Facultad de Informática – Universidad Nacional de La Plata
Cátedra de Bases de Datos II
https://groups.google.com/forum/#!forum/bbdd2-2018

Bases de Datos 2 – 2018

Trabajo Práctico 2

Bases de Datos NoSQL / Práctica con MongoDB

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
2. MongoDB tiene soporte para transacciones, pero no es igual que el de los RDBMS. ¿Cuál es el alcance de una transacción en MongoDB?
3. Para acelerar las consultas, MongoDB tiene soporte para índices. ¿Qué tipos de índices soporta?
4. ¿Existen claves foráneas en 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?

▶ 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:

- con capacidad para 3 personas.
- con capacidad para 4 personas o más
- con wifi
- que incluyan la palabra ‘Apartment’ en su nombre
- con la palabra ‘Apartment’ en su nombre y capacidad para más de 3 personas
- sin servicios (es decir, que el atributo esté ausente)

vuelva a realizar la última consulta pero proyecte sólo el nombre del departamento en los resultados, omitiendo incluso el atributo **_id** de la proyección.

▶ 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.
8. Agregue “laundry” al listado de services del “Nice apt for 6”.
9. Agregue una persona más de capacidad a **todos** los departamentos con wifi.

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 ‘generador.js’>)`.

```
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]
        }
      });
}
```

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

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.

12. Busque los departamentos dentro de la ciudad de Londres. Para esto, puede obtener el polígono del archivo provisto **`greaterlondon.geojson`** y definir una variable en la terminal para facilitar la consulta. 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.

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 startDate = randomDate(new Date(2012, 0, 1), new Date());
              var days = Math.ceil(Math.random()*8);
              var toDate = startDate.addDays(days);
              var randomReservations = Math.ceil(Math.random() * 5);
              var randomAmount = days * ((Math.random() * 100) + 80).toFixed(2);
              for (var r = 1; r <= randomReservations; r++){
                     db.reservations.insert({
                            apartmentName:'Apartment '+i,
                            from: startDate,
                            to: toDate,
                            amount: randomAmount
                     });}
       }
}
```

13. Obtenga 5 departamentos aleatorios de la colección.

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.

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. 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.