Facultad de Informática – Universidad Nacional de La Plata
Cátedra de Bases de Datos II

Bases de Datos 2 2022 -TP3

Bases de Datos NoSQL / Práctica con MongoDB

entrega: 13/6

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 **vaccination**, y una colección llamada **nurses**. En esa colección inserte un nuevo documento (una enfermera) con los siguientes atributos:

   ```
   {name:'Morella Crespo', experience:9}
   ```

   recupere la información de la enfermera usando el comando `db.nurses.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 enfermeros:

```
{name:’Gale Molina', experience:8, vaccines: ['AZ', ‘Moderna']}
{name:’Honoria Fernández', experience:5, vaccines: ['Pfizer', ‘Moderna’, ‘Sputnik V’]}
{name:’Gonzalo Gallardo', experience:3}
{name:’Altea Parra', experience:6, vaccines: ['Pfizer']}
```

Y busque los enfermeros:

- de 5 años de experiencia o menos
- que hayan aplicado la vacuna “Pfizer”
- que no hayan aplicado vacunas (es decir, que el atributo *vaccines* esté ausente)
- de apellido ‘Fernández’
- con 6 o más años de experiencia y que hayan aplicado la vacuna ‘Moderna’

vuelva a realizar la última consulta pero proyecte sólo el nombre del enfermero/a 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 a “Gale Molina” cambiándole la experiencia a 9 años.
8. Cree el *array* de vacunas (*vaccines*) para ”Gonzalo Gallardo".
9. Agregue “AZ” a las vacunas de “Altea Parra”.
10. Duplique la experiencia de **todos** los enfermeros que hayan aplicado la vacuna “Pfizer”

Parte 3: Índices

▶ Elimine a todos los enfermeros de la colección. Guarde en un archivo llamado 'generador.js' el siguiente código JavaScript y ejecútelo con: `load(<ruta del archivo 'generador.js'>)`. Si utiliza un cliente que lo permita (ej. Robo3T), se puede ejecutar directamente en el espacio de consultas.

```
var vaccines = ['AZ', 'Pfizer', 'Moderna', 'Sputnik V', "Johnson"];
for (var i = 1; i <= 2000; i++) {
      var randomVax = vaccines.sort( function() { return 0.5 -
Math.random() } ).slice(1, Math.floor(Math.random() * 5));
      var randomExperience = Math.ceil(2+(Math.random() * 20 - 2));
      db.nurses.insert({
            name:'Enfermero '+i,
            experience:randomExperience,
            tags: randomVax
      });}
var patientNumber = 0;
for (var i = 1; i <= 2000; i++) {
            var dosesCount = 50 + Math.ceil(Math.random() * 100);
            for (var r = 1; r <= dosesCount; r++){
                  var randomLong = -34.56 - (Math.random() * .23);
                  var randomLat = -58.4 - (Math.random() * .22);
                  db.patients.insert({
                    name:'Paciente '+patientNumber,
                    address: {type: "Point",coordinates: [randomLat, randomLong]}
                  });
                  patientNumber++;
                  var year = 2020 + Math.round(Math.random());
                  var month = Math.ceil(Math.random() * 12);
                  var day = Math.ceil(Math.random() * 28);
                  db.doses.insert({
                    nurse:'Enfermero '+i,
                    patient:'Paciente '+patientNumber,
                    vaccine: vaccines[Math.floor(Math.random()*vaccines.length)],
                    date: new Date(year+'-'+month+'-'+day)
                  });
            }}
```

11. Busque en la colección de compras (`doses`) si existe algún índice definido.

12. Cree un índice para el campo `nurse` de la colección `doses`. Busque las dosis que tengan en el nombre del enfermero 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.

13. Busque los pacientes que viven dentro de la ciudad de Buenos Aires. Para esto, puede definir una variable en la terminal y asignarle como valor el polígono del archivo provisto **`caba.geojson`** (copiando y pegando directamente). Cree un índice geoespacial de tipo **`2dsphere`** para el campo **`location`** de la colección **`patients`** y, de la misma forma que en el punto 12, 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.

14. Obtenga 5 pacientes aleatorios de la colección.

15. Usando el framework de agregación, obtenga los pacientes que vivan a 1km (o menos) del centro geográfico de la ciudad de Buenos Aires (`[-58.4586,-34.5968]`) y guárdelos en una nueva colección.

16. Obtenga una colección de las dosis aplicadas a los pacientes del punto anterior. Note que sólo es posible ligarlas por el nombre del paciente.

▶ 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.

17. Obtenga una nueva colección de *nurses*, cuyos nombres incluyan el string “111”. En cada documento (cada *nurse*) se debe agregar un atributo *doses* que consista en un array con todas las dosis que aplicó después del 1/5/2021.