Deepak Katta; dkatta@kent.edu

ADVANCED DATABASE SYSTEM DESIGN
PROJECT

In this project I perform some basic operations on Mongo DB in which we load large number of records into database, along with basic operations I also demonstrated *Query performance*, I followed *indexing method to enhance the Query execution performance and I compared Query performances before and after indexing.*

In any database, it is important to have less query execution time so that data can be retrieved fast. Generally, In databases we have to work with large data when we posed a query to database it should search all the data records and return the results according to our query specifications, this process takes some time called query execution time. There are some techniques to minimize query execution time called Query Optimization Techniques. In this project I used Indexing to minimize the Query Execution Time.

Indexing is creating indexes to support our Query; Indexes store the small portion of the collection's data, which facilitates to search easily for specified criteria in Query.

It is very interesting to do indexing on Mongo DB and analyze Query performance because Mongo DB is NoSQL Database and unlike TinyDB it can handle large no. Of data records. In Mongo DB Data is stored in collections and collections contains documents.

In the class, I have learnt the basic information on functionality of NoSQL Databases. Till now I just used SQL Databases for projects through this class for the first time I came to know about NoSQL Databases and how to use them, while doing this project I had an opportunity to work practically with NoSQL Database Mongo DB, I learn all basic and intermediate commands in Mongo DB.

For this project, I used restaurants collection, which has the information regarding restaurants like restaurant_id, restaurant name, Address and reviews.

Totally there are 25,359 records (Documents)

```
> db.restaurants.find().count()
25359
>
```

We can use count ( ) to find how many records are retrieved for that query.

Basic Queries on Mongo DB:

The following Query retrieves restaurants whose cuisine is American.

```
> db.restaurants.find( {cuisine: "American "} ).pretty().count()
6183
>
```

The above query retrieved 6183 records out of 25,359 records.

```
> db.restaurants.find( {cuisine: "American "} ).pretty()
{
        "_id" : ObjectId("565f265a5929401562fd5093"),
        "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
                },
```

Pretty ( ) – Pretty method shows ordered output. It prints Document in collection in ordered way.

The following query retrieves the American cuisine restaurants whose restaurant_id is greater than 40000000.

```
> db.restaurants.find( {cuisine: "American ", "restaurant_id": { $gt: "40000000" } } ).pretty().count()
6183
>
```

Totally, 6183 records are retrieved.

```
> db.restaurants.find( {cuisine: "American ", "restaurant_id": { $gt: "40000000" } } ).pretty()
{
        "_id" : ObjectId("565f265a5929401562fd5093"),
        "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"
}
{
        "_id" : ObjectId("565f265a5929401562fd5095"),
        "address" : {
                "building" : "8825",
                "coord" : [
                        -73.8803827,
                        40.7643124
                ],
                "street" : "Astoria Boulevard",
                "zipcode" : "11369"
        },
        "borough" : "Queens",
        "cuisine" : "American ",
        "grades" : [
                {
                        "date" : ISODate("2014-11-15T00:00:00Z"),
```

The below query retrieves the ‘American” cuisine restaurants whose restaurant_id is lessthan 45000000.

```
> db.restaurants.find( {cuisine: "American ", "restaurant_id": { $lt: "45000000" } } ).pretty().count()
4932
>
```

Total, 4932 records are retrieved from database.

The below query retrieves the restaurants those having A grade with 10 score (Good Rating in reviews)

```
> db.restaurants.find({"grades":{ $elemMatch:{ "grade": "A", "score":10 } }} ).pretty().count()
7954
>
```

Total 7954 records are retrieved from database.


```
> db.restaurants.find({"grades":{ $elemMatch:{ "grade": "A", "score":10 } }} ).pretty()
{
        "_id" : ObjectId("565f265a5929401562fd5091"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2011-03-10T00:00:00Z"),
                        "grade" : "B",
                        "score" : 14
                }
        ],
        "name" : "Morris Park Bake Shop",
        "restaurant_id" : "30075445"
}
{
        "_id" : ObjectId("565f265a5929401562fd5095"),
        "address" : {
                "building" : "8825",
                "coord" : [
                        -73.8803827,
                        40.7643124
                ],
                "street" : "Astoria Boulevard",
                "zipcode" : "11369"
        },
```

The below Query retrieves the restaurants which are located in street “Avenue. J” and having Grade A and score “12”. (Restaurants, which are located in Avenue. J with Good Ratings)

```
> db.restaurants.find({'address.street': "Avenue J", "grades":{$elemMatch:{"grade": "A", "score":12 } }}).pretty()
{
        "_id" : ObjectId("565f265a5929401562fd51d7"),
        "address" : {
                "building" : "1223",
                "coord" : [
                        -73.9633849,
                        40.625169
                ],
                "street" : "Avenue J",
                "zipcode" : "11230"
        },
        "borough" : "Brooklyn",
        "cuisine" : "Jewish/Kosher",
        "grades" : [
                {
                        "date" : ISODate("2014-05-08T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2013-04-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 12
                },
                {
                        "date" : ISODate("2012-08-16T00:00:00Z"),
                        "grade" : "B",
                        "score" : 27
                },
                {
                        "date" : ISODate("2012-02-23T00:00:00Z"),
                        "grade" : "C",
                        "score" : 44
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
                        "grade" : "P",
                        "score" : 40
                },
                {
                        "date" : ISODate("2011-05-31T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                }
        ],
        "name" : "Kosher Delight",
        "restaurant_id" : "40371419"
}
{
        "_id" : ObjectId("565f265a5929401562fd5413"),
        "address" : {
                "building" : "1416",
                "coord" : [
                        -73.96171079999999,
```

```
> db.restaurants.find({'address.street': "Avenue J", "grades":{$elemMatch:{"grade": "A", "score":12 } }}).pretty().count()
8
>
```

The above Query retrieved only 8 records from 25,359 records; In this way we can short list our search

Till now we displayed all fields in documents but Mongo DB allows us to limit the no. Of fields to be displayed in document.
The below Query only retrieves names of restaurants which are located in zip code areas 11210 and 11230

```
> db.restaurants.find({'address.zipcode': {$in: ["11210", "11230"]}}, {"name": 1, "_id":0}).pretty()
{ "name" : "Kosher Bagel Hole" }
{ "name" : "Mama Lucia" }
{ "name" : "Kosher Delight" }
{ "name" : "Luigi'S Pizza" }
{ "name" : "Garden Of Eat-In" }
{ "name" : "Mcdonald Avenue Diner" }
{ "name" : "Jerusalem Pizza" }
{ "name" : "Lords Bakery" }
{ "name" : "Angel Flakes Patties" }
{ "name" : "Vinnie'S Pizzeria/ Luncheonette" }
{ "name" : "Metropolitan Food Cafe Of Brooklyn College" }
{ "name" : "Gourmet On J" }
{ "name" : "Crystal Manor" }
{ "name" : "Anna'S Cafe" }
{ "name" : "Hillel Bagels" }
{ "name" : "Kent Theatre" }
{ "name" : "Kids 'N' Action" }
{ "name" : "Dynasty" }
{ "name" : "Burger King" }
{ "name" : "Pizza Bagel Burger" }
Type "it" for more
>
```

The above query only displays names of the restaurants.

```
> db.restaurants.find({'address.zipcode': {$in: ["11210", "11230"]}}, {"name": 1, "_id":0}).pretty().count()
197
>
```

Totally, 197 records are retrieved by above query.

The below Query retrieves the data records of restaurants those have word “Indian” in their names: (This is just like “LIKE” Query in SQL).

```
> db.restaurants.find({"name": /Indian/},{"name": 1, 'address.street': 1}).pretty()
{
        "_id" : ObjectId("565f265a5929401562fd5461"),
        "address" : {
                "street" : "East 50 Street"
        },
        "name" : "Indigo Indian Cuisine"
}
{
        "_id" : ObjectId("565f265b5929401562fd561d"),
        "address" : {
                "street" : "East  233 Street"
        },
        "name" : "Jackie'S West Indian Bakery"
}
{
        "_id" : ObjectId("565f265b5929401562fd56df"),
        "address" : {
                "street" : "East    6 Street"
        },
        "name" : "Raj Mahal Indian Restaurant"
}
{
        "_id" : ObjectId("565f265b5929401562fd5719"),
        "address" : {
                "street" : "Gerard Avenue"
        },
        "name" : "Feeding Tree Style West Indian Restaurant"
}
{
        "_id" : ObjectId("565f265b5929401562fd5725"),
        "address" : {
                "street" : "Lexington Avenue"
        },
        "name" : "Pongal Indian Cuisine"
}
{
        "_id" : ObjectId("565f265b5929401562fd5792"),
        "address" : {
                "street" : "Fort Washington Avenue"
        },
        "name" : "Kismat Indian Cuisine"
}
{
        "_id" : ObjectId("565f265b5929401562fd580c"),
        "address" : {
                "street" : "Union Turnpike"
        },
        "name" : "Santoor Indian Restaurant"
}
{
        "_id" : ObjectId("565f265b5929401562fd588d"),
        "address" : {
                "street" : "Hillside Avenue"
        },
```

The above query also displays the street in which restaurant is located.

```
> db.restaurants.find({"name": /Indian/},{"name": 1, 'address.street': 1}).pretty().count()
106
>
```

Totally, 106 records are retrieved. i.e 106 restaurants have word “Indian” in their names.

Query Execution Analysis: In order to know the Query Execution Performance Mongo DB provides built in function explain( ) when we send “executionStats” as argument into function it provides detailed information regarding Query Execution.

```
> db.restaurants.find( {cuisine: "American "} ).pretty().explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "test.restaurants",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "cuisine" : {
                                "$eq" : "American "
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "cuisine" : {
                                        "$eq" : "American "
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 6183,
                "executionTimeMillis" : 16,
                "totalKeysExamined" : 0,
                "totalDocsExamined" : 25359,
                "executionStages" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "cuisine" : {
                                        "$eq" : "American "
                                }
                        },
                        "nReturned" : 6183,
                        "executionTimeMillisEstimate" : 10,
                        "works" : 25361,
                        "advanced" : 6183,
                        "needTime" : 19177,
                        "needFetch" : 0,
                        "saveState" : 198,
                        "restoreState" : 198,
                        "isEOF" : 1,
                        "invalidates" : 0,
                        "direction" : "forward",
                        "docsExamined" : 25359
                }
        },
        "serverInfo" : {
                "host" : "Deepaks-MacBook-Air.local",
                "port" : 27017,
                "version" : "3.0.7",
                "gitVersion" : "nogitversion"
        },
```

<u>Without Indexing:</u>

Without Indexing Mongo DB performs ‘Collection Scan” (“COLLSCAN”) means it will scan or search all documents in the collection, in the above scenario Mongo DB searched 25,359 documents and returned 6183 records which took 16 Milliseconds to execute the given query. Even though number of records to be returned are 6183 Mongo DB had performed search operation on all records.

Index creation:
In the below Query I created index over field “cuisine”:

```
> db.restaurants.createIndex({cuisine: 1})
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.restaurants.find( {cuisine: "American "} ).pretty().explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "test.restaurants",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "cuisine" : {
                                "$eq" : "American "
                        }
                },
                "winningPlan" : {
                        "stage" : "FETCH",
                        "inputStage" : {
                                "stage" : "IXSCAN",
                                "keyPattern" : {
                                        "cuisine" : 1
                                },
                                "indexName" : "cuisine_1",
                                "isMultiKey" : false,
                                "direction" : "forward",
                                "indexBounds" : {
                                        "cuisine" : [
                                                "[\"American \", \"American \"]"
                                        ]
                                }
                        }
                },
                "rejectedPlans" : [ ]
        },
        "executionStats" : {
                "executionSuccess" : true,
                "nReturned" : 6183,
                "executionTimeMillis" : 7,
                "totalKeysExamined" : 6183,
                "totalDocsExamined" : 6183,
                "executionStages" : {
                        "stage" : "FETCH",
                        "nReturned" : 6183,
                        "executionTimeMillisEstimate" : 0,
                        "works" : 6184,
                        "advanced" : 6183,
                        "needTime" : 0,
                        "needFetch" : 0,
                        "saveState" : 48,
                        "restoreState" : 48,
                        "isEOF" : 1,
                        "invalidates" : 0,
```

With Indexing:
We created an Index over field “cuisine” If we have indexes Mongo DB performs “IXSCAN” – Index Scan and search for American in cuisine and retrieve the matching records. Mongo DB had scanned only 6183 records and retrieved all the records, which took less execution time than before, with indexing it, only took 7 Milli seconds to retrieve the same number of results.

```
> db.restaurants.find({'address.street': "Avenue J", "grades":{$elemMatch:{"grade": "A", "score":12 } }}).pretty().explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "test.restaurants",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [
                                {
                                        "grades" : {
                                                "$elemMatch" : {
                                                        "$and" : [
                                                                {
                                                                        "grade" : {
                                                                                "$eq" : "A"
                                                                        }
                                                                },
                                                                {
                                                                        "score" : {
                                                                                "$eq" : 12
                                                                        }
                                                                }
                                                        ]
                                                }
                                        }
                                },
                                {
                                        "address.street" : {
                                                "$eq" : "Avenue J"
                                        }
                                }
                        ]
                },
 },
 "executionStats" : {
         "executionSuccess" : true,
         "nReturned" : 8,
         "executionTimeMillis" : 57,
         "totalKeysExamined" : 0,
         "totalDocsExamined" : 25359,
         "executionStages" : {
                 "stage" : "COLLSCAN",
                 "filter" : {
                         "$and" : [
                                 {
                                         "grades" : {
                                                 "$elemMatch" : {
                                                         "$and" : [
                                                                 {
                                                                         "grade" : {
                                                                                 "$eq" : "A"
                                                                         }
                                                                 },
                                                                 {
                                                                         "score" : {
                                                                                 "$eq" : 12
                                                                         }
                                                                 }
                                                         ]
                                                 }
                                         }
                                 },
                                 {
                                         "address.street" : {
                                                 "$eq" : "Avenue J"
                                         }
                                 }
                         ]
                 },
```

We did not create any index on the fields that present in the above query. In this Query we have street, grades and score. The execution time is 57 Milliseconds and retrieved 8 records because it has to check all the three constraints on three fields. We did not create any index that combine these three fields so the Mongo DB simple Performed COLLSCAN (Searched all documents in collection).

To overcome this type of situations Mongo DB allows us to create "Compound Index".
Single index on multiple fields is Compound Index.

```
> db.restaurants.createIndex( {"address.street": 1, "grades.grade": 2, "grades.score": 3} )
{
        "createdCollectionAutomatically" : false,
        "numIndexesBefore" : 1,
        "numIndexesAfter" : 2,
        "ok" : 1
}
> db.restaurants.getIndexes()
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },
                "name" : "_id_",
                "ns" : "test.restaurants"
        },
        {
                "v" : 1,
                "key" : {
                        "address.street" : 1,
                        "grades.grade" : 2,
                        "grades.score" : 3
                },
                "name" : "address.street_1_grades.grade_2_grades.score_3",
                "ns" : "test.restaurants"
        }
```

In the above snap I created compound index on fields address, grade and score now I executed same query

```
> db.restaurants.find({'address.street': "Avenue J", "grades":{$elemMatch:{"grade": "A", "score":12 } }}).pretty().explain("executionStats")
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "test.restaurants",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "$and" : [
                                {
                                        "grades" : {
                                                "$elemMatch" : {
                                                        "$and" : [
                                                                {
                                                                        "grade" : {
                                                                                "$eq" : "A"
                                                                        }
                                                                },
                                                                {
                                                                        "score" : {
                                                                                "$eq" : 12
                                                                        }
                                                                }
                                                        ]
                                                }
                                        }
                                },
```

```
"executionStats" : {
        "executionSuccess" : true,
        "nReturned" : 8,
        "executionTimeMillis" : 0,
        "totalKeysExamined" : 8,
        "totalDocsExamined" : 8,
        "executionStages" : {
                "stage" : "KEEP_MUTATIONS",
                "nReturned" : 8,
                "executionTimeMillisEstimate" : 0,
                "works" : 9,
                "advanced" : 8,
                "needTime" : 0,
                "needFetch" : 0,
                "saveState" : 0,
                "restoreState" : 0,
                "isEOF" : 1,
                "invalidates" : 0,
                "inputStage" : {
                        "stage" : "FETCH",
                        "filter" : {
                                "grades" : {
                                        "$elemMatch" : {
                                                "$and" : [
                                                        {
                                                                "grade" : {
                                                                        "$eq" : "A"
                                                                }
                                                        },
                                                        {
                                                                "score" : {
                                                                        "$eq" : 12
                                                                }
                                                        }
                                                ]
                                        }
                                }
                        },
                        "nReturned" : 8,
```

After creating compound index the Mongo DB almost took negligible time to execute the query. It just examined 8 records those are associated with that index and returned the results.

In this way Indexing improves the Query performance by optimizing the search to be done.

Weather in SQL Databases or NoSQL Databases Indexing is one of the good technique to improve Query Execution performance.