Práctica 7 – NoSQL-Solución

Preparación:
0. Iniciar el servidor (Desde un terminal: mongod -dbpath datos)
1. Bajar del campus el fichero tweet.json y user.json
2. Desde un terminal de linux teclear
mongoimport --db test --collection tweet --file tweet.json
mongoimport --db test --collection user --file user.json

Importará 17834 tweets y 141 usuarios

Importante: En las siguientes consultas **no usar comillas simples (‘)**, solo comillas dobles

1) Vamos a contar mentiras. Define una consulta con una expresión regular ($regex) para encontrar el número de tweets (clave text) que contienen la subcadena “lies”, no importa si en mayúsculas o en minúsculas. Debe salír 162. Nota: No hace falta crear ningún índice.
Copia en la solución la consulta
Solución
db.tweet.find({text:{$regex:'lies', $options:”i”}}).count()

2) Crea un índice de tipo texto sobre la clave “text”. Llámale “itext”. Copia en la solución la instrucción para crear el índice.

Solución.
db.tweet.createIndex({text:”text”},”itext”)

3) Cuenta el número de veces que se repite la palabra “lies” (o derivados de esta palabra) usando el índice textual.
Copia en la solución la consulta (no hace falta copiar la creación del índice), que debe devolver el valor 135.
Solución
db.tweet.find({$text:{$search:"lies"}}).count()

4) Vamos a combinar los dos modos de buscar para ver sus diferencias. Escribamos una consulta ahora para obtener número de tweets con derivados de la palabra “lies” usando el índice textual (pregunta 3), pero que **no** contengan el substring “lies” como substring según el método de la pregunta 1). La solución debe ser 70.
Nota: Para asegurarse de que no se contiene el substring “lies” podemos acompañar “regex” de la expresión regular "^((?!lies).)*$"

Copia en la solución la consulta.
Solución:
db.tweet.find({$text:{$search:"lies"}, text:{"$regex" : "^((?!lies).)*$"}},{text:1,_id:0}).count()

5) Crea
- Un índice sobre la clave mentions de la colección user, descendente, con nombre “mendown”
- Otra sobre la clave followers de la misma colección, ascendente y único, con nombre

“follup”
Copia en la solución las instrucciones que han permitido crear los dos índices

Solución
db.user.createIndex({mentions:-1},"mendown")
db.user.createIndex({followers:1},"follup",{unique:1})

6) Escribe una instrucción que muestre las claves followers y mentions (solo estas claves), de la colección *user*, para aquellos usarios conn más de 30 seguidores.
Copia en la solución la consulta.

Solución
db.user.find({followers:{$gt:30}},{followers:1,mentions:1,_id:0})

7) ¿Qué índices se han usado? ¿cuántos documentos y claves ha hecho falta consultar para ejecutar la consulta anterior?
Copia en la solución:
Instrucción: instrucción/instrucciones que permiten ver el plan de ejecución con suficiente detalle y que te han servido opara averiguar esta información (no hace falta poner el resultado de la instrucción)
indices: nombres de los índices separados por comas (o “ninguno”),
documentos: copia una estructura de la forma “clave:valor” del documento JSON que ha devuelto la instrucción que contenta el número de documentos examinados
claves: copia una estructura de la forma “clave:valor” del documento JSON que ha devuelto la instrucción que contenta el número de claves examinadas

Solución
Instrucciones:db.user.explain("executionStats").find({followers:{$gt:30}},
{mentions:1,followers:1,_id:0})
Índices: “follup”
documentos: "totalDocsExamined" : 141
claves: "totalKeysExamined" : 141,

8) ¿Qué indice crearías para mejorar la eficiencia de esta consulta?
Copia en la solución:
índice: la instrucción para crear el índice
documentos: como en el ej. anterior
claves: como en el ej. anterior
explicación: ¿Alguno de los dos valores anteriores (“documentos” o “claves”) ha variado? (S/N). Si la respuesta es sí explicar por qué.

Solución
índice: db.user.createIndex({followers:1, mentions:1})
documentos: "totalDocsExamined" : 0
claves: "totalKeysExamined" : 141,
explicación: Sí, ahora "totalDocsExamined" : 0, No se examina ningún documento porque está

toda la informción en el índice, mejorando la eficiencia.

9) Escribe en el shell una consulta que muestre aquellos usuarios que tienen más “followers” que mentions, usando el operador “$where”
Copia en la solución:
consulta: la consulta
índices: nombre de los índices utilizados para resolverlos (“ninguno” si no hay)
explicación: por qué se usa ese índice (o por qué no se usa ninguno, en su caso)
rechazados:la lista de planes rechazados ([] para ninguno)
No hace falta decir cómo se ha averiguado el nombre de los índices ni la lista de planes rechazados.

Solución
consulta: db.user.find({$where:"this.followers>this.mentions"})
índices: ninguno
explicación: en un $where se ejecuta javascript, no funcionan los planes de ejecución
rechazados: []

10) Con los índices creados hasta ahora, ¿son igual de eficientes estas dos consultas?
a) db.user.explain().find().sort({followers:-1,mentions:-1})
b) db.user.explain().find().sort({followers:-1,mentions:1})
Índica en la solución
Más eficiente: a), b) o ninguna
Explicación: Cómo has llegado a esta conclusión (un frase o dos, incluyendo algún “clave:valor” de los planes de ejecución)

Solución
Más eficiente: a)
Explicación: en b) no se usa índice ("stage" : "SORT"). En a) sí (stage" : "IXSCAN")
