

# Not Only SQL : Vers des SGBD large échelle

Jonathan Lejeune

Sorbonne Université/LIP6-INRIA

DataCloud - Master 2 SAR 2020/2021

# Les SGBD relationnels

## Les plus populaires



## Principaux avantages

- Données structurées, modélisation intuitive des données
- Normalisation des données (1NF, 2NF, 3NF, ...)
- Indexation des données, Optimisation d'accès
- Un langage d'interrogation : le SQL
- Simple à administrer/à déployer car centralisé sur un serveur
- Propriétés ACID :
  - **Atomicité** : une transaction se fait au complet ou pas du tout
  - **Cohérence** : l'état du système reste valide à chaque transaction
  - **Isolation** : Aucune dépendance possible entre les transactions
  - **Durabilité** : une transaction confirmée demeure enregistrée

# Les limites des SGBD relationnels

Hypothèse : nb lectures  $\gg$  nb écritures

Nb Users	Problème	Solution possible
1000	tout va bien	
10000	serveur central du SGBD de + en + chargé en I/O et CPU	ajouter machines esclaves + load balancer : lectures pour les esclaves, écritures pour le maître
100000	Les esclaves sont de + en + chargés	ajouter un service de cache (Memcached, Redis, ...) perte des garanties de cohérence entre cache et database
>100000	les écritures font goulot d'étranglement	remplacer le master par une machine plus puissante (=vertical scaling) augmentation des coûts d'exploitation

# Les limites des SGBD relationnels

## Ils ne passent pas à l'échelle

- Espace de stockage limité et centralisé
- Inadapté à la vélocité des données
- Les traitements classiques deviennent coûteux :
  - CRUD : Create, Read, Update, Delete sont de + en + lent
  - Respect des contraintes relationnelles de + en + difficile
  - Les index sont de + en + difficiles à maintenir
- Les schéma de données sont peu flexibles

Ajout d'une colonne ⇒ modification de tous les tuples de la table

## Objectifs

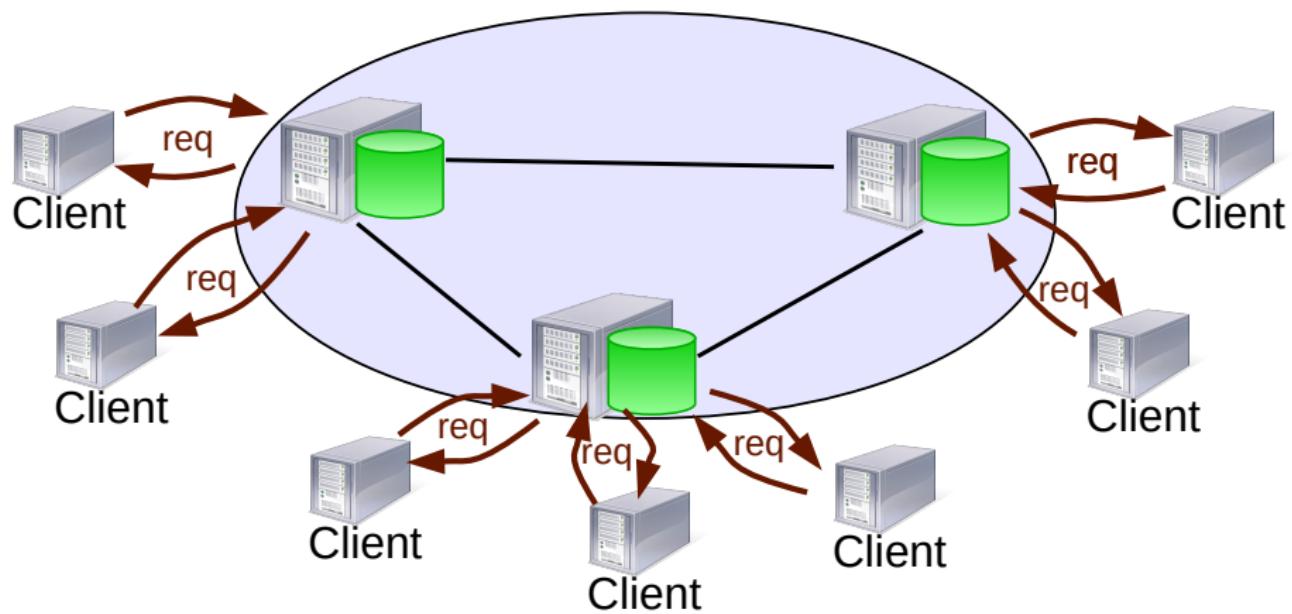
Élaborer des SGBD large échelle adaptés aux données massives :

- 1) en se basant sur des architectures distribuées
- 2) en révisant les modèles de données relationnels classique et leur mécanisme d'interrogation

# Apports de la géo-réPLICATION des données (1/3)

## Équilibrage de charge

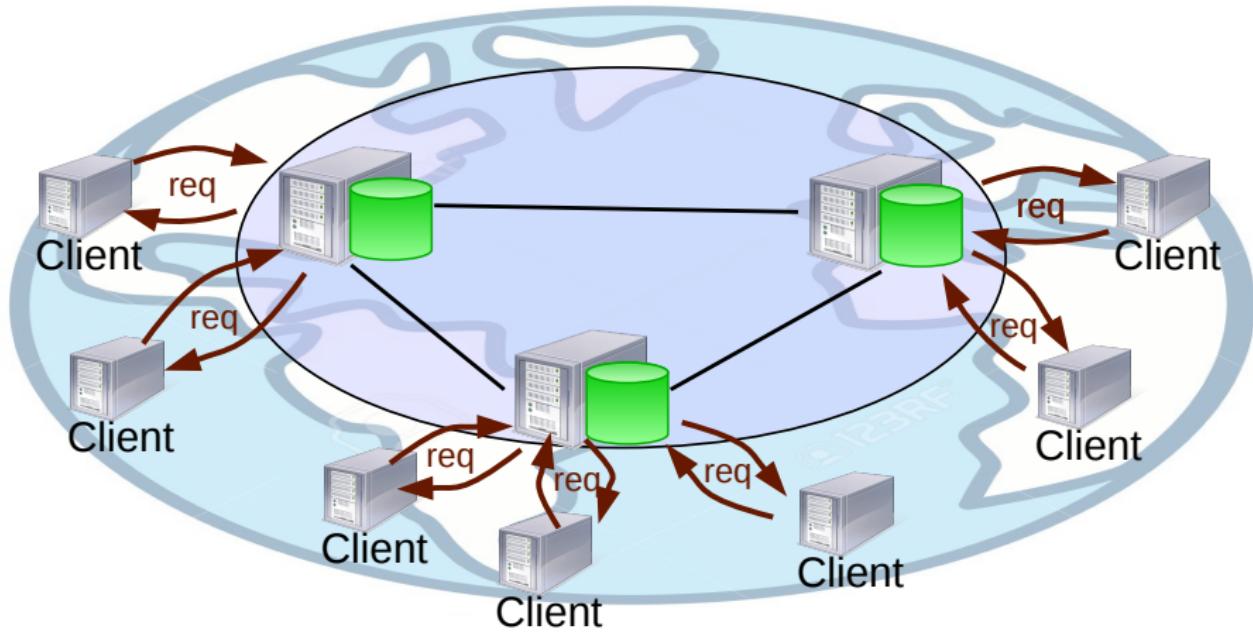
Les requêtes clientes sont réparties sur l'ensemble des réplicas



# Apports de la géo-réPLICATION des données (2/3)

## Localité géographique

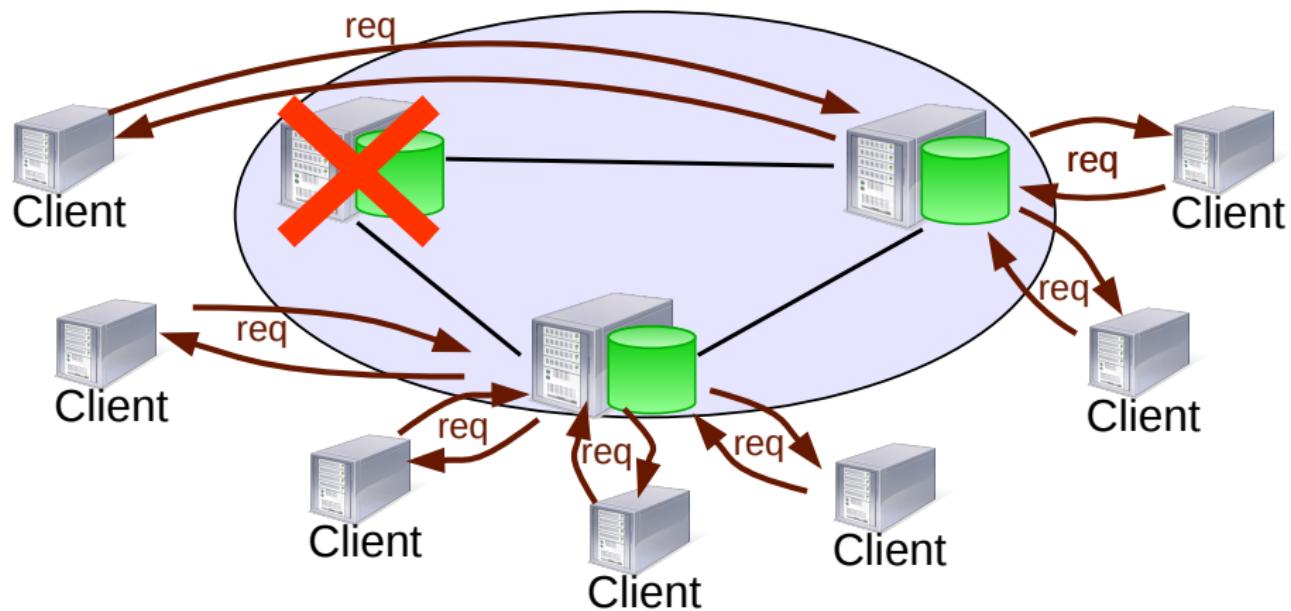
Les clients peuvent s'adresser au réplica le plus proche



# Apports de la géo-réPLICATION des données (3/3)

## Robustesse

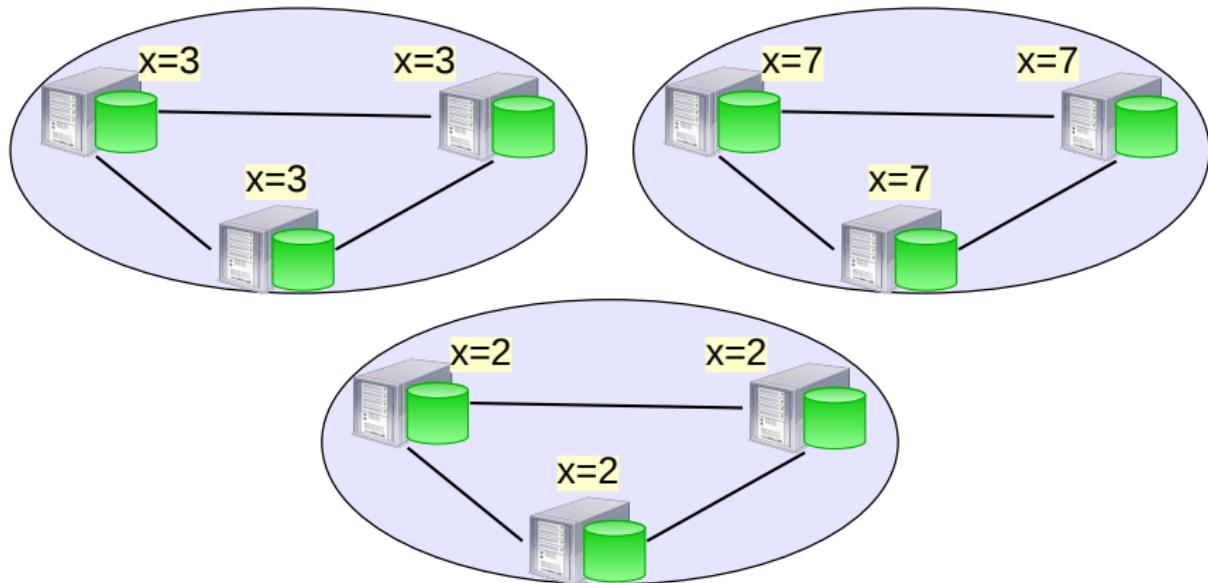
Les données sont toujours accessibles même en cas de panne d'un réplica.



# Problématiques de la géo-réPLICATION des données (1/3)

## Strong Consistency (C)

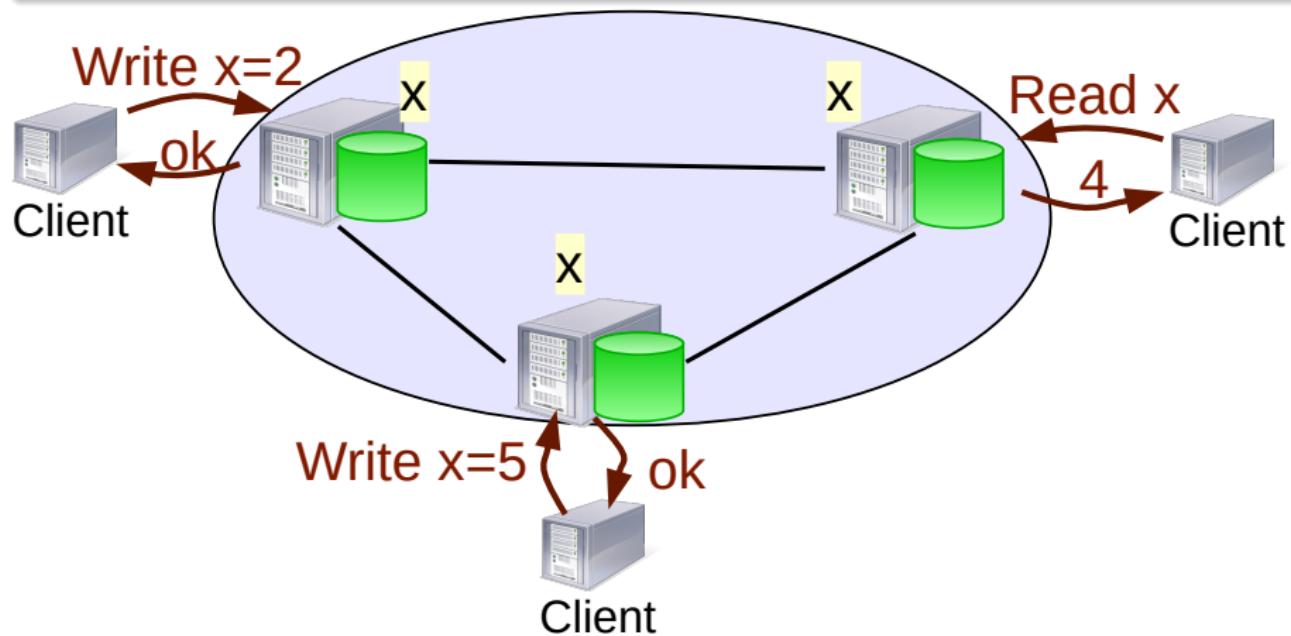
Comment respecter une cohérence forte entre les répliques ?



# Problématiques de la géo-réPLICATION des données (2/3)

## Read and Write Availability (A)

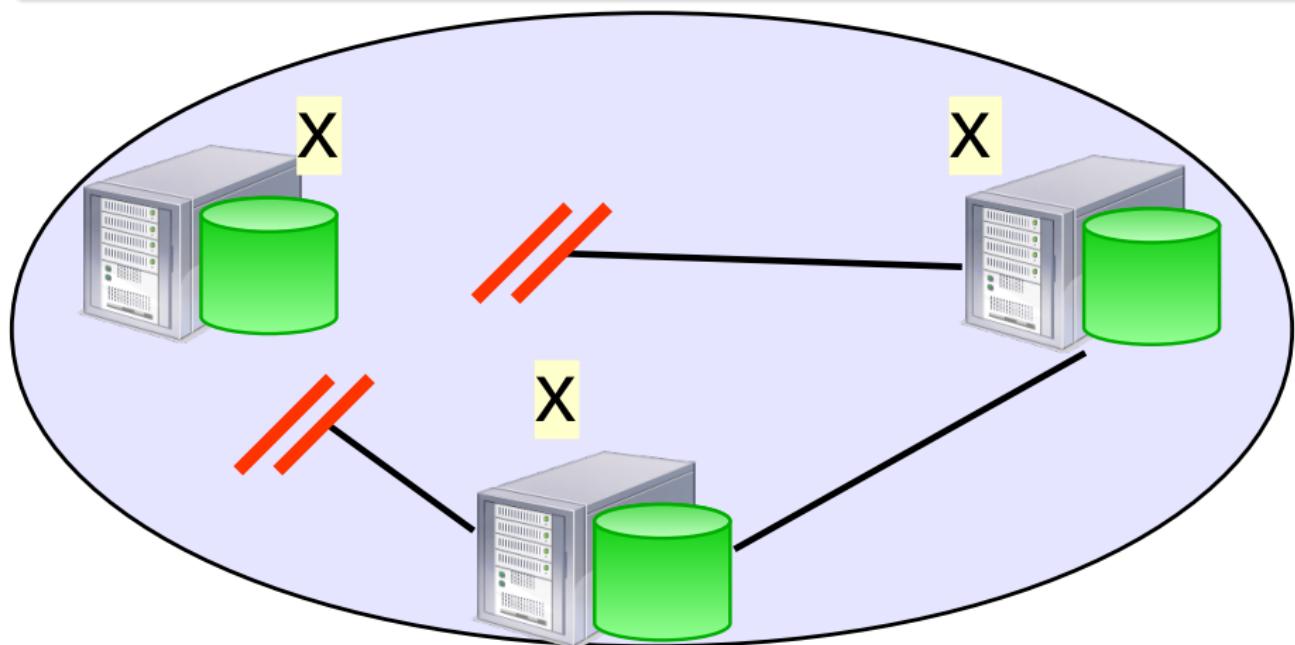
Comment ne pas dégrader la latence des opérations de lecture et d'écriture ?



# Problématiques de la géo-réPLICATION des données (3/3)

## Partition tolerance

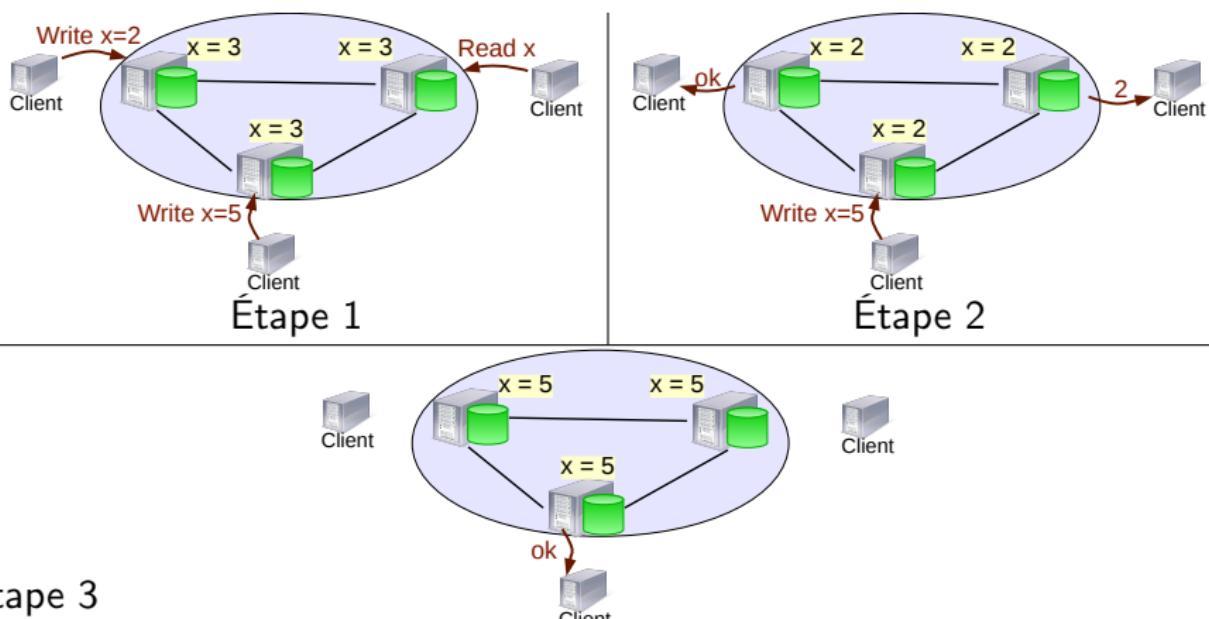
Que faire en cas de partition du système dues aux pannes/à la latence du réseau ?



# Avoir C et A

## Besoin de Linéarisabilité

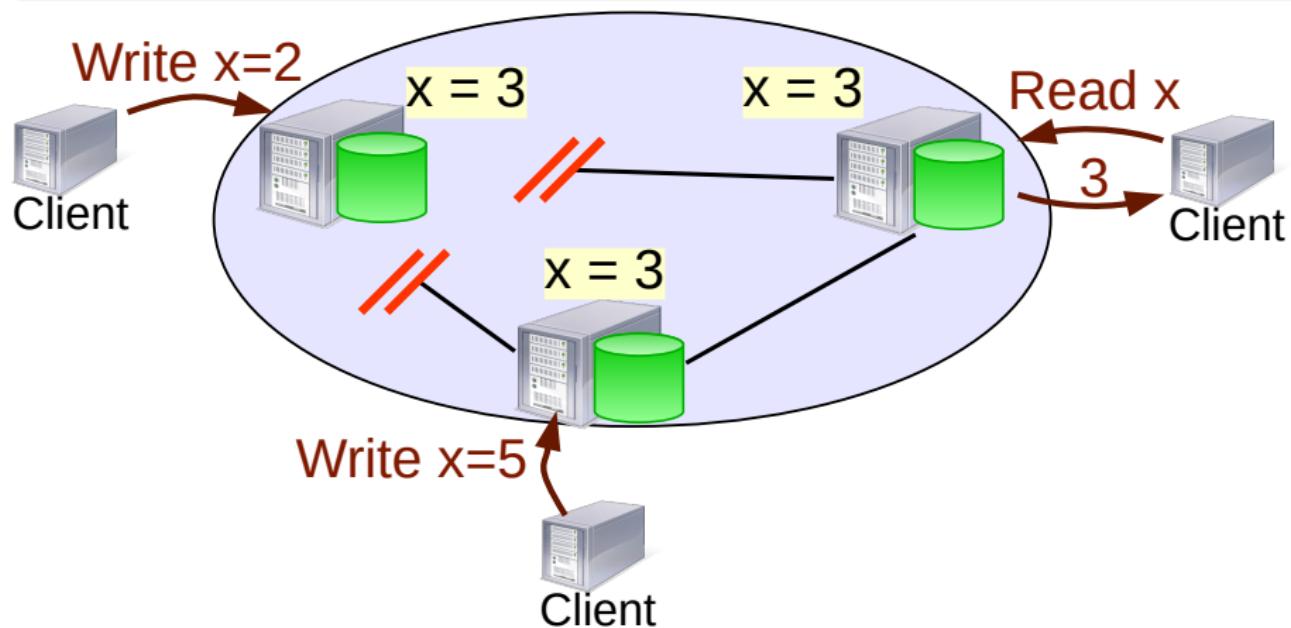
Toutes les écritures doivent être vues dans le même ordre sur chaque réplica  
⇒ Algorithme de consensus



# Ajout de P

## Choix de Consistency

Obligation d'attendre le rétablissement du réseau pour les écritures  
⇒ violation de A

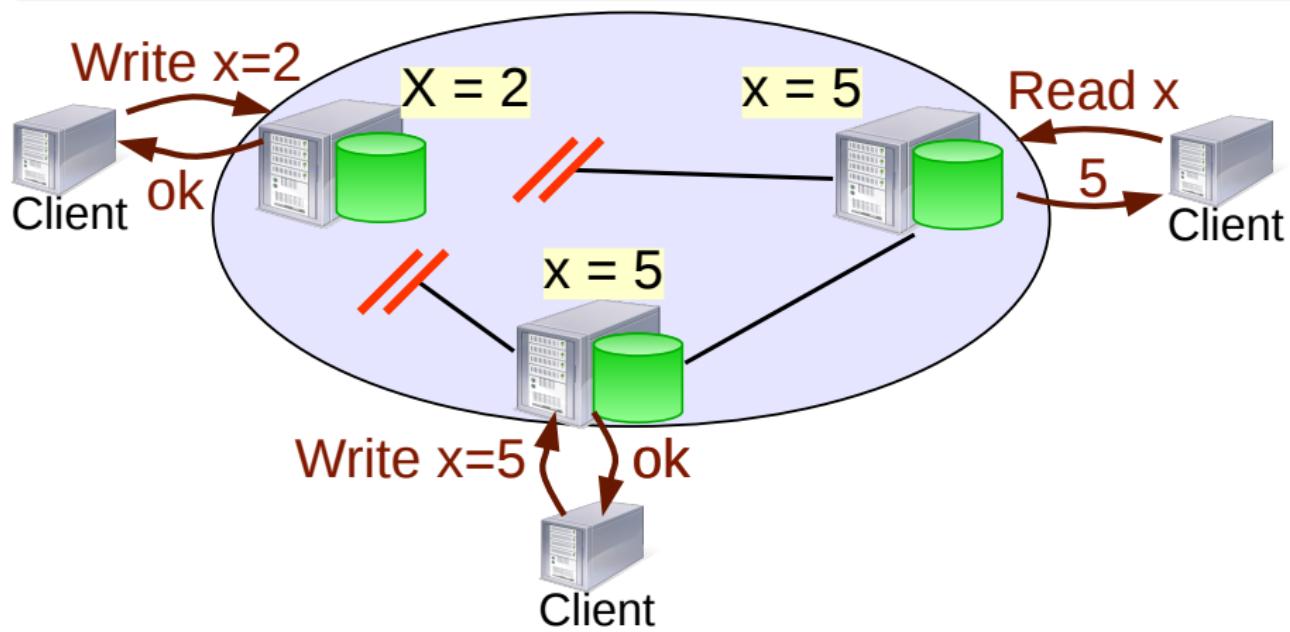


# Ajout de P

## Choix de Availability

Validation locale des transactions

⇒ violation de C



# Le théorème d'impossibilité CAP

BREWER, Eric A. *Towards robust distributed systems*. In : PODC. 2000.

Dans un système de stockage distribué et géo-répliqué, il n'est possible que d'avoir deux de ces trois garanties au même instant :

- **C** : Cohérence forte des réplicas
- **A** : Disponibilité en lecture et écriture des données
- **P** : Partitionnement du réseau

## Conséquences

- P étant inévitable, il faut choisir le moment venu si on souhaite garder C ou A
- Il devient trop coûteux de respecter les propriétés ACID

# Des propriétés ACID vers les propriétés BASE



## Basically Available

Quelle que soit la charge du SGBD (données/requêtes), on garantie un taux de disponibilité minimal de la donnée.

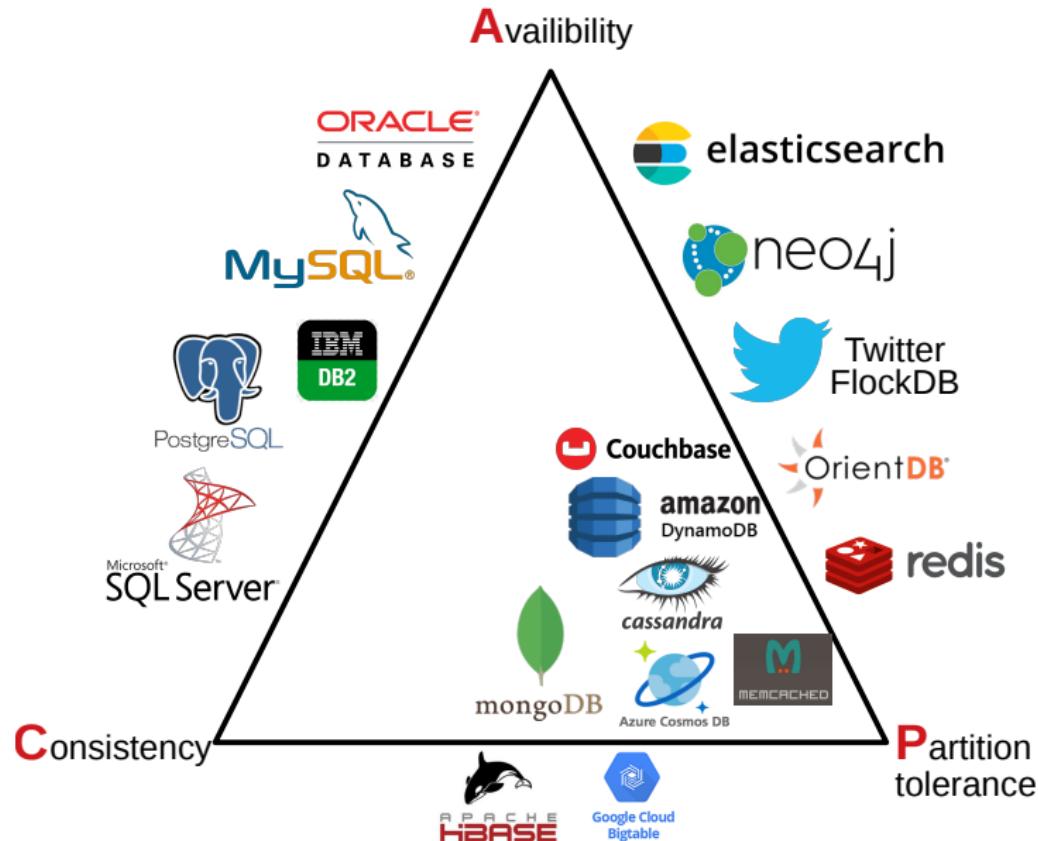
## Soft-state

La base NoSQL n'a pas à être fortement cohérente à tout instant

## Eventually consistent

À terme, la base atteindra un état cohérent

# Théorème CAP et SGBD



# Les SGBD NoSQL

## Caractéristiques communes

- Les APIs d'interrogation sont simplifiées (pas de SQL)
- Les données et les calculs sont distribués et géo-répliqués
- Le schéma des données est très flexibles
- Pas de relation complexe entre les données (ex : pas de clé étrangère)
- Adapté aux propriétés BASE

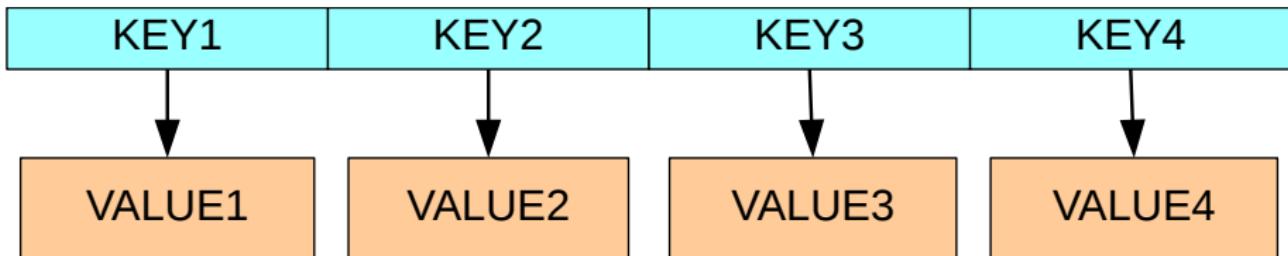
## Les familles de SGBD NoSQL

- Clé-valeur
- Orienté colonne
- Orienté document
- À base de graphe

# Les SGBD NoSQL clé-valeur

## Caractéristiques

- Similaire à une énorme table de hachage distribuée
- Une clé identifie une valeur (= une donnée) de manière unique
- Les données ne sont pas typées  
    ⇒ aucun schéma ni structure n'est imposé
- Seules les opérations CRUD sont autorisées :  
`Create(key, val), Read(key) , Update(key, val), Delete(key)`



# Les SGBD NoSQL clé-valeur

## Exemples de SGBD NoSQL clé-valeur et leurs utilisateurs

-  **Redis** : Twitter, Github, StackOverflow, Trip Advisor, Nokia, Samsung
-  **Memcached** : Linkedin, Instagram, DropBox, Facebook, Wikipédia
-  **Azure Cosmos DB** : MSN, LG, Schneider Electric

## Exemples d'application

- IoT
- e-commerce
- gestion de cache
- transactions rapides
- fichiers de logs
- chat

# Les SGBD NoSQL orienté colonne

## Caractéristiques

- Le stockage des tables se fait par colonne
- Les colonnes sont stockées en chunk sur une architecture distribuée
  - ⇒ adapté au traitement sur les colonnes comme les agrégats
  - ⇒ adapté aux gros traitements analytiques

Stockage traditionnel orienté ligne

PRIMARY KEY	ATTRIBUTE 1	ATTRIBUTE 2	ATTRIBUTE 3
Key 1	data 1.1	data 2.1	data 3.1
Key 2	data 1.2	data 2.2	data 3.2
Key 3	data 1.3	data 2.3	data 3.3
Key 4	data 1.4	data 2.4	data 3.4

Stockage orienté colonne

PK	ATT 1	PK	ATT 2	PK	ATT 3
Key 1	data 1.1	Key 1	data 2.1	Key 1	data 3.1
Key 2	data 1.2	Key 2	data 2.2	Key 2	data 3.2
Key 3	data 1.3	Key 3	data 2.3	Key 3	data 3.3
Key 4	data 1.4	Key 4	data 2.4	Key 4	data 3.4

# Les SGBD NoSQL orienté colonne

## Exemples de SGBD NoSQL orienté colonne et leurs utilisateurs



### Google BigTable



Hbase : Hortonworks, Facebook, Adobe, Twitter, Yahoo !



Elasticsearch : Netflix, Stack Overflow, LinkedIn, OpenStack Cloud

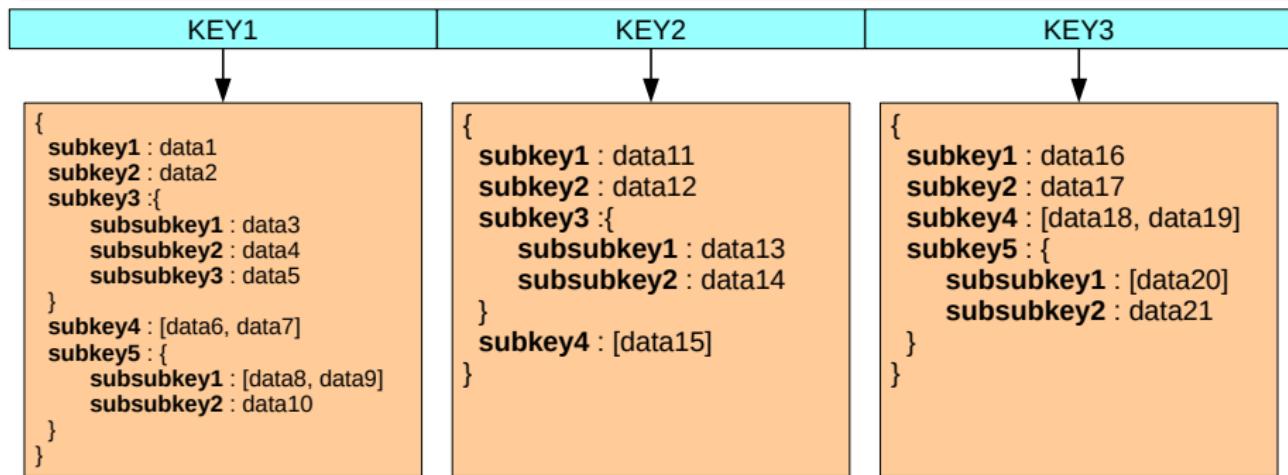
## Exemples d'application

- Comptage (vote en ligne, compteur, etc)
- Journalisation
- Recherche de produits dans une catégorie
- Reporting à large échelle.

# Les SGBD NoSQL orienté document

## Caractéristiques

- Stockage et manipulation de documents
- Chaque document est identifié par une clé globale
- Les informations du documents sont structurés en arbre :  
    ⇒ possibilité d'avoir un langage d'interrogation riche tout en passant à l'échelle



# Les SGBD NoSQL orienté document

## Exemples de SGBD NoSQL orienté document et leurs utilisateurs



**MongoDB** : Facebook, ebay, Cisco, Adobe, Bouygues Telecom, Nokia



**CouchBase** : General Electric, Coyote, Ryanair, LinkedIn, Cisco



**DynamoDB** : Shazam, NetFlix, Washington Post



**Cassandra ?** : Instagram, NetFlix, GitHub, CERN

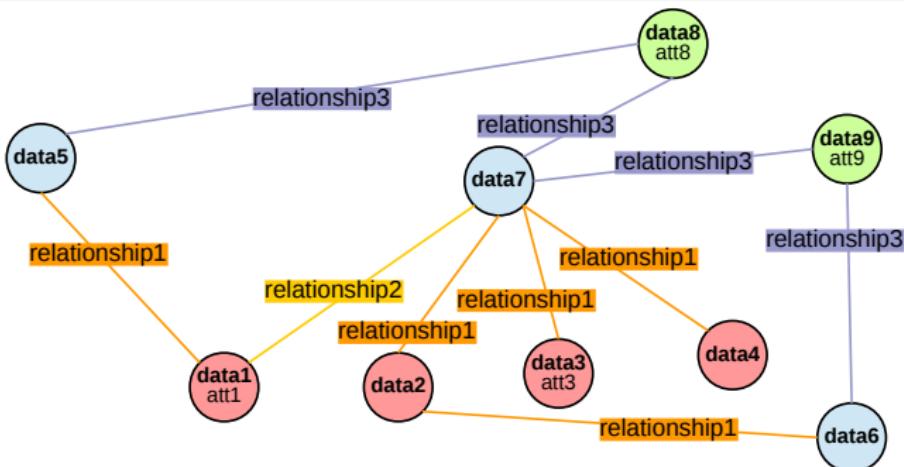
## Exemples d'application

- gestion de contenu (bibliothèques numériques, collections de produits, dépôts de logiciels, collections multimédia, etc.)
- framework stockant des objets
- collection d'événements complexes
- gestion des historiques d'utilisateurs sur réseaux sociaux.

# Les SGBD NoSQL à base de graphe

## Caractéristiques

- Les données sont stockées sous la forme de nœuds et de liens
- Définition de propriété sur les nœuds et les liens
- Les requêtes exprimables sont basées sur la gestion de chemin, de propagation, d'agrégation voire de recommandation :  
⇒ possibilité d'exprimer des corrélations entre les éléments



# Les SGBD NoSQL à base de graphe

## Exemples de SGBD NoSQL à base de graphe et leurs utilisateurs



**Neo4j** : Armée US, LinkedIn, eBay



**OrientDB** : Nations Unies, Dell, Warner Music Group, Cisco



**FlockDB** : Twitter

## Exemples d'application

- réseaux sociaux (recommandation, plus court chemin, cluster...)
- réseaux SIG (routes, réseau électrique, fret...)
- web social (Linked Data)

# De la normalisation à la dénormalisation

## Normalisation (SGBD relationnel)

Avoir un seul exemplaire de la donnée dans le schéma relationnel

- ⇒ Évite la redondance
- ⇒ La modification d'une donnée ne concerne qu'une seule table
- ⇒ Gain d'espace de stockage
- ⇒ Les requêtes à jointures ne passent pas à l'échelle

## Dénormalisation (SGBD NoSQL)

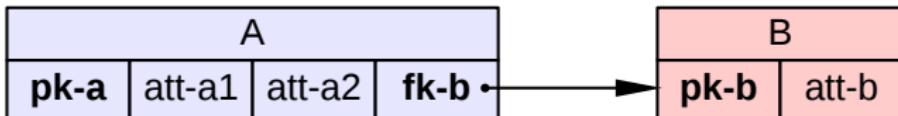
Copie des mêmes données dans plusieurs documents ou tables

- ⇒ Simplification et optimisation du traitement des requêtes en lecture
- ⇒ Les jointures peuvent être évitées
- ⇒ Il faut garder une cohérence des copies dans le schéma lors des écritures

## Comment dénormaliser ?

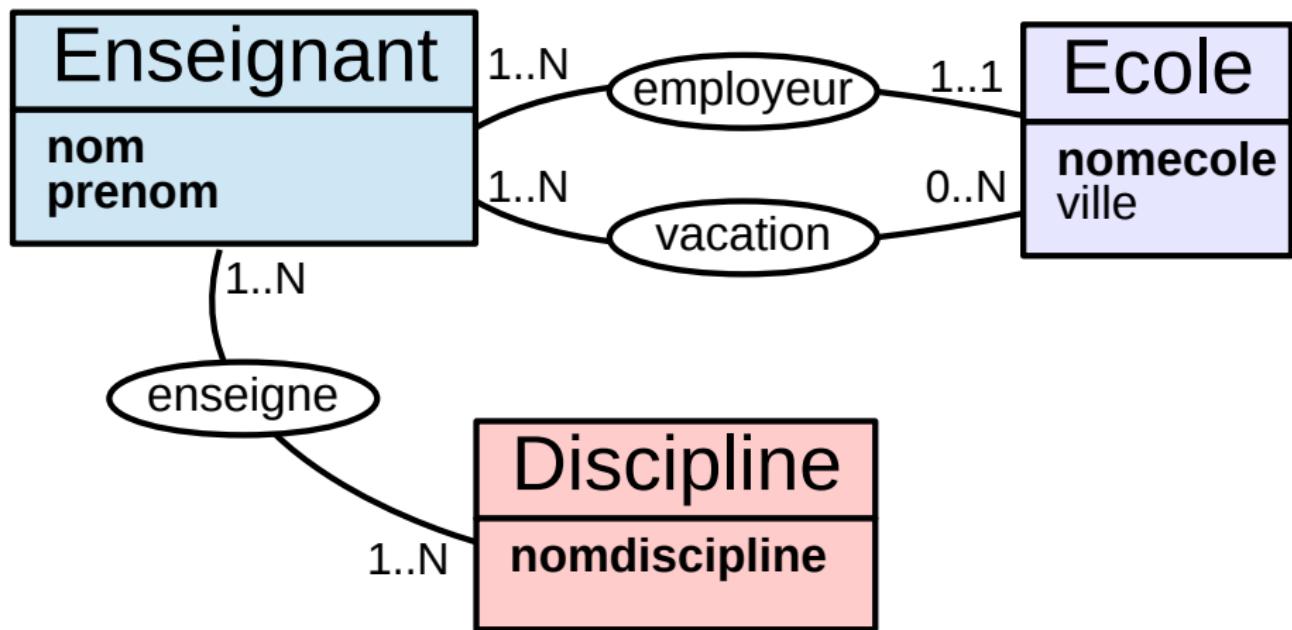
- 1) Normaliser le modèle de données à un niveau satisfaisant
- 2) Analyser les requêtes les plus fréquentes des applications clientes
- 3) Il est possible de dénormaliser lorsque :
  - Des requêtes critiques reposent sur des données provenant de plusieurs tables
    - ⇒ Fusion de tables
    - ⇒ Ajout de colonnes redondantes
  - La création de colonnes d'agrégat nécessite de nombreux calculs
    - ⇒ Ajout des colonnes d'agrégat dans la table

# Exemples de dénormalisation d'une relation 1-N



Fusion de tables	<table border="1"><thead><tr><th colspan="5">A</th></tr><tr><th>pk-a</th><th>att-a1</th><th>att-a2</th><th>fk-b</th><th>att-b</th></tr></thead></table>	A					pk-a	att-a1	att-a2	fk-b	att-b								
A																			
pk-a	att-a1	att-a2	fk-b	att-b															
Ajout de colonnes redondantes	<table border="1"><thead><tr><th colspan="5">A</th></tr><tr><th>pk-a</th><th>att-a1</th><th>att-a2</th><th>fk-b</th><th>att-b</th></tr></thead><tbody><tr><td></td><td><table border="1"><thead><tr><th colspan="2">B</th></tr><tr><th>pk-b</th><th>att-b</th></tr></thead></table></td></tr><tr></tr></tbody></table>	A					pk-a	att-a1	att-a2	fk-b	att-b		<table border="1"><thead><tr><th colspan="2">B</th></tr><tr><th>pk-b</th><th>att-b</th></tr></thead></table>	B		pk-b	att-b		
A																			
pk-a	att-a1	att-a2	fk-b	att-b															
	<table border="1"><thead><tr><th colspan="2">B</th></tr><tr><th>pk-b</th><th>att-b</th></tr></thead></table>	B		pk-b	att-b														
B																			
pk-b	att-b																		
Ajout de colonnes d'agrégat	<table border="1"><thead><tr><th colspan="5">A</th></tr><tr><th>pk-a</th><th>att-a1</th><th>att-a2</th><th>fk-b</th><th>•</th></tr></thead><tbody><tr><td></td><td><table border="1"><thead><tr><th colspan="3">B</th></tr><tr><th>pk-b</th><th>att-b</th><th>moy(att-a2)</th></tr></thead></table></td></tr><tr></tr></tbody></table>	A					pk-a	att-a1	att-a2	fk-b	•		<table border="1"><thead><tr><th colspan="3">B</th></tr><tr><th>pk-b</th><th>att-b</th><th>moy(att-a2)</th></tr></thead></table>	B			pk-b	att-b	moy(att-a2)
A																			
pk-a	att-a1	att-a2	fk-b	•															
	<table border="1"><thead><tr><th colspan="3">B</th></tr><tr><th>pk-b</th><th>att-b</th><th>moy(att-a2)</th></tr></thead></table>	B			pk-b	att-b	moy(att-a2)												
B																			
pk-b	att-b	moy(att-a2)																	

## Exemple relationnel



# Adaptation de l'exemple à un SGBD clé-valeur

Nicolas Durand	Régis Dupont	Céline Martin
<p><b>Employeur</b> : CNAM <b>Vacation</b> : Paris5, Supelec <b>Disciplines</b> : BDD, NoSQL</p>	<p><b>Employeur</b> : Paris5 <b>Vacation</b> : Supelec <b>Disciplines</b> : Machine learning dev</p>	<p><b>Employeur</b> : Supelec <b>Vacation</b> : Paris5, CNAM <b>Disciplines</b> : Ontologie, logique datavise</p>

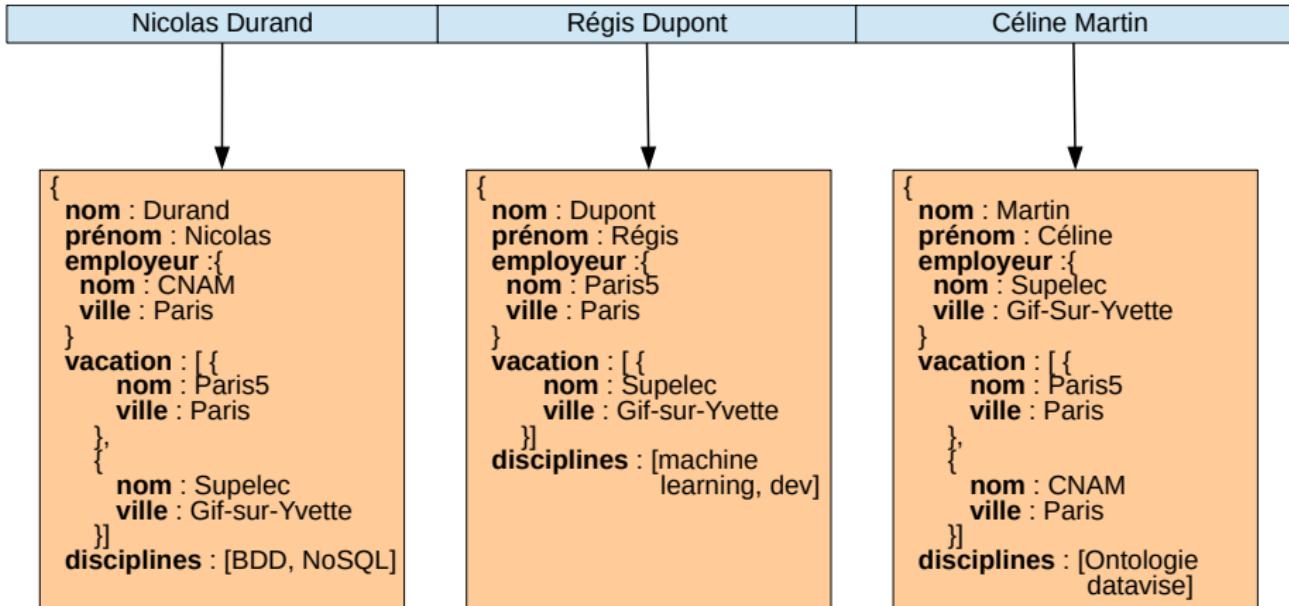
# Adaptation de l'exemple à un SGBD orienté colonne

enseignant	Employeur
Nicolas Durand	CNAM
Régis Dupont	Paris5
Céline Martin	Supelec

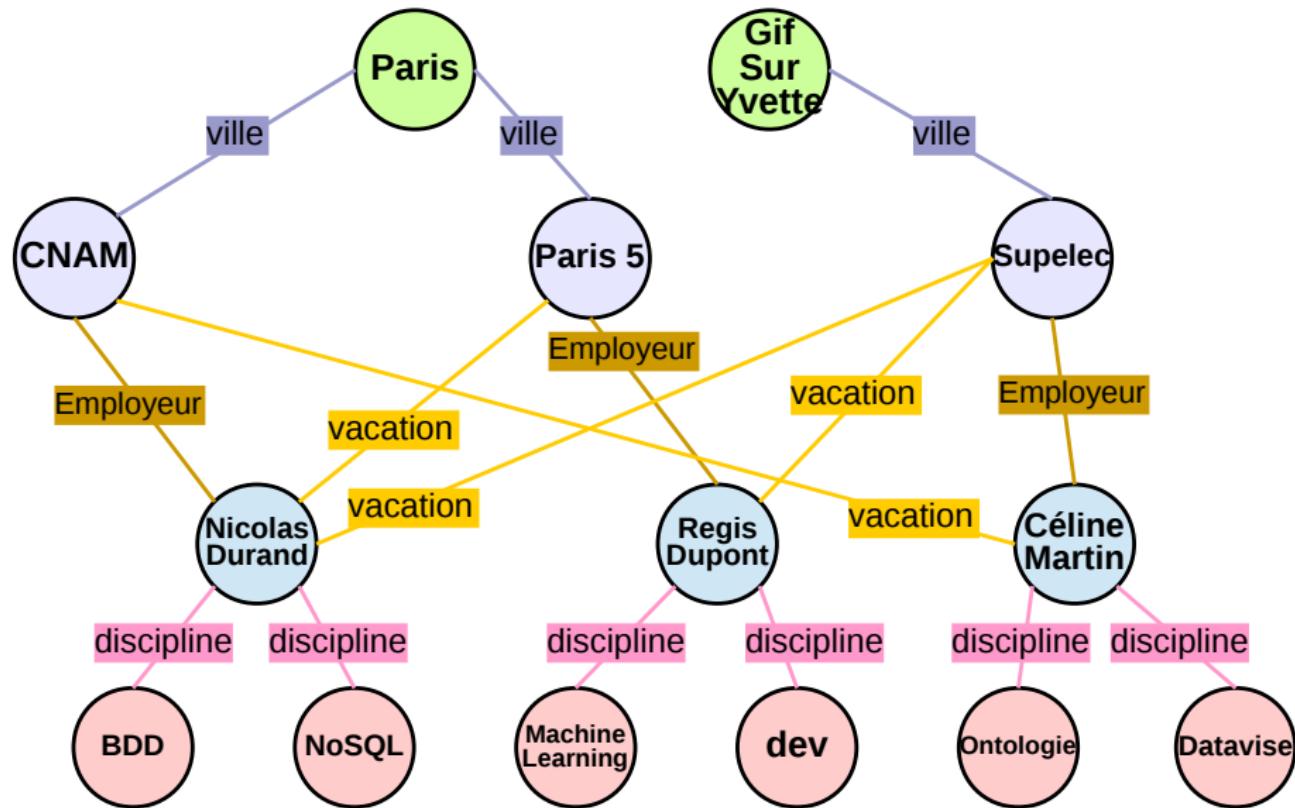
enseignant	Vacation
Nicolas Durand	Paris5 , Supelec
Régis Dupont	Supelec
Céline Martin	Paris5, CNAM

enseignant	Disciplines
Nicolas Durand	BDD, NoSQL
Régis Dupont	Machine Learning dev
Céline Martin	Ontologie Logique datavise

# Adaptation de l'exemple à un SGBD orienté document



# Adaptation de l'exemple un SGBD à base de graphe



# Le sharding

## Définition du sharding

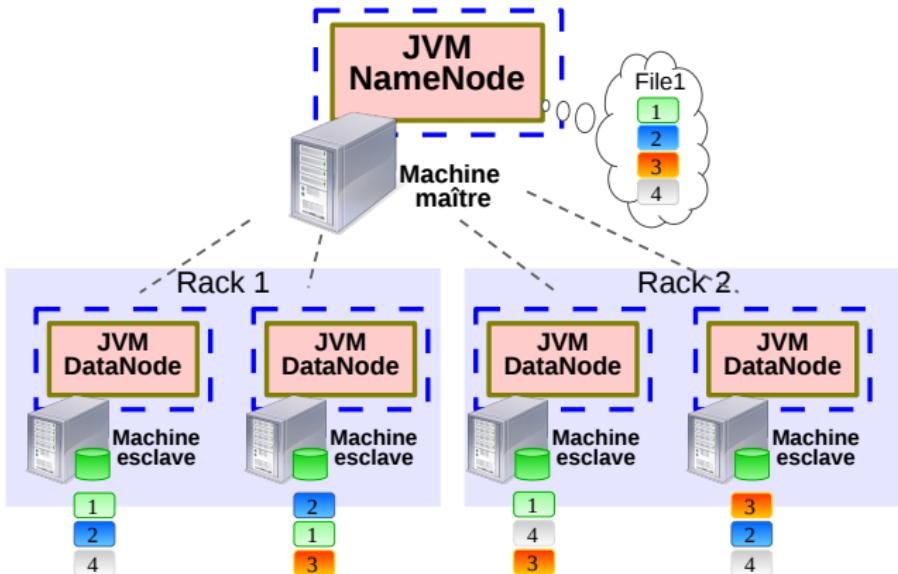
Technique de distribution des chunks (morceaux de fichiers) sur un ensemble de serveurs, avec la capacité de gérer :

- l'élasticité des serveurs et des données
- la tolérance aux pannes

## Quelques techniques

- **Hadoop Distributed File System**
- B-arbre distribué
- Table de hachage distribuée (DHT)

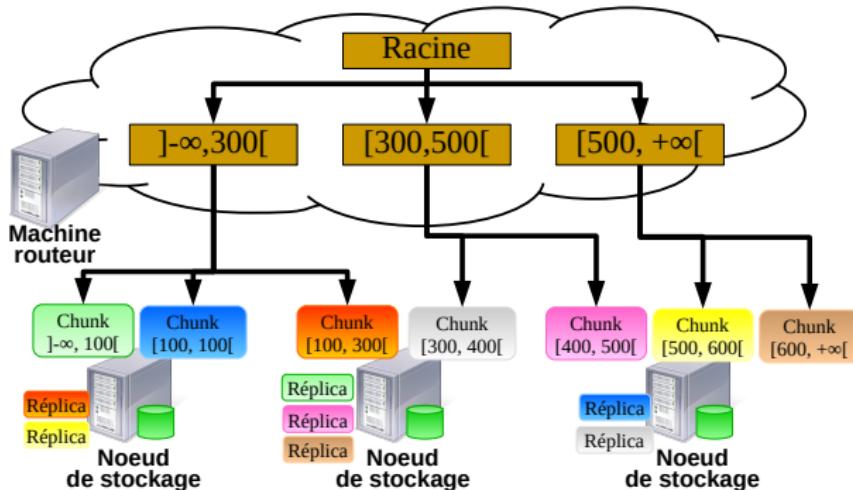
# Le sharding avec HDFS



## Caractéristiques

- Répartition et réPLICATION gérée par le namenode
- Prise en compte de la localité physique des données

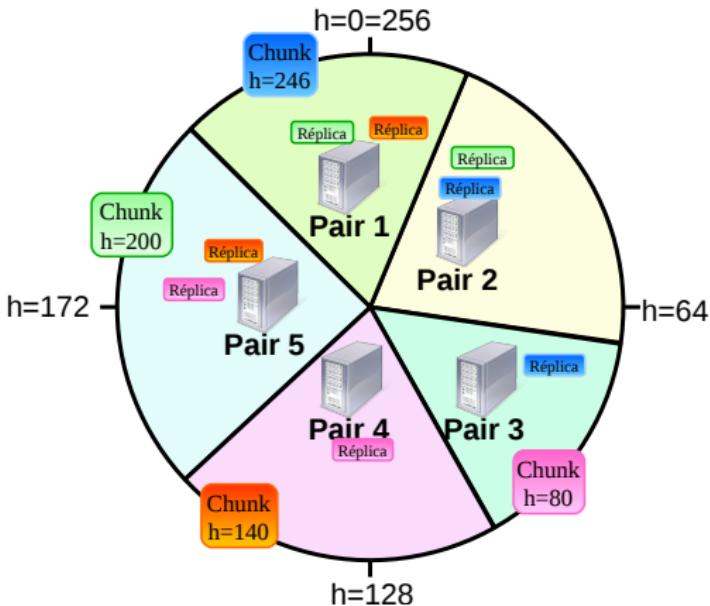
# Le sharding avec B-arbre distribué



## Caractéristiques

- La répartition se fait en fonction d'un B-arbre dont le cœur est stocké sur une machine routeur  
    ⇒ les données sont indexées et triées
- La réPLICATION d'un chunk est gérée par son nœud de stockage

# Le sharding avec une table de hachage distribuée (DHT)



## Caractéristiques

- Répartition en fonction d'une fonction de hachage
- Un pair est responsable d'une plage dans l'espace de hachage
- Les répliques sont hébergés par les pairs suivants dans l'espace de hachage

# Résumé : NoSQL vs. SGBD relationnel

	NoSQL	SGBD relationnel
<b>Structure des données</b>	sans schéma, colonnes non fixées	structure fixée sur l'ensemble des tables
<b>Taille</b>	très grandes tables (milliard de lignes)	taille moyenne (million de lignes)
<b>Traitement</b>	analytique et non transactionnel	transactionnel
<b>Stockage</b>	en ligne ou en colonne	généralement en ligne
<b>Passage à l'échelle</b>	linéaire et horizontal	vertical sur un serveur central
<b>Requetage</b>	simple (put/get)	élaboré (SQL, PL/SQL)
<b>valeur nulle</b>	non matérialisée	stockée

# Synthèse

## Caractéristique d'un SGBD NoSQL

- sa relation envers le théorème CAP :
  - AP : le SGBD privilégie la disponibilité à la cohérence
  - CP : le SGBD privilégie la cohérence à la disponibilité
- N.B. : Il existe des SGBD pouvant respecter soit l'un soit l'autre
- son modèle de données
  - clé-valeur
  - orienté colonne
  - orienté document
  - à base de graphe
- sa technique de sharding

## Comment choisir ?

Le choix d'un SGBD NoSQL dépend :

- des besoins métiers de l'application
- de l'infrastructure que l'on dispose