Introducción a MongoDB

Harry Vargas Rodríguez

Listar todas las bases de datos

```
show dbs
```

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

```
use catalunya
```

Inserte registros a la base de datos `catalunya`

```
db.coleccion.insert()
```

Usted puede darle cualquier nombre a la colección. En este caso la colección se llamará `agenda`

Note que en el tercer comando se insertan dos documentos al tiempo.

```
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"
}
}])
```

Consultas sobre la colección ageda

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})
```

Taller práctico

Usted hace parte de un equipo de análisis de datos para una empresa que realiza inteligencia de negocios. Se le ha encargado el análisis de una base de datos de restaurantes de la ciudad de Nueva York. 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"
}
```


Actividad

- Descargue los documentos del siguiente enlace: data.json
- Cree la base de datos e ingrese los documentos en la misma. Consulte tutorial aquí.

- Cree las siguientes consultas:
    1. Todos los restaurantes de la base de datos.
    2. Todos los restaurantes: únicamente los campos `restaurant_id`, `name`, `cuisine`.
    3. Todos los restaurantes: únicamente los campos `restaurant_id`, `name`, `zipcode` y SIN `_id`.
    4. Restaurante en el `borough` “Manhattan”.
    5. Restaurantes con `score` mayor que 90.
    6. Restaurante con `score` mayor que 80 y menor que 90.
    7. Restaurantes ubicados en `latitude` menor a -95.754168.
    8. Restaurantes para los cuales `cuisine` es diferente de “American”, tiene un `grade` de “A” y no pertenece al `borough` “Brooklyn”.
    9. Restaurantes en los cuales el nombre comienza por la palabra “Wil”. (Hint: usar expresión regular sobre el campo `name`).
    10. 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`).
    11. Restaurantes ordenados en ascendentemente por el `name`.
    12. Restaurantes para los cuales el `address.street` existe. (Hint: utilizar operador `$exists`).

Entregable

Archivo de texto plano con todos los comandos de creación de base de datos, carga de datos y consultas.

Siga la siguiente estructura:

```
1. Todos los restaurantes de la base de datos.

Query
# de documentos que cumplen la condición (count)

2. Todos los restaurantes: únicamente los campos restaurant_id, name, cuisine.
```

```
Query
# de documentos que cumplen la condición (count)

.
.
.
```