

SHAKE THE FUTURE



Bases de Données

NoSQL avec Cassandra

JY Martin

Plan

- 1 Généralités
- 2 Le modèle de données
- 3 Introduction à CQL
- 4 Concevoir un Keyspace
- 5 Connexion à Cassandra

Cassandra

C*

- Base de données NoSQL.
- Mise en œuvre, à l'origine, par Facebook pour la recherche InBox.
- Actuellement maintenue par Apache.
- C'est aujourd'hui une base de données distribuées assez répandue.

<http://cassandra.apache.org>

Un peu similaire à Riak.

C* ...

- Base de Données
- Distribuée
- Hautement disponible
- Orientée Colonnes
- A consistance éventuelle
- Ajustable

Plus concrètement

- Base de données NoSQL Distribuée.
Ensemble de nœuds formant un cluster
- Décentralisée –
 - Pas de gestionnaire ou de nœud principal
 - Tous les nœuds sont égaux
- Forte tolérance aux pannes
 - Réplication des informations sur d'autres nœuds
- NoSQL format colonne
 - Modèle riche
 - Élastique

Le logiciel

- Open Source, sous Licence **Apache 2.0**
- Ecrit en JAVA
- Interface Thrift - interface historique d'interaction
Ruby, Perl, Python, Scala, Java, ...
- Langage CQL (Cassandra Query Language)
Driver dans beaucoup de langages

Qui ? Pour quoi ?

Apple utilise un cluster de 75000 noeuds Cassandra (Maps, iCloud, ...)

Plus de 10 petabytes (1 petabyte = 10^{15} octets)

<https://www.techrepublic.com/article/apples-secret-nosql-sauce-includes-a-hefty-dose-of-cassandra/>

NetFlix : 2000 nœuds (1 million d'écritures par seconde)

<https://www.datastax.com/tag/use-case>

A quoi sert Cassandra ?

- Pourquoi ?
 - Gestion de **GROS** volumes de données (Tera, Petabytes)
 - Problématique de charge, de disponibilité
 - Ecrire des centaines de milliers de valeurs par seconde
 - Latence (temps de réponse) faible
 - Ajouter des machines à la volée

A quoi sert Cassandra ?

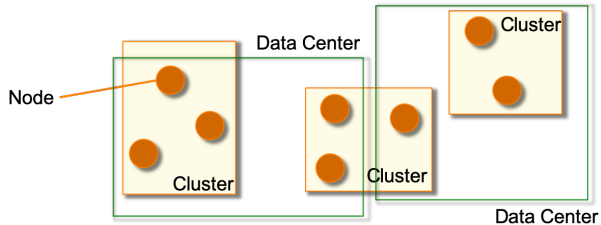
- Comment ?
 - Distribution de données (configurable : ajout de nœuds)
 - Réplication de données (configurable)
 - Cohérence (configurable)
- Pas d'architecture maitre/esclave

Par rapport au CAP

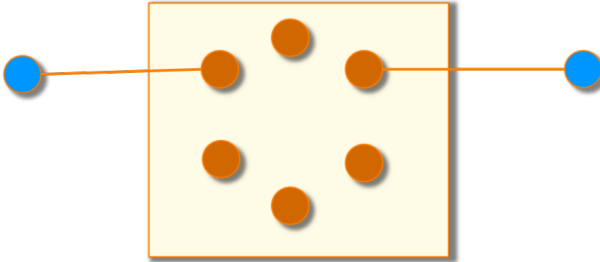


Architecture générale

- Nœud = une instance permettant de stocker les données
- Cluster = Ensemble logique contenant un certain nombre de nœuds
- Data Center = ensemble de nœuds

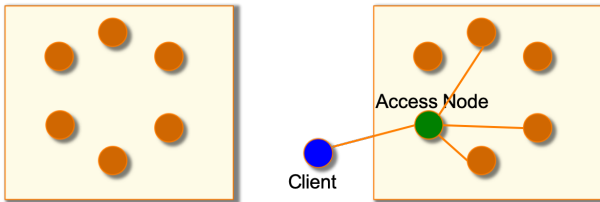


Fonctionnement général



On peut accéder à n'importe quel nœud

Comportement



Le nœud d'accès devient un "Coordonateur". Il détermine à quels autres nœuds il transmettra la requête.

Le protocole GOSSIP

Cassandra utilise le protocole **GOSSIP** pour fonctionner.

Chaque noeud discute en P2P avec ses voisins pour avoir une visibilité locale, s'harmoniser avec les autres, vérifier l'état des autres nœuds, contrôler les actions à faire sur les données, ...

Le modèle Colonne

Cassandra est basé sur un modèle **Colonne**.

Chaque noeud du cluster conserve des lignes de données en fonction de clés de partition.

Contrairement aux serveurs relationnels, les données sont mémorisées colonne par colonne et pas ligne par ligne. Ce mode de fonctionnement est toutefois invisible pour l'utilisateur.

Avantage : quand on remonte une colonne, toute la colonne est placée dans le cache.

Inconvénient : il vaut mieux éviter de remonter beaucoup de colonnes lors des requêtes.

La distribution des données

Nœud	Ligne	Clé	Valeur
1	3	1	XXXX
1	6	2	XXXX
1	7	3	XXXX
2	1	4	XXXX
2	2	5	XXXX
3	5	6	XXXX
3	4	7	XXXX

Je cherche la valeur correspondant à la clé 5

La distribution des données

Nœud	Ligne	Clé	Valeur
1	3	1	xxxx
1	6	2	xxxx
1	7	3	xxxx
2	1	4	xxxx
2	2	5	xxxx
3	5	6	xxxx
3	4	7	xxxx

Nœud
Indisponible

Je cherche la valeur correspondant à la clé 5

Donnée non accessible ...

La distribution des données

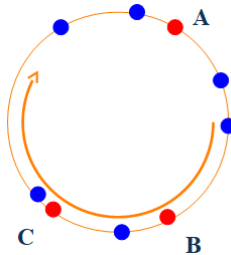
Duplication des données sur les nœuds.

Nœud	Ligne	Clé	Valeur
1	3	1	xxxx
1	6	2	xxxx
1	7	3	xxxx
1	1	4	xxxx
1	2	5	xxxx
2	1	4	xxxx
2	2	5	xxxx
2	5	6	xxxx
2	4	7	xxxx
3	5	6	xxxx
3	4	7	xxxx
3	3	1	xxxx
3	6	2	xxxx
3	7	3	xxxx

Il faudra que les nœuds 1 et 2 tombent pour que la donnée correspondant à la clé 5 soit inaccessible (facteur de réplication).

Le HashRing

Les serveurs et données sont répartis sur un anneau



Utilisation de la fonction de hachage Murmur3

Les données sont sur le serveur qui les précèdent sur l'anneau du HashRing.

Les réplifications sont faites sur les serveurs qui suivent sur l'anneau

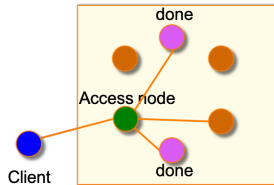
Synchronisation des données

Et si un nœud est modifié....

	Nœud	Ligne	Clé	Valeur
	1	3	1	xxxx
	1	6	2	xxxx
	1	7	3	xxxx
	1	1	4	xxxx
->	1	2	5	xxxx
	2	1	4	xxxx
->	2	2	5	yyyy
	2	5	6	xxxx
	2	4	7	xxxx
	3	5	6	xxxx
	3	4	7	xxxx
	3	3	1	xxxx
	3	6	2	xxxx
	3	7	3	xxxx

Comment savoir quelle est la bonne valeur ?

Degré de consistance



Le coordonateur transmet la donnée à modifier au nœud concerné.
GOSSIP gère les échanges entre serveurs pour que les nœuds se synchronisent. Si le **degré de consistance** est inférieur ou égal au nombre d'écritures, la donnée est considérée comme modifiée.
Les données sont ensuite réparées en tâche de fond par GOSSIP.

Synchronisation des données

	Nœud	Ligne	Clé	Valeur	Timestamp
	1	3	1	xxxx	3/09 à 9h01
	1	6	2	xxxx	3/09 à 9h01
	1	7	3	xxxx	3/09 à 9h01
	1	1	4	xxxx	3/09 à 9h01
->	1	2	5	xxxx	3/09 à 9h01
	2	1	4	xxxx	3/09 à 9h01
->	2	2	5	yyyy	7/09 à 12h00
	2	5	6	xxxx	3/09 à 9h01
	2	4	7	xxxx	3/09 à 9h01
	3	5	6	xxxx	3/09 à 9h01
	3	4	7	xxxx	3/09 à 9h01
	3	3	1	xxxx	3/09 à 9h01
	3	6	2	xxxx	3/09 à 9h01
	3	7	3	xxxx	3/09 à 9h01

Le timestamp indique quelle est la dernière valeur.

Synchronisation des données

Supprimer une donnée

Nœud	Ligne	Clé	Valeur	Timestamp
1	3	1	xxxx	3/09 à 9h01
1	6	2	xxxx	3/09 à 9h01
1	7	3	xxxx	3/09 à 9h01
1	1	4	xxxx	3/09 à 9h01
1	2	5	xxxx	3/09 à 9h01
2	1	4	xxxx	3/09 à 9h01
2	2	5	yyyy	3/09 à 9h01
2	5	6	xxxx	3/09 à 9h01
2	4	7	xxxx	3/09 à 9h01
3	5	6	xxxx	3/09 à 9h01
3	4	7	xxxx	3/09 à 9h01
3	3	1	xxxx	3/09 à 9h01
3	6	2	xxxx	3/09 à 9h01
3	7	3	xxxx	3/09 à 9h01

La suppression se fait sur tous les nœuds concernés en même temps

Synchronisation des données

Nœud	Ligne	Clé	Valeur	Timestamp
1	3	1	xxxx	3/09 à 9h01
1	6	2	xxxx	3/09 à 9h01
1	7	3	xxxx	3/09 à 9h01
1	1	4	xxxx	3/09 à 9h01
1	2	5	xxxx	3/09 à 9h01
2	1	4	xxxx	3/09 à 9h01
2	2	5	yyyy	3/09 à 9h01
2	5	6	xxxx	3/09 à 9h01
3	5	6	xxxx	3/09 à 9h01
3	3	1	xxxx	3/09 à 9h01
3	6	2	xxxx	3/09 à 9h01
3	7	3	xxxx	3/09 à 9h01

Synchronisation des données

Supprimer une donnée... avec un nœud indisponible

Nœud	Ligne	Clé	Valeur	Timestamp
1	3	1	xxxx	3/09 à 9h01
1	6	2	xxxx	3/09 à 9h01
1	7	3	xxxx	3/09 à 9h01
1	1	4	xxxx	3/09 à 9h01
1	2	5	xxxx	3/09 à 9h01
2	1	4	xxxx	3/09 à 9h01
2	2	5	yyyy	3/09 à 9h01
2	5	6	xxxx	3/09 à 9h01
2	4	7	xxxx	3/09 à 9h01
3	5	6	xxxx	3/09 à 9h01
3	4	7	xxxx	3/09 à 9h01
3	3	1	xxxx	3/09 à 9h01
3	6	2	xxxx	3/09 à 9h01
3	7	3	xxxx	3/09 à 9h01

Nœud
Indisponible

Synchronisation des données

Supprimer une donnée... avec un nœud indisponible

Nœud	Ligne	Clé	Valeur	Timestamp
1	3	1	xxxx	3/09 à 9h01
1	6	2	xxxx	3/09 à 9h01
1	7	3	xxxx	3/09 à 9h01
1	1	4	xxxx	3/09 à 9h01
1	2	5	xxxx	3/09 à 9h01
2	1	4	xxxx	3/09 à 9h01
2	2	5	yyyy	3/09 à 9h01
2	5	6	xxxx	3/09 à 9h01
2	4	7	xxxx	3/09 à 9h01
3	5	6	xxxx	3/09 à 9h01
3	3	1	xxxx	3/09 à 9h01
3	6	2	xxxx	3/09 à 9h01
3	7	3	xxxx	3/09 à 9h01

Nœud
Indisponible

La donnée est supprimée... sur tous les nœuds disponibles.

Synchronisation des données

Supprimer une donnée... avec un nœud indisponible

Nœud	Ligne	Clé	Valeur	Timestamp
1	3	1	xxxx	3/09 à 9h01
1	6	2	xxxx	3/09 à 9h01
1	7	3	xxxx	3/09 à 9h01
1	1	4	xxxx	3/09 à 9h01
1	2	5	xxxx	3/09 à 9h01
2	1	4	xxxx	3/09 à 9h01
2	2	5	yyyy	3/09 à 9h01
2	5	6	xxxx	3/09 à 9h01
2	4	7	xxxx	3/09 à 9h01
3	5	6	xxxx	3/09 à 9h01
3	3	1	xxxx	3/09 à 9h01
3	6	2	xxxx	3/09 à 9h01
3	7	3	xxxx	3/09 à 9h01

Nœud
Disponible

Le nœud redevient disponible.

Synchronisation des données

Supprimer une donnée... avec un nœud indisponible

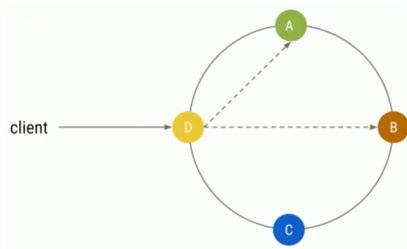
Nœud	Ligne	Clé	Valeur	Timestamp
1	3	1	xxxx	3/09 à 9h01
1	6	2	xxxx	3/09 à 9h01
1	7	3	xxxx	3/09 à 9h01
1	1	4	xxxx	3/09 à 9h01
1	2	5	xxxx	3/09 à 9h01
2	1	4	xxxx	3/09 à 9h01
2	2	5	yyyy	3/09 à 9h01
2	5	6	xxxx	3/09 à 9h01
2	4	7	xxxx	3/09 à 9h01
3	5	6	xxxx	3/09 à 9h01
3	4	7	xxxx	3/09 à 9h01
3	3	1	xxxx	3/09 à 9h01
3	6	2	xxxx	3/09 à 9h01
3	7	3	xxxx	3/09 à 9h01

Nœud
Disponible

La donnée réapparaît.

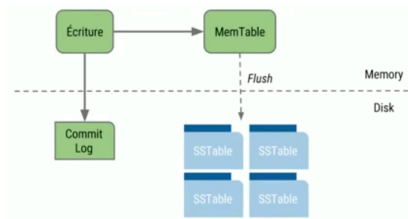
Optimisation des écritures

Quand on écrit une donnée, elle est transmise à un nœud.
Via le protocole GOSSIP, l'information est transmise / synchronisée avec les autres nœuds susceptible d'accueillir la donnée.

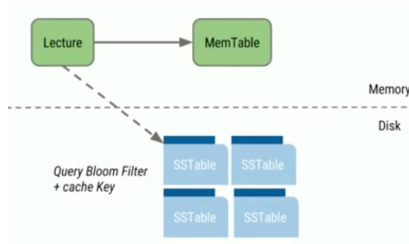


Optimisation des écritures

Techniquement...



Optimisation des lectures : le Bloom Filter



Bloom Filter : probabilité de présence d'une donnée. Mémorisé comme un tableau d'entier (0,1)

Le Bloom Filter

0	0	0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---	---	---

On prend une donnée. On la passe dans 2 filtres de hashage. On met à 1 sur les cases correspondantes.

Pourquoi 2 ? parce que 2 données différentes peuvent correspondre à la même valeur hashée.

0	0	0	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---

Le Bloom Filter

Lorsqu'on recherche une donnée, on applique les 2 filtres de hashage et on regarde les valeurs.

0	0	0	0	1	0	0	0	1	0
---	---	---	---	---	---	---	---	---	---

- L'une des valeurs de hashage correspond à un 0 : la donnée ne peut pas être présente
- Les 2 sont à 1 : il est possible que la donnée soit présente.

Du coup C*...

- Base de Données : NoSQL
- Distribuée : Cluster de serveurs
- Hautement disponible : Réplication + GOSSIP
- Orientée Colonnes : modèle de stockage d'information (interne)
- A consistance éventuelle : Lors d'une requête on garantit qu'on fournira une réponse mais ça ne sera pas obligatoirement la plus à jour
- Ajustable : on peut ajouter des noeuds à la volée

Remarques

- Impossibilité de faire des RollBack. Toute modification sera enregistrée, mais plus ou moins rapidement.
- A un instant donné, les informations répliquées ne sont pas toujours identiques. Elles le seront lorsque les nœuds seront synchronisés par GOSSIP.
- Une ligne supprimée peut apparaître, mais vide lors d'une requête. Les colonnes sont marquées comme "à supprimer" mais tant que la suppression n'est pas effective, la ligne existe toujours.
- Une information (colonne) supprimée peut très bien réapparaître au bout de quelques temps. Il suffit qu'un nœud soit HS suffisamment longtemps pour faire réapparaître l'information quand il revient dans le circuit.

En clair

- Des effets... ennuyeux
 - Perte des propriétés ACID
 - Pas de jointures
 - Requêtes pré-définies, pas de tri autres que ceux explicitement prévus
 - Quelques effets de bord... perturbants
- Mais
 - Permet de mémoriser de très grosses quantités de données
 - données réparties sur des clusters
 - Structuration -> accès rapide aux informations
 - Informations modulable en fonction des circonstances
 - Traitements possibles via Hadoop (Map-Reduce)

Conclusion

- Ne pas utiliser si vous avez besoin des propriétés ACID
Restez sur du relationnel
- Ne pas utiliser si vous ne connaissez pas a priori vos requêtes
- Attention à votre modélisation : vous n'avez pas de jointure
-> Modélisation spécifique
- Permet de gérer rapidement de grosses masses d'informations
- Priorité à la disponibilité des données, permet des accès temps-réel
- Extensible facilement
- Forte tolérance aux pannes

Plan

- 1 Généralités
- 2 Le modèle de données
- 3 Introduction à CQL
- 4 Concevoir un Keyspace
- 5 Connexion à Cassandra

Le modèle de données = format colonnes

- Keyspace = le conteneur de données
En relationnel : keyspace = la base, le schéma
- Le Keyspace contient au moins 1 famille de colonnes
En relationnel : famille de colonnes = la table
- Chaque famille de colonnes contient des lignes
En relationnel : la ligne = une ligne, un tuple
- Chaque ligne contient une clé et des colonnes
En relationnel une colonne = un attribut
- La colonne est l'entité de base

La colonne

- Plus petite unité mémorisée
- Trio (clé – valeur - timestamp)
 - Clé = le nom (max 64 Ko)
 - Valeur = le contenu (max 2 Go)
 - Facultative
 - Possède un type
 - Timestamp : pour déterminer la dernière version
- Comporte :
 - 1 "comparator" = type de données du nom de la colonne
 - 1 "validator" = type de données de la valeur de la colonne

clé
valeur
timestamp

La ligne

- Ensemble de colonnes (max 2 milliards)
- Identifié par une clé (max 64 Ko)

Remarques

- Ne comportent pas forcément le même nombre de colonnes
- Pas forcément les mêmes colonnes (peuvent être différentes à la création ou dans le temps)

Pascal	Nom	Prénom	Adresse	Tel
	DAVID	Pascal	St Nazaire	+33 6 ...
Carole	Nom	Prénom	Fonction	
	ROQUES	Carole	Designer	

La famille de colonne

Regroupement logique de lignes

Personne				
Pascal	Nom	Prénom	Adresse	Tel
	DAVID	Pascal	St Nazaire	+33 6 ...
Carole	Nom	Prénom	Fonction	
	ROQUES	Carole	Designer	

Le Keyspace

Regroupement logique de familles de colonnes

Personne				
Pascal	Nom	Prénom	Adresse	Tel
	DAVID	Pascal	St Nazaire	+33 6 ...
Carole	Nom	Prénom	Fonction	
	ROQUES	Carole	Designer	
Service				

Distribution des données

Problématique

La base est répartie sur plusieurs nœuds, plusieurs clusters.
Comment faire en sorte qu'une requête ne soit pas obligée de s'exécuter sur tous les nœuds pour gagner en temps d'exécution ?

Solution

Il faut savoir où sont potentiellement les données recherchées

- La répartition des données sur les nœuds doit être planifiée
- utilisation d'un mécanisme type "table de hachage" pour répartir les éléments.
- utilisation de clés de partitionnement

Le Partitionnement

Les lignes sont regroupées en “partitions”.

Une partition est toujours mémorisée entièrement sur un nœud.

On ajoute des clés de partitions pour répartir les lignes sur le cluster

Les clés de partitions sont calculées comme des parties de la clé primaire.

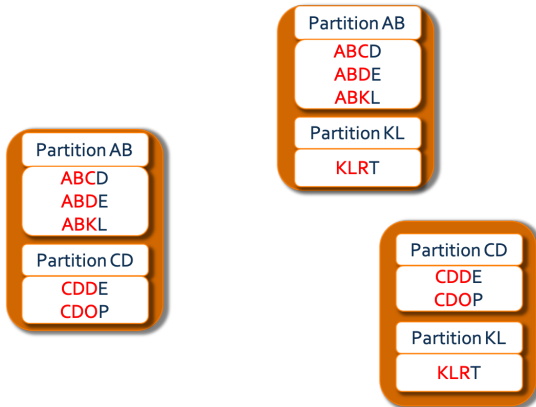
Le Partitionnement

- On découpe la clé primaire en 2 éléments
 - Une partie commune à un certain nombre de clés, la clé de partitionnement
 - Le reste de la clé, ou clé de clustering
- On regroupe sur un même nœud toutes les lignes avec la même clé de partitionnement
- Comme les valeurs de la clé de partitionnement sont identiques sur une partition, on les mémorise 1 fois, au niveau de la partition.
- Les lignes sont ordonnées dans une partition en suivant la clé de clustering

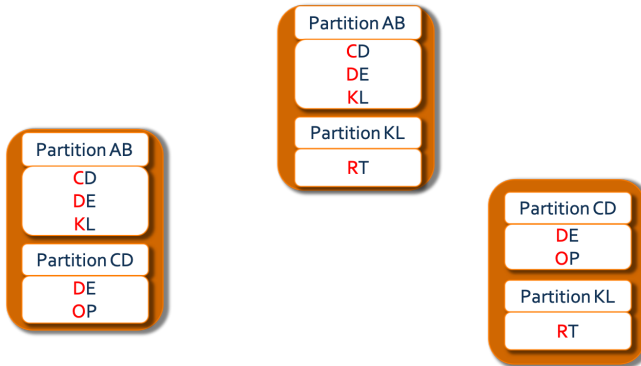
Le Partitionnement

AB CD AB DE AB KL	Partition AB
CD DE CD OP	Partition CD
KL RT	Partition KL

Le Partitionnement



Le Partitionnement



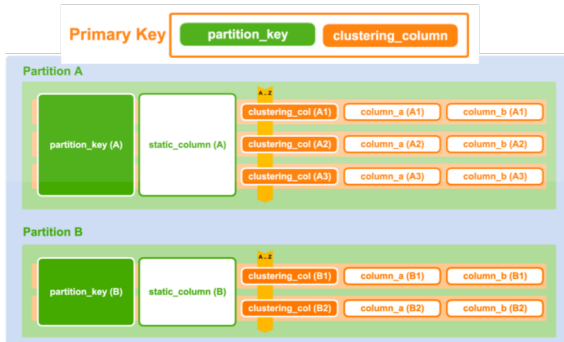
Les colonnes statiques

Certaines colonnes, bien que ne faisant pas partie de la clé de partitionnement, ont la même valeur pour toutes les lignes.

- > Pourquoi les mémoriser pour chaque ligne ?
- > Mémorisation au niveau de la partition
- > colonnes **statiques**

Partitionnement

Au final, on obtient :



Plan

- 1 Généralités
- 2 Le modèle de données
- 3 Introduction à CQL**
- 4 Concevoir un Keyspace
- 5 Connexion à Cassandra

CQL = Cassandra Query Language

- Langage inspiré de SQL
- Actuellement version 3
- Spécifique à Cassandra
- Utilisable en ligne de commande depuis cqlsh
- Quelques outils graphiques, mais limités

<http://cassandra.apache.org/doc/latest/cql/index.html>

Les commandes

- Commandes pour la création
- Commandes de manipulation : CRUD
- Non sensible à la casse, sauf si vous indiquez des noms entre guillemets
table = TABLE <> "Table"
- Les chaînes de caractères sont entre apostrophes.
L'apostrophe est doublée dans une chaîne de caractères

Types de données manipulées

- int, smallint, bigint, varint, float, double, boolean
- text, varchar
- date, time, timestamp
- counter
- uuid, timeuuid
- blob
- list, map, set
- ...
- types définis par l'utilisateur

<http://cassandra.apache.org/doc/latest/cql/types.html>

Quelques commandes

- CREATE ROLE nomRole WITH PASSWORD = 'motDePasse' AND SUPERUSER = true | false AND LOGIN = true | false ;
- LIST USERS
- CREATE KEYSPACE keyspace
- DROP KEYSPACE keyspace
- CREATE TABLE table ...
- DROP TABLE ...
- ALTER TABLE ...

Quelques commandes

- INSERT INTO ...
- SELECT ... FROM ...
- UPDATE ...
- DELETE ...

Quelques commandes liées au shell (cqlsh)

- EXIT : quitte cqlsh
- SHOW : affiche les informations sur Cassandra
- SOURCE fichier : exécution d'un fichier de commandes CQL
- COPY ... : import/export de fichier CSV
- ...

Les opérations du CRUD : La création

```
INSERT INTO [keyspace.] table (liste_de_colonnes)  
VALUES (liste_de_valeurs) [IF NOT EXISTS]  
[USING TTL secondes | USING TIMESTAMP epoch_en_microsecondes]
```

- La liste de colonnes DOIT comporter la clé primaire
- IF NOT EXISTS permet de retourner une valeur (vrai / faux + ligne)
- USING TTL indique que la donnée sera obsolète (donc effaçable) au bout d'un certain temps (TTL = Time-to-live)
- USING TIMESTAMP indique la valeur de timestamp à utiliser pour l'insertion des valeurs de colonnes

ATTENTION : **IF NOT EXISTS** et **USING TIMESTAMP** ne sont pas compatibles

Les opérations du CRUD : La création

Remarques :

- Les colonnes non mentionnées ne sont pas créées
- Vous ne pouvez pas mettre de valeur de compteur (counter). Ces valeurs ne peuvent être valuées qu'avec UPDATE
- Les colonnes de clustering ne peuvent faire plus de 64Ko
- Les valeurs peuvent être
 - Des littéraux
 - Des collections
 - Set : { valeur, ... }
 - List : [valeur, ...]
 - Map { clé : valeur, ... }

Les opérations du CRUD : la mise à jour

```
UPDATE [keyspace.] table  
[USING TTL secondes | USING TIMESTAMP epoch_en_microsecondes]  
SET affectation [, affectation] ... WHERE conditions  
[IF EXISTS | IF condition [AND condition] ...];
```

- **USING TTL** et **USING TIMESTAMP** sont identiques à INSERT
- **IF EXISTS** et **IF condition** permettent de retourner une information (vrai/faux) pour indiquer ce qui est mis à jour
- Les affectations permettent d'affecter des littéraux, et des collections
- Les conditions fonctionnent quasiment comme les conditions SQL (AND, OR, NOT, =, <>, ... IN ...).
- La condition doit porter :
 - Sur la clé de partition si vous mettez à jour une colonne statique
 - Sur toute la clé primaire sinon.

Les opérations du CRUD : la suppression

```
DELETE [column_name (term)][, ...]  
FROM [keyspace.] table  
[USING TIMESTAMP valeur_timestamp]  
WHERE PK_column_conditions  
[IF EXISTS | IF static_column_conditions]
```

- Attention : la suppression n'est pas immédiate
- Par défaut, la suppression supprime les colonnes indiquées.
S'il n'y a pas de colonne indiquée, toute la ligne est supprimée
- Si **USING TIMESTAMP** est indiqué, les valeurs plus anciennes que la valeur indiquée seront supprimées
- La condition porte sur la clé primaire et ne peut comporter que des = et des IN

Les opérations du CRUD : la lecture d'informations

```
SELECT * | expression | DISTINCT partition  
FROM [keyspace.] table  
[WHERE partition_value [AND clustering_filters [AND static_filters]]]  
[ORDER BY PK_column_name ASC|DESC]  
[LIMIT N]  
[ALLOW FILTERING]
```

Les opérations du CRUD : la lecture d'informations

- Uniquement sur **une seule famille de colonne**. Pas de jointure
- Tri uniquement sur une colonne de la clé primaire.
- Conditions uniquement
 - sur les colonnes de la partition,
 - Sur les colonnes statiques, mais pas garanti, risque d'erreur. Dans ce cas, utilisez ALLOW FILTERING.
 - Eventuellement, et en complément, sur les colonnes de la clé de clustering.

La sélection de colonnes du SELECT

- * = toutes les colonnes
- Une liste de colonnes identifiées
- DISTINCT partition
- Des fonctions d'agrégation sur les partitions
COUNT(...), MIN(...), MAX(...), SUM(...)
- WRITETIME(colonne) = date d'écriture de la colonne
Attention : ne fonctionne pas sur les LIST, SET et MAP

Les opérations du CRUD : Importer / Exporter

Pour faciliter les échanges, on peut aussi importer / exporter des données dans / depuis un fichier. Ceci ne fonctionne que depuis CQLSH.
Le format de fichier est habituellement CSV.

- Exporter une table vers un fichier

```
COPY mytable TO 'monFichier.csv';
```

- Importer un fichier CSV dans une table

```
COPY mytable FROM 'monFichier.csv';
```

Attention, certaines limitations de Python peuvent entrainer des problèmes si l'une des données fait plus de 128Ko.

Les opérations du CRUD : Importer / Exporter

Vous pouvez également ajouter des options

```
COPY mytable FROM 'monFichier.csv' WITH DELIMITER=';' AND HEADER=TRUE
```

La création de l'infrastructure

Créer un Keyspace :

```
CREATE KEYSPACE nomKeyspace [ IF NOT EXISTS ]  
    WITH REPLICATION = {  
        'class' : strategy,  
        'replication_factor' : replic_fact };
```

- strategy = stratégie d'allocation. Utilisez 'SimpleStrategy'
- replic_fact = facteur de réplication (nombre de copie d'une ligne)

Utiliser un Keyspace

Utiliser un keyspace :

```
USE nomKeyspace ;
```

Lister les keyspaces :

```
DESCRIBE keyspaces ;
```

Modification de l'infrastructure

Modifier un keyspace :

```
ALTER KEYSPACE nomKeyspace  
    WITH strategy_class=...  
        AND strategy_options:replication_factor=... ;
```

Supprimer un keyspace :

```
DROP KEYSPACE nomKeyspace ;
```

Créer une famille de colonne

```
CREATE TABLE maFamille (  
    unecolonne sonType,  
    uneAutreColonne leType,  
    ...  
    PRIMARY KEY ( colonne_clé, clés_de_clustering )  
);
```

- La **première colonne** de PRIMARY KEY est la **clé de partitionnement**.
- Les autres colonnes de PRIMARY KEY sont les **clés de clustering**.
- Les colonnes statiques sont définies en ajoutant le mot clé **static** lors de la définition de la colonne

Modifier une famille de colonnes

```
ALTER TABLE [keyspace_name.] table_name  
    [ALTER column_name TYPE cql_type]  
    [ADD (column_definition_list)]  
    [DROP column_list | COMPACT STORAGE ]  
    [RENAME column_name TO column_name]  
    [WITH table_properties];
```


Les types de données

- Types classiques
text, varchar, date, time, timestamp, decimal, ...
- types ensemblistes
set <... >, list <... >, map <..., ... >
- "identifiant" / universally unique id
uuid, timeuuid

La création d'un uuid passe généralement par l'utilisation de la fonction uuid().

Plan

- 1 Généralités
- 2 Le modèle de données
- 3 Introduction à CQL
- 4 Concevoir un Keyspace**
- 5 Connexion à Cassandra

Remarques préalables

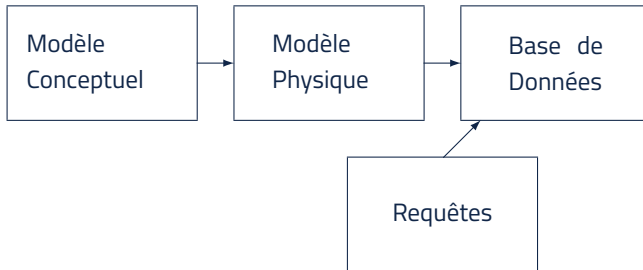
- Sur le modèle
 - Les lignes ne comportent pas forcément les mêmes colonnes
 - Il n'y a pas de notion de clé étrangère
- CQL
 - Pas de jointures
 - Notion de tri limitée

=> façon différente de concevoir le modèle

Concevoir un modèle Relationnel

- Conception d'un MCP
- Construction des relations -> MPD, table
- Normalisation du modèle (supprimer les doublons)
- Les requêtes s'adaptent en effectuant des jointures

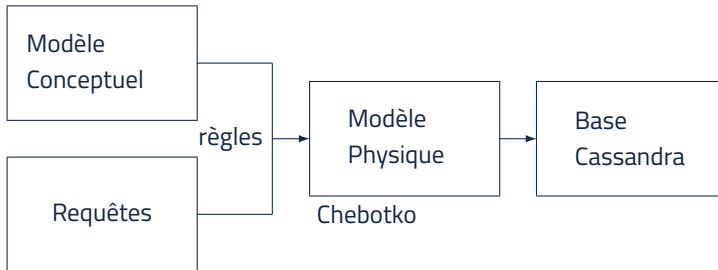
Concevoir un modèle Relationnel



Concevoir un modèle noSQL / Cassandra

- Pas de jointure
 - => une famille de colonnes doit contenir toutes les informations
 - => La structure est construite en fonction des requêtes à faire
- Une lecture, ça coute cher
 - => dupliquer les données
 - => Dénormalisation du modèle (introduire des doublons)
- Distribution de données
 - => Bien choisir la clé de partitionnement

Concevoir un modèle noSQL / Cassandra



Pourquoi introduire les requêtes ?

- Pas de notion de jointure
- La structure d'une famille de colonnes doit permettre de répondre à une requête
-> elle doit comporter toutes les colonnes nécessaires pour répondre aux requêtes.
- Pas de notion d'ordre sur les lignes
- L'ordre de placement des informations doit être défini à la création de la famille de colonnes
- Les clés de partitionnement placeront les lignes sur les nœuds
Le reste de la clé ordonne les informations dans la partition

Diagramme de CHEBOTKO

Le Diagramme de Chebotko consiste à intégrer les requêtes prévues dans le Modèle Conceptuel des Données, de manière à produire un schéma adapté à une base NoSQL.

Notation

On conserve la notation “en tableau” des modèles et on ajoute une information sur le rôle de chaque colonne.

Famille		
Colonne_1	type_1	role_1
Colonne_2	type_2	role_2
...

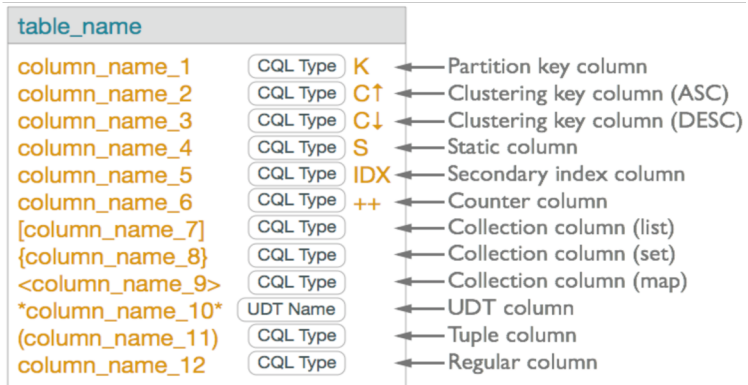
Rôles des colonnes du diagramme de Chebotko

- K = Primary Key (clé primaire)
- S = colonne statique (aura une valeur fixe dans la partition)
- C = colonne de clustering (clé de clustering)
- $C\uparrow$ = ordre ascendant
- $C\downarrow$ = ordre descendant
- IDX = colonne d'index secondaire
- ++ = compteur

Les noms des colonnes du diagramme de Chebotko

- colonne = une colonne (clé, valeur, timestamp)
- [colonne] = liste de colonnes (ordonné)
- { colonne } = ensemble de colonnes
- < colonne > = carte de colonnes (= map : indexée)
- * colonne * = UDT
- (colonne) = tuple

Diagramme de Chebotko



Construire le diagramme de Chebotko

Considérons un Modèle Conceptuel des Données et une requête.

- La première étape consiste à intégrer dans un seul tableau tous les éléments nécessaires à la requête.
- Le Modèle Conceptuel et les éléments intégrés doivent permettre d'en déduire un identifiant.
- Au niveau de l'identifiant, certaines des colonnes font partie des éléments de recherche. Ils correspondent alors à la clé de partitionnement
- Les colonnes de l'identifiant qui ne sont pas dans la clé de partitionnement correspondent à la clé de clustering
- Les colonnes liées directement à la clé de partitionnement mais pas à la clé de clustering sont des colonnes statiques

Exemple de construction 1/11

Sujet :

Un client passe des commandes qui comportent un certain nombre de lignes

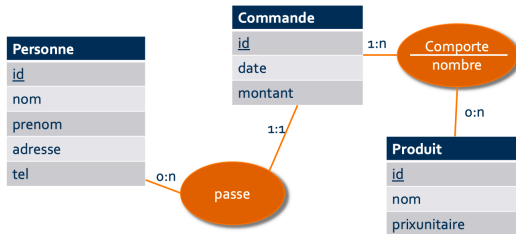
Construction du MCD

Les éléments manipulés :

- Les clients / personnes
- Les commandes qu'ils ont passé
- Le détail de la commande (liste des produits)

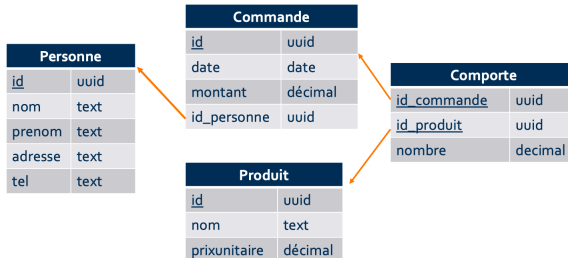
Exemple de construction 2/11

Le MCD



Exemple de construction 3/11

Le MPD



Exemple de construction 4/11

Les requêtes :

- Q1 : La liste des factures d'un client (non détaillée)
- Q2 : le détail d'une facture d'un client.

Si on analyse nos requêtes, il est clair que Q2 n'est réalisé que lorsqu'on connaît les factures du client, donc le résultat de Q1.

Exemple de construction 5/11

Considérons Q1.

En relationnel, ce serait la jointure entre personne et commande.

Les lignes de résultat sont structurées comme suit :

Q1	
id_personne	uuid
nom	text
prenom	text
adresse	text
tel	text
id_commande	uuid
date	date
montant	decimal

Exemple de construction 6/11

Analysons notre ligne de résultat

Son identifiant est (id_personne, id_commande)

Du point de vue de la requête, on cherche les commandes d'une personne.
Donc pour aller vite, il faut regrouper les données d'une personne.

- id_personne sera notre clé de partitionnement
- id_commande sera notre clé de clustering
- nom, prenom, adresse, tel sont lié à id_personne mais pas à id_commande, donc ce sont des colonnes statiques.

Exemple de construction 7/11

Le diagramme de Chebotko correspondant est donc :

Q1	Commande_par_personne		
→	id_personne	uuid	K
	nom	text	S
	prenom	text	S
	adresse	text	S
	tel	text	S
	id_commande	uuid	C↑
	date	date	
	montant	decimal	

Exemple de construction 8/11

Il nous reste à prendre en compte Q2.

Pour effectuer Q2, on se base sur le résultat de Q1.

Q2 est une jointure entre personne, commande, comporte et produit

Q2	
id_personne	uuid
nom	text
prenom	text
adresse	text
tel	text
id_commande	uuid
date	date
montant	decimal
id_produit	uuid
nom_produit	text
nombre	int
prix_unitaire	decimal

Exemple de construction 9/11

On en déduit Chebotko

Q2
→

Personne_commande			
id_personne	uuid	K	
nom	text	S	
prenom	text	S	
adresse	text	S	
tel	text	S	
id_commande	uuid	K	
date	date	S	
montant	decimal	S	
id_produit	uuid	C↑	
nom_produit	text		
nombre	int		
prix_unitaire	decimal		

Exemple de construction 10/11

En fait, on peut simplifier le schéma puisque le point de départ de la requête est id_commande. On n'a plus besoin de id_personne

Q2
→

Personne_commande		
nom	text	S
prenom	text	S
adresse	text	S
tel	text	S
id_commande	uuid	K
date	date	S
montant	decimal	S
id_produit	uuid	C↑
nom_produit	text	
nombre	int	
prix_unitaire	decimal	

Diagramme global 11/11

Q1	Commande_par_personne			Q2	Personne_commande		
→	id_personne	uuid	K	→	nom	text	S
	nom	text	S		prenom	text	S
	prenom	text	S		adresse	text	S
	adresse	text	S		tel	text	S
	tel	text	S		id_commande	uuid	K
	id_commande	uuid	C↑		date	date	S
	date	date			montant	decimal	S
	montant	decimal			id_produit	uuid	C↑
					nom_produit	text	
					nombre	int	
					prix_unitaire	decimal	

Mise en oeuvre de l'exemple

Nous devons :

- Créer un keyspace
- Créer les 2 des familles de colonnes
 - Préciser la clé, et indiquer en premier la clé de partitionnement, le reste de la clé est la clé de clustering
 - Ne pas oublier pas les colonnes statiques

Création du Keyspace

Il faut juste choisir la stratégie de réplication et la facteur de réplication.

```
CREATE KEYSPACE factures IF NOT EXISTS  
  WITH REPLICATION = {  
    'class' : 'SimpleStrategy',  
    'replication_factor' : 1 };
```

Création des familles de colonnes 1/3

- Les familles de colonnes sont indépendantes, on les définit les unes à la suite des autres.
- Les colonnes sont créées en fonction de ce qu'indique Chebotko.
 - L'ordre de définition n'a pas d'importance
 - Les colonnes statiques sont suivies du mot clé static
 - Lors de la définition de la clé primaire, la première colonne est la clé de partitionnement.

Création des familles de colonnes 2/3

```
CREATE TABLE commandepersonne (  
    id_personne uuid,  
    nom text static,  
    prenom text static,  
    adresse text static,  
    tel text static,  
    id_commande uuid,  
    date date,  
    montant decimal,  
    PRIMARY KEY ( id_personne, id_commande )  
);
```

Création des familles de colonnes 3/3

```
CREATE TABLE personnecommande (  
    id_commande uuid,  
    nom text static,  
    prenom text static,  
    adresse text static,  
    tel text static,  
    date date static,  
    montant decimal static,  
    id_produit uuid,  
    nom_produit text,  
    nombre decimal,  
    prix_unitaire decimal,  
    PRIMARY KEY ( id_commande, id_produit )  
);
```

Autres exemples de création de familles de colonnes

```
CREATE TABLE grades (  
    student_id uuid,  
    student_name text static,  
    grade_id uuid,  
    obtained_at timestamp,  
    grade text,  
    subject text,  
    PRIMARY KEY (student_id, obtained_at, grade_id)  
);
```

Autres exemples de création de familles de colonnes

```
CREATE TABLE courses (  
    email text PRIMARY KEY,  
    etudiant_nom text static,  
    etudiant_prenom text static,  
    cours set<text>  
);
```

Exemples d'insertions / modifications de données

```
INSERT INTO grades (student_id, student_name, grade_id, obtained_at, grade, subject)
VALUES (uuid(), 'Marc Down', uuid(), 'B+', 'chinois');
```

```
INSERT INTO courses(email, etudiant_nom, etudiant_prenom)
VALUES (uuid(), 'bidule.machin@ec-nantes.fr','bidule','machin') IF NOT EXISTS;
```

```
UPDATE courses SET cours=cours + ['japonais'] WHERE email='bidule.machin@ec-nantes.fr';
```

```
UPDATE courses SET cours = cours - ['chinois'] WHERE email='bidule.machin@ec-nantes.fr';
```


Plan

- 1 Généralités
- 2 Le modèle de données
- 3 Introduction à CQL
- 4 Concevoir un Keyspace
- 5 Connexion à Cassandra

Installer Cassandra



- Plusieurs outils
 - Cassandra
 - CQL
- Pas tous développés dans le même environnement
=> a besoin de certains outils
- Java : JDK 8 (évitez la version 9, pas sûr de la compatibilité)
- Python :
 - 2.7 pour cqlsh
 - 3 pour le développement

Installer Cassandra - Environnement

- JDK 8
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>
Prenez la version JDK, mais pas au delà de la version 8
Vérifiez que JAVA_HOME est configuré correctement
- Python 2.7
<https://www.python.org/downloads/>
Prenez la version adaptée à votre système
Vérifiez que l'accès à python est dans le PATH
- Python 3 - si vous développez en python

Installer Cassandra

Sur le site d'Apache Cassandra

- Au choix
 - All : Décompresser le fichier (tar.gz)
 - Linux : apt-get
 - Linux : yum install
 - Clone depuis le dépôt git
- **Ajouter au PATH le dossier bin de cassandra**

Installer Cassandra

bin : scripts

conf : fichiers de configuration

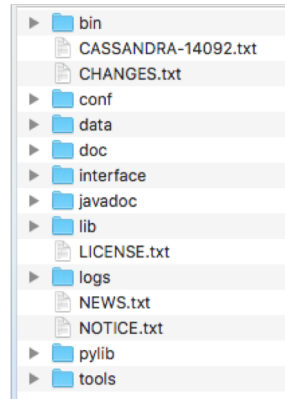
data : données

interface : connecteurs vers des
langages de programmation

lib : bibliothèques de fonctions

pylib : bibliothèques python

tools : utilitaires



Installer Cassandra

Dans bin/, quels outils ?

- `cassandra` : le serveur
- `cqlsh` : le client
- `debug-cql` : debugger cql
- `nodetool` : gestionnaire de cluster
- `sstable...` : gestionnaire de masse

Lancer le serveur 1/3

- Ouvrir le terminal / invite de commande
- Aller dans le répertoire bin de cassandra
- Lancer cassandra
 - Linux/macOS :
 - En foreground (terminal bloqué) : `./cassandra`
 - En background (Libère de terminal) : `./cassandra -f`
 - Windows : `cassandra`
sous entendu... `cassandra.bat`, lancé en Foreground

Quelle différence entre Foreground / Background ?

- Foreground : arrêt via CTRL C
- Background... : il faut utiliser la commande kill avec le PID de cassandra

Lancer le serveur 2/3

```
bin — java -Xloggc:./logs/gc.log -ea -XX:+UseThreadPriori
bin kwhrf ./cassandra -f
objc[17233]: Class JavaLaunchHelper is implemented in both /Library/Java/
1.jdk/Contents/Home/jre/lib/libinstrument.dylib (0x1046ef4e0). One of the
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.des
b/Columns;
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.ser
CompilerOracle: dontinline org/apache/cassandra/db/Columns$Serializer.ser
CompilerOracle: dontinline org/apache/cassandra/db/commitlog/AbstractComm
CompilerOracle: dontinline org/apache/cassandra/db/transform/BaseIterator
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTran
CompilerOracle: dontinline org/apache/cassandra/db/transform/StoppingTran
CompilerOracle: dontinline org/apache/cassandra/ia/util/BufferedDataOutput
CompilerOracle: dontinline org/apache/cassandra/ia/util/BufferedDataOutput
CompilerOracle: dontinline org/apache/cassandra/ia/util/BufferedDataOutput
CompilerOracle: dontinline org/apache/cassandra/ia/util/RebufferingInputS
CompilerOracle: inline org/apache/cassandra/db/rows/UnfilteredSerializer.
util/DataOutputPlus;J
CompilerOracle: inline org/apache/cassandra/ia/util/Memory.checkBounds (C
CompilerOracle: inline org/apache/cassandra/ia/util/SafeMemory.checkBound
CompilerOracle: inline org/apache/cassandra/utills/AsymmetricOrdering.sele
```

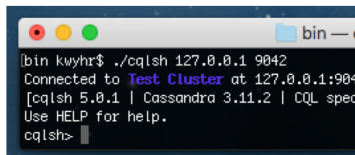

Lancer le serveur 3/3

```
INFO [main] 2018-05-29 08:40:50,260 NativeTransportService.java:75 - Netty using Java NIO event loop
INFO [main] 2018-05-29 08:40:50,367 Server.java:155 - Using Netty Version: [netty-buffer=netty-buffer-4.0.44.Final.452812a, netty-codec-haproxy-4.0.44.Final.452812a, netty-codec-http=netty-codec-http-4.0.44.Final.452812a, netty-codec-socks=netty-codec-socks-4.0.44.Final.452812a, netty-handler=netty-handler-4.0.44.Final.452812a, netty-tcnative=netty-tcnative-1.1.33.Fork26.142ecbb, netty-transport=netty-transport-native-epoll-4.0.44.Final.452812a, netty-transport-rxtx=netty-transport-rxtx-4.0.44.Final.452812a, netty-transport-sctp=netty-transport-udt-4.0.44.Final.452812a]
INFO [main] 2018-05-29 08:40:50,368 Server.java:156 - Starting listening for CQL clients on localhost/127.0.0.1:9042 (unencrypted)...
INFO [main] 2018-05-29 08:40:50,450 CassandraDaemon.java:529 - Not starting RPC server as requested. Use JMX (StorageService->startRPC)
INFO [IndexSummaryManager:1] 2018-05-29 09:40:49,827 IndexSummaryRedistribution.java:76 - Redistributing index summaries
```

Notez l'adresse et le port d'installation de cassandra (fin de la commande du slide précédent). Normalement 9042 ou 9160

Utiliser CQLSH 1/2

Lancez `cqlsh` dans une nouvelle fenêtre de terminal si vous l'avez lancé en **Foreground**

A screenshot of a terminal window on a macOS system. The window title is "bin — c". The prompt is "[bin kwyhr\$]". The user has entered the command "./cqlsh 127.0.0.1 9042". The output shows "Connected to Test Cluster at 127.0.0.1:9042", followed by "[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec" and "Use HELP for help.". The prompt is now "cqlsh>".

```
[bin kwyhr$] ./cqlsh 127.0.0.1 9042
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec
Use HELP for help.
cqlsh>
```

Nb : suivant votre configuration, la commande `cqlsh` suffit

Utiliser CQLSH 2/2

Exemple de commandes

```
DESCRIBE keyspaces;
```

```
CREATE KEYSPACE students WITH REPLICATION = {'class': 'SimpleStrategy', 'replication_factor': 1};  
DESCRIBE keyspaces;
```

```
USE students;
```

```
CREATE TABLE student ( student_id uuid, name text static, cours set<text> static, grade_id uuid,  
when timestamp, value text, eval text, primary key(student_id, grade_id, when));
```

```
INSERT INTO student(student_id, name, cours, grade_id, when, value, eval)  
VALUES (uuid(), 'Jacques WEBER', 'chinois', uuid(), toTimestamp(now()), 'A+', 'chinois');
```

```
SELECT * FROM student;
```

```
exit;
```

Un peu de sécurité ...

Par défaut, n'importe quel utilisateur peut se connecter sur un serveur Cassandra. Pour éviter cela, il faut ajouter des comptes utilisateur et configurer le serveur pour qu'il n'autorise les connexions que pour les utilisateurs identifiés.

<http://cassandra.apache.org/doc/latest/operating/security.html>

Un peu de sécurité ...

Première étape : le fichier de configuration : `cassandra.yaml`

Rubrique : `authenticator`

Remplacer `AllowAllAuthenticator` par `PasswordAuthenticator`

Et si vous le pouvez, changez également le `replication_factor` de `system_auth`.

Un peu de sécurité ...

Redémarrez Cassandra et connectez vous avec le login/mot de passe du superuser

```
cqlsh -u cassandra -p cassandra
```

Créez ensuite vos utilisateurs / mot de passe et indiquez s'ils peuvent se connecter, leur rôle, ... Et déconnectez le login cassandra

```
CREATE ROLE nomRole WITH PASSWORD='motDePasse' AND LOGIN=true AND  
SUPERUSER=true ;  
ALTER ROLE cassandra WITH SUPERUSER = false AND LOGIN = false ;
```

Un peu de sécurité ...

Maintenant vous ne pourrez vous connecter qu'avec un Login et Mot de passe prédéfinis.

Attention :

Quand vous vous connectez, les login sont tous convertis en minuscule.

Un peu de sécurité ...

Pour aller un peu plus loin, on peut définir les droits sur les keyspace.
Il est recommandé de faire la manip sur un noeud qui ne reçoit momentanément aucune requête.

Reprenez le fichier de configuration : `cassandra.yaml`

Rubrique : `authorizer`

Remplacer `AllowAllAuthorizer` par `CassandraAuthorizer`

Redémarrez Cassandra

Un peu de sécurité ...

Connectez vous ensuite avec un compte superuser

Et indiquez les droits pour un utilisateur (ici le droit SELECT pour db_user pour le keyspace k1)

```
GRANT SELECT ON KEYSpace k1 TO db_user ;
```

<http://cassandra.apache.org/doc/latest/cql/security.html#grant-permission-statement>

Connexion en JAVA

Pour se connecter à une base Cassandra, nous utilisons 2 notions :

- Cluster : le cluster Cassandra
- Session : une session de connexion à ce cluster.

Connexion en JAVA

Nous utilisons l'API de Datastax

Datastax : émanation de développeurs de Facebook qui développent pour Cassandra.

- Javadoc : <https://docs.datastax.com/en/drivers/java/3.5/>
- Jar : <http://downloads.datastax.com/java-driver/cassandra-java-driver-3.5.1.tar.gz>
- Maven :
 - groupId : org.apache.cassandra
 - artifactId : cassandra-all
 - version : 3.11

La Session nous servira à faire toutes nos requêtes.

Connexion en JAVA

Concrètement, il faut ...

- Créer un projet d'application Maven sous Netbeans.
- Au niveau des dépendances, ajouter celles indiquées au slide suivant
- Mettre en oeuvre le programme

Connexion en JAVA

Dépendances MAVEN

```
<dependency>
  <groupId>com.datastax.cassandra</groupId>
  <artifactId>cassandra-driver-core</artifactId>
  <version>3.1.4</version>
</dependency>
<dependency>
  <groupId>com.datastax.cassandra</groupId>
  <artifactId>cassandra-driver-mapping</artifactId>
  <version>3.1.4</version>
</dependency>
<dependency>
  <groupId>com.datastax.cassandra</groupId>
  <artifactId>cassandra-driver-extras</artifactId>
  <version>3.1.4</version>
</dependency>
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j-impl</artifactId>
  <version>2.11.1</version>
</dependency>
```

Connexion en JAVA

Le programme...

Importer la librairie de datastax

```
import com.datastax.driver.core.* ;
```

Définir le cluster et la session dans la classe

```
private Cluster cluster ;  
private Session session ;
```

Connexion en JAVA

Création du cluster

Par défaut :

```
public void connectCluster(String node) {  
    if (cluster == null) {  
        cluster = Cluster.builder().addContactPoint(node).build();  
    }  
    session = null ;  
}
```

Attention, si le cluster requiert une identification, il faudra l'indiquer

```
cluster = Cluster.builder().withCredentials(login, password).addContactPoint(node).build();
```

Connexion en JAVA

Création de la session

```
public void getSession() {  
    if ((cluster != null) && (session == null)) {  
        session = cluster.connect();  
    }  
}
```


Connexion en JAVA

Sans oublier de libérer le cluster et la session quand ils ne sont plus utilisés

```
public void closeSession() {  
    if (session != null) {  
        session.close();  
        session = null;  
    }  
}
```

```
public void closeCluster() {  
    closeSession();  
    if (cluster != null) {  
        cluster.close();  
        cluster = null;  
    }  
}
```

Connexion en JAVA

Il ne reste plus qu'à utiliser le tout.

Notre serveur est sur 127.0.0.1

Le vôtre est sans doute ailleurs. Pensez également au login / password s'il y en a un.

```
...
public class TestCassandra {
...
    public static void main(String[] args) {
        TestCassandra client = new TestCassandra();
        client.connectCluster("127.0.0.1");
        client.getSession();
        ...
        client.closeSession();
        client.closeCluster();
    }
}
```

Connexion en JAVA

Pour effectuer des requêtes ne nécessitant pas de résultat (CREATE, UPDATE, ...) on utilise la méthode `execute` de `Session`.

```
public void loadData() {  
    session.execute(...);  
}
```

Connexion en JAVA

Pour des requêtes en sélection (SELECT) on passe par des requêtes préparées.

4 étapes :

- Déclaration de la requêtes
- Mise en œuvre des paramètres
- Execution de la requêtes
- Exploitation des résultats

Connexion en JAVA

```
public void queryData() {  
    session.execute("USE students");  
    // Etape 1  
    PreparedStatement statement = session.prepare(  
        "SELECT * from school.grades WHERE student_id = ?;");  
    // Etape 2  
    BoundStatement boundStatement = new BoundStatement(statement);  
    boundStatement.bind(1);  
    // Etape 3  
    ResultSet rs = session.execute(boundStatement);  
    // Etape 4  
    for (Row row : rs) {  
        System.out.println(row.getString("subject") + "-" + row.getString("grade"));  
    }  
}
```

Connexion en Python

Pour vous connecter à Cassandra en Python, vous aurez besoin du driver :
<https://datastax.github.io/python-driver/installation.html>

Remarques :

- Fonctionne avec toutes les versions de python 2.7 et 3... en théorie.
En pratique, utilisez python3
- CQL3

Connexion en Python

Installation du driver sous python 3

```
pip3 install cassandra-driver
```

- MacOS :
Installez Homebrew, python3
<https://wsvincent.com/install-python3-mac/>
Puis installez le driver
- Windows : ... ça peut bien se passer

Connexion en Python

Quelques.... mésaventures

- Vérifiez que vous avez bien python3 installé
- "bad magic number in 'cassandra' : b'\x03\xf3\r\n'"
videz le cache de python : `find . -name *.pyc -delete`
- DeprecationWarning : Using or importing....
Bug du parser python3.7. Sera corrigé.
Ne devrait pas empêcher de fonctionner

Connexion en Python

Mettre en œuvre un. programme

- Importer Cluster
- Se connecter au cluster
- Acquérir une session
- Exécuter les requêtes et exploiter les résultats

Connexion en Python

```
from cassandra.cluster import Cluster

cluster = Cluster({'127.0.0.1'})
session = cluster.connect('students')

rows = session.execute('SELECT * FROM student')
for user_row in rows :
    print (user_row.name, user_row.eval, user_row.value)
```

```
python3 testcassandra.py
```

Connexion en Python

La même, mais avec login / mot de passe

```
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider

auth_provider = PlainTextAuthProvider(username='username', password='password')
cluster = Cluster(contact_points=['127.0.0.1'], auth_provider=auth_provider)
session = cluster.connect('students')

rows = session.execute('SELECT * FROM student')
for user_row in rows :
    print (user_row.name, user_row.eval, user_row.value)
```

