

# Bases de Données Complexes

Rapport final



# *cassandra*

Khalil Bennis  
Houda Boukham  
El Mokhtar Nourhira

Encadrés par Mme  
Laïla Benhlila

## Table des matières

---

Introduction à Apache Cassandra .....	2
Architecture.....	2
L'écriture des données.....	2
Configuration d'Apache Cassandra .....	3
Installation d'Apache Cassandra .....	4
Prérequis .....	4
Installation.....	4
Exemple de manipulation dans Apache Cassandra.....	6
Créer un keyspace .....	6
Créer une table.....	6
Insérer dans une table.....	6
Interroger la base de données .....	7
L'indexation dans Apache Cassandra .....	8
Primary key.....	8
Partition key .....	8
Clustering key .....	8
Secondary index .....	9
Application .....	10
Situation .....	10
Modélisation des données .....	10
Modélisation par le modèle de Chebotko.....	10
Environnement de développement .....	11
Fonctionnement de l'application .....	11
La communication avec la base de données.....	12
La manipulation des fichiers image.....	12
Conclusion .....	13

## Introduction à Apache Cassandra

---

Apache Cassandra est un système de gestion de base de données de type NoSQL orienté colonnes, conçu pour gérer des quantités massives de données sur un grand nombre de serveurs.

### Architecture<sup>1</sup>

L'architecture de Cassandra est constituée des composants principaux suivants :

- Node : la machine où les données sont stockées.
- Datacenter : un ensemble de nœuds.
- Cluster : un ensemble de datacenter.
- Commit log : log où les données sont écrites pour la récupération, avant d'être écrites dans les SSTable.
- SSTable (Sorted String Table) est un fichier immuable, fonctionnant en mode ajouter seulement, permettant ainsi aux memtables (des tables temporaires dans la mémoire cache) d'écrire périodiquement sur ce fichier. Il s'agit de l'implémentation physique de la table.
- CQL Table : une collection ordonnée de colonnes possédant une clé primaire. C'est la représentation logique de la table.

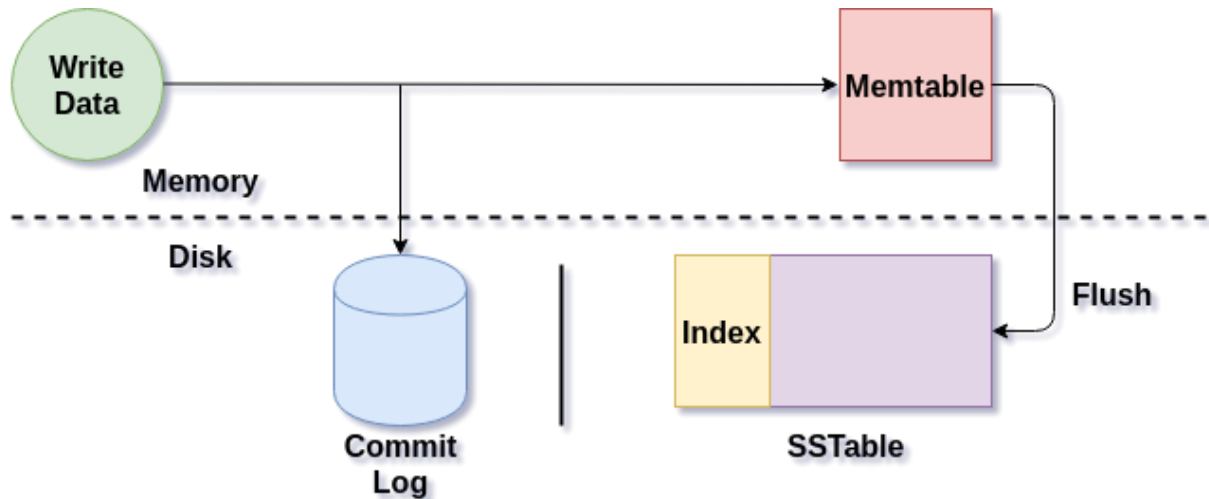
Cassandra possède une architecture peer-to-peer, c'est-à-dire que les données et leurs répliquions sont distribuées de manière homogène sur les différents nœuds. Cela la distingue du modèle master-slave, adopté par HDFS et HBase par exemple. Chaque nœud du cluster échange son état avec ses pairs en utilisant un protocole de communication appelé *gossip*.

### L'écriture des données

Les données sont d'abord écrites dans un fichier log appelé Commit Log. Elles sont ensuite écrites dans un memtable – une table temporaire dans la mémoire cache. Une fois la mémoire saturée, les données sont persistées dans un fichier dit SSTable.

---

<sup>1</sup> Référence : <https://docs.datastax.com/en/cassandra/3.0/cassandra/architecture/archTOC.html>

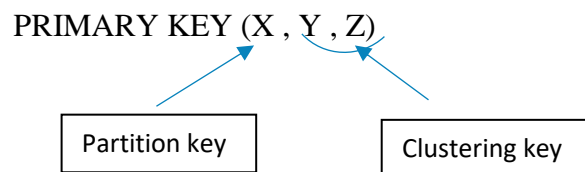


La procédure d'écriture des données dans Apache Cassandra

Source: <http://sudotutorials.com/tutorials/cassandra/how-data-is-written-to-cassandra.html>

## Configuration d'Apache Cassandra

Lors d'une requête cassandra, la clé primaire est un élément essentiel puisqu'elle permet de connaître l'emplacement d'une partition au sein d'un cluster. En effet, la clé primaire est utilisée pour le partitionnement d'une table CQL et permet d'identifier une ligne de façon unique. Elle est définie comme suit:



Ainsi, pour configurer Cassandra, il faut prendre en compte plusieurs éléments:

- Le *partitioner* : c'est le composant responsable de distribuer les données dans un cluster.
- Le facteur de réplication : le nombre de copies pour chaque ligne dans un cluster.
- La stratégie de réplication : comment les répliques seront distribuées dans le cluster (Network Topology Strategy est la stratégie la plus recommandée).

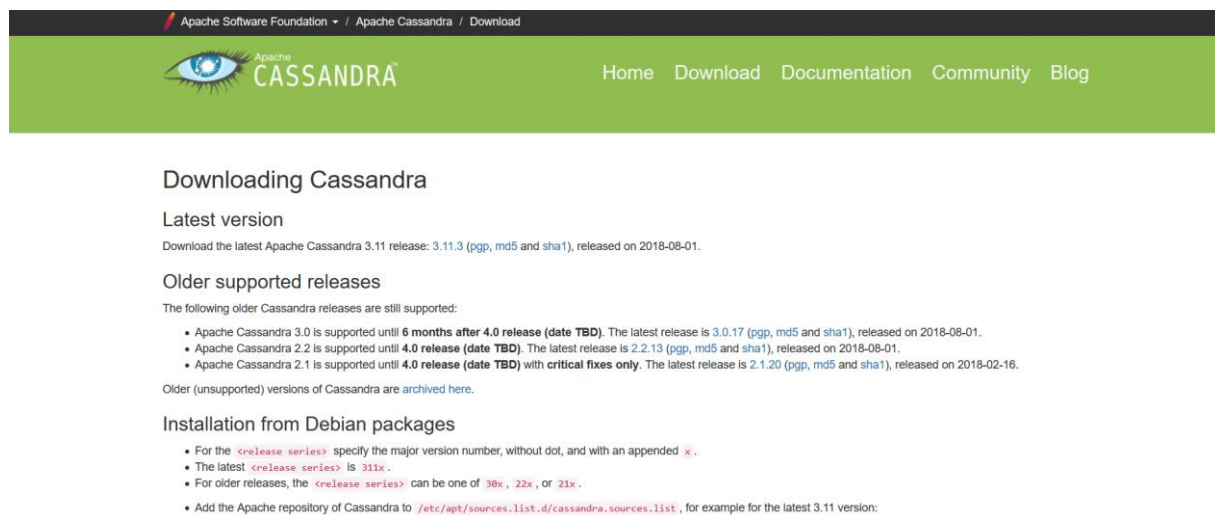
## Installation d'Apache Cassandra

### Prérequis

- Windows OS
- JDK doit être installé
- Python 2.7 doit être installé

### Installation

- Accéder au site web <http://cassandra.apache.org/download/>



Apache Software Foundation / Apache Cassandra / Download

Home Download Documentation Community Blog

### Downloading Cassandra

**Latest version**  
Download the latest Apache Cassandra 3.11 release: 3.11.3 (pgp, md5 and sha1), released on 2018-08-01.

**Older supported releases**  
The following older Cassandra releases are still supported:

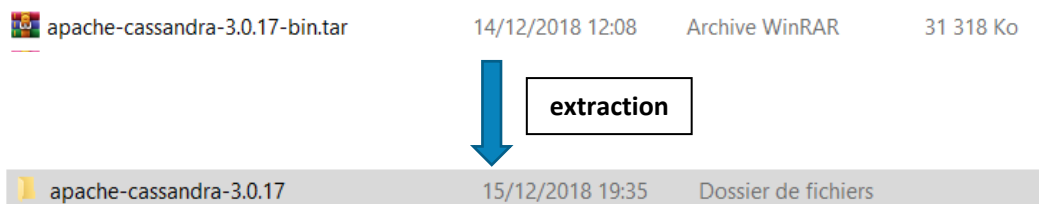
- Apache Cassandra 3.0 is supported until **6 months after 4.0 release (date TBD)**. The latest release is 3.0.17 (pgp, md5 and sha1), released on 2018-08-01.
- Apache Cassandra 2.2 is supported until **4.0 release (date TBD)**. The latest release is 2.2.13 (pgp, md5 and sha1), released on 2018-08-01.
- Apache Cassandra 2.1 is supported until **4.0 release (date TBD)** with **critical fixes only**. The latest release is 2.1.20 (pgp, md5 and sha1), released on 2018-02-16.

Older (unsupported) versions of Cassandra are [archived here](#).

**Installation from Debian packages**

- For the `<release series>` specify the major version number, without dot, and with an appended `x`.
- The latest `<release series>` is `311x`.
- For older releases, the `<release series>` can be one of `30x`, `22x`, or `21x`.
- Add the Apache repository of Cassandra to `/etc/apt/sources.list.d/cassandra.sources.list`, for example for the latest 3.11 version:

- Télécharger la dernière version de Cassandra.
- Extraire le document téléchargé.



apache-cassandra-3.0.17-bin.tar 14/12/2018 12:08 Archive WinRAR 31 318 Ko

extraction

apache-cassandra-3.0.17 15/12/2018 19:35 Dossier de fichiers

- Accéder au dossier bin (emplacementdudossier\apache-cassandra-3.0.17\bin)

cassandra	25/07/2018 07:32	Fichier	11 Ko
cassandra	15/12/2018 19:33	Fichier de comma...	7 Ko
cassandra.in	25/07/2018 07:32	Fichier de comma...	4 Ko
cassandra.in	25/07/2018 07:32	Shell Script	3 Ko
cassandra	25/07/2018 07:32	Script Windows Po...	13 Ko
cqlsh	25/07/2018 07:32	Fichier	2 Ko
cqlsh	25/07/2018 07:32	Fichier de comma...	2 Ko
cqlsh	25/07/2018 07:32	Python File	104 Ko
debug-cql	25/07/2018 07:32	Fichier	3 Ko
debug-cql	25/07/2018 07:32	Fichier de comma...	2 Ko
nodetool	25/07/2018 07:32	Fichier	4 Ko
nodetool	25/07/2018 07:32	Fichier de comma...	2 Ko
source-conf	25/07/2018 07:32	Script Windows Po...	2 Ko
sstableloader	25/07/2018 07:32	Fichier	2 Ko
sstableloader	25/07/2018 07:32	Fichier de comma...	2 Ko
sstablescrub	25/07/2018 07:32	Fichier	2 Ko

## Bases de Données Complexes

### Projet Final – Apache Cassandra

- Ouvrir cassandra.bat avec le bloc-notes et ajouter l'emplacement du JDK dans l'emplacement souligné.

```
goto runLegacy

REM -----
:runPowerShell
echo Detected powershell execution permissions. Running with enhanced startup scripts.
set errorlevel=
powershell /file "%CASSANDRA_HOME%\bin\cassandra.ps1" %*
exit /b %errorlevel%

REM -----
:runLegacy
echo WARNING! Powershell script execution unavailable.
echo Please use 'powershell Set-ExecutionPolicy Unrestricted'
echo on this user-account to run cassandra with fully featured
echo functionality on this platform.

echo Starting with legacy startup options

if NOT DEFINED CASSANDRA_MAIN set CASSANDRA_MAIN=org.apache.cassandra.service.CassandraDaemon
if NOT DEFINED JAVA_HOME set JAVA_HOME=C:\Program Files\Java\jdk1.8.0_181

REM -----
REM JVM Opts we'll use in legacy run or installation
set JAVA_OPTS=-ea^
-javaagent:"%CASSANDRA_HOME%\lib\jamm-0.3.0.jar"^
-Xms2G^
-Xmx2G^
-XX:+HeapDumpOnOutOfMemoryError^
-XX:+UseParNewGC^
-XX:+UseConcMarkSweepGC^
-XX:+CMSParallelRemarkEnabled^
-XX:+CMSClassUnloadingEnabled^
```

- Enregistrer et fermer le bloc-notes.
- Exécuter « cassandra.bat » pour démarrer Cassandra (cliquer deux fois sur le fichier).
- Exécuter « cqlsh.bat » pour manipuler les données dans Cassandra à partir de la console (cliquer deux fois sur le fichier).

```
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 3.0.17 | CQL spec 3.4.0 | Native protocol v4]
Use HELP for help.
WARNING: pyreadline dependency missing. Install to enable tab completion.
cqlsh>
```

- Maintenant vous pouvez exécuter des requêtes CQL à travers cette console.

## Exemple de manipulation dans Apache Cassandra

---

### Créer un keyspace

Un « Keyspace » est l'équivalent d'une « database » dans une Base de données relationnelle telle que Oracle. On peut créer un Keyspace avec la syntaxe suivante :

```
CREATE KEYSPACE "nom du KeySpace" WITH replication = {'class': 'nom  
de la stratégie de réplication utilisé', 'replication_factor' :  
'Nombre de réplication '};
```

Exemple :

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

### Créer une table

Pour créer une table, on utilise la syntaxe suivante :

```
CREATE TABLE tablename(  
    column1 name data type PRIMARYKEY,  
    column2 name data type,  
    column3 name data type.  
)
```

Exemple :

```
create table emp( emp_id int PRIMARY KEY , emp_name text);
```

### Insérer dans une table

Pour insérer dans une table, on utilise la syntaxe suivante :

```
INSERT INTO <tablename>  
(<column1 name>, <column2 name>....)  
VALUES (<value1>, <value2>....)  
USING <option>
```

Exemple :

```
INSERT INTO emp (emp_id, emp_name) VALUES (1,'Boukham Houda');
```

## Interroger la base de données

Exemple :

```
select * from emp;
```

Le résultat est le suivant :

emp_id	emp_name
1	Boukham Houda



## L'indexation dans Apache Cassandra<sup>2</sup>

### Primary key

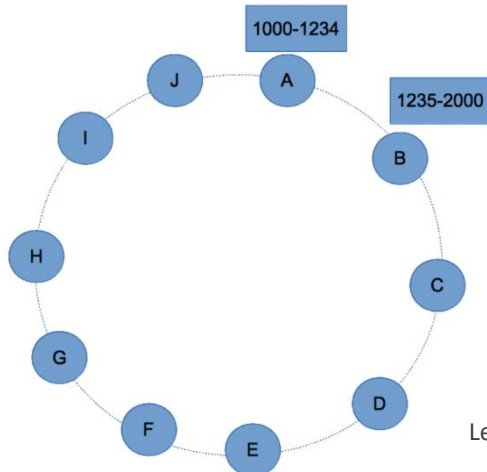
Chaque ligne est identifiée par une clé primaire ou primary key, dite aussi row key. Cette clé peut être composée, auquel cas le 1<sup>er</sup> composant constitue le partition key, et le reste le clustering key.

Différents cas:

Primary key	Partition key	Clustering key
(id)	id	aucun
(id, nom)	id	nom
(id, nom, prenom)	id	nom, prenom
(id, (nom, prenom))	id	nom, prenom
((id, nom, prenom), (date_naissance, lieu_naissance))	Id, nom, prenom	date_naissance, lieu_naissance

### Partition key

Le partition key permet de retrouver le nœud dans le cluster où est stockée une ligne donnée. Lorsqu'une donnée est lue ou écrite, une fonction dite The Partionner associe à son partition key un hash code. C'est ce code-là qui permet de déterminer le nœud qui contient la donnée. Considérons par exemple la figure ci-dessous :



Le nœud A contient les lignes dont les partition keys sont compris entre 1000 et 1234, le nœud B ceux compris entre 1235 et 2000. Si une ligne possède un partition key dont le hash code est de 1200, elle sera stockée sur le nœud A.

Le partitioning dans Apache Cassandra

Source : [https://dzone.com/articles/cassandra-data-modeling-primary-clustering-partiti?fbclid=IwAR0wwluGusn78ZtSnyJQJ46bXMI\\_fbgBb0s2S53hVbL5bWnnSQQWj5Fz4w](https://dzone.com/articles/cassandra-data-modeling-primary-clustering-partiti?fbclid=IwAR0wwluGusn78ZtSnyJQJ46bXMI_fbgBb0s2S53hVbL5bWnnSQQWj5Fz4w)

### Clustering key

Le clustering key permettent de trier les données d'une même partition, afin de pouvoir récupérer les données de manière plus efficace.

<sup>2</sup> Référence : [https://www.datastax.com/dev/blog/cassandra-native-secondary-index-deep-dive?fbclid=IwAR01NNLbHoK9Eri\\_QntdrOB2r7ixlu01DvwuDKRXF5\\_OR9KdVqnzYtIsn1s](https://www.datastax.com/dev/blog/cassandra-native-secondary-index-deep-dive?fbclid=IwAR01NNLbHoK9Eri_QntdrOB2r7ixlu01DvwuDKRXF5_OR9KdVqnzYtIsn1s)

## Secondary index

Dans Apache Cassandra, l'index – dit Secondary Index – permet d'accéder à des colonnes qui ne font pas partie de la clé primaire.

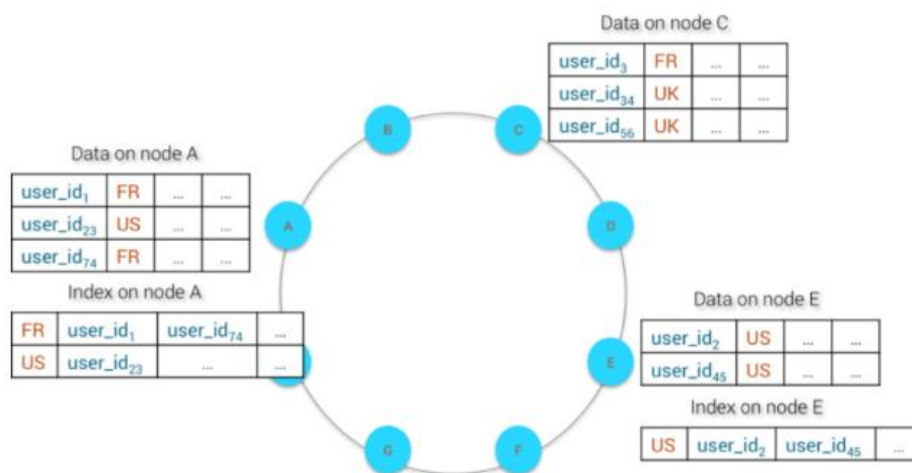
En effet, si l'on essaie de faire une requête sur des données autre que la clé primaire, le nom de patient par exemple, le message d'erreur suivant apparaît : `InvalidRequest: code=2200 [Invalid query] message="No supported secondary index found for the non primary key columns restrictions"`.

La solution serait d'ajouter un index secondaire sur la colonne nom. Pour cela, il faudrait écrire :

```
cqlsh:gestion_medicale> CREATE INDEX ON patient ( nom );
```

A présent, l'on pourrait effectuer la requête suivante :

```
cqlsh:my_keyspace> SELECT * FROM patient WHERE nom = 'Smith';
```



Source : [https://www.datastax.com/dev/blog/cassandra-native-secondary-index-deep-dive?fbclid=IwAR01NNLbHoK9Eri\\_QntdrOB2r7ixlu01DvwuDKRXF5\\_OR9KdVqnzYtIsn1s](https://www.datastax.com/dev/blog/cassandra-native-secondary-index-deep-dive?fbclid=IwAR01NNLbHoK9Eri_QntdrOB2r7ixlu01DvwuDKRXF5_OR9KdVqnzYtIsn1s)

Considérons la situation suivante, représentée dans la figure ci-dessus. Nous avons une table `users`, composée d'une clé primaire `user_id`, et d'une colonne `country`. Pour pouvoir effectuer une recherche sur `country`, nous devons ajouter un index secondaire sur cette colonne. L'index sera alors une table cachée possédant cette structure :

```
CREATE TABLE country_index(  
    country text,  
    user_id bigint,  
    PRIMARY KEY((country), user_id)  
);
```

L'index est stocké dans le même nœud que la table, et est donc lui aussi distribué.

Note : Bien que Cassandra soit orienté colonne, la recherche s'y fait par ligne.

## Application

### Situation

Notre application doit permettre la gestion des patients, de leurs maladies et des radiographies associées à celles-ci.

Les données à modéliser sont les suivantes :

Patients : nom, prénom, date naissance,

Maladies du patient : médecin traitant, nom maladie, symptômes.

Radios du patient : image, date radio, médecin ayant demandé la radio.

### Modélisation des données

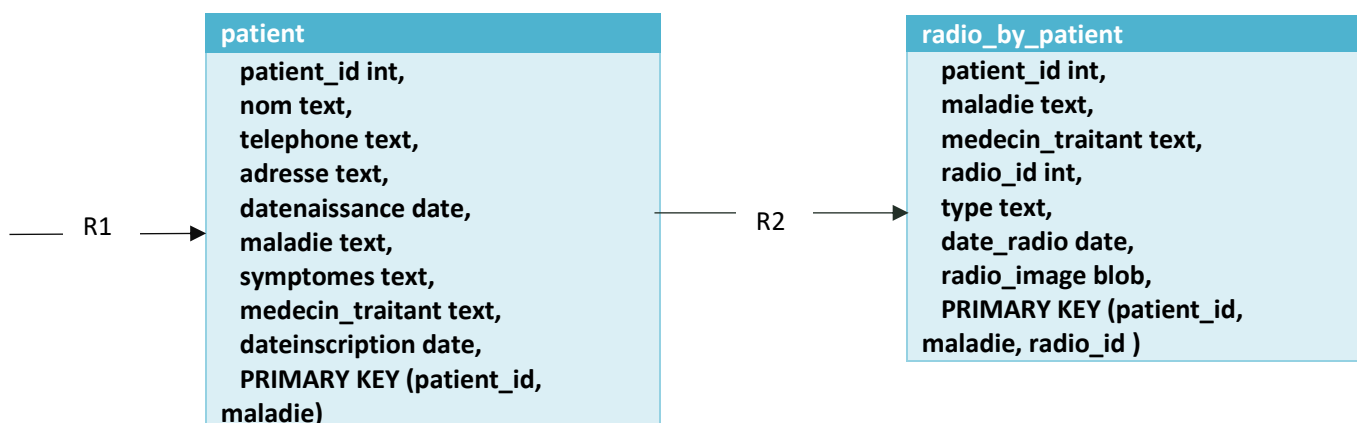
Les soucis de modélisation diffèrent entre les bases de données relationnelles et les bases de données NoSQL. Alors que dans les premières, la normalisation était un souci primordial, dans la seconde, la dénormalisation est « normale ». Il n'importe pas autant d'éviter les redondances que d'optimiser la performance, par exemple. Comme les jointures n'existent pas dans Apache Cassandra, il sert toujours de réunir les données qui vont ensemble dans une même table.

Modélisation par le modèle de Chebotko<sup>3</sup>

La communauté Cassandra a proposé plusieurs notations pour modéliser les données sous forme d'un diagramme. L'une de ces notations est le modèle Chebotko représenté ci-dessous.

R1. Afficher les informations sur un patient.

R2. Afficher les radiographies d'un patient.



Le patient est identifié par un patient\_id. Il est représenté par son nom, son numéro de téléphone, son adresse et sa date de naissance. Un patient peut être traité pour une ou plusieurs maladies. Pour chaque maladie, le patient est traité par un médecin.

<sup>3</sup> Référence : <https://www.oreilly.com/ideas/cassandra-data-modeling>

Les radiographies d'un patient sont liées à sa maladie. Une radiographie est définie par un identifiant `radio_id`, un type (scanner, cliché pulmonaire, échographie...), la date à laquelle elle a été prise, le médecin l'ayant demandé, et la maladie pour laquelle elle a été demandée. Le fichier image est lui-même stocké dans la table des radiographies.

## Environnement de développement

Notre projet est conçu sous forme d'une application desktop, développée sous Java et utilisant le framework JavaFX. La connexion à la base de données est assurée par le driver Datastax (disponible sur <https://docs.datastax.com/en/developer/java-driver/3.0/>). Nous avons employé Scene Builder pour la construction de l'interface graphique.

## Fonctionnement de l'application

L'application comporte deux interfaces.

La première interface permet de sélectionner un patient et d'afficher ses informations.

The screenshot shows a desktop application window titled "Gestion médicale" with a sub-header "Gestion des patients". It includes a patient selection dropdown, date range pickers, a patient information form, a table of medical history, and a button to view radiographs. Four blue callout boxes provide instructions:

- 1 Choisir le patient à afficher**: Points to the "Sélectionnez un patient:" dropdown menu.
- 2 Choisir un intervalle où seront comprises les dates d'ouverture de dossier**: Points to the "Choisissez un intervalle:" date range pickers.
- 3 Consulter la liste des maladies pour lesquelles le patient a été traité**: Points to the table of medical history.
- 4 Consulter les radiographies du patient sélectionné**: Points to the "Consulter les radiographies" button.

**Sélectionnez un patient:** John Smith

**Choisissez un intervalle:** Du 01/02/2018 Au 22/02/2018 Go

Nom: John Smith  
Date de naissance: 1989-05-13  
Numéro de téléphone: 0624269011  
Adresse: Rabat

Maladie	Médecin traitant	Date d'ouverture du dossier	Symptômes
Grippe	Jane Doe	2018-02-08	Fatigue, nez qui coule
TB	Jane Doe	2018-02-14	Ganglion enflé, dur, indolore

Consulter les radiographies

En cliquant sur le bouton « Consulter les radiographies », l'utilisateur est dirigé vers la deuxième interface. Celle-ci permet en effet de consulter la liste des radiographies effectuées par le patient sélectionné.

The screenshot shows a web application window titled "Radiographie de John Smith". It contains three main sections: a filter section at the top, a table of radiographies, and a preview of a selected radiograph.

**5 Sélectionner une maladie par laquelle filtrer**: Points to the "Sélectionnez une maladie:" dropdown menu, which currently shows "Grippe".

**6 Choisir un intervalle où seront comprises les dates de radiographie**: Points to the "Sélectionnez un intervalle de temps:" section, which includes date pickers for "01/02/2018" and "15/02/2018", and a "Go" button.

**7 Consulter l'identifiant de la radiographie dont on souhaite afficher l'image**: Points to the "Id" column in the table, which shows the value "13".

**8 Sélectionner l'identifiant de la radiographie dont on souhaite afficher l'image**: Points to the "Sélectionnez la radiographie à afficher:" dropdown menu, which also shows "13".

Id	Type	Date	Médecin traitant
13	Radiographie	2018-02-08	Jane Doe

The preview section on the right displays a chest X-ray image.

## La communication avec la base de données

Le Java driver de Datastax pour Apache Cassandra rend possibles la communication entre notre application et notre base de données. Il suffit de créer un cluster, une session, de déterminer l'adresse et le port sur lesquels Cassandra est configuré, et d'introduire la requête à exécuter.

```
cluster =  
Cluster.builder().addContactPoint("127.0.0.1").WithPort(9042).build();  
//connexion à la base de données  
session = cluster.connect("gestionmedicale");  
//connexion au keyspace gestionmedicale  
ResultSet results = session.execute("Select * from patient");  
//exécution d'une requête CQL et enregistrement du résultat dans un  
ResultSet  
Cluster.close();  
//fermer la connexion
```

## La manipulation des fichiers image

Les fichiers image des radiographies sont stockés sous format Blob sur la base de données.

L'image est récupérée sous format Blob et est ensuite transformée en un tableau de bytes. Nous créons un flux streaming qui permet d'injecter ce tableau de bytes dans un objet ImageView (un élément graphique qui permet d'afficher les images sur JavaFX).

## Conclusion

---

Ce projet a servi de bonne introduction aux bases de données NoSQL en général, et à Cassandra Apache en particulier. Il nous aura permis de comprendre le principe des bases de données NoSQL orientées colonne, et de prendre connaissance des spécificités de Cassandra.

Cassandra, en effet, est intéressante, en cela qu'elle possède une architecture peer-to-peer, par opposé à l'architecture master-slave à laquelle nous sommes accoutumés. D'autant plus que ce choix d'architecture nous évite les Single Point of Failure, un problème récurrent dans plusieurs bases de données NoSQL, HBase à titre d'exemple.

Les bases de données NoSQL sont désormais moins intimidantes, et nous sommes motivés à en explorer d'autre, et à continuer notre découverte du domaine du Big Data.

Nous trouvons, somme toute, que les projets académiques de ce semestre – les derniers de notre formation – ont été bien instructifs et riches en valeur ajoutée, et c'est avec enthousiasme et motivation que nous débarquons en PFE.