

**Curso:** "Big Data I: Cloud Computing y almacenamiento masivo de datos"

**Parte No SQL:** Entrega NoSQL MongoDB

**Estudiante:** Edgar Pérez Rivera

**Correo:** edjperez@correo.ugr.es

**DNI:** PA1099417

**Ejercicio 1:** Crear en vuestra base de datos MongoDB la colección "restaurants" desde el archivo /var/tmp/restaurantes1.json conforme se indica en la transparencia 44 de la presentación sobre NoSQL. Elaborar el código MapReduce que resuelva la consulta:

"Obtener, para el barrio "Manhattan" y para cada calle "street" el par de restaurantes de cocina española y el par de restaurantes de cocina italiana más próximos, mostrando el la calle, la cocina, el nombre, la distancia entre ellos y la cantidad de restaurantes evaluados para cada ("street", cocina); para aquellos restaurantes que hayan tenido un "score" menor que 12 en alguna ocasión".

#### Resolución:

Para realizar esta parte de la entrega, utilizaré la máquina virtual de Bitnami conectada con el servicio de Stuido 3T en MongoDb, la consulta requiere que se haga, en la función de mapeo, se filtran los datos según el barrio ("borough"), el tipo de cocina ("cuisine") y la puntuación ("score") de las evaluaciones. Se emiten las calles ("address.street"), tipos de cocina, nombres y coordenadas de los restaurantes que cumplen con los criterios de filtrado. En la función de reducción, se calcula la distancia euclidiana entre los pares de restaurantes y se identifican los dos más cercanos para cada calle y tipo de cocina. Los resultados se almacenan en una estructura de datos que contiene los pares de restaurantes más cercanos. En estos resultados de algunas capturas los devolvemos en línea, para comprobar los resultados, pero las indicaciones nos sugiere realizar guardo en almacenarse en sendas colecciones.

#### Consulta:

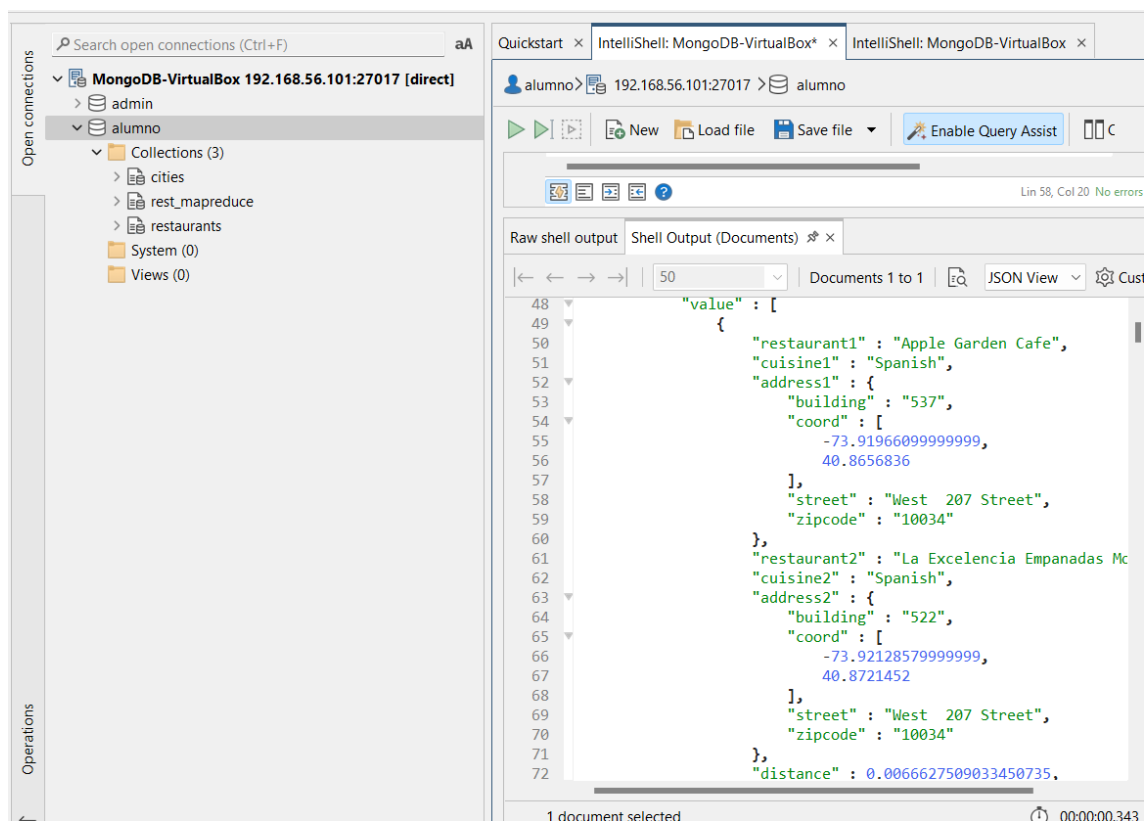
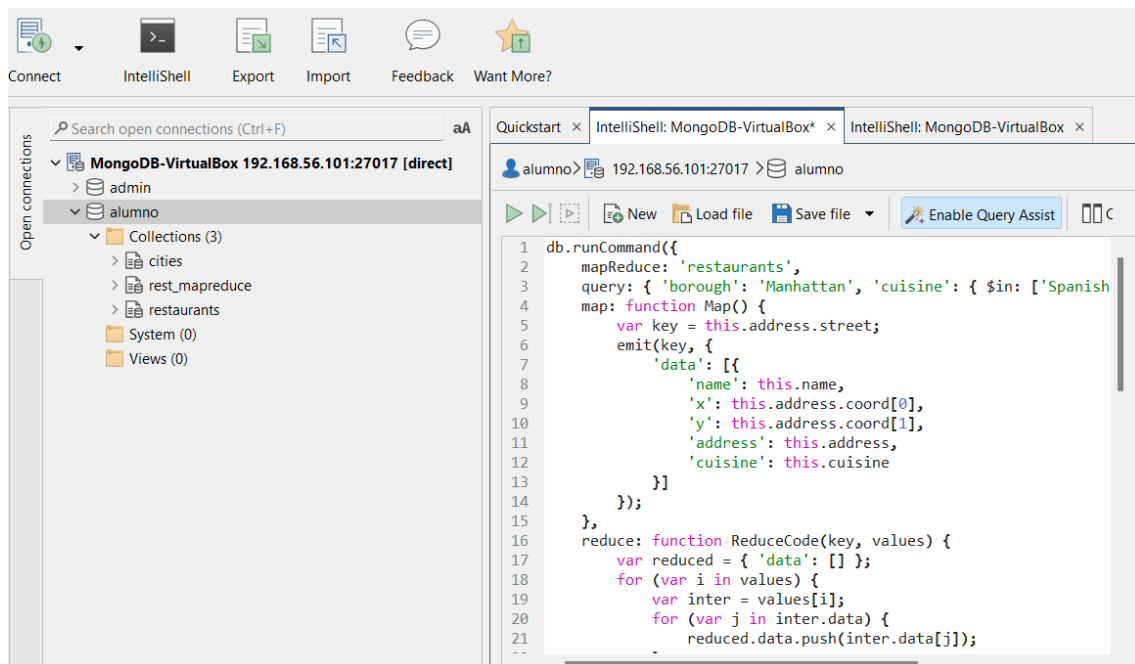
```
db.runCommand({
  mapReduce: 'restaurants',
  query: { 'grades.score': { $gte: 12 }, 'borough': 'Manhattan',
    'cuisine': { $in: ['Spanish', 'Italian'] } },
  map: function Map() {
    var key = this.address.street;
    emit(key, {
      'data': [{
        'name': this.name,
        'x': this.address.coord[0],
        'y': this.address.coord[1]
      }
    ]
  }
})
```

```

    'address': this.address,
        'cuisine': this.cuisine
    }]
    });
},
    reduce: function ReduceCode(key, values) {
        var reduced = { 'data': [] };
        for (var i in values) {
            var inter = values[i];
            for (var j in inter.data) {
                reduced.data.push(inter.data[j]);
            }
        }
        return reduced;
    },
    finalize: function Finalize(key, reduced) {
        if (reduced.data.length == 1) {
            return null;
        }
        var min_dist = Infinity;
        var bestRests = [];
        for (var i in reduced.data) {
            for (var j in reduced.data) {
                if (i >= j) continue;
                var r1 = reduced.data[i];
                var r2 = reduced.data[j];
                var distance = Math.sqrt(Math.pow(r1.x - r2.x, 2) +
Math.pow(r1.y - r2.y, 2));
                if (distance < min_dist) {
                    bestRests = [{
                        'restaurant1': r1.name, 'cuisine1': r1.cuisine,
'address1': r1.address,
                        'restaurant2': r2.name, 'cuisine2':
r2.cuisine, 'address2': r2.address,
                        'distance': distance,
                        'cantidad_restaurantes_evaluados':
reduced.data.length
                    }];
                    min_dist = distance;
                } else if (distance == min_dist) {
                    bestRests.push({
                        'restaurant1': r1.name, 'cuisine1': r1.cuisine,
'address1': r1.address,
                        'restaurant2': r2.name, 'cuisine2': r2.cuisine,
'address2': r2.address,
                        'distance': distance,
                        'cantidad_restaurantes_evaluados':
reduced.data.length
                    });
                }
            }
        }
        return bestRests;
    },
    out: { replace: "rest_mapreduce" });
}

```

## Capturas de pantalla:



## Con enfoque Aggregate

```
db.restaurants.aggregate([
  {
    $match: {
      borough: "Manhattan",
      cuisine: { $in: ["Spanish", "Italian"] },
      "grades.score": { $not: { $lt: 12 } }
    }
  },
  {
    $group: {
      _id: { street: "$address.street", cuisine: "$cuisine" },
      restaurants: { $push: { name: "$name", address: "$address" } },
      count: { $sum: 1 }
    }
  },
  {
    $unwind: "$restaurants"
  },
  {
    $sort: { "restaurants.name": 1 }
  },
  {
    $group: {
      _id: "$_id",
      restaurants: { $push: "$restaurants" },
      count: { $first: "$count" }
    }
  },
  {
    $addFields: {
      restaurants: { $slice: ["$restaurants", 2] }
    }
  },
  {
    $project: {
      _id: 0,
      street: '$_id.street',
      cuisine: '$_id.cuisine',
      restaurants: 1,
      count: 1
    }
  },
  {
    $out: "rest aggregate"
  }
])
```

## Consulta en Aggregate

The screenshot shows the MongoDB Aggregation Pipeline Builder interface. The pipeline consists of seven stages: \$match, \$group, \$unwind, \$sort, \$group, \$addFields, and \$project. The \$match stage is selected, and its operator is set to \$match. The match criteria are: borough: "Manhattan", cuisine: { \$in: ["Spanish", "Italian"] }, and grades.score: { \$not: { \$lt: 12 } }. The Pipeline Output tab shows the result of the pipeline, which is a single document with a count of 1 and a list of restaurants. The first restaurant is "Two Boots Grand Central" located at "East 61 St Street" in Manhattan, with a score of 12.

```
1 {
2   borough: "Manhattan",
3   cuisine: { $in: ["Spanish", "Italian"] },
4   "grades.score": { $not: { $lt: 12 } }
5 }
```

```
73   "count" : NumberInt(1),
74   "street" : "East 61 St Street",
75   "cuisine" : "Italian"
76 }
77 {
78   "restaurants" : [
79     {
80       "name" : "Two Boots Grand Central",
81       "address" : {
82         "building" : "231",
83         "coord" : [
84           -73.9772294,
85           40.7527262
86         ]
87       }
88     }
89   ]
90 }
```

## Nuevo documento con los restaurantes

The screenshot shows the MongoDB Query Builder interface. The Query field is empty, and the Projection field is set to {}. The Result tab shows the result of the query, which is a single document with a count of 2 and a list of restaurants. The first restaurant is "Pizza Guys" located at "Broadway" in Manhattan, with a score of 12.

```
10   40.7581421
11   },
12   "street" : "Broadway",
13   "zipcode" : "10036"
14 }
15 },
16 {
17   "name" : "Pizza Guys",
18   "address" : {
19     "building" : "3609",
20     "coord" : [
21       -73.9491109,
22       40.8288547
23     ]
24   },
25   "street" : "Broadway",
26   "zipcode" : "10031"
27 }
28 },
29 "count" : NumberInt(2),
30 "street" : "Broadway",
31 "cuisine" : "Italian"
```