MongoDB. Home Task 1

1. Install MongoDB

Result:

Mongod.exe

2018-01-12T16:35:35.929+0300 I NETWORK [initandlisten] waiting for connections on port 27017

2. Import Restaurants Collection

Follow these steps to import restaurants collection to you local data base:

2.1. Save restaurants.jsonon your PC

Result

saved to d:\restaurants.json

2.2. Run local instance of MongoDB • Assuming you want to use default data directory and port for the instance run mongodwithout any parameters

Result:

Mongod.exe

2018-01-12T16:35:35.929+0300 I NETWORK [initandlisten] waiting for connections on port 27017

2.3. Use mongoimport(it’s in MongoDB installation folder) to import the collection to the database • Assuming you run local MongoDB on the default port the following command should create “restaurants” collection in “frontcamp” database

Query:

mongoimport --dbfrontcamp --collection restaurants --file d:\restaurants.json

Result:

2018-01-12T17:09:54.997+0300 connected to: localhost

2018-01-12T17:09:56.034+0300 imported 25359 documents

2.4. Verify that collection was correctly imported • Assuming local MongoDB instance uses the default port, run mongo without any parameters

• Switch to frontcampdatabase

Query :

usefrontcamp

Result:

switched to dbfrontcamp

3. Querying Restaurants Collection

Query:

db.restaurants.findOne()

Result:

>db.restaurants.findOne()

{

"\_id" :ObjectId("5a58c1b37063c1ce46365f58"),

"address" : {

"building" : "2780",

"coord" : [

-73.98241999999999,

40.579505

],

"street" : "Stillwell Avenue",

"zipcode" : "11224"

},

"borough" : "Brooklyn",

"cuisine" : "American",

"grades" : [

{

"date" : ISODate("2014-06-10T00:00:00Z"),

"grade" : "A",

"score" : 5

},

{

"date" : ISODate("2013-06-05T00:00:00Z"),

"grade" : "A",

"score" : 7

},

{

"date" : ISODate("2012-04-13T00:00:00Z"),

"grade" : "A",

"score" : 12

},

{

"date" : ISODate("2011-10-12T00:00:00Z"),

"grade" : "A",

"score" : 12

}

],

"name" : "Riviera Caterer",

"restaurant\_id" : "40356018"

}

Answer the following questions and include both query and the result (if applicable) into your report:

3.1. How many “Chinese” (cuisine) restaurants are in “Queens” (borough)?

Query:

>db.restaurants.find({borough: 'Queens',cuisine: "Chinese"}).count();

Result:

728

3.2. What is the \_id of the restaurant which has the grade with the highest ever score?

Query:

>db.restaurants.find({},{\_id:1}).sort({'grades.score':-1}).limit(1)

Result:

{ "\_id" : ObjectId("5a58c1b37063c1ce463660b9") }

3.3. Add a grade { grade: "A", score: 7, date: ISODate() } to every restaurant in “Manhattan” (borough).

Query:

db.restaurants.updateMany({borough : 'Manhattan'}, {$push: {'grades': { grade: "A", score: 7, date: ISODate() } }})

Result:

{ "acknowledged" : true, "matchedCount" : 10259, "modifiedCount" : 10259 }

3.4. What are the names of the restaurants which have a grade at index 8 with score lessthen 7? Use projection to include only names without \_id.

Query:

>db.restaurants.find({'grades.8.score' :{$lt : 7} },{\_id:0,name:1})

Result:

{ "name" : "Silver Krust West Indian Restaurant" }

{ "name" : "Pure Food" }

3.5. What are \_id and borough of “Seafood” (cuisine) restaurants which received at least one “B” grade in period from 2014-02-01 to 2014-03-01? Use projection to include only \_id and borough.

Query:

db.restaurants.find({$and:[{cuisine: 'Seafood'},{"grades" : {$elemMatch: {"grade": "B","date": {$gte: new ISODate('2014-02-01'),$lt: new ISODate('2014-03-01')}}}}]},{\_id:1,borough:1})

Result:

{ "\_id" : ObjectId("5a58c1b37063c1ce463694c6"), "borough" : "Bronx" }

{ "\_id" : ObjectId("5a58c1b37063c1ce4636973a"), "borough" : "Manhattan" }

4. Indexing Restaurants Collection

Note: you may use MongoDB Compass for this task if you want to

Create the following indexes:

4.1. Create an index which will be used by this query and provide proof (from explain() or Compass UI) that the index is indeed used by the winning plan:

Query:

db.restaurants.createIndex({ name: 1 })

Result:

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

Query:

db.restaurants.find({ name: "Glorious Food" }).explain()

Result:

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"name" : {

"$eq" : "Glorious Food"

}

},

"winningPlan" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"name" : 1

},

"indexName" : "name\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"name" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"name" : [

"[\"Glorious Food\", \"Glorious Food\"]"

]

}

}

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

>

4.2. Drop index from task 4.1

Query:

>db.restaurants.dropIndex("name\_1");

Result:

{ "nIndexesWas" : 2, "ok" : 1 }

4.3. Create an index to make this query **covered** and provide proof (from explain() or Compass UI) that it is indeed covered: db.restaurants.find({ restaurant\_id: "41098650" }, { \_id: 0, borough: 1 })

Query before index:

>db.restaurants.find({ restaurant\_id: "41098650" }, { \_id: 0, borough: 1 }).explain("executionStats");

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"restaurant\_id" : {

"$eq" : "41098650"

}

},

"winningPlan" : {

"stage" : "PROJECTION",

"transformBy" : {

"\_id" : 0,

"borough" : 1

},

"inputStage" : {

"stage" : "COLLSCAN",

"filter" : {

"restaurant\_id" : {

"$eq" : "41098650"

}

},

"direction" : "forward"

}

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 1,

"executionTimeMillis" : 38,

"totalKeysExamined" : 0,

"totalDocsExamined" : 25359,

"executionStages" : {

"stage" : "PROJECTION",

"nReturned" : 1,

"executionTimeMillisEstimate" : 30,

"works" : 25361,

"advanced" : 1,

"needTime" : 25359,

"needYield" : 0,

"saveState" : 198,

"restoreState" : 198,

"isEOF" : 1,

"invalidates" : 0,

"transformBy" : {

"\_id" : 0,

"borough" : 1

},

"inputStage" : {

"stage" : "COLLSCAN",

"filter" : {

"restaurant\_id" : {

"$eq" : "41098650"

}

},

"nReturned" : 1,

"executionTimeMillisEstimate" : 30,

"works" : 25361,

"advanced" : 1,

"needTime" : 25359,

"needYield" : 0,

"saveState" : 198,

"restoreState" : 198,

"isEOF" : 1,

"invalidates" : 0,

"direction" : "forward",

"docsExamined" : 25359

}

}

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

>

Query create index:

>db.restaurants.createIndex({ restaurant\_id: 1 })

Result:

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

Query after index:

>db.restaurants.find({ restaurant\_id: "41098650" }, { \_id: 0, borough: 1 }).explain("executionStats");

Result

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"restaurant\_id" : {

"$eq" : "41098650"

}

},

"winningPlan" : {

"stage" : "PROJECTION",

"transformBy" : {

"\_id" : 0,

"borough" : 1

},

"inputStage" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"restaurant\_id" : 1

},

"indexName" : "restaurant\_id\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"restaurant\_id" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"restaurant\_id" : [

"[\"41098650\", \"41098650\"]"

]

}

}

}

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 1,

"executionTimeMillis" : 19,

"totalKeysExamined" : 1,

"totalDocsExamined" : 1,

"executionStages" : {

"stage" : "PROJECTION",

"nReturned" : 1,

"executionTimeMillisEstimate" : 10,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"transformBy" : {

"\_id" : 0,

"borough" : 1

},

"inputStage" : {

"stage" : "FETCH",

"nReturned" : 1,

"executionTimeMillisEstimate" : 10,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"docsExamined" : 1,

"alreadyHasObj" : 0,

"inputStage" : {

"stage" : "IXSCAN",

"nReturned" : 1,

"executionTimeMillisEstimate" : 10,

"works" : 2,

"advanced" : 1,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"keyPattern" : {

"restaurant\_id" : 1

},

"indexName" : "restaurant\_id\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"restaurant\_id" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"restaurant\_id" : [

"[\"41098650\", \"41098650\"]"

]

},

"keysExamined" : 1,

"seeks" : 1,

"dupsTested" : 0,

"dupsDropped" : 0,

"seenInvalidated" : 0

}

}

}

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

>

4.4. Create a **partial** index on cuisine field which will be used only when filtering on borough equal to “Staten Island”:

Query create index:

>db.restaurants.createIndex({ borough: 1, cuisine: 1 },{ partialFilterExpression: {$and:[ {borough: { $eq: "Staten Island"}},{cuisine :{$exists:true}}]}})

Result:

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 1,

"numIndexesAfter" : 2,

"ok" : 1

}

db.restaurants.find({ borough: "Staten Island", cuisine: "American" }) – uses index

>db.restaurants.find({ borough: "Staten Island", cuisine: "American" }).explain()

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"$and" : [

{

"borough" : {

"$eq" : "Staten Island"

}

},

{

"cuisine" : {

"$eq" : "American"

}

}

]

},

"winningPlan" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"borough" : 1,

"cuisine" : 1

},

"indexName" : "borough\_1\_cuisine\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"borough" : [ ],

"cuisine" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : true,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"borough" : [

"[\"Staten Island\", \"Staten Island\"]"

],

"cuisine" : [

"[\"American\", \"American\"]"

]

}

}

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

db.restaurants.find({ borough: "Staten Island", name: "Bagel Land" }) – does not use index

>db.restaurants.find({ borough: "Staten Island", name: "Bagel Land" }).explain()

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"$and" : [

{

"borough" : {

"$eq" : "Staten Island"

}

},

{

"name" : {

"$eq" : "Bagel Land"

}

}

]

},

"winningPlan" : {

"stage" : "COLLSCAN",

"filter" : {

"$and" : [

{

"borough" : {

"$eq" : "Staten Island"

}

},

{

"name" : {

"$eq" : "Bagel Land"

}

}

]

},

"direction" : "forward"

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

db.restaurants.find({ borough: "Queens", cuisine: "Pizza" }) – does not use index

>db.restaurants.find({ borough: "Queens", cuisine: "Pizza" }).explain()

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"$and" : [

{

"borough" : {

"$eq" : "Queens"

}

},

{

"cuisine" : {

"$eq" : "Pizza"

}

}

]

},

"winningPlan" : {

"stage" : "COLLSCAN",

"filter" : {

"$and" : [

{

"borough" : {

"$eq" : "Queens"

}

},

{

"cuisine" : {

"$eq" : "Pizza"

}

}

]

},

"direction" : "forward"

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

>

5. Create an index to make query from task 3.4 **covered** and provide proof (from explain() or Compass UI) that it is indeed covered

>db.restaurants.find({'grades.8.score' :{$lt : 7} },{\_id:0, name:1}).explain('executionStats');

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"grades.8.score" : {

"$lt" : 7

}

},

"winningPlan" : {

"stage" : "PROJECTION",

"transformBy" : {

"\_id" : 0,

"name" : 1

},

"inputStage" : {

"stage" : "COLLSCAN",

"filter" : {

"grades.8.score" : {

"$lt" : 7

}

},

"direction" : "forward"

}

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 2,

"executionTimeMillis" : 86,

"totalKeysExamined" : 0,

"totalDocsExamined" : 25359,

"executionStages" : {

"stage" : "PROJECTION",

"nReturned" : 2,

"executionTimeMillisEstimate" : 81,

"works" : 25361,

"advanced" : 2,

"needTime" : 25358,

"needYield" : 0,

"saveState" : 199,

"restoreState" : 199,

"isEOF" : 1,

"invalidates" : 0,

"transformBy" : {

"\_id" : 0,

"name" : 1

},

"inputStage" : {

"stage" : "COLLSCAN",

"filter" : {

"grades.8.score" : {

"$lt" : 7

}

},

"nReturned" : 2,

"executionTimeMillisEstimate" : 71,

"works" : 25361,

"advanced" : 2,

"needTime" : 25358,

"needYield" : 0,

"saveState" : 199,

"restoreState" : 199,

"isEOF" : 1,

"invalidates" : 0,

"direction" : "forward",

"docsExamined" : 25359

}

}

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}

Create index:

db.restaurants.createIndex({ 'grades.8.score': 1})

{

"createdCollectionAutomatically" : false,

"numIndexesBefore" : 3,

"numIndexesAfter" : 4,

"ok" : 1

}

Query with index:

>db.restaurants.find({'grades.8.score' :{$lt : 7} },{\_id:0, name:1}).explain('executionStats');

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"grades.8.score" : {

"$lt" : 7

}

},

"winningPlan" : {

"stage" : "PROJECTION",

"transformBy" : {

"\_id" : 0,

"name" : 1

},

"inputStage" : {

"stage" : "FETCH",

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"grades.8.score" : 1

},

"indexName" : "grades.8.score\_1",

"isMultiKey" : true,

"multiKeyPaths" : {

"grades.8.score" : [

"grades"

]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"grades.8.score" : [

"[-inf.0, 7.0)"

]

}

}

}

},

"rejectedPlans" : [ ]

},

"executionStats" : {

"executionSuccess" : true,

"nReturned" : 2,

"executionTimeMillis" : 14,

"totalKeysExamined" : 2,

"totalDocsExamined" : 2,

"executionStages" : {

"stage" : "PROJECTION",

"nReturned" : 2,

"executionTimeMillisEstimate" : 10,

"works" : 3,

"advanced" : 2,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"transformBy" : {

"\_id" : 0,

"name" : 1

},

"inputStage" : {

"stage" : "FETCH",

"nReturned" : 2,

"executionTimeMillisEstimate" : 10,

"works" : 3,

"advanced" : 2,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"docsExamined" : 2,

"alreadyHasObj" : 0,

"inputStage" : {

"stage" : "IXSCAN",

"nReturned" : 2,

"executionTimeMillisEstimate" : 10,

"works" : 3,

"advanced" : 2,

"needTime" : 0,

"needYield" : 0,

"saveState" : 0,

"restoreState" : 0,

"isEOF" : 1,

"invalidates" : 0,

"keyPattern" : {

"grades.8.score" : 1

},

"indexName" : "grades.8.score\_1",

"isMultiKey" : true,

"multiKeyPaths" : {

"grades.8.score" : [

"grades"

]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : false,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"grades.8.score" : [

"[-inf.0, 7.0)"

]

},

"keysExamined" : 2,

"seeks" : 1,

"dupsTested" : 2,

"dupsDropped" : 0,

"seenInvalidated" : 0

}

}

}

},

"serverInfo" : {

"host" : "EPBYMINW1652",

"port" : 27017,

"version" : "3.6.0",

"gitVersion" : "a57d8e71e6998a2d0afde7edc11bd23e5661c915"

},

"ok" : 1

}