

Projet NFE 204 : Etude de Cassandra comme base NoSQL

Rédigé par : Zeïnab MADOUGOU
Année : 2015-2016

Sommaire

1	Cassandra comme SGBD	3
1.1.	Introduction	3
1.2.	Théorème du CAP.....	3
1.3.	Architecture et stockage physique de données	3
1.3.1.	Vocabulaire pour le stockage des données.....	4
1.3.2.	Organisation	4
1.4.	Installation Cassandra et prise en main.....	4
1.5.	Client DevCenter (Datasax)	5
1.6.	Client cqlsh (Python).....	6
1.7.	Premiers pas avec Cassandra	7
2	Etude : La réplication avec Cassandra	8
2.1.	Cas pratique : SimpleStrategy	8
2.2.	Cas pratique : NetworkTopologyStrategy	8
3	Les données	9
3.1.	Site et export des données	9
3.2.	Import des données dans Cassandra	9
3.3.	Interrogation de la base.....	11
3.4.	Application de la réplication sur les données	12
4	Problématique	13
4.1.	Import des données dans SAS	13
4.2.	Représentation cartographique	15
4.3.	Classification	16
4.4.	Conclusion et difficultés rencontrées	20
5	Annexes.....	21
5.1.	Code SAS.....	21
5.2.	Sources	26

1 Cassandra comme SGBD

1.1. Introduction

Apache Cassandra est un système de gestion de base de données (SGBD) de type NoSQL conçu pour gérer des quantités massives de données sur un grand nombre de serveurs. Il permet une répartition sur plusieurs serveurs, une réplication asynchrone et une architecture de type multi-nœuds pour les serveurs.

Initialement développée par Facebook, l'application a été libérée dans l'espace open source en juillet 2008 ; le projet est Open source et porté par la Fondation Apache.

1.2. Théorème du CAP

Dans le monde des bases de données NoSQL, on entend souvent parler du théorème CAP. Ce théorème établit 3 paramètres sur lesquels on peut jouer pour configurer une base de données distribuée :

- ✓ La cohérence (C pour Consistency)
- ✓ La disponibilité (A pour Availability)
- ✓ La tolérance aux pannes et aux coupures réseaux (P pour Partition-tolerance)

Le théorème postule que pour toute base de données distribuée, on ne peut choisir que 2 de ces 3 paramètres, jamais les 3 en même temps.

Cassandra fait clairement le choix d'AP pour une tolérance aux pannes et une disponibilité absolue. En contrepartie, Cassandra sacrifie la cohérence forte contre une cohérence à terme.

1.3. Architecture et stockage physique de données

L'architecture relationnelle est orientée colonne (stocke les données par colonne et non par ligne), les données sont distribuées selon leur clé primaire. Cette architecture permet de plus une compression par colonne, efficace lorsque les données de la colonne se ressemblent. L'orientation colonne permet d'ajouter des colonnes plus facilement aux tables (les lignes n'ont pas besoin d'être redimensionnées).

Du point de vue de l'application cliente, tous les nœuds sont égaux dans un cluster Cassandra, ce qui signifie qu'il n'y a pas de nœud maître ou un processus centralisant leur gestion.

Le mode de communication utilisé entre les différents nœuds d'un cluster est un protocole appelé Gossip. Cassandra s'en sert afin de découvrir la localisation et les informations sur l'état des autres nœuds du cluster.

1.3.1. Vocabulaire pour le stockage des données

Nœud : serveur où sont stockées les données

Data center : ensemble de plusieurs serveurs de données ; ils peuvent être physiques ou virtuels

Cluster : Un cluster contient un ou plusieurs serveurs.

Commit log : Fichier journal d'écriture des données ; toutes les données sont d'abord écrites dans le journal assurer la réplication.

Table : Une table dans Cassandra a la même signification qu'une table dans le monde SQL, avec des lignes et des colonnes. La terminologie de table est apparue avec CQL3. L'ancien terme pour désigner une table est column family.

Keyspace : Un keyspace peut être vu comme une « base de données ». A l'intérieur de chaque keyspace, on trouve des tables.

1.3.2. Organisation

Dans un cluster Cassandra, on trouve des keyspaces qui contiennent des tables dans lesquelles sont les données. On sépare les données de chaque partie dans les keyspaces. Une table dans Cassandra a la même signification qu'une table dans le monde SQL, avec des lignes et des colonnes.

Les données sont stockées sous forme de lignes et de colonnes comme suit :

Cluster								
Keyspace 1				Keyspace 2				
Table 1				Table 1			Table 2	
ligne		ligne		ligne			ligne	
colonne 1	colonne 2	colonne 1	colonne 2	colonne 1	colonne 2	colonne 3	colonne 1	colonne 2
valeur	valeur	valeur	valeur	valeur	valeur	valeur	valeur	valeur

1.4. Installation Cassandra et prise en main

Via le site Apache (<http://wwwftp.ciril.fr/pub/apache/cassandra/2.2.4/apache-cassandra-2.2.4-bin.tar.gz>), il est possible de télécharger la version 2.2.4 de Cassandra à installer. Le fichier d'installation est une archive .tar à décompresser.

Dans une fenêtre Terminal, se placer dans le répertoire de téléchargements et décompresser avec la commande :

```
tar -xzf apache-cassandra-2.2.4-bin.tar
```

Copie du dossier décompressé dans le répertoire :
/Users/Zeinab/Documents/NFE204/Projet/Cass/

Pour lancer le serveur Cassandra, voici les 2 lignes de commandes à écrire dans une fenêtre Terminal :

```
cd /Users/Zeinab/Documents/NFE204/Projet/Cass/apache-cassandra-2.2.4/bin  
./cassandra
```

La ligne suivante indique que le serveur est prêt à être utilisé sur le poste de travail :

```
INFO 21:25:47 Node localhost/127.0.0.1 state jump to NORMAL
```

Une fois le serveur installé, il faut installer une application cliente pour « dialoguer » avec le serveur.

Il y a entre autres deux possibilités: l'interpréteur de commande cqlsh ou une application graphique Devcenter distribué par Datasax.

1.5. Client DevCenter (Datasax)

DataStax est une entreprise éditrice de logiciels. La société assure la distribution et le support d'une version pour l'entreprise de Cassandra, un projet open-source d'Apache.

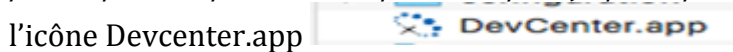
Une fois la version de Devcenter téléchargée sur le site de Datasax (à l'adresse http://downloads.datastax.com/devcenter/DevCenter-1.4.1-macosx-x86_64.tar.gz – choix de la version 1.4.1 pour la version Cassandra 2.2.4).

Décompresser le fichier archivé :

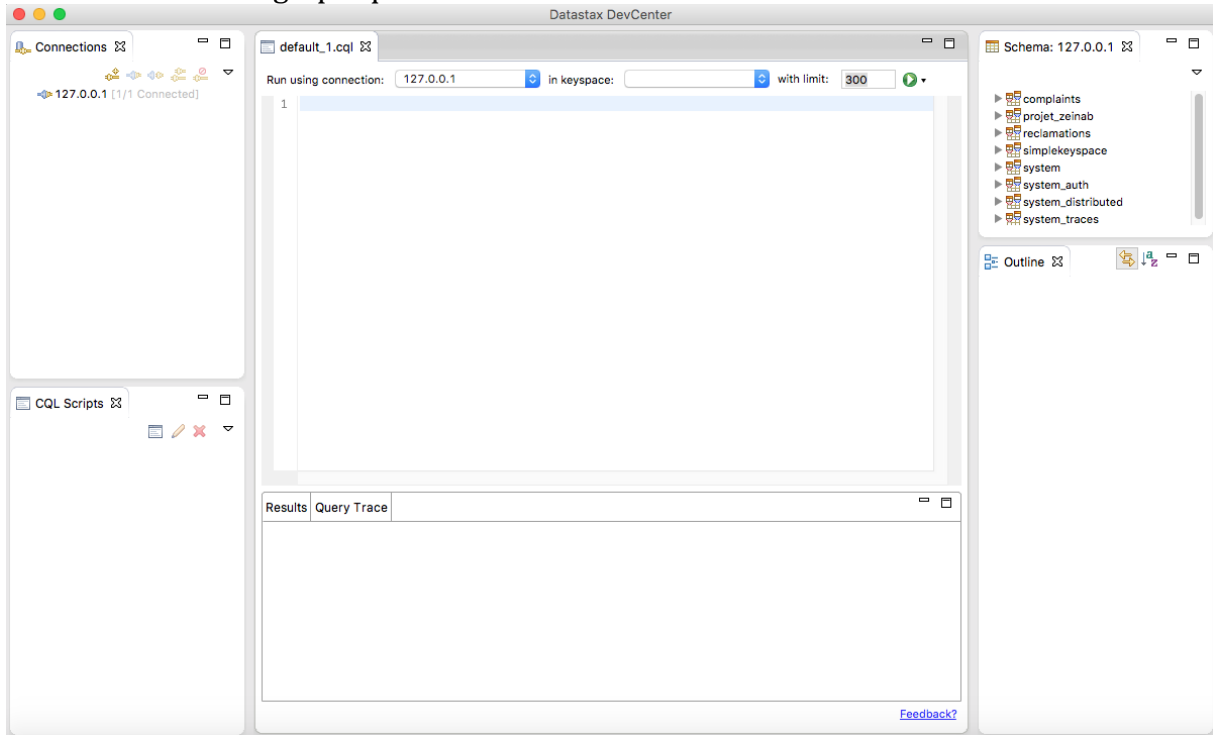
```
tar -xzf DevCenter-1.4.1-macosx-x86_64.tar
```

Copie du dossier décompressé dans le répertoire :
/Users/Zeinab/Documents/NFE204/Projet/Cass/DevCenter

Pour lancer Devcenter, se placer dans le répertoire /Users/Zeinab/Documents/NFE204/Projet/Cass/DevCenter et double cliquer sur l'icône Devcenter.app



On obtient le client graphique suivant :



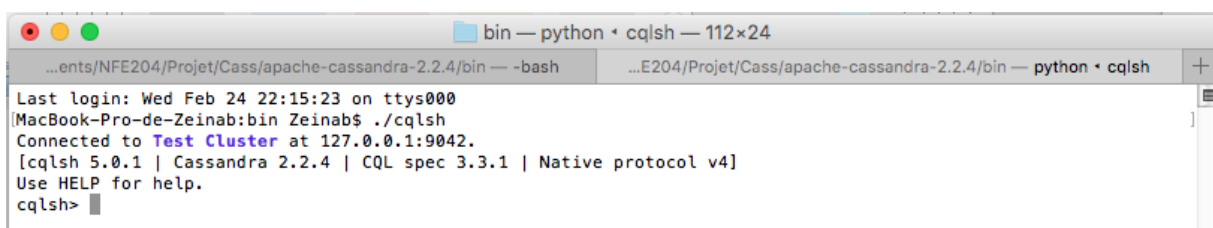
C'est un moyen plus confortable pour les développeurs.

1.6. Client cqlsh (Python)

L'interpréteur de commandes Python se lance comme suit (il est inclus dans le package Apache Cassandra) :

```
cd /Users/Zeinab/Documents/NFE204/Projet/Cass/apache-cassandra-2.2.4/bin
./cqlsh
```

Les lignes suivantes indiquent que le client est opérationnel :



1.7. Premiers pas avec Cassandra

Les lignes de commandes suivantes peuvent être exécutées dans Devcenter (partie centrale), ou dans l'interpréteur de commandes cqlsh.

- ✓ Création d'un keyspace :

```
CREATE KEYSPACE exemplekeyspace WITH REPLICATION = { 'class' :  
'SimpleStrategy', 'replication_factor' : 1 };
```

- ✓ Création d'une table :

```
USE exemplekeyspace;  
CREATE TABLE exempletable ( pk text PRIMARY KEY, col1 text, col2  
text, col3 text);
```

- ✓ Insertion de valeurs :

```
USE exemplekeyspace;  
INSERT INTO exempletable (pk, col1, col2, col3) VALUES ('pk1',  
'col11', 'col21', 'col31');  
INSERT INTO exempletable (pk, col1, col2, col3) VALUES ('pk2',  
'col12', 'col22', 'col32');  
INSERT INTO exempletable (pk, col1, col2, col3) VALUES ('pk3',  
'col13', 'col23', 'col33');  
INSERT INTO exempletable (pk, col1, col2, col3) VALUES ('pk4',  
'col14', 'col24', 'col34');  
INSERT INTO exempletable (pk, col1, col2, col3) VALUES ('pk5',  
'col15', 'col25', 'col35');  
INSERT INTO exempletable (pk, col1, col2, col3) VALUES ('pk6',  
'col16', 'col26', 'col36');
```

- ✓ Requête sur les données :

```
USE exemplekeyspace; SELECT * FROM exempletable;
```

Résultat obtenu:

Results	Query Trace			
pk	col1	col2	col3	
pk1	col11	col21	col31	
pk5	col15	col25	col35	
pk6	col16	col26	col36	
pk4	col14	col24	col34	
pk2	col12	col22	col32	
pk3	col13	col23	col33	

2 statements successfully executed in 301 ms. Retrieved 6 rows [Feedback?](#)

2 Etude : La réplication avec Cassandra

La réplication est le processus permettant de stocker des copies de données sur de multiples nœuds afin de permettre leur fiabilité et la tolérance à la panne. Quand un keyspace est créé dans Cassandra, il lui est affecté la stratégie de distribution des répliques, c'est-à-dire le nombre de répliques et la manière dont ils sont répliqués dans le cluster.

Tous les nœuds de Cassandra sont égaux. Ainsi, une demande de lecture ou d'écriture peut interroger indifféremment n'importe quel nœud du cluster. Quand un client se connecte à un nœud et demande une opération d'écriture ou de lecture, le nœud courant sert de coordinateur du point de vue du client.

Le facteur de réplication est initialisé par `replication_factor` au moment de la création du keyspace. Il désigne le nombre total de répliques dans le cluster ; il peut prendre 2 modalités :

- SimpleStrategy: est utilisé quand on a une seule grappe de serveurs;
- quand il y en a plusieurs, il faut utiliser NetworkTopologyStrategy.

2.1. Cas pratique : SimpleStrategy

On peut changer le nombre de répliques souhaitées dans l'exemple du [1.7](#); pour cela voici la commande testée :

```
ALTER KEYSPACE exemplekeyspace WITH REPLICATION =  
{ 'class' : 'SimpleStrategy', 'replication_factor' : 3 };
```

On obtient la ligne suivante :

```
Parsing ALTER KEYSPACE exemplekeyspace WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor' : 3 } 12:42:01.051000 C
```

Le nombre de réplication (copies des données) passe donc à 3.

2.2. Cas pratique : NetworkTopologyStrategy

Dans le cas de plusieurs grappes de serveurs (2 par exemple), on peut choisir de faire la copie dans ces 2 grappes :

```
ALTER KEYSPACE exemplekeyspace WITH REPLICATION =  
{ 'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 2 };
```


On a en tout 5 réplifications ; 3 dans la grappe 1 et 2 dans la grappe 2.

En utilisant la stratégie de réplification NetworkTopologyStrategy, on peut utiliser l'option DURABLE_WRITES qui peut se passer du journal d'écriture (commitLog) et copie directement les données sur le disque.

```
ALTER KEYSPACE exemplekeyspace WITH REPLICATION =  
    {'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 2} AND  
DURABLE_WRITES = false;
```

3 Les données

<http://www.data.gov/> est le site de l'Open data des Etats Unis.

3.1. Site et export des données

Il s'agit de la base des plaintes enregistrées de consommateurs américains concernant des produits financiers (hypothèque, carte de crédit, recouvrement de créances, etc..). Pour chaque identifiant de plainte, l'objet de la plainte est décrit et la réponse donnée par l'organisme financier aussi ; figurent aussi dans la base, les dates d'envoi et de réception de la plainte, le canal d'envoi de la plainte (fax, téléphone, web, ...), le lieu d'habitation du plaignant (code de l'état à 2 lettres, le zip code). La base a environ 500 000 observations et 16 variables et est téléchargeable sur le lien suivant :

<http://catalog.data.gov/dataset/consumer-complaint-database>

3.2. Import des données dans Cassandra

Pour importer le fichier CSV, j'utilise Python préalablement installé. Je fais le choix de n'importer que les variables dont j'aurais besoin pour la problématique (voir chapitre [4](#)).

Repertoire où se trouve le fichier et script Python:
`/Users/Zeinab/Documents/NFE204/Projet/import_python`

On lance le script python après avoir lancé le serveur Cassandra et créé le « squelette » de la table dans cqlsh :

Création de la table :

```
CREATE KEYSPACE reclamations  
WITH REPLICATION = { 'class' : 'SimpleStrategy',  
'replication_factor' : 1 };
```

```

use reclamations;
create table consumer_complaints ( date_received text, Product
text, state text , ZIP_code text,
submitted_via text , date_sent_to_company text,complaint_ID
int ,
PRIMARY KEY ( Complaint_ID, state));

```

Lancement du script Python :

```

cd /Users/Zeinab/Documents/NFE204/Projet/import_python
python complaints_CSV.py

```

Script Python :

```

""" Import du fichier csv consumerscomplaints
"""

from pprint import pprint
import csv
import os
import cassandra
from cassandra.cluster import Cluster

#Connection to complaints keyspace
cluster = Cluster()
session = cluster.connect('reclamations')

session.execute("""
    truncate table consumer_complaints
""")

csv_data=csv.reader(open('Consumer_Complaints.csv', 'rb'))

for row in csv_data:
    c_complaint_ID = row[15]
    c_date_received = "" + row[0] + ""

    c_date_sent_to_company = "" + row[11] + ""

    c_product = "" + row[1] + ""

    c_issue = "" + row[3] + ""

    c_compagny = "" + row[7] + ""
    c_state = "" + row[8] + ""
    c_ZIP_code = "" + row[9] + ""

```

```

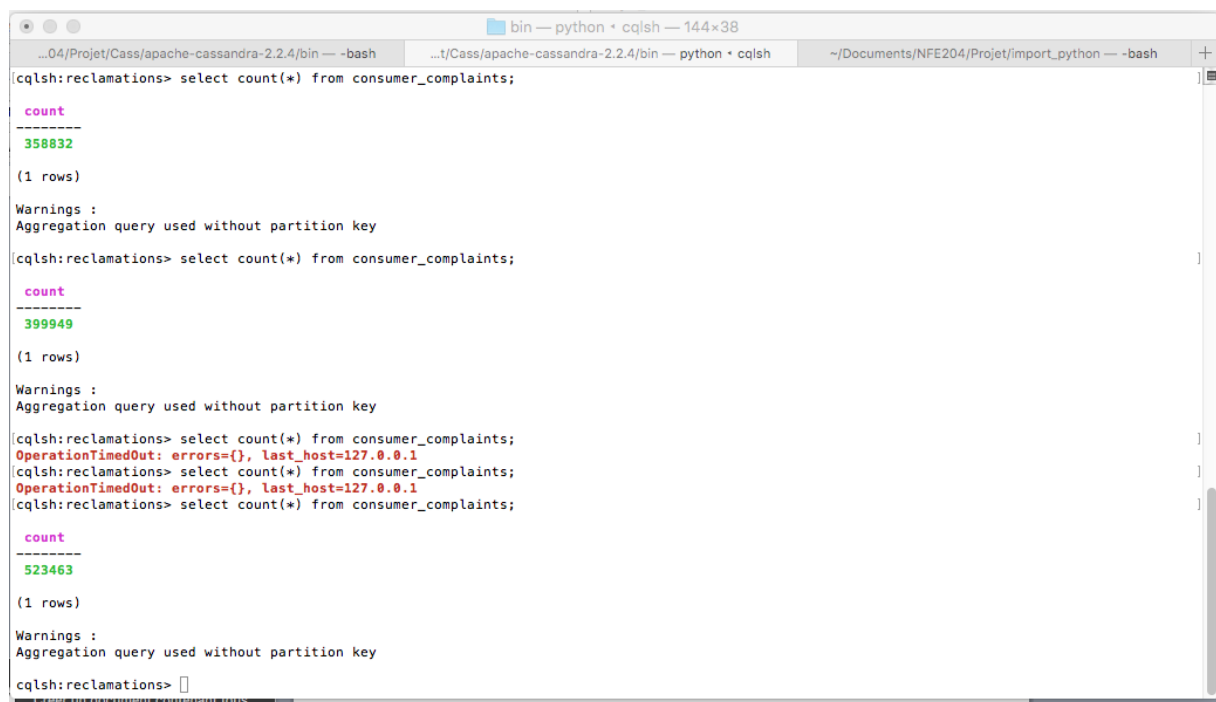
c_submitted_via = "'" + row[10] + "'"

insertQuery =" insert into consumer_complaints (complaint_ID ,
date_received,"
insertQuery += "date_sent_to_company, product, state, "
insertQuery += "ZIP_code, submitted_via )"
insertQuery += "values ( " + c_complaint_ID + ", " +
c_date_received + ", "
insertQuery += c_date_sent_to_company + ", " + c_product + ", "
insertQuery += c_state + ", " + c_ZIP_code + ", "
insertQuery += c_submitted_via + " ) "

print insertQuery
prepared_stmt=session.prepare(insertQuery)
session.execute(prepared_stmt)

```

J'obtiens le résultat suivant en contrôlant que l'insertion des données s'est bien faite :



```

bin — python • cqlsh — 144x38
...04/Projet/Cass/apache-cassandra-2.2.4/bin — -bash
...t/Cass/apache-cassandra-2.2.4/bin — python • cqlsh
~/Documents/NFE204/Projet/import_python — -bash
[cqlsh:reclamations> select count(*) from consumer_complaints;

count
-----
358832
(1 rows)

Warnings :
Aggregation query used without partition key

[cqlsh:reclamations> select count(*) from consumer_complaints;

count
-----
399949
(1 rows)

Warnings :
Aggregation query used without partition key

[cqlsh:reclamations> select count(*) from consumer_complaints;
OperationTimedOut: errors={}, last_host=127.0.0.1
[cqlsh:reclamations> select count(*) from consumer_complaints;
OperationTimedOut: errors={}, last_host=127.0.0.1
[cqlsh:reclamations> select count(*) from consumer_complaints;

count
-----
523463
(1 rows)

Warnings :
Aggregation query used without partition key

cqlsh:reclamations>

```

3.3. Interrogation de la base

Ce qu'il faut savoir avant la construction du modèle, c'est qu'il faut définir les colonnes sur lesquelles on peut faire des recherches ; il y a 2 cas de figures :

- ✓ La variable (colonne) a été préalablement définie comme Primary Key :

```
select * from consumer_complaints limit 10;
```

pour afficher les 10 premières observations de la table

```
select * from consumer_complaints where state = 'NE' allow filtering;
```

- ✓ Un index a été créé sur la variable (colonne) au moment de la création de table, il faut ajouter les lignes « create index on » :

```
CREATE KEYSPACE reclamations
WITH REPLICATION = { 'class' : 'SimpleStrategy',
'replication_factor' : 1 };
use reclamations;
create table consumer_complaints ( date_received text, Product
text, state text , ZIP_code text,
submitted_via text , date_sent_to_company text,complaint_ID
int ,
PRIMARY KEY ( Complaint_ID, state));

CREATE INDEX user_Product ON reclamations.consumer_complaints
(Product);
CREATE INDEX ON reclamations.consumer_complaints
(submitted_via );
```

3.4. Application de la réplication sur les données

Pour mettre en place la réplication il faut modifier la configuration de Cassandra ; pour cela dans le répertoire
/Users/Zeinab/Documents/NFE204/Projet/Cass/apache-cassandra-2.2.4/conf

1. Modification du fichier **cassandra.yaml**
endpoint_snitch: SimpleSnitch --→ GossipingPropertyFileSnitch pour la prise en compte du fichier **cassandra-rackdc.properties**

2. Verifier seed_provider et listen_address du fichier cassandra.yaml :
seeds: "127.0.0.1"
listen_address : localhost → 127.0.0.1

3. Création d'un 2ème serveur en dupliquant le répertoire apache-cassandra-2.2.4

4. Dans les fichiers cassandra-rackdc.properties et cassandra.yaml du 2ème serveur modifier les lignes comme suit, cela pour changer l'adresse IP et les ports sur lesquels pointe le serveur car on est sur une même machine:
dc=dc2

seed : 127.0.0.2
storage_port : 7005
ssl_storage_port : 7006

NB : la 2^{ème} instance se lance à partir du repertoire :
/Users/Zeinab/Documents/NFE204/Projet/Cass/apache-cassandra-2.2.4_2/bin

5. Lancement des 2 serveurs (nœuds)

L'obtention de la dernière ligne après le lancement du 2^{ème} serveur permet de voir que les 2 nœuds sont actifs :

```
INFO 20:33:00 Node /127.0.0.1 state jump to NORMAL
INFO 20:33:00 Updating topology for all endpoints that have changed
```

6. Configuration dans le client cqlsh du premier nœud où se trouvent les données :

```
alter keyspace "reclamations" with replication = {'class' :
'SimpleStrategy', 'replication_factor' : 2};
```

7. Vérification que la réplication s'est bien faite sur le 2^{ème} nœud

Lancement de la requête en lançant le client à partir du 2^{ème} nœud :

```
./cqlsh
Use reclamations ;
select count(*) from consumer_complaints;
```

4 Problématique

Objectif : Analyse statistique de la base avec le logiciel SAS (fourni pour le cours de STA211), pour répondre à la question : les plaintes reçues sont-elles liées au lieu d'habitation ou au mode de réception de la plainte (canal_envoi) ?

1^{ère} étape : Export de la base au format CSV et import dans SAS

2^{ème} étape : Appliquer une classification

3^{ème} étape : Représentation cartographique

4.1. Import des données dans SAS

Export des données avec sélection des colonnes au format CSV :

```
copy consumer_complaints (complaint_id, state, date_received,
product, submitted_via, zip_code) to 'titi.csv' ;
```

La copie du fichier s'est bien faite dans le répertoire suivant :

/Users/Zeinab/Documents/NFE204/Projet/Cass/apache-cassandra-2.2.4/bin

```

cqlsh:reclamations>
cqlsh:reclamations> copy consumer_complaints (complaint_id, state, date_received, product, submitted_via, zip_code) to 'titi.csv' ;

Starting copy of reclamations.consumer_complaints with columns ['complaint_id', 'state', 'date_received', 'product', 'submitted_via', 'zip_code'].
Processed 523463 rows; Written: 14745.209800 rows/s
523463 rows exported in 31.123 seconds.
cqlsh:reclamations> █

```

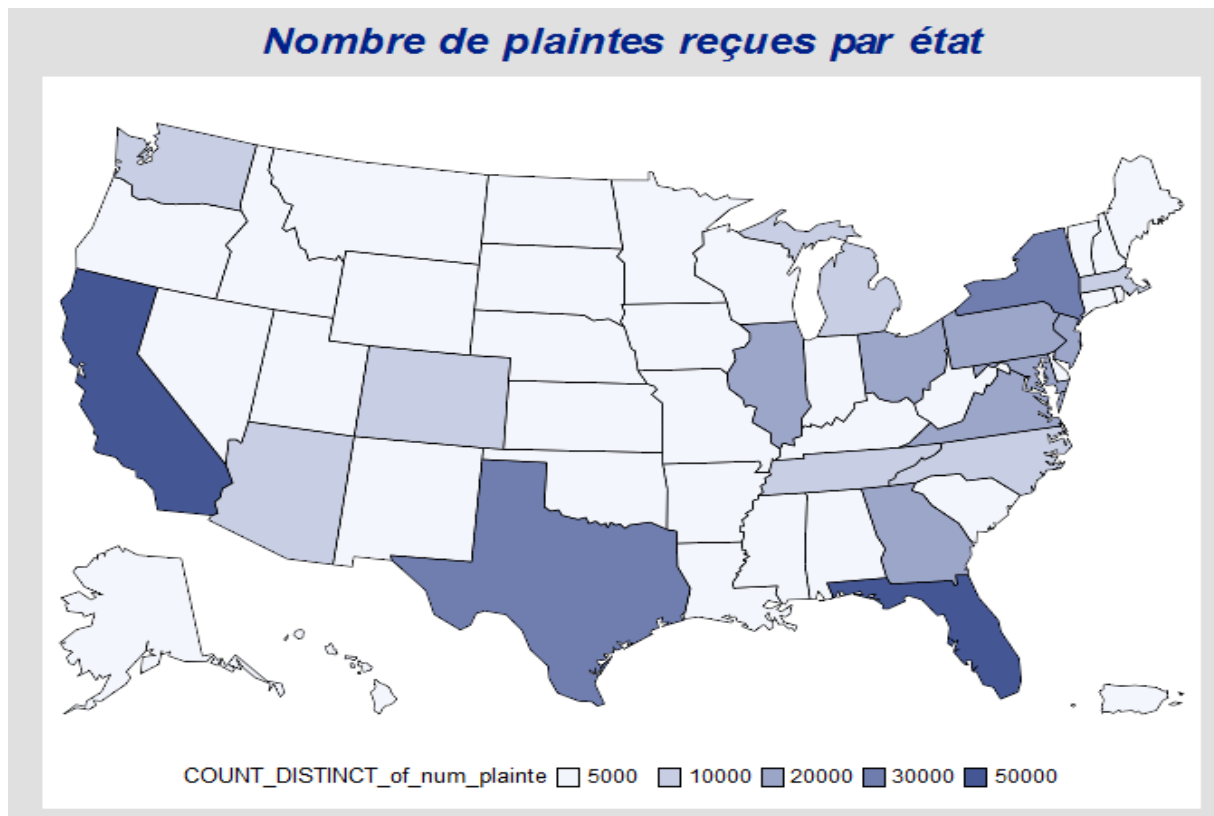
Pour importer les données dans SAS voici le code utilisé :

```

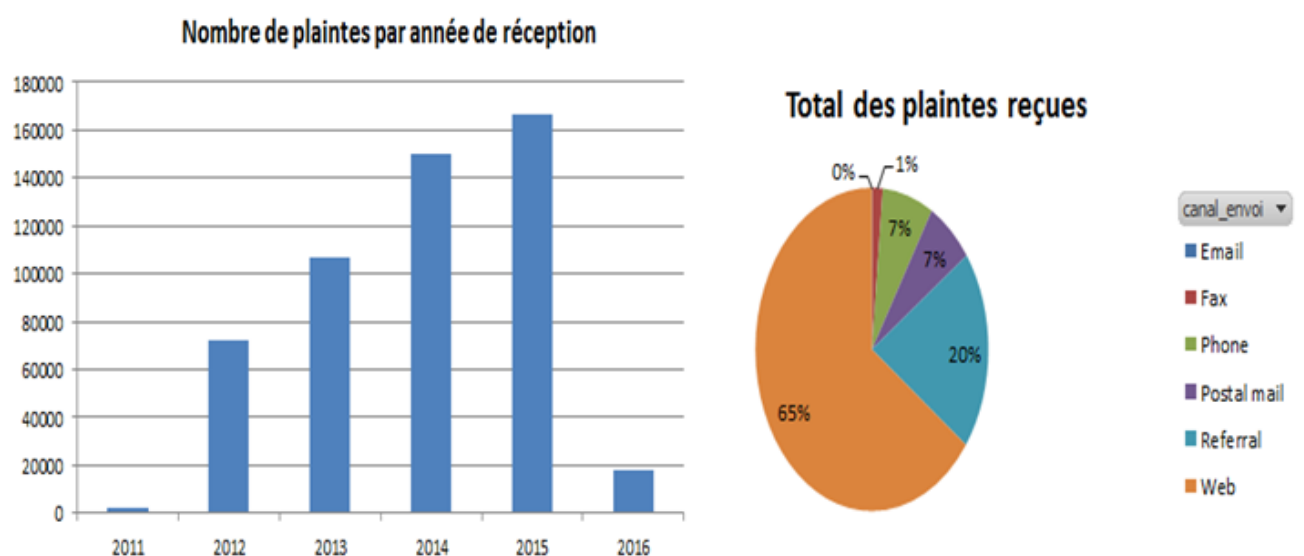
DATA WORK.plaintes;
  LENGTH
    num_plainte      8
    STATECODE        $ 2
    date_reception    8
    produit           $ 23
    canal_envoi       $ 11
    zip_code          $ 5 ;
  FORMAT
    num_plainte       BEST7.
    STATECODE         $CHAR2.
    date_reception    MMDDYY10.
    produit           $CHAR23.
    canal_envoi       $CHAR11.
    zip_code          $CHAR5. ;
  INFILE  '/Users/Zeinab/Documents/NFE204/Projet/Cass/apache-
cassandra-2.2.4/bin/titi.csv'
    LRECL=63
    ENCODING="WLATIN1"
    TERMSTR=CRLF
    DLM='7F'x
    MISSOVER
    DSD ;
  INPUT
    num_plainte
    STATECODE
    date_reception
    produit
    canal_envoi
    zip_code ;
RUN;

```

4.2. Représentation cartographique



A première vue les états dont viennent le plus de plaintes sont ceux de la cote est (à droite), on y trouve des états avec plus de 20 000 plaintes toutes années de réception confondues.



Ces 2 graphiques nous montrent que le principal mode de transmission de la plainte est le Web et que le nombre de plaintes croît avec les ans (on peut se poser la question d'une campagne d'information et de la mise en place du service de réception des plaintes par le gouvernement).

Pour classer les états nous pouvons nous intéresser à l'année 2015 qui est la plus représentative en nombre.

4.3. Classification

La procédure pour appliquer une classification dans SAS est la suivante :

```
PROC CLUSTER DATA=plaintes_3 METHOD=Average;
id STATENAME;
var COUNT DISTINCT of num plainte ;
PROC TREE horizontal out=out ncl=5
RUN ;
proc freq data=out ;table cluster ; run ;
```

Les classes ainsi obtenues sont fonction du nombre de plaintes, cela n'est pas très révélateur sur mode de réception de la plainte, cela équivaut à un tri sur le nombre de plaintes. Pour prendre en compte le mode de réception de la plainte il faut transformer la table de données sous cette forme :

	STATENAME	STATECODE	an_reception	Fax	Phone	Postal mail	Referral	Web	Email
1	Alabama	AL	2015	28	134	82	183	1544	.
2	Alaska	AK	2015	1	14	14	13	122	.
3	Arizona	AZ	2015	37	158	205	486	2928	.
4	Arkansas	AR	2015	6	64	48	98	541	.
5	California	CA	2015	335	1418	1363	2899	17711	1
6	Colorado	CO	2015	19	116	118	291	2301	.
7	Connecticut	CT	2015	18	113	87	315	1248	.
8	Delaware	DE	2015	43	64	30	108	628	.
9	District of Columbia	DC	2015	4	35	43	105	785	.
10	Florida	FL	2015	196	913	904	2183	11673	1

Pour l'année 2015 et pour chaque état nous obtenons après transformation (voir code SAS dans l'[annexe](#)) les variables listant le mode de réception de la plainte. Pour la 1^{ère} ligne l'Alabama a reçu 28 plaintes par Fax en 2015, 134 plaintes par téléphone, etc...

Avec cette base de travail ainsi obtenue, appliquons la classification :

```
PROC CLUSTER DATA=plaintes_6_bis METHOD=Average;
id statename;
var SUM_of_Fax1 SUM_of_Phone 'SUM_of_Postal mail'n SUM_of_Referral
SUM_of_Web SUM_of_Email1;
PROC TREE horizontal out=out ncl=5 ; /* choix du nombre de
classes =5,
sortie des classes et les des statecode qui y sont dans la table
out*/
RUN ;
```



```
proc freq data=out ;
table cluster;
run ;
```

La répartition des états en 5 classes est la suivante (le choix des 5 classes a été à partir de la représentation de l'arbre de classification) :

CLUSTER	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	40	76,92	40	76,92
2	8	15,38	48	92,31
3	2	3,85	50	96,15
4	1	1,92	51	98,08
5	1	1,92	52	100

CLUSTER	N Obs	Mean	Minimum	Maximum	N
1	40	1 369	151	3 974	40
2	8	5 904	4 805	7 527	8
3	2	14 570	13 270	15 870	2
4	1	11 524	11 524	11 524	1
5	1	23 727	23 727	23 727	1

40 états se retrouvent dans la classe 1 et équivalent à 77% des états.

Etudions dans les caractéristiques des classes (les 2 ont chacune une observation - état) :

- ✓ Le nombre moyen de plaintes reçus de la classe 1 est de 1369 plaintes ; dans cette classe on retrouve les 40 premiers états en triant par nombre de plaintes croissant (voir graphique page 18), le canal d'envoi de la plainte ou la situation géographique n'ont pas eu d'incidence les états regroupés dans cette classe.
- ✓ Mêmes remarques pour la classe 2, on y retrouve les 8 états qui suivent les états de la classe 1.
- ✓ La classe 4 a une seule observation qui est l'état de New York, qui a pour particularité d'avoir reçu 67 % de ces plaintes par le Web , même si New York a reçu moins de plaintes que les états de la classe 3 (14 570 plaintes reçus en moyennes en Floride et au Texas). On peut donc dire que la classification des états tient donc compte du canal d'envoi de la plainte et non du caractère géographique.
- ✓ La classe 3 a 2 éléments, les états de Floride et du Texas, pour lesquels respectivement 74% (pour un total de 15 870 plaintes) et 80%(pour un total de 13 270) de leurs plaintes ont été reçues par le Web.

- ✓ La classe 5 a un unique élément qui est l'état de Californie pour lequel 75% des plaintes ont été reçues par le Web, cet état a reçu au total 23 727 plaintes.

Carte des Etats Unis avec le nom des états

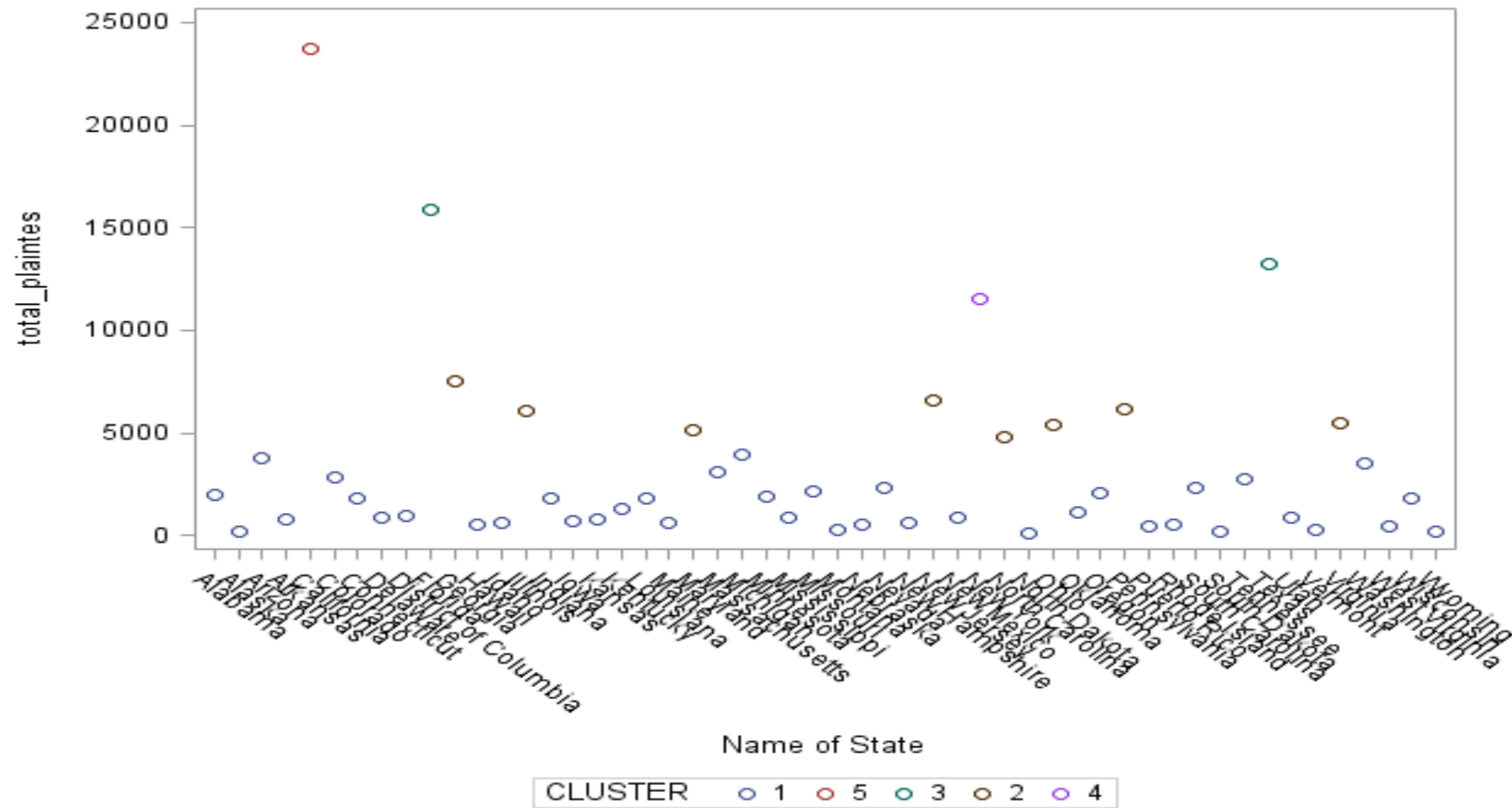


http://fr.cdn.v5.futura-sciences.com/builds/images/rte/RTEmagicC_9877_image003_02_txdam25216_9dd4e4.png

Représentons le nuage de points (états) obtenus après la classification avec le code suivant :

```
/* graphique nuage de points *****/
PROC sgPLOT DATA=classes;
scatter y = total_plaintes x = STATENAME / group=cluster ;
title2 "";
RUN; QUIT;
```

Représentation des 52 états, les couleurs sont fonction de leur classe d'appartenance :



4.4. Conclusion et difficultés rencontrées

La conclusion de l'étude de classification est que la classification des états tient donc compte du canal d'envoi de la plainte et du nombre de plaintes reçues par état.

Difficultés rencontrées : ma première idée était de synthétiser les données par état (faire « un group by » pour avoir le nombre de plaintes reçues par -état, cannal_envoi- et ensuite exporter cette table). Je n'ai pas pu faire le group by sur Cassandra et je n'ai pas pu copier les résultats d'une requête dans une table.

D'autre part je n'ai pas réussi à afficher le nom des états sur la représentation cartographique dans SAS.

5 Annexes

5.1. Code SAS

```
/* -----  
-----  
IMPORT DES DONNEES DANS SAS  
-----  
----- */  
  
DATA WORK.plaintes;  
  LENGTH  
    num_plainte      8  
    STATECODE        $ 2  
    date_reception    8  
    produit           $ 23  
    canal_envoi       $ 11  
    zip_code          $ 5 ;  
  FORMAT  
    num_plainte       BEST7.  
    STATECODE         $CHAR2.  
    date_reception    MMDDYY10.  
    produit           $CHAR23.  
    canal_envoi       $CHAR11.  
    zip_code          $CHAR5. ;  
  
  INFILE '/Users/Zeinab/Documents/NFE204/Projet/Cass/apache-  
cassandra-2.2.4/bin/titi.csv'  
    LRECL=63  
    ENCODING="WLATIN1"  
    TERMSTR=CRLF  
    DLM='7F'x  
    MISSEVER  
    DSD ;  
  INPUT  
    num_plainte  
    STATECODE  
    date_reception  
    produit  
    canal_envoi  
    zip_code          ;  
  
RUN;  
/* -----  
-----  
-----  
----- */  
  
PROC SQL;  
  CREATE TABLE WORK.PLAINTES_2 AS
```

```

SELECT distinct t1.STATECODE,
    /* somme des plaintes reçues par état */
    (COUNT(DISTINCT(t1.num_plainte))) AS
COUNT_DISTINCT_of_num_plainte
FROM WORK.PLAINTES t1
GROUP BY t1.STATECODE
order BY t1.STATECODE;
QUIT;

/* Il y a 52 états au usa , or j'obtiens 63 observations dans
notre table PLAINTES_2.
Il y a peut etre des "erreurs de saisies", choix de supprmer
ces lignes, en passant par
une comparaison avec la table MAPS.US2 fournie par SAS (elle a
bien 52 lignes) */

proc sort data=MAPS.US2 out= us2;
by STATECODE;
run;

data plaintes_3;
merge PLAINTES_2(in=a) us2 (in=b);
by STATECODE;
if a and b;
run;

/* -----
-----
REPRESENTATION CARTOGRAPHIQUE
-----
----- */

proc gmap data = plaintes_3 /*map=maps.us2*/;
id _MAP_GEOMETRY_;
choro COUNT_DISTINCT_of_num_plainte /discrete legend=legend1
midpoints= 5000 10000 20000 30000 50000;
/*tranches des nombres de plaintes , couleurs différentes
pour les tranches: 0-5000; 5001-10 000;
10 001-20 000; 20 001 - 30 000; 30 001 - 50 000; plus de 50
000*/
title 'Nombre de plaintes reçues par état';
run;
quit;

/* -----
-----
CLASSIFICATION

```

```

----- */

PROC CLUSTER DATA=plaintes_3 METHOD=Average;
id STATENAME;
var COUNT_DISTINCT_of_num_plainte ;
  PROC TREE horizontal out=out ncl=5 ; /* choix du nombre de
classes =5,
sortie des classes et les des statecode qui y sont dans la
table out*/
  RUN ;
proc freq data=out ;
table cluster ;
run ;

/*transformation de la variable canal_envoi pour l'utiliser
dans la classification*/
/* group sur le statecode et le canal d'envoi */
proc sort data=MAPS.US2 out= us2;
by STATECODE;
run;
PROC SQL;
  CREATE TABLE WORK.PLAINTES_2bis AS
  SELECT t1.STATECODE,
         t1.canal_envoi, year(date_reception) as
an_reception,
         (COUNT(DISTINCT(t1.num_plainte))) AS
COUNT_DISTINCT_of_num_plainte
  FROM WORK.PLAINTES t1
  GROUP BY t1.STATECODE,
           t1.canal_envoi, an_reception;
QUIT;

/*suppression des lignes avec state vide ou mal codifiés*/
data plaintes_3bis;
merge PLAINTES_2bis (in=a) us2 (in=b);
by STATECODE;
if a and b;
run;
data plaintes_4 (where=(an_reception= 2015));
set plaintes_3bis;
/*format num_canal_envoi 8. ;
if canal_envoi='Email' then num_canal_envoi= 1;
if canal_envoi='Fax' then num_canal_envoi= 2;
if canal_envoi='Phone' then num_canal_envoi= 3;
if canal_envoi='Postal mail' then num_canal_envoi= 4;
if canal_envoi='Referral' then num_canal_envoi= 5;
if canal_envoi='Web' then num_canal_envoi= 6;*/
run;

```

```

/*****/
proc sort data= plaintes_4;
by statename statecode an_reception ;;
run;

proc transpose data= plaintes_4 out=plaintes_5(drop= _name_
_label_);
id canal_envoi;
by statename statecode an_reception;;
run;

/* pour avoir 52 lignes (états)*/

proc sort data=plaintes_5 out= plaintes_6 nodupkey;
by statename statecode an_reception;;
run;

/*remplacement des valeurs vides par 0 pour les variables
concernées car sinon,
elles seront considérées comme observations manquantes*/

data plaintes_6_bis /*(drop=SUM_of_Phone SUM_of_Email )*/;
set plaintes_6;
if Email =. then Email1=0;
if Email ne . then Email1=Email ;

if Fax =. then Fax1=0;
if Fax ne . then Fax1=Fax ;
run;

/* application de la classification sur la table obtenue */

PROC CLUSTER DATA=plaintes_6_bis METHOD=Average;
id statename;
var Fax1 Phone 'Postal mail'n Referral Web Email1;
PROC TREE horizontal out=out ncl=5 ; /* choix du nombre de
classes =5,
sortie des classes et les des statecode qui y sont dans la
table out*/
RUN ;
proc freq data=out ;
table cluster ;
run ;

/* jointure pour avoir l'info de la classe d'appartenante de
chaque état, pour la représentation du nuage de points aussi*/
PROC SQL;
CREATE TABLE WORK.classes AS
SELECT t1.STATENAME,

```



```

t1.STATECODE,
t1.an_reception,
t1.Fax,
t1.Phone,
t1.'Postal mail'n,
t1.Referral,
t1.Web,
t1.Email,
t1.Email1,
t1.Fax1,
t2.CLUSTER,
t2.CLUSNAME,
/* total_plaintes */
(Fax1 + Phone + 'Postal mail'n + Referral + Web +
Email1)
        LABEL="total_plaintes" AS total_plaintes
FROM WORK.PLAINTES_6_BIS t1
INNER JOIN WORK.OUT t2 ON (t1.STATENAME =
t2._NAME_);
QUIT;

/* ETUDES DES CLASSES *****/

PROC MEANS DATA= classes
FW=12 PRINTALLTYPES CHARTYPE NWAY MEAN MIN MAX N ;
VAR total_plaintes;
CLASS CLUSTER /ASCENDING;

RUN;

/* graphique nuage de points *****/

PROC sgPLOT DATA=classes;
scatter y = total_plaintes x = STATENAME / group=cluster ;
title2 "";
RUN; QUIT;

```



tableaux et sorties
SAS.xlsx

5.2. Sources

[https://fr.wikipedia.org/wiki/Cassandra_\(base_de_donn%C3%A9es\)](https://fr.wikipedia.org/wiki/Cassandra_(base_de_donn%C3%A9es))
<http://www.infoq.com/fr/articles/modele-stockage-physique-cassandra>
<http://soat.developpez.com/articles/cassandra/>
<https://docs.datastax.com/en/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html>
<http://mbaron.developpez.com/tutoriels/nosql/cassandra/installation-outils-administration/>
https://docs.datastax.com/en/cassandra/2.0/cassandra/operations/ops_add_node_to_cluster_t.html
<http://www.prowareness.com/blog/replication-and-snitches-in-cassandra/>
Cassandra the definitive guide - Eben Hewitt (Editions O'Reilly)