MongoDB – No SQL Database

MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability.

Un structured data

Document – contains 5 data members

Another document contains 4 data members

RDBMS No SQL

Database Database

Table Collection

Rows Documents

Columns Fields

Primary Key Primary Key (default \_id)

Join Embedded Document

SQL Commands MongoDB APIs

Background Service Background Service (mongod CLI)

SQL Plus, SSMS Compass (mongo cli)

Native Drivers

JDBC, ODBC, OLEDB MongoDB Packages, Monk

ORM (Hibernate, EF) ODM (Mongoose)

Where to use MongoDB?

Big Data, Content Management and Delivery,

Mobile and Social Infrastructure, User Data Management, Data Hub

To start the server

Launch node command prompt

Go to mogodb path generally

C:// program files/ mongodb /server/ version no/ bin

Run following command

Mongod –dbpath “name of the folder where databases would get created”

Launch another command prompt

Set the path of the mogodb

C:// program files/ mongodb /server/ version no/ bin

Run following command

Mongo - > to have mongo shell where you can write following commands

Create a Database

use DataBase\_Name

To insert records in the collection

db.movies.insert(object);

to insert multiple records

db.movies.insert(array of objects);

to show the collection’s data or to query the documents

db.movies.find().pretty();

drop the collection

db.collection\_name.drop()

to update any document or to save it

db.movies.update({name:'Drishyam 2'},{$set:{name:'RRR'}},{multi:true});

db.movies.save({ "\_id" : ObjectId("637b3b9d847058c54316a512"),"name":'New Movie'})

to remove the document based on the deletion criteria

db.movies.remove(deletion criteria)

db.movies.remove({“name”:”New Movie”});

Projection of the records

db.movies.find({},{KEY:1}); - only primary keys

only name of the movie and no \_id

db.movies.find({},{"name":1,\_id:0});

limiting records

db.movies.find().pretty().limit(2)

db.movies.find({},{"name":1,\_id:0}).limit(2);

sorting the records

db.collection\_name.find().sort({key:1})

sort ascending

db.movies.find().sort({"name":1});

sort descending

db.movies.find().sort({"name":-1});

For grouping add director to the all-movie documents

> db.movies.update({"name":"Fast & Furious 3"},{$set:{"director":"Vin Diesel"}});

For aggregation modifying database collection

db.movies.update({"name":"Fast & Furious 1"},{$set:{"likes":1000}}); adding likes for all the records

aggregation

Sum of number of movies director wise

> db.movies.aggregate([{$group:{\_id:"$director", num\_movies:{$sum:1}}}])

Average of likes director wise

db.movies.aggregate([{$group:{\_id:"$director", avg\_likes:{$avg:"$likes"}}}])

sum of likes director wiser

db.movies.aggregate([{$group:{\_id:"$director", avg\_likes:{$sum:"$likes"}}}])

min of likes director wise

db.movies.aggregate([{$group:{\_id:"$director", avg\_likes:{$min:"$likes"}}}])

max of likes director wise

db.movies.aggregate([{$group:{\_id:"$director", avg\_likes:{$max:"$likes"}}}])