is either an automatic or semi-automated task that changes a file from one format to another so that it can be used externally by other applications.
- It involves "interpreting" a set of data from one format used in a particular application into another.
- Such type of interpretation process gets accomplished either automatically or through the user's intervention.

MongoDB – Export

Exporting Data from MongoDB

MongoDB provides a utility called mongoexport to its users, through which users can export data from MongoDB databases to a JSON file format or a CSV file format. This utility is to be found in the **MongoDB's bin subfolder** (the path can be read like this: /mongodb/bin). As you run this feature, and while running, supply the database name, the name of the collection as well as the file you wish to export to, it performs the exporting task.

Exporting a MongoDB Collection to a JSON File

So, for performing the exporting of data, you have first to open a new terminal or command prompt window (cmd) and then write the appropriate command like this:

Example:

```
mongoexport --db techwriter --collection techwriter --out /data/dump/tw/tw.json
```

MongoDB – Export

- In case you discover that the above code is not running the **mongoexport** command,
- then it is for sure that you have either exited from the mongo utility or else you have opened a new Terminal or
- command prompt (cmd) window before executing this mongoexport, as it comes under a separate service.

MongoDB – Export

- The mongoexport statement written above will assume that your MongoDB bin folder is in your PATH, but if it is not in the place,

- then you have to specify the full path of the mongoexport file which will be as follows:

- /mongodb/bin/mongoexport or whatever path you have for the established directory of MongoDB.

MongoDB – Export

- In case, you do not offer a path for your exported file; the file gets created wherever (the path) you are residing at the time of running the command.

- Moreover giving the full path, or navigating to where you want your exported data file to be will make the task neat and easy to access.

MongoDB – Export

- **Exporting a MongoDB Collection to a CSV File**
- Till now, you have encountered how to export files in JSON format.
- Now, there is another file format which is proven to be the right way of representing data - the CSV (comma-separated value) file.

- For exporting your data to a CSV file, you have to add **-type = csv** to your command.

Example:

```
mongoexport --db techwriter --collection writers -type = csv --fields _id, writername --out
/data/dump/tw/techwriter.csv
```

Also, you can identify the fields in the documents for exporting. In this example, you have use `mongoexport` utility for exporting the techwriter collection to a CSV file. Exporting of _id and writername fields are done here. Also, note that the file name has a **.csv** extension.

Additional Attributes of Mongoexport Utility

You can make use of -limit, --sort and -skip attributes to your mongoexport statement like this:

```
mongoexport --db techwriter --collection writers --limit 4 --out /data/dump/tw.json
```

```
mongoexport --db techwriter --collection writers --limit 3 --sort '{_id: 2}' --out
/data/dump/tw_sorted.json
```

```
mongoexport --db techwriter --collection writers --limit 6 --sort '{_id: 2}' --skip 2 --out
/data/dump/tw_sorted_skipped.json
```