FORMATION CASSANDRA

XEBIA

Created by Matthieu Nantern / @mnantern

LE PROGRAMME !

1. Tour de table
2. Introduction
3. Architecture de Cassandra
4. Installation et configuration
5. Modèle de données
6. Le driver Java
7. Administration Cassandra

TOUR DE TABLE

- Qui suis-je ?
- Trois attentes sur la formation

INTRODUCTION

CASSANDRA ET LES BASES NOSQL

CASSANDRA ET LES BASES NOSQL

NoSQL ?

CASSANDRA ET LES BASES NOSQL

NoSQL ?

“NoSQL (Not only SQL en anglais) désigne une catégorie de systèmes de gestion de base de données qui n'est plus fondée sur l'architecture classique des bases relationnelles. Il renonce aux fonctionnalités classiques des SGBD relationnels au profit de la simplicité. Les performances restent bonnes en multipliant simplement le nombre de serveurs, solution raisonnable avec la baisse des coûts.”

LES BASES NOSQL

LES BASES NOSQL

...

LES BASES NOSQL

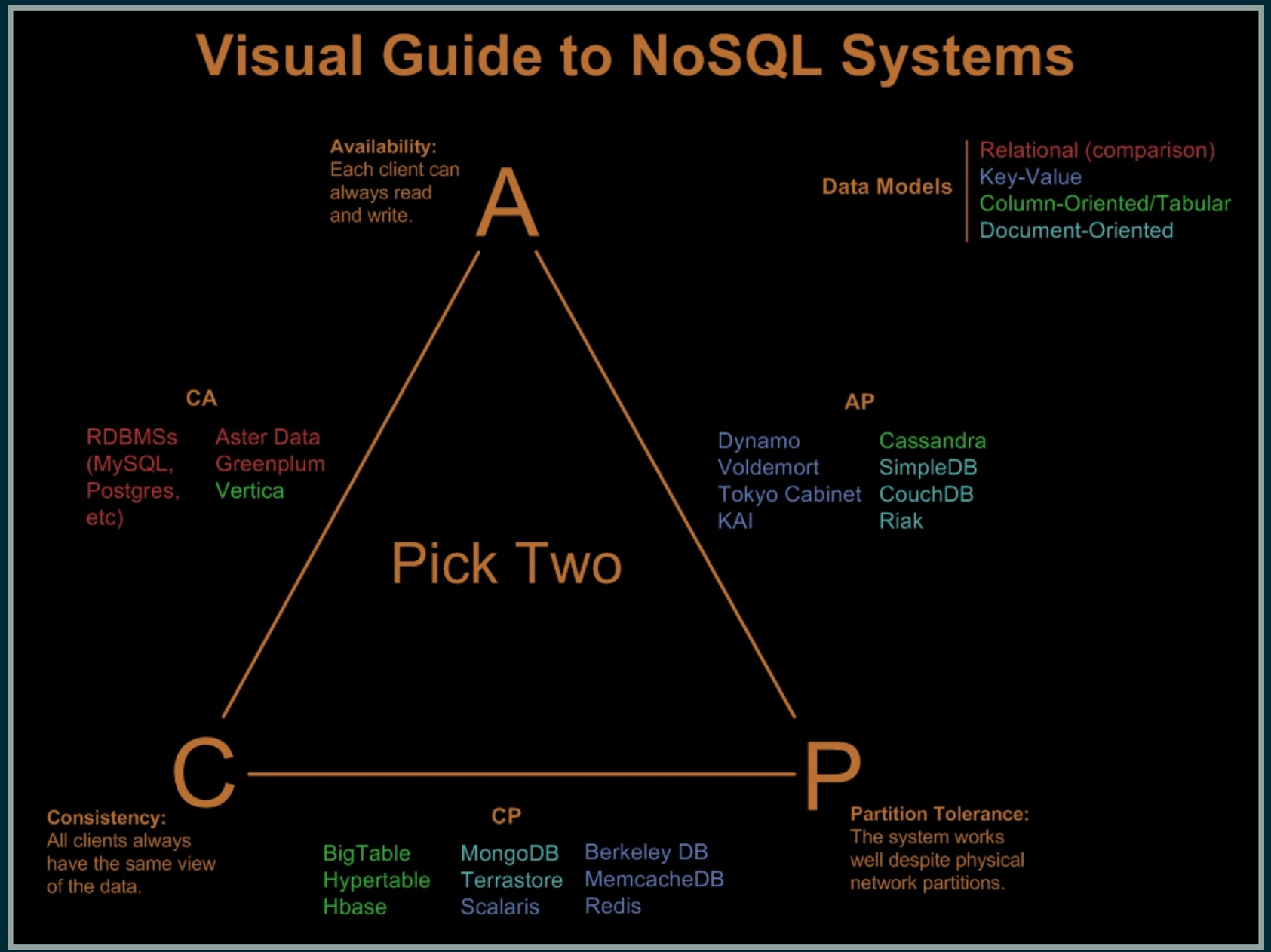
Les bases NoSQL se classent en quatre catégories:

- Colonnes: Cassandra, HBase
- Documents: MongoDB, ElasticSearch
- Clés/valeurs: Redis, Riak, Couchbase
- Graph: Neo4j

UN PEU DE THÉORIE: LE THÉORÈME DE CAP

Le théorème CAP également connu sous le nom de théorème de Brewer montre qu'il est impossible pour un système distribué de satisfaire de manière simultanée les trois contraintes suivantes :

- Cohérence ("Consistency") : tous les noeuds du système voient la même donnée au même moment;
- Disponibilité ("Availability") : chaque requête recevra une réponse (que cela soit un succès ou un échec);
- Résistance à la partition ("Partition tolerance") : le système continue de fonctionner malgré des défaillances pouvant aller jusqu'à la séparation du cluster en sous-système.


Visual Guide to NoSQL Systems
Availability:
Each client can always read and write.
A
Data Models
Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented
CA
RDBMSs (MySQL, Postgres, etc)
Aster Data
Greenplum
Vertica
AP
Dynamo
Voldemort
Tokyo Cabinet
KAI
Cassandra
SimpleDB
CouchDB
Riak
Pick Two
C
P
Consistency:
All clients always have the same view of the data.
CP
BigTable
Hypertable
Hbase
MongoDB
Terrastore
Scalaris
Berkeley DB
MemcacheDB
Redis
Partition Tolerance:
The system works well despite physical network partitions.

CASSANDRA

- **Technologies de base:**
  - Google BigTable : modèle de stockage
  - Amazon Dynamo : modèle de réplication
- Historique:
  - 2008 : libération par Facebook
  - 2010 : Top Level Project Apache
  - Octobre 2011 : version 1.0
  - Septembre 2013 : version 2.0

QUAND UTILISER CASSANDRA ?

- Pas de SPOF (Single point of failure)
- Réplication native entre serveurs et datacenters
- Scalabilité linéaire
- Beaucoup d'écriture et de lecture

QUAND NE PAS UTILISER CASSANDRA ?

- Besoin de transaction avec rollback
- ??

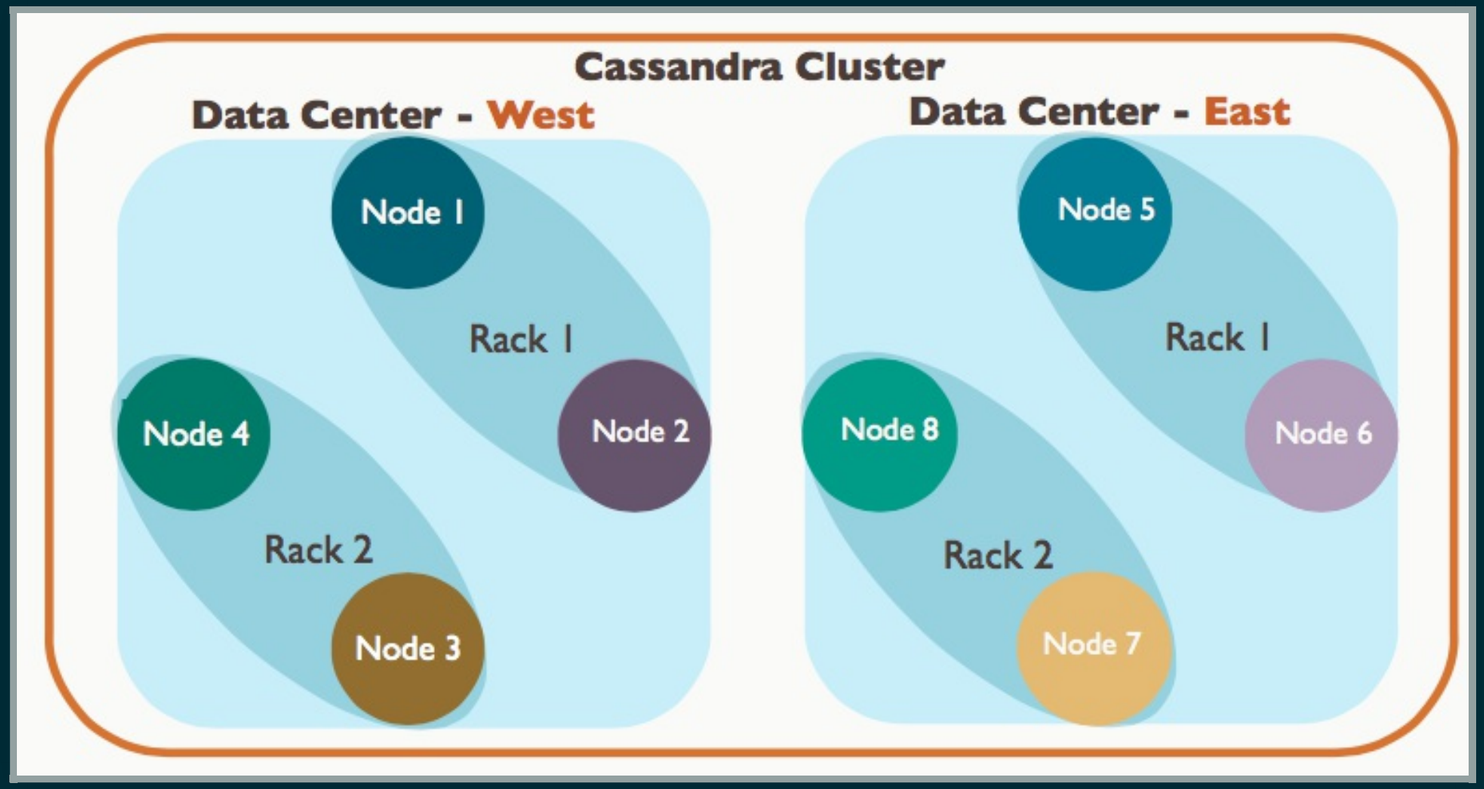
LES CAS D'UTILISATION CLASSIQUES DE CASSANDRA

- Time series / IOT
- Messaging
- Recommandation
- Catalogue produit / Playlists

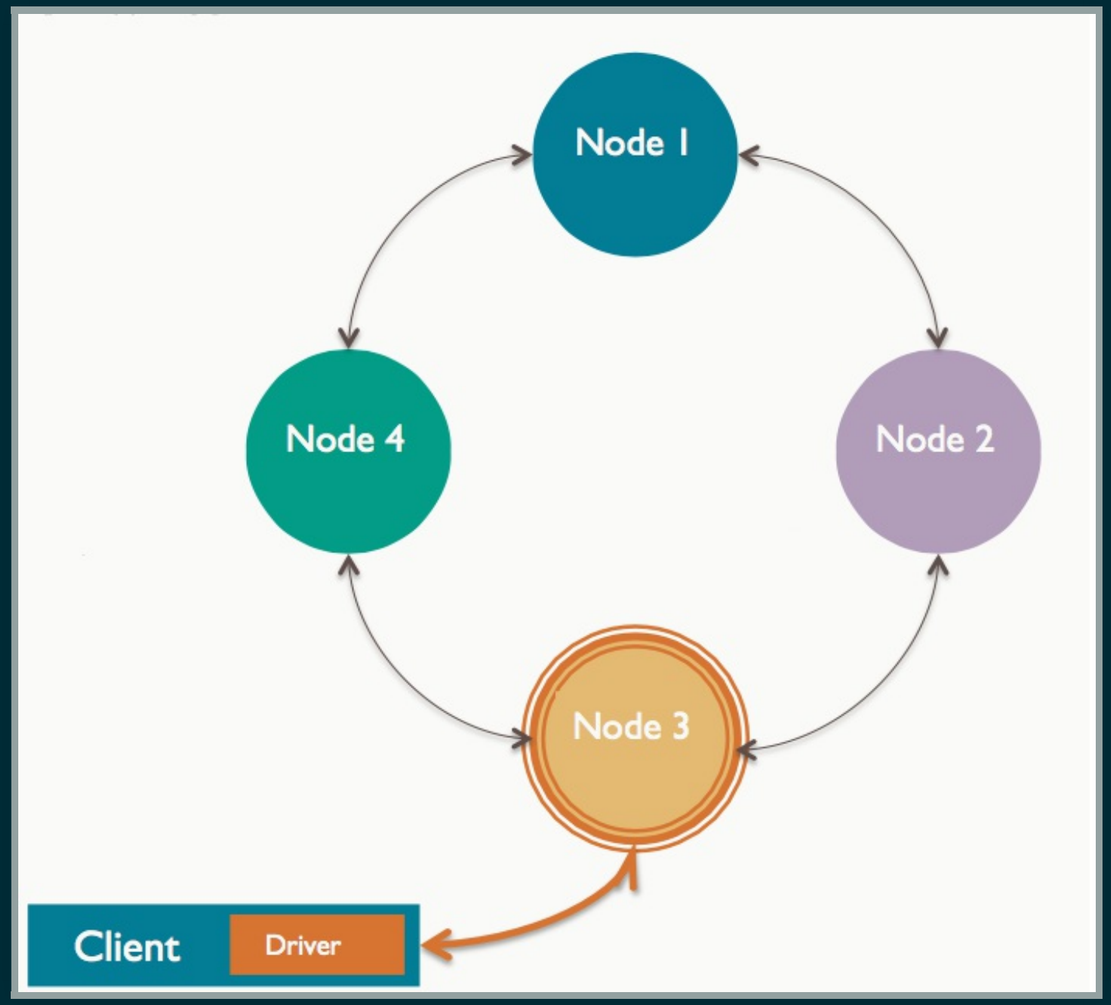
liste de cas d'utilisation C*

ARCHITECTURE DE CASSANDRA

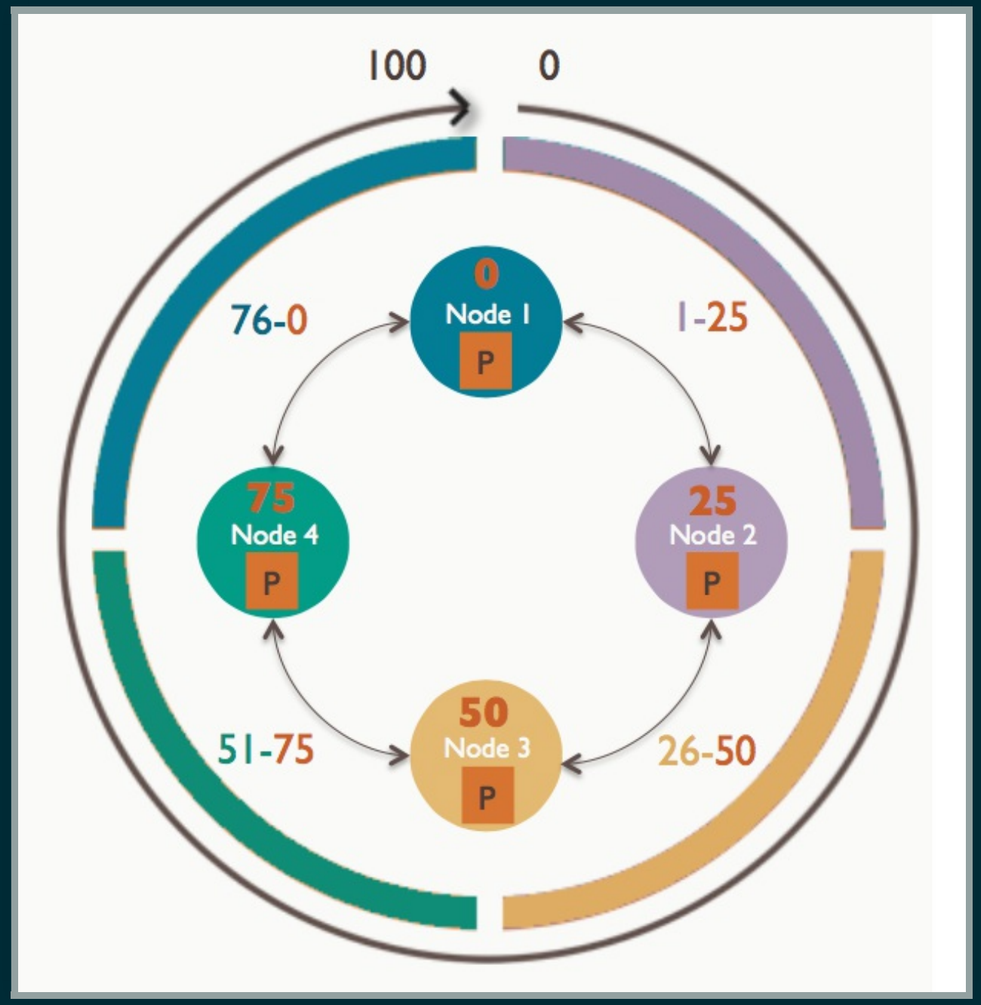
LE CLUSTER


Cassandra Cluster
Data Center - West
Node 1
Rack 1
Node 2
Node 4
Rack 2
Node 3
Data Center - East
Node 5
Rack 1
Node 6
Node 8
Rack 2
Node 7

LE COORDINATOR



LE PARTITIONNEMENT



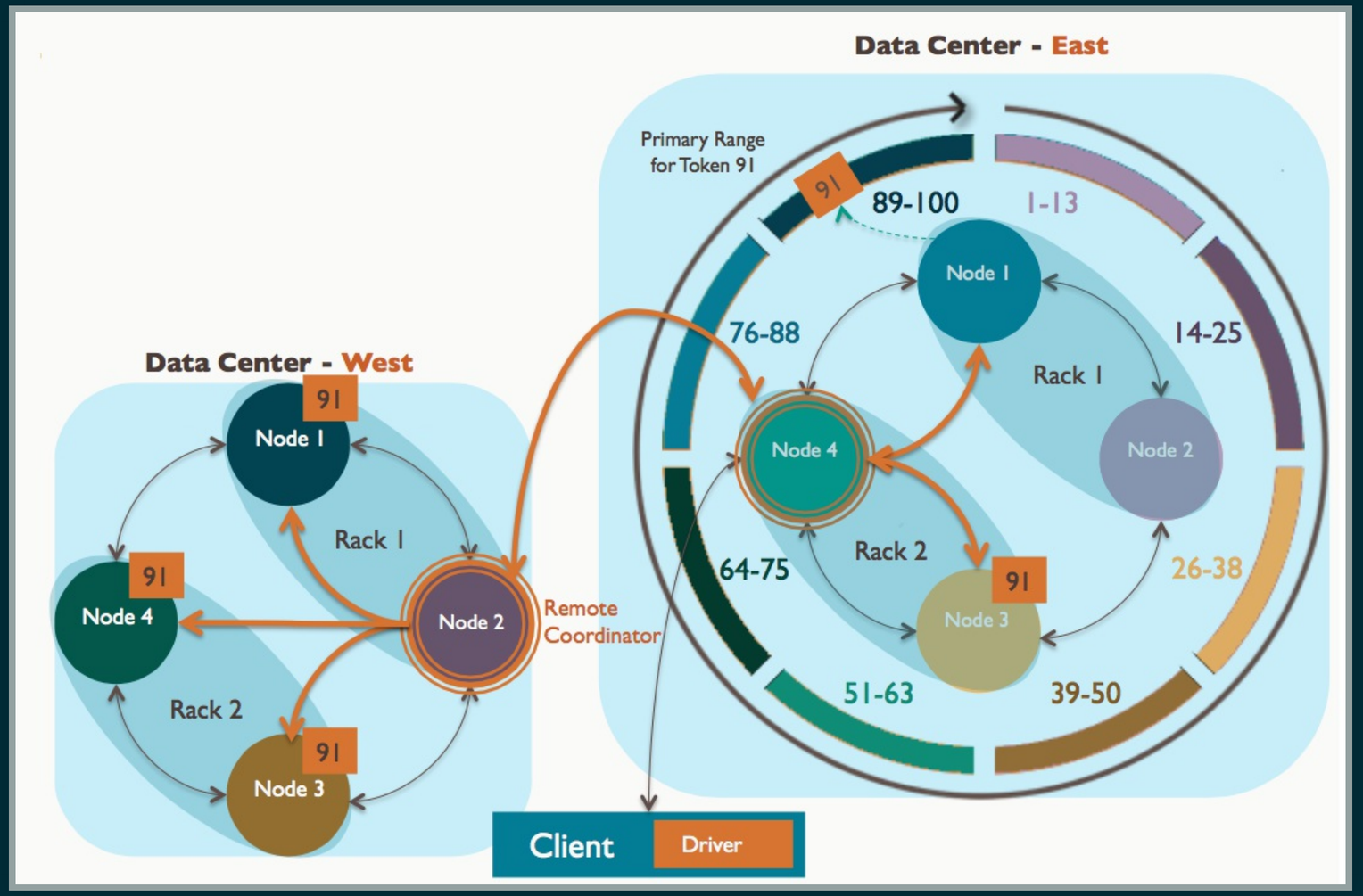
LA RÉPLICATION (1/2)

Deux éléments influent sur la réplication:

1. Le facteur de réplication: indique le nombre de fois qu'une donnée doit être répliquée sur le cluster
2. La stratégie de réplication: indique quelle machine doit stocker la donnée

```
CREATE KEYSPACE Demo
        WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc-ec
```

LA RÉPLICATION (2/2)



LA COHÉRENCE

La cohérence d'une requête définit le nombre de noeuds devant répondre à une requête avant que celle-ci ne soit considérée comme terminée par le cluster Cassandra.

Dans le cadre d'une lecture cela définit le nombre de noeud devant envoyer la copie la plus récente de la donnée.

Dans le cadre d'une écriture cela définit le nombre de noeud devant confirmer avoir écrit la donnée.

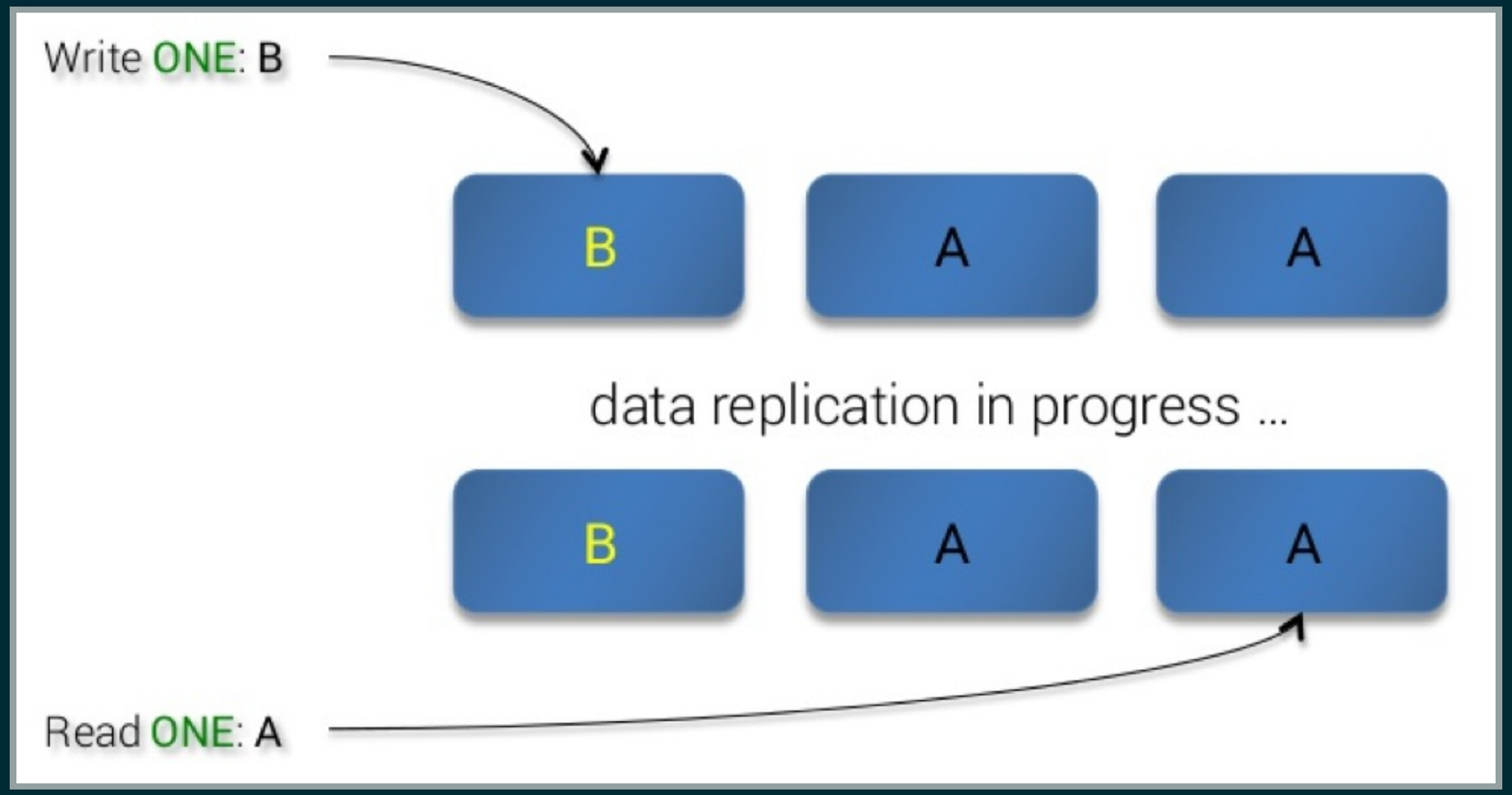
La cohérence est choisie par requête dans Cassandra.

LES DIFFÉRENTS NIVEAUX DE CONSISTENCE

Nom	Description
ONE	Attend la réponse d'une noeud avant de répondre au client.
QUORUM	Attend la réponse de RF/2+1 avant de répondre au client.
ALL	Attend la réponse de tous les noeuds avant de répondre au client.

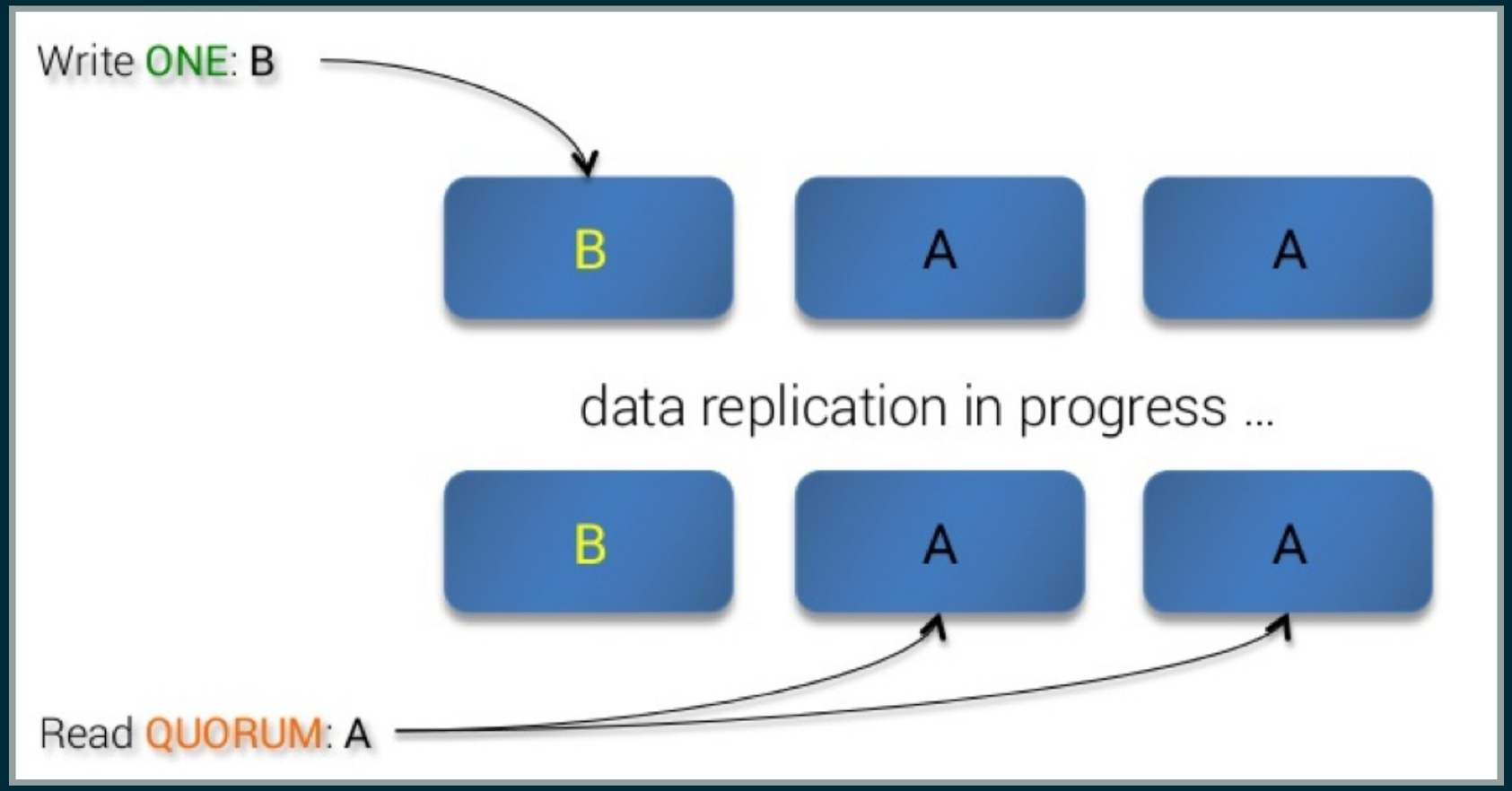
LA COHÉRENCE EN ACTION

RF = 3, Write ONE, Read ONE



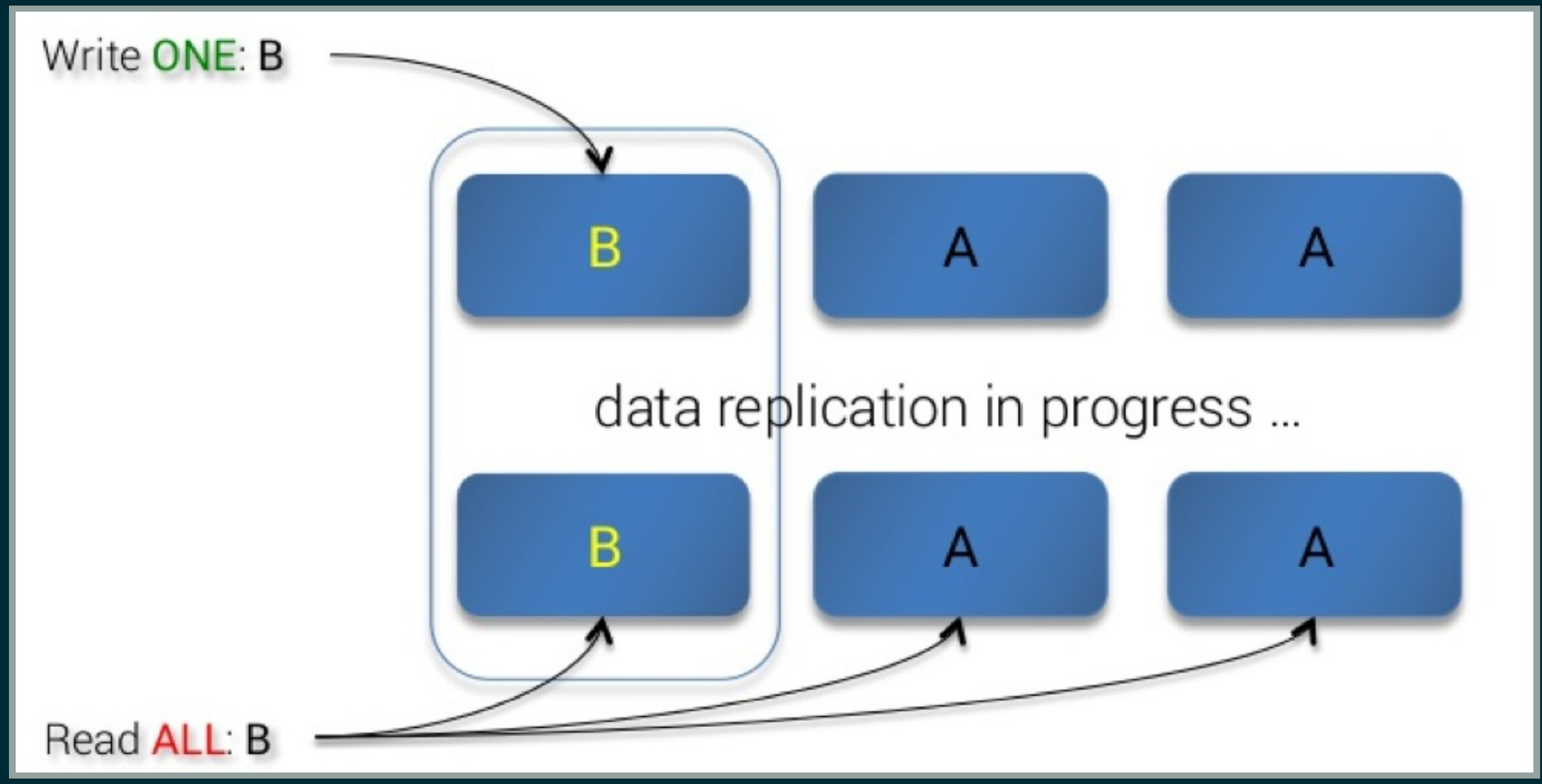
LA COHÉRENCE EN ACTION

RF = 3, Write ONE, Read QUORUM



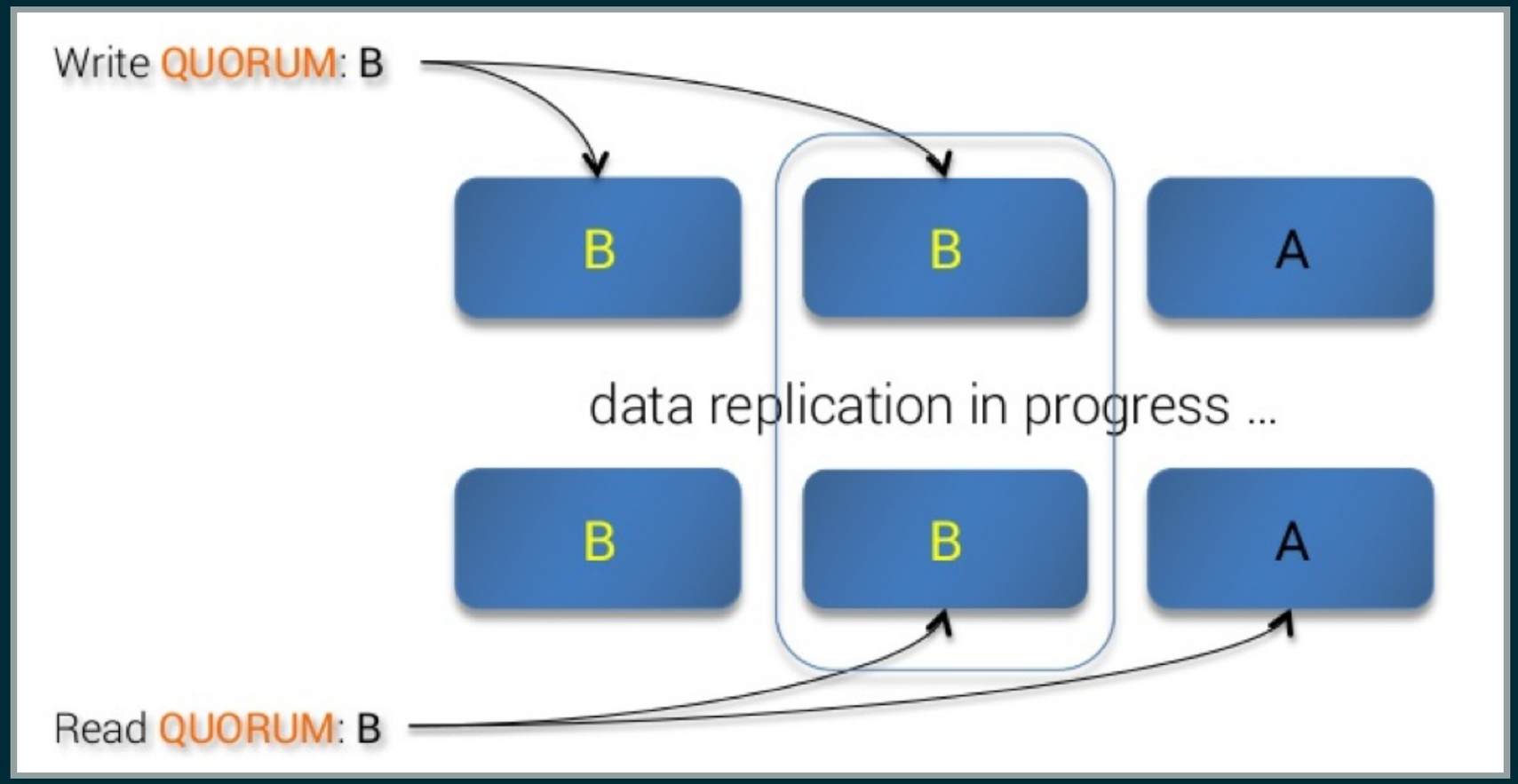
LA COHÉRENCE EN ACTION

RF = 3, Write ONE, Read ALL



LA COHÉRENCE EN ACTION

RF = 3, Write QUORUM, Read QUORUM



LES USAGES DES NIVEAUX DE CONSISTENCE (1/2)

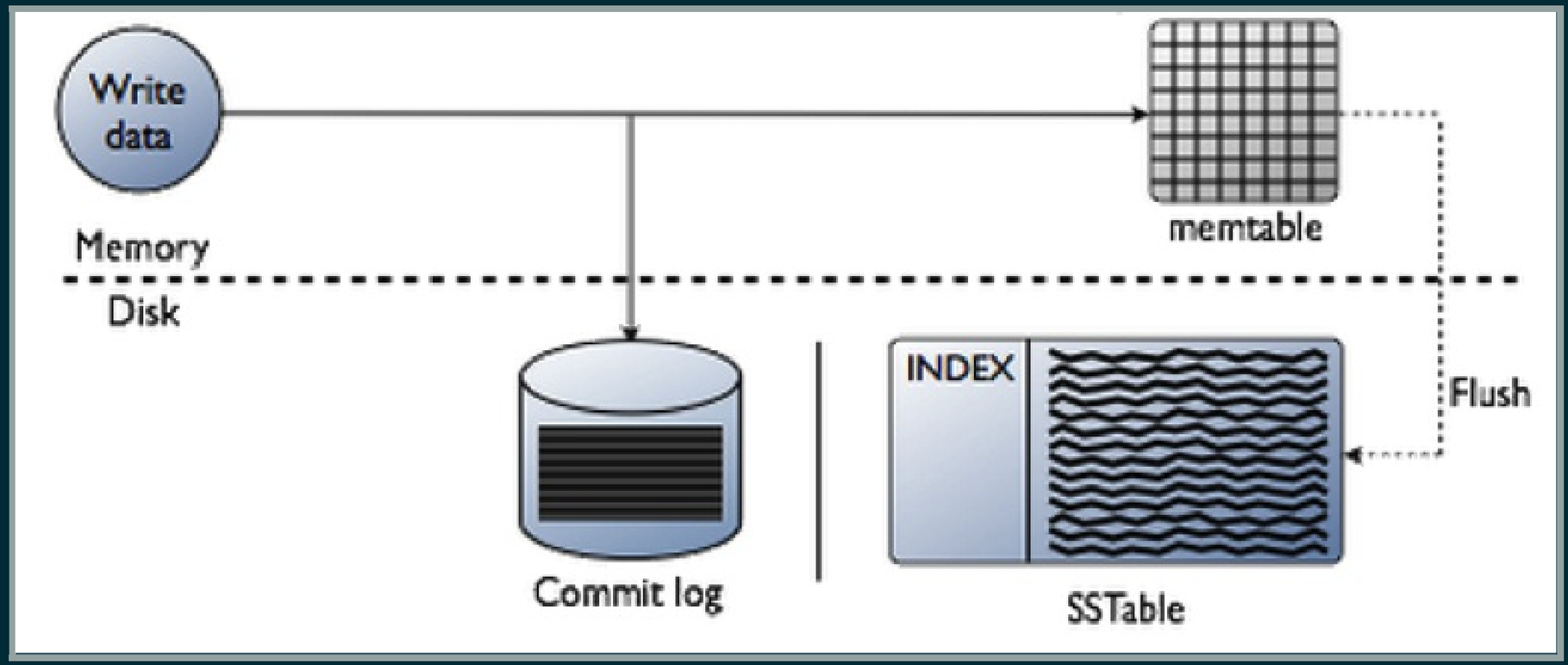
Nom	Usage
ONE	Rapide. Très bonne disponibilité. Ne lit pas forcément la toute dernière valeur.
QUORUM	Bon compromis entre la consistence et la rapidité.
ALL	Plus haut niveau de consistence. Très faible disponibilité.

LES USAGES DES NIVEAUX DE CONSISTENCE (2/2)

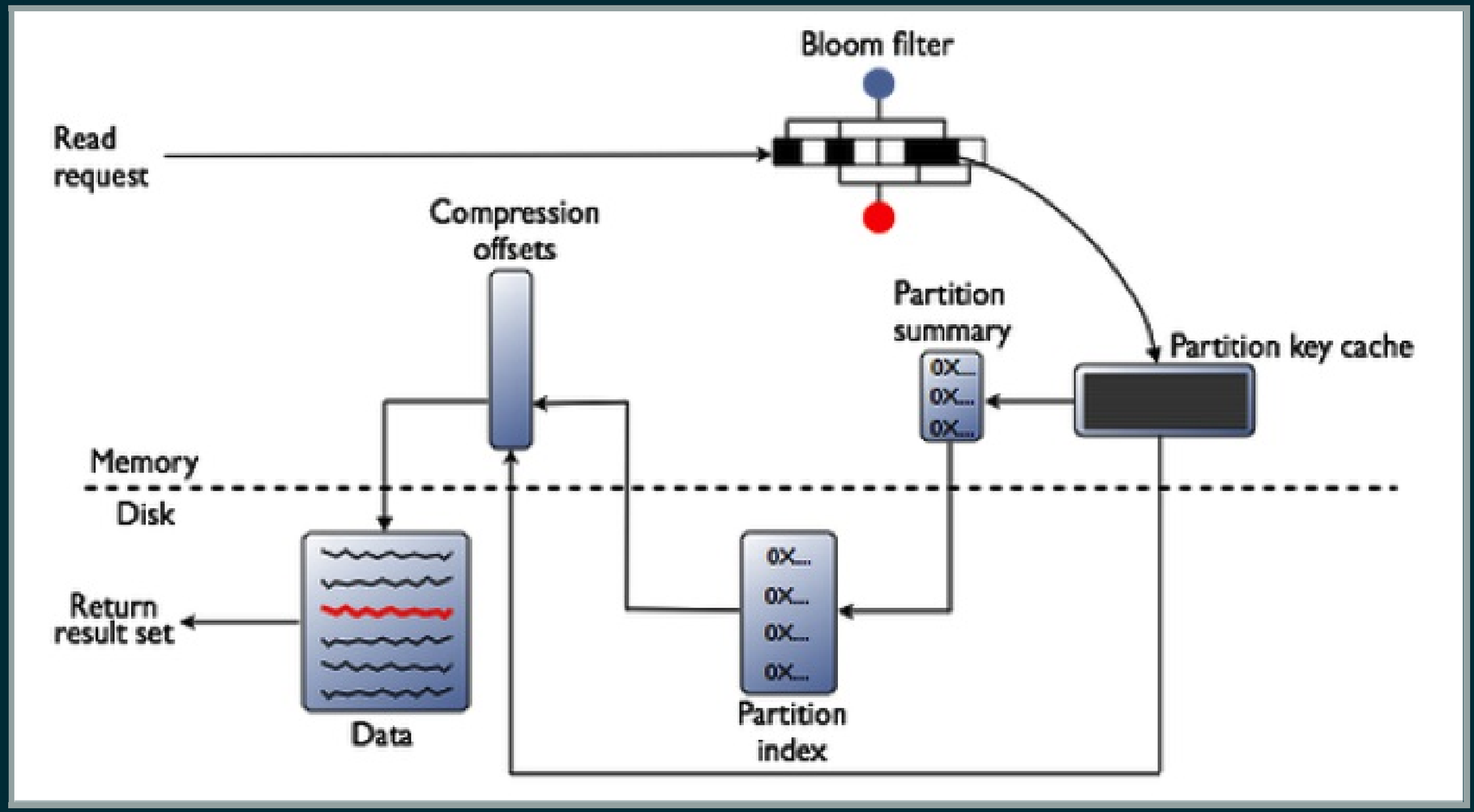
Les deux niveaux les plus couramment utilisés sont:

- ONE read + ONE write: rapide et fonctionne même avec RF - 1 noeuds indisponibles
- QUORUM read + QUORUM write: consistent. On lit toujours la dernière valeur insérée.

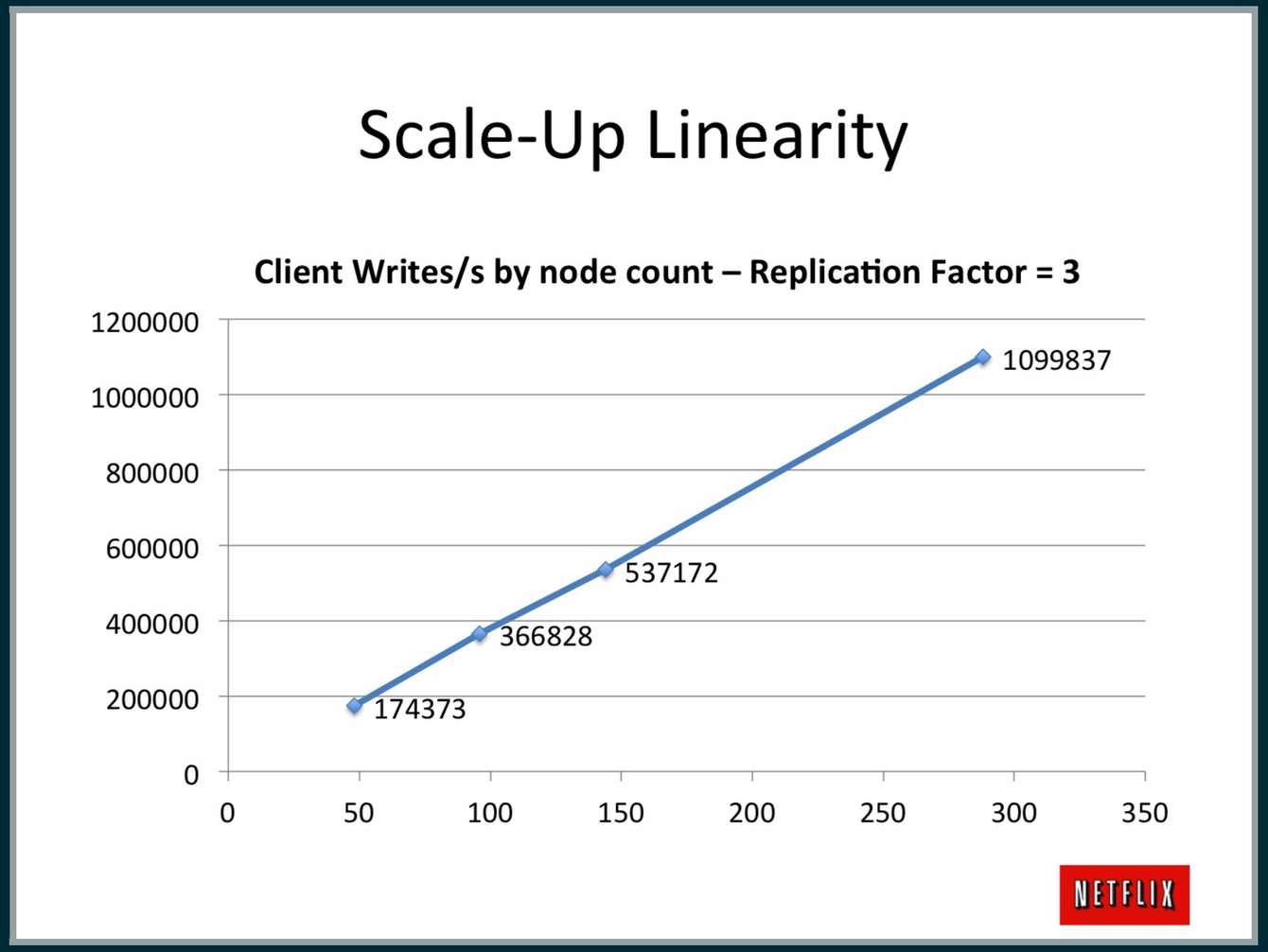
CASSANDRA WRITE PATH



CASSANDRA READ PATH



LA SCALABILITÉ



INSTALLATION ET CONFIGURATION

INSTALLATION DE C*

PRÉREQUIS

- Oracle JDK 1.7 64 bits
- Configurer JAVA_HOME
- Sélectionner sa distribution C*:
  - Apache Cassandra
  - Datastax Community Edition
  - Datastax Enterprise

INSTALLATION DE C*

UTILISATION DU PAQUET DEB SUR NOTRE VM

```
sudo dpkg -i TODO.deb
```

INSTALLATION DE C*

FICHIERS DE CONFIGURATION

- cassandra.yaml: fichier principal de configuration
- cassandra-env.sh: configuration de l'environnement Java (heap size,...)
- logback.xml: configuration des logs

LE FICHIER CASSANDRA.YAML, PRINCIPALES PROPRIÉTÉS

- cluster_name: toutes les machines d'un cluster doivent avoir le même nom
- listen_address: le port sur lequel C* écoute les autres noeuds

INSTALLATION DE C*

ON DÉMARRE !

1. Démarrer le service Cassandra

```
sudo service cassandra start
```

2. Visualiser les logs de C*

```
ls /var/log/cassandra/
```

3. Arrêter le service Cassandra

```
sudo service cassandra stop
```

LES OUTILS CASSANDRA

NODETOOL

Nodetool est le couteau suisse de Cassandra. Il permet d'obtenir des informations et de gérer un cluster de machines. Les commandes les plus couramment utilisées sont:

- status: fournit des informations sur le cluster (état, charge, ID)
- info: fournit des informations sur la machine locale (mémoire, espace disque,...)
- ring: affiche un résumé pour chaque noeud dans le cluster. Permet de repérer les machines déséquilibrées. Privilégier plutôt status et info

NODETOOL

1. Explorer la liste des commandes de nodetool

```
nodetool help
```

2. Obtenir de l'aide sur une commande

```
nodetool help <command>
```

3. Tester les commandes status,info et ring

LES OUTILS CASSANDRA

CQLSH

CQLSH est un shell interactif permettant de tester des commandes dans le langage CQL (Cassandra Query Language). Il offre également d'autres commandes uniquement disponibles dans le shell:

- COPY: import ou export les données au format CSV
- DESCRIBE: fournit des informations sur le cluster, les keyspaces ou les tables
- TRACING: active ou désactive le tracing des requêtes
- SOURCE: exécute un fichier contenant des requêtes CQL
- Et d'autres...

ON CRÉÉ NOTRE PREMIER KEYSPACE !

1. Affichons la liste des keyspaces (Tab pour l'auto-complétion)

```
cqlsh> DESCRIBE KEYSPACES;
```

2. Création d'un keyspace:

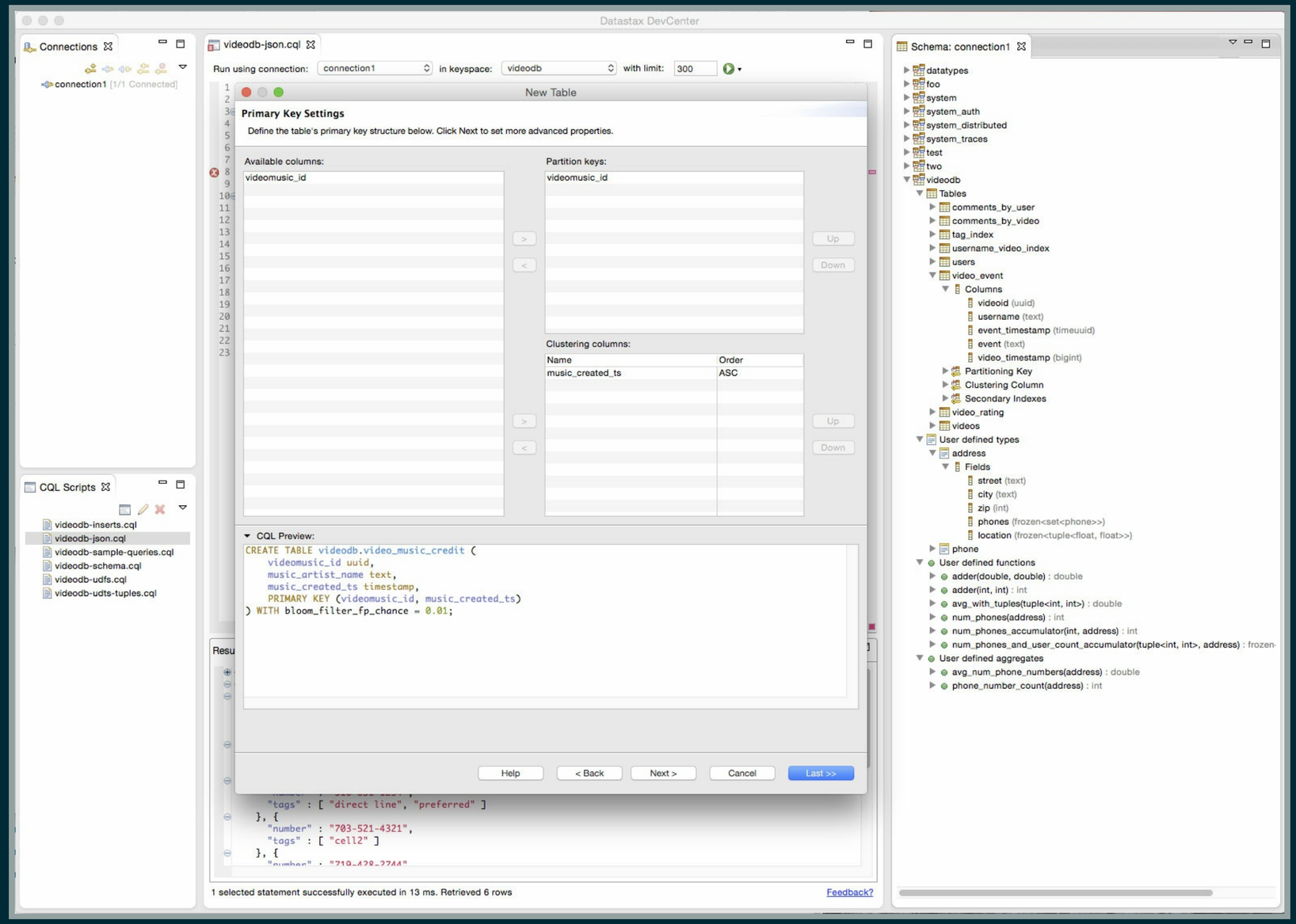
```
CREATE KEYSPACE XCass
        WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor
```

3. Afficher à nouveau la liste des keyspaces et le détail pour notre keyspace XCass

LES OUTILS CASSANDRA

DEVCENTER

DevCenter est un IDE (basé sur Eclipse) permettant de visualiser les tables et d'exécuter des commandes CQL avec auto-complétion, sauvegarder des requêtes et maintenir plusieurs connections simultanées


Datastax DevCenter
Connections
connection1 [1/1 Connected]
CQL Scripts
videodb-inserts.cql
videodb-json.cql
videodb-sample-queries.cql
videodb-schema.cql
videodb-udfs.cql
videodb-udts-tuples.cql
videodb-json.cql
Run using connection: connection1 in keyspace: videodb with limit: 300
New Table
Primary Key Settings
Define the table's primary key structure below. Click Next to set more advanced properties.
Available columns:
videomusic_id
Partition keys:
videomusic_id
Clustering columns:
Name
Order
music_created_ts
ASC
Up
Down
CQL Preview:
CREATE TABLE videodb.video_music_credit (
videomusic_id uuid,
music_artist_name text,
music_created_ts timestamp,
PRIMARY KEY (videomusic_id, music_created_ts)
) WITH bloom_filter_fp_chance = 0.01;
Help
< Back
Next >
Cancel
Last >>
"tags" : ["direct line", "preferred"]
"number" : "703-521-4321",
"tags" : ["cell2"]
1 selected statement successfully executed in 13 ms. Retrieved 6 rows
Feedback?
Schema: connection1
datatypes
foo
system
system_auth
system_distributed
system_traces
test
two
videodb
Tables
comments_by_user
comments_by_video
tag_index
username_video_index
users
video_event
Columns
videoid (uuid)
username (text)
event_timestamp (timeuuid)
event (text)
video_timestamp (bigint)
Partitioning Key
Clustering Column
Secondary Indexes
video_rating
videos
User defined types
address
Fields
street (text)
city (text)
zip (int)
phones (frozen<set<phone>>)
location (frozen<tuple<float, float>>)
phone
User defined functions
adder(double, double) : double
adder(int, int) : int
avg_with_tuples(tuple<int, int>) : double
num_phones(address) : int
num_phones_accumulator(int, address) : int
num_phones_and_user_count_accumulator(tuple<int, int>, address) : frozen
User defined aggregates
avg_num_phone_numbers(address) : double
phone_number_count(address) : int

MODÈLE DE DONNÉES

PRINCIPES DE MODÉLISATIONS DES DONNÉES

Construire un schéma de base relationnelle passe bien souvent par la description des types d'objets, des attributs et des liens entre les tables. Les requêtes arrivent bien souvent ensuite.

Pour Cassandra il est primordial de commencer par les requêtes. Les tables sont ensuites construites pour supporter ces requêtes. Cela permet d'obtenir un modèle répondant rapidement aux requêtes et scalant correctement.

L'APPLICATION XCASS

CLÉ PRIMAIRE

Clé de partitionement, colonnes de clustering, clé composée,...

CLÉ DE PARTITIONEMENT

Toutes les requêtes doivent spécifier la clé de partitionement. Pas de range query (=> full cluster scan)

STOCKAGE PHYSIQUE DES DONNÉES

partitions, partition key,clustering columns,cellules,...

LES TYPES DE DONNÉES

UUID, Time UUID, ...

PREMIÈRE TABLE ET PREMIÈRES REQUÊTES

Créer une table permettant d'héberger les données d'un objet connecté

UPSERT

LWW: last write wins

CLUSTERING COLUMNS

Possibiliter d'ordonner les colonnes de clustering

Possibiliter de faire des range requests

REPRENDRE L'EXEMPLE PRÉCÉDENT EN ORDONNANT LES DONNÉES DE LA PLUS RÉCENTE À LA PLUS ANCIENNE

2 possibilités: directement dans la requête ou lors de la construction de la table

INDEX SECONDAIRE

Quand ne pas les utiliser, comment faire autrement ?

LES COLLECTIONS

UDT: USER DEFINED TYPE

LES COMPTEURS

LA DÉNORMALISATION

COMMENT MODÉLISER UNE RELATION 1-N ?

LWT

CRÉER UNE APPLICATION JAVA SCALABLE

PRÉPARER CASSANDRA POUR LA PRODUCTION

MERCI !