Advanced topics in NoSql databases

S2/3 - Cassandra

Submitted by

Camara Abdoul-karim, Maxime de la Tour, Yuning LI

DIA1 31 Jan. 2021

Basic information

Code link: https://github.com/Yuning-LI/Nosql-Cassandra

Dataset Name : ENRON

Complexity: 1 (requires 5 simple queries, 2 complex queries, 2 hard queries)

Schema:

▼ [datasets.enron]	100.0%	Collection
▶_id	100.0%	ObjectId
▶bcc	100.0%	Array
▶cc	100.0%	Array
▶ctype	100.0%	String
▶date	100.0%	String
▶fname	100.0%	String
▶folder	100.0%	String
▶fpath	100.0%	String
▶ mid	100.0%	String
▶ recipients	100.0%	Array
▶replyto	100.0%	Null
▶sender	100.0%	String
▶subject	100.0%	String
▶text	100.0%	String
▶to	100.0%	Array

Data cleaning

In order to import the dataset, we need to firstly organize the data format.

What we need to do is as follow:

- 1. add the CQL command 'INSERT INTO <tablename> JSON' to the beginning.
- 2. add " and; to each line
- 3. change the key "to" to "toaddress" as it can not be recognized by Cassandra
- 4. rewrite the key "_id" to simplify it.
- 5. rewrite the key "fname" and change it to integer type.

So we create a python script (dataCleaning.py) for data cleaning, the python code is follow:

```
dataCleaning.py ×
def ctype_folder.py
                                            JS index.js config
                                                                 JS index.js fbeamer
Users > liyuning > Desktop > nosql > S2_Cassandra > ♦ dataCleaning.py > ♦ dataCleaning
       import json
       def dataCleaning (input_json_file, output_json_file):
           file_in = open(input_json_file, "r")
           file_out = open(output_json_file, "w")
           data = file_in.readlines() # read each json line in the dataset
           for line in data:
               json_data = json.loads(line) # get the json object
               json_data['id'] = json_data['_id']['$oid'] # rewrite the key "_id"
               del json_data['_id']
               json_data['toaddress'] = json_data['to'] # rewrite the key "to"
               del json_data['to']
 15
               json_data['fname'] = int(json_data['fname'][:-1]) # rewrite the key "fname"
               oldline = json.dumps(json_data) # transform the json object to string
               newline = 'INSERT INTO emails JSON \'' + oldline + '\';' + '\n' # add CQL command
               file_out.writelines(newline)
           file_in.close()
           file_out.close()
       \label{lem:dataCleaning('/Users/liyuning/Desktop/nosql/enron.json', '/Users/liyuning/Desktop/nosql/enronClean.json')} \\
```

After dealt by python, we get the dataset with the form as follow:

```
1 - {
 2
       "toaddress": ["sherri.reinartz@enron.com"],
 3
       "sender": "rosalee.fleming@enron.com",
 4
       "recipients": ["sherri.reinartz@enron.com"],
 5
       "cc": [],
 6
       "text": "Ken will attend both meetings.\n\nRosie\n\n\nSherri Reinartz\n01/12,
 7
       "mid": "18133935.1075840283210.JavaMail.evans@thyme",
 8
       "fpath": "enron mail 20110402/maildir/lay-k/ sent/1.",
 9
       "bcc": [],
10
       "replyto": null,
11
       "ctype": "text/plain; charset=us-ascii",
12
       "fname": 1,
13
       "date": "2000-01-12 08:24:00-08:00",
14
       "folder": "_sent",
15
       "id": "52af48b5d55148fa0c199643",
16
       "subject": "Re: EXECUTIVE COMMITTEE MEETINGS - MONDAY, JANUARY 17"
17 }__
```

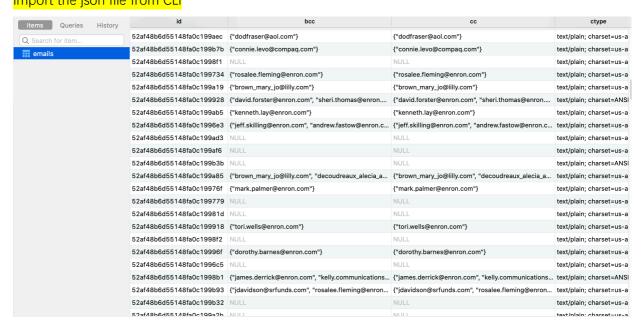
And the new json file "enronClean.json" as follow:

```
["sherri.reinartz@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["rob.bradley@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["rob.bradley@enron.com"], "sender": "tori.wells@enron.com", "reci ["michael@optsevents.com"], "sender": "tori.wells@enron.com", "recipi ["jaime.alatorre@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["michael.mann@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["michael.mann@enron.com"], "sender": "rosalee.fleming@enron.com", "re ["tori.wells@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["ted.enloe@compaq.com"], "sender": "rosalee.fleming@enron.com", "reci ["ted.enloe@compaq.com"], "sender": "rosalee.fleming@enron.com", "reci ["tatherine.brown@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["michael.hicks@enron.com"], "sender": "rosalee.fleming@enron.com", "reci ["diane.bazelides@enron.com"], "sender": "rosalee.fleming@enron.com", "["diane.bazelides@enron.com"], "sender": "rosalee.fleming@enron.com", "recipi ["dyergin@cera.com"], "sender": "rosalee.fleming@enron.com", "recipi ["tori.wells@enron.com", "sender": "rosalee.fleming@enron.com", "recipi ["tori.wells@enron.com", "sender": "rosalee.fleming@enron.com", "recipi ["tori.wells@enron.com"], "sender": "rosalee.fleming@enron.com", "recipi ["tori.wells@e
INSERT INTO emails JSON INSERT INTO emails JSON
                                                                                                  '{"toaddress":
'{"toaddress":
                            INTO emails JSON
                                                                                                   '{"toaddress":
 INSERT INTO emails JSON
                                                                                                             'toaddress":
 INSERT INTO emails JSON
                                                                                                    '{"toaddress":
                            INTO emails JSON
 INSERT
                                                                                                              'toaddress":
 INSERT INTO emails JSON
                                                                                                    '{"toaddress":
 INSERT INTO emails JSON INSERT INTO emails JSON
                                                                                                    '{"toaddress":
                                                                                                            "toaddress"
                            INTO emails JSON
                                                                                                   '{"toaddress":
 INSERT INTO emails JSON INSERT INTO emails JSON
                                                                                                   '{"toaddress":
                                                                                                    '{"toaddress":
  INSERT
                            INTO emails JSON
                                                                                                            "toaddress
 INSERT INTO emails JSON
                                                                                                    '{"toaddress":
 INSERT INTO emails JSON
 INSERT
                            INTO emails JSON
                                                                                                              'toaddress":
  INSERT INTO emails JSON
 INSERT INTO emails JSON INSERT INTO emails JSON
                                                                                                              toaddress":
                                                                                                   '{"toaddress":
  INSERT INTO emails JSON
                                                                                                    '{"toaddress":
 INSERT INTO emails JSON
                            INTO emails JSON
 INSERT
```

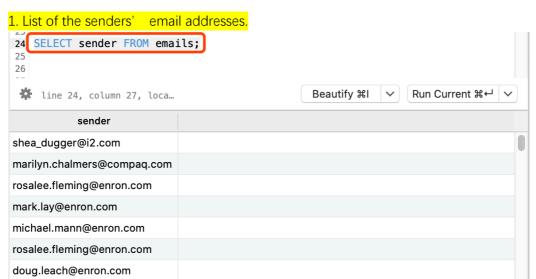
Then we create the table "emails"

```
3 CREATE TABLE emails (
       id TEXT,
 4
        sender TEXT,
 5
       recipients LIST<TEXT>,
 6
       cc LIST<TEXT>,
text TEXT,
 7
 8
       mid TEXT,
 9
       fpath TEXT
10
       bcc LIST<TEXT>,
11
       toaddress LIST<TEXT>,
12
       replyto TEXT,
13
       ctype TEXT,
14
        fname INT,
15
       date TEXT,
16
       folder TEXT
17
        subject TEXT
18
19
       PRIMARY KEY (sender)
20 );
21
22 ALTER TABLE emails WITH GC_GRACE_SECONDS = 0;
```

Import the ison file from CLI



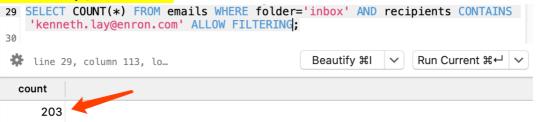
Simple queries * 5



2. Number of the emails sent from "tom.siekman@compag.com"



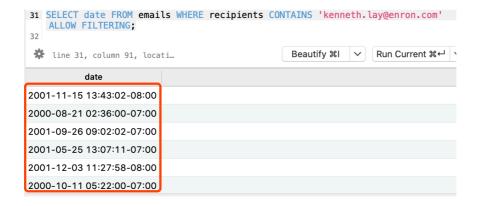
3. Number of the emails where folder equals to 'inbox' and the recipients contains 'kenneth.lay@enron.com'



4. The content of the email sent from jeffrey.westphal@enron.com at the date 2001-09-26

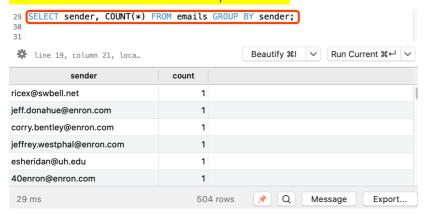


5. See on which days did the address 'kenneth.lay@enron.com' receive new emails.



Complex queries * 2

1. Count the number of the emails per address



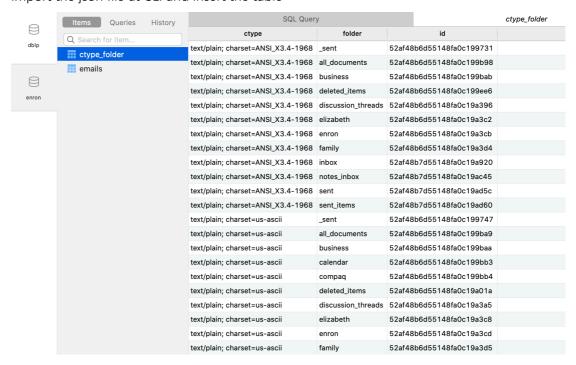
2. Distribution of text types for the emails in the folder "notes_inbox"

In order to avoid the single primary key problem, we create another table containing "ctype" and "folder" information.

Create another python file (ctype_folder.py) to reform the original data

```
ctype_folder.py X
Js index.js config
                                                                JS index.js fbeamer
dataCleaning.py
Users > liyuning > Desktop > nosql > S2_Cassandra > ♦ ctype_folder.py > ♦ dataReform
       import json
       def dataReform (input_json_file, output_json_file):
           file_in = open(input_json_file, "r")
           file_out = open(output_json_file, "w")
           data = file_in.readlines() # read each json line in the dataset
           for line in data:
               json_data = json.loads(line) # get the json object
               newjson = {}
               newjson['id'] = json_data['_id']['$oid']
               newjson['ctype'] = json_data['ctype']
               newjson['folder'] = json_data['folder']
               oldline = json.dumps(newjson) # transform the json object to string
               newline = 'INSERT INTO ctype_folder JSON \'' + oldline + '\';' + '\n'
               file_out.writelines(newline)
 19
 20
           file_in.close()
           file_out.close()
       dataReform('/Users/liyuning/Desktop/nosql/enron.json', '/Users/liyuning/Deskto
```

Import the ison file at CLI and insert the table



Imply the query

```
32 DROP TABLE ctype_folder;
CREATE TABLE ctype_folder (
id TEXT,
         ctype TEXT,
35
         folder TEXT,
36
         PRIMARY KEY ((ctype), folder)
38 );
40 ALTER TABLE emails WITH GC_GRACE_SECONDS = 0;
41 CREATE INDEX btree_ctype_folder_id on ctype_folder(id);
42 SELECT ctype FROM ctype_folder WHERE folder='notes_inbox' GROUP BY ctype
ALLOW FILTERING;
43
44
                                                           Beautify XI ∨ Run Current X← ∨
 🗱 line 43, column 1, loca...
               ctype
text/plain; charset=ANSI_X3.4-1968
text/plain; charset=us-ascii
```

Hard queries * 2

1. the first recipient of the email with subject "Anonymous report on violations by senior Enron officials"

Modify the Cassandra.yaml config file to allow user define function:

```
# This threshold can be adjusted to minimize logging if necessary
# gc_log_threshold_in_ms: 200
# If unset, all GC Pauses greater than gc_log_threshold_in_ms will log at
# INFO level
# UDFs (user defined functions) are disabled by default.
# As of Cassandra 3.0 there is a sandbox in place that should prevent execution
of evil code
enable_user_defined_functions: true
# Enables scripted UDFs (JavaScript UDFs).
Java UDFs are always enabled, if enable_user_defined_functions is true.
# Enable this option to be able to use UDFs with "language javascript" or any cu
stom JSR-223 provider.
# This option has no effect, if enable_user_defined_functions is false.
enable_scripted_user_defined_functions: false
# The default Windows kernel timer and scheduling resolution is 15.6ms for power
 conservation.
# Lowering this value on Windows can provide much tighter latency and better thr
oughput, however
# some virtualized environments may see a negative performance impact from chang
ing this setting
"cassandra.yaml" 1279L, 59843C
                                                          1110,1
                                                                       87%
```

Restart Cassandra server, define function to return the first element of the list, imply the query:

```
33 CREATE OR REPLACE FUNCTION projectList (num INT, tab LIST<TEXT>)
       RETURNS NULL ON NULL INPUT RETURNS TEXT LANGUAGE Java
34
       AS 'return (String)tab.get(num);';
35
36 SELECT projectList(0, toaddress) AS first_recipient FROM emails WHERE
    subject='Anonymous report on violations by senior Enron officials' ALLOW
   FILTERING;
                                                      Beautify XI
                                                                       Run Current #←
 🗱 line 36, column 52, location 1...
  first_recipient
jskilli@enron.com
```

2. Compute the average fname of the emails sent to 'kenneth.lay@enron.com'

Define User Define Aggregate functions

```
40 CREATE OR REPLACE FUNCTION avgState ( state tuple<int,bigint>, val int )
   CALLED ON NULL INPUT RETURNS tuple<int, bigint> LANGUAGE java
41 AS 'if (val !=null) {
           state.setInt(0, state.getInt(0)+1);
42
           state.setLong(1, state.getLong(1)+val.intValue()); }
43
44
       return state;';
45
46 CREATE OR REPLACE FUNCTION avgFinal ( state tuple<int,bigint> ) CALLED ON NULL
   INPUT RETURNS double LANGUAGE java
47 AS 'double r = 0;
48
       if (state.getInt(0) == 0) return null;
       r = state.getLong(1);
49
       r/= state.getInt(0);
       return Double.valueOf(r);';
51
53 CREATE AGGREGATE IF NOT EXISTS average ( int )
       SFUNC avgState STYPE tuple<int,bigint>
       FINALFUNC avgFinal INITCOND (0,0);
55
57 SELECT average(fname) FROM emails WHERE toaddress CONTAINS
   'kenneth.lay@enron.com' ALLOW FILTERING;
58
50
line 57, column 99, location 2...
                                                    Beautify XI
                                                                    Run Current #← \
enron.average(fname)
```

688.8919860627178