Homework: MongoDB (Part 1)

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

***Query:*** *db.restaurants.find({borough: "Queens", cuisine: "Chinese"}).count()*

***Result:*** 728; or

***Query:*** *db.restaurant.find({$and: [{brough: "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({"grades.score": {$gt:100}}, {\_id: 1})*

***Result:*** ObjectId("5a5c76a2c58bbaf0e39a0f05")

**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: {"date": ISODate(), "grade": "A", "score": 7}}});*

**3.4.** What are the names of the restaurants which havea grade at index 8 with score less then 7? Use projection to include only names without \_id.

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

***Result:*** 1. Silver Krust West Indian Restaurant

2. 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({cuisine: "Seafood", grades: { $elemMatch: {date: {$gte: ISODate("2014-02-01"), $lte: ISODate("2014-03-01")}, grade: "B" }}}, {borough: 1})*

***Result:*** { "\_id" : ObjectId("5a5c76a2c58bbaf0e39a4312"), "borough" : "Bronx" }

{ "\_id" : ObjectId("5a5c76a2c58bbaf0e39a458c"), "borough" : "Manhattan" }

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

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

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

{

"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" : "EPBYGOMW0483",

"port" : 27017,

"version" : "3.6.2",

"gitVersion" : "489d177dbd0f0420a8ca04d39fd78d0a2c539420"

},

"ok" : 1

}

**4.2.** Drop index from task 4.1

***Query:*** *db.restaurants.dropIndex({"name": 1})*

***Result:*** db.restaurants.getIndexes()

[

{

"v" : 2,

"key" : {

"\_id" : 1

},

"name" : "\_id\_",

"ns" : "frontcamp.restaurants"

}

]

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

***Query:*** *db.restaurants.createIndex({"restaurant\_id": 1});*

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

{

"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" : [ ]

},

"serverInfo" : {

"host" : "EPBYGOMW0483",

"port" : 27017,

"version" : "3.6.2",

"gitVersion" : "489d177dbd0f0420a8ca04d39fd78d0a2c539420"

},

"ok" : 1

}

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

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

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

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

**Query:** *db.restaurants.createIndex({"cuisine": 1}, {partialFilterExpression: {borough: "Staten Island"}})*

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",

"filter" : {

"borough" : {

"$eq" : "Staten Island"

}

},

"inputStage" : {

"stage" : "IXSCAN",

"keyPattern" : {

"cuisine" : 1

},

"indexName" : "cuisine\_1",

"isMultiKey" : false,

"multiKeyPaths" : {

"cuisine" : [ ]

},

"isUnique" : false,

"isSparse" : false,

"isPartial" : true,

"indexVersion" : 2,

"direction" : "forward",

"indexBounds" : {

"cuisine" : [

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

]

}

}

},

"rejectedPlans" : [ ]

},

"serverInfo" : {

"host" : "EPBYGOMW0483",

"port" : 27017,

"version" : "3.6.2",

"gitVersion" : "489d177dbd0f0420a8ca04d39fd78d0a2c539420"

},

"ok" : 1

}

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" : "EPBYGOMW0483",

"port" : 27017,

"version" : "3.6.2",

"gitVersion" : "489d177dbd0f0420a8ca04d39fd78d0a2c539420"

},

"ok" : 1

}

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" : "EPBYGOMW0483",

"port" : 27017,

"version" : "3.6.2",

"gitVersion" : "489d177dbd0f0420a8ca04d39fd78d0a2c539420"

},

"ok" : 1

}

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

***Query:*** *db.restaurants.createIndex({"grades.8.score": 1})*

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

{

"queryPlanner" : {

"plannerVersion" : 1,

"namespace" : "frontcamp.restaurants",

"indexFilterSet" : false,

"parsedQuery" : {

"grades.8.score" : {

"$lt" : 7

}

},

"winningPlan" : {

"stage" : "PROJECTION",

"transformBy" : {

"name" : 1,

"\_id" : 0

},

"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" : [ ]

},

"serverInfo" : {

"host" : "EPBYGOMW0483",

"port" : 27017,

"version" : "3.6.2",

"gitVersion" : "489d177dbd0f0420a8ca04d39fd78d0a2c539420"

},

"ok" : 1

}