Projeto: Relacionando Covid e clima

Isabela Caroline de Sousa 218071
Lucas Antevere Santana 201775
Matheus Bulhões Barbosa 222157

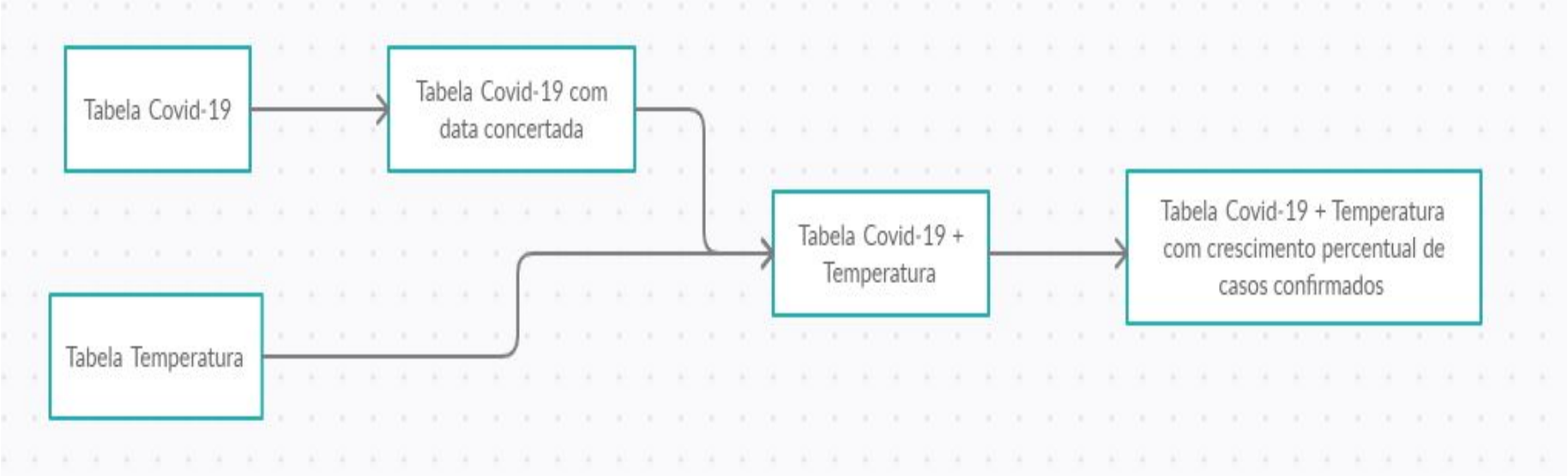
Etapa 3

Nessa etapa, inicialmente, elaboramos o modelo conceitual e em seguida fizemos uma primeira análise dos dados usando o modelo relacional por meio do SQL.

Modelo Conceitual

Nota: o objetivo do projeto é verificar se a relação "influencia" de fato existe.

Representação da Análise



Análise

1. Importamos as tabelas Covid-19 e Temperatura
2. Selecionamos um intervalo de datas a partir da tabela Covid-19
3. Para que houvesse compatibilidade entre os formatos de datas das dua tabelas, alteramos a coluna “Data” da tabela Covid-19 para um formato padrão de data
4. Fizemos a junção das duas tabelas a partir da data
5. Calculamos o crescimento percentual de casos confirmados

Seleção de uma data na tabela Covid-19

```
SELECT * FROM Covid
WHERE Data LIKE '%-03-%' AND RIGHT(LEFT(Data,10),2) < 21 AND RIGHT(LEFT(Data,10),2) > 09
```

Como visto na sentença SQL acima, selecionar um intervalo de datas na tabela Covid-19 original era um tanto trabalhoso. Essa dificuldade motivou a alteração da tabela para um formato de data convencional.

Alteração da coluna “Data” na tabela Covid-19

```
DROP TABLE IF EXISTS Covid_Data;

CREATE TABLE Covid_Data (
  Confirmed INTEGER,
  Deaths INTEGER,
  Recovered INTEGER,
  Active INTEGER,
  Data DATE,
  PRIMARY KEY(Data)
) AS SELECT
    Confirmed,
    Deaths,
    Recovered,
    Active,
    LEFT(Data,10)
FROM Covid;

SELECT * FROM Covid_Data
```

Crescimento Percentual de casos confirmados

```sql
DROP VIEW IF EXISTS Covid_Percent;

CREATE VIEW Covid_Percent AS
SELECT
C.Confirmed as Cases,
COALESCE((C.Confirmed - CL.Confirmed)/ NULLIF(CAST(CL.Confirmed AS FLOAT), 0), 0) + 0.0 AS CasesVariation,
C.Deaths,
COALESCE((C.Deaths - CL.Deaths)/ NULLIF(CAST(CL.Deaths AS FLOAT), 0), 0) + 0.0 AS DeathsVariation,
C.Recovered,
T.Maximo,
T.Minimo,
T.Media,
T.Data
FROM Covid_Data AS C, Covid_Data AS CL, Temperatura AS T
WHERE CL.Data = DATEADD(day, -1, C.Data) AND C.Data = T.Data
ORDER BY T.Data;

SELECT * FROM Covid_Percent
```

Etapa 4

- Para a realização dessa etapa utilizamos o modelo de grafos
- Nosso objetivo: realizar alguma análise que possibilitasse agrupar os países por semelhanças climáticas e, assim, selecionar quais os países cujos dados relacionados à covid-19 seriam usados no decorrer do nosso projeto
- Estabelecer relações de similaridade entre os países e agrupá-los é uma tarefa mais facilmente executável utilizando o modelo de grafos do que o modelo relacional
- A característica visual que os grafos possuem facilitou a escolha dos países depois de agrupados

Etapa 4

Nessa etapa, inicialmente, fizemos um programa em python para unir duas APIs, uma vez que não conseguimos encontrar uma API que contivesse todos os dados necessários. Com a tabela que criamos, usando o NEO4J, verificamos os países com características climáticas próximas (temperatura, pressão e umidade) e tentamos separá-los em grupos de acordo com similaridade.

Código em Python

```
import io
import requests
import json
url = "https://api.ipgeolocationapi.com/countries"
data = requests.get(url)
js = data.json()
countries = []

for i in js:
    country = [i, str(js[i]['geo']['latitude'])[0:6], str(js[i]['geo']['longitude'])[0:6]]
    while (len(country[1]) < 6):
        country[1] = country[1]+'0'
    while (len(country[2]) < 6):
        country[2] = country[2]+'0'
    countries.append(country)

url2 = "http://api.openweathermap.org/data/2.5/weather?lat=000000&lon=111111&appid=9de243494c0b295cca9337e1e96b00e2"
for i in countries:
    old = url2[51:57]
    url2 = url2.replace(old, i[1], 1)
    old = url2[62:68]
    url2 = url2.replace(old, i[2], 1)
    data = requests.get(url2)
    js = data.json()
    print(i[0]+','+str(js['main']['temp'])+','+str(js['main']['pressure'])+','+str(js['main']['humidity']))
```

Análise no Neo4j

1. Importamos a tabela criada anteriormente e criamos os nós
2. Conectamos os países com características similares em termos de temperatura, pressão e umidade
3. Consideramos que os países que possuem os 3 tipos de relações de similaridade possuem uma similaridade climática
4. Deletamos as outras relações de similaridade para que os grafos não fiquem poluídos visualmente
5. Determinamos os países com os quais um determinado país é semelhante
6. Visualizamos graficamente os grupos de países similares entre si

Queries no Neo4j

- Para temperatura, pressão e umidade, conectamos os países que têm características próximas

```
MATCH (c1:Country)
MATCH (c2:Country)
WHERE ABS(toFloat(c1.temp) - toFloat(c2.temp)) < 4 AND c1 <> c2
CREATE (c1)-[:similar_temperature]->(c2)

MATCH (c1:Country)
MATCH (c2:Country)
WHERE ABS(toFloat(c1.pressure) - toFloat(c2.pressure)) < 6 AND c1 <> c2
CREATE (c1)-[:similar_pressure]->(c2)

MATCH (c1:Country)
MATCH (c2:Country)
WHERE ABS(toFloat(c1.humidity) - toFloat(c2.humidity)) < 10 AND c1 <> c2
CREATE (c1)-[:similar_humidity]->(c2)
```

- Consideramos que os países que têm os 3 tipos de similaridades possuem uma similaridade climática, no geral

```
MATCH p=(c1)-[:similar_humidity]->(c2)<-[:similar_pressure]-(c1)-[:similar_temperature]->(c2)
CREATE (c1)-[:similar]->(c2)
```

Queries no Neo4j

- E deletaremos as outras relações de similaridade para que os grafos não fiquem visualmente poluídos

```
MATCH p=()-[r:similar_temperature]->()
DELETE r

MATCH p=()-[r:similar_humidity]->()
DELETE r

MATCH p=()-[r:similar_pressure]->()
DELETE r
```

- É possível agora, utilizar o algoritmo de comunidade para relacionar vários países que são similares entre si

```
CALL gds.graph.create('similarCountries','Country',{similar: {orientation:'UNDIRECTED'}})

CALL gds.louvain.stream('similarCountries')
YIELD nodeId, communityId
MATCH (c:Country {nome: gds.util.asNode(nodeId).nome})
SET c.community = communityId
```

- Também é possível manualmente contar os países com o qual um determinado país é semelhante

```
MATCH (c)-[:similar]->(c1)
MERGE (c)-[r:count]->()
ON CREATE SET r.similarity=1
ON MATCH SET r.similarity=r.similarity+1
```

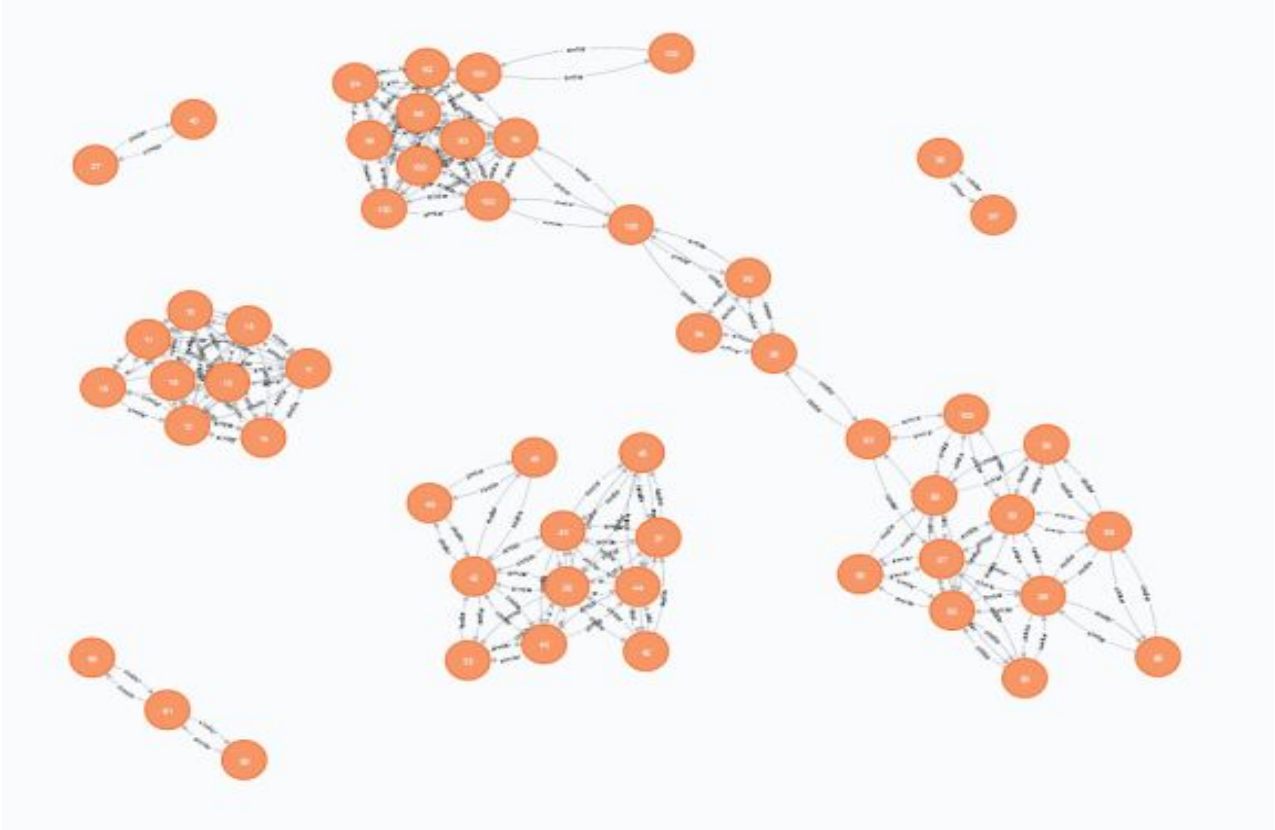
Queries no Neo4j

- E, assim, visualizar graficamente os grupos de países similares entre si

```
MATCH ()<-[r:count]-(c1)-[s:similar]->(c2)
WHERE r.similarity < 10
RETURN c1,c2
LIMIT 50

MATCH ()<-[r:count]-(c1)-[s:similar]->(c2)
WHERE r.similarity > 25
RETURN c1,c2
LIMIT 30
```

Grafo contendo grupos de Países



Conclusão

Para os cinco grupos expressivos que foram encontrados, como mostrado anteriormente, escolhemos os seguintes países:

- Malásia, Brunei, Tailândia
- Estados Unidos, Holanda, Coreia do Sul
- Egito, Emirados Árabes
- Uruguai, Brasil, Costa do Marfim
- Senegal, Sudão