

Ministère de l'Enseignement Supérieure
Et de la Recherche Scientifique

République de Côte d'Ivoire
Union - Discipline - Travail



Université
de Technologie
d'Abidjan



Niveau d'Etude : Licence3

RAPPORT DE PROJET : BASE DE DONNEES ORIENTEES COLONNES

Année Académique : 2024 – 2025

Groupe 3

 **AGOH CHRIS**

 **KOUAKOU YANN**

 **DABO ALI**

 **KASSI JOSEPH**

Professeur

Dr SORO

Table des matières

I.	Introduction.....	P3
II.	Caractéristiques et exemples base de données NoSQL orienté Colonne.....	P4-P6
1.	Caractéristiques.....	P4
2.	Exemples de base de données.....	P5-P6
III.	Stockage dans les bases de données NoSQL.....	P6-P8
IV.	Cas d'utilisation, forces et faiblesses.....	P8-P10
1.	Cas d'utilisation.....	P8-P9
2.	Forces.....	P9
3.	Faiblesses.....	P9-P10
V.	Choix, Mise en Place et Installation.....	P10-P17
1.	Choix de la base de données NoSQL orientée colonne.....	P10
2.	Préparation des Environnements d'installation.....	P10
3.	Installation sur Windows.....	P10-P13
4.	Installation sur Linux.....	P13-P15
5.	Requêtes de base (CREATE, INSERT, SELECT, UPDATE, DELETE).....	P15-P17
VI.	Conclusion.....	P18
VII.	Bibliographie.....	P19

I. Introduction

Une base de données NoSQL est un système de base de données non relationnelle utilisée pour stocker des données sous une forme structurée ou non, notamment les objets et les graphes. Elle est une solution des limites des bases de données SQL, à savoir la difficulté de gestion des données de grand volume et de différents types. Cependant, il existe plusieurs bases de données NoSQL. En ce qui nous concerne, nous nous intéressons aux bases de données NoSQL orientées colonne. Dans la suite de notre travail, nous élaborons différents aspects concernant une base de données NoSQL orientée colonne, à savoir ses caractéristiques et exemples de stockage, ses cas d'utilisation, ses forces et ses faiblesses, ensuite ses modèles de base de données et le choix de la base de données pour terminer l'installation et les requêtes de base sur la base de données choisie.

II. Caractéristiques et exemples de base de données No SQL orienté colonne

1. Caractéristiques

1. Stockage en Colonnes

- Les données sont stockées en colonnes plutôt qu'en lignes. Cela permet un accès rapide aux données pertinentes pour les requêtes analytiques.

2. Familles de Colonnes

- Les données sont organisées en familles de colonnes, ce qui permet un stockage efficace et une récupération rapide des données.

3. Scalabilité Horizontale

- Facilité de scalabilité en ajoutant des serveurs supplémentaires pour gérer des volumes de données importants.

4. Tolérance aux Pannes :

- Conçues pour être distribuées, les bases de données colonnaires offrent une haute disponibilité et une tolérance aux pannes grâce à la réplication des données.

2. Exemples de base de données

Ils existent plusieurs bases de données NoSQL orientée colonne à savoir :

a. Cassandra



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, assurant une haute disponibilité en éliminant les points de défaillance unique. Il permet une répartition robuste sur plusieurs centres de données, avec une réplication asynchrone sans nœud maître et une faible latence pour les opérations de tous les clients.

Développé par : Fondation Apache

Langage : java

Dernière version : 5.0.2 (19 octobre 2024)

Système d'exploitation : Linux et type Unix

b. HBase



HBase est un système de gestion de base de données non-relationnelles distribué, écrit en Java, disposant d'un stockage structuré pour les grandes tables.

Développé par : Apache Software Foundation

Langage : java

Dernière version : 2.0.0 (13 Mars 2015)

Version avancée : 2.5.0 (31 Août 2022)

Système d'exploitation : Multiplateforme

c. Hypertable



Hypertable est un système de gestion de bases de données compressées inspiré par le système Big Table développé par Google, mais distribué sous licence libre (GPLv3).

Langage : C++

Système d'exploitation : Linux

III. Stockage dans les bases de données NoSQL orienté colonne

▪ Données à stocker

Clé	Nom	Prénoms	Contact	Adresse	Profession
A	Agoh	Chris	Null	Adjamé	Etudiant
B	Dabo	Ali	01-02-03-04-05	Null	Etudiant
C	Kassi	Joseph	Null	Abobo	Null
D	Kouakou	Yann	05-06-07-08-09	Null	Etudiant

▪ Stockage en BD No SQL Orienté colonne

A	Nom : Agoh
---	------------

	Prénoms : Chris
	Adresse : Adjamé
	Profession : Etudiant

B	Nom : Dabo
	Prénoms : Ali
	Contact :01-02-03-04-05
	Adresse : Adjamé
	Profession : Etudiant

C	Nom : Kassi
	Prénoms : Joseph
	Adresse : Abobo

D	Nom : Kouakou
	Prénoms : Yann
	Contact :05-06-07-08-09
	Profession : Etudiant

NB : GESTION DES VALEURS NULLS

Dans une base de données NoSQL orientée colonne, il peut y avoir des valeurs nulles. Cependant, cela dépend du système de gestion de base de données NoSQL spécifique que vous utilisez, car chaque système peut gérer les valeurs nulles de manière différente.

Exemple : Cassandra

Cassandra permet également de stocker des valeurs nulles explicites, mais cela nécessite une gestion spéciale (par exemple, les tombstones dans Cassandra).

IV. Cas d'utilisation, forces et faiblesses d'une base de données NoSQL orientée colonne

1. Cas d'utilisation

Les bases de données colonnaires sont particulièrement adaptées aux applications qui nécessitent un accès rapide à de grandes quantités de données pour des analyses complexes. Voici quelques cas d'utilisation typiques :

a. Analyse de Big Data

Exemple : Les entreprises collectent de grandes quantités de données à partir de diverses sources (logs de serveurs, réseaux sociaux, transactions, etc.). Les bases de données colonnaires permettent de stocker ces données de manière efficace et de les analyser rapidement pour extraire des insights utiles.

b. Systèmes de Recommandation

Exemple : Les plateformes de commerce électronique et de streaming utilisent des bases de données colonnaires pour stocker les interactions des utilisateurs et recommander des produits ou des contenus pertinents en temps réel.

c. Applications IoT (Internet of Things)

Exemple : Les dispositifs IoT génèrent un flux continu de données. Les bases de données colonnaires peuvent ingérer ces données en temps réel et permettre des analyses rapides pour des applications telles que la maintenance prédictive.

d. Gestion de Log et Monitoring (Surveillance à l'aide d'un moniteur)

Exemple : Les systèmes de surveillance des performances des applications et des infrastructures IT utilisent des bases de données colonnaires pour stocker et analyser des logs en temps réel, permettant ainsi de détecter rapidement des anomalies.

2. Forces

- **Scalabilité :** Les bases de données colonnaires peuvent évoluer horizontalement pour gérer des volumes de données massifs sans compromettre les performances.
- **Performance :** Optimisées pour les requêtes analytiques, permettant une récupération rapide des données pertinentes.
- **Flexibilité :** Capacité à gérer des schémas de données flexibles et à s'adapter aux besoins changeants des applications.
- **Tolérance aux Pannes :** Conçues pour être distribuées, offrant une haute disponibilité et une tolérance aux pannes.

3. Faiblesses

- **Complexité :** La gestion et la configuration des bases de données colonnaires peuvent être complexes par rapport aux SGBDR.
- **Coût :** Les solutions de bases de données colonnaires peuvent nécessiter des ressources matérielles importantes, augmentant ainsi les coûts.
- **Support Transactions :** Les bases de données colonnaires ne sont pas toujours optimisées pour les transactions ACID, ce qui peut être une limitation pour certaines applications.

V. Choix, Installation et Mise en Place

1. Choix de la base de données NoSQL orientée colonne

Nous avons fait le choix d'utiliser Apache Cassandra pour l'installation sur les deux systèmes d'exploitation (Windows et Linux).

2. Préparation des Environnements d'installation

- **Prérequis sur Windows et Linux**

Installation de **docker** sur Windows et Linux.

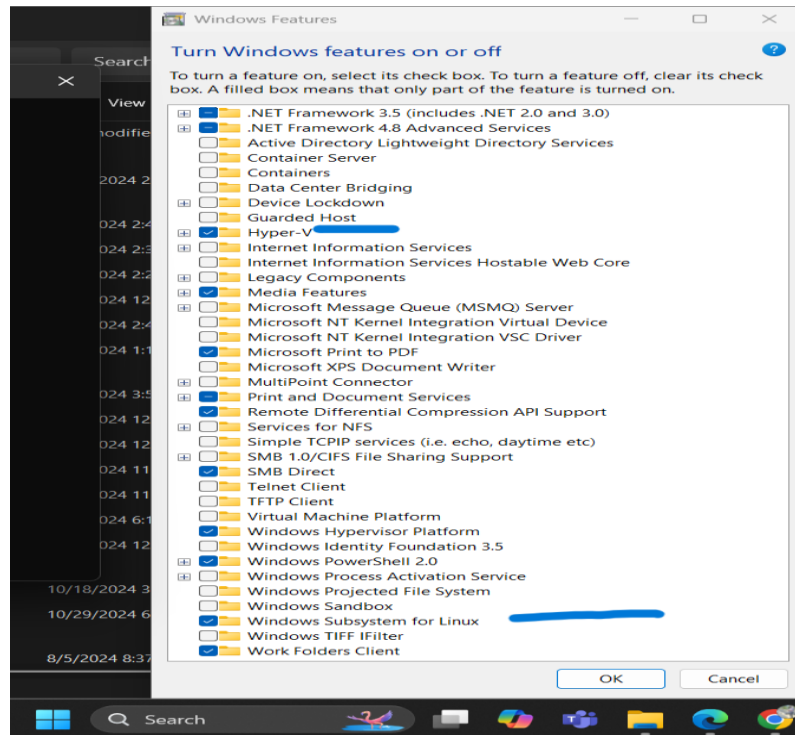
Vérification des prérequis :

- Système d'exploitation : Windows 11 (64-bit, version 21H2 minimum)
- RAM : minimum 4 Go
- Sous-système Linux : Activer WSL2 et Hyper-V
- Compte Docker : (Inscription sur <https://www.docker.com> si vous n'avez pas de compte).

3. Installation sur Windows

- **Installation de docker**

- a) Activer WSL2 et Hyper-V : Aller dans "Activer ou désactiver des fonctionnalités Windows".

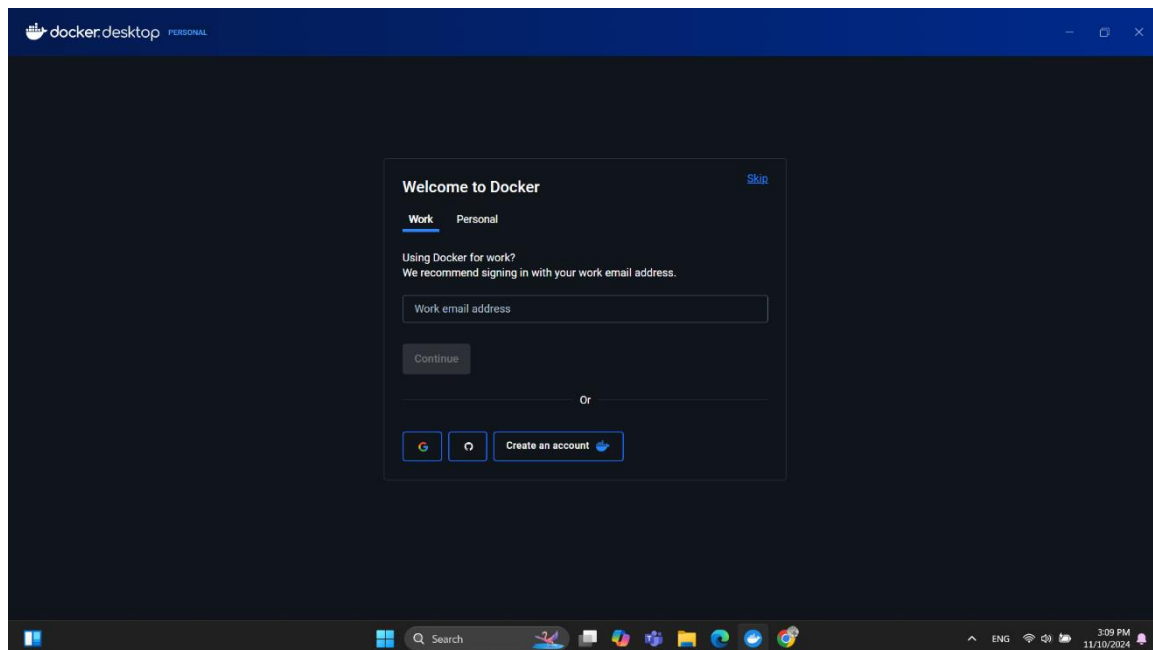


- b) Télécharger Docker Desktop : Aller sur docker.com et cliquer sur "Download Docker Desktop [Windows](#) | [Docker Docs](#)



- c) Installation de Docker Desktop : Suivre les étapes en cochant "Utiliser WSL2 au lieu de Hyper-V".

Tester Docker : Lancer Docker et vérifier avec ``docker version`` dans le terminal.



• Installation de Cassandra sur Windows avec docker

a) Télécharger l'image Cassandra : ``docker pull cassandra``

```

Microsoft Windows [Version 10.0.26100.2161]
(c) Microsoft Corporation. All rights reserved.

C:\Users\HP>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
mariadb/columnstore latest             376333191e88       15 months ago      2.18GB

C:\Users\HP>docker pull cassandra
Using default tag: latest
latest: Pulling from library/cassandra
2bfd35935537: Download complete
6717478e96f8: Download complete
89bb1bb42e9a: Download complete
143733ae87a4: Download complete
18154685d2bd: Download complete
17da8ec43a12: Download complete
f189a9d82ae7: Download complete
f4d133ca2b7f: Download complete
6414378b6477: Download complete
d12988e98d61: Download complete
Digest: sha256:5d4795c41491654e2bda432179e020c7c2cd782bbb22b7d1314747658efd71b4
Status: Downloaded newer image for cassandra:latest
docker.io/library/cassandra:latest

```

b) Lancer Cassandra avec Docker : ``docker run -p 7000:7000 -p 7001:7001 -p 7199:7199 -p 9042:9042 -p 9160:9160 --name cassandra -d cassandra:latest``

```

C:\Users\HP>docker run -p 7000:7000 -p 7001:7001 -p 7199:7199 -p 9042:9042 -p 9160:9160 --name cassandra -d cassandra:latest
9ad4da06791b497b4db829c31e5e8b13d2f97a344adb047d3e0c1ba7d359ab31

C:\Users\HP>docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
cassandra            latest             5d4795c41491       3 weeks ago        577MB
mariadb/columnstore latest             376333191e88       15 months ago      2.18GB

C:\Users\HP>docker ps
CONTAINER ID   IMAGE                  COMMAND
9ad4da06791b   cassandra:latest      "docker-entrypoint.s..."
0.0.0.0:9160->9160/tcp, 9042/tcp   cassandra
d45ef17bbd61   mariadb/columnstore   "/usr/bin/tini -- do..."
                                mariadb-columnstore
41 seconds ago   Up 40 seconds        0.0.0.0:7000-7001->7000-7001/tcp, 0.0.0.0:7199->7199/tcp,
2 hours ago     Up About an hour     0.0.0.0:3306->3306/tcp

```

d) Connexion à Cassandra : **`docker exec -it cassandra bash`**

```
NAME
9ad4da06791b  cassandra:latest  "docker-entrypoint.s..."  41 seconds ago  Up 40 seconds  0.0.0.0:7000-7001->7000-7001/tcp, 0.0.0.0:7199->7199/tcp,
0.0.0.0:9160->9160/tcp, 9042/tcp  cassandra
d45ef17b8de1  mariadb/columnstore  "/usr/bin/tini -- do..."  2 hours ago  Up About an hour  0.0.0.0:3306->3306/tcp
mariadb-columnstore

C:\Users\HP>docker exec -it 9ad4da06791b bash
```

e) Tester la connexion : **`DESCRIBE keyspaces;`** dans cqlsh.

```
Command Prompt
C:\Users\HP>docker exec -it 9ad4da06791b bash
root@9ad4da06791b:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE keyspace test WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> desc keyspaces;
```

4. Installation sur Linux

- **Installation de docker**

a) Mettre à jour le système : **`sudo apt update -y`**

b) Installer Docker : **`sudo apt install -y docker.io docker-compose`**

```
(dan@kali)-[~]
$ sudo apt install docker.io docker-compose
```

c) Vérifier l'installation : **`docker version`**

```
File Actions Edit View Help
(dan@kali)-[~]
$ docker version
Client:
Version: 20.10.25+dfsg1
API version: 1.41
Go version: go1.21.5
Git commit: b82b9f3
Built: Mon Jan 8 00:09:17 2024
OS/Arch: linux/amd64
Context: default
Experimental: true
permission denied while trying to connect to the
.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1
onnect: permission denied
(dan@kali)-[~]
```

- **Installation de Cassandra sur Linux avec docker**

a) Télécharger l'image Cassandra : **`docker pull cassandra`**

```
(alson@192)-[~]
$ sudo docker pull cassandra:latest
[sudo] Mot de passe de alson :
latest: Pulling from library/cassandra
6414378b6477: Pull complete
17da8ec43a12: Pull complete
d12988e90d61: Pull complete
f4d133ca2b7f: Pull complete
143733ae87a4: Pull complete
6717475e96f8: Pull complete
f189a9d82ae7: Pull complete
09bb1bb42e9a: Pull complete
10154685d2bd: Pull complete
2bfd35935537: Pull complete
Digest: sha256:5d4795c41491654e2bda432179e020c7c2cd702bbb22b7d1314747658efd7
Status: Downloaded newer image for cassandra:latest
docker.io/library/cassandra:latest
(alson@192)-[~]
```

- b) Démarrer le conteneur Cassandra : ``docker run -p 7000:7000 -p 7001:7001 -p 7199:7199 -p 9042:9042 -p 9160:9160 --name cassandra -d cassandra:latest``

```
(alson@192)-[~]
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
cassandra     latest   9a2732832972   3 weeks ago   379MB
hello-world    latest   d2c94e258dcb   18 months ago 13.3kB

(alson@192)-[~]
$ docker run -p 7000:7000 -p 7001:7001 -p 7199:7199 -p 9042:9042 -p 9160:9160 --name cassandra -d cassandra:latest
33b640d8cc2ada0df2d9821f8f1e718462f71f795877c8f271809d0431a2a6
```

- c) Connexion à Cassandra : Utiliser ``docker exec -it cassandra cqlsh``.

```
(alson@192)-[~]
$ sudo docker exec -it 33b640d8cc2a bash
[sudo] Mot de passe de alson :
root@33b640d8cc2a:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE keyspace test WITH replication = {'class': 'SimpleStrategy',
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

5. Requêtes de base (CREATE, INSERT, SELECT, UPDATE, DELETE)

- ✚ Test de l'installation de Cassandra : Créer un Keyspace et une Table

Créer un Keyspace : ``CREATE KEYSPACE test_keyspace WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};``

```
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE keyspace test WITH replication = {'class': 'SimpleStrategy',
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

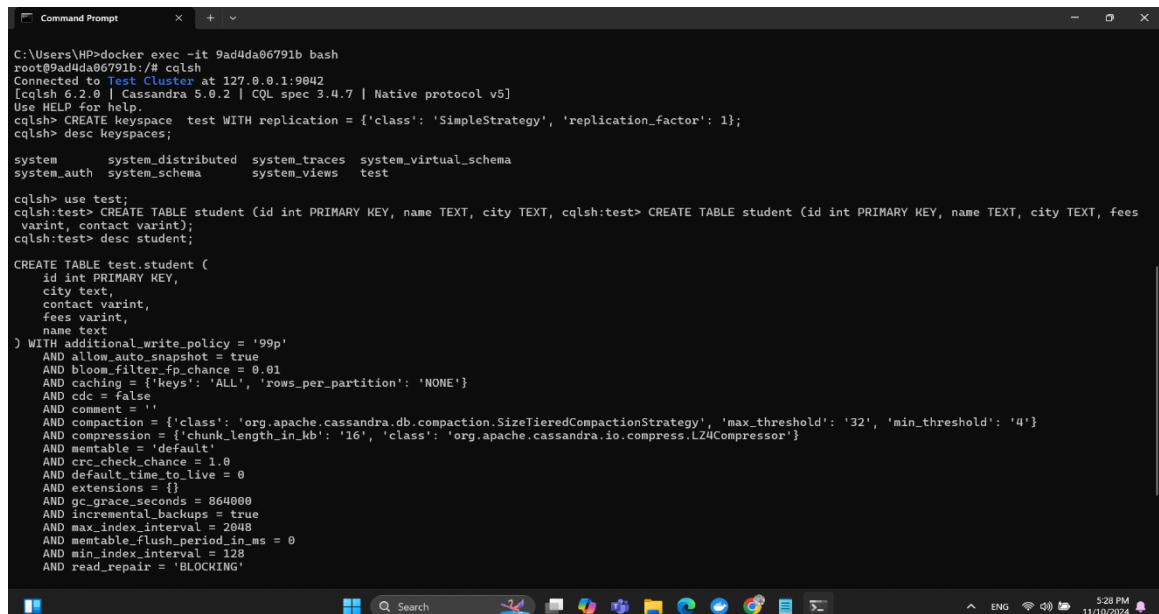
Créer une Table : Dans `test_keyspace`, `CREATE TABLE etudiants (id UUID PRIMARY KEY, nom TEXT, age INT, ville TEXT);`

Insérer des Données et Tester les Résultats

- Insertion partielle de données : `INSERT INTO etudiants (id, nom) VALUES (uuid(), 'Alice');`
- Visualisation des résultats : `SELECT * FROM etudiants;`

Pratique

- Sur Windows



```
C:\Users\HP>docker exec -it 9ad4da06791b bash
root@9ad4da06791b:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.2.0 | Cassandra 5.0.2 | CQL spec 3.4.7 | Native protocol v5]
Use HELP for help.
cqlsh> CREATE keyspace test WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh> desc keyspaces;

system          system_distributed  system_traces      system_virtual_schema
system_auth      system_schema       system_views        test

cqlsh> use test;
cqlsh:test> CREATE TABLE student (id int PRIMARY KEY, name TEXT, city TEXT, cqlsh:test> CREATE TABLE student (id int PRIMARY KEY, name TEXT, city TEXT, fees
varint, contact varint);
cqlsh:test> desc student;

CREATE TABLE test.student (
  id int PRIMARY KEY,
  city text,
  contact varint,
  fees varint,
  name text
) WITH additional_write_policy = '99p'
AND allow_auto_snapshot = true
AND bloom_filter_fp_chance = 0.01
AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
AND cdc = false
AND comment = ''
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_threshold': '32', 'min_threshold': '4'}
AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
AND memtable = 'default'
AND crc_check_chance = 1.0
AND default_time_to_live = 0
AND extensions = {}
AND gc_grace_seconds = 864000
AND incremental_backups = true
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair = 'BLOCKING'
```



```

varint, contact varint);
cqlsh:test> desc student;

CREATE TABLE test.student (
  id int PRIMARY KEY,
  city text,
  contact varint,
  fees varint,
  name text
) WITH additional_write_policy = '99p'
AND allow_auto_snapshot = true
AND bloom_filter_fp_chance = 0.01
AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
AND cdc = false
AND comment = ''
AND compaction = {'class': 'org.apache.cassandra.db.compaction.SizeTieredCompactionStrategy', 'max_
AND compression = {'chunk_length_in_kb': '16', 'class': 'org.apache.cassandra.io.compress.LZ4Compre
AND memtable = 'default'
AND crc_check_chance = 1.0
AND default_time_to_live = 0
AND extensions = {}
AND gc_grace_seconds = 864000
AND incremental_backups = true
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 0
AND min_index_interval = 128
AND read_repair = 'BLOCKING'
AND speculative_retry = '99p';
cqlsh:test> insert into student(id,name,city) values(1,'Alson','Abidjan');
cqlsh:test> select * from student;

id | city | contact | fees | name
---|---|---|---|---
1 | Abidjan | null | null | Alson

(1 rows)
cqlsh:test> exit
root@9ad4da06791b:/# exit
exit
C:\Users\HP>

```

- Sur linux

```

Debian 5 - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
Metasploit
Debian 5

root@33b640d8cc2a: /
Fichier Actions Éditer Vue Aide
cqlsh:test> INSERT INTO student (id, name, city) VALUES (1, 'Ali', 'Abidjan');
cqlsh:test> INSERT INTO student (id, name, city) VALUES (2, 'Py', 'Bouake');
cqlsh:test> SELECT * FROM student;

id | city | contact | fees | name
---|---|---|---|---
1 | Abidjan | null | null | Ali
2 | Bouake | null | null | Py
4 | Yamoussoukro | null | null | Kassi
3 | Daloa | null | null | Yan

(4 rows)
cqlsh:test> SELECT name, city FROM student;

name | city
---|---
Ali | Abidjan
Py | Bouake
Kassi | Yamoussoukro
Yan | Daloa

(4 rows)
cqlsh:test> history
CREATE keyspace test WITH replication = {'class': 'SimpleStrategy', 'replic
desc keyspaces;
desc test;
CREATE TABLE student (id int PRIMARY KEY, name TEXT, city TEXT, fees varint,
use test;
CREATE TABLE student (id int PRIMARY KEY, name TEXT, city TEXT, fees varint,
INSERT INTO users (id, name) VALUES (1, 'ali');
INSERT INTO users (id, name) VALUES (2, 'py');
INSERT INTO users (id, name) VALUES (3, 'yan');
INSERT INTO users (id, name) VALUES (4, 'kassi');
select * from test;
select * from test;
select * from test select * from test;
select * from test;
SELECT * FROM test;
INSERT INTO users (id, name) VALUES (1, 'ali');
INSERT INTO student (id, name, city) VALUES (1, 'Ali', 'Abidjan');
INSERT INTO student (id, name, city) VALUES (2, 'Py', 'Bouake');
INSERT INTO student (id, name, city) VALUES (3, 'Yan', 'Daloa');
INSERT INTO student (id, name, city) VALUES (4, 'Kassi', 'Yamoussoukro');
SELECT * FROM student;
SELECT name, city FROM student;
cqlsh:test>

```

VI. Conclusion

En conclusion, les bases de données NoSQL orientées colonne, telles qu'Apache Cassandra, offrent des solutions puissantes et flexibles pour le stockage et l'analyse de grandes quantités de données. En effet, leurs capacités à évoluer horizontalement et à offrir une haute disponibilité en font un choix idéal pour les applications modernes nécessitant des performances élevées et une grande résilience. À mesure que les besoins en données continuent de croître, les bases de données colonnaires joueront un rôle clé dans l'infrastructure de gestion des données des entreprises.

VII. Bibliographie

<https://fr.wikipedia.org/wiki/Cassandra>

https://en.wikipedia.org/wiki/Apache_HBase

<https://www.illustradata.com/bases-nosql-orientees-colonnes-quest-cest>

<https://waytolearnx.com/2019/10/quest-ce-quune-base-nosql.html>

<https://www.docker.com>