



ÉCOLE
D'INGÉNIEURS
PARIS-LA DÉFENSE

Intégration avec une base Cassandra

Travaux Pratiques

ESiLV

nicolas.travers (at) cnam.fr

1	Guide d'utilisation de Cassandra	3
1.1	Téléchargement	3
1.2	Installation de Cassandra	3
1.2.1	Sous Windows	3
1.2.2	Sous Linux/MacOSX	3
1.2.3	Installation de Cassandra sous Docker	4
1.3	DevCenter	4
2	Création de la base de données	6
2.1	KEYSPACE	6
2.2	COLUMN FAMILY	6
3	Interrogation de Cassandra	7
3.1	CQL : Cassandra Query Language	7
3.2	Jointure & Dénormalisation	7
3.2.1	Imbrication : bonne pratique	8
3.3	Indexation	8
3.4	Mises à jour	9
3.5	User Define Aggregate function (Map/Reduce)	9
3.5.1	Bonus	9

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 distribué 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’installateur 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 :

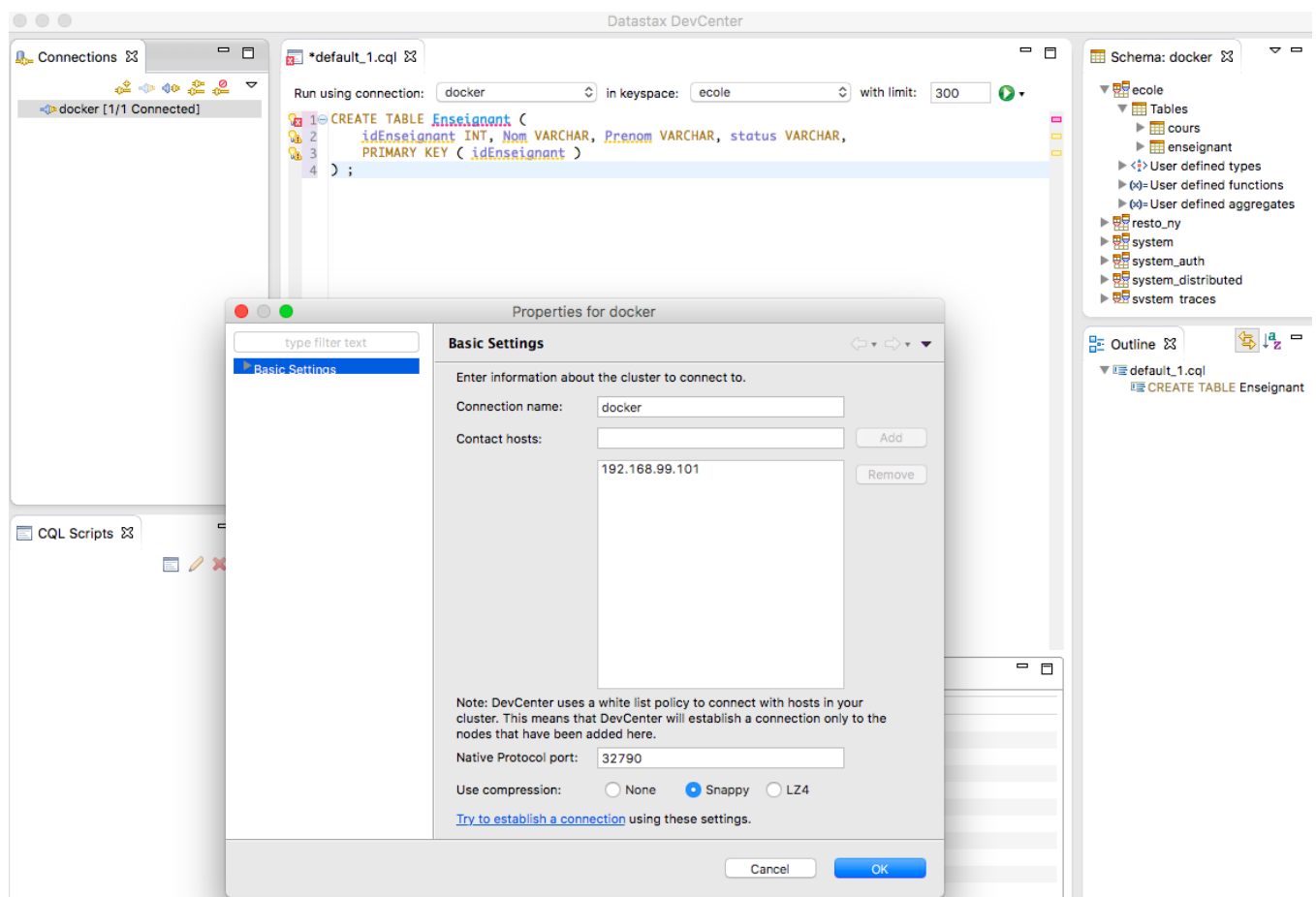
<https://academy.datastax.com/downloads>

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.



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 pourrions 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;
```

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 ;

3.1.2 Liste des intitulés de cours ;

3.1.3 Nom de l'enseignant n°4 ;

3.1.4 Intitulé des cours du responsable n° 1 ;

3.1.5 Intitulé des cours dont le responsable '1' et dont le niveau est 'M1' ;

Utiliser 'ALLOW FILTERING'.

3.1.6 Intitulé des cours dont les id de responsable ont une valeur inférieure à 5 ;

3.1.7 Intitulé des cours dont l'identifiant est inférieur à 5 ;

3.1.8 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) ;

3.1.9 Compter le nombre de lignes retournées par la requête précédente ;

Utiliser 'COUNT(*)'

3.2 Jointure & Dénormalisation

3.2.1 Donner les intitulés des cours dont le statut du responsable est 'Vacataire' ;

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 falloir 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.

3.2.3 Insérer les données suivantes :

```
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
(1,'Introduction aux Bases de Donnees',4,'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',4,'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',4,'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',4,'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',4,'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',4,'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',4,'idenseignant':'8','nom':'Znaty','prenom':'David','statut':'Vacataire',M2,10,1);
```

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

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

3.2.6 Donner l'intitulé des cours dont le responsable est 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.

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

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;

3.2.10 Donner les intitulés des cours dont le responsable est 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'architecture distribuée. Pour ce qui est de l'imbrication pour la jointure, nous allons maintenant nous concentrer sur les identifiants des cours pour un responsable, plutôt que toutes les informations.

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

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;

3.2.4 Que doit-on faire si l'on souhaite trouver le responsable du cours de 'Skyline'?

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';
```

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?

3.3.3 Nous souhaiterions optimiser la requête suivante :

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

Actuellement, un index est positionné 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).

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;
```


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
- 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
- 3.4.3 Supprimer les cours (pas les enseignants) dont l'idEnseignant=1;
- 3.4.4 Supprimer les enseignants dont le statut est '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 chaîne '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'

3.5.5 La même mais pour des responsables '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>'.