Intéraction avec une base Cassandra

Travaux Pratiques

ESILV
nicolas.travers (at) cnam.fr

Chapitre

Table des matières

Chapitre 1

Guide d'utilisation de Cassandra

CASSANDRA est un serveur de Bases de Données NoSQL basé sur une table de hachage distribuée (DHT) inspiré de DynamoDB (Amazon). Il a été créé par Facebook et maintenant est maintenu par Apache et disribué par DataStax.

Pour une utilisation en environnement de production à large échelle, il est recommandé de l'installer sur un serveur Linux.

1.1 Téléchargement

Vous aurez besoin du serveur **Cassandra** et de l'interface graphique **DevCenter**. Ils sont tous deux disponibles dans l'archive de la "*Community Edition*" v3.9.0 (Linux / MacOSX / Windows) sur :
`https://academy.datastax.com/planet-cassandra/cassandra`

Une version **JDK 1.8** est nécessaire pour faire fonctionner Cassandra :
`http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
Il sera nécessaire de changer les variables d'environnement pour que le `JAVA_HOME` soit défini :
`https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/`

1.2 Installation de Cassandra

1.2.1 Sous Windows

Un guide complet à suivre pour l'installation est disponible ici :
`http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way`

Pour lancer le serveur :

1.2.1 Lancer le MSI ;

1.2.2 Installer Cassandra sous `C:\Program Files\DataStax-DDC\` ;

1.2.3 Vous n'êtes pas obligé de faire démarrer le service automatiquement ;

1.2.4 Pour faire démarrer le serveur cassandra : `C:\Program Files\DataStax-DDC\apache-cassandra\bin\cassandra.bat`
⚠Ne pas oublier de donner les droits d'accès au programme. Vérifier le JAVA_HOME.

L'exécutable `cassandra.bat` lance le serveur, il ne doit pas être éteint, et doit toujours être lancé avant de faire des requêtes sur la base de données.

L'UI `DevCenter` ou la console pour CQLSH (voir plus bas) devront être ouverts séparément.

1.2.2 Sous Linux/MacOSX

Pour lancer le serveur :

1.2.1 Décompresser l'archive ;

1.2.2 Le répertoire "bin" contient le script *cassandra*. Il faut donc ouvrir une `invite de commande` et lancer la commande avec le paramètre "cassandra" : `./cassandra`
⚠Il est possible que des droits d'administration soient demandés pour installer les fichiers nécessaires au bon fonctionnement de la table de routage de `cassandra`.

Il faut savoir que vous installez un serveur, il faudra donc le lancer avant toute utilisation de Cassandra. Pour cela, une console va s'ouvrir à ne surtout pas fermer.

La console pour CQLSH (voir plus bas) devra être ouverte séparément.

1.2.3 Installation de Cassandra sous Docker

Dans le cadre des Travaux Pratiques sur Cassandra, nous utiliserons "DOCKER" pour nous permettre de simplifier son utilisation sous MacOS. Tout d'abord, il faut se référer au guide d'installation de Docker et de l'outil "KITEMATIC". L'image officielle de "Cassandra" sera utilisée. Une fois téléchargée et instanciée, regarder le port d'écoute équivalent au 9042 (comme pour Oracle).

D'autres guides sont disponibles pour l'installation de Cassandra :

— Téléchargement de Cassandra : `https://cassandra.apache.org/download/`

— Guide complet windows :
`http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way`

— Guide complet Mac :

  — Avec DataStax :
  `http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way`

  — Avec l'installeur Python "*pip install*" :
  `https://dbglory.wordpress.com/2015/02/22/installing-cassandra-on-mac-os-x/`

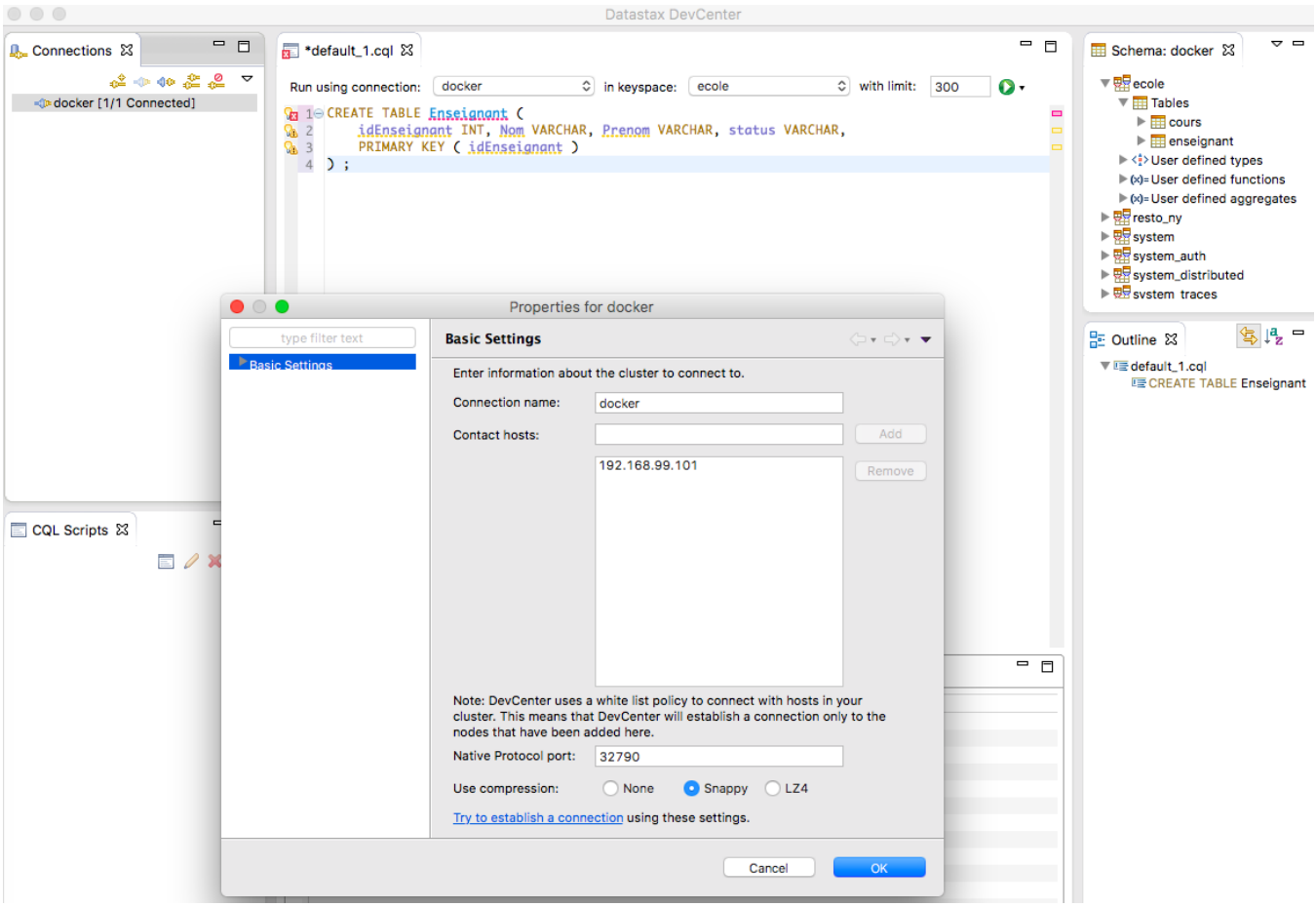
1.3 DevCenter

Si l'exécutable `DevCenter` n'est pas disponible dans l'archive téléchargée, vous pourrez la retrouver à cette adresse :
`https://academy.datastax.com/downloads`
C'est une application Java *Eclips-like* pour se connecter au serveur Cassandra.

DEVCENTER est un client pour Cassandra développé par `DataStax`. Vous pouvez l'utiliser quelquesoit le serveur installé, du moment que le port "9042" (ou équivalent redirigé sous Docker) soit ouvert.

Une fois le serveur lancé, vous pouvez donc démarrer DevCenter. Il vous suffit de créer une connexion comme ci-dessus.

Cette interface permettra de créer un Keyspace, puis une fois sélectionné (onglet keyspace) vous pourrez créer des column family et exécuter des requêtes CQL sur votre schéma.


Datastax DevCenter
Connections
docker [1/1 Connected]
*default_1.cql
Run using connection: docker
in keyspace: ecole
with limit: 300
CREATE TABLE Enseignant (
idEnseignant INT, Nom VARCHAR, Prenom VARCHAR, status VARCHAR,
PRIMARY KEY (idEnseignant)
) ;
Schema: docker
ecole
Tables
cours
enseignant
User defined types
User defined functions
User defined aggregates
resto_ny
system
system_auth
system_distributed
system_traces
Outline
default_1.cql
CREATE TABLE Enseignant
CQL Scripts
Properties for docker
type filter text
Basic Settings
Basic Settings
Enter information about the cluster to connect to.
Connection name: docker
Contact hosts:
Add
Remove
192.168.99.101
Note: DevCenter uses a white list policy to connect with hosts in your cluster. This means that DevCenter will establish a connection only to the nodes that have been added here.
Native Protocol port: 32790
Use compression: None Snappy LZ4
Try to establish a connection using these settings.
Cancel
OK

Chapitre 2

Création de la base de données

2.1 Keyspace

Avant d'interroger la base de données, il nous la créer. Pour commencer :

```
CREATE KEYSPACE IF NOT EXISTS ecole WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 3 };
```

Nous créons ainsi une base de données ECOLE pour laquelle le facteur de réplication est mis à 3 pour gérer les problèmes de reprise sur panne.

Vous pouvez maintenant sélectionner la base de données pour vos prochaines requêtes (à effectuer après chaque démarrage de CQLSH) :

```
USE ecole;
```

2.2 Column Family

Nous pouvons maintenant créer les tables (ou *Column Family* sous Cassandra) COURS et ENSEIGNANT à partir de notre schéma :

```
CREATE TABLE IF NOT EXISTS Cours (
        idCours INT, Intitule VARCHAR, Responsable INT, Niveau VARCHAR, nbHeuresMax INT, Coeff INT,
        PRIMARY KEY ( idCours )
) ;
CREATE INDEX IF NOT EXISTS fk_Enseignement_Enseignant_idx ON Cours ( Responsable ) ;
```

```
CREATE TABLE IF NOT EXISTS Enseignant (
        idEnseignant INT, Nom VARCHAR, Prenom VARCHAR, status VARCHAR,
        PRIMARY KEY ( idEnseignant )
) ;
```

Pour vérifier si les tables ont bien été créées (console CQLSH, dans DevCenter > menu de droite)

```
DESC ecole;
```

Nous pourrons voir le schéma des deux tables mais également des informations relatives au stockage dans Cassandra.

Nous pouvons maintenant y rajouter les données correspondantes :

```
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (1,'Introduction aux Bases de Donnees',1,'M1',30,3);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (2,'Immeubles de Grandes Hauteurs',4,'M1',30,2);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (3,'Production et distribution de biens et de ser',5,'M1',30,2);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (4,'Bases de Donnees Avancees',1,'M2',30,5);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (5,'Architecture des Systemes Materiel',6,'M2',8,1);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (6,'IT Business / Introduction',7,'M2',20,3);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (7,'IT Business / Strategie et Management',8,'M2',10,1);
```

```
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (1,'Travers','Nicolas','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (2,'Mourier','Pascale','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (3,'Boisson','Francois','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (4,'Mathieu','Eric','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (5,'Chu','Chengbin','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (6,'Boutin','Philippe','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (7,'Escribe','Julien','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (8,'Znaty','David','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (9,'Abal-Kassim','Cheik Ahamed','Vacataire');
```

Pour vérifier le contenu de la table :

```
SELECT * FROM Cours;
```

Chapitre 3

Interrogation de Cassandra

Dans ce chapitre, nous allons étudier l'interrogation de la base Cassandra avec CQL. Nous allons réutiliser notre base de données relationnelle vue dans les Travaux Pratiques précédents pour l'intégrer dans un environnement NoSQL. Nous allons pouvoir constater les différences avec une base de données relationnelle standard, en particulier sur les problèmes de jointures.

3.1 CQL : Cassandra Query Language

Pour interroger la base de données, la syntaxe CQL (pour *Cassandra Query Language*[1]) est fortement inspirée de SQL. Pour la suite des exercices, exprimer en CQL la requête demandée :

3.1.1 Liste tous les cours ;
Correction :

```
SELECT * FROM Cours;
```

3.1.2 Liste des intitulés de cours ;
Correction :

```
SELECT intitule FROM Cours;
```

3.1.3 Nom de l'enseignant n°4 ;
Correction :

```
SELECT nom FROM Enseignant WHERE idEnseignant=4;
```

3.1.4 Intitulé des cours du responsable n° 1 ;
Correction :

```
SELECT intitule FROM Cours WHERE Responsable=1 ;
```

3.1.5 Intitulé des cours dont le nombre d'heures maximum est égal à 30 ;
Correction : L'erreur suivante apparaît :`InvalidRequest: code=2200 [Invalid query] message="No secondary indexes`
Aucun index n'a été créé sur cours.nbHeuresMax. De fait, il ne peut effectuer une recherche.

```
SELECT intitule FROM Cours WHERE nbHeuresMax=30 ;
```

3.1.6 Intitulé des cours dont le responsable '1' et dont le niveau est 'M1' ;
Utiliser 'ALLOW FILTERING'.
Correction : Du fait de la présence de deux critères dans la requête, dont un est indexé, Cassandra ne peut prédire la taille du résultat et envoi un message d'alerte :`InvalidRequest: code=2200 [Invalid query] message="Cannot exec`
Il suffit de rajouter ALLOW FILTERING à la fin de la requête l'exécuter. Attention, dans le cadre de notre TP, cette requête est très peu couteuse. Ce n'est pas toujours le cas.

```
SELECT intitule FROM Cours WHERE Responsable=1 AND Niveau = 'M1' ALLOW FILTERING;
```

3.1.7 Intitulé des cours dont les id de responsable ont une valeur inférieure à 5 ;
Correction : Cette opération peut couter cher, il faut ajouter le 'ALLOW FILTERING' pour pouvoir l'exécuter.

3.1.8 Intitulé des cours dont l'identifiant est inférieur à 5 ;
Correction : Il n'est pas possible de faire ce type d'opération sur la clé de hachage.

1. Vous trouverez la syntaxe complète ici : `https://cassandra.apache.org/doc/latest/cql/dml.html#select`

3.1.9 Utiliser la fonction '`token()`' pour filtrer sur l'identifiant. Faites une requête pour consulter les token, et filtrer l'identifiant sur ce token : token(5) ;
Correction : On peut constater que les valeurs des tokens ne sont pas linéaires. Elles permettent de donner "l'angle" de placement des données sur l'anneau Cassandra.

La deuxième requête permet de retrouver toutes les données qui ont lemême token que la valeur 5.

Faire une requête d'inégalité sur ce critère permet uniquement de trouver un ensemble de données avec des token (angle dans l'anneau). La troisième requête illustre cela.

```
SELECT idCours, intitule, token(idCours) FROM Cours;

SELECT idCours, intitule, token(idCours) FROM Cours WHERE token(idCours) = token(5);

SELECT idCours, intitule, token(idCours) FROM Cours WHERE token(idCours) > token(1) and token(idCours
```

3.1.10 Compter le nombre de lignes retournées par la requête précédente ;
Utiliser 'COUNT(*)'
Correction :

```
SELECT COUNT(*) FROM Cours WHERE token(idCours) < 5;
```

3.2 Jointure & Dénormalisation

3.2.1 Donner les intitulés des cours dont le statut du responsable est 'Vacataire' ;
Correction : Naturellement, nous voudrions faire cela :

```
SELECT intitule FROM cours, enseignant
WHERE responsable = idenseignant and statut='Vacataire';

SELECT intitule FROM cours
WHERE responsable IN (SELECT idenseignant FROM Enseignant WHERE statut='Vacataire');
```

Sauf qu'il n'est pas possible de faire de jointures en CQL. Le langage ne donnera aucune possibilité à cette requête. Nous allons étudier la solution par la suite.

3.2.2 La jointure n'est pas possible avec CQL (à cause du stockage par hachage distribué). Nous allons créer une solution alternative en fusionnant les deux tables. Pour cela, il va nous allons choisir d'intégrer la table 'Enseignant' dans la table 'Cours', nous appellerons cette table '*coursEnseignant*'. Trois possibilités sont disponibles pour ce faire : *SET, LIST, MAP, SubType*. Choisissez la solution qui permettra de faire un filtrage sur le statut de l'enseignant.
Correction :

```
CREATE TABLE IF NOT EXISTS CoursEnseignant (
        idCours INT, Intitule VARCHAR,
        Responsable map<text, text>,
        Niveau VARCHAR, nbHeuresMax INT, Coeff INT,
        PRIMARY KEY ( idCours )
) ;
```

Bien sur, il est également possible d'intégrer les attributs directement à la table cours.

3.2.3 Insérer les données suivantes :

```
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (1,'Introduction aux Bases de Donnees',{'idenseignant':'1','nom':'Travers','prenom':'Nicolas','statut':'Vacataire'},'M1',30,3);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (2,'Immeubles de Grandes Hauteurs',{'idenseignant':'4','nom':'Mathieu','prenom':'Eric','statut':'Titulaire'},'M1',30,2);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (3,'Production et distribution de biens et de ser',{'idenseignant':'5','nom':'Chu','prenom':'Chengbin','statut':'Titulaire'},'M1',30,2);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (4,'Bases de Donnees Avancees',{'idenseignant':'1','nom':'Travers','prenom':'Nicolas','statut':'Vacataire'},'M2',30,5);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (5,'Architecture des Systemes Materiel',{'idenseignant':'6','nom':'Boutin','prenom':'Philippe','statut':'Titulaire'},'M2',8,1);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (6,'IT Business / Introduction',{'idenseignant':'7','nom':'Escribe','prenom':'Julien','statut':'Vacataire'},'M2',20,3);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (7,'IT Business / Strategie et Management',{'idenseignant':'8','nom':'Znaty','prenom':'David','statut':'Vacataire'},'M2',10,1);
```

3.2.4 Créer un index sur le Niveau de la table CoursEnseignant ;

Correction :

```
CREATE INDEX IF NOT EXISTS CoursEnseignant_Niveau ON CoursEnseignant ( Niveau ) ;
```

3.2.5 Donner les noms des responsables des cours (table CoursEnseignant) de niveau 'M1' ;

Correction : Nous pourrons remarquer qu'il n'est pas possible de projeter les valeurs d'une clé spécifique provenant d'un "*MAP*". Il est donc nécessaire de projeter l'ensemble de l'attribut '*responsable*'.

```
SELECT Responsable FROM CoursEnseignant where Niveau = 'M1';
```

3.2.6 Donner l'intitulé des cours dont le responsable est vacataire ;

Correction :

```
CREATE INDEX IF NOT EXISTS CoursEnseignant_statut ON CoursEnseignant ( responsable ) ;

SELECT Intitule FROM CoursEnseignant where Responsable CONTAINS 'Vacataire';
```

3.2.7 Nous allons inverser la fusion des tables en créant une table 'EnseignantCours' avec un sous-type 'Cours'. Pour pouvoir rajouter un ensemble de valeurs pour les cours d'un responsable, on peut utiliser : SET, MAP ou LIST.x Nous utiliserons ici le MAP. Créer le type 'Cours', la table 'EnseignantCours', et imbriquer les cours pour chaque enseignant.

Correction :

```
CREATE TYPE IF NOT EXISTS CoursType (idCours INT, Intitule VARCHAR, Niveau VARCHAR, nbHeuresMax INT, Coeff INT);

CREATE TABLE IF NOT EXISTS EnseignantCours (
        idEnseignant INT, Nom VARCHAR, Prenom VARCHAR, statut VARCHAR,
        cours map<int, frozen<CoursType>>,
        PRIMARY KEY ( idEnseignant )
) ;
```

3.2.8 Insérer les données suivantes :

```
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (1,'Travers','Nicolas','Vacataire',
 {1:{idcours:1,intitule:'Introduction aux Bases de Donnees',niveau:'M1',nbHeuresMax:30,Coeff:3},
 4:{idcours:4,intitule:'Bases de Donnees Avancees',niveau:'M2',nbHeuresMax:30,coeff:5}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (2,'Mourier','Pascale','Titulaire', {});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (3,'Boisson','Francois','Vacataire', {});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (4,'Mathieu','Eric','Titulaire',
 {4:{idcours:2,intitule:'Immeubles de Grandes Hauteurs',niveau:'M1',nbHeuresMax:30,coeff:2}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (5,'Chu','Chengbin','Titulaire', {});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (6,'Boutin','Philippe','Titulaire',
 {5:{idcours:5,intitule:'Architecture des Systemes Materiel',niveau:'M2',nbHeuresMax:8,coeff:1}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (7,'Escribe','Julien','Vacataire',
 {6:{idcours:6,intitule:'IT Business / Introduction',niveau:'M2',nbHeuresMax:20,coeff:3}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (8,'Znaty','David','Vacataire',
 {7:{idcours:7,intitule:'IT Business / Strategie et Management',niveau:'M2',nbHeuresMax:10,coeff:1}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,statut,cours) VALUES (9,'Abal-Kassim','Cheik Ahamed','Vacataire', {});
```

3.2.9 Créer un index sur le statut de la table EnseignantCours ;

Correction :

```
CREATE INDEX IF NOT EXISTS EnseignantCours_statut ON EnseignantCours ( statut ) ;
```

3.2.10 Donner les intitulés des cours dont le responsable est Vacataire ;
Correction : Ici non plus il n'est pas possible de projeter sur l'intitulé du (des) cours.

```
SELECT Cours FROM EnseignantCours where statut = 'Vacataire';
```

3.2.1 Imbrication : bonne pratique

De fait, Cassandra n'a pas la même manière de fonctionner qu'une base de données relationnelle. Il faut donc changer les habitudes de requêtes pour adapter les besoins à l'archictecture distribuée. Pour ce qui est de l'imbrication pour la jointure, nous allons maintenant nous concentrer sur les identifiants des cours pour un responsable, plutot que toutes les informations.

3.2.1 Créer une table EnseignantCours2 qui imbrique les identifiants de cours avec un "SET" ;
Correction :

```
CREATE TABLE IF NOT EXISTS EnseignantCours2 (
        idEnseignant INT, Nom VARCHAR, Prenom VARCHAR, statut VARCHAR,
        cours SET<INT>,
        PRIMARY KEY ( idEnseignant )
) ;
```

3.2.2 Insérer les données suivantes ;

```
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (1,'Travers','Nicolas','Vacataire', {1,4});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (2,'Mourier','Pascale','Titulaire', {});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (3,'Boisson','Francois','Vacataire', {});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (4,'Mathieu','Eric','Titulaire', {2,3});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (5,'Chu','Chengbin','Titulaire', {});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (6,'Boutin','Philippe','Titulaire',{5});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (7,'Escribe','Julien','Vacataire', {6});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (8,'Znaty','David','Vacataire', {7});
INSERT INTO EnseignantCours2 (idEnseignant,Nom,Prenom,statut,cours) VALUES (9,'Abal-Kassim','Cheik Ahamed','Vacataire', {});
```

3.2.3 Trouver l'enseignant qui est responsable du cours 2 ;
Correction :

```
CREATE INDEX IF NOT EXISTS cours2 on EnseignantCours2 (cours);

SELECT * FROM EnseignantCours2 WHERE cours CONTAINS 2;
```

3.2.4 Que doit-on faire si l'on souhaite trouver le responsable du cours de 'Skyline' ?
Correction : Il faut tout d'abord trouver l'identifiant de ce cours, puis faire une seconde requête pour trouver le responsable de ce cours. Cette solution comporte toutefois un risque si les deux requêtes retournent de nombreux résultats.

3.3 Indexation

3.3.1 La requête ci-dessous ne fonctionne qu'avec ALLOW FILTERING, trouver une solution pour la faire fonctionner sans ;

```
SELECT nom, prenom FROM enseignant WHERE statut='Vacataire';
```

Correction : Il faut pour cela créer un index sur 'statut' :

```
CREATE INDEX Enseignant_statut ON Enseignant ( statut ) ;
```

3.3.2 Créer un deuxième index sur cours.niveau. Exécutez de nouveau la requête . Regarder la trace générée par la requête (DevCenter : onglet 'Query Trace' à côté des résultats - Console : TRACING ON, avant la requête). Quel index a-t-il été utilisé pour cette requête ?

Correction : La trace est assez longue. On peut y constater le *parsing* de la requête, la gestion du *statement*, la *concurrence*, la sélection des *indexes candidats* (choix des cardinalités), exécution, fichiers (sstables), fusion de résultats...
L'index sur le responsable sera sélectionné : `Scanning with cours.fk_enseignement_enseignant_idx`

```
Execute CQL3 query,1477946885779000,172.17.0.2,COORDINATOR,0,
Parsing SELECT intitule FROM Cours WHERE Responsable=1 AND Niveau = 'M1' ALLOW FILTERING,1477946885779000,172.17.0.2,COORDINATOR,131,SharedPool-Worker-1
Preparing statement,1477946885779000,172.17.0.2,COORDINATOR,258,SharedPool-Worker-1
Computing ranges to query,1477946885780000,172.17.0.2,BOTH,903,SharedPool-Worker-1
Submitting range requests on 257 ranges with a concurrency of 257 (0.00234375 rows per range expected),1477946885780000,172.17.0.2,BOTH,1071,SharedPool-Worker-1
Submitted 1 concurrent range requests covering 257 ranges,1477946885804000,172.17.0.2,BOTH,25188,SharedPool-Worker-1
Executing indexed scan for (min(-9223372036854775808), min(-9223372036854775808)],1477946885805000,172.17.0.2,BOTH,26197,SharedPool-Worker-2
Candidate index mean cardinalities are CompositesIndexOnRegular{columnDefs=[ColumnDefinition{name=responsable, type=org.apache.cassandra.db.marshal.Int32Type, kind=REGULAR, componentIndex=0, inde
Executing single-partition query on cours.fk_enseignement_enseignant_idx,1477946885805000,172.17.0.2,BOTH,26400,SharedPool-Worker-2
Acquiring sstable references,1477946885805000,172.17.0.2,BOTH,26411,SharedPool-Worker-2
Merging memtable tombstones,1477946885805000,172.17.0.2,BOTH,26435,SharedPool-Worker-2
Partition index with 0 entries found for sstable 1,1477946885805000,172.17.0.2,BOTH,26609,SharedPool-Worker-2
Seeking to partition beginning in data file,1477946885805000,172.17.0.2,BOTH,26626,SharedPool-Worker-2
Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones,1477946885805000,172.17.0.2,BOTH,26747,SharedPool-Worker-2
Merging data from memtables and 1 sstables,1477946885805000,172.17.0.2,BOTH,26761,SharedPool-Worker-2
Read 2 live and 0 tombstone cells,1477946885805000,172.17.0.2,BOTH,26788,SharedPool-Worker-2
Executing single-partition query on cours,1477946885806000,172.17.0.2,BOTH,26859,SharedPool-Worker-2
Acquiring sstable references,1477946885806000,172.17.0.2,BOTH,26871,SharedPool-Worker-2
Merging memtable tombstones,1477946885806000,172.17.0.2,BOTH,26887,SharedPool-Worker-2
Key cache hit for sstable 1,1477946885806000,172.17.0.2,BOTH,26916,SharedPool-Worker-2
Seeking to partition beginning in data file,1477946885806000,172.17.0.2,BOTH,26925,SharedPool-Worker-2
Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones,1477946885806000,172.17.0.2,BOTH,26998,SharedPool-Worker-2
Merging data from memtables and 1 sstables,1477946885806000,172.17.0.2,BOTH,27010,SharedPool-Worker-2
Read 1 live and 0 tombstone cells,1477946885806000,172.17.0.2,BOTH,27036,SharedPool-Worker-2
Executing single-partition query on cours,1477946885806000,172.17.0.2,BOTH,27174,SharedPool-Worker-2
Acquiring sstable references,1477946885806000,172.17.0.2,BOTH,27186,SharedPool-Worker-2
Merging memtable tombstones,1477946885806000,172.17.0.2,BOTH,27199,SharedPool-Worker-2
Key cache hit for sstable 1,1477946885806000,172.17.0.2,BOTH,27222,SharedPool-Worker-2
Seeking to partition beginning in data file,1477946885806000,172.17.0.2,BOTH,27230,SharedPool-Worker-2
Skipped 0/1 non-slice-intersecting sstables, included 0 due to tombstones,1477946885806000,172.17.0.2,BOTH,27247,SharedPool-Worker-2
Merging data from memtables and 1 sstables,1477946885806000,172.17.0.2,BOTH,27256,SharedPool-Worker-2
Read 1 live and 0 tombstone cells,1477946885806000,172.17.0.2,BOTH,27286,SharedPool-Worker-2
Scanned 1 rows and matched 1,1477946885806000,172.17.0.2,BOTH,27317,SharedPool-Worker-2
Request complete,1477946885810041,172.17.0.2,COORDINATOR,31041,
```

3.3.3 Nous souhaiterions optimiser la requête suivante :

```
SELECT idCours, Intitule FROM Cours WHERE Responsable=X;
```

Actuellement, un index est postionné sur l'attribut "Responsable", toutefois plusieurs noeuds sont interrogés. Créer une table "Cours2" qui améliore l'interrogation de cette requête (pour n'en interroger qu'un seul noeud).
Correction : Il faut pour cela changer le clustering. Actuellement, il est sur "idCours" : PRIMARY KEY(idCours).
Il suffit pour cela de :

(a) Rajouter "responsable" dans la clé primaire

(b) Mettre responsable comme clé de regroupement

⇒

```
CREATE TABLE Cours2 (
        idCours INT, Intitule VARCHAR, Responsable INT,
        Niveau VARCHAR, nbHeuresMax INT, Coeff INT, Niveau VARCHAR,
        PRIMARY KEY ( (Responsable), idCours )
) ;
```

3.3.4 Exécuter la requête suivante sur "Cours" et "Cours2". Regarder la trace générée par la requête. Quelles différences constatez-vous ?

```
SELECT idCours, Intitule FROM Cours WHERE Responsable=1;
```

Correction : Cours : **Executing read on ecole.cours using index fk_enseignement_enseignant_idx**. On peut également constater la récupération de données provenant de plusieurs 'sstable'
Cours2 : **Executing single-partition query on cours2**. Un seul sstable est contacté, celui contenant les 3 lignes. Le temps d'exécution est également plus faible dans le deuxième cas.

3.4 Mises à jour

3.4.1 Mettre à jour la table EnseignantCours pour ajouter le cours {'idcours' :10,'intitule' :'Cassandra','niveau' :'M2','coeff' :1} à l'enseignant 1

Correction :

```
UPDATE EnseignantCours SET cours = cours +
     {10 : {'idcours':10,'intitule':'Cassandra','niveau':'M2',nbHeuresMax:10,'coeff':1}}
    WHERE idEnseignant = 1;
```

3.4.2 Mettre à jour la table EnseignantCours en remplaçant le cours 7 de l'enseignant n°8 avec le cours {idcours :7,intitule :'IT Business / Block Chain Management',niveau :'M2',quota :30,coeff :1} à l'enseignant 1

Correction :

```
UPDATE EnseignantCours SET cours[7] =
     {idcours:7,intitule:'IT Business / Block Chain Management',niveau:'M2',quota:30,coeff:1}
     WHERE idEnseignant = 8;
```

3.4.3 Supprimer les cours (pas les enseignants) dont l'idEnseignant=1 ;

Correction :

```
DELETE cours FROM EnseignantCours WHERE idEnseignant=1;
```

3.4.4 Supprimer les enseignants dont le statut est 'vacataire' ;

Correction : La requête ci-dessous n'est pas possible. En effet, il est nécessaire de supprimer les enregistrements par clés primaires. De fait, il faudra faire une requête pour les vacataires, puis de supprimer les clés primaires de chaque enregistrement.

```
DELETE FROM EnseignantCours WHERE statut='Vacataire';
```

3.5 User Define Aggregate function (Map/Reduce)

Nous allons tester la fonction '*Average*' donnée en exemple dans le cours. Toutefois, il faut auparavant activer le paramètre '*enable_user_defined_functions*' pour activer les fonctions utilisateurs (donc Map/Reduce).

Pour ce faire :

- Editer le fichier '`cassandra.yaml`' dans le répertoire de configuration de cassandra
- Chercher la chaine '`user_define`'
- Modifier le paramètre en le passant à '`true`' (bien mettre un espace entre ' :' et 'true')
- Sauvegarder le fichier de configuration
- Redémarrer Cassandra

3.5.1 Créer la fonction *Map*

```
CREATE OR REPLACE FUNCTION avgState ( state tuple<int,bigint>, val int )
        CALLED ON NULL INPUT RETURNS tuple<int,bigint> LANGUAGE java
        AS 'if (val !=null) { state.setInt(0, state.getInt(0)+1);
                state.setLong(1, state.getLong(1)+val.intValue()); }
                return state;';
```

3.5.2 Créer la fonction *Reduce*

```
CREATE OR REPLACE FUNCTION avgFinal ( state tuple<int,bigint> )
        CALLED ON NULL INPUT RETURNS double LANGUAGE java
        AS 'double r = 0;
                if (state.getInt(0) == 0) return null;
                r = state.getLong(1);
                r/= state.getInt(0);
                return Double.valueOf(r);';
```

3.5.3 Créer la fonction d'agrégat utilisateur (*UDA*)

```
CREATE AGGREGATE IF NOT EXISTS average ( int )
        SFUNC avgState STYPE tuple<int,bigint>
        FINALFUNC avgFinal INITCOND (0,0);
```

3.5.4 Calculer la moyenne des '*nbHeuresMax*' pour la table 'coursEnseignant'

Correction :

```
SELECT average(nbHeuresmax) FROM coursenseignant WHERE niveau='M1'
```

3.5.5 La même mais pour des responsables 'Vacataire'

Correction :

```
SELECT average(nbHeuresmax) FROM coursenseignant WHERE responsable CONTAINS 'Vacataire';
```

3.5.1 Bonus

3.5.1 Créer une fonction Map/Reduce pour faire l'équivalent d'un "*GROUP BY + COUNT*" sur du texte pour la requête suivante :

```
SELECT countGroup(niveau) from CoursEnseignant;
```

Le paramètre du 'state' doit être un '*map<text, int>*'.

Correction :

```
CREATE OR REPLACE FUNCTION countGroupState ( state map<text, int>, val1 text)
        CALLED ON NULL INPUT RETURNS map<text, int> LANGUAGE java
        AS 'Integer count = (Integer)state.get(val1);
                if(count == null)
                        count = 0;
                count ++;
                state.put(val1, count);
                return state;';

CREATE OR REPLACE AGGREGATE countGroup ( text)
        SFUNC countGroupState STYPE map<text, int>
        INITCOND {};
```