

Assignment No. 16

AIM: Implement the aggregation and indexing with suitable example in MongoDB.
Demonstrate the following:

- Aggregation framework
- Create and drop different types of indexes and explain () to show the advantage of the indexes.

Indexing:

Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient and requires the mongod to process a large volume of data.

Indexes are special data structures, which 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.

Indexing can be achieved on any field in a document using `ensureIndex ()` method.

ensureIndex () Syntax:

>DB.COLLECTION_NAME.ENSUREINDEX ({KEY: 1})

Here key is the name of field on which you want to create index and 1 is for ascending order. To create index in descending order one need to use -1.

Example:

>db.ensureIndex({eid:1})

This is simple/unique index.

We can define index on multiple fields as well resulting into composite index.

>db.ensureIndex({eid:1,ename:-1})

Aggregation:

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

In `sqlcount(*)` and `with group by` is an equivalent of MongoDB aggregation.

The aggregate () Method

For the aggregation in MongoDB one should use `aggregate()` method.

Syntax:

```
>db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)
```

Aggregate operation could be finding sum on particular field/key or taking an average or finding maximum or minimum values associated with particular field from various documents in a single collection.

Example:

Assume a sample collection with sample documents inserted as below:

```
>db.emp1.insert(
.. [{eid:101,ename:"sachin",dept:"IT",sal:40000},
.. {eid:110,ename:"pranav",dept:"IT",sal:60000},
.. {eid:105,ename:"manoj",dept:"COMP",sal:45000},
.. {eid:102,ename:"yogesh",dept:"FE",sal:20000}])
```

If we want to display all employees based on their departments along with average salary of particular department we can write following query:

```
>db.emp1.aggregate([{$group:{_id:"$dept",'avgsal':{$avg:"$sal"}}}])
```

The above query computes average salary per department.

Following query displays number of employees associated along with particular department.

```
db.emp1.aggregate([{$group:{_id:"$dept",'number of emp':{$sum:1}}}])
```

In the above example we have grouped documents by field dept and on each occurrence of dept previous value of sum is incremented.

The various available aggregation expressions are listed below:

Expression	Description
\$sum	Sums up the defined value from all documents in the collection.
\$avg	It computes average of defined value from all documents in the collection.
\$max	It displays maximum of defined value from all documents in the collection.
\$min	It displays minimum of defined value from all documents in the collection.

Conclusion:

Implemented the aggregation and indexing in MongoDB.

Output:

```
hp@hp-HP-Notebook:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
> db.population.insert({"id":1,"city":"pune","state":"maharashtra"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":2,"city":"mumbai","state":"maharashtra"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":3,"city":"nagpur","state":"maharashtra"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":4,"city":"nashik","state":"gujarat"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":5,"city":"ahmedabad","state":"gujarat"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":6,"city":"indore","state":"gujarat"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":7,"city":"satara","state":"tamilnadu"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":8,"city":"dhule","state":"tamilnadu"})
WriteResult({ "nInserted" : 1 })
> db.population.insert({"id":9,"city":"karad","state":"tamilnadu"})
WriteResult({ "nInserted" : 1 })
> db.population.find()
{ "_id" : ObjectId("5da9f907299476502a666eef"), "id" : 1, "city" : "pune", "state" : "maharashtra" }
{ "_id" : ObjectId("5da9f915299476502a666ef0"), "id" : 2, "city" : "mumbai", "state" : "maharashtra" }
{ "_id" : ObjectId("5da9f921299476502a666ef1"), "id" : 3, "city" : "nagpur", "state" : "maharashtra" }
{ "_id" : ObjectId("5da9f947299476502a666ef2"), "id" : 4, "city" : "nashik", "state" : "gujarat" }
{ "_id" : ObjectId("5da9f95d299476502a666ef3"), "id" : 5, "city" : "ahmedabad", "state" : "gujarat" }
```

```
{ "_id" : ObjectId("5da9f96f299476502a666ef4"), "id" : 6, "city" : "indore", "state" : "gujarat"
}
{ "_id" : ObjectId("5da9f98a299476502a666ef5"), "id" : 7, "city" : "satara", "state" :
"tamilnadu" }
{ "_id" : ObjectId("5da9f99b299476502a666ef6"), "id" : 8, "city" : "dhule", "state"
: "tamilnadu" }
{ "_id" : ObjectId("5da9f9a9299476502a666ef7"), "id" : 9, "city" : "karad", "state"
: "tamilnadu" }
> db.population.find().pretty()
{
  "_id" : ObjectId("5da9f907299476502a666eef"),
  "id" : 1,
  "city" : "pune",
  "state" : "maharashtra"
}
{
  "_id" : ObjectId("5da9f915299476502a666ef0"),
  "id" : 2,
  "city" : "mumbai",
  "state" : "maharashtra"
}
{
  "_id" : ObjectId("5da9f921299476502a666ef1"),
  "id" : 3,
  "city" : "nagpur",
  "state" : "maharashtra"
}
{
  "_id" : ObjectId("5da9f947299476502a666ef2"),
  "id" : 4,
  "city" : "nashik",
  "state" : "gujarat"
}
{
  "_id" : ObjectId("5da9f95d299476502a666ef3"),
  "id" : 5,
  "city" : "ahmedabad",
  "state" : "gujarat"
}
{
  "_id" : ObjectId("5da9f96f299476502a666ef4"),
  "id" : 6,
  "city" : "indore",
  "state" : "gujarat"
}
{
  "_id" : ObjectId("5da9f98a299476502a666ef5"),
  "id" : 7,
  "city" : "satara",
```

```

"state" : "tamilnadu"
}
{
  "_id" : ObjectId("5da9f99b299476502a666ef6"),
  "id" : 8,
  "city" : "dhule",
  "state" : "tamilnadu"
}
{
  "_id" : ObjectId("5da9f9a9299476502a666ef7"),
  "id" : 9,
  "city" : "karad",
  "state" : "tamilnadu"
}
>
db.population.aggregate([{$group:{"id":"$state","totalpopulation":{$sum:"$population"}}}])
assert: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
at Error (<anonymous>)
at doassert (src/mongo/shell/assert.js:11:14)
at
      Function.assert.commandWorked
(src/mongo/shell/assert.js:244:5)    at    DBCollection.aggregate
(src/mongo/shell/collection.js:1149:12) at (shell):1:15
2019-10-18T23:18:42.651+0530 Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed at src/mongo/shell/assert.js:13
> db.population.aggregate([{$group:{id:"$state","totalpopulation":{$sum:"$population"}}}])
assert: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",

```

```

"code" : 15951,
  "ok" : 0
} : aggregate failed
  at Error (<anonymous>)
  at doassert (src/mongo/shell/assert.js:11:14)
  at
    Function.assert.commandWorked
    (src/mongo/shell/assert.js:244:5)    at    DBCollection.aggregate
    (src/mongo/shell/collection.js:1149:12) at (shell):1:15
2019-10-18T23:19:30.034+0530 Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed at src/mongo/shell/assert.js:13
> db.population.update({"id":1},{ $set:"population":880000})
2019-10-18T23:22:33.592+0530 SyntaxError: Unexpected token :
> db.population.update({"id":1},{ $set:{ "population":880000} })
2019-10-18T23:22:52.030+0530 SyntaxError: Unexpected token :
> db.population.update({"id":1},{ $set:{ "population":880000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.find()
{ "_id" : ObjectId("5da9f907299476502a666eef"), "id" : 1, "city" : "pune", "state" :
"maharashtra", "population" : 880000 }
{ "_id" : ObjectId("5da9f915299476502a666ef0"), "id" : 2, "city" : "mumbai", "state" :
"maharashtra" }
{ "_id" : ObjectId("5da9f921299476502a666ef1"), "id" : 3, "city" : "nagpur", "state" :
"maharashtra" }
{ "_id" : ObjectId("5da9f947299476502a666ef2"), "id" : 4, "city" : "nashik", "state" :
"gujarat" }
{ "_id" : ObjectId("5da9f95d299476502a666ef3"), "id" : 5, "city" : "ahmedabad", "state"
: "gujarat" }
{ "_id" : ObjectId("5da9f96f299476502a666ef4"), "id" : 6, "city" : "indore", "state" : "gujarat"
}
{ "_id" : ObjectId("5da9f98a299476502a666ef5"), "id" : 7, "city" : "satara", "state" :
"tamilnadu" }
{ "_id" : ObjectId("5da9f99b299476502a666ef6"), "id" : 8, "city" : "dhule", "state"
: "tamilnadu" }
{ "_id" : ObjectId("5da9f9a9299476502a666ef7"), "id" : 9, "city" : "karad", "state"
: "tamilnadu" }
> db.population.update({"id":2},{ $set:{ "population":770000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.update({"id":3},{ $set:{ "population":70000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.update({"id":4},{ $set:{ "population":18000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.update({"id":5},{ $set:{ "population":90000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.update({"id":6},{ $set:{ "population":5000} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

```

```

> db.population.update({"id":7},{ $set: {"population":1000}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.update({"id":8},{ $set: {"population":15000}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.update({"id":9},{ $set: {"population":40000}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.population.find()
{ "_id" : ObjectId("5da9f907299476502a666eef"), "id" : 1, "city" : "pune", "state" :
"maharashtra", "population" : 880000 }
{ "_id" : ObjectId("5da9f915299476502a666ef0"), "id" : 2, "city" : "mumbai", "state" :
"maharashtra", "population" : 770000 }
{ "_id" : ObjectId("5da9f921299476502a666ef1"), "id" : 3, "city" : "nagpur", "state" :
"maharashtra", "population" : 70000 }
{ "_id" : ObjectId("5da9f947299476502a666ef2"), "id" : 4, "city" : "nashik", "state" :
"gujarat", "population" : 18000 }
{ "_id" : ObjectId("5da9f95d299476502a666ef3"), "id" : 5, "city" : "ahmedabad", "state" :
"gujarat", "population" : 90000 }
{ "_id" : ObjectId("5da9f96f299476502a666ef4"), "id" : 6, "city" : "indore", "state" :
"gujarat", "population" : 5000 }
{ "_id" : ObjectId("5da9f98a299476502a666ef5"), "id" : 7, "city" : "satara", "state" :
"tamilnadu", "population" : 1000 }
{ "_id" : ObjectId("5da9f99b299476502a666ef6"), "id" : 8, "city" : "dhule", "state" :
"tamilnadu", "population" : 15000 }
{ "_id" : ObjectId("5da9f9a9299476502a666ef7"), "id" : 9, "city" : "karad", "state" :
"tamilnadu", "population" : 40000 }
> db.population.aggragate({ $group: {"id": "$state", "totalpopulation": { $sum: "$population" } } })
2019-10-18T23:27:57.130+0530 TypeError: Property 'aggregate' of object test.population is not a
function
> db.population.aggregate({ $group: {"id": "$state", "totalpopulation": { $sum: "$population" } } })
assert: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
at Error (<anonymous>)
at doassert (src/mongo/shell/assert.js:11:14)
at Function.assert.commandWorked (src/mongo/shell/assert.js:244:5)
at DBCollection.aggregate (src/mongo/shell/collection.js:1149:12) at
(shell):1:15
2019-10-18T23:28:08.286+0530 Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",

```

```

"code" : 15951,
  "ok" : 0
} : aggregate failed at src/mongo/shell/assert.js:13
> db.population.aggregate({$group:{id:"$state","totalpopulation":{$sum:"$population"}}})
assert: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
at Error (<anonymous>)
at doassert (src/mongo/shell/assert.js:11:14)
at Function.assert.commandWorked (src/mongo/shell/assert.js:244:5)
at DBCollection.aggregate (src/mongo/shell/collection.js:1149:12) at
(shell):1:15
2019-10-18T23:29:00.743+0530 Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed at src/mongo/shell/assert.js:13
> db.population.aggregate([{$group:{id:"$state","totalpopulation":{$sum:"$population"}}}])
assert: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as an
expression inside an object",
  "code" : 15951,
  "ok" : 0
} : aggregate failed
at Error (<anonymous>)
at doassert (src/mongo/shell/assert.js:11:14)
at Function.assert.commandWorked
(src/mongo/shell/assert.js:244:5) at DBCollection.aggregate
(src/mongo/shell/collection.js:1149:12) at (shell):1:15
2019-10-18T23:30:58.029+0530 Error: command failed: {
  "errmsg" : "exception: the group aggregate field 'id' must be defined as
an expression inside an object",
  "code" : 15951,

```

```

"ok" : 0
} : aggregate failed at src/mongo/shell/assert.js:13
>
db.population.aggregate([{$group:{_id:"$state","totalpopulation":{$sum:"$population"}}}])
{ "_id" : "tamilnadu", "totalpopulation" : 56000 }
{ "_id" : "gujarat", "totalpopulation" : 113000 }
{ "_id" : "maharashtra", "totalpopulation" : 1720000 }
> db.population.aggregate([{$group:{_id:"$State","avgpopulation":{$avg:"$population"}}}])
{ "_id" : null, "avgpopulation" : 209888.888888888888 }
> db.population.aggregate([{$group:{_id:"$state","avgpopulation":{$avg:"$population"}}}])
{ "_id" : "tamilnadu", "avgpopulation" : 18666.6666666666668 }
{ "_id" : "gujarat", "avgpopulation" : 37666.6666666666664 }
{ "_id" : "maharashtra", "avgpopulation" : 573333.33333333334 }
> db.population.aggragate([{$group:{_id:"$state","minpopulation":{$min:"$population"}}}])
2019-10-18T23:36:46.901+0530 TypeError: Property 'aggregate' of object test.population is not a
function
> db.population.aggregate([{$group:{_id:"$state","minpopulation":{$min:"$population"}}}]) {
  "_id" : "tamilnadu", "minpopulation" : 1000 }
{ "_id" : "gujarat", "minpopulation" : 5000 }
{ "_id" : "maharashtra", "minpopulation" : 70000 }
>
db.population.aggregate([{$group:{_id:"$state","maxpopulation":{$max:"$population"}}}]) {
  "_id" : "tamilnadu", "maxpopulation" : 40000 }
{ "_id" : "gujarat", "maxpopulation" : 90000 }
{ "_id" : "maharashtra", "maxpopulation" : 880000 }
>
db.population.aggregate([{"$group":{"_id":"$state","maxpopulation":{$max:"$population"}}}]
)
{ "_id" : "tamilnadu", "maxpopulation" : 40000 }
{ "_id" : "gujarat", "maxpopulation" : 90000 }
{ "_id" : "maharashtra", "maxpopulation" : 880000 }
>
db.population.aggregate([{"$group":{"_id":"$state","maxpopulation":{$max:"$population"}}}])
{ "_id" : "tamilnadu", "maxpopulation" : 40000 }
{ "_id" : "gujarat", "maxpopulation" : 90000 }
{ "_id" : "maharashtra", "maxpopulation" : 880000 }
>
db.population.aggregate([{"$group":{"_id":"$state","maxpopulation":{$max:"$population"
}}}]
{ "_id" : "tamilnadu", "maxpopulation" : 40000 }
{ "_id" : "gujarat", "maxpopulation" : 90000 }
{ "_id" : "maharashtra", "maxpopulation" : 880000 }
>
db.population.aggregate({"$group":{"_id":"$state","maxpopulation":{$max:"$population" } })
{ "_id" : "tamilnadu", "maxpopulation" : 40000 }
{ "_id" : "gujarat", "maxpopulation" : 90000 }
{ "_id" : "maharashtra", "maxpopulation" : 880000 }

```

```

>
db.population.aggregate({"$group":{"_id":"$state","maxpopulation":{"$max":"$population"}}},{"totalpopulation":{"$gt:40000}}})
assert: command failed: {
  "errmsg" : "exception: Unrecognized pipeline stage name: 'totalpopulation'",
  "code" : 16436,
  "ok" : 0
} : aggregate failed
Error: command failed: {
  "errmsg" : "exception: Unrecognized pipeline stage name: 'totalpopulation'",
  "code" : 16436,
  "ok" : 0
} : aggregate failed
at Error (<anonymous>)
at doassert (src/mongo/shell/assert.js:11:14)
at Function.assert.commandWorked (src/mongo/shell/assert.js:244:5)
at DBCollection.aggregate (src/mongo/shell/collection.js:1149:12) at
(shell):1:15
2019-10-18T23:42:07.072+0530 Error: command failed: {
  "errmsg" : "exception: Unrecognized pipeline stage name: 'totalpopulation'",
  "code" : 16436,
  "ok" : 0
} : aggregate failed at src/mongo/shell/assert.js:13
>
db.population.aggregate({"$group":{"_id":"$state","maxpopulation":{"$max":"$population"}}},{$match:{"totalpopulation":{"$gt:40000}}})


>
db.population.aggregate([{"$group":{"_id":"$state","maxpopulation":{"$max":"$population"}}},{$match:{"totalpopulation":{"$gt:40000}}}]])
>
db.population.aggregate([{"$group":{"_id":"$state","maxpopulation":{"$max":"$population"}}},{$match:{"totalpopulation":{"$gt:400000}}}]])
>
db.population.aggregate([{"$group":{"_id":"$state","maxpopulation":{"$max":"$population"}}},{$match:{"totalpopulation":{"$lt:400000}}}]])
>
db.population.aggregate([{$group:{_id:"$state","totalpopulation":{"$sum":"$population"}}}]])
{ "_id" : "tamilnadu", "totalpopulation" : 56000 }
{ "_id" : "gujarat", "totalpopulation" : 113000 }
{ "_id" : "maharashtra", "totalpopulation" : 1720000 }
>
db.population.aggregate([{$group:{_id:"$state","totalpopulation":{"$sum":"$population"}}},{$match:{"totalpopulation":{"$gt:40000}}}]])
{ "_id" : "tamilnadu", "totalpopulation" : 56000 }
{ "_id" : "gujarat", "totalpopulation" : 113000 }
{ "_id" : "maharashtra", "totalpopulation" : 1720000 }
>
db.population.aggregate([{$group:{_id:"$state","totalpopulation":{"$sum":"$population"}}},{$match:{"totalpopulation":{"$lt:40000}}}]])

```

```

>
db.population.aggregate([{$group: {_id: "$state", "totalpopulation": {$sum: "$population"} }}, {$
match: {"totalpopulation": {$lte: 40000}} }])
>
db.population.aggregate([{$group: {_id: "$state", "totalpopulation": {$sum: "$population"} }}, {$
match: {"totalpopulation": {$lte: 56000}} }])
{ "_id" : "tamilnadu", "totalpopulation" : 56000 }
>
db.population.aggregate([{$group: {_id: "$state", "totalpopulation": {$sum: "$population"} }}, {$
match: {"totalpopulation": {$gt: 56000}} }])
{ "_id" : "gujarat", "totalpopulation" : 113000 }
{ "_id" : "maharashtra", "totalpopulation" : 1720000 }
> db.population.ensureIndex({"city": 1})
2019-10-18T23:49:40.362+0530 ReferenceError: population is not defined
> db.population.ensureIndex({"city": 1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
> db.population.getIndex
test.population.getIndex
> db.population.getIndex()
2019-10-18T23:50:15.573+0530 TypeError: Property 'getIndex' of object
test.population is not a function
> db.population.getIndexes()
2019-10-18T23:50:19.585+0530 TypeError: Property 'getIndexes' of object test.population is
not a function
> db.population.getIndexes()
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "test.population"
  },
  {
    "v" : 1,
    "key" : {
      "city" : 1
    },
    "name" : "city_1",
    "ns" : "test.population"
  }
]
> db.population.ensureIndex({"city": -1})

```



```
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 2,
  "numIndexesAfter" : 3,
  "ok" : 1
}
> db.population.getIndexes()
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "test.population"
  },
  {
    "v" : 1,
    "key" : {
      "city" : 1
    },
    "name" : "city_1",
    "ns" : "test.population"
  },
  {
    "v" : 1,
    "key" : {
      "city" : -1
    },
    "name" : "city_-1",
    "ns" : "test.population"
  }
]
```
