

**ANO**  
**2024**



# **UNINTER**

**CADERNO DE RESPOSTAS DA  
ATIVIDADE PRÁTICA DE:**

**NoSQL**

**ALUNO: GUILHERME MENESES RAMALHO  
RU: 4708628**

**Caderno de Resposta Elaborado por:  
Prof. MSc. Guilherme Ditzel Patriota**

## Atividade Prática – NEO4J

### Questão 01 – IMPORTAÇÃO DOS ARQUIVOS JSON E CRIAÇÃO DE NÓS E ARESTAS COM BASE NOS DADOS DOS ARQUIVOS (Usar quantas páginas forem necessárias)

**O QUE FAZER:** Após configurar seu banco de dados em branco (novo DBMS em versão 4.\*) e colocar no mínimo 3 e máximo 10 arquivos sequenciais do trabalho (dentre os 500 arquivos JSON disponibilizados) na pasta Import, crie um comando na linguagem do banco de dados Neo4j (Cypher), usando a biblioteca de importação de dados APOC, que leia os arquivos JSON desta pasta e crie os nós e relacionamentos com base nos dados neles. Faça a separação dos nós de mensagem em Tweet (mensagens originais), Retweet (mensagens repostadas), Quoted (mensagens que citam outras mensagens), Replied\_to (mensagens de resposta à outras mensagens) com uso do campo `data[x].ref_tweet.type` (CUIDADO! Este campo só aparece no JSON de mensagens não originais e o uso de UNWIND para acessá-la pode impedir a criação dos nós de mensagens originais). Apenas com esta separação será possível resolver a questão 02 do trabalho.

**Observação Importante:** Seu banco de dados precisará conter todas as informações para resolver as questões 02 e 03. Leia elas antes, para entender o que você deseja fazer em cada uma e quais os nós, relacionamentos e atributos você irá importar dos arquivos JSON para conseguir resolver todo o trabalho sem a necessidade de recriar todo o seu banco de dados apenas para uma questão.

- I. Apresentação dos comandos (apenas query Cypher) usados (não esquecer do identificador pessoal/seu RU como parte do seu código, como um nome de atributo dos nós ou dado de um atributo de nós). **Nenhum dos comandos apresentados aqui pode conter a palavra RETURN (Peso na nota: 12.5%):**

#### Primeiro

```
1 call apoc.load.directory('*.json') yield value
2 with value as arquivo
3 call apoc.load.json(arquivo) yield value
4 unwind value.data as tweet
5 unwind tweet.entities.hashtags as hashtag
6 merge (RU4708628:Tweet {tweet_id: tweet.id})
7 on create set RU4708628.mensagem = tweet.text
8 merge (u:User {user_id: tweet.author_id})
9 merge (h:Hashtag {tag: apoc.text.replace(apoc.text.clean(hashtag.tag), '^a-zA-Z0-9', '')})
10 merge (RU4708628)-[:POSSUI_A_HASHTAG]->(h)
11 merge (u)-[:USUARIO_TWEETOU]->(RU4708628)
```

#### Código:

```
call apoc.load.directory('*.json') yield value
with value as arquivo
call apoc.load.json(arquivo) yield value
unwind value.data as tweet
unwind tweet.entities.hashtags as hashtag
merge (RU4708628:Tweet {tweet_id: tweet.id})
on create set RU4708628.mensagem = tweet.text
merge (u:User {user_id: tweet.author_id})
merge (h:Hashtag {tag: apoc.text.replace(apoc.text.clean(hashtag.tag), '^a-zA-Z0-9', '')})
```

```
merge (RU4708628)-[:POSSUI_A_HASHTAG]->(h)
merge (u)-[:USUARIO_TWEETOU]->(RU4708628)
```

**Legenda:**

1. Carrega todos os arquivos com extensão ".json" do diretório.
2. Atribui cada um desses arquivos à variável chamada "arquivo".
3. Carrega todos os arquivos ".json".
4. Acessa os Tweets que estão dentro de "data" (um array com 99 tweets) nos arquivos ".json". Este caminho está sendo referenciado como "tweet".
5. A partir dos tweets, entra na estrutura "entities", acessa "hashtags" e as armazena na variável "hashtag".
6. Realiza um "MERGE" (criação de um nó, caso ainda não exista) dos tweets, usando o "tweet.id" como propriedade principal do nó.
7. Cria uma propriedade adicional no nó "Tweet" chamada "mensagem", atribuindo o conteúdo de "tweet.text".
8. Faz um "MERGE" do usuário como nó "User", atribuindo o ID do autor via "tweet.author\_id".
9. Cria as hashtags sem duplicações, removendo letras maiúsculas, espaços e underlines, unificando tudo em uma única forma padronizada.
10. Cria relacionamentos entre os tweets e suas respectivas hashtags.
11. Cria relacionamentos entre os usuários e os tweets publicados por eles.

**Segundo**

```
1 call apoc.load.directory('*.json') yield value
2 with value as arquivo
3 call apoc.load.json(arquivo) yield value
4 unwind value.data as tweet
5 unwind tweet.referenced_tweets as ref
6 merge (RU4708628:Tweet {tweet_id: tweet.id})
7 on match set RU4708628.tipo_ref = ref.type
```

**Código:**

```
call apoc.load.directory('*.json') yield value
with value as arquivo
call apoc.load.json(arquivo) yield value
unwind value.data as tweet
unwind tweet.referenced_tweets as ref
merge (RU4708628:Tweet {tweet_id: tweet.id})
on match set RU4708628.tipo_ref = ref.type
```

**Legenda:**

1. Carrega todos os arquivos com extensão ".json" do diretório.

2. Atribui cada um desses arquivos à variável chamada "arquivo".
3. Carrega todos os arquivos ".json".
4. Acessa os tweets que estão dentro de "data" nos arquivos ".json". Esse conteúdo é referenciado como "tweet".
5. Acessa os tweets e, dentro deles, a chave "referenced\_tweets", que é referenciada como "ref".
6. Realiza um MERGE com cada tweet e atribui como identificador o valor "RU4708628".
7. Adiciona uma nova propriedade ao tweet, atribuindo o tipo de Tweet.

### Terceiro

```
1 match (RU4708628:Tweet)
2 where RU4708628.tipo_ref = "retweeted"
3   remove RU4708628: Tweet
4   set RU4708628: Retweet
```

### Código

```
match (RU4708628:Tweet)
where RU4708628.tipo_ref = "retweeted"
  remove RU4708628: Tweet
  set RU4708628: Retweet
```

### Legenda

1. Seleciona um tweet e atribui no identificador RU4708628.
2. Seleciona o tweet onde houver "retweeted" no atributo "tipo\_ref" dos nós.
3. Remove o tipo do nó, que é Tweet.
4. Substitui o tipo do nó para Retweet.

Agora em graph os nós iram aparecer como Retweet, e não mais como Tweet.

### Quarto

```
1 match (RU4708628:Tweet)
2 where RU4708628.tipo_ref = "quoted"
3   remove RU4708628: Tweet
4   set RU4708628: Quoted
```

### Código

```
match (RU4708628:Tweet)
where RU4708628.tipo_ref = "quoted"
  remove RU4708628: Tweet
  set RU4708628: Quoted
```

**Legenda**

1. Seleciona um tweet e atribui no identificador RU4708628.
2. Seleciona o tweet onde houver "quoted" no atributo "tipo\_ref" dos nós.
3. Remove o tipo do nó, que é Tweet.
4. Substitui o tipo do nó para Quoted.

Agora em graph os nós iram aparecer como Quoted, e não mais como Tweet.

**Quinto**

```
1 match (RU4708628:Tweet)
2 where RU4708628.tipo_ref = "replied_to"
3   remove RU4708628: Tweet
4   set RU4708628: Replied_to
```

**Código**

```
match (RU4708628:Tweet)
where RU4708628.tipo_ref = "replied_to"
  remove RU4708628: Tweet
  set RU4708628: Replied_to
```

**Legenda**

1. Seleciona um tweet e atribui no identificador RU4708628.
2. Seleciona o tweet onde houver "replied\_to" no atributo "tipo\_ref" dos nós.
3. Remove o tipo do nó, que é Tweet.
4. Substitui o tipo do nó para Replied\_to.

Agora em graph os nós iram aparecer como Replied\_to, e não mais como Tweet.

- II. Apresentação dos prints do resultado (não esquecer do identificador, seu RU). Estes prints devem ser de cada tela do Neo4j Browser após a execução bem-sucedida de cada comando (não mostrar nenhum grafo nesta parte, apenas as telas de execução dos comandos da parte I. O uso de RETURN zerará a nota desta parte) (Peso na nota: 12.5%):**

**Primeiro**

```
1 call apoc.load.directory('*.json') yield value
2 with value as arquivo
3 call apoc.load.json(arquivo) yield value
4 unwind value.data as tweet
5 unwind tweet.entities.hashtags as hashtag
6 merge (RU4708628:Tweet {tweet_id: tweet.id})
7 on create set RU4708628.mensagem = tweet.text
8 merge (u:User {user_id: tweet.author_id})
9 merge (h:Hashtag {tag: apoc.text.replace(apoc.text.clean(hashtag.tag), '^a-zA-Z0-9}', '')})
10 merge (RU4708628)-[:POSSUI_A_HASHTAG]->(h)
11 merge (u)-[:USUARIO_TWEETOU]->[RU4708628]
```

Added 1249 labels, created 1249 nodes, set 1825 properties, created 2089 relationships, completed after 278 ms.

Table

Code

Added 1249 labels, created 1249 nodes, set 1825 properties, created 2089 relationships, completed after 278 ms.

### Legenda:

Foi atribuído 1249 tipos ou rótulos aos nós, criados 1249 nós, foi definido 1825 propriedades, criados 2089 relacionamentos e essa ação demorou 278 milissegundos.

### Segundo

```
1 call apoc.load.directory('*.json') yield value
2 with value as arquivo
3 call apoc.load.json(arquivo) yield value
4 unwind value.data as tweet
5 unwind tweet.referenced_tweets as ref
6 merge (RU4708628:Tweet {tweet_id: tweet.id})
7 on match set RU4708628.tipo_ref = ref.type
```

Added 413 labels, created 413 nodes, set 870 properties, completed after 174 ms.

Table

Code

Added 413 labels, created 413 nodes, set 870 properties, completed after 174 ms.

### Legenda:

Foi atribuído 413 tipos ou rótulos aos nós, criados 413 nós, foi definido 870 propriedades e essa ação demorou 174 milissegundos.



### Terceiro

```
1 match (RU4708628:Tweet)
2 where RU4708628.tipo_ref = "retweeted"
3   remove RU4708628: Tweet
4   set RU4708628: Retweet
```

Added 299 labels, removed 299 labels, completed after 2 ms.

Table

Code

Added 299 labels, removed 299 labels, completed after 2 ms.

### Legenda:

Foi atribuido 299 tipos ou rótulos aos nós, foi removido 299 tipos ou rótulos dos nós e essa ação demorou 2 milissegundos.

### Quarto

```
1 match (RU4708628:Tweet)
2 where RU4708628.tipo_ref = "quoted"
3   remove RU4708628: Tweet
4   set RU4708628: Quoted
```

Added 54 labels, removed 54 labels, completed after 1 ms.

Table

Code

Added 54 labels, removed 54 labels, completed after 1 ms.

### Legenda:

Foi atribuido 54 tipos ou rótulos aos nós, foi removido 54 tipos ou rótulos dos nós e essa ação demorou 1 milissegundo.

**Quinto**

```
1 match (RU4708628:Tweet)
2 where RU4708628.tipo_ref = "replied_to"
3   remove RU4708628: Tweet
4   set RU4708628: Replied_to
```



Added 104 labels, removed 104 labels, completed after 2 ms.



Added 104 labels, removed 104 labels, completed after 2 ms.

**Legenda:**

Foi atribuído 104 tipos ou rótulos aos nós, foi removido 104 tipos ou rótulos dos nós e essa ação demorou 2 milissegundos.

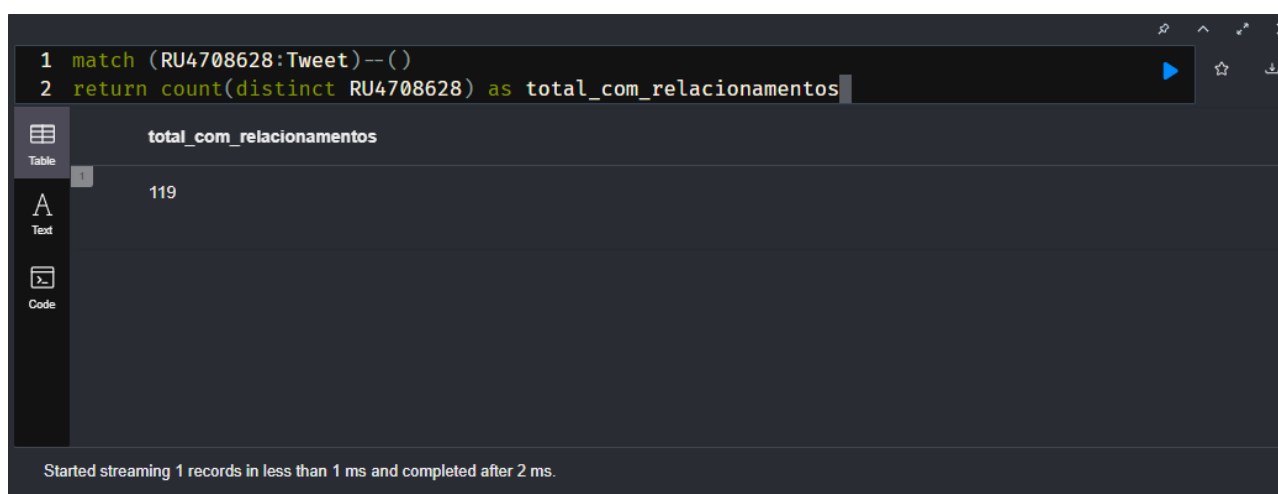


## Atividade Prática – NEO4J

### Questão 02 – DESCOBERTA DA HASHTAG PRINCIPAL (Usar quantas páginas forem necessárias)

**ENUNCIADO:** Você deve criar e executar um comando Cypher em seu banco de dados para descobrir qual hashtag está presente em todas as mensagens originais (excluindo mensagens de retweet, citação e resposta). Este comando **não deve** fazer uso da biblioteca **APOC**. Caso seu comando tente retornar mais de 300 nós, a configuração padrão do Neo4j Browser impedirá a exibição de mais de 300 nós. Seu comando **não pode** ser **MATCH (n) RETURN n;**.

- I. Apresentação do comando (apenas query Cypher) usado (não esquecer do identificador pessoal no comando, seu RU). **Este comando deve conter ao menos uma dupla MATCH/RETURN (sugerimos uso de subquery para resolução completa) (Peso na nota: 12.5%):**



The screenshot shows the Neo4j Browser interface. The query editor contains the following Cypher query:

```
1 match (RU4708628:Tweet)--()
2 return count(distinct RU4708628) as total_com_relacionamentos
```

The results panel shows a table with one column, `total_com_relacionamentos`, and one row with the value `119`.

Started streaming 1 records in less than 1 ms and completed after 2 ms.

#### Código:

```
match (RU4708628:Tweet)--()
return count(distinct RU4708628) as total_com_relacionamentos
```

#### Legenda:

1. Seleciona os tweets que possuem algum relacionamento e atribui ao identificador "RU4708628".
2. Retorna a contagem desses tweets, sem repetir, com o nome "total\_com\_relacionamentos".

Esse código foi utilizado inicialmente para verificar o total de tweets que possuem algum tipo de relacionamento, excluindo da contagem os tweets isolados (provavelmente porque foram excluídos ou pertencem a usuários que foram removidos).

```
1 match (h:Hashtag)
2 call {
3   with h
4   match (h) <- [:POSSUI_A_HASHTAG]-(RU4708628:Tweet)
5   return count(RU4708628) as total_relacionamentos
6 }
7 return h.tag, total_relacionamentos
8 order by total_relacionamentos desc
```

	h.tag	total_relacionamentos
1	"issoaglobonaomostra"	119
2	"fantastico"	15
3	"bbb20"	14
4	"bolsonaroday"	8
5	"depheliolopes"	7
6	"ptnuncamais"	7
7		

Started streaming 185 records after 1 ms and completed after 2 ms.

**Código:**

```
match (h:Hashtag)
call {
  with h
  match (h) <- [:POSSUI_A_HASHTAG]-(RU4708628:Tweet)
  return count(RU4708628) as total_relacionamentos
}
return h.tag, total_relacionamentos
order by total_relacionamentos desc
```

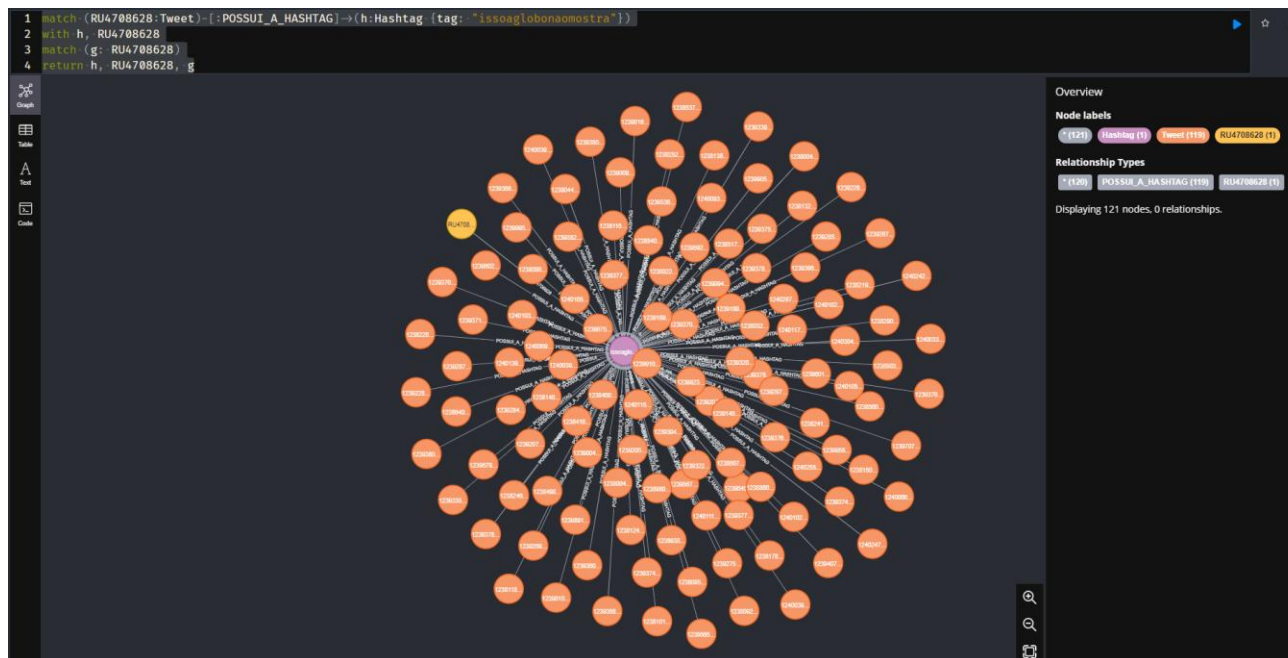
**Legenda:**

1. Seleciona as hashtags e atribui o identificador "h".
2. Abre uma subquery.
3. Dentro da subquery, seleciona a hashtag com o identificador "h".
4. Seleciona todos os tweets que possuem essa hashtag.
5. Retorna a contagem dos tweets selecionados que tem relacionamento com a hashtag selecionada, com o nome "total\_relacionamentos".
6. Fecha a subquery.
7. Retorna a tag da hashtag juntamente com o total de relacionamentos com tweets.

8. Ordena o resultado pela quantidade "total\_relacionamentos" em ordem decrescente.

Com esse código, é possível observar qual hashtag possui mais relacionamentos com os tweets e concluir que a hashtag "issoaglobonaomostra" foi a única que está relacionada a todos os tweets (119 tweets).

- II. Apresentação do grafo gerado contendo apenas 1 nó de hashtag ao centro (a sua resposta) e ao menos mais 10 nós de mensagens relacionadas a este nó de hashtag. O print deve conter o grafo sem zoom e a legenda de tipos de nós e cores geradas pelo neo4j, incluindo a informação da quantidade de nós de Hashtag mostrados (não esquecer do identificador pessoal, seu RU) **(Peso na nota: 12.5%)**:

**Código:**

```
match (RU4708628:Tweet)-[:POSSUI_A_HASHTAG]->(h:Hashtag {tag: "issoaglobonaomostra"})
with h, RU4708628
match (g: RU4708628)
return h, RU4708628, g
```

**Legenda:**

1. Seleciona os tweets com a hashtag "issoaglobonaomostra".
2. Passa os identificadores h e RU4708628 adiante.
3. Seleciona o nó RU4708628.
4. Retorna a Hashtag, os tweets que tem essa hashtag e o nó RU4708628.



Com esse código conseguimos observar todos os tweets com a hashtag “issoaglobonaomostra”(119 tweets), a própria hashtag “issoaglobonaomostra” no centro e o nó RU4708628 no canto superior esquerdo em amarelo.

Figura 1: **(INSERIR LEGENDA)**

- III. Responda à pergunta: Qual foi a hashtag usada como filtro para coleta dos dados analisados? (Esta hashtag só estará presente em nós do tipo Tweet e não em Retweet. Sua resposta deve conter apenas 1 palavra com o texto da hashtag) **(Peso na nota: 12.5%)?**

Resposta (apenas 1 palavra): issoaglobonaomostra

## Atividade Prática – NEO4J

### Questão 03 – ANÁLISE DOS DADOS SEGUNDO VIÉS A SUA ESCOLHA (Usar quantas páginas forem necessárias)

**ENUNCIADO:** Usando o mesmo banco de dados já criado na questão 01 e usado na questão 02, busque alguma informação que você julgue relevante nos dados (seu comando **não pode** ser **MATCH (n) RETURN n**; nem pode conter a biblioteca **APOC**). Sua tarefa aqui é analisar os dados do banco e definir qual informação você gostaria de obter e que tenha potencial de gerar um grafo com 10 ou mais nós interligados entre si. Sua análise deve responder à uma pergunta clara, como por exemplo:

- Qual o dispositivo mais usado para tuitar? (depende de seu banco de dados já possuir os nós de equipamentos usados, criados na questão 01).
  - Qual a Hashtag que menos foi usada?
- Qual o usuário mais movimentou a rede? (depende de você ter criado nós de usuário na questão 01)
  - Quais os usuários mais citados? (depende dos seus nós de mensagem possuírem esta informação ou de relacionamentos terem sido criados para este fim).

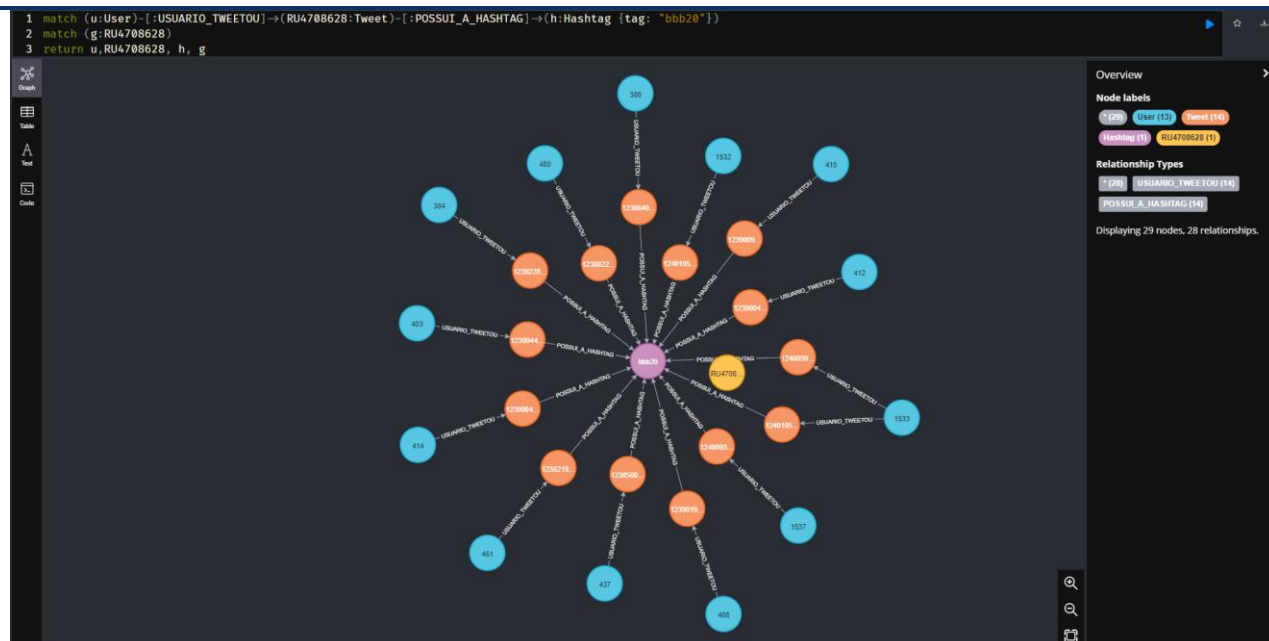
- I. Apresentação dos comandos (apenas queries Cypher) usados por você para realizar sua análise (não esquecer do identificador pessoal, seu RU) **(Peso na nota: 12.5%)**:

```
1 match (u:User)-[:USUARIO_TWEETOU]-(RU4708628:Tweet)-[:POSSUI_A_HASHTAG]->
   (h:Hashtag {tag: "bbb20"})
2 match (g:RU4708628)
3 return u,RU4708628, h, g
```

**Legenda:**

Se espera a exibição dos usuários que possuam relacionamentos com os tweets que estão fazendo outro relacionamento com a hashtag principal "bbb20".

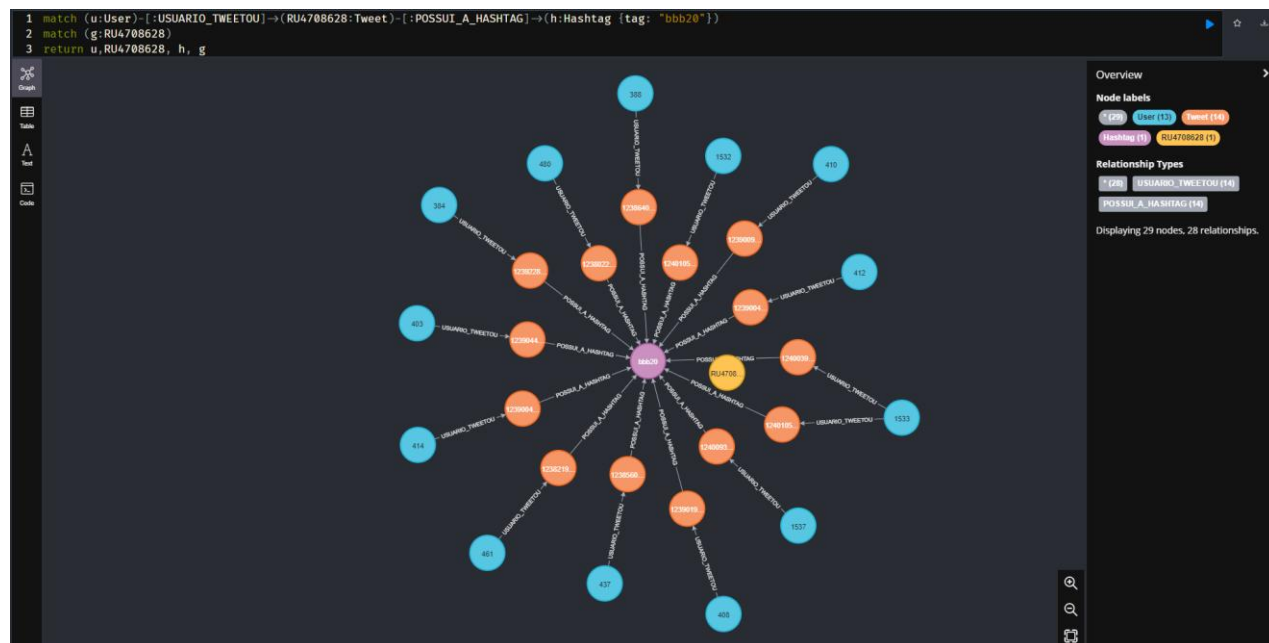
- II. Apresentação do print do resultado, podendo ser uma tabela ou um grafo. O print deve conter o resultado e o comando executado. (não esquecer do identificador, seu RU) **(Peso na nota: 12.5%)**:

**Legenda:**

Grafo contendo 29 nós no total, 13 Users, 14 Tweets, 1 Hashatg e 1 RU4708628.

**III. Explique qual foi a análise realizada, incluindo sua linha de raciocínio para criação da query Cypher e qual era sua expectativa de resultado antes da análise, comparando-a com o resultado realmente obtido após execução do comando (Peso na nota: 12.5%).**

**Resposta:** Ao analisar as hashtags, foi observado que muitas estavam relacionadas ao programa televisivo Big Brother Brasil. Retornando à pesquisa das hashtags mais utilizadas, foi identificado que a terceira mais frequente era "bbb20". Com isso, decidiu-se investigar quantas pessoas e quantos tweets estavam relacionados a essa hashtag.

**Código:**

```
match (u:User)-[:USUARIO_TWEETOU]->(RU4708628:Tweet)-[:POSSUI_A_HASHTAG]->(h:Hashtag {tag: "bbb20"})
```



```
match (g:RU4708628)
return u, RU4708628, h, g
```

**Legenda:**

1. Seleciona os usuários que tweetaram utilizando a hashtag "bbb20", criando identificadores para os usuários, os tweets e a hashtag.
2. Seleciona o nó "RU4708628".
3. Retorna os usuários, os tweets, a hashtag e o nó RU4708628 utilizando seus respectivos identificadores.

Ao realizar essa pesquisa, foi notado que, mesmo quando um assunto está em alta com várias hashtags associadas, nem sempre haverá uma quantidade significativa de tweets utilizando exatamente a mesma hashtag. Para que muitas pessoas usem a mesma hashtag, é necessário que haja algum fator muito forte para engajar e incentivar o público a usa-la.