Bases de Datos NoSQL - Prueba de Evaluación

Santiago Breogán Pérez Pita

28 de diciembre de 2019

Índice

Abstract

Este trabajo está enfocado para adquirir el mayor conocimiento posible sobre las bases de datos no relacionales, en concreto sobre las bases documentales MongoDB. No es por tanto un trabajo con el fin de estructurar de forma adecuada unos datos, o de extraer las conclusiones pertinentes de ellos, sino con el fin de explotar las posibilidades de Mongo al máximo, de cara a su uso en un futuro.

1. Generación de un clúster. Arquitectura maestro - esclavo

Durante las primeras sesiones vimos como mucha de la potencia de Mongo reposaba sobre la forma en que se estructuraba la información. La arquitectura de maestro - esclavo implica la definición de un nodo maestro o principal que es el que recibe la información y las peticiones, y se encarga de registrarlas para que los nodos secundarios repitan las acciones del nodo principal. A esta estructura de nodo principal con nodos secundarios por debajo la llamamos ***replicaset***, cuando trabajamos con conjuntos muy grandes de

datos tendremos multiples replicasets que se reparten la carga de trabajo mediante ***sharding***. Esto es lo elemental en la arquitectura de Mongo, así que lo primero que quiero hacer es crear un replicaset. Al trabajar en *MongoAtlas*, este replicaset es generado de forma automática (cómo hemos visto con GoogleCloud, los servicios en la nube abstraen la replicación de la información al usuario), sin embargo, para familiarizarme con esta arquitectura voy a generar mi propio replicaset con un nodo maestro, 3 nodos secundarios y 1 árbitro, que me parece una fórmula adecuada ya que los nodos continuarán funcionando tanto si cae el maestro (los secundarios eligirán un nuevo maestro), como si caen el maestro y cualquier secundario (el árbitro desempatará entre los 2 secundarios restantes) o si caen el maestro y el árbitro (los tres secundarios eligirán entre ellos sin posibilidad de empate); es decir, estamos protegidos ante la caída 1 o 2 nodos, sean los que sean.
Determinada la estructura debemos definir el nivel de garantía de escritura y de lectura. Al definir estos niveles vamos a colocarnos en el diagrama CAP, ya sabemos que Mongo por defecto es CP, pero según como lo configuremos podemos desplazarnos hacia AP. Para la escritura voy a elegir el nivel de *Write Concern* ***journaled*** lo que significa que graba la recepción de las órdenes de escritura tras haberlas ejecutado en el nodo maestro, esto permitirá a la base de datos restablecerse en caso de reinicio. Para la lectura voy a elegir la opción *primaryPrefered* que permite leer de nodos secundarios si el primario no estuviera disponible. Esto aumenta la disponibilidad sacrificando consistencia. Definida la configuración del clúster procedo a copiar el código empleado para generarlo, este clúster lo he generado directamente desde la consola de Linux, cargando el shell de Mongo cuando lo necesité.

```
1  // Generamos una carpeta propia para la traza de cada servidor (no sé si el árbitro
       también la necesita)
2  mkdir -a /svr/mongodb/rs0-0 /svr/mongodb/rs0-1 /svr/mongodb/rs0-2 /svr/mongodb/rs0-3
       /svr/mongodb/rs0-04
3
4  // Arrancamos los servidores
5  mongod --replSet rs0 --port 27017 --dbpath /srv/mongodb/rs0-0 --smallfiles
       --oplogSize 128
6  mongod --replSet rs0 --port 27018 --dbpath /srv/mongodb/rs0-1 --smallfiles
       --oplogSize 128
7  mongod --replSet rs0 --port 27019 --dbpath /srv/mongodb/rs0-2 --smallfiles
       --oplogSize 128
8  mongod --replSet rs0 --port 27020 --dbpath /srv/mongodb/rs0-3 --smallfiles
       --oplogSize 128
9  mongod --replSet rs0 --port 27021 --dbpath /srv/mongodb/rs0-4 --smallfiles
       --oplogSize 128
10
11 // Iniciamos la consola de mongo
12 mongo
13
14 // Iniciamos el clúster y añadimos los nodos con sus roles (los secundarios al
       principio sin privilegios para asegurarnos que serán secundarios, los privilegios
       (prioridad y voto) se los otorgamos luego, y el árbitro)
15 rs.initialize()
16 rs.add({host:"localhost:27018", priority:0, votes:0})
17 rs.add({host:"localhost:27019", priority:0, votes:0})
18 rs.add({host:"localhost:27020", priority:0, votes:0})
19 rs.addArb("localhost:27021")
20
21 // Para configurar los nodos 1, 2 y 3 como secundarios.
22 var cfg = rs.conf()
23 cfg.members[1].priority = 1
24 cfg.members[1].votes = 1
25 cfg.members[2].priority = 1
26 cfg.members[2].votes = 1
27 cfg.members[3].priority = 1
28 cfg.members[3].votes = 1
29 rs.reconfig(cfg)
30
```

```
31 // Configuramos las opciones de escritura
32 cfg.settings.getLastErrorDefaults = {w:1, j:true, wtimeout:5000}
33 rs.reconfig(cfg)
```

Ahora me he conectado al clúster desde ***Mongo Compass*** eligiendo la opción de lectura *primaryPrefered* y he que aparecía lo siguiente siguiente:

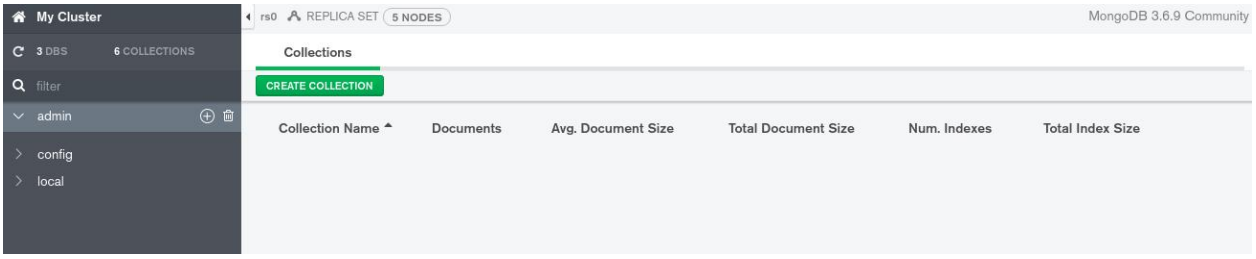

Figura 1: Mongo Compass conectado al clúster

En la esquina superior de nuestra izquierda podemos ver: ***"rs0 REPLICA SET 5 NODES"*** es decir que hemos credo justo lo que queríamos, y de momento dentro solo tenemos las bases de datos que se generan por defecto. Como andar arrancando los 5 servidores para trabajar no es eficaz, el resto del ejercicio lo realizaré sobre un único servidor local, pero esta primera parte me parecía un requisito esencial para comprender la potencia que nos proporciona Mongo.

2. Selección e importación de datos

Antes de ponerme a mostrar el trabajo realizado sobre los datos he de aclarar 2 cosas. La primera es que no he elegido los datos por que me interesasen de cara a realizar un análisis ni porque me pareciera que de ellos se podría extraer una información sorprendente, los he elegido pensando en que me diesen el máximo juego a la hora de usar los comandos de Mongo, por ello he buscado un fichero que contuviese datos de distintos tipos y varios documentos embebidos. Dado este enfoque se deduce la segunda aclaración, que es que voy a invertir el orden propuesto para la entrega. En una primera etapa manipularé los datos con el fin de explorar las posibilidades que ofrece Mongo, en la siguiente los manipularé con un fin específico y más académico, aunque posiblemente de forma inconexa.
Hechas las aclaraciones pertinentes se podrá comprender que vaya a usar la **colección** de tweets que se recomendó. Es una base bastante grande (de 1.6 Mb con 966 documentos) que contiene datos de todo tipo (aunque, por ejemplo, contiene 1 solo documento con coordenadas), multiples documentos embebidos y bastante información duplicada. Todo esto me va a servir para depurar los datos, rellenar algunos campos (que me inventaré) y en general manipular la colección de distintas formas. Para cargar esta base de datos he descargado el fichero *.bson* de **GitHub** y he usado los siguientes comandos en mi consola linux:

```
1 // Podría usar variables tipo "path/to/old/file", "path/to/new/file",
      databaseNameame, collectionName, etc. Pero así se entenderá mejor cuando marque
      los comandos sobre la collección.
2 bsondump --outfile ~/Desktop/masterBD/NoSQL/tweets.json ~/Descargas/tweets.bson
3 mongoimport --db entrega --collection tweets_0 --type json --file
      ~/Desktop/masterBD/NoSQL/tweets.json --drop --stopOnError
```

3. Manipulación de los datos

He echado un vistazo a los documentos en ***Mongo Compass*** para saber lo que estaba manejando y he encontrado varias cosas que me gustaría corregir. En primer lugar el campo *id str* es el campo id convertido en una variable tipo **string**. Lo mismo ocurre con las parejas de campos *in reply to status id - in reply to status id str* e *in reply to user id - in reply to user id str*. Toda esta información duplicada la voy a eliminar ya que solo entorpece el análisis de los datos, quedándome en cada caso la variable del tipo que más me interese. Si miramos los docuentos que tienen el campo *geo* no nulo (únicamente hay uno), veremos que el campo *coordinates* aparece embebido dentro de *geo*, sin embargo vemos que *coordinates* aparece también como un campo del documento principal, de nuevo se trata de información duplicada que eliminaremos, además como tan solo tenemos un documento con un valor no nulo en el campo *geo*, rellenaremos este campo con documentos embebidos con información aleatoria para otros documentos. Podemos también ver que hay tan solo 5 documentos con un valor no nulo en el campo *place*. Cuando este campo no es nulo contiene un documento embebido, con varios campos nuevos, de entre ellos los campos *attributes* y *contained within* aparecen a su vez como documentos embebidos, sin embargo están vacíos para todos los documentos, así que los eliminaremos también. Finalmente lidiaremos con el campo *user*, este campo contiene un documento con mucha información, en primer lugar hay que señalar que el campo *profile background image url https* y *profile background image url*, contienen la misma url, solo cambia el protocolo de seguridad, así que eliminaremos el más inseguro. En *user* tenemos además múltiple información del perfil de usuario que a priori va a ser inútil de cara al análisis, sin embargo, no la borraré. Finalmente voy a borrar el campo *extended entities* ya que muchos documentos no lo presentan, y no aporta información que vaya a usar.
Lo primero que he hecho es duplicar mi colección, y llamar a la nueva **tweets 1**, así la puedo actualizar sin miedo. Los trozos de código que presentaré a continuación no constituyen todo el código que he utilizado para consultar y trabajar con los datos, son las partes esenciales para justificar los cambios que se han producido en la colección, ya que no me parecía de especial interés mostrar el mismo comando en múltiples ocasiones (y aún así hay varios comandos que se repetirán).

3.1. Campos duplicados como variables tipo string

Ahora voy a eliminar el campo *id str*, conservando el campo *id*. Primero he comparado ambos campos para ver que eran iguales en todos los documentos y luego he borrado. El código empleado es el siguiente:

```
tt = db.tweets_1
// Para conocer el n de documentos
tt.find().count()
// No puedo usar $toString ni $convert ni $to<loquesea> ya que mi version de mongo es
    3.9.1
// Paso el id_tipo string a tipo int y los comparo, para ver que todos son iguales
tt.find().forEach(function(x){
    x.id_str = parseInt(x.id_str);
    tt.save(x);
});
tt.aggregate([
    {$addFields: {idsareequal: {$cond: [{$eq: ["$id", "$id_str"]}, 1, 0]}}},
    {$match: {idsareequal: 1}},
    {$count: "Documentos con id_str = id"}
])
/* Salida---------------------------------
    {
        "Documentos con id_str = id" : 966
    }
--------------------------------------- */
// Los campos coinciden en los 966 documentos luego elimino el campo id_str
tt.update([
    {_id:{$exists:true}},
    {$unset:{"id_str":""}},
    {multi:true}
```

```
])
// Inspeccionamos la coleccion ahora
db.getCollection('tweets_1').find()
// Y vemos que el campo id_str ha desaparecido
```

A continuación me dispuse a repetir el proceso con los campos: *in reply to status id*, *in reply to status id str*, *in reply to user id* e *in reply to user id str*, en las parejas correspondientes. No obstante, a la vista de que los primeros documentos presentaban **null** en este campo, conte cuantos documentos tenían estos campos nulos, y eran 965, es decir, existía un único documento que no tenía nulos estos 4 campos. Comprobé que el documento no coincidiese con el que no tenía nulas las coordenadas, y viendo que no era así me decidí a eliminar esos 4 campos, ya que al tener un solo valor no iba a poder visualizar interacciones entre los usuarios. Además he observado que ocurre lo mismo con el campo vecino *in reply to screen name* y que el campo *truncated* es **false** para todos los documentos, luego no aporta ninguna información, así que lo eliminé también. El código empleado lo adjunto para una de las 4 variables para evitar redundancias.

```
// Cuento los documentos con el campo nulo
tt.aggregate([
    {$match: {in_reply_to_status_id: {$eq: null}}},
    {$count: "Documentos con in_reply_to_status_id = null"}
])
/* Salida---------------------------------
    {
        "Documentos con in_reply_to_status_id = null" : 965
    }
--------------------------------------- */
// Visualizo el documento con el campo no nulo
tt.aggregate([
    {$match: {in_reply_to_status_id: {$ne: null}}},
])
// Elimino el campo (aquí incluí todos los campos)
tt.update(
    {_id: {$exists:true}},
    {$unset: {in_reply_to_status_id:"",
                in_reply_to_status_id_str:"",
                in_reply_to_user_id:"",
                in_reply_to_user_id_str:"",
                in_reply_to_screen_name:""}},
    {multi:true}
)

// Cuento los documentos con el campo false
tt.aggregate([
    {$match: {truncated: {$eq: false}}},
    {$count: "Documentos con truncated = false"}
])
/* Salida---------------------------------
    {
        "Documentos con truncated = false" : 966
    }
--------------------------------------- */
tt.update(
    {_id: {$exists:true}},
    {$unset: {truncated:""}},
    {multi:true}
)

// Inspeccionamos la coleccion ahora
db.getCollection('tweets_1').find()
// Veo que los campos han desaparecido
```

3.2. Trabajando con las coordenadas

Ahora procedo a eliminar el campo *coordinates* que aparece duplicado, y a rellenar *geo* con coordenadas inventadas en varios documentos, para ello he generado los 120 documentos que aparecen recogidos en el Apéndice 6.1

```
// Eliminamos el campo coordinates
tt.update(
    {_id: {$exists:true}},
    {$unset: {coordinates:""}},
    {multi:true}
)
// Rellenamos el campo geo (tofillgeo es un array con los documentos del apendice)
counter = 0
tt.find({favorite_count:{$gte:120}, geo:{$eq:null}}).limit(120).forEach(function(x){
    x.geo = tofillgeo[counter];
    counter = counter +1
    tt.save(x);
});
```

3.3. Los documentos embebidos en *user* y *place*

De estos documentos tan sólo borraremos los campos indicados más arriba, es decir, las urls repetidas y los subdocumentos vacíos.

```
// Eliminamos los subcampos en user
tt.update(
    {_id: {$exists:true}},
    {$unset: {"user.profile_background_image_url":"",
              "user.profile_image_url":""}},
    {multi:true}
)
// Eliminamos los subcampos en place
tt.update(
    {place: {$ne: null}},
    {$unset: {"place.attributes":"",
              "place.contained_within":""}},
    {multi:true}
)
// Inspeccionamos la coleccion ahora
db.getCollection('tweets_1').find({place:{$ne:null}})
// Veo que los campos han desaparecido
```

3.4. Otros campos prescindibles

En este momento me di cuenta también de que el campo contributors aparecía como nulo en muchas ocasiones, consulté si era así en todos los documentos y así era, así que lo elimine. Igualmente noté que el campo *possibly sensitive appealable* solo era **false** o nulo de forma que decidí borrarlo también. Igualmente los campos *retweeted* y *favorited* son siempre **false** así que los eliminé también y lo mismo haré con *extended entities*.

```
// Documentos con contributors nulo
tt.aggregate([
    {$match: {contributors: {$eq: null}}},
    {$count: "Documentos con contributors nulo"}
```

```
])
// Eliminamos el campo contributors
tt.update(
    {_id: {$exists:true}},
    {$unset: {contributors:""}},
    {multi:true}
)
//Documentos con pos_sens iguales
tt.aggregate([
    {$addFields:
        {sensequals:
            {$cond:
                [{$eq: ["$possibly_sensitive", "$possibly_sensitive_appealable"]},
                 1,
                 0]}}},
    {$match: {sensequals: 1}},
    {$count: "Documentos con sens iguales"}
])
/* Salida----------------------------------
    {
        "Documentos con sens iguales" : 964
    }
---------------------------------------- */
// Documentos cos pos_sens = true y pos_sens_app = false (habia visto muchos false
    comunes)
tt.aggregate([
    {$match: {possibly_sensitive:true, possibly_sensitive_appealable:false}},
    {$count: "Documentos con pos_sens distintas"}
])
/* Salida----------------------------------
    {
        "Documentos con pos_sens distintas" : 2
    }
---------------------------------------- */
// Eliminamos los campos indicados
tt.update(
    {_id: {$exists:true}},
    {$unset: {possibly_sensitive_appealable:"",
                favorited:"",
                retweeted:"",
                extended_entities:""}},
    {multi:true}
)
```

4. Análisis de los datos

Ahora que ya he limpiado a mi gusto la colección voy a empezar a ejecutar comandos de análisis sobre ella, como ya he dicho no pretendo realizar un análisis exhaustivo que me conduzca a unas conclusiones elaboradas (creo que no es el objetivo del módulo), sino usar múltiples comandos de mongo para así coger soltura a la hora de realizar consultas sobre estas bases de datos.
El análisis de ***Mongo Compass*** de la colección es de cierta ayuda pero tampoco aporta información significativa, así que no voy a determe en él. Paso directamente a realizar un script que me ayude a encontrar cierta información.

4.1. Promedio de retweets y frecuencia según el lenguaje

El código empleado para la búsqueda y las soluciones obtenidas han sido

```
tt.aggregate([
    {"$group" : {"_id" : {"lengua" : "$lang"},
                 "media" : {$avg : "$retweet_count"},
                 "conteo" : {$sum : 1}}},
    {"$sort" : {"media" : -1}}
    ])
```

Lenguaje	Media	Conteo
it	831.0	2.0
ht	393.166666666667	6.0
es	210.0	21.0
tl	116.333333333333	3.0
en	113.578347578348	702.0
in	103.777777777778	9.0
und	96.0	10.0
pt	58.4285714285714	7.0
tr	52.6506024096386	166.0
de	27.0	3.0
fr	11.5789473684211	19.0
ro	2.0	4.0
sk	2.0	1.0
zh	1.0	3.0
da	1.0	2.0
ru	1.0	1.0
nl	1.0	2.0
fa	1.0	2.0
ja	1.0	3.0

Tabla 1: Promedio retweets y apariciones por lenguaje

Aquí ya nos llevamos algunas sorpresas, por ejemplo el inglés no aparece hasta la cuarta posición, aunque es muy mayoritario en cuanto a apariciones, y el líder de la tabla es el italiano aunque aparece tan solo dos veces. Mención especial merece el español, que teniendo un número significativo de apariciones tiene mayor promedio de retweets que el inglés.

4.2. Número total de favoritos por número de seguidores según localización del usuario

En este comando voy a agrupar por el contenido del campo *location*, a dividir las veces que se ha marcado el perfil como favorito (*user.favourites count* entre los *user.friends count* y a sacar las métricas máxima, media y mínima del conjunto. Como algunos usuarios no tienen amigos, primero he sumado uno a este campo en todos los usuarios, lo que equivale a considerar que todos los usuarios son amigos de sí mismos.

```
tt.find().forEach(function(x){
    x.user.friends_count = x.user.friends_count +1;
    tt.save(x);
});

tt.aggregate([
    {$addFields: {fav_fri: {$divide: ["$user.favourites_count",
    "$user.friends_count"]}}},
    {$group : {"_id" : "$user.location",
               "maximo" : {$max: "$fav_fri"},
               "medio" : {$avg: "$fav_fri"},
               "minimo" : {$avg: "$fav_fri"},
               "conteo" : {$sum : 1}}}
    ])
```

Las soluciones de esta consulta, dada su longitud aparecen recogidas en la tabla 4, en el apéndice 6.2. El ellas vemos que con bastante frecuencia los usuarios dejan en blanco el campo *localización*, pero además podemos notar que

4.3. Selección de los tweets con más de 2 hashtags y de una mención

: Esta consulta no la voy a agrupar (ya que tan sólo hay 16 casos) si no que la voy a proyectar, para emplear así comandos nuevos y mostrar a la salida el *screen name* y el número de hashtags y menciones.

```
tt.aggregate([{
     $addFields:{
        hashtags_size:{
            $cond:{
                if: { $isArray: "$entities.hashtags" },
                then: {$size: "$entities.hashtags"},
                else: null}},
        mentions_size:{
            $cond:{
                if: { $isArray: "$entities.user_mentions" },
                then: {$size: "$entities.user_mentions"},
                else: null}}}},
    {$match: {$and: [{hashtags_size: {$gt: 2}}, {mentions_size: {$gt:1}}]}},
    {$project: {
        "_id" : 0,
        "user.screen_name" : 1,
        "hashtags_size" : 1,
        "mentions_size" : 1}},
    {$sort: {"n_mentions": -1}}
])
```

Sorprende ver que el *screen name* coincide en múltiples casos, lo que nos hace pensar que la colección contiene tweets de unos pocos usuarios, y de los que más menciones y hashtags usan son menos aún ya que en la tabla aparecen solo 3 nombres diferentes.

Screen Name	Nº hashtags	Nº menciones
cnnbrk	4	2
ameanmbot	3	2
ameanmbot	6	2
ameanmbot	5	2
ameanmbot	3	2
IzmirGDG	3	2
ameanmbot	3	2
ameanmbot	3	2
ameanmbot	3	2
ameanmbot	4	2
ameanmbot	4	2
ameanmbot	4	2
ameanmbot	6	2
ameanmbot	3	2
ameanmbot	3	2
Java Agent	7	2

Tabla 2: Usuarios con más hashtags de 2 y mas menciones que 1

4.4. Promedio de retweets por hora de publicación

Este caso nos interesa para ver cómo trabajar con fechas, lo que haré será crear el campo *pub hour*, agrupar los documentos por este campo y ordenarlos por número de retweets. Como el campo *created at* es tipo string primero tendré que pasarlo a fecha.

```
tt.aggregate([
    {$addFields: {pub_hour: {$hour: {$dateFromString: {dateString: "$created_at"}}}}},
    {$group: {"_id" : "$pub_hour",
              "ret_avg" : {$avg : "$retweet_count"}}},
    {$sort: {"ret_avg": -1}}
])
```

Hora de publicación	promedio de retweets
14	568.764705882353
16	166.192307692308
18	109.48
15	104.373376623377
17	76.5833333333333
7	66.6470588235294
8	44.9649122807018
6	30.8070175438597
9	9.0

Tabla 3: Promedio de retweets por hora de publicación

Los resultados de esta consulta si son bastante interesantes ya que nos muestran que los retweets varían mucho según la hora a la que los publiques, entre las 14 y las 18 horas las publicaciones registran una actividad mucho mayor que durante las primeras horas del día.

4.5. Comparación entre la búsqueda por expresiones regulares de texto y la creación de índices sobre el contenido del tweet

Por último quería explotar la potencia de la creación de índices de texto en ***MongoDB*** junto con la de la búsqueda de expresiones regulares. No obstante he leido que esto no está implementado así que he decidido comparar el rendimiento de la base de datos mediante uno y otro camino. No se trata de una colección muy poblada así que a priori la creación del índice sería innecesaria, pero servirá para la comaración.

```
1 tt.createIndex({"text" : "text"})
2 tt.find({$text : {$search : "football"}}) //encuentra 2 en 0.002s
3 tt.find({text : {$regex: "football"}}) // encuentra 1 en 0.009
4 tt.find({text : {$regex: "[f,F]ootball"}}) // encuentra 7 en 0.017s
5 tt.find({$text : {$search : "@Barcelona"}}) //encuentra 3 en 0.002s
6 tt.find({text : {$regex: "Barcelona"}})  // encuentra 6 en 0.009
7 tt.find({$text : {$search : "!"}}) //encuentra 0 en 0.002s
8 tt.find({text : {$regex: "!"}}).count()  // encuentra 100 en 0.008
```

Tras varias consultas (aqui recojo 7 significativas) he notado que la búsqueda sobre el índice con el comando **$search** nos permite buscar palabras completas, no distingue entre mayúsculas y minúsculas y es más rápido. De esta forma si para usar como un diccionario, es decir, para buscar coicidencia por palabras es la mejor opción. Las búsquedas regulares son más lentas pero nos permiten la libertad propia de esta herramienta a la hora de buscar combinaciones de caracteres, y si buscamos tweets que contengan preguntas o exclamaciones son la mejor opción, ya que devuelve los tweets que contengan el signo especificado, con el comando **$search** tendríamos que poner la última palabra con el signo adherido como último caracter, es decir, esta búsqueda sería imposible. Es por tanto que usaré el índice para mejorar la eficacia de is busquedas, y seguramente como un método muy optimizado de filtrar los documentos, pero para búsquedas específicas de texto o caracteres tendré que recurrir a las expresiones regulares, que funcionarán más lento pero puedo buscar cosas más concretas.

5. Conclusiones

Tras trabajar estás semanas y durante la realización del trabajo con ***MongoDB*** he de reconocer que me ha asombrado lo práctico que resulta. La forma de almacenamiento de la información estructurada me resultó mucho más natural que esta fórmula documental que funciona como una muñeca rusa. Pero una vez te acostumbras a trabajar con este modelo de agrupación de la información lo cierto es que la navegación, las consultas e incluso el análisis se simplifican enormemente.
Estoy muy contento con el trabajo realizado e ilusionado con esta herramienta y desde luego creo que coger soltura a la hora de manejar este modelo de almacenamiento de datos es imprescindible ya que me parece que no solo es una forma extremadamente inteligente y eficiente de guardar la información sino que también permite una enorme libertad en cuanto a la naturaleza de los propios datos.
Solo queda por tanto darle las gracias por habernos dado las primeras pautas en el manejo de este nuevo paradigma de las bases de datos.

6. Apéndice

6.1. Para rellenar *geo*

```
{"type" : "Point", "coordinates" : [9.090009,126.162783]},
{"type" : "Point", "coordinates" : [7.2466086,16.4346979]},
{"type" : "Point", "coordinates" : [37.039991,115.667208]},
{"type" : "Point", "coordinates" : [-7.3350849,111.46038]},
{"type" : "Point", "coordinates" : [59.418208,60.514706]},
{"type" : "Point", "coordinates" : [31.129098,120.839842]},
{"type" : "Point", "coordinates" : [38.322579,23.3204309]},
{"type" : "Point", "coordinates" : [35.026516,111.00746]},
{"type" : "Point", "coordinates" : [22.923431,112.732632]},
{"type" : "Point", "coordinates" : [-21.2001343,24.8656498]},
{"type" : "Point", "coordinates" : [14.1811693,-16.8502879]},
{"type" : "Point", "coordinates" : [45.0075322,19.8227166]},
{"type" : "Point", "coordinates" : [36.035581,102.822686]},
{"type" : "Point", "coordinates" : [40.6236334,-8.503294]},
{"type" : "Point", "coordinates" : [15.847683,120.9187827]},
{"type" : "Point", "coordinates" : [10.2511369,123.7810708]},
{"type" : "Point", "coordinates" : [-7.9839081,112.6213938]},
{"type" : "Point", "coordinates" : [-0.3030988,36.080026]},
{"type" : "Point", "coordinates" : [-4.9333,122.5167]},
{"type" : "Point", "coordinates" : [33.6450129,133.4789243]},
{"type" : "Point", "coordinates" : [53.1210091,17.9794988]},
{"type" : "Point", "coordinates" : [23.0664459,113.1964636]},
{"type" : "Point", "coordinates" : [-2.4050206,11.3572893]},
{"type" : "Point", "coordinates" : [36.276526,109.351019]},
{"type" : "Point", "coordinates" : [56.2862076,43.0264348]},
{"type" : "Point", "coordinates" : [8.1243943,124.0495898]},
{"type" : "Point", "coordinates" : [41.5780148,19.6966667]},
{"type" : "Point", "coordinates" : [10.5729257,-69.7078423]},
{"type" : "Point", "coordinates" : [52.6630895,38.4290322]},
{"type" : "Point", "coordinates" : [9.0592797,-79.4154819]},
{"type" : "Point", "coordinates" : [9.2394244,5.3102505]},
{"type" : "Point", "coordinates" : [44.840524,82.353656]},
{"type" : "Point", "coordinates" : [58.249304,57.783316]},
{"type" : "Point", "coordinates" : [22.517585,113.39277]},
{"type" : "Point", "coordinates" : [28.654639,116.144136]},
{"type" : "Point", "coordinates" : [5.8320421,-74.5833539]},
{"type" : "Point", "coordinates" : [47.2550409,-1.5401497]},
{"type" : "Point", "coordinates" : [48.2618635,37.8561647]},
{"type" : "Point", "coordinates" : [48.679041,5.8805771]},
{"type" : "Point", "coordinates" : [60.0557978,24.4154907]},
{"type" : "Point", "coordinates" : [37.6969518,127.8886827]},
{"type" : "Point", "coordinates" : [40.1941227,-8.8629472]},
{"type" : "Point", "coordinates" : [-6.7654544,108.1682147]},
{"type" : "Point", "coordinates" : [-9.8185351,147.6609064]},
{"type" : "Point", "coordinates" : [14.5694485,121.1877062]},
{"type" : "Point", "coordinates" : [-38.9770913,-68.0772116]},
{"type" : "Point", "coordinates" : [55.3851413,39.0323223]},
{"type" : "Point", "coordinates" : [7.1473706,125.4889603]},
{"type" : "Point", "coordinates" : [-4.76732,11.8627984]},
{"type" : "Point", "coordinates" : [9.141067,-12.664461]},
{"type" : "Point", "coordinates" : [40.609731,120.756479]},
{"type" : "Point", "coordinates" : [-5.0768453,105.0177539]},
{"type" : "Point", "coordinates" : [7.6988579,-72.3607108]},
{"type" : "Point", "coordinates" : [57.6437016,32.4624195]},
{"type" : "Point", "coordinates" : [55.6700613,12.5468826]},
{"type" : "Point", "coordinates" : [39.711179,-8.6689187]},
```

```
{"type" : "Point", "coordinates" : [57.9289181,12.5332741]},
{"type" : "Point", "coordinates" : [49.1169513,101.449837]},
{"type" : "Point", "coordinates" : [52.7572749,78.2041704]},
{"type" : "Point", "coordinates" : [41.1974753,47.1571241]},
{"type" : "Point", "coordinates" : [49.4774391,43.8548861]},
{"type" : "Point", "coordinates" : [31.0409483,31.3784704]},
{"type" : "Point", "coordinates" : [-6.2810348,106.8280147]},
{"type" : "Point", "coordinates" : [-9.6099,119.6237]},
{"type" : "Point", "coordinates" : [45.5833482,-73.4372336]},
{"type" : "Point", "coordinates" : [16.1160065,100.8451467]},
{"type" : "Point", "coordinates" : [-42.8821377,147.3271949]},
{"type" : "Point", "coordinates" : [7.0086472,100.4746879]},
{"type" : "Point", "coordinates" : [-8.5067,115.2298]},
{"type" : "Point", "coordinates" : [-6.511,106.0243]},
{"type" : "Point", "coordinates" : [28.714796,115.730847]},
{"type" : "Point", "coordinates" : [-25.6221943,-54.5655489]},
{"type" : "Point", "coordinates" : [-6.0924606,105.9112424]},
{"type" : "Point", "coordinates" : [2.941214,-73.207468]},
{"type" : "Point", "coordinates" : [55.1264168,39.7951718]},
{"type" : "Point", "coordinates" : [27.347756,118.449526]},
{"type" : "Point", "coordinates" : [39.6545102,-7.6712121]},
{"type" : "Point", "coordinates" : [27.9937685,120.6784858]},
{"type" : "Point", "coordinates" : [-7.6323091,110.4560006]},
{"type" : "Point", "coordinates" : [7.7916349,122.7785327]},
{"type" : "Point", "coordinates" : [3.9671435,103.40391]},
{"type" : "Point", "coordinates" : [31.2777088,49.5993167]},
{"type" : "Point", "coordinates" : [20.5593774,-101.2049226]},
{"type" : "Point", "coordinates" : [49.9333641,16.9699769]},
{"type" : "Point", "coordinates" : [25.33992,119.148479]},
{"type" : "Point", "coordinates" : [29.7857853,-95.8243956]},
{"type" : "Point", "coordinates" : [54.2866841,48.2335118]},
{"type" : "Point", "coordinates" : [5.6661517,-0.1582766]},
{"type" : "Point", "coordinates" : [49.794945,17.865273]},
{"type" : "Point", "coordinates" : [29.5530941,118.9353968]},
{"type" : "Point", "coordinates" : [52.4818411,19.2910855]},
{"type" : "Point", "coordinates" : [51.95914,21.53057]},
{"type" : "Point", "coordinates" : [38.994349,-1.8585424]},
{"type" : "Point", "coordinates" : [-6.957504,112.834114]},
{"type" : "Point", "coordinates" : [5.8734472,124.8979544]},
{"type" : "Point", "coordinates" : [51.5749669,4.6259996]},
{"type" : "Point", "coordinates" : [43.3211671,45.9920397]},
{"type" : "Point", "coordinates" : [59.4294432,24.5475697]},
{"type" : "Point", "coordinates" : [31.690215,113.382458]},
{"type" : "Point", "coordinates" : [16.4267638,103.1048449]},
{"type" : "Point", "coordinates" : [-3.366667,33.95]},
{"type" : "Point", "coordinates" : [-8.5803424,116.3654707]},
{"type" : "Point", "coordinates" : [-30.604304,-71.1969882]},
{"type" : "Point", "coordinates" : [-28.6454883,-53.605355]},
{"type" : "Point", "coordinates" : [31.026665,119.910952]},
{"type" : "Point", "coordinates" : [0.2213005,99.634135]},
{"type" : "Point", "coordinates" : [23.7247599,108.8076195]},
{"type" : "Point", "coordinates" : [-19.5097867,-41.0165586]},
{"type" : "Point", "coordinates" : [44.6381341,41.9381568]},
{"type" : "Point", "coordinates" : [13.4935504,39.465738]},
{"type" : "Point", "coordinates" : [58.3776252,26.7290062]},
{"type" : "Point", "coordinates" : [-7.7177,114.0741]},
{"type" : "Point", "coordinates" : [31.145454,121.121123]},
{"type" : "Point", "coordinates" : [-4.5585849,105.4068079]},
{"type" : "Point", "coordinates" : [49.9713653,29.7636724]},
{"type" : "Point", "coordinates" : [-4.1164768,-38.6949149]},
```

```
{"type" : "Point", "coordinates" : [35.78668,36.4726]},
{"type" : "Point", "coordinates" : [31.2303904,121.4737021]},
{"type" : "Point", "coordinates" : [32.910459,119.85254]},
{"type" : "Point", "coordinates" : [15.5805384,-61.3297698]}
```

6.2. Soluciones segunda consulta del análisis

El tamaño de la fuente es necesario para que la tabla quepa en una página.

Localización	Máximo	Media	Mínimo	Conteo
Freiburg,Germany	0.520686963309914	0.520686963309914	0.520686963309914	1.0
Denver,CO,USA	3.23641304347826	3.23641304347826	3.23641304347826	2.0
Worldwide	8.36666666666667	8.36666666666667	8.36666666666667	1.0
İstanbul/Turkey	0.487394957983193	0.487394957983193	0.487394957983193	1.0
Web	0.0	0.0	0.0	1.0
istanbul	22.8360655737705	12.0067120321683	12.0067120321683	2.0
US,Paris,London,Berlin	0.445103857566766	0.445103857566766	0.445103857566766	1.0
Türkiye	1.54054054054054	1.54054054054054	1.54054054054054	4.0
Ankara,Turkey	0.0	0.0	0.0	1.0
Turkey,Istanbul	0.152173913043478	0.152173913043478	0.152173913043478	1.0
Istanbul/Turkey	0.5	0.5	0.5	2.0
Redmond,WA,USA	0.230593607305936	0.230593607305936	0.230593607305936	1.0
ÚT37.575291,-122.341831	0.83596214511041	0.83596214511041	0.83596214511041	1.0
LINEhesabı	0.0	0.0	0.0	3.0
Romsey,England	2.20338983050847	2.20338983050847	2.20338983050847	1.0
neverland	2.29479768786127	2.29479768786127	2.29479768786127	1.0
LocationAllovertheworld.	2.77219626168224	2.77219626168224	2.77219626168224	1.0
Redmond,WA	1.01901140684411	0.289088751016537	0.289088751016537	4.0
Oakland,CA	0.014755693984816	0.014754545160265	0.014754545160265	3.0
London,England	36.8398268398268	36.5177505217204	36.5177505217204	33.0
C Redmond	0.0135699373695198	0.0135699373695198	0.0135699373695198	7.0
PaloAlto,CA	2.69871794871795	1.79634684312104	1.79634684312104	3.0
MountainView,CA	1.87721518987342	1.87721518987342	1.87721518987342	1.0
İzmir,Turkey	0.888888888888889	0.83546485260771	0.83546485260771	7.0
AméricaLatina	0.303030303030303	0.296969696969697	0.296969696969697	5.0
Brooklyn,NY	35.6965517241379	28.4411986006971	28.4411986006971	5.0
NewYork,NY	9.36317222600409	9.11784800349789	9.11784800349789	8.0
SanFrancisco,CA	11.8266666666667	3.04530121378813	3.04530121378813	7.0
London,UK	0.0	0.0	0.0	27.0
London	2.87209302325581	2.85439238147816	2.85439238147816	3.0
Allovertheworld	33.4975247524753	33.4975247524753	33.4975247524753	1.0
SanFrancisco	54.1512605042017	6.73106141785573	6.73106141785573	12.0
Istanbul,Turkey	3.30963302752294	2.99524204567636	2.99524204567636	2.0
Brighton,EastSussex,England	3.86105263157895	3.86105263157895	3.86105263157895	1.0
Vallauris,France	0.698973774230331	0.698973774230331	0.698973774230331	1.0
Vancouver,Canada	0.40894646785627	0.10794375501171	0.10794375501171	11.0
	104.566666666667	26.4868152026315	26.4868152026315	454.0
Germany	0.0575539568345324	0.0571410697528933	0.0571410697528933	5.0
Barcelona	0.39622641509434	0.39622641509434	0.39622641509434	40.0
Istanbul—Turkey	0.352941176470588	0.352941176470588	0.352941176470588	7.0
Tampa,FL	0.299803149606299	0.250627308074456	0.250627308074456	3.0
Everywhere	1.19748229740362	0.233428401194105	0.233428401194105	9.0
USA	2.64341085271318	2.16223783133256	2.16223783133256	9.0
Cary,NCUSA	2.82758620689655	2.81992337164751	2.81992337164751	3.0
Leeds/London	0.100990099009901	0.100990099009901	0.100990099009901	9.0
LosAngeles	1.04198473282443	1.00381679389313	1.00381679389313	4.0
NewYorkCity	0.389086595492289	0.313472177064302	0.313472177064302	66.0
Turkey	1.6734693877551	1.6734693877551	1.6734693877551	35.0
Sammamish,WAUSA	0.0	0.0	0.0	2.0
İzmir	0.0115359965819269	0.00993567223796143	0.00993567223796143	3.0
Melbourne,Australia	0.00997011247747028	0.0099237297393496	0.0099237297393496	9.0
Seattle,WA	0.0239520958083832	0.0239520958083832	0.0239520958083832	1.0
LA	0.0307167235494881	0.0306644842237236	0.0306644842237236	2.0
Canada,Winnipeg	0.0	0.0	0.0	1.0
Istanbul,TURKEY	1.01941747572816	1.01941747572816	1.01941747572816	1.0
Barranquilla	0.487654320987654	0.487654320987654	0.487654320987654	2.0
Milwaukee,Wisconsin	12.5698841698842	12.5698841698842	12.5698841698842	1.0
Cyprus	0.28486646884273	0.28486646884273	0.28486646884273	1.0
Ankara	7.4390243902439	6.37939591605218	6.37939591605218	8.0
namaste	0.935285053929122	0.935285053929122	0.935285053929122	1.0
ShelterIsland,UnitedStates	1.38352941176471	1.38044395116537	1.38044395116537	2.0
RedwoodShores,CA	1.79633867276888	1.78549204327118	1.78549204327118	3.0
Washington,DC	1.24376369111751e-05	1.24361933443142e-05	1.24361933443142e-05	3.0
SunnyvaleCA	0.0694294294294294	0.0661483909525551	0.0661483909525551	11.0
Philadelphia,PA	0.434889434889435	0.423841032751924	0.423841032751924	3.0
Acrosstheglobe	1.11158798283262	1.11158798283262	1.11158798283262	1.0
Istanbul	0.132978723404255	0.123244680851064	0.123244680851064	4.0
Cheshire,UK	0.153802972737701	0.152974611378629	0.152974611378629	16.0
sabrireyiziniz@gmail.com	0.0431472081218274	0.0431472081218274	0.0431472081218274	2.0
Istanbul-Turkiye	0.176470588235294	0.176470588235294	0.176470588235294	1.0
Anfield,Liverpool	0.0108466807696182	0.0108447348084264	0.0108447348084264	20.0
France	0.646992054483541	0.645100158090411	0.645100158090411	4.0
Istanbul/Turkey	1.34722222222222	1.34722222222222	1.34722222222222	51.0
RessamBob@outlook.com	88.7707317073171	88.4014886957993	88.4014886957993	3.0

Tabla 4: Máximo, promedio y mínimo del cociente favoritos entre amigos por Localización