



cassandra

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

<i>INTRODUCTION AND PROBLEM STATEMENT</i>	3
<i>I. Data Model</i>	3
<i>II. Data Importation</i>	4
<i>III. Data cleaning.....</i>	5
<i>IV. Queries.....</i>	6
A. Simple queries (6).....	6
1. List of Reuters	6
2. List of text titles	6
3. Titles, id and dateline of texts where id is "120"	6
4. Count of Reuters whose place is France.	7
5. Count of "texts" publications,	7
6. Count of texts WHERE title contains "News"	7
B. Complex querys (2).....	8
1. Count the number of articles having the topic 'earn'.	8
2. Give the number articles written by each author ?.....	9
C. Hard query (1).....	9
1. How many articles are there by place	9

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 it has signed a...	'CAMBRIDGE, Mass., March 4 -'	'LOTUS COMPUTER ACCESS CORP'

IV. Queries

A. Simple queries (6)

1. List of Reuters

```
cqlsh:reuters> SELECT * FROM "reuters"."reuters" LIMIT 3;
      id | companies | date           | exchanges | orgs | people | places | text_id | topics
-----+-----+-----+-----+-----+-----+-----+-----+-----+
    4317 |          | 12-MAR-1987 12:15:25.46 |          |      |          | usa    | 4317   | acq
    1371 |          | 9-MAR-1987 16:45:24.85 |          |      |          | canada | 1371   |
   14340 |          | 7-APR-1987 12:33:33.26 |          |      |          | ussr   | 14340  | grain
(3 rows)
cqlsh:reuters>
```

2. List of text titles

```
cqlsh:reuters> SELECT "title" FROM "reuters"."texts" LIMIT 3;
      title
-----
 NIAGARA MOHAWK NMK TO REPLACE NINE MILE VALVES
 CANADA MULLING SELLING PETRO-CANADA - MULRONEY
 USSR CROP WEATHER SUMMARY -- USDA/NOAA
(3 rows)
cqlsh:reuters>
```

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

```
cqlsh:reuters> SELECT id,dateline,title FROM "reuters"."texts" WHERE id = 120 ALLOW FILTERING;
      id | dateline           | title
-----+-----+-----+
    120 | BRANCHVILLE, N.J., Feb 26 - | HIGH POINT FINANCIAL CORP SETS OFFERING
(1 rows)
cqlsh:reuters>
```

4. Count of Reuters whose place is France.

```
① ○ ● 📂 ~ — docker exec -it /bin/sh — docker — com.docker.cli • docker exec -it 9935c3ae747543b5a6dae30ed94b5ba9...  
cqlsh> SELECT COUNT(*) AS "Count of french article" FROM reuters.reuters WHERE places = 'france' ALLOW FILTERING;  
  
Count of french article  
-----  
    372  
  
(1 rows)  
  
Warnings :  
Aggregation query used without partition key  
  
cqlsh> █
```

5. Count of “texts” publications,

```
○ ● ○ ~ - docker exec -it /bin/sh - docker -- com.docker.cli > dock...  
cqsh:reuters> SELECT COUNT(*) FROM reuters.texts;  
  
count  
---  
31495  
  
(1 rows)  
  
Warnings :  
Aggregation query used without partition key  
  
cqsh:reuters>
```

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.

```
# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.1.0 | Cassandra 4.1.3 | CQL spec 3.4.6 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE CUSTOM INDEX ON reuters.texts (title) USING 'org.apache.cassandra.index.sasi.SASIIndex'
... WITH OPTIONS = {
...   'mode': 'CONTAINS',
...   'analyzer_class': 'org.apache.cassandra.index.sasi.analyzer.NonTokenizingAnalyzer',
...   'case_sensitive': 'false'
... };

Warnings :
SASI indexes are experimental and are not recommended for production use.

cqlsh> SELECT COUNT(*) FROM reuters.texts WHERE title LIKE '%news%';

  count
-----
  81

(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh>
```

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);
```

```
cqlsh:reuters> SELECT * FROM reuters WHERE topics = 'earn' LIMIT 30;
      id | companies | date           | exchanges | orgs | people | places | text_id | topics
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
     23 |          | 26-FEB-1987 15:34:16.30 |          | usa  | 23  |          | usa    | 23   | earn
     53 |          | 26-FEB-1987 15:53:54.56 |          | usa  | 53  |          | usa    | 53   | earn
    214 |          | 26-FEB-1987 19:16:39.70 |          | usa  | 214 |          | usa    | 214  | earn
1109 |          | 3-MAR-1987 11:29:01.90 |          | canada | 1109 |          | canada | 1109 | earn
1415 |          | 3-MAR-1987 09:38:16.31 |          | usa  | 1415 |          | usa    | 1415 | earn
148 |          | 26-FEB-1987 17:32:49.67 |          | canada | 148  |          | canada | 148  | earn
1613 |          | 3-MAR-1987 09:37:42.02 |          | canada | 1613 |          | canada | 1613 | earn
129 |          | 26-FEB-1987 17:02:22.77 |          |          | 129  |          |          | 129  | earn
1863 |          | 3-MAR-1987 10:41:05.96 |          | usa  | 1863 |          | usa    | 1863 | earn
1869 |          | 3-MAR-1987 09:32:34.84 |          | usa  | 1869 |          | usa    | 1869 | earn
1866 |          | 3-MAR-1987 10:41:41.92 |          | usa  | 1866 |          | usa    | 1866 | earn
113 |          | 26-FEB-1987 16:57:27.21 |          |          | 113  |          |          | 113  | earn
58 |          | 26-FEB-1987 15:52:33.04 |          | usa  | 58  |          | usa    | 58   | earn
82 |          | 26-FEB-1987 16:23:47.79 |          |          | 82   |          |          | 82   | earn
1111 |          | 3-MAR-1987 11:31:30.98 |          |          | 1111 |          |          | 1111 | earn
83 |          | 26-FEB-1987 16:25:23.18 |          |          | 83   |          |          | 83   | earn
1647 |          | 3-MAR-1987 10:20:45.46 |          |          | 1647 |          |          | 1647 | earn
361 |          | 26-FEB-1987 18:31:34.18 |          |          | 361  |          |          | 361  | earn
318 |          | 26-FEB-1987 19:00:16.13 |          |          | 318  |          |          | 318  | earn
129 |          | 26-FEB-1987 17:32:15.34 |          |          | 129  |          |          | 129  | earn
13 |          | 26-FEB-1987 15:20:13.09 |          |          | 13   |          |          | 13   | earn
169 |          | 26-FEB-1987 17:45:27.43 |          |          | 169  |          |          | 169  | earn
145 |          | 26-FEB-1987 17:34:23.01 |          |          | 145  |          |          | 145  | earn
113 |          | 26-FEB-1987 16:45:44.50 |          |          | 113  |          |          | 113  | earn
160 |          | 26-FEB-1987 17:43:30.51 |          |          | 160  |          |          | 160  | earn
11 |          | 26-FEB-1987 15:18:59.34 |          |          | 11   |          |          | 11   | earn
151 |          | 26-FEB-1987 17:35:59.59 |          |          | 151  |          |          | 151  | earn
1811 |          | 3-MAR-1987 09:35:02.37 |          |          | 1811 |          |          | 1811 | earn
1896 |          | 3-MAR-1987 11:15:00.11 |          |          | 1896 |          |          | 1896 | earn
1864 |          | 3-MAR-1987 11:08:29.13 |          |          | 1864 |          |          | 1864 | earn

(38 rows)
cqlsh:reuters>
```

```
cqlsh:reuters> SELECT COUNT(*) FROM reuters WHERE topics = 'earn' LIMIT 30;
count
-----
96
(1 rows)

Warnings :
Aggregation query used without partition key
cqlsh:reuters>
```

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 unnormalize 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.

author	Number_of_articles
thatcher ryzhkov	1
stoltenberg james-baker	3
stoltenberg volcker	1
hawke	3
concepcion	4
zhao-ziyang cavaco-silva	2
james-baker lawson	1
volcker greenspan	8
du-plessis	5
james-miller reagan	2
aquino ongpin	1
reagan james-baker greenspan	1
stoltenberg balladur	1
eyskens delors stoltenberg lawson	1
suharto subroto	1
sprinkel	1
james-baker	116
mohammed-ali-abal-khail	2
howard-baker	3
james-baker stoltenberg poehl	2
del-mazo mancera-aguayo	2
mikulic	1

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 {};
```

The screenshot shows the DataStax Studio interface. The top bar indicates the session is connected to 'LOCAL | Cassandra 4.1.3 : NoSql : reuters : Favorite / Q1'. The main area contains a SQL query window and a results window.

SQL Query:

```
1 SELECT countGroup(places) FROM reuters.reuters;
```

Results:

```
reuters.countgroup(places)
1 { ...
  "usa": 10874,
  "canada": 849,
  "usr usa": 16,
  "uk austria": 2,
  "": 2761,
  "uk": 933,
  "bahrain saudi-arabia": 8,
  "norway": 19,
  "west-germany": 328,
  "france": 272,
  "usa sudan": 2,
  "usa uk japan": 4,
  "usa japan": 184,
  "argentina usa": 5,
  "usa australia": 14,
  "usa taiwan": 13,
```

Line 3, column 16, location 32

Close Window