Parcialito V: NoSQL

nombre: Marcos Bianchi Fernández - **padrón:** 108921 - **fecha:** 08/11/2024

MongoDB

1.

Para tweets que tengan 1000 o más retweets (‘retweet_count’) y hayan sido creados después de las 12 del mediodía, obtener los ids, texto y cantidad de favoritos para los 5 con más favoritos (‘favorite_count’). Utilice una única consulta básica con find(<query>, <proyección>).sort({}).limit(<n>)

```
db.tweets.find(
    {
        $and: [
        { retweet_count: { $gte: 1000 }},
        { $expr: { $gte: [{ $hour: "$created_at"}, 12] }},
        ],
    },
    {
        _id: 1,
        full_text: 1,
        favorite_count: 1,
    })
    .sort({ favorite_count: -1 })
    .limit(5)
```

2.

Por cada hashtag y hora del día (00, 01, 02, …) obtener el total de favoritos conseguidos por tweets que contengan la palabra “futbol” en el texto. Se debe indicar si se ignoraron o no los tweets que no tienen hashtags (justificar). Se debe utilizar el pipeline de agregación.

```
[
  {
    $unwind: {
      path: "$entities.hashtags"
    },
  },
  {
    $group: {
      _id: {
        hashtag: "$entities.hashtags.text",
        hora: { $hour: "$created_at" },
      },
      favoritos_futbol: {
        $sum: {
          $cond: {
```

```
            if: { full_text: /futbol/ },
            then: "$favorite_count",
            else: 0
          },
        },
      },
    },
  },
]
```

La query de arriba ignora los posts sin hashtags, en caso de querer incluirlos podemos crear un hashtag especial, por ejemplo el: “sin-hashtag” para que actúe como placeholder. Y ahí agrupar a todos esos tweets.

3.

Dada la consulta: Anexo: Consulta ejercicio 3 o disponible en GitHub
Explicar qué sucede en cada paso del pipeline y en forma resumida qué resuelve la query completa.

```
[
    {
        // Primero se hace un match, nos quedamos con los documentos
        // brasileros que están en español o portugues.
        $match: {
            lang: /es|pt/,
            "place.country": "Brasil",
        },
    },
    {
        // Segundo agrupamos en nuevos documentos donde el nuevo id va a ser
        // `$in_reply_to_status_id_str` en caso de no ser `null`, en caso de serlo
        // se toma el valor que tenía el documento antes, osea, $_id.
        $group: {
            _id: {
                $ifNull: [ // coalesce
                    "$in_reply_to_status_id_str",
                    "$_id",
                ]
            },
            // Creamos un campo nuevo de tipo arreglo, llamado `tweets` al que
            // agregamos objetos como el id del tweet, su texto, el usuario que
            // lo hizo y la fecha de creación.
            tweets: {
                $push: {
                    tweet_id: "$_id",
                    text: "$full_text",
                    user: "$user",
                    created_at: "$created_at.date",
                },
            },
            // Creamos otro campo nuevo, llamado `avg_retweets` donde
```

```
            // guaramos un promedio de los retweets.
            avg_retweets: { $avg: "$retweet_count" },
        },
    },
    {
        // Tercero proyectamos, nos quedamos con los siguientes campos.
        $project: {
            // El campo `tweet` contiene un único `$_id` de tweet sacado del campo
            // anterior `tweets` donde se queda el que sea igual a un cierto `$_id`.
            tweet: {
                $arrayElemAt: [{
                    $filter: {
                        input: "$tweets",
                        as: "reply",
                        cond: {
                            $eq: ["$reply.tweet_id", "$_id"],
                        },
                    },
                }, 0],
            },
            // El campo `replies` contiene un arreglo sacado del campo anterior
            // `tweets` ordenado y elimina un cierto tweet si es igual a un
            // cierto `$_id`.
            replies: {
                $sortArray: {
                    input: {
                        $filter: {
                            input: "$tweets",
                            as: "reply",
                            cond: { // eliminamos el mismo tweet
                                $ne: ["$reply.tweet_id", "$_id"],
                            },
                        },
                    },
                    sortBy: {created_at: 1},
                },
            },
            // Un placeholder para que devuelva el `avg_retweets`.
            avg_retweets: 1,
        },
    },
]
```

La query devuelve los tweets que no son replies, junto con sus replies ordenados de más viejos a más nuevos junto con el promedio de retweets de todo el thread.

Neo4j

4. Muestre los familiares de Billy Moore que no han tenido participación en ningún crimen.

```
match (:Person{name: "Billy", surname: "Moore"})-[:FAMILY_REL]-(p:Person)
where not (p)-[:PARTY_TO]-(:Crime)
return p

// Resultado:
// 1. Roger Brooks
```

5. Muestre las (o las) persona(s) que ha(n) realizado más de 7 comunicaciones telefónicas.

```
match (p:Person)-[:HAS_PHONE]->(:Phone)<-[:CALLER]-(l:PhoneCall)
with p as llamador, COUNT(l) as llamados
where llamados > 7
return llamador

// Resultado:
// 1. Kimberly Wood
```