MongoDB

Instalación

1. Descargue el instalador desde:
   Usuarios 32-bit:
   https://fastdl.mongodb.org/linux/mongodb-linux-i686-3.2.16.tgz

   Usuarios 64-bit:

   https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-ubuntu1604-3.4.2.tgz
2. Ingrese a la carpeta de descarga y descomprima el archivo:

```
$ cd /home/usuario/Descargas
$ tar -xzf mongodb-linux-x86_64-ubuntu1604-3.4.2.tgz
```

3. Cree el directorio de datos:

```
$ mkdir -p /home/usuario/mongodb/data
```

4. Inicie el servicio y **deje abierta esta consola**:

```
$ /home/usuario/Descargas/mongodb-linux-x86_64-ubuntu1604-3.4.2/bin/mongod --dbpath /home/usuario/mongodb/data
```

Interacción con la consola

En una **nueva consola**, ejecute el interprete de mongo:

```
$ /home/usuario/Descargas/mongodb-linux-x86_64-ubuntu1604-3.4.2/bin/mongo
MongoDB shell version v3.4.2
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.2
Welcome to the MongoDB shell.
For interactive help, type "help".
>
```

Insertar

Seleccione la base de datos `agenda` (Crea una nueva si no existe)

```
> use agenda
```

Inserte registros a la base de datos creada

```
> db.agenda.insert({
_id : 1,
name : "Juan",
age : 18,
gender : "male",
contact : {
  address : "Fake Street 123",
  phone : "555 123"
}
})

> db.agenda.insert({
```

```
_id : 2,
name : "Maria",
age : 18,
gender : "female",
contact : {
  address : "Fake Street 123",
  phone : "555 987"
}
})

> db.agenda.insert([
{
_id : 3,
name : "Pedro",
age : 19,
gender : "male",
contact : {
  address : "Nowhere",
  phone : "555 444"
}
},
{
_id : 4,
name : "Ana",
age : 20,
gender : "female",
contact : {
  address : "Timesquare",
  phone : "321 678"
}
}])
```

Listar documentos

1. Todos los documentos en la colección

```
> db.agenda.find()
```

2. Los documentos que contengan la pareja `campo:valor`

```
> db.agenda.find({name : "Juan"})
```

3. Los documentos que contengan alguno de los valores en una lista

```
> db.agenda.find({age : {$in : [18, 19]}})
```

4. Los documentos con condiciones `AND` (`age = 18` y `genero = male`)

```
> db.agenda.find({age : 18, gender : "male"})
```

5. Los documentos con condiciones `OR` ((`age = 18` ó `genero = male`)

```
> db.agenda.find({
$or : [
 {age : 18},
 {gender : "male"}
]})
```

6. Los documentos con condiciones `AND` y `OR`

```
> db.agenda.find({
"contact.address" : "Fake Street 123",
$or : [
 {age : 18},
 {gender : "male"}
]})
```

7. Buscar documentos con documentos embebidos

```
> db.agenda.find({
 contact : {address : "Fake Street 123", phone : "555 987"}
})
```

8. Buscar documentos utilizando operadores `$lt` (less than) y `$gt` (greater than)

```
> db.agenda.find({age : {$gt : 18}})
> db.agenda.find({age : {$lt : 20}})
```

9. Filtrar los campos de los documentos obtenidos con una consulta. En este ejemplo sólo el campo `name` y `gender` es retornado.

```
> db.agenda.find({age : 18}, {name : 1, gender : 1, _id : 0})
```

10. Actualizar un valor en el primer documento que cumple la consulta

```
> db.agenda.find({name : "Juan"})
> db.agenda.update({name : "Juan"}, {$set : {"contact.phone" : "Lost"}})
> db.agenda.find({name : "Juan"})
```

11. Actualizar todos los valores de los documentos que cumplan la consulta

```
> db.agenda.updateMany({age : 18}, {$set : {age : 21}})
```

12. Eliminar el primer documento que cumpla la consulta

```
> db.agenda.deleteOne({age : 19})
```

13. Eliminar todos los documentos que cumplan la consulta

```
> db.agenda.deleteMany({age : 21})
```

Búsqueda de Texto

1. Crear una nueva colección

```
> use stores
```

2. Insertar documentos de ejemplo

```
> db.stores.insert([
{_id: 1, name: "Java Hut", description: "Coffee and cakes"},
{_id: 2, name: "Burger Buns", description: "Gourmet hamburgers"},
{_id: 3, name: "Coffee Shop", description: "Just coffee"},
{_id: 4, name: "Clothes Clothes Clothes", description: "Discount clothing"},
{_id: 5, name: "Java Shopping", description: "Indonesian goods"}
])
```

3. Crear índice de texto

```
> db.stores.createIndex({name : "text", description : "text"})
```

4. Buscar documentos con cualquiera de los términos especificados

```
> db.stores.find({$text : {$search : "java coffee shop"}})
```

5. Buscar documentos con la palabra “java” o la frase exacta “coffee shop”

```
> db.stores.find({$text : {$search : "java \"coffee shop\""}})
```

6. Buscar documentos que tengan las palabras “java” o “shop” y excluyan el termino “coffee”

```
> db.stores.find({$text : {$search : "java shop -coffee"}})
```

7. Buscar y ordenar los documentos por relevancia con respecto a la consulta

```
> db.stores.find(
{$text : {$search : "java coffee shop"}},
{score : {$meta : "textScore"}}
).sort({score : {$meta : "textScore"}})
```

Manual de Referencia MongoDB

https://docs.mongodb.com/manual/

Taller entregable

Hace parte de un equipo de análisis de datos para una empresa que realiza inteligencia de negocios. Se ha encargado el análisis de una base de datos de restaurantes de la ciudad.
Para ello se solicitan una serie de consultas sobre la base de datos.
La estructura de los documentos en la base de datos es la siguiente:


```
{
  "address":{
    "building":"1007",
    "coord":[
      -73.856077,
      40.848447
    ],
    "street":"Morris Park Ave",
    "zipcode":"10462"
  },
  "borough":"Bronx",
  "cuisine":"Bakery",
  "grades":[
    {
      "date":{
        "$date":1393804800000
      },
      "grade":"A",
      "score":2
    },
    {
      "date":{
        "$date":1378857600000
      },
      "grade":"A",
      "score":6
    },
    {
```

```
      "date":{
        "$date":1358985600000
      },
      "grade":"A",
      "score":10
    },
    {
      "date":{
        "$date":1322006400000
      },
      "grade":"A",
      "score":9
    },
    {
      "date":{
        "$date":1299715200000
      },
      "grade":"B",
      "score":14
    }
  ],
  "name":"Morris Park Bake Shop",
  "restaurant_id":"30075445"
}
```


Requerimientos

- Descargue los documentos del siguiente enlace: data.json
- Cree la base de datos e ingrese los documentos en la misma (Averiguar como cargar un archivo JSON en MongoDB).
- Cree las siguientes consultas:
  i. Todos los restaurantes de la base de datos.
  ii. Todos los restaurantes: únicamente los campos `restaurant_id`, `name`, `cuisine`.
  iii. Todos los restaurantes: únicamente los campos `restaurant_id`, `name`, `zipcode` y SIN `_id`.
  iv. Restaurante en el `borough` "Manhattan".
  v. Restaurantes con `score` mayor que 90.
  vi. Restaurante con `score` mayor que 80 y menor que 90.
  vii. Restaurantes ubicados en `latitude` menor a -95.754168.
  viii. Restaurantes para los cuales `cuisine` es diferente de "American", tiene un `grade` de "A" y no pertenece al `borough` "Brooklyn".
  ix. Restaurantes en los cuales el nombre comienza por la palabra "Wil". (Hint: usar expresión regular sobre el campo `name`).
  x. Restaurantes en los cuales la `cuisine` NO es "American" NI "Chinese" O el `name` comienza por la palabra "Wil". (Hint: utilizar los operadores `$or` y `$and`).
  xi. Restaurantes ordenados en ascendentemente por el `name`.
  xii. Restaurantes para los cuales el `address.street` existe. (Hint: utilizar operador `$exists`).

Entregable

- Archivo de texto plano con todas las queries de creación de base de datos, carga de datos y consultas.

Map-reduce

El siguiente ejemplo cuenta los restaurantes agrupados por `zipcode`

1. Cree la función `map`:

```
> var mapFunction = function() {
                    emit(this.address.zipcode, this.restaurant_id);
```

```
                };
```

2. Cree la función `reduce` :

```
> var reduceFunction = function(zipcode, restaurants) {
                       return restaurants.length;
                  };
```

3. Ejecute el proceso map-reduce y guárdelo en la colección `restaurants_count`

```
> db.restaurants.mapReduce(
                  mapFunction,
                  reduceFunction,
                  { out: "restaurants_count" }
                )
```

4. Consulte la colección generada:

```
> show collections
> db.restaurants_count.find()
```