Coursework 1: Cassandra with Reuters dataset

CHOORIYAKUMARAN Vidushan, CHASSERAY David,
PUVIKARAN Sajeevan
02/02/2018

Table des matières

Step 1: Creation of tables in Cassandra

The dataset Reuters contains the following attributes:

- _id
- Companies
- Date
- Exchanges
- Orgs
- People
- Places
- Text: An object with 3 attributes – Body, Dateline, Title
- Topics

All the data is in the JSON format in the reuters_26-09-1997 file.

```
CREATE KEYSPACE IF NOT EXISTS reuters WITH REPLICATION = { 'class' :
'SimpleStrategy', 'replication_factor': 3 };

CREATE TYPE IF NOT EXISTS TextObject(dateline VARCHAR, title VARCHAR, body
VARCHAR);

CREATE TABLE IF NOT EXISTS Articles(
      id INT, companies VARCHAR,date VARCHAR, exchanges VARCHAR, orgs
VARCHAR, people VARCHAR,places VARCHAR,
      text FROZEN<TextObject>, topics VARCHAR,
      PRIMARY KEY(id)
);
```

We decide to import all the data in a same table. If it is necessary for a certain query, we will denormalize and get the result.

Step 2: Importation of the dataset into the database

We choose to use a Python driver for Cassandra in order to import data. The driver is available by accessing to the following link: http://datastax.github.io/python-driver/getting_started.html.

```
from cassandra.cluster import Cluster
cluster = Cluster()
session = cluster.connect('reuters')
```

Figure 1: Connection to the cluster

The connect() method takes an optional keyspace argument which sets the default keyspace as reuters for all queries made through that Session.

```
import json
```

```
#data = json.loads('reuters_26-09-1997.json')
#print(data)

data = []

with open('reuters_26-09-1997.json', encoding='utf-8') as data_file:
    for line in data_file:
        data.append(json.loads(line))

for element in data:
    element['id'] = element.pop('_id')
```

Figure 2: Importation of data in an array.

Moreover, we modify the key '_id' by 'id' in order to match with the attribute in the database.

```
prepared = session.prepare('INSERT INTO Articles JSON ?')

for element in data:
    article = {'id': element['id'],
               'companies': element['companies'],
               'date': element['date'],
               'exchanges': element['exchanges'],
               'orgs': element['orgs'],
               'people': element['people'],
               'places': element['places'],
               'text': element['text'],
               'topics': element['topics']
              }

    session.execute(prepared, [json.dumps(article)])
```

Figure 3: Import Query for each element in the array

Verification

```
SELECT COUNT(*) FROM Articles limit 1000000;
```

Results | Query Trace

count
21495

All the data has been imported.

Step 3: Queries

Easy Queries:

1. Search for all the articles released only in USA. (with and without allow filtering)

- With Allow Filtering:

```
SELECT id FROM Article WHERE places = 'usa' ALLOW FILTERING ;
```

1 selected statement successfully executed in 2006 ms. Retrieved 300 rows

- Without Allow Filtering by creating an index:

```
CREATE INDEX IF NOT EXISTS article_places ON Articles(places);

//Query
SELECT id FROM Articles WHERE places = 'usa';
```

1 selected statement successfully executed in 1883 ms. Retrieved 300 rows.


```
SELECT COUNT(*) FROM Articles WHERE places = 'usa' LIMIT 12000;
```

Using an index is better in this example because there are 10874 articles tagged ‘usa’.

2. Search for all the articles that is specifically tagged with someone’s name.

```
SELECT people FROM Articles WHERE people > '' LIMIT 10000 ALLOW
FILTERING;
```

1 selected statement successfully executed in 738 ms. Retrieved 1148 rows

```
SELECT COUNT(people) FROM Articles WHERE people > '' LIMIT 10000
ALLOW FILTERING;
```

Results | Query Trace

system.count(people)
1148

Complex Queries:

1. Count the number of countries in which an article has been released using the 'places' attribute.

```
CREATE OR REPLACE FUNCTION countCountries(val varchar) CALLED ON NULL
INPUT RETURNS int LANGUAGE java as 'int res = 0;
if(val != ""){
String[] countries = val.split(" ");

return countries.length;}
return 0;';
```

```
SELECT places, countCountries(places) FROM Articles;
```

Results | Query Trace

places	reuters.countcountries(places)
usa	1
usa	1
uk austria	2
	0
uk	1
bahrain saudi-arabia	2
usa	1
usa	1
usa	1
usa	1

2. Get titles from articles that has been released only in France

```
SELECT id, text.title FROM Articles WHERE places = 'france';
```

Results | Query Trace

id	text.title
2731	GATT STUDYING AGRICULTURAL SUBSIDIES CEILING
4701	FRENCH 1986 BUDGET DEFICIT CUT TO 141.1 BILLION FRAN...
13535	FRANC ZONE MINISTERS CALL FOR MORE AID
5900	CREDIT AGRICOLE REPLACES BNP IN HACHETTE TV BID

1 selected statement successfully executed in 1199 ms. Retrieved 272 rows

Hard Queries:

1. Number of articles depending on the country

```
CREATE OR REPLACE FUNCTION countGroupPlaces (state map<varchar, int>,
val1 varchar)
CALLED ON NULL INPUT RETURNS map<varchar, int> LANGUAGE java
AS 'Integer count = (Integer)state.get(val1);
if(count == null)
      count = 0;
count ++;
state.put(val1,count);
return state;';
```

```
CREATE OR REPLACE AGGREGATE countGroup2 (varchar)
      SFUNC countGroupState STYPE map<varchar, int>
      INITCOND {};
```

As it takes too much time on DataStax, we try on cqlsh command with a request timeout of 3600. It gives the following result.

C:\Windows\System32\cmd.exe - cqlsh --request-timeout=3600

```
-------
 OrderedMapSerializedKey([(u'usa', 10874), (u'canada', 849), (u'ussr usa', 16), (u'uk austria', 2), (u'', 276
u'usa uk japan', 4), (u'usa japan', 184), (u'argentina usa', 5), (u'usa australia', 14), (u'usa taiwan', 13),
(u'usa sweden', 12), (u'india malaysia', 1), (u'singapore', 43), (u'thailand', 33), (u'belgium', 117), (u'phi
1), (u'philippines', 74), (u'japan usa', 120), (u'usa senegal', 1), (u'brazil usa', 10), (u'china', 97), (u't
```