**MongoDB**

**Schema/Structure**

MongoDB is a NoSQL database solution that focuses on availability and scalability. The data is structured and stored in collections of JSON documents.

By not using a fixed data structure, MongoDB provides a comprehensive solution for high-volume storage in a modern distributed system.

There is no fixed schema for any collection, so there are no functions available in mongo shell to find the collection schema. Fixed Schema is applicable for RDBMS databases. In NoSQL DB, such as mongodb it is not required, but you can enforce same schema using your implementation logic, if required.

**Queries**

**Create DB –**

The use command is typically used to switch to a specific database. However, MongoDB creates a new database if one does not exist.

use customerdata

Output –

> use customerdata

switched to db customerdata

**Import Excel –**

mongoimport --host=127.0.0.1 -d customerdata -c customer --type csv --file .\train\_full.csv –headerline

Output –

2020-09-25T18:31:19.842+0000 [########################] customerdata.customer 2.89GB/2.89GB (100.0%)

2020-09-25T18:31:20.211+0000 5802400 document(s) imported successfully. 0 document(s) failed to import.

**Verify the DB –**

> show dbs

admin 0.000GB

config 0.000GB

customerdata 2.279GB

local 0.000GB

**Read Record –**

**>** > db.customer.find().limit(1)

{ "\_id" : ObjectId("5f6e2f5e4560fe42c274e53e"), "customer\_id" : "TCHWPBT", "gender" : "Male", "status\_x" : 1, "verified\_x" : 1, "created\_at\_x" : "2018-02-07 19:16:23", "updated\_at\_x" : "2018-02-07 19:16:23", "location\_number" : 0, "location\_type" : "Work", "latitude\_x" : -96.44, "longitude\_x" : -67.2, "id" : 4, "authentication\_id" : 118597, "latitude\_y" : -0.5884, "longitude\_y" : 0.7544, "vendor\_category\_en" : "Restaurants", "vendor\_category\_id" : 2, "delivery\_charge" : 0, "serving\_distance" : 6, "is\_open" : 1, "OpeningTime" : "11:00AM-11:30PM", "OpeningTime2" : "-", "prepration\_time" : 15, "commission" : 0, "is\_akeed\_delivering" : "Yes", "discount\_percentage" : 0, "status\_y" : 1, "verified\_y" : 1, "rank" : 11, "language" : "EN", "vendor\_rating" : 4.4, "sunday\_from\_time1" : "00:00:00", "sunday\_to\_time1" : "00:30:00", "sunday\_from\_time2" : "08:00:00", "sunday\_to\_time2" : "23:59:00", "monday\_from\_time1" : "00:00:00", "monday\_to\_time1" : "00:30:00", "monday\_from\_time2" : "08:00:00", "monday\_to\_time2" : "23:59:00", "tuesday\_from\_time1" : "00:00:00", "tuesday\_to\_time1" : "00:30:00", "tuesday\_from\_time2" : "08:00:00", "tuesday\_to\_time2" : "23:59:00", "wednesday\_from\_time1" : "00:00:00", "wednesday\_to\_time1" : "00:30:00", "wednesday\_from\_time2" : "08:00:00", "wednesday\_to\_time2" : "23:59:00", "thursday\_from\_time1" : "00:00:00", "thursday\_to\_time1" : "00:30:00", "thursday\_from\_time2" : "08:00:00", "thursday\_to\_time2" : "23:59:00", "friday\_from\_time1" : "00:00:00", "friday\_to\_time1" : "00:30:00", "friday\_from\_time2" : "10:00:00", "friday\_to\_time2" : "23:59:00", "saturday\_from\_time1" : "00:00:00", "saturday\_to\_time1" : "00:30:00", "saturday\_from\_time2" : "10:00:00", "saturday\_to\_time2" : "23:59:00", "primary\_tags" : "{\"primary\_tags\":\"4\"}", "open\_close\_flags" : 1, "vendor\_tag" : "2,4,5,8,91,22,12,24,16,23", "vendor\_tag\_name" : "Arabic,Breakfast,Burgers,Desserts,Free Delivery,Grills,Lebanese,Salads,Sandwiches,Shawarma", "one\_click\_vendor" : "Y", "country\_id" : 1, "city\_id" : 1, "created\_at\_y" : "2018-01-30 14:42:04", "updated\_at\_y" : "2020-04-07 15:12:43", "device\_type" : 3, "display\_orders" : 1, "location\_number\_obj" : 0, "id\_obj" : 4, "CID X LOC\_NUM X VENDOR" : "TCHWPBT X 0 X 4", "target" : 0 }

**Read Specific query –**

> db.customer.find({"CID X LOC\_NUM X VENDOR" : "TCHWPBT X 0 X 78"}).limit(1)

{ "\_id" : ObjectId("5f6e2f5e4560fe42c274e54b"), "customer\_id" : "TCHWPBT", "gender" : "Male", "status\_x" : 1, "verified\_x" : 1, "created\_at\_x" : "2018-02-07 19:16:23", "updated\_at\_x" : "2018-02-07 19:16:23", "location\_number" : 0, "location\_type" : "Work", "latitude\_x" : -96.44, "longitude\_x" : -67.2, "id" : 78, "authentication\_id" : 118675, "latitude\_y" : -0.555, "longitude\_y" : 0.1963, "vendor\_category\_en" : "Restaurants", "vendor\_category\_id" : 2, "delivery\_charge" : 0.7, "serving\_distance" : 15, "is\_open" : 0, "OpeningTime" : "11:00AM-11:00PM", "OpeningTime2" : "-", "prepration\_time" : 17, "commission" : 0, "is\_akeed\_delivering" : "Yes", "discount\_percentage" : 0, "status\_y" : 0, "verified\_y" : 1, "rank" : 11, "language" : "EN", "vendor\_rating" : 4.4, "sunday\_from\_time1" : "00:01:00", "sunday\_to\_time1" : "02:00:00", "sunday\_from\_time2" : "18:00:00", "sunday\_to\_time2" : "23:59:00", "monday\_from\_time1" : "00:01:00", "monday\_to\_time1" : "02:00:00", "monday\_from\_time2" : "18:00:00", "monday\_to\_time2" : "23:59:00", "tuesday\_from\_time1" : "00:01:00", "tuesday\_to\_time1" : "02:00:00", "tuesday\_from\_time2" : "18:00:00", "tuesday\_to\_time2" : "23:59:00", "wednesday\_from\_time1" : "00:01:00", "wednesday\_to\_time1" : "02:00:00", "wednesday\_from\_time2" : "18:00:00", "wednesday\_to\_time2" : "23:59:00", "thursday\_from\_time1" : "00:01:00", "thursday\_to\_time1" : "02:00:00", "thursday\_from\_time2" : "18:00:00", "thursday\_to\_time2" : "23:59:00", "friday\_from\_time1" : "00:01:00", "friday\_to\_time1" : "02:00:00", "friday\_from\_time2" : "18:00:00", "friday\_to\_time2" : "23:59:00", "saturday\_from\_time1" : "00:01:00", "saturday\_to\_time1" : "02:00:00", "saturday\_from\_time2" : "18:00:00", "saturday\_to\_time2" : "23:59:00", "primary\_tags" : "", "open\_close\_flags" : 1, "vendor\_tag" : "15,34,4,28,27,24,8", "vendor\_tag\_name" : "Pizzas,Italian,Breakfast,Soups,Pasta,Salads,Desserts", "one\_click\_vendor" : "Y", "country\_id" : 1, "city\_id" : 1, "created\_at\_y" : "2018-08-26 21:47:55", "updated\_at\_y" : "2020-03-31 22:16:15", "device\_type" : 3, "display\_orders" : 1, "location\_number\_obj" : 0, "id\_obj" : 78, "CID X LOC\_NUM X VENDOR" : "TCHWPBT X 0 X 78", "target" : 0 }

**Write One Customer –**

> db.customer.insertOne({"customer\_id" : "BITSTest", "gender" : "Male"})

{

"acknowledged" : true,

"insertedId" : ObjectId("5f6e3d651046f753ca46e49e")

}

Verification –

> db.customer.find({"customer\_id" : "BITSTest"})

{ "\_id" : ObjectId("5f6e3d651046f753ca46e49e"), "customer\_id" : "BITSTest", "gender" : "Male" }

**Insert Multiple Records –**

> db.customer.insertMany([{"customer\_id" : "BITSTest1", "gender" : "Male"},{"customer\_id" : "BITSTest2", "gender" : "Male"}])

{

"acknowledged" : true,

"insertedIds" : [

ObjectId("5f6e40861046f753ca46e4a0"),

ObjectId("5f6e40861046f753ca46e4a1")

]

}

**Update Already Existing Record –**

> db.customer.updateOne({"customer\_id" : "BITSTest1"},{$set: {"gender" :"Female"}})

{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }

\*gender was male as shown in previous query, moved to male

**Upsert - whenever you insert a value and it already exist then update would be performed. If the value does not already exist then it would get inserted.**

> db.customer.updateOne({"customer\_id" : "BITSTest4"},{$set: {"gender" :"Female"}}, **{ upsert : true })**

{

"acknowledged" : true,

"matchedCount" : 0,

"modifiedCount" : 0,

"upsertedId" : ObjectId("5f6e45f2576e8d226a5a8148")

}

**Delete a Specific Record –**

> try { db.customer.deleteOne({"customer\_id" : "BITSTest"});} catch(e) { print(e);}

{ "acknowledged" : true, "deletedCount" : 1 }

**GroupBy Records –**

> db.customer.aggregate([{"$group":{\_id:"gender", count:{$sum:1}}}])

\*we can also use db.customer.group

**OrderBy Records –**

> db.customer.aggregate([{$sortByCount:"$gender"}])

{ "\_id" : "Male", "count" : 3789102 }

{ "\_id" : "", "count" : 1705100 }

{ "\_id" : "Female", "count" : 308202 }

**Performance Considerations**

**Indexing-**

Without the index, the query would scan the whole collection. This means the query would go through each and every document, potentially pulling them into memory and analyzing them. This results in an expensive and very slow operation

**Restrictions on Sharded Collection**

Starting in MongoDB 3.0, an index cannot cover a query on a sharded collection when run against a mongos if the index does not contain the shard key, with the following exception for the \_id index: If a query on a sharded collection only specifies a condition on the \_id field and returns only the \_id field, the \_id index can cover the query when run against a mongos even if the \_id field is not the shard key.

**Query Selectivity –**

Query selectivity refers to how well the query predicate excludes or filters out documents in a collection. Query selectivity can determine whether or not queries can use indexes effectively or even use indexes at all. More selective queries match a smaller percentage of documents.

A **covered query** is a query that can be satisfied entirely using an index and does not have to examine any documents. Querying only the index can be much faster than querying documents outside of the index.

**Avoid Long-Running Queries -**

MongoDB uses a locking system to ensure data set consistency. If certain operations are long-running or a queue forms, performance will degrade as requests and operations wait for the lock. Similarly in some cases, the number of connections between the applications and the database can overwhelm the ability of the server to handle requests.

**Scaling –**

As MongoDB is a NoSQL DB, Scaling the DB is very easy compared to SQL DBs with predefined schemas. We can add more columns or rows easily without making hinderance to existing Schema.

**Upgrade –**

MongoDB can be easily upgraded to higher version by replacing the existing binaries with new binaries. All the version till now are backward compatible. Though from 4.4, tools are separated from installation package, this can downloaded and used with previous versions.

**Monitoring -**

1. mongostat will tell you how many time database operations such as insert, query, update, delete, etc. actually occur on the server. This will give a good idea on how much the load the server is handling and will indicate whether you need additional resources on the server or maybe additional servers to distribute the load.
2. mongotop tracks and reports the current read and write activity of a MongoDB instance, and reports these statistics on a per collection basis.
3. Other than this, mongodb also provides performance parameters and DB storage and availability (CPU, Disk, memory) metrics for monitoring

**Backup –**

Below are the backup mechanisms available from within MongoDB

1. Backup by Copying Underlying Data Files – This is probably the easiest mechanism , all that needs to be done is to copy the data files on which MongoDB resides and copy it to another location which ideally should be another server.
2. Backup a Database with mongodump - The mongodump tool reads data from a MongoDB database and creates high fidelity BSON files. What needs to be taken into consideration is that if the data set is large in volume, then mongodump can be very resource intensive, so then to mitigate this problem, the utility should be run on a secondary server.
3. MongoDB Cloud Manager Backup - MongoDB Cloud Manager continually backs up MongoDB replica sets and sharded clusters by reading the oplog data from the MongoDB environment. MongoDB Cloud Manager can create a point in time recovery by storing oplog data so that it can create a restore at any moment in time for a particular replica set or sharded cluster.

Similarly we can recover the DB from unexpected shutdown (**mongod –repair**) or from the backups as and when required.

**Setup and Space -**

MongoDB can’t be installed on Windows Servers, Mac OS, and most flavuors of Linux distributions. This can be installed as a package or standalone components. The Disk space required completely depends on the data, as a NoSQL db, mongodb configuration files are light weight with 50% RAM (of 1GB) or 256MB requirement as prerequisites.

**Performance Visualization using Compass**

**Reading a record using selective query -**

**A screenshot of a computer screen

Description automatically generated**

**Update operations**

**A screenshot of a computer screen

Description automatically generated**

**Time Analysis**

**Load Data –**

**CPU and Memory Utilization – 100% and 95% respectively**

**A screenshot of a computer screen

Description automatically generated**

**Time Taken – 14 Mins 12 Sec**

**OS – Windows Server 2019 (Hosted on AWS SG)**

**Excel Location – Local drive (same location as mongoimport)**

**RAM – 1 GB RAM (1 core CPU)**

**Query Executions –**

**Read query (with Index) – Less than a Second**

**Write Query (with 2 columns) – Less than a second**

**Upsert query – 37 Sec**

**Delete Query – 36 Sec**

**GroupBy – 33 Sec**

**Avg CPU Utilization during Query execution – 71%**