Cassandra Database Report

Junhui LI, Ge QIU, Zhiheng WU, Yue JI

A4 IBO4

Dataset:

Stock exchange

Abstract :

The dataset Stock Exchange (JSON) and Nosql Cassandra are employed for this report. The suitable key space named ‘stock’ is established for three levels of quiries, the table ‘stock_esilv’ is generated after careful consideration of relationship of the data structure and quiries. we used different methods to import JSON format stock exchange dataset including convertion of JSON into CSV and then reading by cassandra and reading JSON directly from cassandra.

1. Introduction

Apache Cassandra is an open source, distributed and decentralized/distributed storage system (database), for managing very large amounts of structured data spread out across the world. It provides highly available service with no single point of failure.

2. Data prepossessing

We inserted data manually and changed « _id » to « id », the parameter of blanks to underline, the time parameter type to ‘yyyy-mm-dd'.

Here is our script about creating table:

```
CREATE KEYSPACE IF NOT EXISTS stock
WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 3 };
USE stock;
CREATE TYPE IF NOT EXISTS description_s (Country VARCHAR, Sector VARCHAR, Industry VARCHAR);
CREATE TYPE IF NOT EXISTS ratio_s (quick double, current double);
CREATE TYPE IF NOT EXISTS performance_s (Year double, Half_Year double,Month double ,Week double);


CREATE TABLE IF NOT EXISTS stock_esilv (
    ident VARCHAR ,
    Company VARCHAR primary key ,
    Price double,
    Earnings_Date date,
    description description_s,
    twenty_Day_Simple_Moving_Average double,
    twohundreds_Day_Simple_Moving_Average double,
    fifty_Day set<double>,
    fiftytwo_Week set<double>,
    Analyst_Recom double,
    Average_True_Range double,
    Average_Volume double,
    Beta double,
    Change double,
    EPS_ttm double,
    ROI double,
    ratio ratio_s,
    performance performance_s,
) ;
```

3. Quiries

3.1 Question: *Count the number of stock where Earnings_Date is 2013-11-14*

Query: select count(*) from stock_esilv where Earnings_Date = '2013-11-14' ALLOW FILTERING;

Result:

```
cqlsh:stock> select count(*) from stock_esilv where Earnings_Date = '2013-11-14' ALLOW FILTERING;

 count
-------
    83

(1 rows)
```

3.2. Question *: Company's name from stock which price is over 500*

Query : select Company from stock_esilv where price > 500 ALLOW FILTERING ;

Result:

```
cqlsh:stock> select Company from stock_esilv where price > 500 ALLOW FILTERING ;

 company
-----------------------------------------
                          Apple Inc.
                         Google Inc.
                        Markel Corp.
            Mastercard Incorporated
       The Washington Post Company
      Chipotle Mexican Grill, Inc.
                          NVR Inc.
        priceline.com Incorporated
 White Mountains Insurance Group, Ltd.
Altisource Asset Management Corporation
                     Seaboard Corp.

(11 rows)

Warnings :
Read 6720 live rows and 11075 tombstone cells for query SELECT * FROM stock.stock_esilv WHERE price > 500.0 AND  LIMIT 1
00 (see tombstone_warn_threshold)

cqlsh:stock>
```

3.3 Question : *Create an index on attribut 'Earnings_Date'of column family stock_esilv*

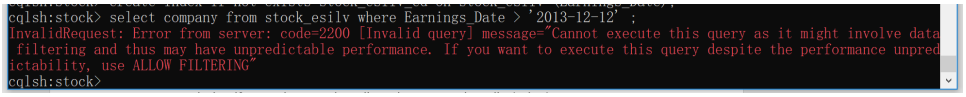
Query : create index if not exists stock_esilv_ed on stock_esilv (Earnings_Date);

Result:

3.4 Question : *Find titles of companies where Earnings_Date over 2013-12-12*

Query *:* select company from stock_esilv where Earnings_Date > '2013-12-12' ;

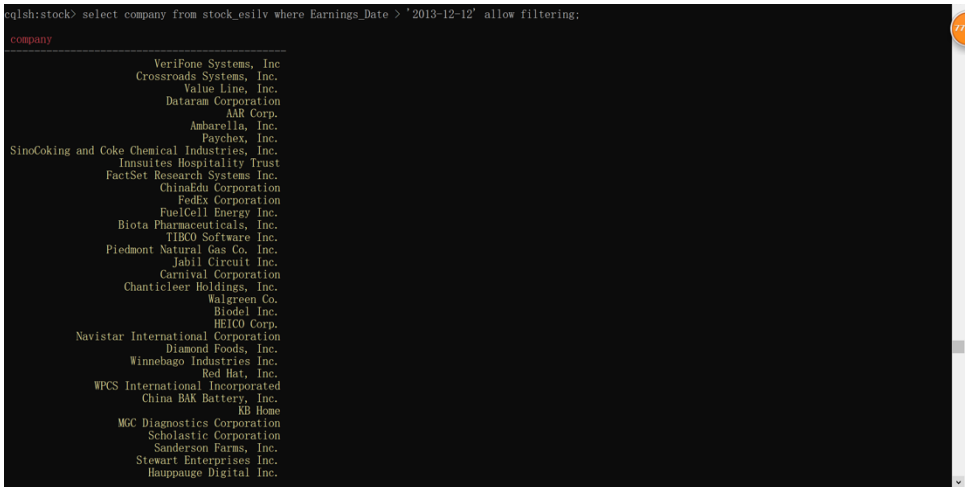
Result:

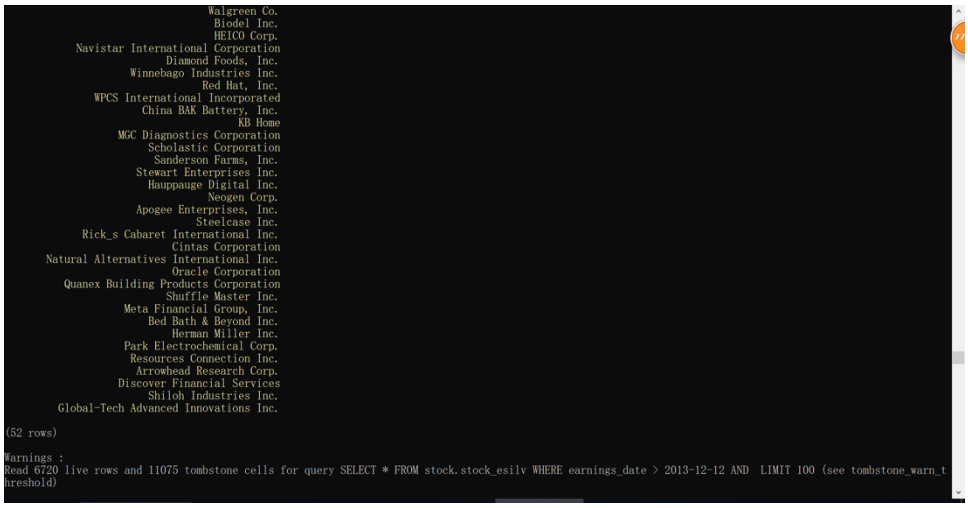


It's wrong,because index in Cassandra doesn't accept fuzzy comparisons except for '='.

Query : select company from stock_esilv where Earnings_Date > '2013-12-12' ALLOW FILTERING;

Result:





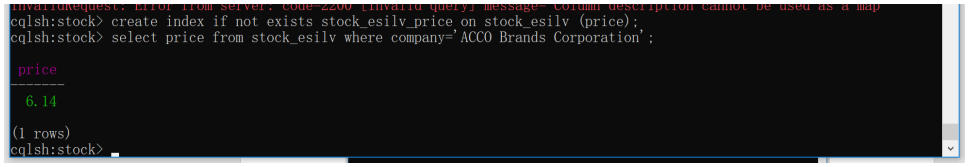
3.5

Question : *create an index on 'price and give price from company of ACCO Brands Corporation*

Query : create index if not exists stock_esilv_price on stock_esilv (price);

select price from stock_esilv where company='ACCO Brands Corporation';

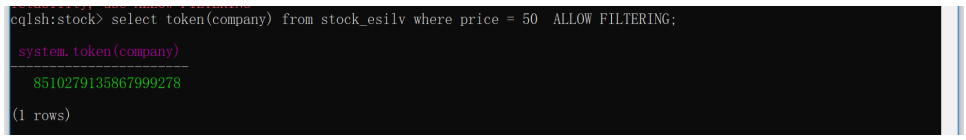
Result:



3.6

Question : *Find companies where price is 50(By using token)*

Query : select token(company) from stock_esilv where price = 50 ALLOW FILTERING;



3.7

Question : Find country of stock where price is 10

Query : select description.country from stock_esilv where price = 10 limit 10;

```
cqlsh:stock> create index if not exists stock_esilv_price on stock_esilv (price);
cqlsh:stock>
cqlsh:stock> Select description.country from stock_esilv where price = 10 limit 10;

 description.country
---------------------
                 USA
                 USA
              Israel
                 USA
                 USA

(5 rows)
cqlsh:stock>
```

3.8

Question : *delete Apple Inc company's stock*

Query : delete from stock_esilv where company='Arden Group Inc.';

Result :

3.9

Question : update stock price from 12 to 13 for company ''Alcoa, Inc'

Query : UPDATE stock_esilv SET price=12 WHERE company='Alcoa, Inc';

Result :

```
cqlsh:stock> select price from stock_esilv where company = 'Alcoa, Inc';

 price
-------
    12

(1 rows)
cqlsh:stock> update stock_esilv set price = 13 where company = 'Alcoa, Inc';
cqlsh:stock> select price from stock_esilv where company = 'Alcoa, Inc';

 price
-------
    13
```

3.10

Question : *Calculate the average price of all US products*

Query : select AVG(price) from stock_esilv where description['Country'] = 'USA' allow filtering;

Result :

```
cqlsh:stock> select AVG(price) from stock_esilv where description['Country'] = 'USA' allow filtering;

 system.avg(price)
-------------------
           34.1448

(1 rows)

Warnings :
Aggregation query used without partition key
```

For use'country' in where statement we modified type of description in map.

```
CREATE KEYSPACE IF NOT EXISTS stock_map
WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 3 };
USE stock_map;
CREATE TYPE IF NOT EXISTS description_s (Country VARCHAR, Sector VARCHAR, Industry VARCHAR);
CREATE TYPE IF NOT EXISTS ratio_s (quick double, current double);
CREATE TYPE IF NOT EXISTS performance_s (Year double, Half_Year double,Month double ,Week double);


CREATE TABLE IF NOT EXISTS stock_esilv (
 ident VARCHAR,
 Company VARCHAR primary key,
 Price double,
 Earnings_Date date,
 description map<text,varchar>,
 twenty_Day_Simple_Moving_Average double,
 twohundreds_Day_Simple_Moving_Average double,
 fifty_Day set<double>,
 fiftytwo_Week set<double>,
 Analyst_Recom double,
 Average_True_Range double,
 Average_Volume double,
 Beta double,
 Change double,
 EPS_ttm double,
 ROI double,
 ratio ratio_s,
 performance performance_s,
) ;
```

4. **Disscussion and Conclusion**

Disscussion

1.

We tried two methods for inserting data set. At first we wanted to write a program that automatically reads the data and inserts it into the database.But there's a problem with the data of TYPE type.TYPE is included in our table and we don't know how to insert the TYPE form so we give up the first method.

2.

We initially wanted to use MAP type because MAP can do queries against an attribute in a map array but TYPE type cannot.

We use TYPE type rather than MAP type at last because MAP does not accept any null values while there are too many null values in our dataset.

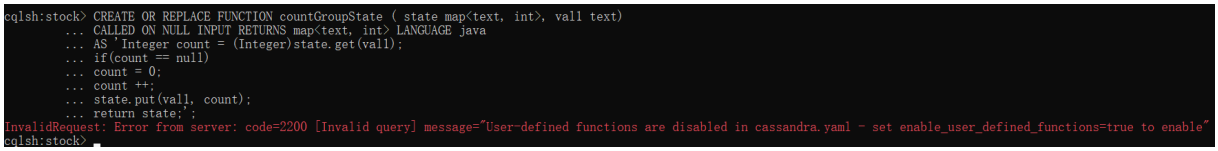
3.

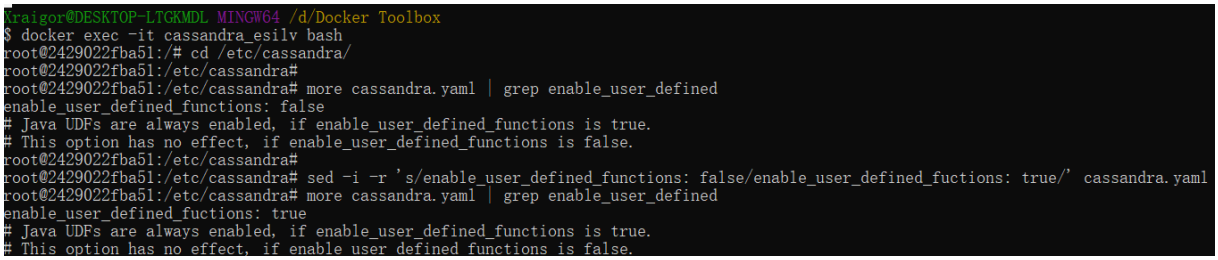
The reason for index does not support fuzzy comparisons (except for '=')

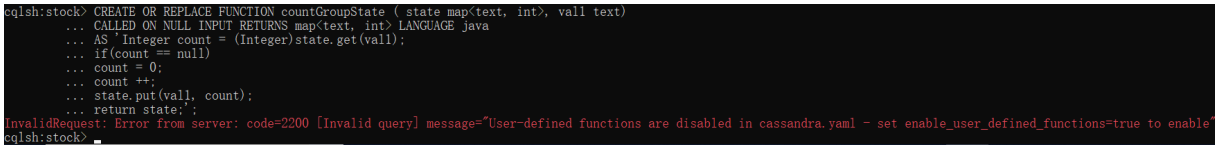
The primary key of the hidden column family which includes the index data is stored disorderly.

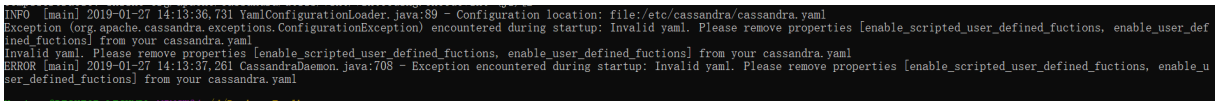
4.

First try to build the function it need to enable the UDP. We cant modify the parameter 'user_define' with 'true' value. After that it doesn't work as well. It shows that the success is changed to a, but when we define the function it can't be defined because the state is always false. We found a way to solve this problem by restarting the container. But we can't start it. And we found the problem below using 'docker logs cassandra_esilv'.









Conclusion:

This project, given the many difficulties encountered, really allowed us to understand the basics of Cassandra. We realized that an organization, clarity and good structure in the Cassandra is paramount so as not to fall into the mistakes and pitfalls of such language.

Finally, this project allowed us to concretely practice the Cassandra and the data structure, which may later in our journey help us better understand the logic of NoSQL.