



**UNIVERSITE DE PARIS 8 SAINT-DENIS**

**UFR STN – MASTER 1 MATHÉMATIQUES**

**PARCOURS CYBERSECURITE ET SCIENCES DES DONNÉES**

**COURS DE BASE DE DONNÉES REPARTIES**

## **Projet NoSQL: Apache CASSANDRA**

**ENCADRE PAR L. BOUBCHIR**

**NOVEMBRE 2020**

**RALAINARIVO  
N. T. THIERRY  
17807592**

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Description de la technologie:</b>	<b>2</b>
2.1	Architecture et définitions: . . . . .	3
2.2	Comprendre le cluster Cassandra: . . . . .	3
2.3	Quand utiliser Cassandra: . . . . .	4
<b>3</b>	<b>Installation:</b>	<b>4</b>
3.1	Télécharger et installer java 1.8.0: . . . . .	4
3.2	Télécharger et installer python 2.7 . . . . .	5
3.3	Installer Cassandra Apache 3.0.9 . . . . .	5
<b>4</b>	<b>Cassandra CQLSH: création de la base de données et requêtes</b>	<b>7</b>
4.1	Cassandra Shell Commands . . . . .	8
4.2	les "data definition commands": . . . . .	8
4.3	CQL data manipulation commands: . . . . .	9
4.4	Cql clauses . . . . .	9
<b>5</b>	<b>Les bases de commandes crud et requêtes cql: Exercices types</b>	<b>10</b>
5.1	Create keyspace . . . . .	10
5.2	Create table and column . . . . .	10
5.3	Manipulation des données . . . . .	12
<b>6</b>	<b>Conclusion</b>	<b>16</b>

# NoSQL\_Cassandra

Projet de documentation sur Cassandra

November 2020

## 1 Introduction

Cassandra est un système de gestion de base de données (SGBD) créé chez Facebook en interne. Il a été conçu pour traiter la masse de données générée par le groupe en temps réel. Cassandra est capable de gérer des **petabytes d'informations et des milliers d'opérations par secondes**. Sa force tient de son système de réplication des données sur des datacenters sans tenir compte de la limite géographique. C'est sa spécificité.

Et puis Facebook est passé sur un autre SGBD. En 2008, le groupe a donné Cassandra à la fondation Apache en tant que projet opensource pour être géré finalement par Datastax jusqu'à aujourd'hui. Les documentations actuelles ont été faites par Datastax. A ce jour, Cassandra est utilisé entre autre par Netflix et Spotify.

Dans la suite, pour des raisons pratiques et pour éviter une traduction erronée des termes utilisées, il est préférable de garder certains termes en *anglais*.

## 2 Description de la technologie:

D'abord, Cassandra est un système de gestion de base de données non relationnel (NoSQL) orienté COLONNE. Il est distribué, décentralisé, scalable horizontalement et tolérant à la panne. En d'autres termes, ce système gère des données non relationnelles qui sont représentées sous forme de colonnes et de lignes. Le concept de la distribution permet de répartir les charges de travail sur plusieurs serveurs locaux ou non. Ensuite, en étant décentralisé, ces données sont copiées ou "répliquées" entre les serveurs. Cela rend Cassandra insensible à la panne car les données restent accessibles à tout moment. Il est important de noter aussi qu'entre les "nodes" et les "clusters", leurs rôles sont identiques, il n'y a pas de relation maître-esclave entre eux.

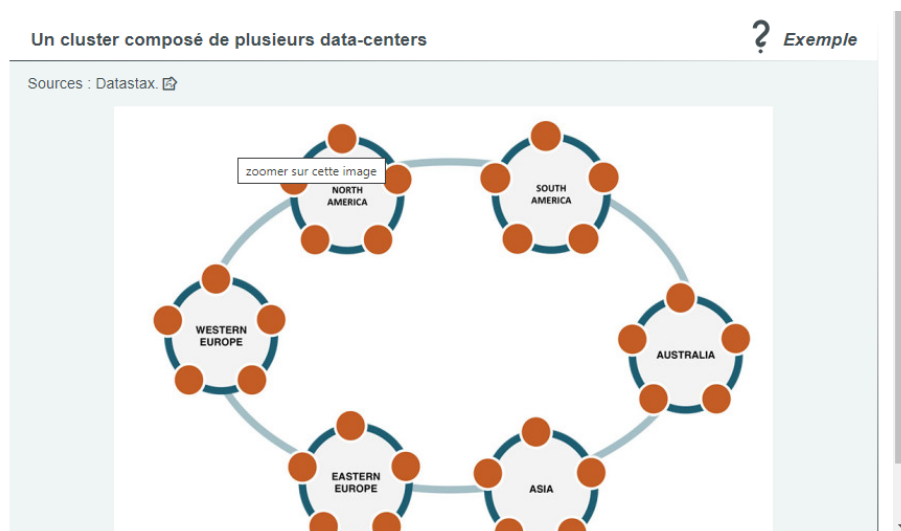
## 2.1 Architecture et définitions:

Nous allons définir les éléments essentiels qui constituent tout le système de Cassandra:

- un node: un noeud, c'est un serveur. Les données sont stockées ici.
- un datacenter: c'est un ensemble de nodes ou de serveurs qui sont dans un environnement géographique proche.
- un cluster: est un ensemble de datacenters. C'est un ensemble de serveur constitué de 2 serveurs au minimum. En général, on définit un seul keyspace par cluster.
- Un keyspace est le niveau le plus élevé des données en tant qu'objet. C'est lui qui contrôle la fonction de réplication des objets qu'il contient au sein de chaque datacenter du cluster.
- replication factor (**RF**): La replication factor correspond au nombre de copies d'une données à travers les serveurs du cluster.
- le commit log: c'est le mécanisme de récupération en cas de panne. Les pannes sont détectées mais cela n'empêche pas la machine de tourner.

## 2.2 Comprendre le cluster Cassandra:

Cassandra fonctionne selon le théorème **CAP** qui consiste à la cohérence (Consistency), la disponibilité (Availability) et la résistance au morcellement (Partition tolerance) des données.



Un cluster peut contenir plusieurs keyspaces. Chaque keyspace est paramétré de façon indépendante. Les paramètres sont le **replicant factor (RF)** et le **consistency level (CL)**.

Ces derniers assurent la gestion de la disponibilité ainsi que de la résistance au morcellement de Cassandra.

## 2.3 Quand utiliser Cassandra:

Il est pertinent d'utiliser Cassandra pour un système dont les critères sont les suivants:

1. décentralisé: la base de données doit être distribuée c'est à dire les données ne sont pas centralisés sur un seul et même serveur. Il y aura donc la présence d'un cluster pour gérer la scalabilité du système. Une évolution des performances est possibles si nécessaire en ajoutant le nombre de serveurs adéquat.
2. Tolérant aux pannes: l'accès aux données doit être garanti même en cas de pannes. La présence de panne n'est pas décisif pour l'intégrité des données.
3. Hautement disponible: la base de données reste disponible par n'importe quel utilisateur à n'importe quel moment même en cas de pics d'utilisations.

## 3 Installation:

Avant de pouvoir utiliser Cassandra, il est nécessaire de vérifier qu'on a **java** et **python** sur notre machine.

Dans ce qui suit, je vais installer Cassandra sur Windows 10. Pour cela, on va d'abord télécharger et installer java correctement, puis on va faire de même avec Python et enfin nous allons installer Cassandra.

### 3.1 Télécharger et installer java 1.8.0:

On va sur le site dédié à java:

[https://www.java.com/fr/download/help/java\\_win64bit.html](https://www.java.com/fr/download/help/java_win64bit.html)

On choisi la configuration Windows qui correspond à notre machine. Il est préférable de télécharger l'exécutable ".exe" pour une installation plus rapide:

1. on lance l'exécutable .exe
2. on ouvre le dossier qui le contient
3. on voit qu'on a un dossier jdk et jre.
4. on ajoute java dans *l'environnement variable de notre machine*:  
on va dans paramètre de configuration, système puis systèmes avancés, environnement variable: on crée une nouvelle variable qui s'appellera "JAVA\_HOME" avec le chemin vers le dossier jre1.8.0.

5. Et puis dans "variable système", cliquer sur "path", on ajoute le chemin vers le "jdk1.8.0/bin". Et on valide.
6. on vérifie si on a fait l'installation correctement:  
Par l'invite de commande CMD, on lance:  
\$echo \$JAVA\_HOME\$ ⇒ nous donne la destination du dossier jre1.8.0.  
\$java -version ⇒ nous donne la version de java qu'on a installé.
7. Java 8 est bien installée si la commande nous donne la version de java utilisé.

### 3.2 Télécharger et installer python 2.7

1. on va dans [python.org/download](https://python.org/download).
2. on choisi le "msi installer 64-bits", on le télécharge et on lance l'installation.
3. on choisi le dossier de la destination pour l'installer.
4. On va de nouveau dans "environnement variable", on ajoute le chemin du dossier python dans le "path". Ici python27
5. on y ajoute aussi le chemin vers python27/scripts.
6. on vérifie avec l'invite de commande: \$python --version
7. conclusion, python est bien installé si cela nous renvoie Python 2.7.

```
C:\Users\trala>echo $JAVA_HOME%
$JAVA_HOME%

C:\Users\trala>echo %JAVA_HOME%
C:\Program Files\Java\jre1.8.0_271

C:\Users\trala>java -version
java version "1.8.0_271"
Java(TM) SE Runtime Environment (build 1.8.0_271-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.271-b09, mixed mode)

C:\Users\trala>python --version
Python 2.7.17
```

### 3.3 Installer Cassandra Apache 3.0.9

1. on va dans <https://cassandra.apache.org/download/>
2. on choisi la version apache-cassandra-3.0.23, on clic sur un lien miroir proposé.

3. on télécharge le dossier ".bin.tar.gz".
4. On dé-"tar" le dossier, puis on choisi la destination de notre dossier "apache-cassandra" que j'ai choisi de mettre dans C:/ pour eviter de s'authentifier en tant qu'administrateur.
5. on va dans "environnement variable", on crée une nouvelle variable "CASSANDRA\_HOME" avec le chemin vers le dossier "apache-cassandra". Et dans "path", on ajoute le chemin vers le apache-cassandra/bin.

Maintenant on peut lancer correctement Cassandra:

On lance le serveur:

1. on va dans powershell
2. on va dans le dossier "bin" de cassandra
3. on lance: cassandra.bat -f

```

Windows PowerShell
3668424552, -4695174573516287153, -4698574133258275884, -4730750106806161621, -4870832960446678714, -4909511874742066260,
-5002113143076950452, -5083354825744415742, -513044572433611240, -5170823871172493847, -5182153440875351367, -52001980
2161072109, -521233281736863357, -5344639902787582148, -5396072730056503553, -5431507745991448346, -5502028276317072695
, -565785411322969030, -5696076622235598797, -5858826871245825602, -587628470351468302, -5983816075322403147, -59882000
34555658170, -6012973869851278445, -6024699807024376276, -6048553193109686654, -606719301624535576, -6264927948666195436
, -6298158236071757956, -635153899018116583, -635546880767268691, -6462624041850823327, -651086183290888091, -6519011
110934814505, -6525390615940509496, -6649721792388001461, -6682768078909241390, -6728033953908114426, -68024386489967580
9, -6804475556197809202, -6845591685877106724, -6864161301132353789, -6961976325593414323, -707158920734051067, -7243083
570954693567, -7243272803967102057, -7444064234507648757, -7506167987312454514, -7625682106267553244, -77217818287851473
55, -773506734333181202, -7955615792870201443, -8008788719933738001, -8098377000644668897, -8178908436002372684, -83178
44833072507639, -8319333679060358402, -841569771881834930, -8416427525793153470, -8487705679375740779, -8573616634846793
509, -8648670045293830082, -8675994196082919856, -8704944550279749133, -8723144653247731722, -8739632155558751919, -8906
390119924236001, -8909266543697037281, -8980073189310305670, -8993986997517993813, -9039065787081403364, -92822842503548
2870, -983309409541294453, 1006914343447680310, 1057143800329059362, 120456135425364389, 1275108964272615828, 1330728481
545565265, 1366765844491424522, 148804789072357335, 1612516222887232205, 1751415325261605993, 1769882647198169698, 17723
7181293410491, 1797043861726330229, 184000396157183442, 1862795024670047142, 1933635251177537934, 1980132701059993861, 1
992172486369645649, 2039969295705871056, 2142325617147180630, 2219626373742155074, 2235775637168778111, 2262534841893581
752, 2290417486961969095, 23100730492289225, 2316721752582504445, 2366130138745109693, 2463710388046830987, 266774576217
100980, 2684386708565363493, 2793024385245185881, 2840059121771380821, 2925215555539674601, 3051336960945041015, 3070926
219089102331, 3072944137511365233, 3247342892427458023, 328121174063519483, 3427741077614262261, 3482122625399430808, 34
933064593709660, 352437635961427564, 3644720770225620081, 3718713687146711900, 4326743108832809457, 44100836421030728
77, 4413886262656543845, 4435454888914124030, 4524629999389111563, 469057203362613728, 4737789630594920312, 47923109184
24659074, 483299899853135103, 4893545112159612452, 5006107808006869420, 5024457391623066369, 5051751097044601620, 51285
89462246748065, 5146933971635437373, 5243356912052125765, 5269937327744497509, 5500502667541252038, 5501098007232295134,
5596121474926842796, 5623420854064843125, 5926990594930982975, 6084743911532639252, 6132648621851219113, 61461856647904
7899, 6207149258599289150, 6287792804643684425, 6413360842109982666, 6420347586045280909, 6494281506075599422, 663098284
2416859709, 6737665690685362793, 6760467071455163012, 6786605106908857687, 6862803450927820474, 6888602062528753327, 706
8415860171432448, 7147493716307916333, 7250003525318558734, 7376557198575073730, 7445751023378223430, 759825783533178299
8, 7617476067706311804, 7631025147733268675, 7676715358814066615, 7682017749158517666, 7738547069118519306, 781007079717
0879158, 783853595987955748, 7899864591011148526, 7938459896577583806, 7939051449933045888, 7975364517371817476, 8030802
458637039904, 8096896900090813776, 8125547660752323729, 8220299952495111742, 8220551973907085927, 8266169305765969659, 8
319813962937904849, 8352365491821930969, 8398138147064766184, 8450518680727644928, 8546324062465457442, 8604032355013655
557, 8960710789709804915, 8977938540696378523, 9052927689431602474, 9069600041131783733, 934883906496451337, 99082195529
88851911]
INFO 15:03:40 Node localhost/127.0.0.1 state jump to NORMAL
INFO 15:03:40 Netty using Java NIO event loop
INFO 15:03:40 Using Netty Version: [netty-buffer-netty-buffer-4.0.44.Final.452812a, netty-codec-netty-codec-4.0.44.Final.452812a, netty-codec-haproxy-netty-codec-haproxy-4.0.44.Final.452812a, netty-codec-http-netty-codec-http-4.0.44.Final.452812a, netty-codec-socks-netty-codec-socks-4.0.44.Final.452812a, netty-common-netty-common-4.0.44.Final.452812a, netty-handler-netty-handler-4.0.44.Final.452812a, netty-tcnative-netty-tcnative-1.1.33.Fork26.142ecbb, netty-transport-netty-transport-4.0.44.Final.452812a, netty-transport-native-epoll-netty-transport-native-epoll-4.0.44.Final.452812a, netty-transport-rxtx-netty-transport-rxtx-4.0.44.Final.452812a, netty-transport-udt-netty-transport-udt-4.0.44.Final.452812a]
INFO 15:03:40 Starting listening for CQL clients on localhost/127.0.0.1:9042 (unencrypted)...
INFO 15:03:41 Not starting RPC server as requested. Use JMX (StorageService->startRPCServer()) or nodetool (enablethrif
t) to start it
INFO 15:03:41 Startup complete

```

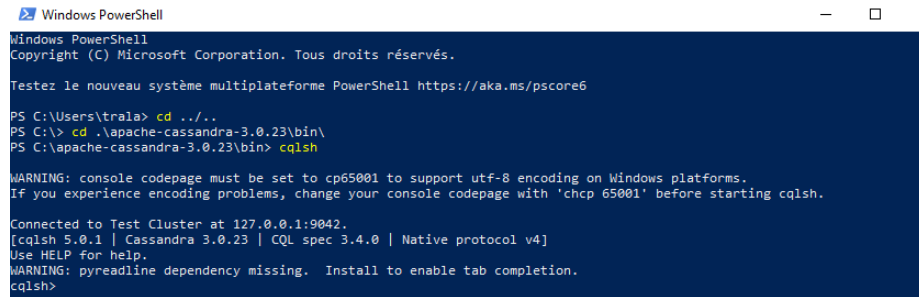
4. le serveur est lancé correctement

Pour ma part, j'ai eu beaucoup de probleme pour lancer ce serveur Cassandra car un problème de compatibilité avec une autre version de java est apparu.

Aussi, j'ai eu un autre problème avec le module Sigar qui est dans la configuration de Cassandra, conf/cassandra-env.sh. Après plusieurs recherches, j'ai trouvé la solution. Il a fallu que je "mute" l'environnement sigar. Et cela a marché.

Ensuite, on lance l'**interface CQLSH**:

1. on ouvre un autre powershell
2. on va dans le dossier "bin" de Cassandra
3. on lance: cqlsh
4. on a lancé l'interface de commande de Cassandra Query Language. En effet, c'est ici qu'on manipule les objets de la database sur le(s) serveur(s) de Cassandra.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\trala> cd ../..
PS C:\> cd .\apache-cassandra-3.0.23\bin\
PS C:\apache-cassandra-3.0.23\bin> cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.0.23 | CQL spec 3.4.0 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh>
```

5. Lancement réussi.

## 4 Cassandra CQLSH: création de la base de données et requêtes

Une fois l'installation effectuée avec succès, on interagit avec les serveurs grâce à la commande cqlsh.

CQLSH est le "cassandra query language shell" qui permet aux utilisateurs d'intervenir sur les données à tout moment. C'est le langage de l'interface de Cassandra. Le langage sur cqlsh est similaire au langage SQL. Il est d'usage de mettre les requêtes en **majuscules**. Mais on peut aussi mettre les requêtes en **minuscules**.

Pour des raisons pratiques, on va définir quelques notions au préalable:

- table ou column family: est un ensemble de colonnes.
- Row représente la ligne
- Column représente la colonne



- keyspace est un ensemble de table. Les keyspaces sont les bases de données; on peut les créer et les supprimer facilement.
  - une "query" est une requête
- Pour la suite on va garder ces termes en *anglais* pour des raisons de commodités.

## 4.1 Cassandra Shell Commands

Le lancement de cqlsh se vérifie par l'affichage suivante du prompt sur cqlsh.

```
PS C:\Users\trala> cd ../../
PS C:\> cd .\apache-cassandra-3.0.23\bin\
PS C:\apache-cassandra-3.0.23\bin> .\cqlsh

WARNING: console codepage must be set to cp65001 to support utf-8 encoding on Windows platforms.
If you experience encoding problems, change your console codepage with 'chcp 65001' before starting cqlsh.

Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.0.23 | CQL spec 3.4.0 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh>
```

Quelques commandes de base de CQLSH:

- help: pour obtenir l'aide de la documentation cqlsh>HELP;
- describe: donne les information sur le cluster et ses objets.
- Version: affiche la version cassandra utilisé. cqlsh>SHOW VERSION;
- U: pour authentifier l'utilisateur, le nom par défaut est Cassandra cqlsh>U<"user name">;
- Exit: pour quitter l'interface cqlsh
- Nocolor: pour annuler les affichages en couleur. cqlsh;NO COLOR;

Il y a 3 sortes de commandes cql:

## 4.2 les "data definition commands":

Pour les keyspaces, on note qu'il faut toujours définir la clé primaire ou **primary key**. Elle est utilisée pour toutes les query initiale.

- create keyspace: pour créer les keyspaces dans Cassandra
- alter keyspace: changer les propriétés d'une keyspace
- drop keyspace: supprimer une keyspace

Pour créer un objet, il suffit de lancer la commande **USE** *< keyspace\_name >* qui nous intéresse.

Maintenant qu'on a les keyspaces, on va pouvoir créer les objets data pour mettre nos données:

- create table: pour créer une table dans une keyspace
- alter table: modifier la propriétés d'une colonne dans une table
- drop table: supprimer une table
- truncate: pour supprimer toutes les données sur une ligne dans une table
- create index: pour définir un nouveau index dans une colonne de la table. Dans Cassandra, si on veut faire des query sur d'autres colonne que le "primary key", il est obligatoire d'ajouter un **deuxième index** sur la colonne qui nous intéresse.
- drop index: supprime un index.

### 4.3 CQL data manipulation commands:

Voici quelques commandes pour créer des insertions et modifications de données:

- insert...into...: pour ajouter des colonnes sur une ligne dans une même table. on utilise "insert into" pour insérer ou compléter le tableau.
- update: pour modifier une colonne
- delete: pour supprimer des données dans une table
- batch: pour exécuter une multiple commande de data manipulation en même temps.

### 4.4 Cql clauses

Ceux ci ne sont pas des commandes propre à Cassandra:

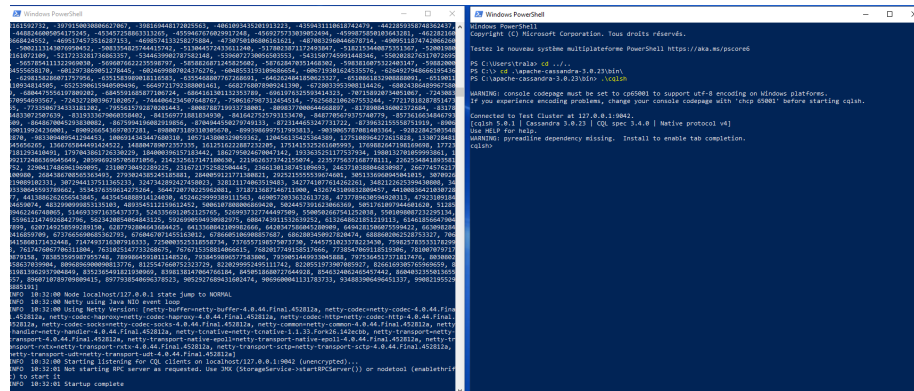
- select: cette "clause" est utilisé pour lire ou selectionner des données d'une "table"
- where: utilisé pour spécifier une donnée dans la column family
- orderby: pour lire des données bien definies dans un ordre specifique.

## 5 Les bases de commandes crud et requêtes cql:

### Exercices types

Dans cette partie , nous allons nous basés sur la documentation officielle de cassandra pour la prise en main. Pour cela on va aller sur le site de Datastax et suivre les étapes proposés dans le tutoriel. Il est important de noter que chaque instruction sera séparée par une point virgule ”;” et qu’il est possible de passer à la ligne si la commande est trop longue.

On lance le serveur et l’interface cqlsh:



Maintenant, on se place dans l’interface CQLSH pour tester les commandes:

### 5.1 Create keyspace

Pour créer la keyspace, on fait:

```
CREATE KEYSPACE tutorialpoint
WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};
```

Pour vérifier que la création a réussi, on utilise la commande:

```
DESCRIBE KEYSPACE tutorialpoint;
```

```
cqlsh> CREATE KEYSPACE tutorialpoint
... WITH replication = {'class': 'SimpleStrategy', 'replication_factor' : 3};
cqlsh> describe tutorialpoint

'tutorialpoint' not found in keyspaces
cqlsh> describe tutorialpoint

CREATE KEYSPACE tutorialpoint WITH replication = {'class': 'SimpleStrategy', 'replication_factor': '3'} AND durable_writes = true;
```

### 5.2 Create table and column

Pour la création de la table: On se met dans la keyspace qui nous intéresse:

```
USE tutorialpoint;
```

puis,

```
CREATE TABLE emp( emp_id int PRIMARY KEY, emp_name text, emp_city
```

text, emp\_sal varint, emp\_phone varint);

on vérifie avec SELECT \* FROM emp;

```
cqlsh> use tutorialspoint;
cqlsh:tutorialspoint> tutorialspoint>; CREATE TABLE emp(
...     emp_id int PRIMARY KEY,
...     emp_name text,
...     emp_city text,
...     emp_sal varint,
...     emp_phone varint
... );
SyntaxException: line 1:0 no viable alternative at input 'tutorialspoint' ([tutorialspoint]...)
SyntaxException: line 7:0 missing ')' at ';'
cqlsh:tutorialspoint> CREATE TABLE emp(
...     emp_id int PRIMARY KEY,
...     emp_name text,
...     emp_city text,
...     emp_sal varint,
...     emp_phone varint
... );
cqlsh:tutorialspoint> select * from emp;

 emp_id | emp_city | emp_name | emp_phone | emp_sal
-----+-----+-----+-----+-----
(0 rows)
cqlsh:tutorialspoint>
```

### 5.3 Manipulation des données

- Create data:

```
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal)
VALUES(1,'ram', 'Hyderabad', 9848022338, 50000);
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal)
VALUES(2,'robin', 'Hyderabad', 9848022339, 40000);
INSERT INTO emp (emp_id, emp_name, emp_city, emp_phone, emp_sal)
VALUES(3,'rahman', 'Chennai', 9848022330, 45000);
```

Lire tous les données: `SELECT * FROM emp;`

```
cqlsh:tutorialspoint> INSERT INTO emp (emp_id, emp_name, emp_city,
... emp_phone, emp_sal) VALUES(1,'ram', 'Hyderabad', 9848022338, 50000);
cqlsh:tutorialspoint> INSERT INTO emp (emp_id, emp_name, emp_city,
... emp_phone, emp_sal) VALUES(2,'robin', 'Hyderabad', 9848022339, 40000);
cqlsh:tutorialspoint> INSERT INTO emp (emp_id, emp_name, emp_city,
... emp_phone, emp_sal) VALUES(3,'rahman', 'Chennai', 9848022330, 45000);
cqlsh:tutorialspoint> select * from emp;
```

emp_id	emp_city	emp_name	emp_phone	emp_sal
1	Hyderabad	ram	9848022338	50000
2	Hyderabad	robin	9848022339	40000
3	Chennai	rahman	9848022330	45000

(3 rows)  
cqlsh:tutorialspoint>

Lire quelques données:

`SELECT emp_name, emp_sal FROM emp;`

```
cqlsh:tutorialspoint> SELECT emp_name, emp_sal from emp;
```

emp_name	emp_sal
ram	50000
robin	50000
rajeev	30000
rahman	50000

- Create Index:

```
CREATE INDEX ON emp(emp_sal);
SELECT * FROM emp WHERE emp_sal=50000;
```

```
cqlsh:tutorialspoint> CREATE INDEX ON emp(emp_sal);
cqlsh:tutorialspoint> SELECT * FROM emp WHERE emp_sal=50000;
```

emp_id	emp_city	emp_name	emp_phone	emp_sal
1	Hyderabad	ram	9848022338	50000
2	Delhi	robin	9848022339	50000
3	Chennai	rahman	9848022330	50000

(3 rows)

```
cqlsh:tutorialspoint> _
```

- Supprimer une donnée: DELETE emp\_sal FROM emp WHERE emp\_id=3;  
select \* from emp;

```
cqlsh:tutorialspoint> DELETE emp_sal FROM emp WHERE emp_id=3;
cqlsh:tutorialspoint> SELECT * FROM emp;
```

emp_id	emp_city	emp_name	emp_phone	emp_sal
1	Hyderabad	ram	9848022338	50000
2	Delhi	robin	9848022339	50000
4	Pune	rajeev	9848022331	30000
3	Chennai	rahman	9848022330	null

(4 rows)

- Batch command:  
 BEGIN BATCH  
 INSERT INTO emp (emp\_id, emp\_city, emp\_name, emp\_phone, emp\_sal)  
 values( 4,'Pune','rajeev',9848022331, 30000);  
 UPDATE emp SET emp\_sal = 50000 WHERE emp\_id =3;  
 DELETE emp\_city FROM emp WHERE emp\_id = 2;  
 APPLY BATCH;

```
cqlsh:tutorialspoint> BEGIN BATCH
... .. INSERT INTO emp (emp_id, emp_city, emp_name, emp_phone, emp_sal) values( 4,'Pune','rajeev',9848022331, 30000);
... .. UPDATE emp SET emp_sal = 50000 WHERE emp_id =3;
... .. DELETE emp_city FROM emp WHERE emp_id = 2;
... .. APPLY BATCH;
SyntaxException: line 2:0 mismatched input '.' expecting K_APPLY (BEGIN BATCH[.]....)
cqlsh:tutorialspoint> BEGIN BATCH
... .. INSERT INTO emp (emp_id, emp_city, emp_name, emp_phone, emp_sal) values( 4,'Pune','rajeev',9848022331, 30000);
... .. UPDATE emp SET emp_sal = 50000 WHERE emp_id =3;
... .. DELETE emp_city FROM emp WHERE emp_id = 2;
... .. APPLY BATCH;
cqlsh:tutorialspoint> select * from emp;
```

emp_id	emp_city	emp_name	emp_phone	emp_sal
1	Hyderabad	ram	9848022338	50000
2	null	robin	9848022339	40000
4	Pune	rajeev	9848022331	30000
3	Chennai	rahman	9848022330	50000

(4 rows)

- Aggregate function:  
SELECT COUNT (emp\_sal) FROM emp;

```
cqlsh:tutorialspoint> select count (emp_sal) from emp;

system.count(emp_sal)
-----
3
(1 rows)
```

SELECT SUM (emp\_sal) FROM emp;

```
cqlsh:tutorialspoint> select sum (emp_sal) from emp;

system.sum(emp_sal)
-----
130000
(1 rows)
```

SELECT AVG (emp\_sal) FROM emp;

```
cqlsh:tutorialspoint> select avg (emp_sal) from emp;

system.avg(emp_sal)
-----
43333
(1 rows)
```



## 6 Conclusion

Pour résumer, Cassandra est le système conçu pour traiter un très grand nombre de données. Les données suivent une stratégie de réplication pour être invulnérable aux pannes mais aussi pour garantir l'accès à ceux-ci par l'utilisateur à tout moment. Il n'y a pas besoin de surveillance permanent de la part de son utilisateur.

Ensuite, après avoir testé les quelques commandes de base qu'on a vu, on peut noter quelques points positifs de l'interface CQLSH:

- quand il y a une erreur de syntaxe sur la ligne de commande, la correction est proposé par cqlsh.
- si la table existe déjà, une information affiche que la colonne existe déjà.
- les requêtes se termine toujours par un point virgule ”;”.

Enfin, Cassandra est utilisé par des grands groupes parce que sa légitimité n'est plus à démontrer. Dans la pratique, il est préférable de prendre des bases de données orientées colonnes pour les données qui doivent être mis à jour régulièrement. Par exemple, le suivi des données reçus d'un capteur ou IoT (montre connectées par exemple) ou le suivi d'une commande faite sur internet (préparation, expédition, livraison).

Quelle est alors la limite de Cassandra comparé à d'autres SGBD NoSQL comme Hbase d'Apache ou du BigTable de Google?

## References

<https://cassandra.apache.org/>

<https://www.datastax.com/>

<https://www.tutorialspoint.com/cassandra/index.htm>

Ce document est édité sur LaTeX par Thierry RALAINARIVO