TP2 – NoSQL Cassandra

Restaurant Inspections

Manon GARDIN

Matias OTTENSEN

Alexandre GARNIER

Tiphaine KACHKACHI

# Chapter 1 – Create the database

## Files transfer

We drag and drop the restaurants.json into the files of our Cassandra container.

Une image contenant texte, logiciel, Icône d’ordinateur, nombre

Description générée automatiquement

## Create the keyspace

In the CLI, use the command :

CREATE KEYSPACE IF NOT EXISTS RESTO\_INSPEC

WITH REPLICATION =

{ 'class': 'SimpleStrategy', 'replication\_factor': 3 };

And then,

USE RESTO\_INSPEC;

Une image contenant texte, Police, capture d’écran, algèbre

Description générée automatiquement

## Create Tables

Let’s understand the JSON structure of our dataset with the structure of one insert :

[

{

"address":

{

"building": "1007",

"coord": {

"type": "Point",

"coordinates" : [-73.856077, 40.848447]

},

"street": "Morris Park Ave",

"zipcode": "10462"

},

"borough": "Bronx",

"cuisine": "Bakery",

"grades": [

{

"date": {

"$date": 1393804800000

},

"grade": "A",

"score": 2

},

{

"date": {

"$date": 1378857600000

},

"grade": "A",

"score": 6

},

{

"date": {

"$date": 1358985600000

},

"grade": "A",

"score": 10

},

{

"date": {

"$date": 1322006400000

},

"grade": "A",

"score": 9

},

{

"date": {

"$date": 1299715200000

},

"grade": "B",

"score": 14

}

],

"name": "Morris Park Bake Shop",

"restaurant\_id": "30075445"

},

INSERT2…

]

Visually, we can see that the first property address will need a full table containing building, coord, street and zipcode. In each coord, we have a type of coordinate and the X and Y coordinates. In order to not create another table, we can add them as different attributes.

Also the property grade is an array with different elements. We can conclude that we will need a grades table with the three attributes grade, date and score. One restaurant can have several grades, so we add the restaurant\_id attributes as the equivalent of a foreign key.

And finally the table restaurants that will have all the other attributes.

Create the tables in file CreaTable :

CREATE TABLE restaurants (

restaurant\_id text PRIMARY KEY,

name text,

borough text,

cuisine text

);

ALTER TABLE restaurants WITH GC\_GRACE\_SECONDS=0;

CREATE TABLE addresses (

address\_id text PRIMARY KEY,

building text,

street text,

zipcode text,

coord\_type text,

coord\_X float,

coord\_Y float

);

ALTER TABLE addresses WITH GC\_GRACE\_SECONDS=0;

CREATE TABLE grades (

restaurant\_id text,

date timestamp,

grade text,

score int,

PRIMARY KEY (restaurant\_id, date)

);

ALTER TABLE grades WITH GC\_GRACE\_SECONDS=0;

Une image contenant texte, capture d’écran, Police, algèbre

Description générée automatiquementNow, we open TablePlus, and select the database we created.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Description générée automatiquement

## Fixing Json file

We found out that the format of the Json is not correct, so we needed to do a script to correct the file. What was wrong in that file ?

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

The file lack the first and last '['. We did the fixing\_json.py :

Une image contenant texte, capture d’écran, logiciel, affichage

Description générée automatiquement

In which we created a whole new file “restaurants\_fixed.json”. We added the '[' character and we join all the lines of the original files separated by a comma.

After that, the data didn’t really need any kind of cleaning.

## Import the data

Then, we execute the data\_importation.py file in the Cassandra container.

In this code below, we setup the connection to the database :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

And then, we add the data with a query, for each table. (See in the data\_importation.py file)

Here is the command :

docker exec -it Cassandra python3 data\_importation.py

# Chapter 2 – Querying Cassandra

## Simple Queries

### List of restaurants located in Bronx :

Query: SELECT \* FROM restaurants WHERE borough = 'Bronx';

This query selects every column from the "restaurants" table where the value of the "borough" column is 'Bronx'.

This query won’t be executed since it would take too much resources. We need to either use ALLOW FILTERING or create an index on the borough column of restaurants :



*Result:*

Une image contenant texte, capture d’écran

Description générée automatiquement

### List of Japanese restaurants’ name and borough

SELECT name, borough FROM restaurants WHERE cuisine='Japanese' ALLOW FILTERING;

This query selects the Japanese restaurants and more particularly the name and the borough in which they are located.

For the same reason of the previous query, we use ALLOW FILTERING here.

*Result :*

*Une image contenant texte, capture d’écran, Police

Description générée automatiquement*

### List the restaurants that are located in Brooklyn

Query: SELECT COUNT(\*) FROM restaurants WHERE borough = 'Brooklyn';

This query counts the number of rows in the "restaurants" table where the value of the "borough" column is 'Brooklyn'.

Since we already have an index on borough that we created earlier, we don’t need to use ALLOW FILTERING.

*Result:* We have 6805 restaurants located in Brooklyn.

Une image contenant texte, Police, capture d’écran, conception

Description générée automatiquement

**List 5 pizzerias of Manhattan**

Query : SELECT name, borough FROM restaurants WHERE borough = 'Manhattan' AND cuisine = 'Pizza' LIMIT 5 ALLOW FILTERING;

This query selects the "name" and "borough" columns from the "restaurants" table where the value of the "borough" column is 'Manhattan' and "cuisine" is ‘Pizza’.

We use LIMIT 5 to limit the result to 5 rows.

*Result* : The names and boroughs of the first 5 restaurants located in Manhattan.

Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement

### Query: SELECT name, addresses.zipcode FROM restaurants WHERE borough = 'Queens' AND grades[0].score > 10;

This query selects the "name" column from the "restaurants" table and the "zipcode" column from the "addresses" table. It retrieves rows where the value of the "borough" column is 'Queens' and the score of the first grade is greater than 10.

Expected result: The names of restaurants in Queens along with their corresponding zip codes, where the first grade's score is above 10.

### Query: SELECT SUM(SIZE(grades)) AS total\_grades FROM restaurants;

This query calculates the sum of the sizes of the "grades" list for all rows in the "restaurants" table.

Expected result: The total number of grades across all restaurants.

## Complex queries

### List of restaurants’ name and cuisine which had a really good first grade

Query: SELECT name, cuisine FROM restaurants WHERE grades[0].score > 8;

This query selects the "name" and "cuisine" columns from the "restaurants" table where the score of the first grade (grades[0].score) is greater than 8.

Expected result: A list of restaurant names and their cuisines where the first grade's score is greater than 8.

### Query: SELECT \* FROM restaurants WHERE FILTER(geodistance(coord['coordinates'][0], coord['coordinates'][1], long, lat) < 1);

This query selects all columns from the "restaurants" table using a filter condition. It filters the rows based on the geodistance between the coordinates stored in the "coord" column and the provided "long" and "lat" values, where the distance is less than 1.

Expected result: A list of restaurants within a 1-unit distance from the specified coordinates.

### Query: SELECT borough, AVG(grades[0].score) AS avg\_score FROM restaurants GROUP BY borough;

This query selects the "borough" column and calculates the average score of the first grade for each unique borough. It groups the results by borough.

Expected result: The average score of the first grade for each borough.

## Hard queries

### Query: SELECT borough, cuisine, AVG(grades[0].score) AS avg\_score FROM restaurants WHERE grades[0].score > 5 GROUP BY borough, cuisine ORDER BY borough, cuisine;

This query selects the "borough", "cuisine", and calculates the average score of the first grade for each unique combination of borough and cuisine. It filters out scores lower than 5 and orders the results by borough and cuisine.

Expected result: The average score of the first grade for each cuisine in each borough, excluding scores below 5, sorted by borough and cuisine.