MONGO

https://bit.ly/2v5EaEv

Aggregation

Con esto se puede hacer una cadena de operaciones una tras otra.
Lo más común (no siempre) es que se usen dos etapas:

- **Match:** Etapa en la que se filtran documentos según algún criterio. Igual que `find()`.
- **Group:** Agrupa los documentos según un identificador y aplica una función para obtener un resultado único.

```
mongo> db.coleccion.aggregate( [
    { $match: { p : "hojas" } },
    { $group: { _id : "p" , score : { $avg : "s" } } }
] )
```

Aggregation

Existen muchas operaciones para que se pueden agregar a una cadena de `aggregate`.
Por ejemplo, se tiene una instrucción especial para hacer consultas espaciales:

```
mongo> db.collection.aggregate( [ {
      $geoNear: {
            near: { type: "Point",coordinates: [ -73.9667, 40.78 ] },
            query: { categoria: "Universidades"},
            distanceField: "distancia"
      }
} ] )
```

Réplicas

Para trabajar con réplicas, se deben hacer cambios en la configuración. Pueden usar el archivo básico del drive, y cambiar los siguientes campos:

```
replication:
    replSetName: "rs0"
net:
    bindIp: 127.0.0.1,<direccion ip>
```

Revisen también el `path` del `log`, seguramente no existe la carpeta por defecto, cámbienla o agréguenla.

Réplicas

Recuerden usar al menos tres máquinas. En cada una deben cambiar la IP del archivo de configuración y correr:

```
$> mongod --config mongo.conf
```

Réplicas

Desde un cliente se pueden conectar al servidor que quieran que sea el servidor primario y correr:

```
mongo> rs.initiate( {
      _id:"rs0",
      members: [
            {_id: 0, host: "<ip0>"},
            {_id: 1, host: "<ip1>"},
            {_id: 2, host: "<ip2>"},
      ]
} )
```

Réplicas

Ya que se arma la configuración, se puede ver su `status`:

```
mongo> rs.status( )
```

Además, se puede verificar si el nodo al que se está conectado es primario o es réplica:

```
mongo> db.isMaster( )
```

Réplicas

Por supuesto, se puede uno conectar siempre al primary y que el cluster sea el encargado de manejar los cambios:

```
$> mongo --host
rs0/192.168.56.14:27017,192.168.56.15:27017,192.168.56.16:27017
```

Actividades

Datos:
`https://s3.amazonaws.com/tripdata/201307-201402-citibike-tripdata.zip`

Desarrollar un sistema de paseos en el que:

- Un usuario se pueda dar de alta junto con información de ubicación (coordenadas), debe usar al menos un lugar (casa por ejemplo) pero con opción de agregar más (trabajo, escuela, etc.)
- El usuario debe poder buscar las estaciones de bicicleta que le queden más cerca a sus lugares.

Actividades

- También debe poder planear viajes, dado un tiempo que quiere viajar, el sistema debe recomendar viajes usando como salida sus estaciones mas cercanas (o estaciones específicas seleccionadas por el usuario) y los destinos que le tomen mas o menos ese tiempo.
- El usuario debe poder dar la opción de que su viaje sea redondo (mismo punto de partida y salida). En este caso, solo se debe tomar en cuenta los datos que pasan por otras estaciones, a menos que el tiempo sea muy corto.
- Si el usuario busca una ruta específica que el sistema no tenga, la debe intentar armar de forma indirecta.

- EXTRA: Los resultados deben tomar en cuenta la hora a la que se hace la consulta y la hora de los viajes registrados para dar sus resultados con mayor confianza.