Assignment - B3

Title :- Implementation of aggregation & indexing with suitable example using MongoDB

Objective :- To understand aggregation & indexing in MongoDB.

Outcome :- Implementation of aggregation & indexing in mongoDB.

S/W & H/W :-
MongoDB, 64 bit OS.

Theory :-

Aggregation :-
Aggregation Operation Process data records & return computed results. Aggregation Operation group values from multiple documents together & can perform variety of operations on grouped data to return a single result.

MongoDB provides three ways to perform aggregation :-

2020/11/21 18:13

① The aggregation pipeline :-

→ sort :-

sorts the function or fields

1 - Ascending order.

-1 - Descending order.

Indexes in MongoDB :-

Indexes support the diffi-cient execution of queries in MongoDB. Indexes are special Data structures to store a small portion of collections data set in an easy to traverse form.

Indexes function :-

1) Creation : creates an Index.
ex. db collection createIndex ({"name":-1})

2) Display : getting all Indexes.
ex. db collection getIndexes().

3) Deletion : Delete the index.
ex. db collection dropIndex ({"name of Index"})

Types of Indexes :-

1) Style Fields :-

MongoDB supports the creati

2020/11/21 18:13

2) map-reduce function
3) single process aggregation.

→ 1] Aggregation Pipeline :-
documents enters multistage pipeline that transfers documents into aggregated result.

ex. db collection.aggregate([{ $match { condition } }, { $group { _id : "$file operation", { $operation : "amount" } } }]

We can use various operation along with $group in aggregation some of them are

1) $sum - returns sum
2) $avg - returns average
3) $min - returns minimum
4) $max - returns maximum.

→ 2] count :-
returns count of fields satisfying condition.

db.collection.count({'condition'})

① $first :- returns first element
② $last :- returns last element

2020/11/21 18:13

- DF user - Defined ascending & Descending indexes on a single field of document

ex. db collection, create Index ({"name": 1})

② Compound Index :-

MongoDB supports user-defined Indexes on multiple fields.

ex. db collection create Index ({ "name": 1, "age": -1 }).

3) Multi key - Index :-

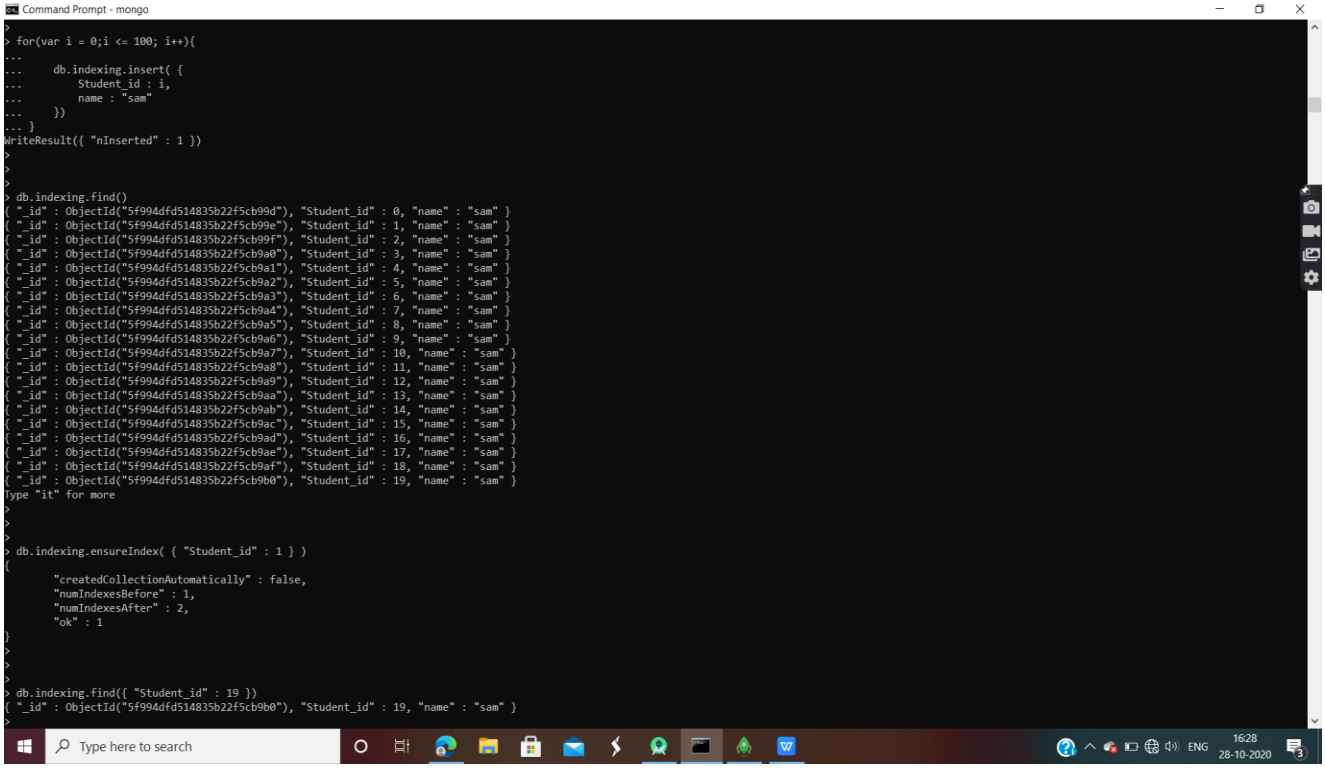
MongoDB uses multi-key Indexes to index content stored in arrays.

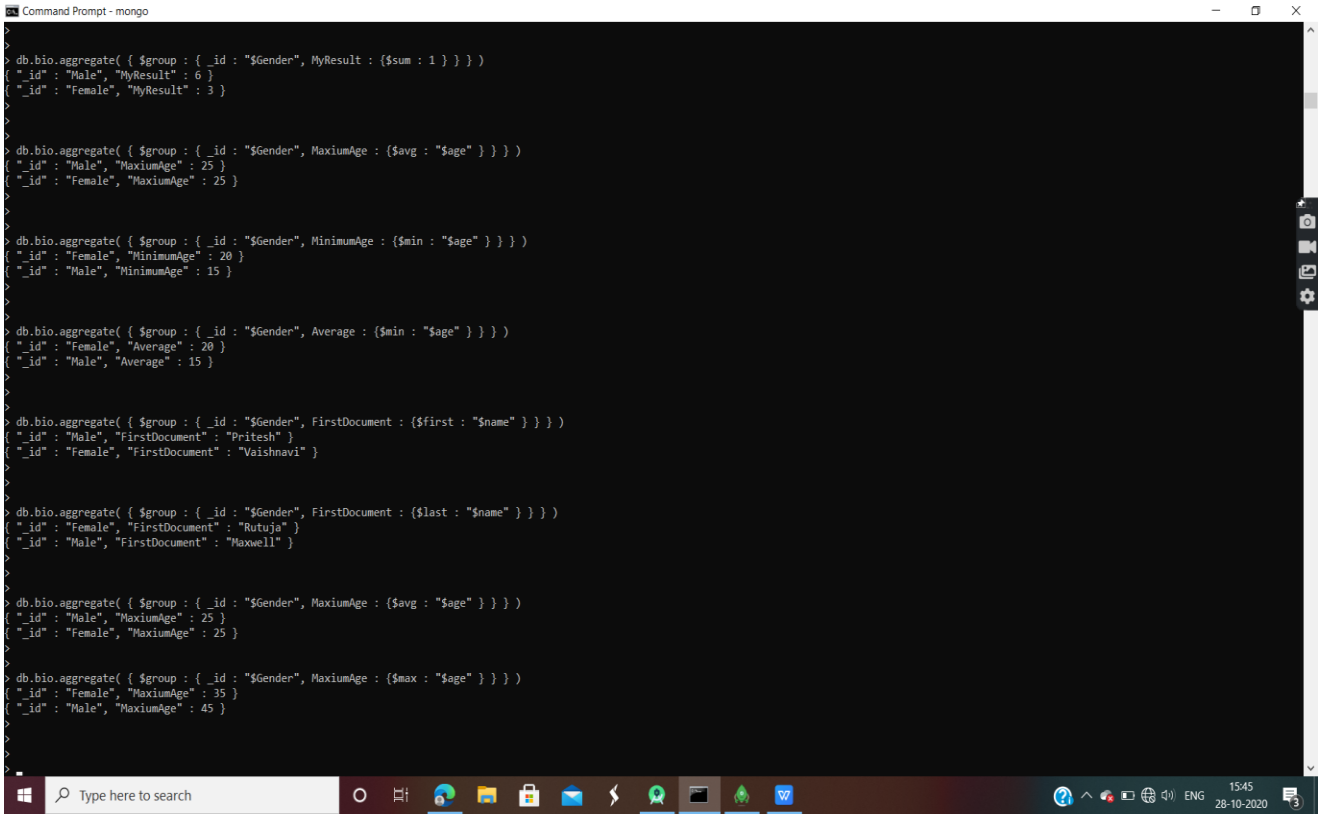
ex. db collection create Index ({ "name frame age" })

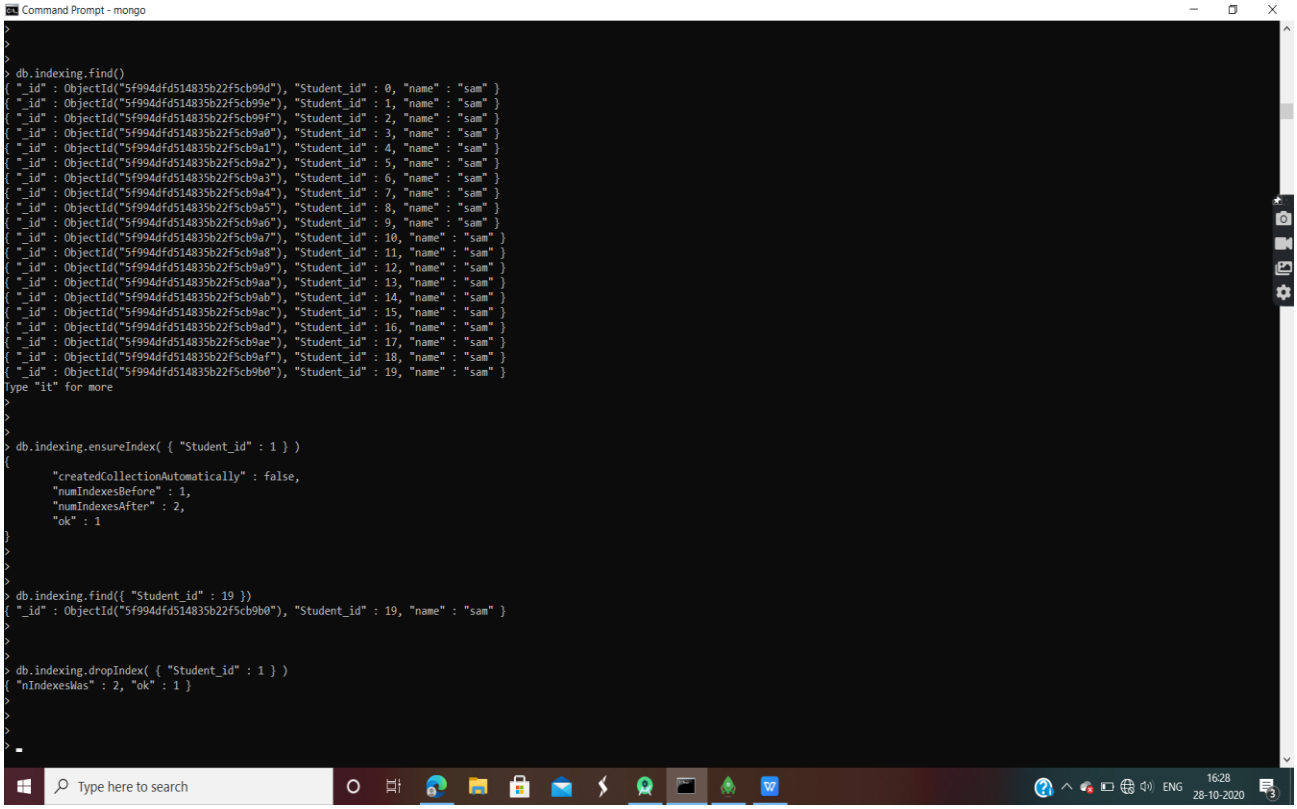
Conclusion :-

We successfully implemented the aggregation & Indexing in MongoDB.

2020/11/21 18:14


Command Prompt - mongo
> for(var i = 0;i <= 100; i++){
...
...   db.indexing.insert({
...   Student_id : i,
...   name : "sam"
...   })
... }
WriteResult({ "nInserted" : 1 })
> db.indexing.find()
{ "_id" : ObjectId("5f994dfd514835b22f5cb99d"), "Student_id" : 0, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb99e"), "Student_id" : 1, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb99f"), "Student_id" : 2, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a0"), "Student_id" : 3, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a1"), "Student_id" : 4, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a2"), "Student_id" : 5, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a3"), "Student_id" : 6, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a4"), "Student_id" : 7, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a5"), "Student_id" : 8, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a6"), "Student_id" : 9, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a7"), "Student_id" : 10, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a8"), "Student_id" : 11, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a9"), "Student_id" : 12, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9aa"), "Student_id" : 13, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ab"), "Student_id" : 14, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ac"), "Student_id" : 15, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ad"), "Student_id" : 16, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ae"), "Student_id" : 17, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9af"), "Student_id" : 18, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9b0"), "Student_id" : 19, "name" : "sam" }
Type "it" for more
> db.indexing.ensureIndex({ "Student_id" : 1 })
{
"createdCollectionAutomatically" : false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
"ok" : 1
}
> db.indexing.find({ "Student_id" : 19 })
{ "_id" : ObjectId("5f994dfd514835b22f5cb9b0"), "Student_id" : 19, "name" : "sam" }
Type here to search
ENG
16:28
28-10-2020


Command Prompt - mongo
> db.bio.aggregate({ $group : { _id : "$Gender", MyResult : {$sum : 1 } } })
{ "_id" : "Male", "MyResult" : 6 }
{ "_id" : "Female", "MyResult" : 3 }
> db.bio.aggregate({ $group : { _id : "$Gender", MaxiumAge : {$avg : "$age" } } })
{ "_id" : "Male", "MaxiumAge" : 25 }
{ "_id" : "Female", "MaxiumAge" : 25 }
> db.bio.aggregate({ $group : { _id : "$Gender", MinimumAge : {$min : "$age" } } })
{ "_id" : "Female", "MinimumAge" : 20 }
{ "_id" : "Male", "MinimumAge" : 15 }
> db.bio.aggregate({ $group : { _id : "$Gender", Average : {$min : "$age" } } })
{ "_id" : "Female", "Average" : 20 }
{ "_id" : "Male", "Average" : 15 }
> db.bio.aggregate({ $group : { _id : "$Gender", FirstDocument : {$first : "$name" } } })
{ "_id" : "Male", "FirstDocument" : "Pritesh" }
{ "_id" : "Female", "FirstDocument" : "Vaishnavi" }
> db.bio.aggregate({ $group : { _id : "$Gender", FirstDocument : {$last : "$name" } } })
{ "_id" : "Female", "FirstDocument" : "Rutuja" }
{ "_id" : "Male", "FirstDocument" : "Maxwell" }
> db.bio.aggregate({ $group : { _id : "$Gender", MaxiumAge : {$avg : "$age" } } })
{ "_id" : "Male", "MaxiumAge" : 25 }
{ "_id" : "Female", "MaxiumAge" : 25 }
> db.bio.aggregate({ $group : { _id : "$Gender", MaxiumAge : {$max : "$age" } } })
{ "_id" : "Female", "MaxiumAge" : 35 }
{ "_id" : "Male", "MaxiumAge" : 45 }
Type here to search
ENG
15:45
28-10-2020


Command Prompt - mongo
> db.indexing.find()
{ "_id" : ObjectId("5f994dfd514835b22f5cb99d"), "Student_id" : 0, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb99e"), "Student_id" : 1, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb99f"), "Student_id" : 2, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a0"), "Student_id" : 3, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a1"), "Student_id" : 4, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a2"), "Student_id" : 5, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a3"), "Student_id" : 6, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a4"), "Student_id" : 7, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a5"), "Student_id" : 8, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a6"), "Student_id" : 9, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a7"), "Student_id" : 10, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a8"), "Student_id" : 11, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9a9"), "Student_id" : 12, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9aa"), "Student_id" : 13, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ab"), "Student_id" : 14, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ac"), "Student_id" : 15, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ad"), "Student_id" : 16, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9ae"), "Student_id" : 17, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9af"), "Student_id" : 18, "name" : "sam" }
{ "_id" : ObjectId("5f994dfd514835b22f5cb9b0"), "Student_id" : 19, "name" : "sam" }
type "it" for more
> db.indexing.ensureIndex({ "Student_id" : 1 })
{
"createdCollectionAutomatically" : false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
"ok" : 1
}
> db.indexing.find({ "Student_id" : 19 })
{ "_id" : ObjectId("5f994dfd514835b22f5cb9b0"), "Student_id" : 19, "name" : "sam" }
> db.indexing.dropIndex({ "Student_id" : 1 })
{ "nIndexesWas" : 2, "ok" : 1 }
Type here to search
ENG
16:28
28-10-2020