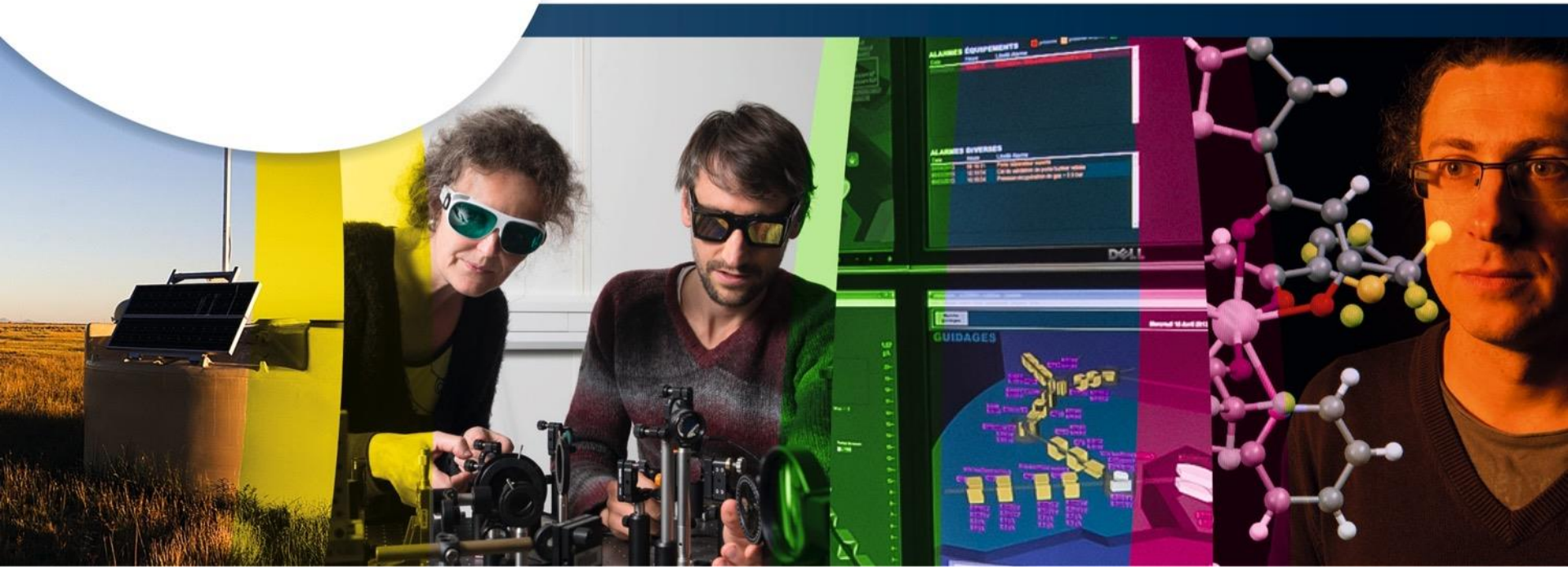
Bases de données Colonnes avec

www.cnrs.fr



www.cnrs.fr

Sommaire

- Introduction
- Architecture
- Modèle de données
- Travaux pratiques

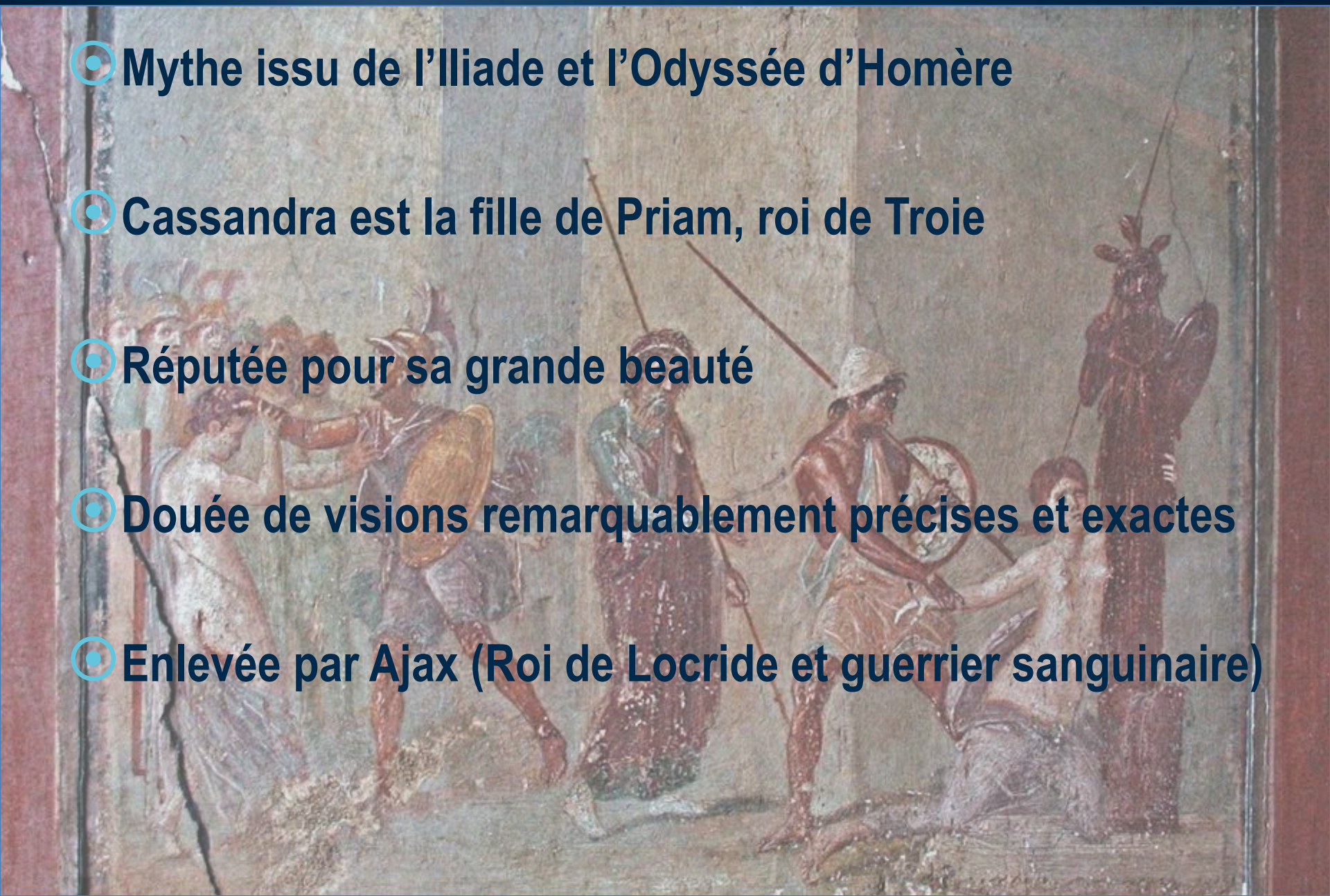
www.cnrs.fr

Introduction

Historique

- **Développé par facebook Avinash LAKSHMAN**
  (un des développeurs de la base Dynamo d'Amazon)
- **Inspiré de Big Table développé par Google**
- **Code libéré en 2008**
- **Top Level Apache Project en 2010**
- **Abandonné par Facebook en 2010**

Historique



- Mythe issu de l'Iliade et l'Odyssée d'Homère
- Cassandra est la fille de Priam, roi de Troie
- Réputée pour sa grande beauté
- Douée de visions remarquablement précises et exactes
- Enlevée par Ajax (Roi de Locride et guerrier sanguinaire)

Qu'est-ce que Cassandra?

- **Base de données orientées colonnes**
- **Scalable**
- **Distribuée (peer-to-peer)**
- **Massivement parallèle**

- **Objectifs**
  - gérer une grande quantité de données
  - résister aux pannes
  - être performante

Mises en œuvre

- Facebook
- Twitter
- Digg
- Cisco WebEx
- IBM
- eBay
- NetFlix

Installation

Ajout du dépôt officiel pour Ubuntu

- `sudo apt-key adv --keyserver pgp.mit.edu --recv F758CE318D77295D`
- `sudo apt-key adv --keyserver pgp.mit.edu --recv 749D6EEC0353B12C`
- `echo "deb http://www.apache.org/dist/cassandra/debian 21x main" | sudo tee /etc/apt/sources.list.d/cassandra-2.1.list`
- `echo "deb-src http://www.apache.org/dist/cassandra/debian 21x main" | sudo tee /etc/apt/sources.list.d/cassandra-src-2.1.list`
- `sudo apt-get update`

Installation depuis le dépôt

- `sudo apt-get install cassandra`

Lancement

- `cassandra [-f]`

Connection

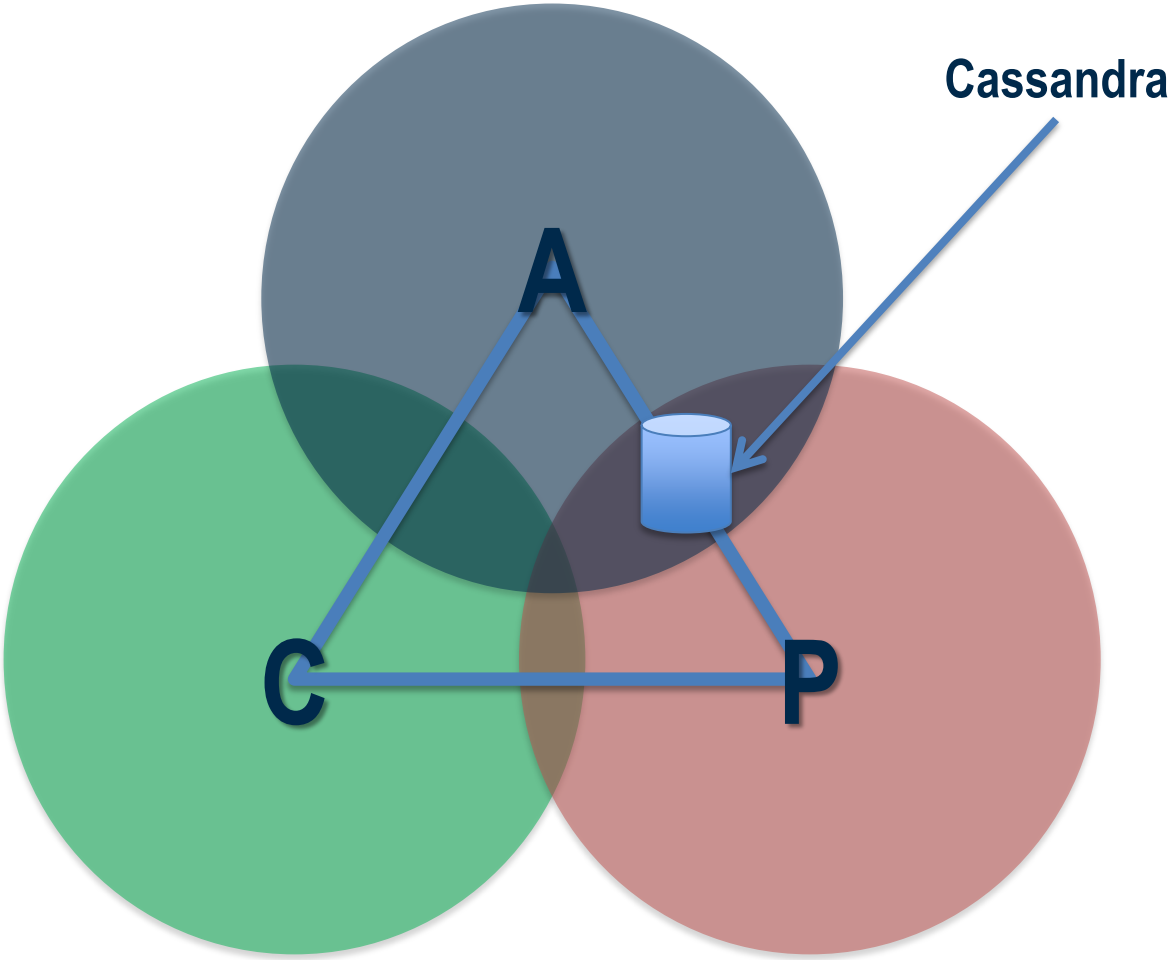
- `cqlsh`
- `cassendra-cli`

www.cnrs.fr

Architecture

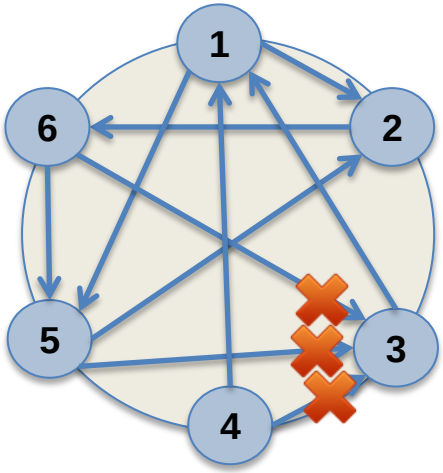
- CAP
- Cluster
- Réplication

CAP



Cluster

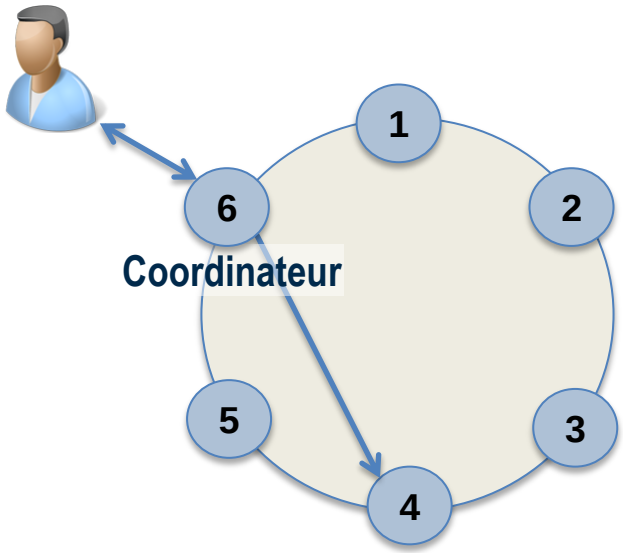
- **Tous les nœuds ont la même importance**
- **Tous les nœuds communiquent avec les autres**
  - Chaque nœud contacte entre 1et 3 nœuds toutes les secondes
    - Protocole de bavardage (gossip)
    - Conçu pour éviter la formation d'ilots indépendants
    - Conçu pour que l'information se propage vite



Cluster

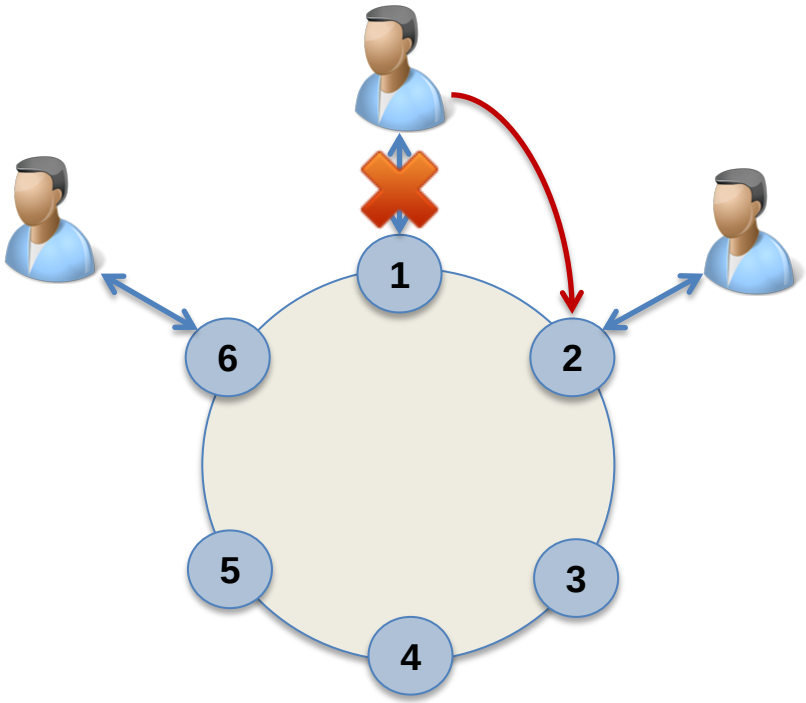
- **Lecture/écriture depuis n'importe quel nœud**
- **Le client ne sait pas où la donnée se trouve**
  1. le client contacte n'importe quel nœud
  2. le nœud contacté («storage proxy») sert de coordinateur

  Note: Le client peut disposer d'une pool de connexions vers quelques nœuds.



Réplication

- Pas de point faible (SPOF= Single Point of Failure)



Réplication

- **Combien de réplicas doivent répondre pour considérer l'opération réussie ?**
- **Consistance ajustable**
  - Pour la lecture et l'écriture
  - Choix entre consistance forte ou partielle
  - A régler en fonction du besoin

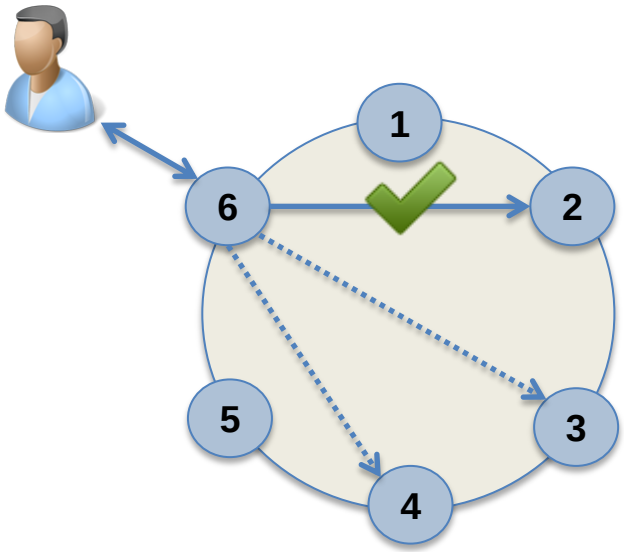
Réplication en écriture

Level	Vérification de l'écriture sur
ANY	1 nœud (inclu hinted handoff)
ONE	1 nœud
QUORUM	N/2 + 1 replicas
LOCAL_QUORUM	N/2 + 1 replicas dans le datacenter local
EACH_QUORUM	N/2 + 1 replicas dans tous les datacenter local
ALL	Tous les replicas

Réplication en écriture

Consitency Level ONE

- Exemple avec un **facteur de réplication** à **3**



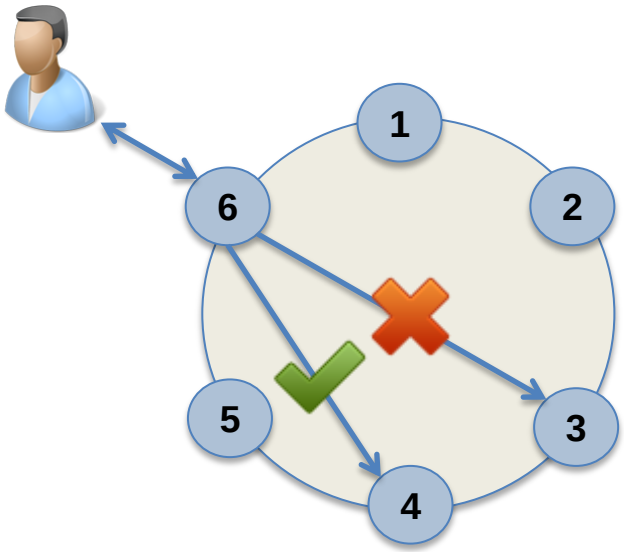
Opération réussie

Écriture asynchrone

Réplication en écriture

Consitency Level ONE

- Exemple avec un **facteur de réplication** à **3**

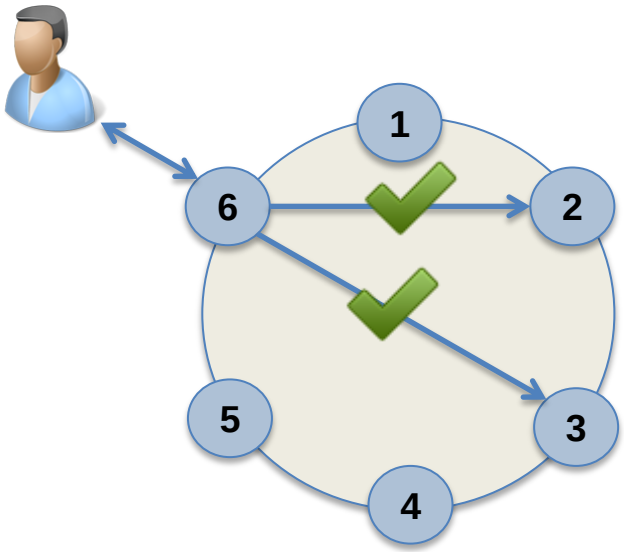


Puis en background

Réplication en écriture

Consitency Level QUORUM

- Exemple avec **un facteur de réplication** à **2**

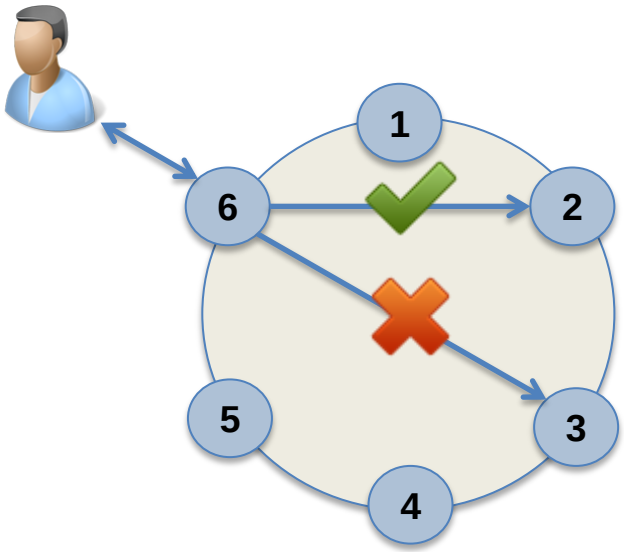


Opération réussie

Réplication en écriture

Consitency Level QUORUM

- Exemple avec un **facteur de réplication** à **2**

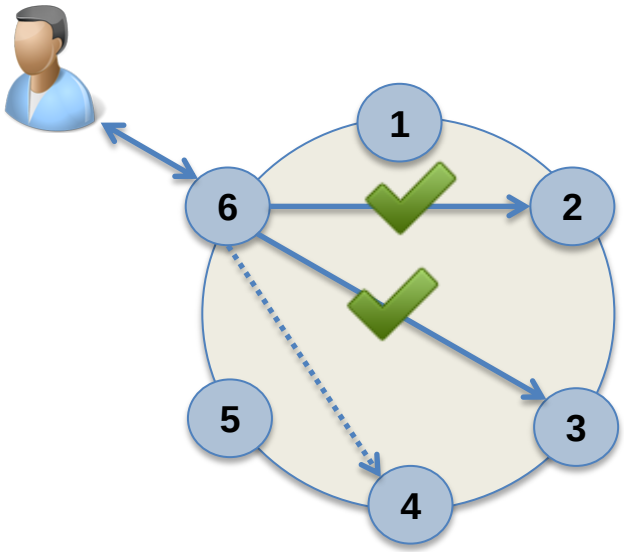


Échec

Réplication en écriture

Consitency Level QUORUM

- Exemple avec un **facteur de réplication** à **3**



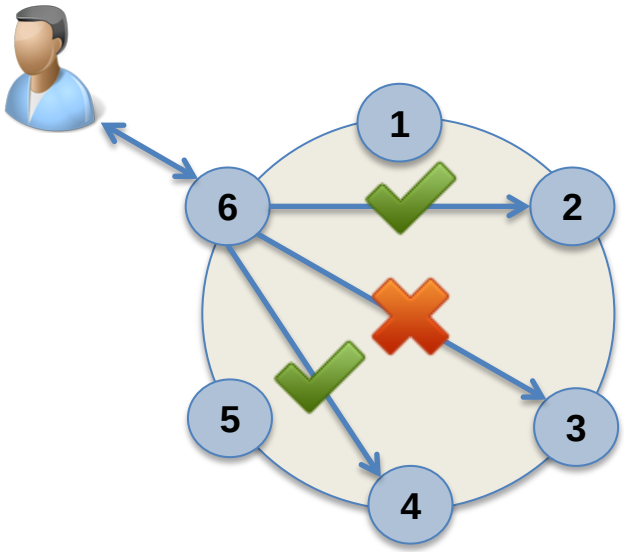
Opération réussie

Écriture asynchrone

Réplication en écriture

Consitency Level QUORUM

- Exemple avec **un facteur de réplication** à **3**



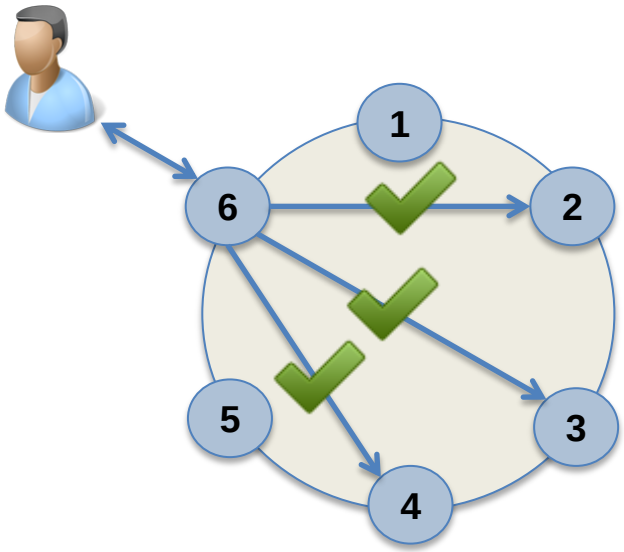
Opération réussie

Écriture asynchrone

Réplication en écriture

Consitency Level ALL

- Exemple avec un **facteur de réplication** à **3**



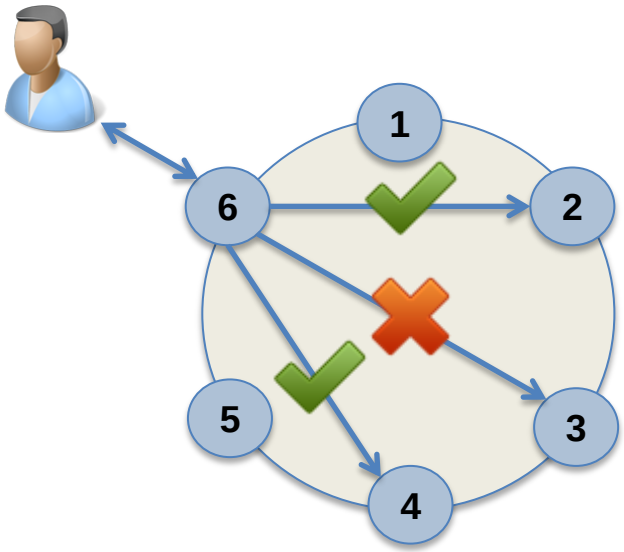
Opération réussie

Écriture asynchrone

Réplication en écriture

Consitency Level ALL

- Exemple avec un **facteur de réplication** à **3**

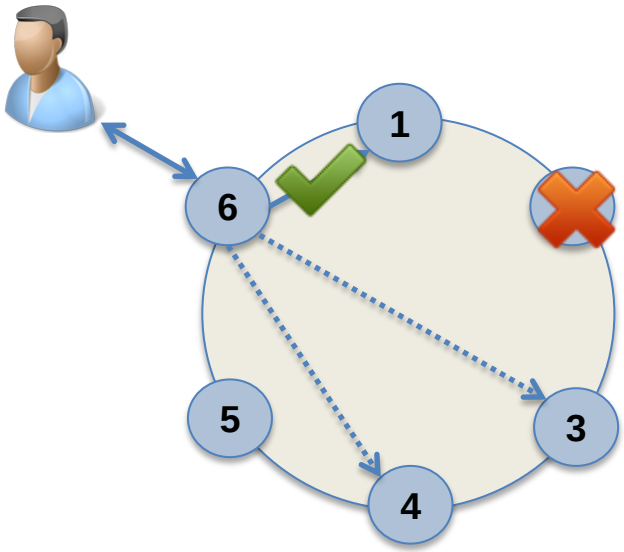


Échec

Réplication en écriture

Consitency Level ANY

- Exemple avec un **facteur de réplication** à **3**



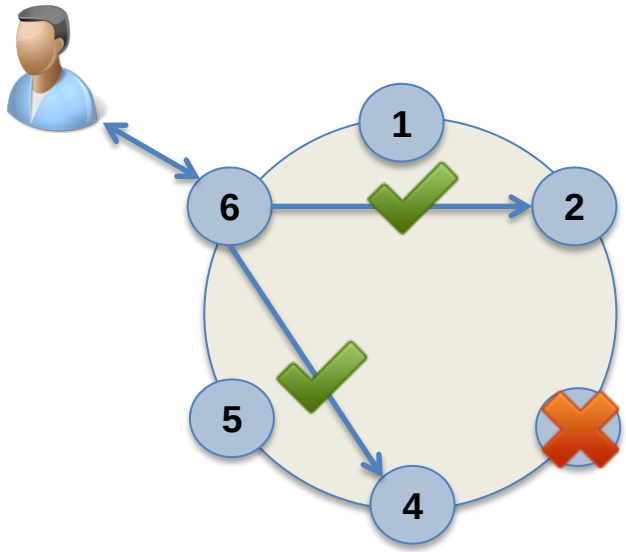
Opération réussie

Écriture asynchrone

Réplication en écriture

Hinted Handoff

- Exemple avec
  - facteur de réplication à 3
  - Consitency Level à QUORUM

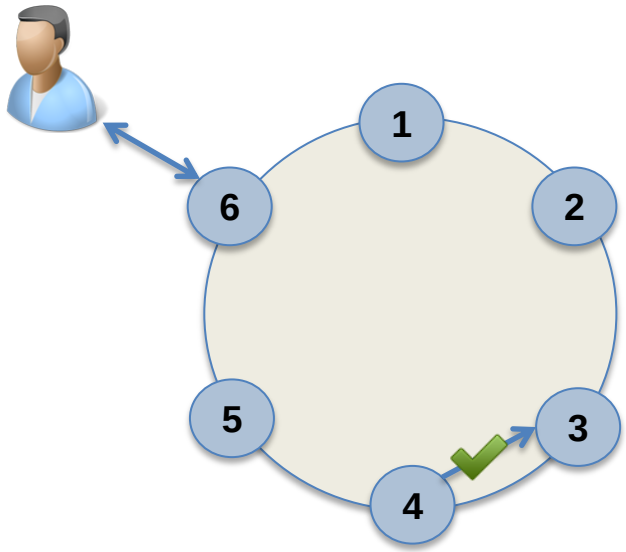


Opération réussie

Réplication en écriture

Hinted Handoff

- Lorsque le nœud 3 revient

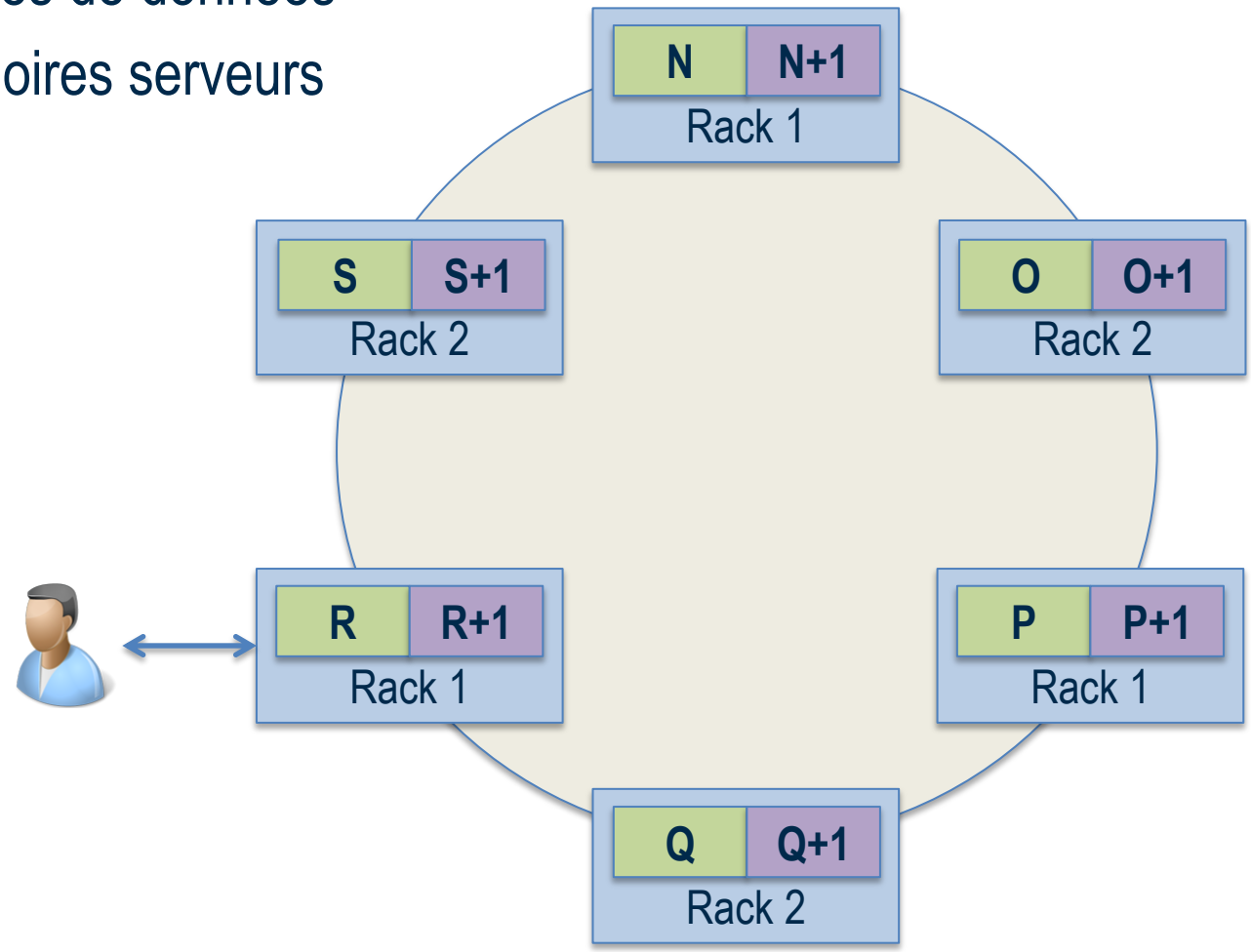


Envoi de la donnée manquante

Attention à
`max_hint_window_in_ms` (1h par défaut)

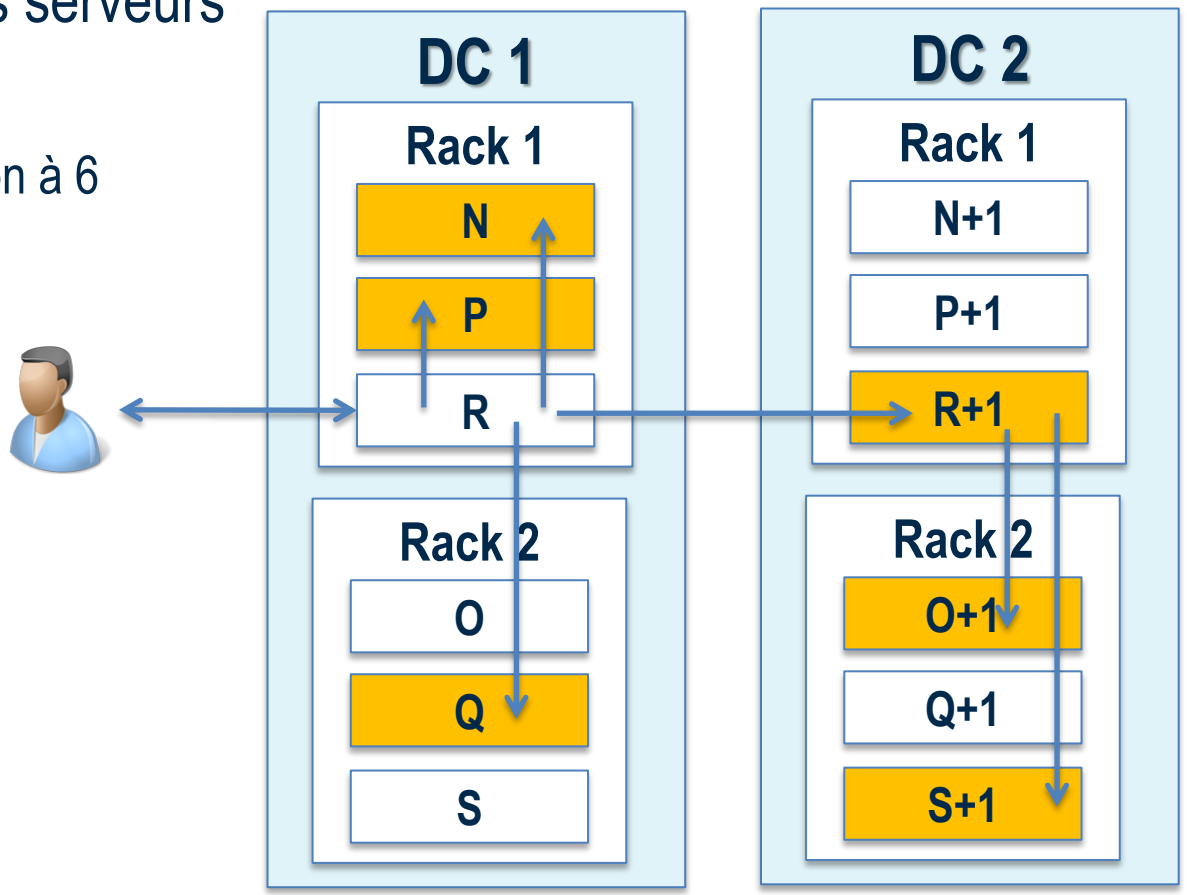
Réplication en écriture

- **Réplication possible entre**
  - Différents centres de données
  - Différentes armoires serveurs



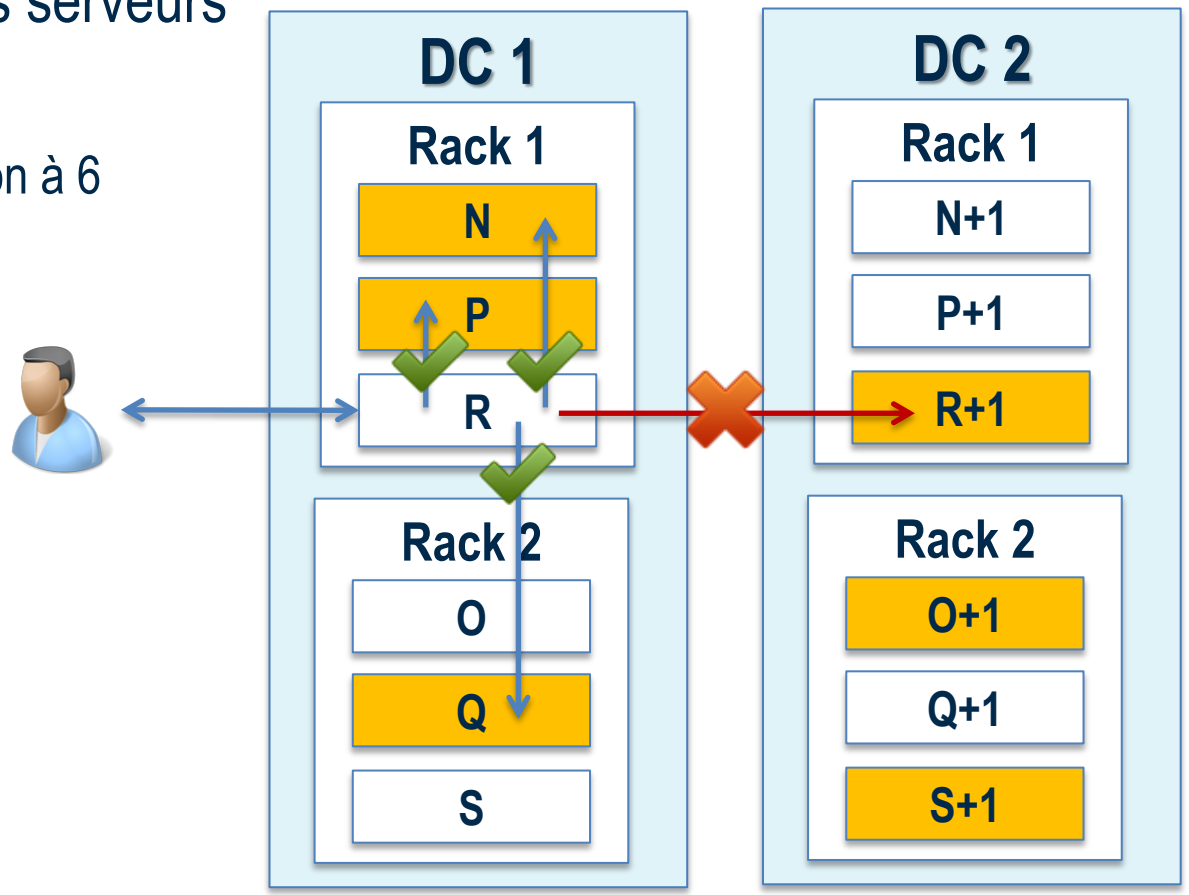
Réplication en écriture

- **Réplication possible entre**
  - Différents centres de données
  - Différentes armoires serveurs
  - Exemple avec
    - facteur de réplication à 6



Réplication en écriture

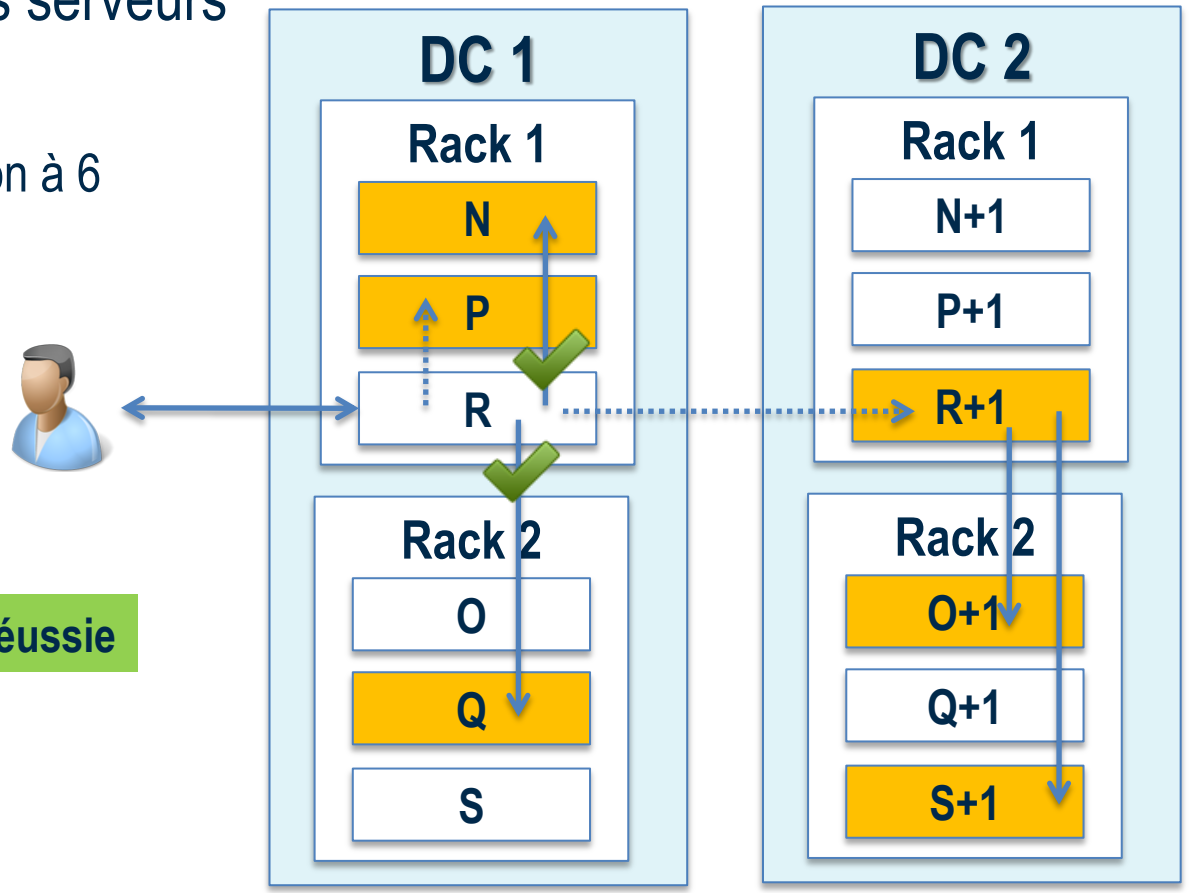
- **Consitency Level QUORUM**
  - Différents centres de données
  - Différentes armoires serveurs
  - Exemple avec
    - facteur de réplication à 6



Échec

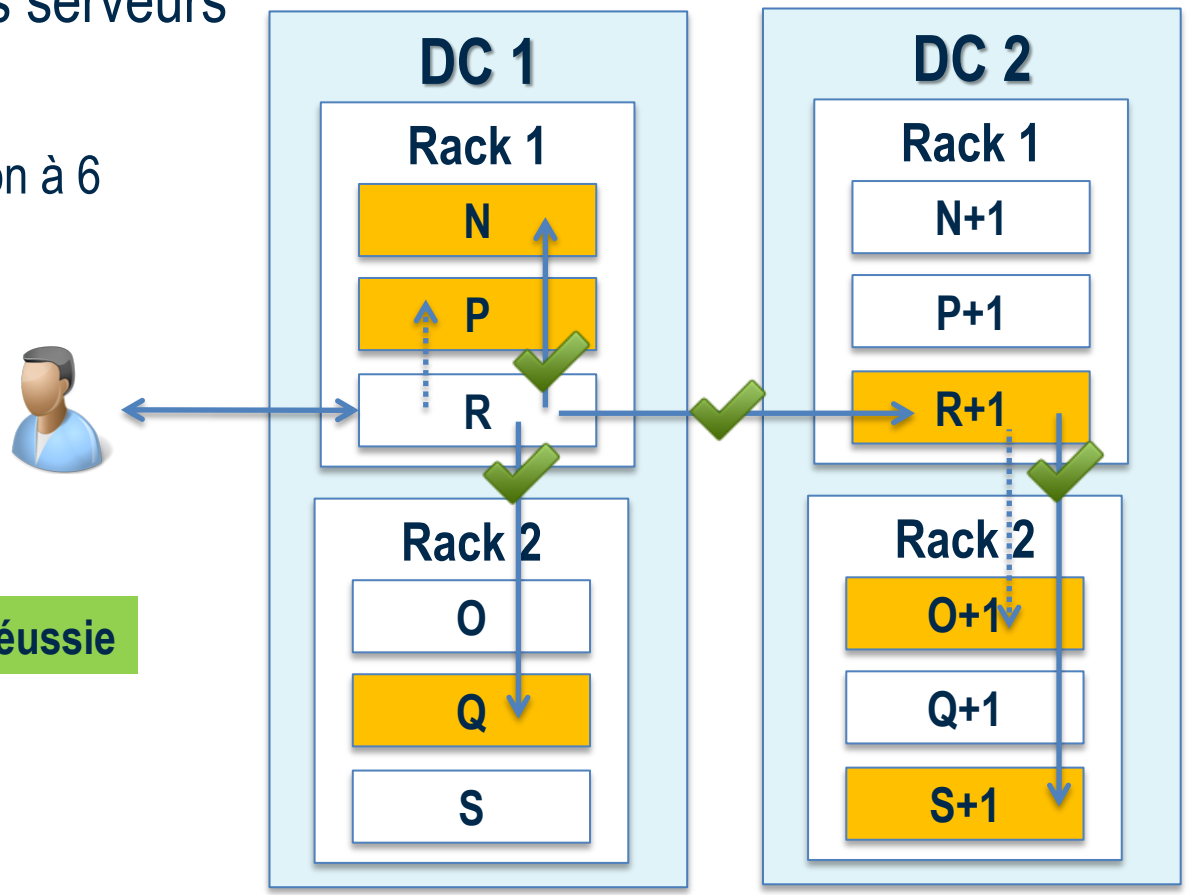
Réplication en écriture

- **Consitency Level LOCAL_QUORUM**
  - Différents centres de données
  - Différentes armoires serveurs
  - Exemple avec
    - facteur de réplication à 6



Réplication en écriture

- **Consitency Level EACH_QUORUM**
  - Différents centres de données
  - Différentes armoires serveurs
  - Exemple avec
    - facteur de réplication à 6



Réplication en écriture

Configuration de la topologie

- Décrire le nœud courant

CASSANDRA_HOME/conf/cassandra-rackdc.properties

```
dc=dc1
rack=rack1
```

- Lister les adresses connues

CASSANDRA_HOME/conf/cassandra.yaml

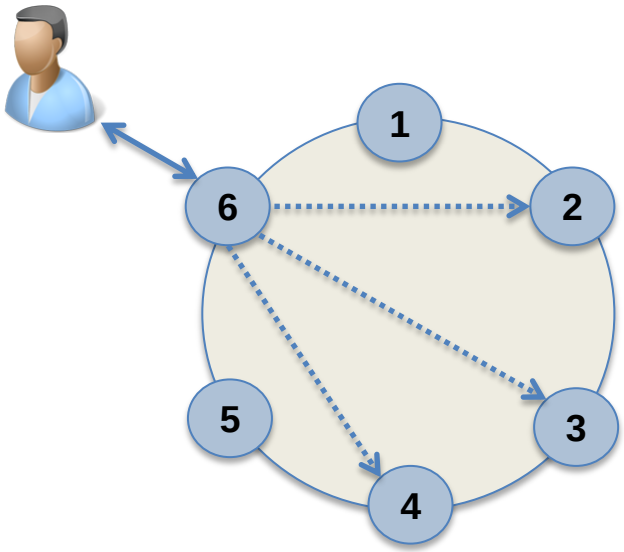
```
seeds: "127.0.0.1","127.0.0.2"
```

Réplication en lecture

Level	Vérification des valeurs sur
ONE	1 nœud (1ère réponse)
QUORUM	N/2 + 1 replicas
LOCAL_QUORUM	N/2 + 1 replicas dans le datacenter local
EACH_QUORUM	N/2 + 1 replicas dans tous les datacenter local
ALL	Tous les replicas

Réplication en lecture

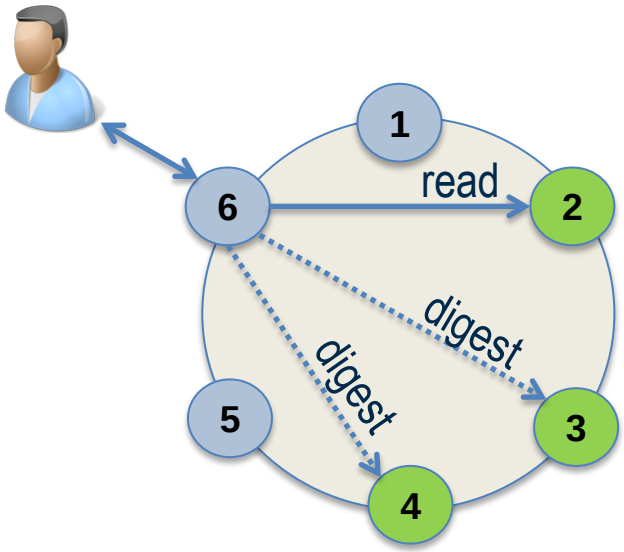
- **Le client ne sait pas où la donnée sera lue**
  - Exemple de réplication facteur 3



Réplication en lecture

Consitency Level ONE

- Exemple avec un facteur de réplication à 3



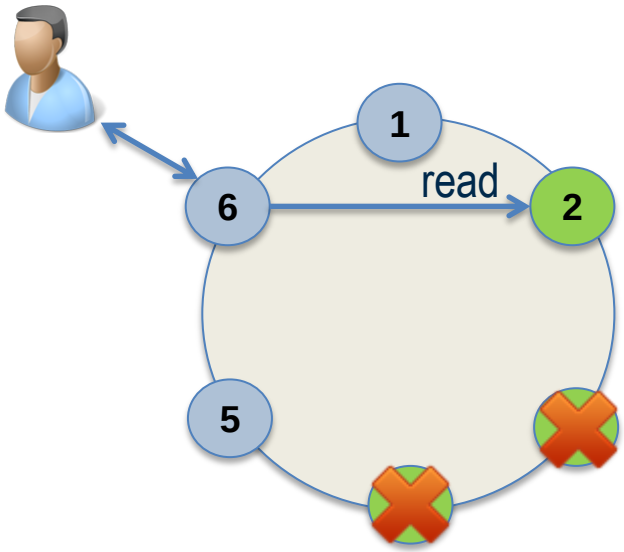
Opération réussie

lecture asynchrone , puis « read repair » si besoin et si configuré

Réplication en lecture

Consitency Level ONE

- Exemple avec un facteur de réplication à 3



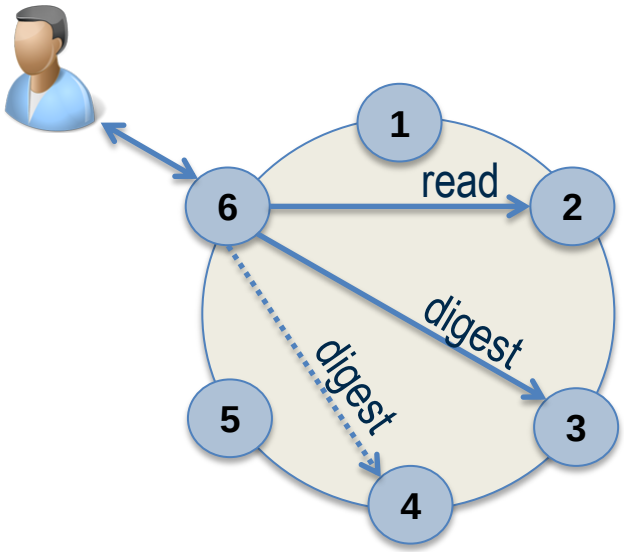
Opération réussie

lecture asynchrone , puis « read repair » si besoin et si configuré

Réplication en lecture

Consitency Level QUORUM

- Exemple avec un facteur de réplication à 3



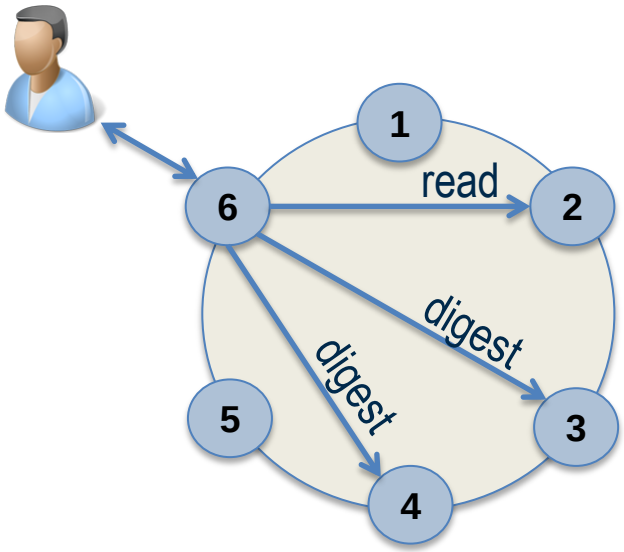
Opération réussie

Écriture asynchrone

Réplication en lecture

Consitency Level ALL

- Exemple avec un facteur de réplication à 3



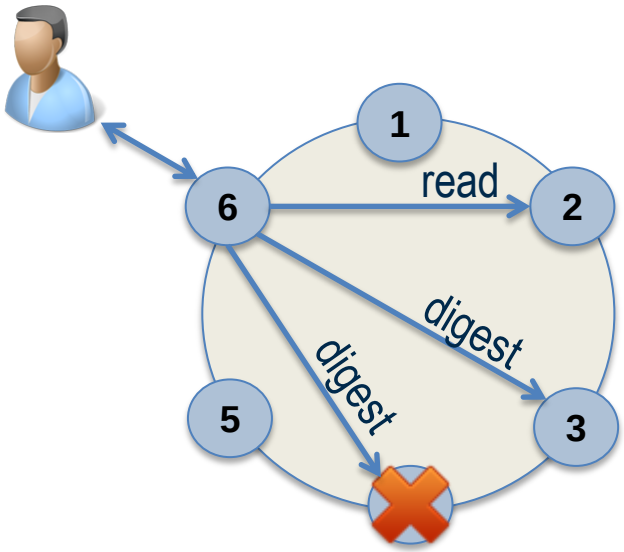
Opération réussie

Écriture asynchrone

Réplication en lecture

Consitency Level ALL

- Exemple avec un facteur de réplication à 3



Échec

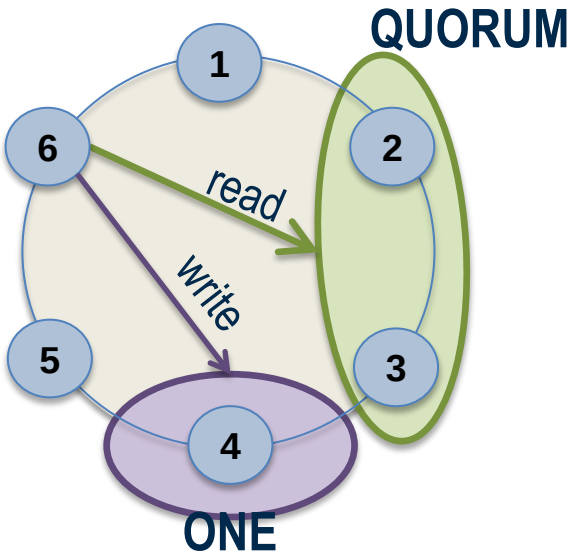
Écriture asynchrone

Réplication

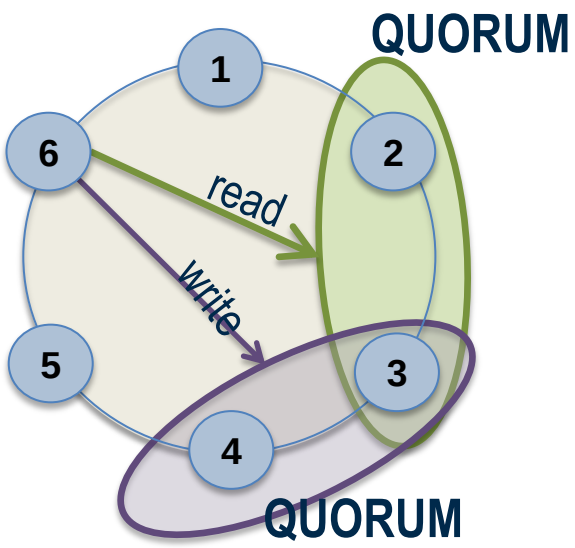
Consistance

- Exemple avec un facteur de réplication à 3

Faible



Forte

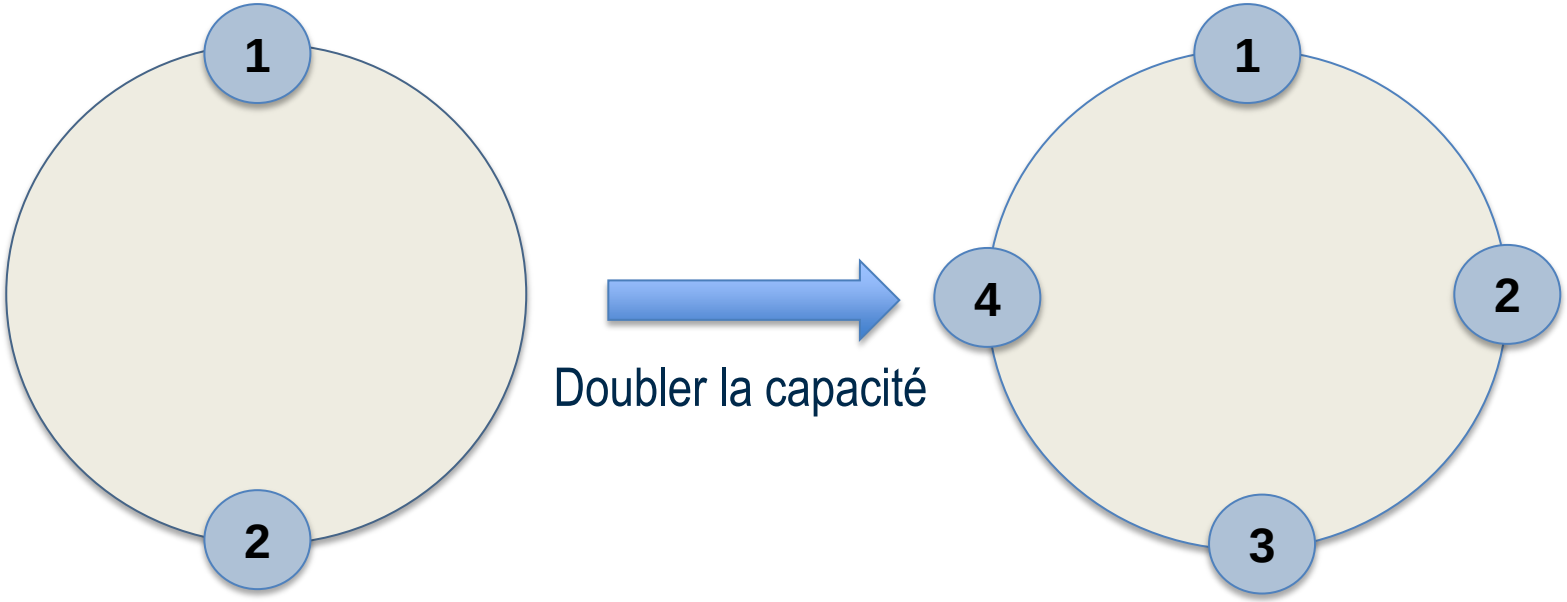


Réplication

- **Consistance forte**
  - Formule : **W + R > RF**
    - W: nb de réplicas contactées en écriture
    - R: nb de réplicas contactées en lecture
    - RF: réplication factor
  - Cas trivial 1 :
    - écriture ALL
    - lecture ONE
  - Cas trivial 2 :
    - écriture QUORUM
    - lecture QUORUM

Scalabilité

- Ajout de nœud en ligne
- Augmentation linéaire de la performance



Exercices

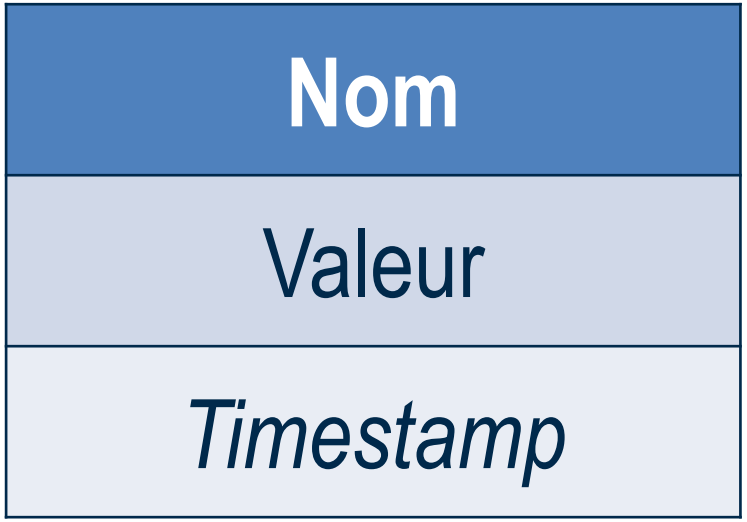
- **Se connecter à la VM**
  - https://192.168.56.4:4200/ ou dans VirtualBox
  - `su -`
- **Lancer le nœud 1**
  - `cd /home/cassandra/cassandra1/bin`
  - `./cassandra`
- **Vérifier l'état**
  - `./nodetool info`
  - `./nodetool status`
- **Lancer le nœud 2 et vérifier l'état**
  - `cd /home/cassandra/cassandra2/bin`

www.cnrs.fr

Modèle de données

- De la colonne à
  la famille de colonnes
- Partitionnement
- Indexation

Colonne



Colonne

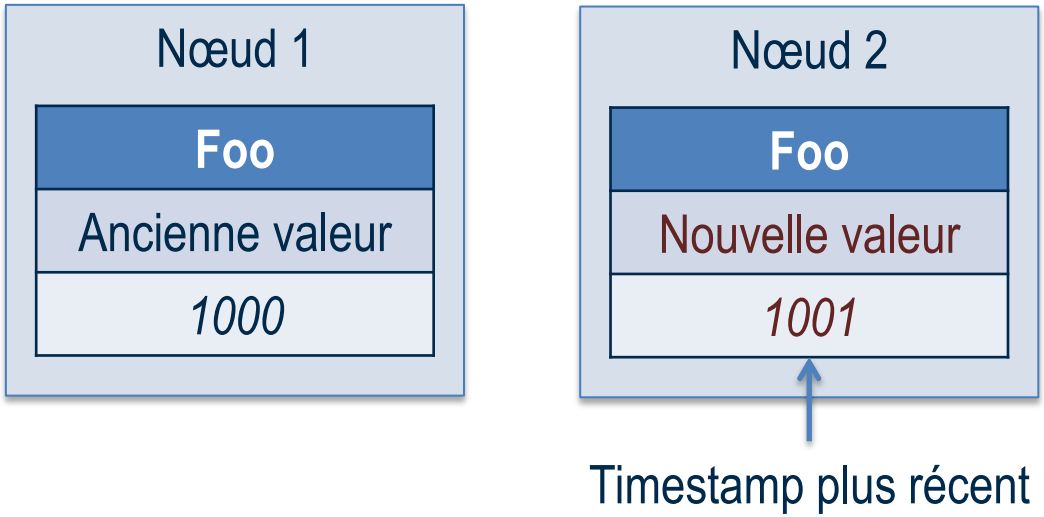
- **Nom**

  Porteur d'information pour le requêtage

- **Valeur : 2Go maximum**

- **Timestamp**

  Utilisé par Cassandra pour résoudre les conflits dus à l'architecture distibuée



Colonne

Types de valeur

Internal /CLI Type	CQL Name	Description
BytesType	blob	Arbitrary hexadecimal bytes (no validation)
AsciiType	ascii	US-ASCII character string
UTF8Type	text, varchar	UTF-8 encoded string
IntegerType	varint	Arbitrary-precision integer
Int32Type	int	4-byte integer
InetAddressType	inet	IP address string in IPv4 or IPv6 format
LongType	bigint	8-byte long
UUIDType	uuid	UUID
TimeUUIDType	timeuuid	Type 1 UUID only (CQL3)
DateType	timestamp	Date plus time, encoded as 8 bytes since epoch
BooleanType	boolean	true or false
FloatType	float	4-byte floating point
DoubleType	double	8-byte floating point
DecimalType	decimal	Variable-precision decimal
CounterColumnType	counter	Distributed counter value (8-byte long)

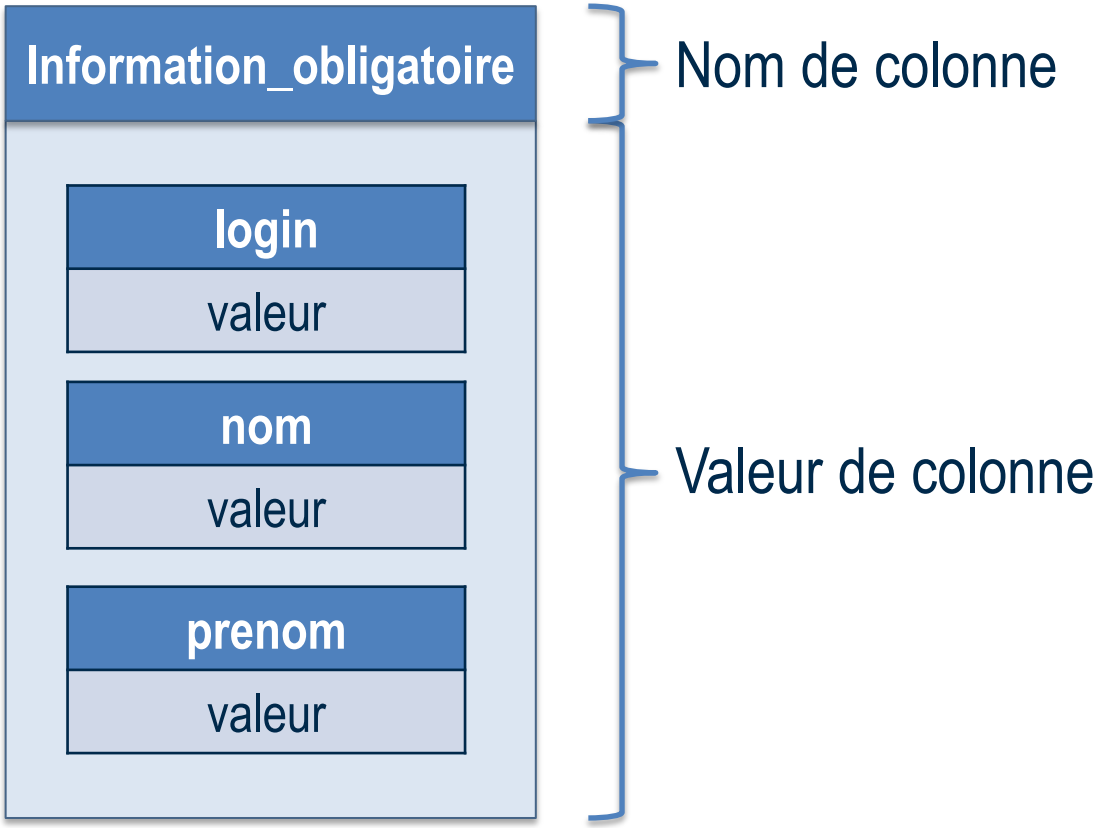
Colonne composite

- **Nom de colonne porte plusieurs informations**

Critère 1	Critère 2	Critère 3
Valeur		

Supercolonne

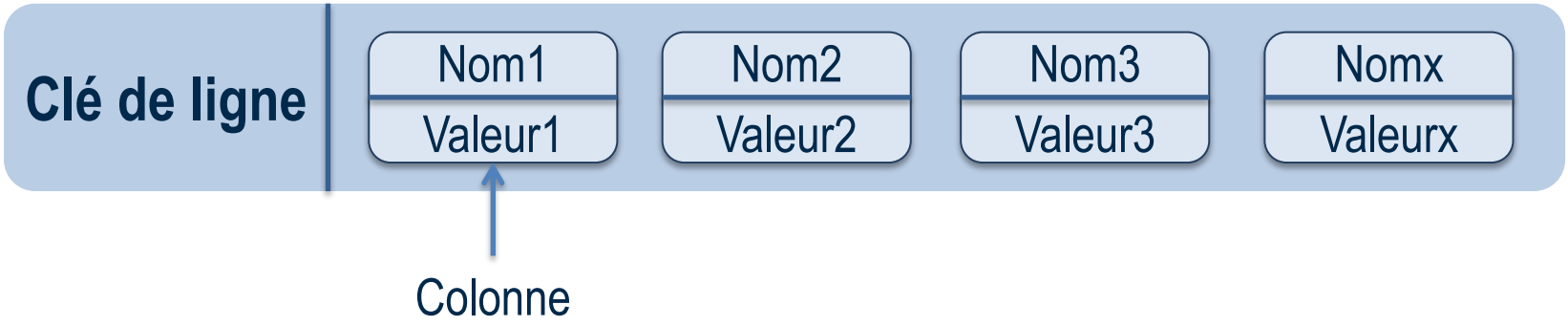
- Colonne contenant des colonnes



- A éviter

Ligne

- **Ensemble de colonnes**
  - Identifié par sa clé
  - 2 milliards de colonnes maximum
  - 0 contraintes sur les colonnes



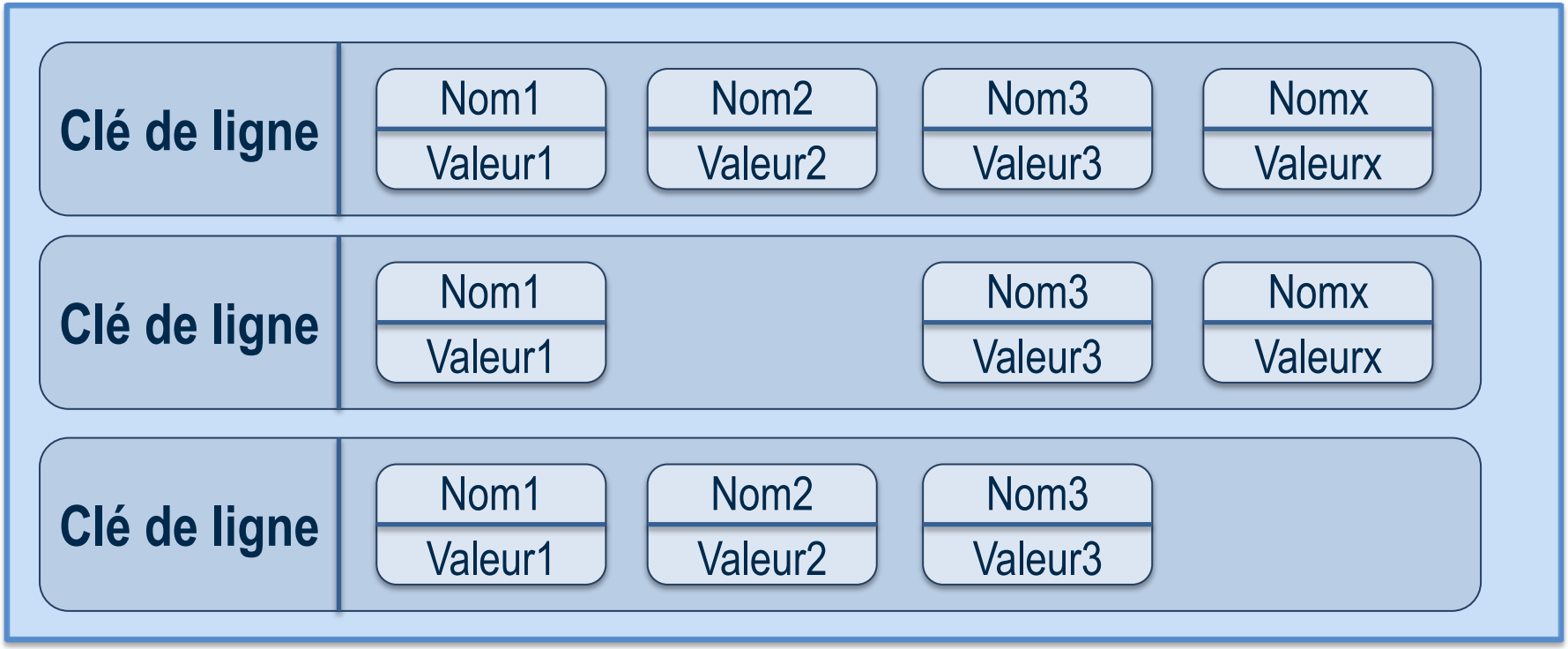
- **1 ligne ne peut pas être scindée sur plusieurs nœuds**
  - La taille maximum d'une ligne est la taille disque du nœud

Ligne

- **Ordre des colonnes**
  - Choix de la stratégie de tri : Comparator
    - ASCII
    - UTF8
    - Long
    - UUID
    - CompositeType
      - Utile pour les colonnes avec plusieurs données
      - Alternative aux supercolonnes
      - ex : UTF8Type, UTF8Type, UTF8Type pour le login, nom, prenom

Famille de colonnes

- **Conteneur de colonnes, supercolonnes**
  - Equivalent à une table dans une base de données relationnelle



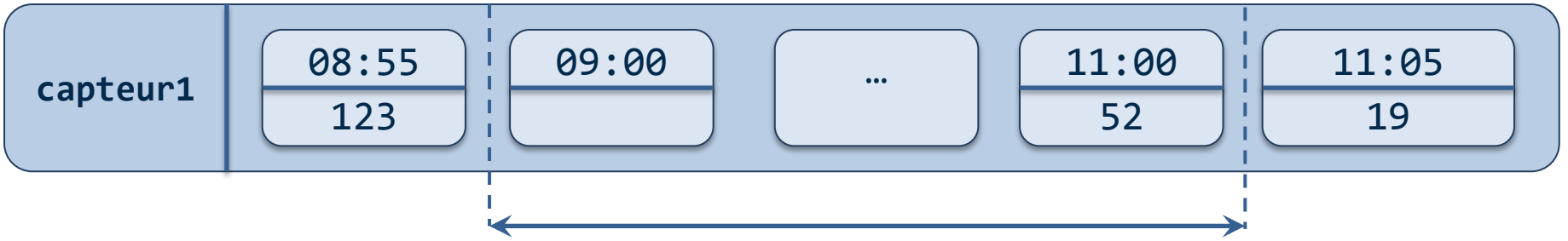
La colonne est une valeur !

Nom de colonne

Tous les trains qui passent à Valence

Tri sur les colonnes (slice)

Toutes les données de 9h à 11h



MAIS PAS DE JOINTURE !

Keyspace

- **Conteneur de plus haut niveau**
  - Equivalent à un schéma dans une base de données relationnelle
- **Pas d'intégrité référentielle**
  - Donc pas de formes normales

Exemple de modélisation

NETFLIX

Partitionnement

- **Les données sont réparties vers tous les nœuds**
  - En fonction de la clé primaire
- **Stratégie de placement**
  - Repose sur la clé de ligne => token
  - Murmur3Partitioner (pardéfaut)
    - Répartir uniformément les différentes lignes entre les nœuds
    - Basé sur le hash MurmurHash des clés
  - RandomPartitioner
    - Répartir uniformément les différentes lignes entre les nœuds
    - Basé sur le hash MD5 des clés
  - ByteOrderedPartitioner
    - Répartir suivant un ordre de nœuds

Index

- **Index primaire (*partition key*)**
  - Automatiquement créé
  - Sur les clés de ligne
  - Permet de rapidement trouver les lignes présentes sur les nœuds

- **Index secondaire (*clustering key*)**
  - Sur la valeur des colonnes
  - Permet de filtrer les résultat sur les valeurs indexées

www.cnrs.fr

Travaux Dirigés

Wide rows (lignes sauvages)

Exercices

Mode Thrift

- Connexion au cluster
- Créer un keyspace
- Créer une famille de colonne
- Insérer une première ligne

43	72	20
inséré en premier	inséré en second	inséré en troisième

- Interroger les données
- Utiliser le tri sur les colonnes (slice)

Exercices

- **Créer une famille de colonnes d'évènement**
  - Créer la famille log_experiment avec un comparateur de colonnes ASCII
  - Paginer la consultation des colonnes
- **Créer une famille de colonne hygrometrie avec un comparator composé**
  - Les colonnes seront au format
    - %Y%m%d,"temperature"
    - %Y%m%d,"humidite"

www.cnrs.fr

Travaux Dirigés

CQL

CQL

- Interface ressemblant à du langage relationnel sur un modèle NON relationnel
- Clause WHERE uniquement sur toutes les colonnes indexées
- Pas de jointure
- Pas de GROUP BY
- Tri préétabli

www.cnrs.fr

Travaux Dirigés

CQL – premiers pas

Exercices

- **Connexion au cluster**
- **Créer un keyspace**

```
CREATE  KEYSPACE IF NOT EXISTS keyspace_name
WITH REPLICATION = Class TopologyStrategy
```

  - Class TopologyStrategy
  - Réplication sur plusieurs datacenters

```
{'class' : 'NetworkTopologyStrategy', 'dc1' : 3, 'dc2' : 2};
```

  - Réplication sur un seul datacenter

```
{'class' : 'SimpleStrategy', 'replication_factor' : 3 };
```

www.cnrs.fr

Travaux Dirigés

Tester le partitionnement

Exercices

- **Créer une famille de colonne "videos" pour enregistrer les informations sur les vidéos**

Column Name	Data Type
video_id	timeuuid
added_date	timestamp
description	text
title	text
user_id	uuid

- **Charger la table**
- **Vérifier la répartition des données**

www.cnrs.fr

Travaux Dirigés

Indexer

Exercices

- **Créer la famille de colonne "videos_by_title_year" pour requêter sur les champs title et year**

Column Name	Data Type
title	text
added_year	int
added_date	timestamp
description	text
user_id	uuid
video_id	uuid

```
CREATE TABLE videos_by_title_year (
…
PRIMARY KEY ((title, added_year))
```

Exercices

- **Créer la famille de colonne videos_by_tag_year avec une clé primaire sur tag et des index secondaires sur added_year (avec un tri décroissant) et video_id (avec un tri**

Column Name	Data Type
tag	text
added_year	int
video_id	uuid
added_date	timestamp
description	text
title	text
user_id	uuid

www.cnrs.fr

Travaux Dirigés

Wide rows (lignes sauvages) en CQL aussi ?

Exercices

- Consulter la famille de colonnes d'hygrométrie en CQL

- En CQL, « wide rows » = plusieurs lignes par clé primaire !

Exercices

- Fondamentaux
- Tester le partitionnement
- Indexer
- Lignes sauvages (journaux d'évènements)
- Pagination sur les colonnes
- Colonnes composites

www.cnrs.fr

Conclusion

Conclusion

- **Dénormalisez !**
  1 besoin = requête = 1 famille de colonnes

- **Adaptez les données aux requêtes**
  Pas l'inverse !