MESIIN480323 : No SQL 19/01

PROJECT REPORT

On

CASSANDRA

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. The objective is to create a Cassandra database to store Reuters articles and perform various queries for data analysis.

I. Data Model

We utilized TablePlus to execute CQL queries for setting up our reuters keyspace in Cassandra. This keyspace comprises two principal tables: reuters for article metadata and texts for the articles' textual content.

To efficiently retrieve articles by ID, we designated id as the primary key for both tables. This choice simplifies data access patterns and maintains high performance for writes and reads, aligning with Cassandra's design principles.

```
1 create keyspace reuters if not exists;
2
3 create table if not exists reuters (id int primary key, companies
  text, date text, exchanges text, orgs text, people text, places text,
  text_id int, topics text);
4
5 create table if not exists texts (id int primary key, body text,
  dateline text, title text);
```

Table creation - Figure 1.1

We formatted the dataset as a valid JSON array for Cassandra's importation tools by enclosing it in brackets and adding commas, ensuring compatibility and ease of data ingestion. Our model's simplicity avoids secondary indexes to favor write efficiency, in line with Cassandra's strengths in handling large-scale data.

II. Data Importation

```
# Connection to reuters keyspace
cluster = Cluster()
session = cluster.connect('reuters')

session.execute("TRUNCATE TABLE reuters")
session.execute("TRUNCATE TABLE texts")
json_file = 'Cassandra/reuters.json'

with open(json_file) as data_file:
    data = json.load(data_file)

    for v in data:
        try:
            r_id = int(v['_id'])
            print('Uploading row ' + str(r_id))
            r_companies = "'" + v.get('companies', '') + "'"
            r_date = "'" + v.get('date', '') + "'"
            r_exchanges = "'" + v.get('exchanges', '') + "'"
            r_orgs = "'" + v.get('orgs', '') + "'"
            r_people = "'" + v.get('people', '') + "'"
            r_places = "'" + v.get('places', '') + "'"
            r_text_id = r_id

            t_id = r_id
            t_body = "'" + v['text'].get('body', '') + "'"  # Check if 'body' exists
            t_dateline = "'" + v['text'].get('dateline', '') + "'"
            t_title = "'" + v['text'].get('title', '') + "'"

            r_topics = "'" + v.get('topics', '') + "'"

            # Create and execute prepared statements for insertion into reuters table
            query = "INSERT INTO reuters (id, companies, date, exchanges, orgs, people,
places, text_id, topics) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"
            prepared_stmt = session.prepare(query)
            session.execute(prepared_stmt, [r_id, r_companies, r_date, r_exchanges, r_orgs,
r_people, r_places, r_text_id, r_topics])

            # Create and execute prepared statements for insertion into texts table
            query = "INSERT INTO texts (id, body, dateline, title) VALUES (?, ?, ?, ?)"
            prepared_stmt = session.prepare(query)
            session.execute(prepared_stmt, [t_id, t_body, t_dateline, t_title])

        except Exception as e:
            print("Une exception s'est produite : ", e)
```

Then here is the importation script we've written.

←

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

III. Data cleaning

The first step we've take into cleaning had been to update all the empty entries to NULL. To address this problem, we wrote a Python script.

```
from cassandra.cluster import Cluster

cluster = Cluster()
session = cluster.connect('reuters')

# Define the columns for the 'reuters' table
reuters_columns = ['companies', 'date', 'exchanges', 'orgs', 'people', 'places',
'topics']

# Define the columns for the 'text' table
text_columns = ['body', 'dateline', 'title']


# Function to update rows with empty strings to null
def clean_table(table_name, columns):
    rows = session.execute(f'SELECT id, {", ".join(columns)} FROM {table_name}')

    for row in rows:
        updates = []
        for column in columns:
            # Check for empty string
            if getattr(row, column) == "''":
                print(f"Row with id {row.id} has empty string in {column}")
                updates.append(f"{column} = null")

        if updates:
            update_query = f"UPDATE {table_name} SET {', '.join(updates)} WHERE id
= {row.id}"
            session.execute(update_query)
            print(f"Updated row with id {row.id}")

clean_table('reuters', reuters_columns)
clean_table('texts', text_columns)

cluster.shutdown()
```

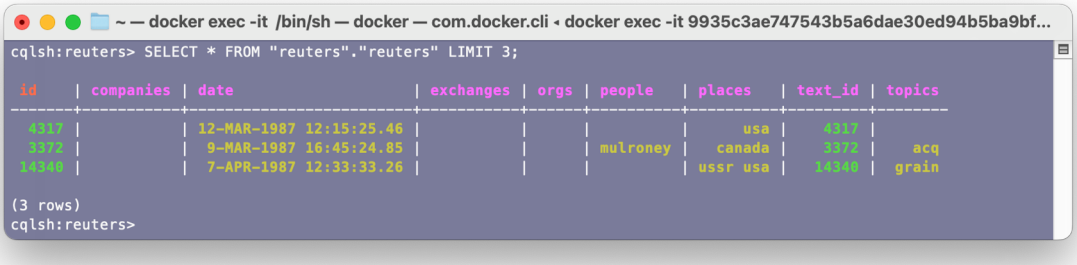
We also observe that every text attribute had quotes, due to the importation script, since we didn't succeed in modifying the importation script, we added a similar Python function to take them off.

body	dateline	title
'Lotus Development Corp said ithas signed a…	'CAMBRIDGE, Mass., March 4 -'	'LOTUS COMPUTER ACCESS CORP'

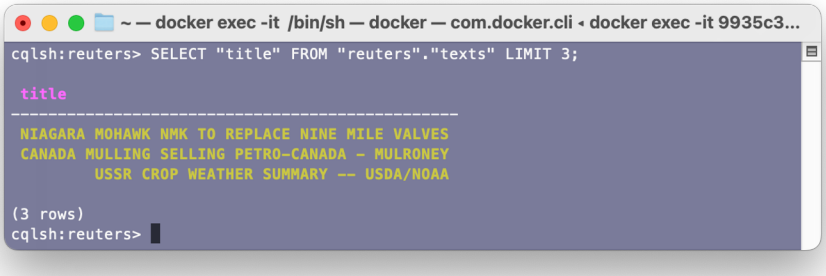
IV. Queries

A. Simple queries (6)

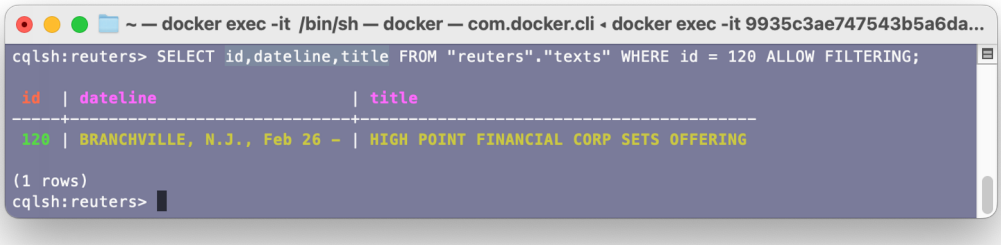
1. List of Reuters



2. List of text titles



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



4. Count of Reuters whose place is France.

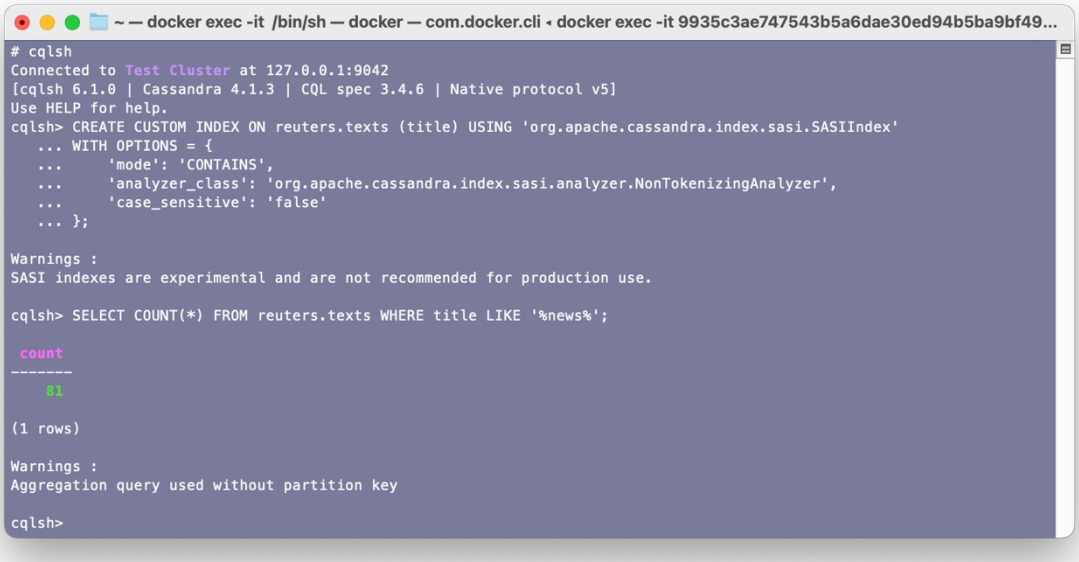
5. Count of “texts” publications,

6. Count of texts WHERE title contains “News”

To use the LIKE statement, we need to create a SASI index, let’s enable it in cassandra.yaml file.

```
# cd etc/cassandra
# ls
cassandra-env.sh  cassandra.yaml …
# apt update
# apt install vim
# vim cassandra.yaml
enable_sasi_indexes: false  → true
restart Cassandra
```

Then we can create our custom index and use LIKE statement on the desire column.

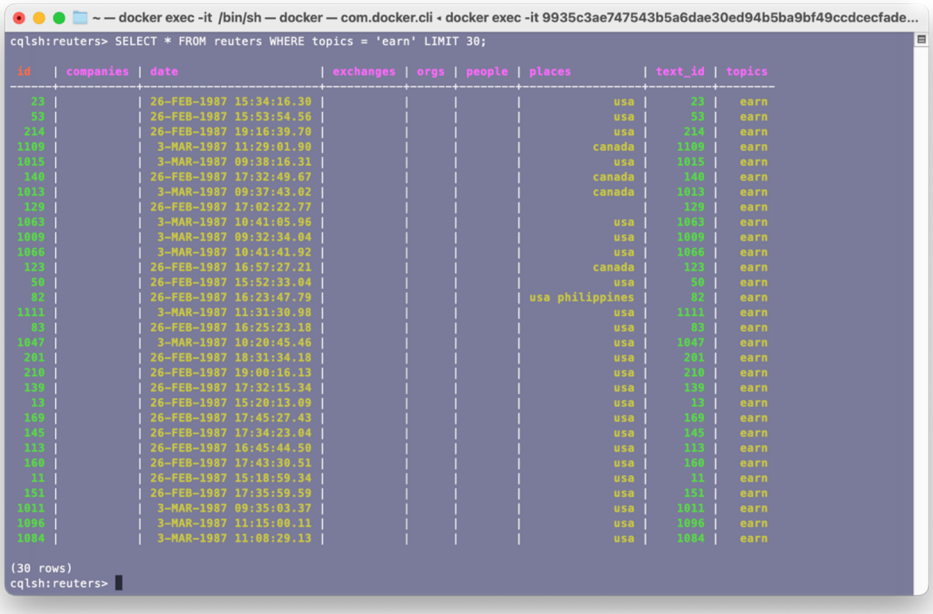


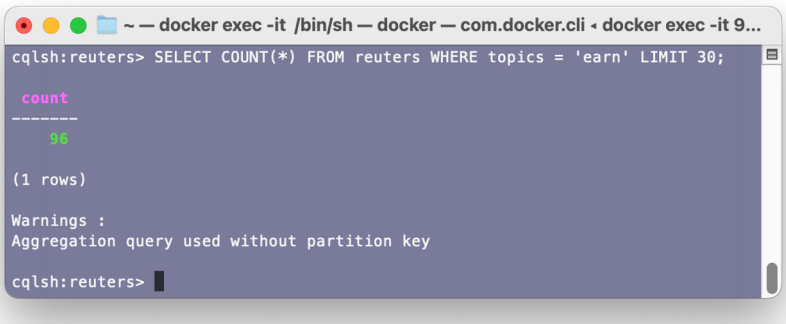
B. Complex querys (2)

1. Count the number of articles having the topic ‘earn’.

To use the WHERE =’’ statement, we had to create a btree indexation on the topic column of Reuters table.

```
cqlsh> USE reuters;
cqlsh:reuters> CREATE INDEX btree_reuters_topics ON reuters(topics);
```



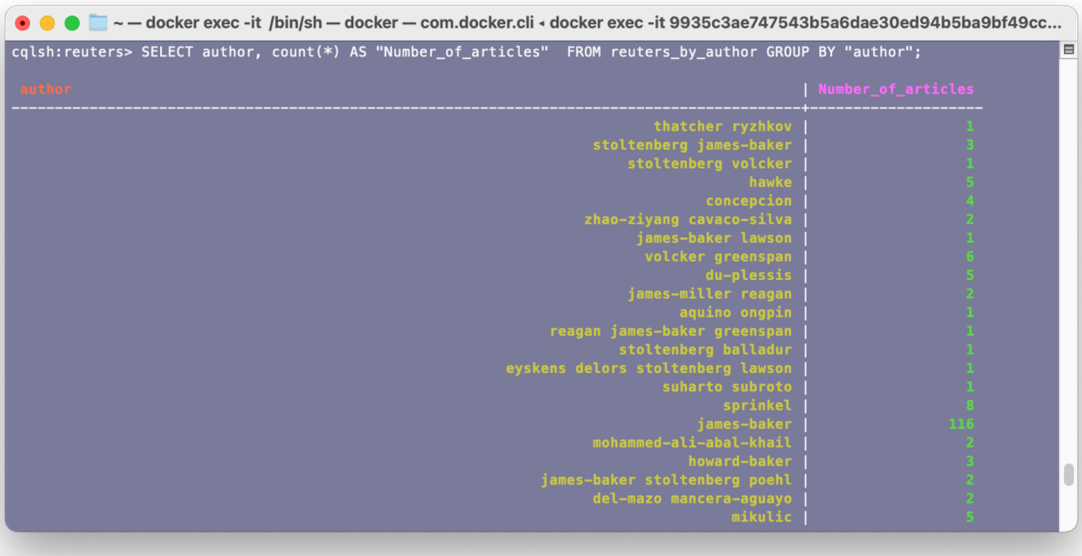


2. Give the number articles written by each author ?

To achieve this query, we need to make a join between Reuters and text tables. To do this in CQL, we unormalize the Reuters table by adding to it an attribute for author name. We create a new table and fill and clean it using our Python script :

```
CREATE TABLE reuters_by_author (
    author text,
    id int,
    title text,
    companies text,
    date text,
    exchanges text,
    orgs text,
    people text,
    places text,
    text_id int,
    topics text,
    PRIMARY KEY (author, id)
);
```

Now that author is part of the primary key, we can group by author.



C.Hard query (1)

1. How many articles are there by place

```
# cd etc/cassandra
# ls
cassandra-env.sh  cassandra.yaml …
# apt update
# apt install vim
# vim cassandra.yaml
user_defined_functions_enabled: false  → true
restart Cassandra
```

```
##"Define the User-Defined Function (UDF)"

CREATE FUNCTION reuters.countState(state map<text, bigint>, place text)
CALLED ON NULL INPUT
RETURNS map<text, bigint>
LANGUAGE java AS
$$
    if (place != null) {
        state.put(place, state.getOrDefault(place, 0L) + 1);
    }
    return state;
$$;

##"Define the User-Defined Aggregate (UDA)"

CREATE AGGREGATE reuters.countGroup(text)
SFUNC countState
STYPE map<text, bigint>
INITCOND {};
```