Facultad de Informática – Universidad Nacional de La Plata
Bases de Datos 2 2020 -TP2
Bases de Datos NoSQL / Práctica con MongoDB
Entrega: 11/5

Parte 1: Bases de Datos NoSQL y Relacionales

▸ Si bien las BBDD NoSQL tienen diferencias fundamentales con los sistemas de BBDD Relacionales o RDBMS, algunos conceptos comunes se pueden relacionar. Responda las siguientes preguntas, considerando MongoDB en particular como Base de Datos NoSQL.

1. ¿Cuáles de los siguientes conceptos de RDBMS existen en MongoDB? En caso de no existir, ¿hay alguna alternativa? ¿Cuál es?

- *Base de Datos*
- *Tabla / Relación*
- *Fila / Tupla*
- *Columna*

En MongoDB, existe el concepto de Base de Datos. Para referirse a una tabla o relación, existe el concepto de colección. Las filas o tuplas son representadas por documentos y las columnas por campos.

2. MongoDB tiene soporte para transacciones, pero no es igual que el de los RDBMS. ¿Cuál es el alcance de una transacción en MongoDB?

El alcance de las transacciones en MongoDB es a nivel de documentos. Es decir, cada documento se altera atomicamente.

3. Para acelerar las consultas, MongoDB tiene soporte para índices. ¿Qué tipos de índices soporta?

Los tipos de índices que soporta MongoDB son:

- **Single field:** Estos índices se aplican a un solo campo de la colección. Comando para crearlo: *db.collection.createIndex({ <key> : <options> })*
- **Compound index:** En este caso el índice se generará sobre varios campos. Comando para crearlo: *db.collection.createIndex({ <field1> : <type>, <field2> : <type> }).* El índice que se generará con la instrucción anterior, agrupará los datos primero por el campo field1 y luego por el campo field2.
- **Multikey index:** MongoDb usa los índices multikey para indexar el contenido almacenado en arreglos. Si se crea un índice en un campo de tipo arreglo, MongoDb crea un índice separado para cada elemento del arreglo. Éste tipo de índice le permite a las consultas seleccionar documentos que contengan arreglos matcheando por un elemento o varios en el arreglo. Además, para crear índices multikey no se necesita especificar de forma explícita, sino que MongoDB lo hace automáticamente.

- **Geospatial index:** Para soportar consultas eficientes sobre datos de coordenadas geoespaciales, MongoDB provee los índices 2dsphere, los cuales usan geometría esférica para retornar resultados. Comando para crearlo: *db.collection.createIndex({ <location field> : "*2dsphere*" })*
- **Text index:** Éste tipo soporta la búsqueda por contenido de tipo string en una colección, y además no almacena palabras específicas de cada lenguaje (como por ejemplo "de", "a", "o"), sino que sólo almacena palabras raíces. Comando para crearlo: *db.collection.createIndex({ <field> : "*text*" })*
- **Hashed index:** Éste tipo indexa un valor hash de un campo, y es lo que le permite a MongoDB realizar lo que se denomina Hashed Sharding.

4. ¿Existen claves foráneas en MongoDB?

En MongoDB no existen las claves foráneas, las bases de datos basadas en colecciones y documentos y su falta de estructura permiten guardar la mayor información necesaria posible en los documentos y así evitar tantas operaciones JOIN, ésto agiliza mucho las consultas. Sin embargo, las claves foráneas se pueden manejar por nosotros mismos, pero no es un comportamiento nativo de MongoDB.

Parte 2: Primeros pasos con MongoDB

▸ Descargue la última versión de MongoDB desde el sitio oficial. Ingrese al cliente de línea de comando para realizar los siguientes ejercicios.

5. Cree una nueva base de datos llamada airbdb, y una colección llamada apartments. En esa colección inserte un nuevo documento (un departamento) con los siguientes atributos: {name:'Apartment with 2 bedrooms', capacity:4} recupere la información del departamento usando el comando **db.apartments.find()** (puede agregar la función .pretty() al final de la expresión para ver los datos indentados). Notará que no se encuentran exactamente los atributos que insertó. ¿Cuál es la diferencia?

Crear la DB:

use airbdb

Crear la colección e insertar documento:

db.apartments.save({name:'Apartment with 2 bedrooms', capacity:4})

Ver resultados:

db.apartments.find().pretty()

Resultado:

```
{
        "_id" : ObjectId("5eb1f66f44deef6c666cc0d6"),
        "name" : "Apartment with 2 bedrooms",
        "capacity" : 4
```

}

La diferencia es que MongoDB agrega automáticamente el campo "**_id**", el cual es de tipo ObjectId y será único en la DB.

▸ Una característica fundamental de MongoDB y otras bases NoSQL es que los documentos no tienen una estructura definida, como puede ser una tabla en un RDBMS. En una misma colección pueden convivir documentos con diferentes atributos, e incluso atributos de múltiples valores y documentos embebidos.

6. Agregue los siguientes documentos a la colección de departamentos:

{name:'New Apartment', capacity:3, services: ['wifi', 'ac']}
{name:'Nice apt for 6', capacity:6, services: ['parking']}
{name:'1950s Apartment', capacity:3}
{name:'Duplex Floor', capacity:4, services: ['wifi', 'breakfast', 'laundry']}

Y busque los departamentos:

1. con capacidad para 3 personas.
2. con capacidad para 4 personas o más
3. con wifi
4. que incluyan la palabra 'Apartment' en su nombre
5. con la palabra 'Apartment' en su nombre y capacidad para más de 3 personas
6. sin servicios (es decir, que el atributo esté ausente)
7. vuelva a realizar la última consulta pero proyecte sólo el nombre del departamento en los resultados, omitiendo incluso el atributo _id de la proyección.

Consultas

1. db.apartments.find({ capacity: 3 })
2. db.apartments.find({ capacity: { $gte: 4 } })
3. db.apartments.find({ services: 'wifi' })
4. db.apartments.find({ name: { $regex: /Apartment/ } })
5. db.apartments.find({ name: { $regex: /Apartment/ }, capacity : {$gt: 3} })
6. db.apartments.find({ services: {$exists: false} })
7. db.apartments.find(
   { services: {$exists: false }},
   { name: 1 }
)

▸ En MongoDB hay diferentes maneras de realizar actualizaciones, de acuerdo a las necesidades del esquema flexible de documentos.

7. Actualice el "Duplex Floor" asignándole capacidad 5.

```
db.apartments.updateOne(
                { name: 'Duplex Floor' },
                { $set: { capacity: 5 } }
                )
```

8. Agregue “laundry” al listado de services del “Nice apt for 6”.

```
db.apartments.updateOne(
                { name: 'Nice apt for 6' },
                { $push:{ services: 'laundry' } }
                )
```

9. Agregue una persona más de capacidad a todos los departamentos con wifi.

```
db.apartments.updateMany(
                { services: 'wifi' },
                { $inc: { capacity:1 } }
                )
```

Parte 3: Índices

▸ Elimine todos los departamentos de la colección. Guarde en un archivo llamado ‘generador.js’ el siguiente código JavaScript y ejecútelo con: load(<ruta del archivo

```
for (var i = 1; i <= 50000; i++) {
       var randomServices = ['wifi', 'pool', 'parking', 'breakfast'].sort( function()
{ return 0.5 - Math.random() } ).slice(1, Math.floor(Math.random() * 5));
       var randomCapacity = Math.ceil(Math.random() * 5);
       var randomLong = ((Math.random()/1.3)+51);
       var randomLat = Math.random() - .4;
       db.apartments.insert({
              name:’Apartment '+i,
              capacity:randomCapacity,
              services: randomServices,
              location: {
             type: "Point",
             coordinates: [randomLat, randomLong]
          }
       });
}
```

‘generador.js’>).

Borrar todos los datos de la colección:

db.apartments.deleteMany({})

Cargar generador.js:

1) Mover el archivo generador.js que deje en la carpeta de drive a /home/bd2
2) Abrir el terminal, y desde la consola de mongo ejecutar load(“/home/bd2/generador.js”)

10. Busque en la colección de departamentos si existe algún índice definido.

db.apartments.getIndexes()

MongoDB por defecto crea un índice sobre el campo _id durante la creación de una colección. Este índice controla la unicidad del valor del _id. No se puede eliminar este índice.

11. Cree un índice para el campo name. Busque los departamentos que tengan en su nombre el string “11” y utilice el método explain("executionStats") al final de la consulta, para comparar la cantidad de documentos examinados y el tiempo en milisegundos de la consulta con y sin índice.

db.apartments.find({ name: { $regex: /11/ } }).explain("executionStats")

“executionTimeMillis”: 98
“totalDocsExamined”: 5000

db.apartments.createIndex({name:-1})

db.apartments.find({ name: { $regex: /11/ } }).explain("executionStats")

“executionTimeMillis”: 244
“totalDocsExamined”: 2291

12. Busque los departamentos dentro de la ciudad de Londres. Para esto, puede definir una variable en la terminal y asignarle como valor el polígono del archivo provisto greaterlondon.geojson (copiando y pegando directamente). Cree un índice geoespacial de tipo 2dsphere para el campo location de la colección apartments y, de la misma forma que en el punto 11, compare la performance de la consulta con y sin dicho índice.

Use esta web para eliminar caracteres inválidos del archivo:

https://www.nousphere.net/cleanspecial.php

Copie el contenido del archivo “greaterlondon.geojson”, lo “limpie” con esa web y luego hice lo siguiente:

Abrí mongo desde el terminal y declare una variable de esta manera:

coord = (aca pegas el contenido del archivo limpio)

Después ejecute:

```
db.apartments.find(
        { location:
                { $geoWithin: { $geometry: coord } }
        }
        ).explain("executionStats")
```

"executionTimeMillis" : 622
"totalDocsExamined" : 50000

Cree el índice 2dsphere

```
db.apartments.createIndex({location:"2dsphere"})
```

Después ejecute:

```
db.apartments.find(
        { location:
                { $geoWithin: { $geometry: coord } }
        }
        ).explain("executionStats")
```

"executionTimeMillis" : 447
"totalDocsExamined" : 18341

Parte 4: Aggregation Framework

▸ MongoDB cuenta con un Aggregation Framework que brinda la posibilidad de hacer analítica en tiempo real del estilo OLAP (Online Analytical Processing), de forma similar a otros productos específicos como Hadoop o MapReduce. En los siguientes ejercicios se verán algunos ejemplos de su aplicabilidad.

Al igual que en la parte 3, guarde en un archivo llamado ‘generadorReservas.js’ el siguiente

código JavaScript y ejecútelo con la función load():

```
Date.prototype.addDays=function(d){return new Date(this.valueOf()+864E5*d);};
function randomDate(start, end) {
  return new Date(start.getTime()+Math.random()*(end.getTime()-start.getTime()));
}
for (var i = 1; i <= 50000; i++) {
      if (Math.random() > 0.7) {
            var randomReservations = Math.ceil(Math.random() * 5);
            for (var r = 1; r <= randomReservations; r++){
                  var startDate = randomDate(new Date(2012, 0, 1), new Date());
                  var days = Math.ceil(Math.random()*8);
                  var toDate = startDate.addDays(days);
                  var randomAmount = days * ((Math.random() * 100) + 80).toFixed(2);
                  db.reservations.insert({
                        apartmentName:'Apartment '+i,
                        from: startDate,
                        to: toDate,
                        amount: randomAmount
                  });}
      }
}
```

13. Obtenga 5 departamentos aleatorios de la colección.

db.apartments.aggregate([{ $sample: { size: 5 } }]).pretty()

14. Usando el framework de agregación, obtenga los departamentos que estén a 15km (o menos) del centro de la ciudad de Londres ([-0.127718, 51.507451]) y guárdelos en una nueva colección.

```
db.apartments.aggregate([
  {
    $geoNear: {
      near: { type: "Point", coordinates: [ -0.127718, 51.507451 ]},
      distanceField: "location.coordinates",
      maxDistance: 15000
    }
  },
  { $out: "apartmentsNearLondon" }
])
```

15. Para los departamentos hallados en el punto anterior, obtener una colección con cada departamento agregando un atributo reservas que contenga un array con todas sus reservas.

```
db.apartmentsNearLondon.aggregate([
        {
          $lookup: {
                from: "reservations",
                localField: "name",
                foreignField: "apartmentName",
                as: "reservations"
          }
        }
]).pretty()
```

Note que sólo es posible ligarlas por el nombre del departamento.
▸ Si la consulta se empieza a tornar difícil de leer, se pueden ir guardando los agregadores en variables, que no son más que objetos en formato JSON.

16. Usando la colección del punto anterior, obtenga el promedio de precio pagado por reserva (precio completo, no dividir por la cantidad de noches) de cada departamento.

esto tarda bastante

```
db.apartmentsNearLondon.aggregate([
        {
          $lookup: {
                from: "reservations",
                localField: "name",
                foreignField: "apartmentName",
                as: "reservations"
                }
        },
        {
          $unwind : "$reservations"
        },
        {
          $group: {
                _id: "$name",
                avg: {
                        $avg: "$reservations.amount"
                    }
                }
        }
])
```