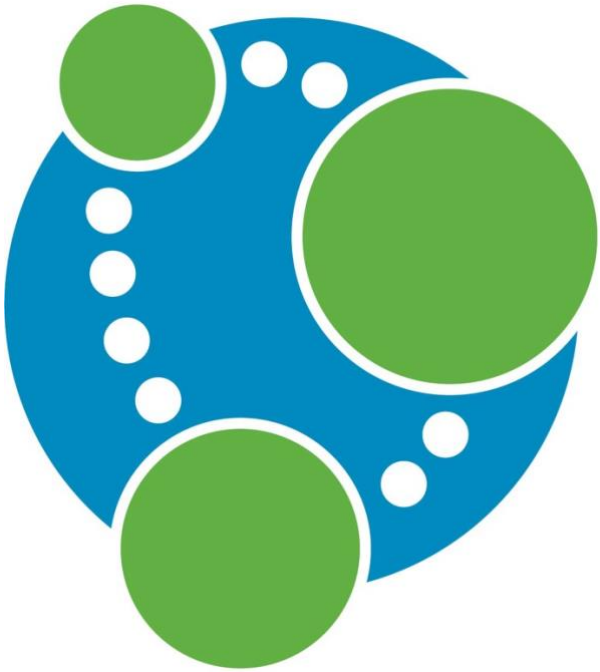


MESIIN480323 : No SQL 13/03

PROJECT REPORT

On

Neo4j

reuters.json

GABRIEL CHAIX & MALO LEROY & CHARLES TERRIER

ÉCOLE SUPERIEURE D'INGENIEURS LEONARD DE VINCI

ANNEE 2023 / 2024

INDEX

INTRODUCTION AND PROBLEM STATEMENT

Reuters is a news agency founded in 1851 in London. It is one of the global and generalist news agencies, an activity that represents a part of its revenue, mainly devoted to financial information. It is one of the largest news agencies in the world. We want to create a MongoDB database with reuters articles and query it.

I. Data Importation

In a first manner, we'll setup the docker container with neo4j image and make sure everything is working properly, we try to connect on port 7687.

Now we must create our graph using python and the py2neo package. To do this, we run the following script to create and populate the graph schema and. After downloading our [json file] put it in the same directory as our code.

```python
import json
from py2neo import Graph, Node, Relationship

# Initialize the graph connection
graph = Graph("bolt://localhost:7687", auth=('neo4j', 'neo4jneo4j'))

# Load the JSON data from the file
with open('reuters.json') as f:
    data = json.load(f)

# Clear the graph
graph.delete_all()

# Iterate over each item in the loaded JSON data
for idx, doc in enumerate(data):
    _id = doc['_id'] if '_id' in doc else ''
    date = doc['date'] if 'date' in doc else ''
    topics = set(doc['topics'].split()) if 'topics' in doc else set()
    places = set(doc['places'].split()) if 'places' in doc else set()
    people = set(doc['people'].split()) if 'people' in doc else set()
    orgs = set(doc['orgs'].split()) if 'orgs' in doc else set()
    exchanges = set(doc['exchanges'].split()) if 'exchanges' in doc else set()
    companies = set(doc['companies'].split()) if 'companies' in doc else set()

    text = doc['text'] if 'text' in doc else {}
    title = text['title'] if 'title' in text else ''
    dateline = text['dateline'] if 'dateline' in text else ''
    body = text['body'] if 'body' in text else ''

    # Create a new Node representing the document
    doc_node = Node("article", id=_id, date=date, topics=list(topics),
places=list(places), people=list(people),
                    orgs=list(orgs), exchanges=list(exchanges),
companies=list(companies))

    # Create a new Node representing the text
    text_node = Node("text", title=title, dateline=dateline, body=body)

    # Create a relationship between the document and the text
    relationship = Relationship(doc_node, "has-text", text_node)

    # Save the newly created Node and Relationship
    graph.create(relationship)
    graph.create(doc_node)
    print(f"Document {idx} added to the graph")

print("Nodes & relationships added successfully!")
```

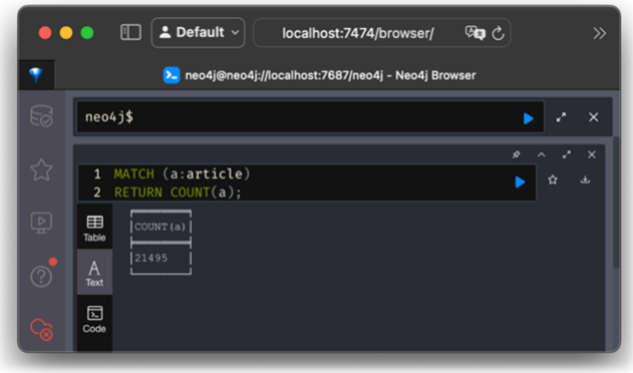
Then here is the formatting script we've wrote

←

After importing all the rows, we make sure that each row is correctly imported.

After importing all the rows, we make sure that we have the count and that each row is correct.

Which we have, using the Neo4j browser, we can see everything looks alright.



The queryable properties are:

Nodes:

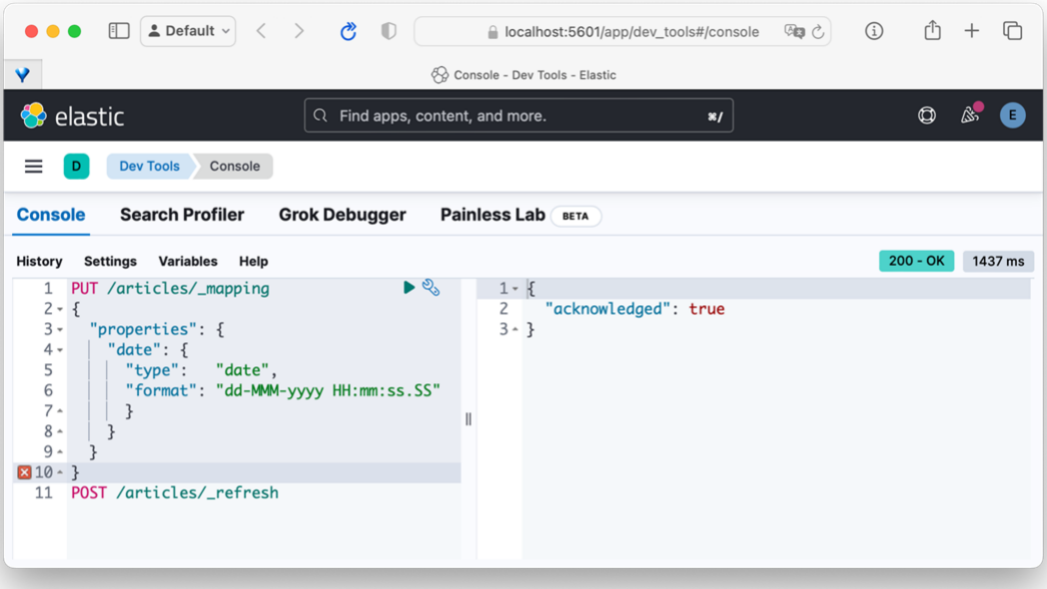
- Article nodes with labels **article** and properties **id**, **date**, **topics**, **places**, **people**, **orgs**, **exchanges**, **companies**
- Text nodes with labels **text** and properties **title**, **dateline**, **body**

Relationships:

- A **has-text** relationship connecting **article** nodes to related **text** nodes.

II. Data cleaning

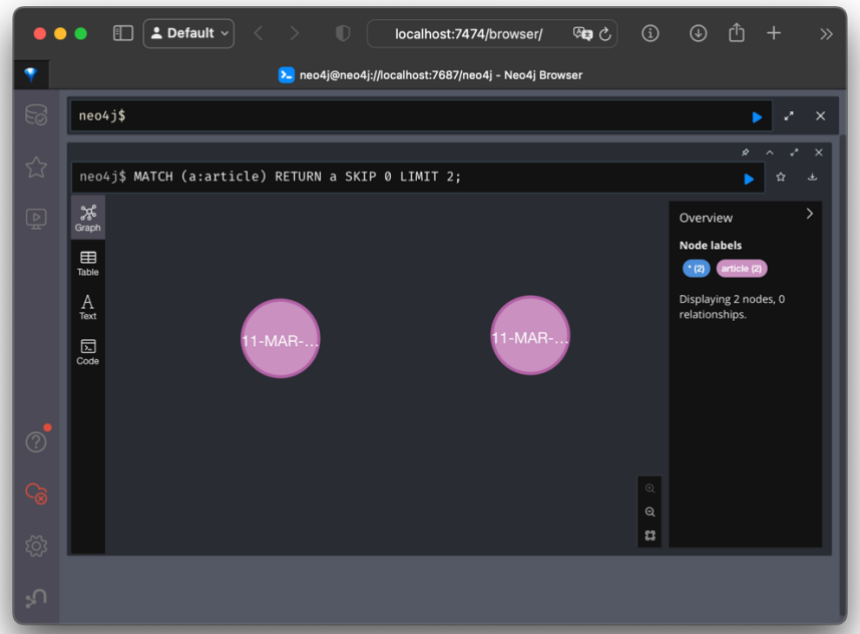
The only cleaning, we success to do had been to update all date entry to the valid date type. Compared to other noSQL language, we found hard to word with elastic search.



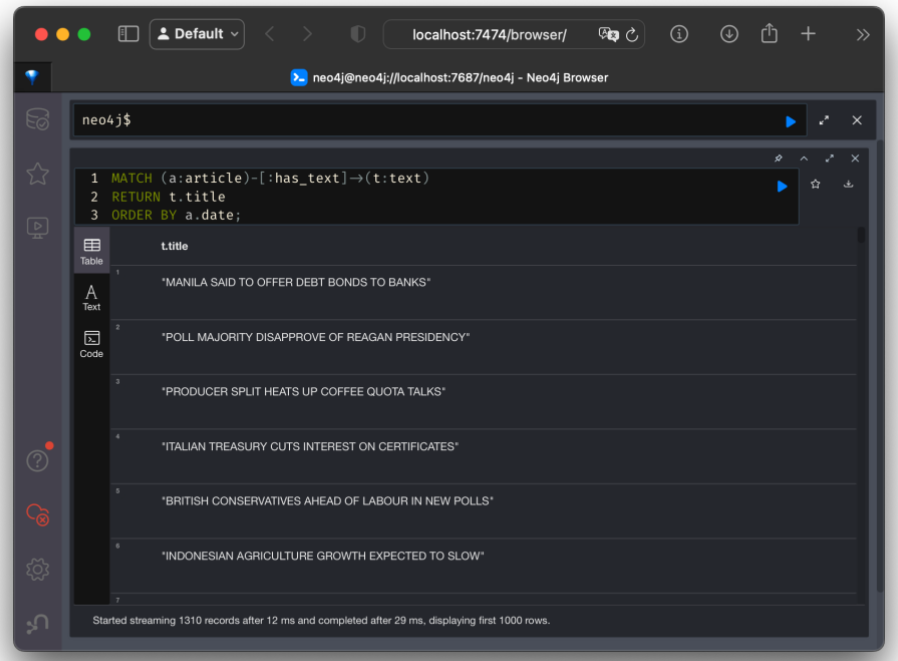
III. Querys

A. Simple querys (6)

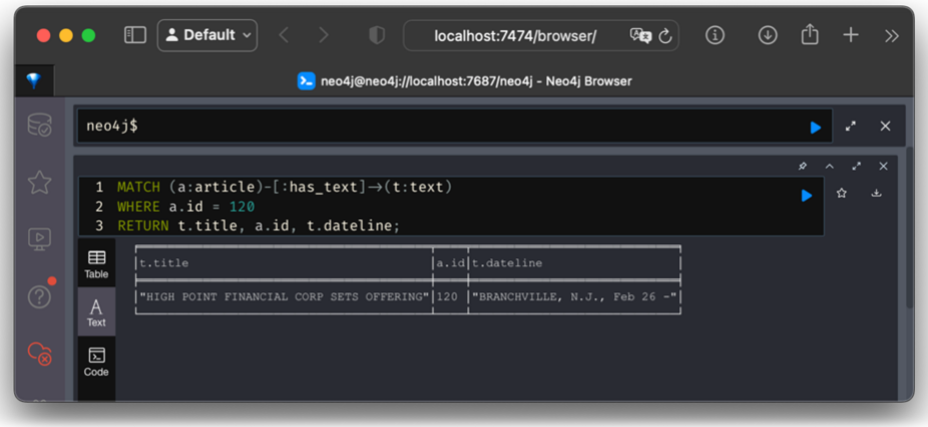
1. Graph of the two first articles



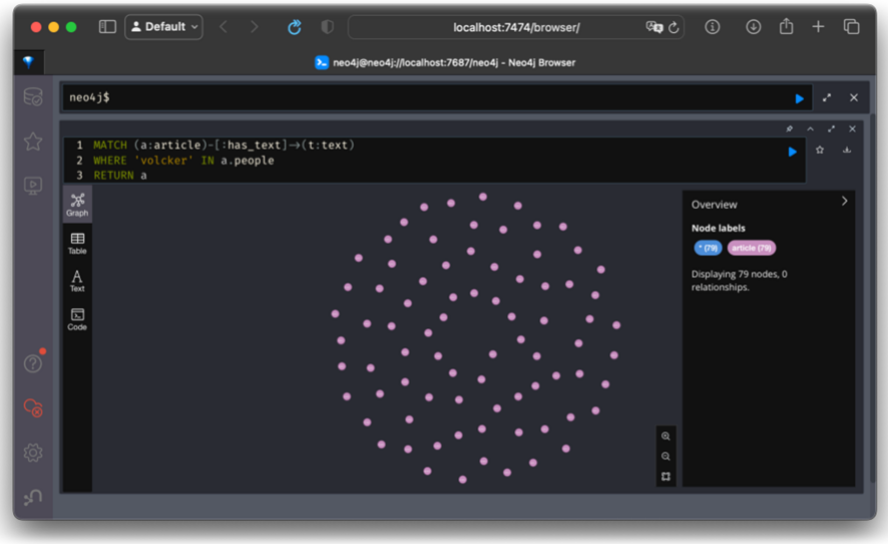
2. List of text titles sorted by date



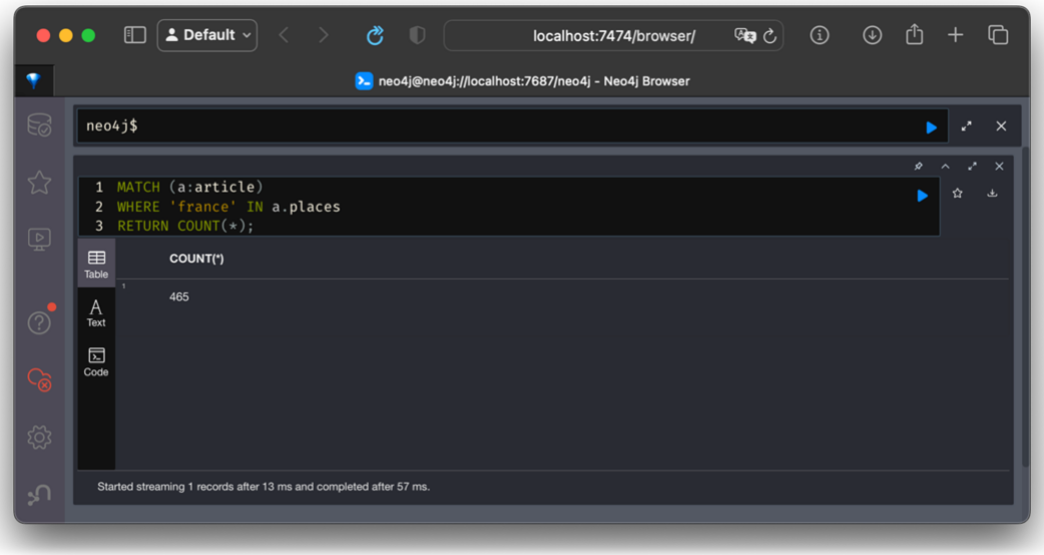
3. Titles, id and dateline of texts where id is 120



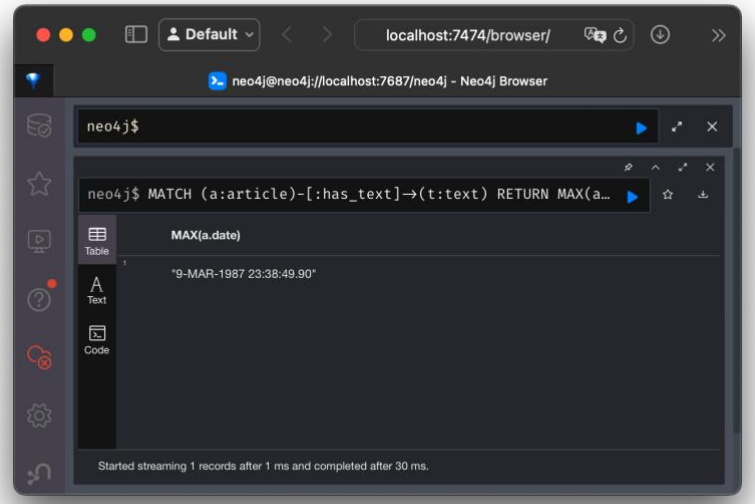
4. Articles written by volcker



5. Count of articles whose place is France

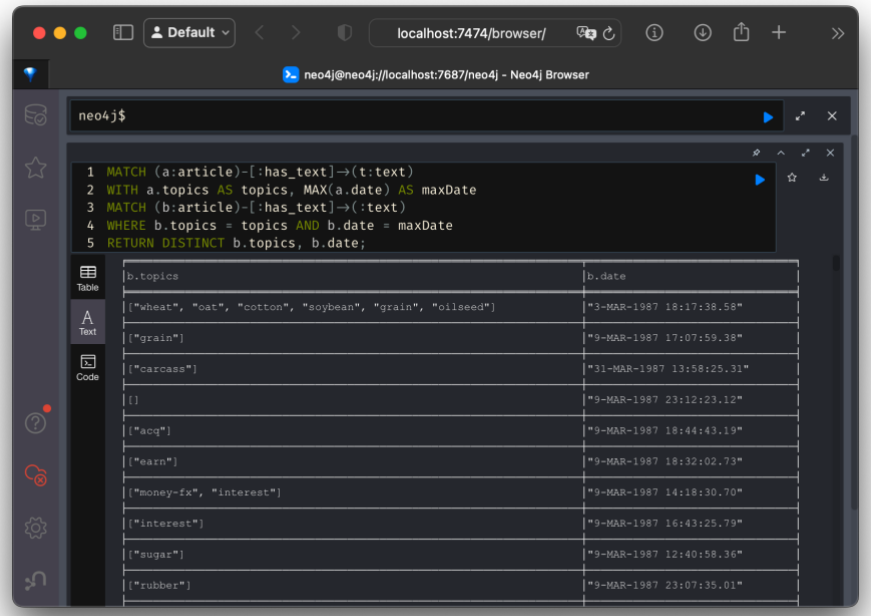


6. Find the earliest publication date among all articles

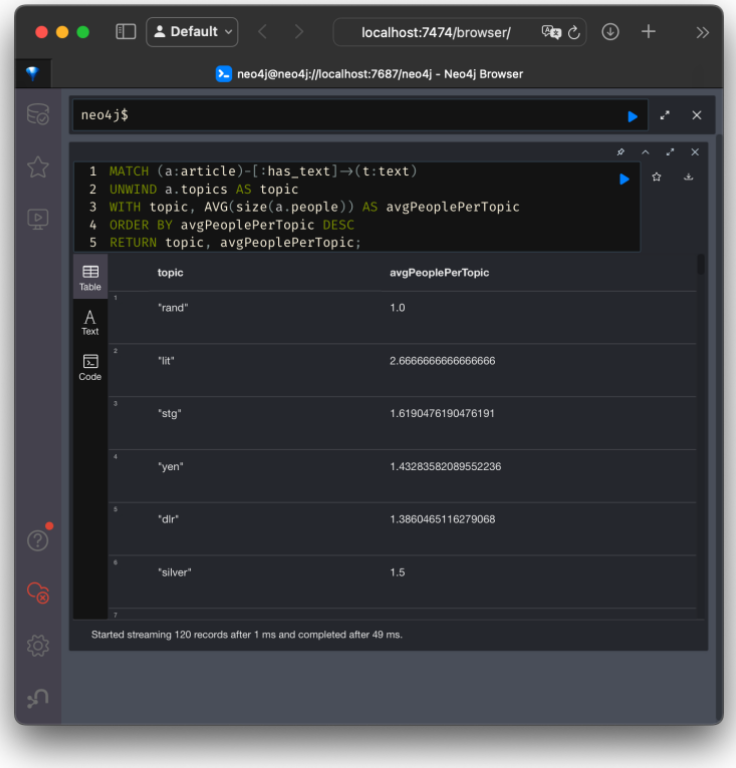


B. Complex querys (2)

1. Return the latest article date for each distinct topic:



2. For each topic, give the average number of people that wrote on.


Default
localhost:7474/browser/
neo4j@neo4j://localhost:7687/neo4j - Neo4j Browser
neo4j$
1 MATCH (a:article)-[:has_text]→(t:text)
2 UNWIND a.topics AS topic
3 WITH topic, AVG(size(a.people)) AS avgPeoplePerTopic
4 ORDER BY avgPeoplePerTopic DESC
5 RETURN topic, avgPeoplePerTopic;
Table
Text
Code
topic
avgPeoplePerTopic
1 "rand" 1.0
2 "lit" 2.6666666666666666
3 "stg" 1.6190476190476191
4 "yen" 1.43283582089552236
5 "dlr" 1.3860465116279068
6 "silver" 1.5
7
Started streaming 120 records after 1 ms and completed after 49 ms.

C. Hard query (1)

1. How much articles have been published per month?