

Assignment No 2ADMS Lab

Roll No-33271

Problem Statement - Implement Map-Reduce and aggregation indexing with suitable example in MongoDB. Demonstrate the following

- i) Aggregation framework
- ii) Create and drop different type of index
- iii) explain () to show advantage of indices.

Objectives: 1) To understand the concept of map reduce in MongoDB

2) To implement the concept of document oriented DB

Theory:-

1) Map Reduce - It is a data processing paradigm for condensing large volume of data into useful aggregated results.

Syntax:-

```
db.collectionName.mapReduce(  
  map → function () { emit (this._id, this.amt); }  
  reduce → function (key, value) { return Array.sum(code)(code)  
  query → { query: {} }  
  output → out: "output name"  
}
```


In the above example, MongoDB applies the map phase to each input document. The map function emits key-value pairs. For those keys that have multiple values, MongoDB applies the reduce phase which collects and condenses the aggregated data. MongoDB then stores the result in collection.

All Map-Reduce functions in MongoDB are Javascript.

2) Aggregation

The operations process data records and return completed results. Aggregation group values from multiple documents together and can perform a range of operations on the grouped data to return single result.

The aggregation framework is modelled on the concept of data processing pipelines. Documents enter a multi-stage pipeline that transform the document into an aggregated result.

Example with syntax.

```
db.orders.aggregate([
  { $match: { status: "A" } },
  { $group: { _id: "$post_id", total: { sum:
    = "A" }
  } }
])
```

First stage - The \$match stage filters the document by the status field and passes to next stage.



PICT, PUNE

33271

Second stage - The \$group stage filters documents by cust_id status field and calculate sum of amt for each unique cust_id.

9) Indexing -

MongoDB uses indexing in order to make query processing more efficient without it. It must scan every document in MongoDB and retrieve only matched ones. Indices are special data structures that store some info related to documents. It becomes easy for MongoDB to find documents.

Syntax -

→ db.collection.name.createIndex({key:1})

'key' determines the field and 1 or -1 determines ascending or descending order.

→ db.collection.name.dropIndex({key:1}) will drop the index

→ db.collection.name.getIndexes() will retrieve all desc. of indices.

Screenshots of Implementation:

A) Map and Reduce Functions

1) Map and Reduce functions

```
Command Prompt - mongo
> var mapfunction = function(){emit(this._id,this.age)};
> var reducefunction = function(key,values){return Array.sum(values)};
> db.Driver.mapReduce(mapfunction,reducefunction,{"out":"Result1_mapReduce"});
{"result":"Result1_mapReduce","ok":1}
{"_id":1010,"value":42}
{"_id":1003,"value":83}
{"_id":1011,"value":20}
{"_id":1007,"value":54}

Command Prompt - mongo
> db.Driver.mapReduce(function(){emit(this.gender,this.age);},function(key,values){return Array.avg(values)},{query:{age:{$gt:23}},out:"Result2_mapReduce"});
{"result":"Result2_mapReduce","ok":1}
> db.Result2_mapReduce.find();
{"_id":"female","value":43.333333333333336}
{"_id":"male","value":45.666666666666664}
```

2) Aggregate Function

```
> var pipeline = [
... {$group:{_id:"$city"}},
... {$sort:{name:-1}},
... {$limit:2}
... ];
> db.Driver.aggregate(pipeline);
{"_id":"Bhopal"}
{"_id":"Chandigarh"}
> var pipeline = [ {$group:{_id:"$city"}}, {$sort:{name:-1}} ];
> db.Driver.aggregate(pipeline);
{"_id":"Bhopal"}
{"_id":"Chandigarh"}
{"_id":"Delhi"}
{"_id":"Pune"}
{"_id":"Mumbai"}
> db.Driver.aggregate([{$match:{city:"Mumbai"}},{$count:"Mumbaikars"}]);
{"Mumbaikars":3}
```

3) Explain Keyword

```
Command Prompt - mongo
> "city":"Delhi"
> db.Driver.find().explain()
{"queryPlanner":{"plannerVersion":1,"namespace":"test.Driver","indexFilterSet":false,"parsedQuery":{"city":"Delhi"}},
"queryHash":"8B3D4A88",
"planCacheKey":"8B3D4A88",
"winningPlan":{"stage":"COLLSCAN","direction":"forward"},
"rejectedPlans":[]},
"serverInfo":{"host":"LAPTOP-GIDV5390","port":27017,"version":"4.4.3","gitVersion":"913d6862acfb3446de1b116f4161360acd6fd13"},
"ok":1}
> db.Driver.explain()
explainable(test.Driver)
```

B) Index

1) Create Index

```
db.Driver.createIndex({age:-1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
db.Driver.find({age:22});
{ "_id" : ObjectId("612f084d4d01273e25c14a75"), "lno" : 1017, "name" : "Mayra", "age" : 22, "gender" : "female", "city" : "Delhi" }
```

2) Create Index with multiple attributes

```
db.Driver.createIndex({age:1,lno:-1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "ok" : 1
}
```

3) Get Indexes

```
db.Driver.getIndexes();
{
  {
    "v" : 2,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_"
  },
  {
    "v" : 2,
    "key" : {
      "age" : -1
    },
    "name" : "age_-1"
  },
  {
    "v" : 2,
    "key" : {
      "age" : 1,
      "lno" : -1
    },
    "name" : "age_1_lno_-1"
  }
}
```

4) Drop Index

```
db.Driver.dropIndex("age_1_lno_-1");
{"indexesWas" : 3, "ok" : 1}
```


Conclusion- Understood map-reduce, aggregation framework and concept of Indexing in MongoDB.