TICS575 - Técnicas para Big Data

MongoDB

1. Datos

Trabajaremos en el contexto de una red social, con una colección de `usuarios`, otra colección de `mensajes` y una colección de productos. Los datos se ven así:

Usuarios:

```
{
  "uid": <id del usuario>,
  "name": <nombre del usuario>,
  "last_name": <apellido del usuario>,
  "follows": [<lista de usuarios a los que sigue>],
  "ocupation": <actividad que realiza>,
  "age": <edad del usuario>
}
```

Mensajes:

```
{
  "mid": <id del mensaje emitido>,
  "uid": <id del usuario que emite el mensaje>,
  "msj": {"title": <título del mensaje>,
          "content": <contenido del mensaje>},
  "likes": <número de likes del mensaje>
}
```

Productos:

```
{
  "pid": <id del mensaje emitido>,
  "name": <id del usuario que emite el mensaje>,
  "tags": [<tag 1>, ..., <tag n>],
  "price": <precio del producto>
}
```

2. Importando los datos

Para importar los datos de `usuarios.json` en la colección `usuarios`:

```
mongoimport --db test --collection usuarios --drop --file usuarios.json --jsonArray
```

Para importar los datos de `mensajes.json` en la colección `mensajes`:

```
mongoimport --db test --collection mensajes --drop --file mensajes.json --jsonArray
```

Para importar los datos de `productos.json` en la colección `productos`:

```
mongoimport --db test --collection productos --drop --file productos.json --jsonArray
```

3. Utilizando la consola de MongoDB

3.1. Consultando MongoDB

Para utilizar MongoDB debes tener un servidor corriendo. Una vez hecho esto, debes conectarte en una terminal usando el comando `mongo`. Debería aparecer la consola de MongoDB, en la que puedes hacer consultas. Ahora ejecuta las siguientes instrucciones:

```
use test // Nos conectamos a la base de datos test
db.usuarios.find(); // Obtenemos todos los usuarios
```

La consulta básica se realiza mediante la función `find()`. Opcionalmente puedes ejecutar find con dos parámetros que también son documentos JSON. El primer parámetro corresponde a filtros (selección) y el segundo a proyección. Ahora ejecuta la misma consulta anterior pero usando el comando `pretty()` para hacer que aparezcan los documentos de una forma un poco más legible:

```
db.usuarios.find({}, {}).pretty(); //
```

3.2. Filtros

El primer parámetro del comando `find()` son los filtros, o selección. La selección por igualdad es simple: la siguiente consulta busca el usuario con `uid` igual a 2:

```
db.usuarios.find({uid: 2}, {}).pretty();
```

Ahora vamos a buscar todos los productos con precio mayor a 300:

```
db.productos.find({price: {$gt: 300}}, {}).pretty();
```

Si te fijas, usamos la instrucción `price: $gt: 300`. Esto nos exige que debemos buscar todos los elementos donde la *key* `price` sea mayor que 300. Tienes los siguientes comandos a disposición:

- `$gt`: mayor que.
- `$lt`: menor que.
- `$gte`: mayor o igual que.
- `$lte`: menor o igual que.

Ahora busquemos los productos con precio entre 200 y 400. Para esto vamos a utilizar la instrucción `$and`:

```
db.productos.find({$and: [{price: {$gte: 200}}, {price: {$lte: 400}}]}, {}).pretty();
```

Como ves, esto se empieza a complicar. Vamos a escribir la consulta anterior en más de una línea para hacerla más legible:

```
db.productos.find(
  {
    $and: [
      {price: {$gte: 200}},
      {price: {$lte: 400}}
    ]
  }, {}).pretty();
```

La instrucción `$and` recibe un arreglo de selecciones. Se van a retornar los documentos que cumplan todas las condiciones. También existe un comando `$or`. Finalmente, también tenemos acceso a desigualdades con la instrucción `$ne`:

```
db.usuarios.find({uid: {$ne: 2}}, {}).pretty();
```

En general las consultas no se ven tan amigables o legibles. Puedes ver la documentación de MongoDB para ver qué otros filtros se pueden usar.

3.3. Proyección

El segundo parámetro de `find()` corresponde a los campos a proyectar. Ahora proyectaremos solamente los nombres de los usuarios:

```
db.usuarios.find({}, {name: 1});
```

También podemos excluir en la proyección, usando un 0 en vez de un 1:
El segundo parámetro de `find()` corresponde a los campos a proyectar. Ahora proyectaremos solamente los nombres de los usuarios:

```
db.productos.find({}, {tags: 0});
```

En el caso anterior, estamos excluyendo el arreglo de tags de los productos.

4. Ejemplos de consultas

Aquí hay algunos ejemplos breves de consultas que se pueden hacer en MongoDB.

- Todos los mensajes:

  ```
  db.mensajes.find({}, {}).pretty();
  ```

- El contenido de todos los mensajes:

  ```
  db.mensajes.find({}, {"msj.content": 1});
  ```

  En el ejemplo anterior estamos accediendo a una key dentro de un JSON dentro de los documentos.

- El contenido de todos los mensajes del usuario con id i:

  ```
  db.mensajes.find({"uid": i}, {"msj.content": 1});
  ```

- El contenido de todos los mensajes del usuario con id i o id j:

  ```
  db.mensajes.find({$or: [{"uid": i}, { "uid": j }]},  {"msj.content": 1});
  ```

  En este ejemplo estamos usando la instrucción `$or`.

- Los usuarios mayores a 18 años ordenados por nombre:

  ```
  db.usuarios.find({age: {$gt: 18}}, {}).sort({"name": 1});
  ```

Aquí estamos haciendo uso de la función `sort()`.

- Todos los productos cuyos tags sean exactamente el arreglo `["Tag 1", "Tag 2"]`, en ese orden:

```
db.productos.find({tags: ["Tag 1", "Tag 2"]}, {});
```

- Todos los productos que en sus tags contengan los elementos `"Tag 4"` y `"Tag 3"`:

```
db.productos.find({tags: {$all: ["Tag 4", "Tag 3"]}}, {});
```

- Todos los productos que en sus tags contengan los elementos `"Tag 4"` y `"Tag 3"`:

```
db.productos.find({tags: {$all: ["Tag 4", "Tag 3"]}}, {});
```

- Todos los productos que en sus tags contengan el elemento `"Tag 5"`:

```
db.productos.find({tags: "Tag 5"}, {});
```

- Todos los productos que en sus tags contengan el elemento `"Tag 5"` o `"Tag 1"`:

```
db.productos.find({tags: {$in: ["Tag 5", "Tag 1"]}}, {});
```

- Todos los productos que en sus tags **no** contengan el elemento `"Tag 5"` o `"Tag 1"`:

```
db.productos.find({tags: {$nin: ["Tag 5", "Tag 1"]}}, {});
```

- El promedio del precio de los productos:

```
db.productos.aggregate(
  [
    {$group: {_id: null, total: {$avg: "$price"}}}
  ]
);
```

- La suma del precio de todos los productos que contengan la etiqueta `"Tag 3"`:

```
db.productos.aggregate(
  [
    {$match: {tags: "Tag 3"}},
    {$group: {_id: null, total: {$sum: "$price"}}}
  ]
);
```

- La suma de todos los `likes` agrupadas por usuario:

```
db.mensajes.aggregate(
  [
    {$group: {_id: "$uid", total: {$sum: "$likes"}}}
  ]
);
```

5. Utilizando JavaScript para extraer datos

Aquí hay algunos ejemplos de consultas más complejas que se pueden hacer en MongoDB con la ayuda de *Javascript*. Para correr estos ejemplos, si tu solución está en el archivo `ejemplo.js`, entonces debes correr el comando:

```
mongo ejemplo.js
```

Las consultas de ejemplo son:

- Cada id de usuario junto al nombre de la gente a la que sigue:

```
1  var cursor = db.usuarios.find({}, {});
2  cursor.forEach(
3    (element) => {
4      try {
5        var follows = element["follows"];
6        print("## User ID: " + element["uid"]);
7        for (var followed in follows) {
8          var user_cursor = db.usuarios.find({"uid": follows[followed]},{});
9          user_cursor.forEach(
10           (user_element) => {
11             print(user_element["name"]);
12           }
13         )
14       }
15       print("##");
16     }
17     catch(e) {
18       print("GG", e);
19     }
20   }
21 );
```

- Cada id de usuario, junto a su nombre y la cantidad de likes total de sus mensajes:

```
var cursor = db.usuarios.find({}, {});
cursor.forEach(
  (element) => {
    try {
      var msj_cursor = db.mensajes.find({"uid": element["uid"]}, {})
      var likes = 0;
      msj_cursor.forEach(
        (msj_element) => {
          likes += msj_element["likes"];
        }
      )
      print("User ID: " + element["uid"] +
            " - Name: " + element["name"] +
            " - Likes: " + likes);
    }
    catch(e) {
      print("GG", e);
    }
  }
);
```