Mongo Database

RDBMS ---------- MongoDB

Database ---------- Database

Table ---------- Collection

Tuple/Row ---------- Document

column ----------Field Table

Join Embedded ----------Documents

Primary Key ---------- Primary Key

ADVANTAGES

Data is stored in the form of JSON style

Index on any attribute

 Replication and high availability 

Auto-sharding 

Rich queries 

Fast in-place updates

 Professional support by MongoDB

INSTALLATION :

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

in etc/apt/sources.list

echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list

sudo apt-get update

apt-get install mongodb-10gen=2.2.3

sudo service mongodb start

sudo service mongodb stop

sudo service mongodb restart

mongo

COMMANDS :

stats about MongoDB server, type the command db.stats()

list of commands, type db.help()

MongoDB use DATABASE\_NAME is used to create database.

use DATABASE\_NAME

check your databases list : show dbs

Insert doc into db : db.movie.insert({"name":"tutorials point"})

db.dropDatabase() command is used to drop a existing database

db.createCollection(name, options) is used to create collection.

check the created collection by using the command show collections.

insert data into MongoDB collection,

>db.COLLECTION\_NAME.insert(document)

db.ships.insert({name:'USS Enterprise D',operator:'Starfleet',type:'Explorer',class:'Galaxy',crew:750,codes:[10,11,12]})

query data from MongoDB collection,

>db.COLLECTION\_NAME.find() ---- db.mycol.find().pretty()

db.ships.findOne() Finds one arbitrary document

db.ships.find().prettyPrint() Finds all documents and using nice formatting

db.ships.find({}, {name:true, \_id:false}) Shows only the names of the ships

db.ships.findOne({'name':'USS Defiant'}) Finds one document by attribute

find using operators :

$gt / $gte : greater than / greater than equals

db.ships.find({class:{$gt:’P'} String

2 $lt / $lte : lesser than / lesser than equals

db.ships.find({class:{$lte:’P'} Array

4 $exists : does an attribute exist or not

db.ships.find({type:{$exists:true}}) Binary Data

5 $regex : Perl-style pattern matching

db.ships.find({name:{$regex:’^USS\\sE’}}) Date

9 $type : search by type of an element

db.ships.find({name : {$type:2}})

AND in MongoDB

>db.mycol.find({key1:value1, key2:value2}).pretty()

>db.mycol.find({"by":"tutorials point","title": "MongoDB Overview"}).pretty()

OR in MongoDB

>db.mycol.find( { $or: [ {key1: value1}, {key2:value2} ] } ).pretty()

>db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]}).pretty()

MongoDB Update method

>db.COLLECTION\_NAME.update(SELECTIOIN\_CRITERIA, UPDATED\_DATA)

>db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})

MongoDB save method

save method replaces the existing document with the new document passed in save method

>db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA})

>db.mycol.save( { "\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic", "by":"Tutorials Point" } )

delete document

>db.COLLECTION\_NAME.remove(DELLETION\_CRITTERIA)

>db.mycol.remove({'title':'MongoDB Overview'})

delete just first record , out of multiple records :

>db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1)

mongodb projection :

selecting only necessary data rather than selecting whole of the data of a document.

>db.COLLECTION\_NAME.find({},{KEY:1})

>db.mycol.find({},{"title":1,\_id:0})

LIMIT DOCUMENTS

limit the records in MongoDB,

>db.COLLECTION\_NAME.find().limit(NUMBER)

>db.mycol.find({},{"title":1,\_id:0}).limit(2)

SKIP METHOD

accepts number type argument and used to skip number of documents.

>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

>db.mycol.find({},{"title":1,\_id:0}).limit(1).skip(1)

SORT METHOD :

sort documents in MongoDB, you need to use sort method.

>db.COLLECTION\_NAME.find().sort({KEY:1})

>db.mycol.find({},{"title":1,\_id:0}).sort({"title":-1})

1/-1 : asc /desc order

MONGODB INDEXING :

Indexes are special data structures, that store a small portion of the data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field as specified in index.

create an index :

>db.COLLECTION\_NAME.ensureIndex({KEY:1})

>db.mycol.ensureIndex({"title":1,"description":-1})

Creating an index

db.ships.ensureIndex({name : 1})

Dropping an index

db.ships.dropIndex({name : 1})

Creating a compound index

db.ships.ensureIndex({name : 1, operator : 1, class : 0})

Dropping a compound index

db.ships.dropIndex({name : 1, operator : 1, class : 0})

Creating a unique compound index

db.ships.ensureIndex({name : 1, operator : 1, class : 0}, {unique : true})

Indexes – Hints & Stats

db.ships.find ({'name':'USS Defiant'}).explain() : Explains index usage

./mongotop : Shows time spent per operations per collection

db.ships.stats() : Index statistics

./mongostat : Shows snapshot on the MongoDB system

MONGODB AGGREGATION :

s process data records and return computed results . group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

> db.mycol.aggregate([{$group : {\_id : "$by\_user", num\_tutorial : {$sum : 1}}}])

**$sum** Sums up the defined value from all documents in the collection.

db.mycol.aggregate [$group: id:" $byu ser " , numtutorial: $sum:" $likes " ]

**$avg** Calculates the average of all given values from all documents in the collection.

db.mycol.aggregate [$group: id:" $byu ser " , numtutorial: $avg:" $likes " ]

**$min** Gets the minimum of the corresponding values from all documents in the collection.

db.mycol.aggregate [$group: id:" $byu ser " , numtutorial: $min:" $likes " ]

**$max** Gets the maximum of the corresponding values from all documents in the collection.

db.mycol.aggregate [$group: id:" $byu ser " , numtutorial: $max:" $likes " ]

**$push** Inserts the value to an array in the resulting document.

db.mycol.aggregate [$group: id:" $byu ser " , url: $push:" $url " ]

**$addToSet** Inserts the value to an array in the resulting document but does not create duplicates. db.mycol.aggregate [$group: id:" $byu ser " , url: $addToSet:" $url " ]

**$first** Gets the first document from the db.mycol.aggregate source documents according to the grouping. Typically this makes only sense together with some previously applied

“$sort”-stage. [$group: id:" $byu ser " , firstu rl: $first:" $url " ]

**$last** Gets the last document from the source documents according to the grouping. Typically this makes only sense together with some previously applied “$sort”-stage. db.mycol.aggregate [$group: id:" $byu ser " , lastu rl: $last:" $url " ]

MONGODB REPLICATION:

Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability with multiple copies of data on different database servers, replication protects a database from the loss of a single server.

MongoDB achieves replication by the use of replica set.

replica set is a group of mongod instances that host the same data set.

Replica set can have only one primary node.

data replicates from primary to secondary node.

mongod --port "PORT" --dbpath "YOUR\_DB\_DATA\_PATH" --replSet "REPLICA\_SET\_INSTANCE\_NAME"

mongod --port 27017 --dbpath "D:\set up\mongodb\data" --replSet rs0

initiate a new replica set. : rs.initiate

Replica set configuration : rs.conf.

status of replica sete issue : rs.status.

MONGODB CREATE BACKUP

MONGODUMP

mongodump --host HOST\_NAME -- port PORT\_NUMBER :backup all databases of specified mongod instance.

mongodump --collection COLLECTION --db DB\_NAME: backup only specified collection of specified database.

MONGO RESTORE :

restore backup data mongodb's mongorestore

mongorestore

MONGODB SHARDING

Sharding is the process of storing data records across multiple machines and it is MongoDB's approach to meeting the demands of data growth. As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable read and write throughput. Sharding solves the problem with horizontal scaling.

* In replication all writes go to master node
*  Latency sensitive queries still go to master 
* Single replica set has limitation of 12 nodes 
* Memory can't be large enough when active dataset is big 
* Local Disk is not big enough 
* Vertical scaling is too expensive
* Shards: Shards are used to store data. They provide high availability and data consistency. In production environment each shard is a separate replica set.
* Config Servers: Config servers store the cluster's metadata
*  Query Routers: Query Routers are basically mongos instances, interface with client applications and direct operations to the appropriate shard.

MONGODB DEPLOYMENT

mongostat

This command checks the status of all running mongod instances and return counters of database operations.

D:\set up\mongodb\bin>mongostat

mongotop

track and report the read and write activity of MongoDB instance on a collection basis. By default mongotop returns information in each second,

D:\set up\mongodb\bin>mongotop

D:\set up\mongodb\bin>mongotop 30

MONGODB JAVA

Create a collection To create a collection, createCollection() method of com.mongodb.DB class is used.

DBCollection coll = db.createCollection("mycol");

To get/select a collection from the database, getCollection() method of com.mongodb.DBCollection class is used.

DBCollection coll = db.getCollection("mycol");

To insert a document into mongodb, insert() method of com.mongodb.DBCollection class is used.

coll.insert(doc);

To select all documents from the collection, find() method of com.mongodb.DBCollection class is used.

DBCursor cursor = coll.find();

int i=1;

while (cursor.hasNext())

{

System.out.println("Inserted Document: "+i);

System.out.println(cursor.next());

i++;

}

}c

atch(Exception e)

{

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

UPDATE DOCUMENT :

To update document from the collection, update() method of com.mongodb.DBCollection class is used.

DBObject updateDocument = cursor.next();

updateDocument.put("likes","200")

col1.update(updateDocument);

examples :

Replaces the whole document

db.ships.update({name : 'USS Prometheus'}, {name : 'USS Something'})

sets / changes certain attributes of a given document

db.ships.update({name : 'USS Something'}, {$set : {operator : 'Starfleet', class : 'Prometheus'}})

removes an attribute from a given document

db.ships.update({name : 'USS Something'}, {$unset : {operator : 1}})

DELETE DOCUMENT

To delete first document from the collection, you need to first select the documents using findOne() method and then remove method of com.mongodb.DBCollection class

DBObject myDoc = coll.findOne();

col1.remove(myDoc);

examples :

db.ships.remove({name : 'USS Prometheus'})

removes the document

db.ships.remove({name:{$regex:’^USS\\sE’}})

removes using operator