

Les bases de données No SQL

2021/2022

tasnim.abar@supcom.tn

tasnim.abar@tek-up.tn

Plan

- Rappel aux BDs SQL
- Pourquoi le No SQL
- Limites du SGBDR : théorème de CAP
- Introduction au NO SQL
- Les familles No SQL

Rappel au BD SQL

- Les bases de données **SQL** :
 - ✓ Sont principalement appelées bases de données relationnelles (SGBDR)
 - ✓ Sont basées sur des tables : les BDs **SQL** représentent des données sous la forme de tables composées de n nombre de lignes de données
 - ✓ ont un schéma prédéfini
 - ✓ sont évolutives verticalement
 - ✓ utilisent **SQL (structured query language)** pour définir et manipuler les données.
 - ✓ Postegre SQL, SQL server, Sqlite, MySQL ...
 - ✓ Dans un fichier , il faut tout faire ligne par ligne à la main .



Rappel au BD SQL

- Les commandes de manipulations de données:

- ✓ *Select*
- ✓ *Insert*
- ✓ *Update*
- ✓ *Delete*

- Les commandes de définitions de données:

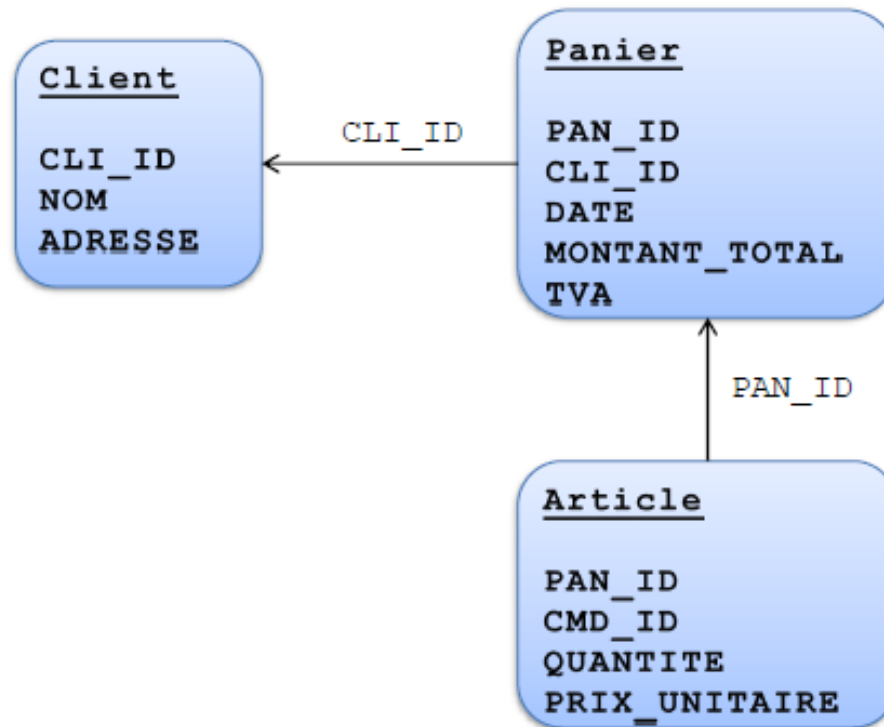
- ✓ *Create*
- ✓ *Rename*
- ✓ *Drop*
- ✓ *Alter*

The screenshot shows a PostgreSQL SQL Editor window. The top toolbar includes icons for file operations, execution, and help. The connection string is 'nyc on postgres@localhost:54321'. The SQL Editor tab is active, displaying the query: `SELECT * FROM geometry_columns;`. The Output pane at the bottom shows the 'Data Output' tab with a table of results.

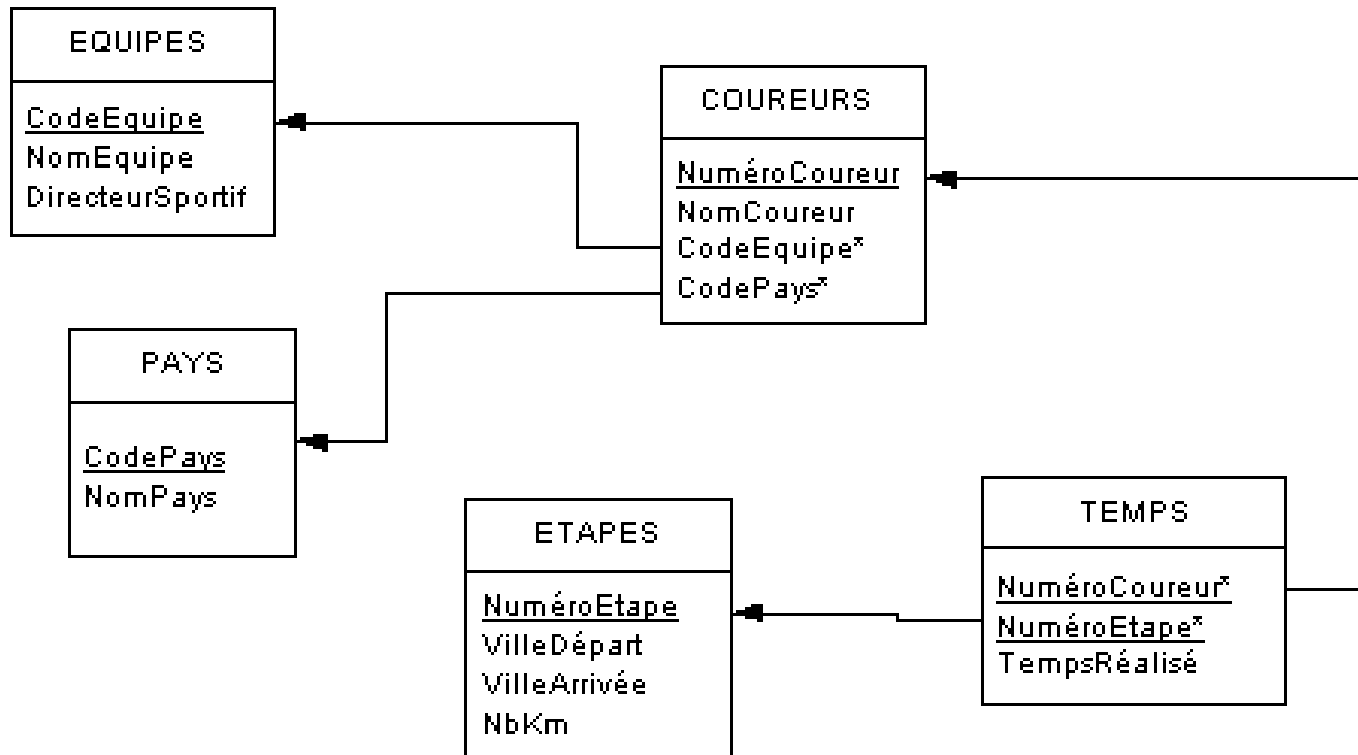
	f_table_catal	f_table_sche	f_table_name	f_geometry_	coord_dimen	srid	type
	character var	character var	character varying(256)	character var	integer	integer	character varying(100)
1		public	nyc_census_blocks	the_geom	2	26918	MULTIPOLYGON
2		public	nyc_neighborhoods	the_geom	2	26918	MULTIPOLYGON
3		public	nyc_streets	the_geom	2	26918	MULTILINESTRING
4		public	nyc_subway_stations	the_geom	2	26918	POINT
5		public	geometries	geom	2	-1	POINT

The status bar at the bottom indicates 'OK', 'Unix', 'Ln 1 Col 32 Ch 32', '5 rows.', and '26 ms'.

Exemple : modèle relationnel



Exemple : modèle relationnel



Rappel au BD SQL

Avantages

- Contexte mathématique (algebre relationnel)
- Langage de requête standard SQL
- Eviter la duplication des donnés

Inconvénients

- les requêtes sont parfois très complexes (jointure)
- Utiliser des jointures pour agréger des données liées (Les jointures sont couteuses)
- Difficile pour la Mise à l'échelle horizontale (horizontal scaling)
- Un peut lent

=> SQL est efficace pour un mode de gestion courant de donnés mais ne marche pas avec des BD de grandes quantités (big data)

Pourquoi le NoSQL

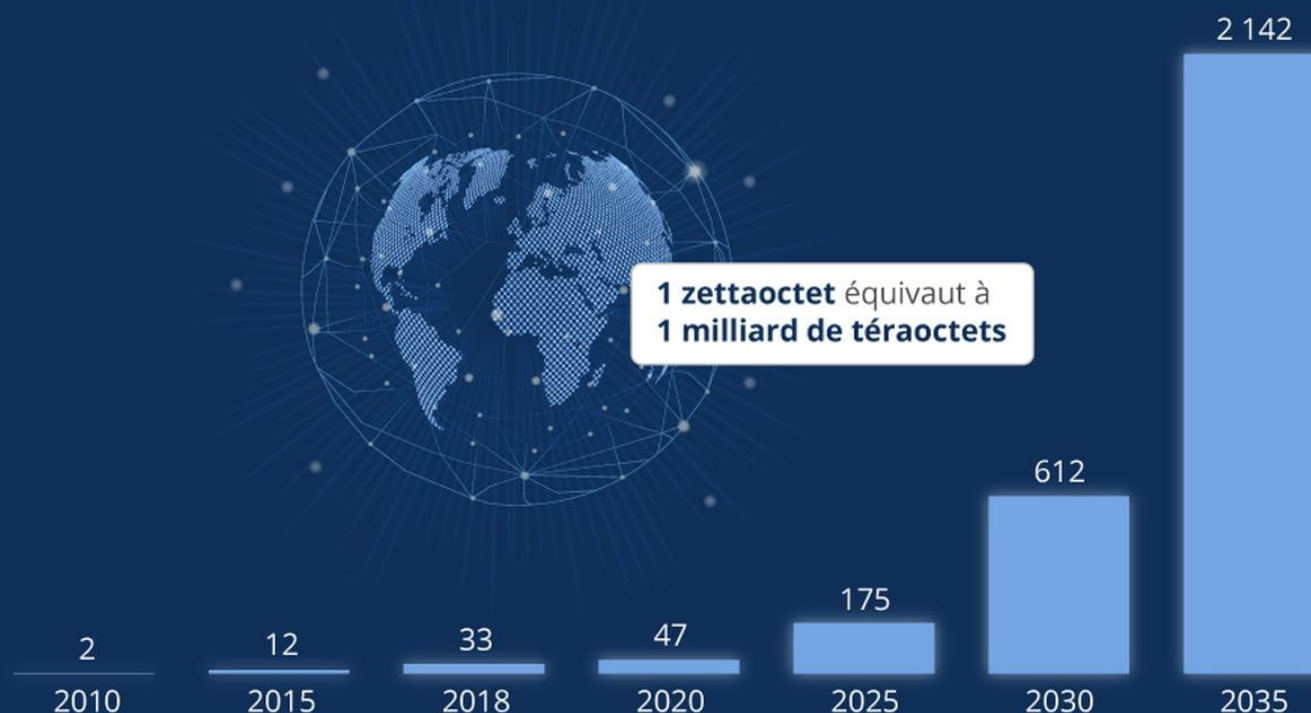
Big User




Pourquoi le NoSQL

Big data : le volume de données créées va exploser

Volume de données numériques créées dans le monde depuis 2010 (en zettaoctets) *



Pourquoi le NoSQL

- Essor des très grandes plateformes et applications Web (Google, Facebook, Twitter, LinkedIn, Amazon, ...)
 - Volume considérable de données à gérer par ces applications nécessitant une distribution des données et leur traitement sur de nombreux serveurs : « Data Centers »
 - Ces données sont souvent associées à des objets complexes et hétérogènes
=> Limites des SGBD traditionnels (relationnels et transactionnels) basés sur SQL
- 
- D'où nouvelles approches de stockage et de gestion des données :
 - permettant une meilleure scalabilité dans des contextes fortement distribués
 - permettant une gestion d'objets complexes et hétérogènes sans avoir à déclarer au préalable l'ensemble des champs représentant un objet
 - regroupées derrière le terme NoSQL (proposé par Carl Strozzi), ne se substituant pas aux SGBD Relationnels mais les complétant en comblant leurs faiblesses (Not Only SQL)

Pourquoi le NoSQL

- Les bases de données relationnelles ont une philosophie d'organisation des données bien spécifiques, avec notamment le [langage d'interrogation SQL](#)
- Bien utiles pour [gérer les données qualifiées de l'entreprise](#), (BigData) elles ne sont pas du tout adaptées au stockage de très grandes dimension et au traitement ultra rapide.
- Les bases NoSQL autorisent la redondance pour mieux servir les besoins en matière de flexibilité, de tolérance aux pannes et d'évolutivité.

Pourquoi le NoSQL

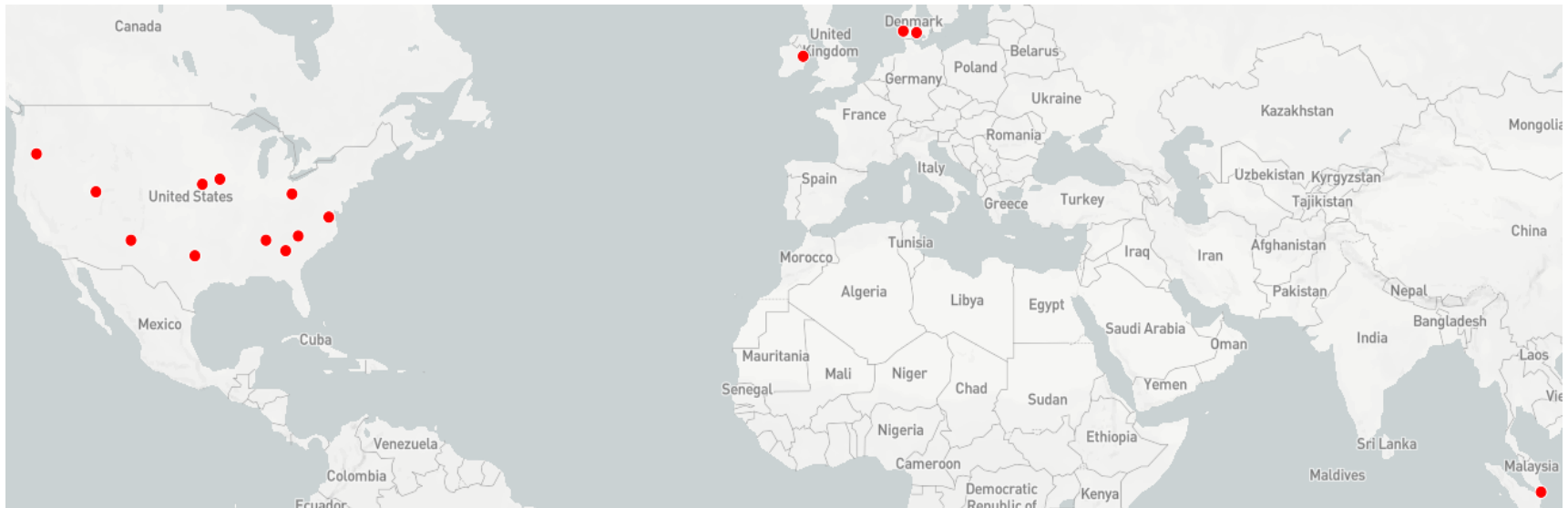
- Nouveaux besoins en gestion de données: systèmes distribués:
- 2 scénarios de traitement des données :
 - ✓ Par distribution des traitements (scaling des traitements) :
 - On distribue les traitements sur un nombre de machines important afin d'absorber des charges très importantes.
 - On envoie les données aux endroits appropriés, et on enchaîne les exécutions distantes (scénario type Workflow implémentable avec des Web services).
 - ✓ Par distribution des données (scaling des données) :
 - On distribue les données sur un nombre important de serveurs afin de stocker de très grands volumes de données.
 - On pousse les programmes vers ces serveurs (plus efficace de transférer un petit programme sur le réseau plutôt qu'un grand volume de données - Ex : algorithme MapReduce).

Pourquoi le NoSQL

- Exemple de systemes distribués: les **Data Centers**
- Utilise des LANs (Local Area Networks). On distingue 3 niveaux de communication :
 - 1) Les serveurs sont regroupés en racks. Liaison réseau rapide, environ 1Go/sec.
 - 2) Un data center consiste en un grand nombre de racks, interconnectés par des routeurs (switches). Liaison à 100 Mo/sec.
 - 3) Entre différents data centers, il y a aussi une possibilité de communication mais par liaison assez lente (internet - 2-3 Mo/sec).

Pourquoi le NoSQL

- DataCenter Facebook (2010) : 2500 CPU (serveurs), 1 PetaByte d'espace disque (= milleTerabytes = 1 million de Gigabytes), plus de 250 Gigabytes données compressées (plus de 2 Terabytes non compressés)



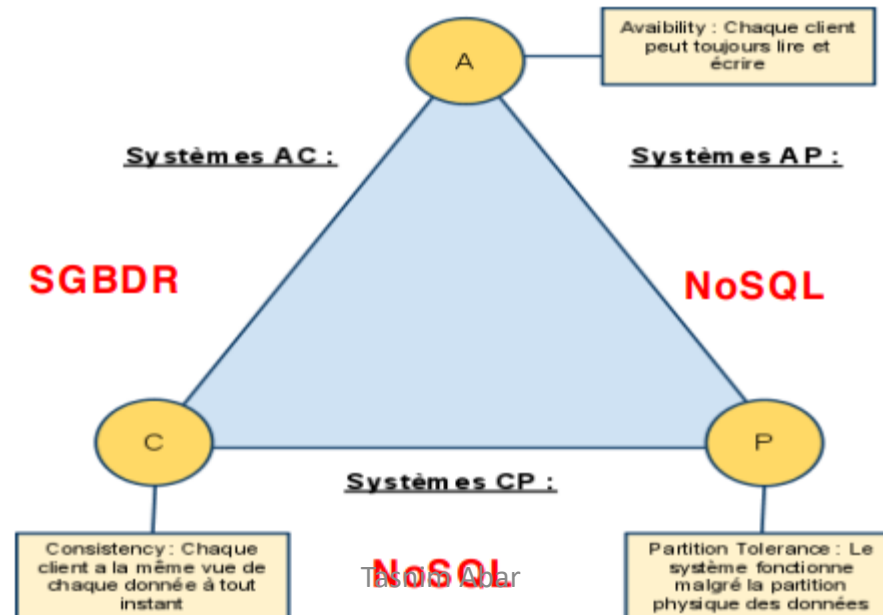
Limites du SGBDR : théorème de CAP

- 3 propriétés fondamentales pour les systèmes distribués :
- ✓ Coherence ou Consistance : tous les nœuds du système voient exactement les mêmes données au même moment .
- ✓ Availability ou Disponibilité : garantie que toutes les requêtes reçoivent une réponse.
- ✓ Partition tolerance ou Résistance au partitionnement : Quel que soit le nombre de serveurs, toute requête doit fournir un résultat correct

Théorème de CAP (Brewer, 2000) : Dans un système distribué, il est impossible d'obtenir ces 3 propriétés en même temps, il faut en choisir 2 parmi les 3

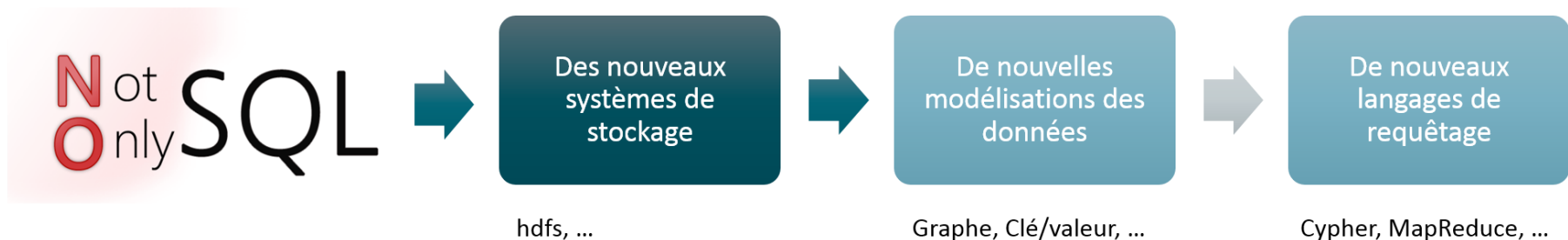
Limites du SGBDR : théorème de CAP

- Les SGBDR assurent les propriétés de Consistance et de Disponibilité (Availability) => systèmes AC
- Les SGBD « NoSQL » sont des systèmes : AP (Disponible et Résistant au partitionnement) ou CP (Cohérent et Résistant au partitionnement)



Introduction au NO SQL

- BD non relationnelles.
- est principalement appelée base de données distribuée ou non relationnelle.
- Pas forcement de schéma fixe de donnés
- Pas d'utilisation de concept de jointure
- Exemple de No sql : mongodb, neo4j, cassandra...
- Nécessaire pour le traitement big data
- Des besoins différents, des structures de donnés différentes
- Utilisées pour des objets complexes et hétérogènes.



Introduction au NO SQL

- Ont un schéma dynamique pour les données non structurées.
- NoSQL sont évolutives horizontalement:
- sont mises à l'échelle en augmentant le nombre de serveurs de bases de données dans le pool de ressources afin de réduire la charge.
- Les requêtes sont axées sur la collecte de documents. Parfois, il est également appelé UnQL (Unstructured Query Language). La syntaxe d'utilisation de UnQL varie d'une base à l'autre.
- Données distribuées : partitionnement horizontal des données sur plusieurs nœuds (serveurs) généralement par utilisation d'algorithmes MapReduce.
- Réplication des données sur plusieurs nœuds.
- Privilégient la Disponibilité à la Cohérence (théorème de CAP) : AP (Disponible + Résistant au partitionnement) plutôt que CP (Cohérent + Résistant au partitionnement). \Rightarrow N'ont en général pas de gestion de transactions.

Introduction au NO SQL:

Caractéristiques

- Les bases de données No SQL assure 2 principales caractéristiques:
- Prise en charge de l'extensibilité
 - ✓ Réplication
 - ✓ Distribution des données
- Tolérance au partitionnement

Introduction au NO SQL:

Caractéristiques

Prise en charge de l'extensibilité :

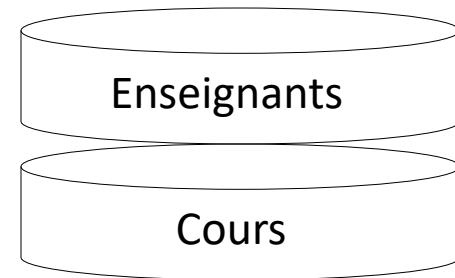
- Par réplication:
 - ✓ Réplication des données sur plusieurs nœuds
- Distribution de données (Sharding)
 - ✓ Répartir les données sur plusieurs machines pour assurer la scalabilité
 - Partitionnement horizontal
 - Partitionnement vertical

Introduction au NO SQL:

Caractéristiques

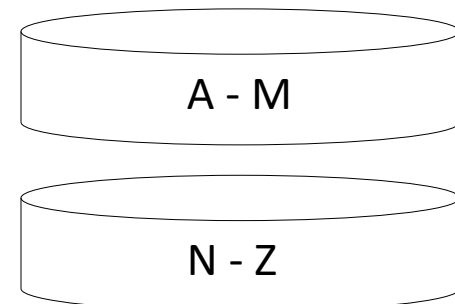
Partitionnement vertical

- ✓ Le moins utilisé
- ✓ Isoler les concepts métiers: stocker les enseignants sur une base et leurs cours sur autre.



Partitionnement horizontal

- ✓ Répartir l'ensemble des enregistrements d'une seule table sur plusieurs machines.
- ✓ Nécessite une clé de répartition (la 1^{ère} lettre du nom)



Introduction au NO SQL:

Caractéristiques

Exemple: Consistent hashing:

- Exemple de partitionnement horizontal
- Basé sur le hashage des objets et hashage des nœuds

Principe

- facilite la distribution des données sur un ensemble de nœuds de manière à minimiser le remappage / la réorganisation des données lorsque des nœuds sont ajoutés ou supprimés.
- 1- **Création de l'espace de clé de hachage:** on considère qu'on a une fonction de hachage qui génère des valeurs de hachage entières dans la plage $[0, 2^{32}-1]$

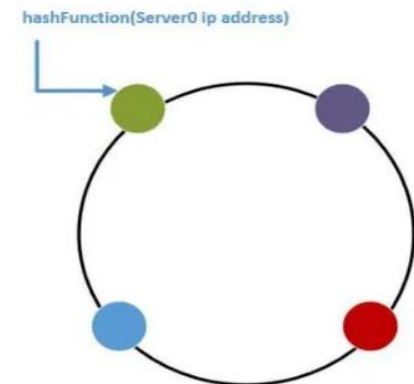
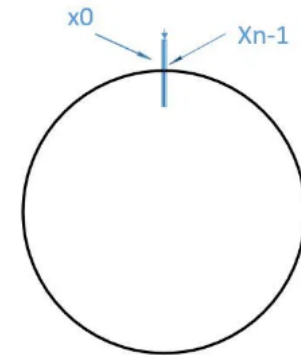
Introduction au NO SQL:

Caractéristiques

2- Représenter le hashSpace comme un anneau

3- Placement des serveurs DB dans l'espace clé

(HashRing) : on a reçu une liste de serveurs de base de données pour commencer. En utilisant la fonction de hachage, on mappe chaque serveur db à un endroit spécifique sur l'anneau. Par exemple, si on a 4 serveurs, on peut utiliser un hachage de leur adresse IP pour les mapper à différents entiers à l'aide de la fonction de hachage. Cela simule le placement des quatre serveurs à un endroit différent sur l'anneau.



Introduction au NO SQL:

Caractéristiques

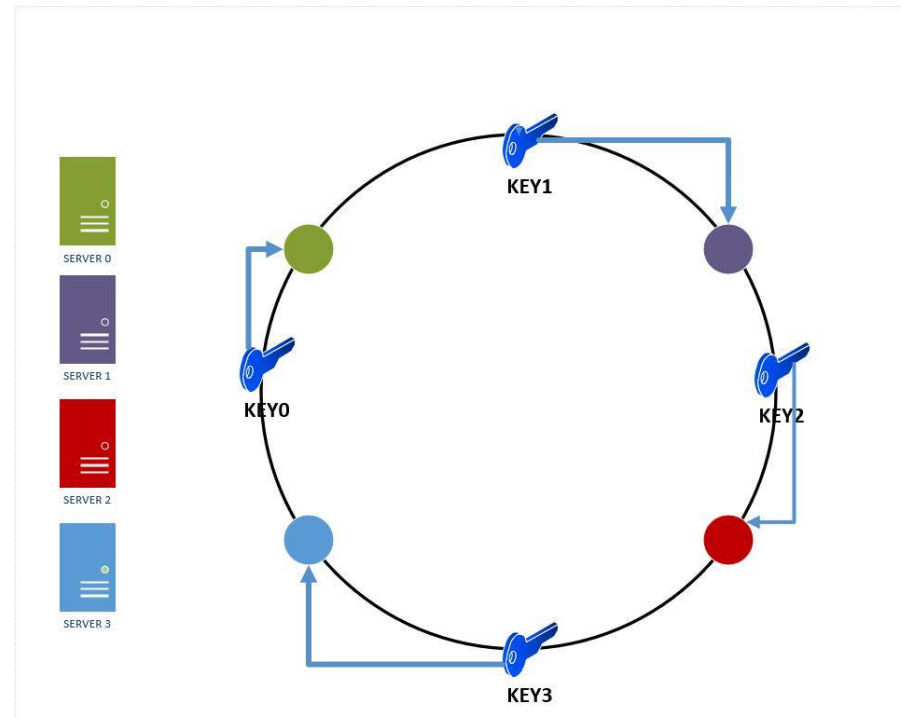
- 4- **Détermination du placement des clés sur les serveurs:** pour trouver sur quel serveur de base de données réside une clé entrante (pour l'insérer ou la rechercher) il faut :
- ✓ Exécuter la clé via la même fonction de hachage utilisée pour déterminer le placement du serveur db sur l'anneau.
 - ✓ Après avoir haché la clé, on obtient une valeur entière qui sera contenue dans l'espace de hachage, c'est-à-dire qu'elle peut être mappée à une position dans l'anneau de hachage. Il peut y avoir deux cas:
 - La valeur de hachage est mappée à un endroit de l'anneau qui n'a pas de serveur db. Dans ce cas, on voyage dans le sens des aiguilles d'une montre sur l'anneau à partir du point où la clé mappée jusqu'à ce que on trouve le premier serveur db. Une fois que on trouve le premier serveur db voyageant dans le sens des aiguilles d'une montre sur l'anneau, on y insère la clé. La même logique s'appliquerait en essayant de trouver une clé dans le ring.
 - La valeur de hachage de la clé est directement mappée sur la même valeur de hachage d'un serveur db - auquel cas on la place sur ce serveur.

Introduction au NO SQL:

Caractéristiques

Exemple :

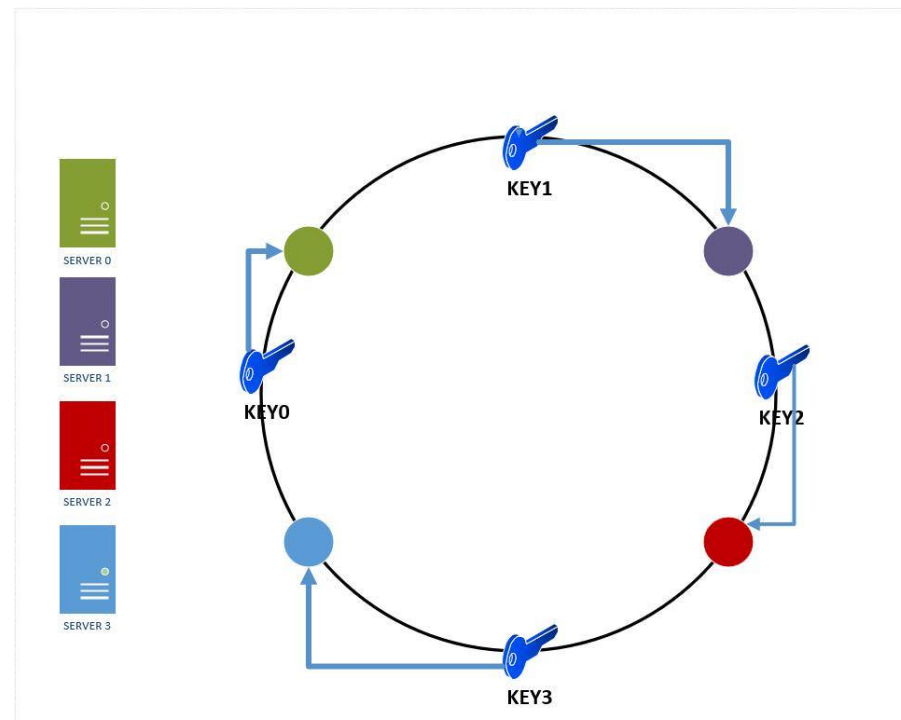
On suppose qu'on a 4 clés entrantes: key0, key1, key2, key3 et aucune d'entre elles ne correspond directement à la valeur de hachage de l'un des 4 serveurs de notre anneau de hachage. On va donc dans le sens des aiguilles d'une montre à partir du point où ces clés sont mappées dans notre anneau jusqu'à ce qu'on trouve le premier serveur db et y insère la clé..



Introduction au NO SQL:

Caractéristiques

5. Ajout d'un serveur à l'anneau: Si nous ajoutons un autre serveur à l'anneau de hachage, le serveur 4, nous devons remapper les clés. Cependant, SEULEMENT les clés qui résident entre le serveur 3 et le serveur 0 doivent être remappées au serveur 4. En moyenne, nous n'aurons besoin de remapper que k / n clés, où k est le nombre de clés et n est le nombre des serveurs. Ceci contraste fortement avec notre approche de placement basée sur modulo où nous devons remapper presque toutes les clés.

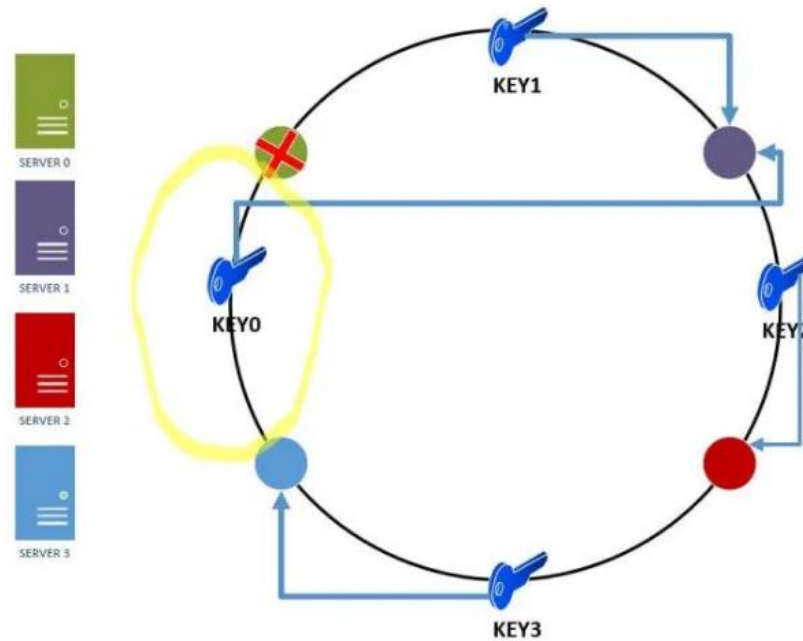


Introduction au NO SQL:

Caractéristiques

6- **Retrait d'un serveur du ring** : Un serveur peut tomber en panne et notre système de hachage cohérent garantit qu'il a un effet minimal sur le nombre de clés et de serveurs concernés.

Exemple: si le serveur 0 tombe en panne, seules les clés situées entre le serveur 3 et le serveur 0 devront être redirigées vers le serveur 1 (la zone est encerclée en jaune). Les autres clés ne sont pas affectées



Introduction au NO SQL:

Caractéristiques

Tolérance au partitionnement:

- Le système continue à fonctionner même si une partie du système est perdue ou tombe en panne.
- Aucune panne moins importante qu'une coupure totale du réseau ne doit empêcher le système de répondre correctement (ou encore : en cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome).

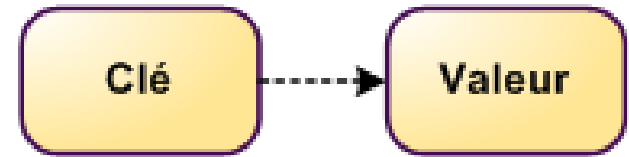
Les familles No SQL

- Clé-valeur
- Orientée colonne
- Orientée document
- Orientée graphe

Les familles No SQL

Clé-valeur

- La représentation en clé-valeur est la plus simple et est très adaptée aux caches ou aux accès rapides aux informations.
- BD = 1 tableau associatif unidimensionnel
- Les clés sont triées en ordre lexicographique



BDD Clé-Valeur

magasin	cout
---------	------

client	id
--------	----

Les familles No SQL

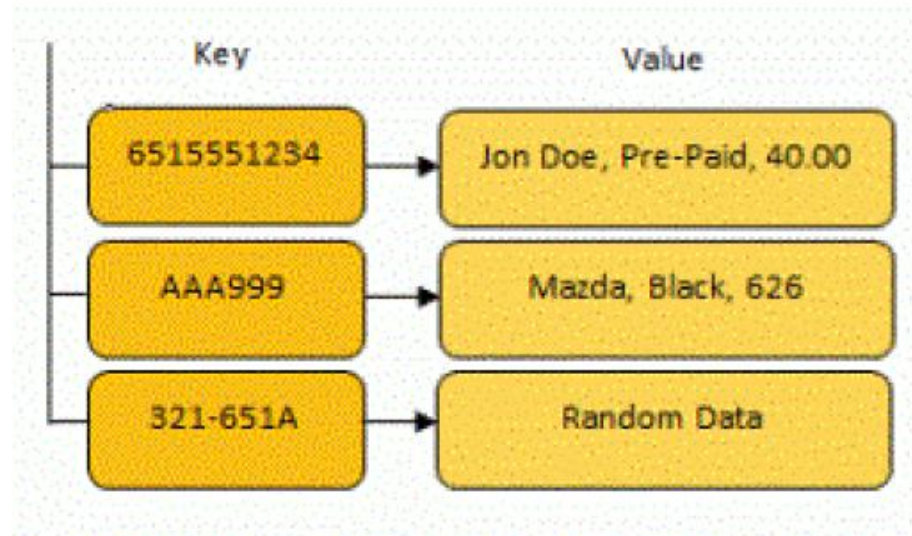
Clé-valeur

- **Opérations**
- Les 4 opérations CRUD:
 - ✓ create (clé,valeur) : crée un couple (clé,valeur)
 - ✓ Read (clé) : lit une valeur à partir de sa clé.
 - ✓ Update (clé,valeur) : modifie une valeur à partir de sa clé.
 - ✓ Delete (clé) : supprime un couple à partir de sa clé.
- **Cas d'utilisation**
- Dépôt de masses de données avec des besoins de requêtage simple pour des analyses en temps-réel :
 - sessions web
 - Profils utilisateurs,
 - Contenu du panier de shopping,
 - ...

Les familles No SQL

Clé-valeur

- **Logiciels**
 - ✓ Redis (projet sponsorisé par VMWare).
 - ✓ Voldemort (développé par LinkedIn en interne puis passage en open source).
 - ✓ Oracle No SQL Database
- **Exemple:**



Les familles No SQL

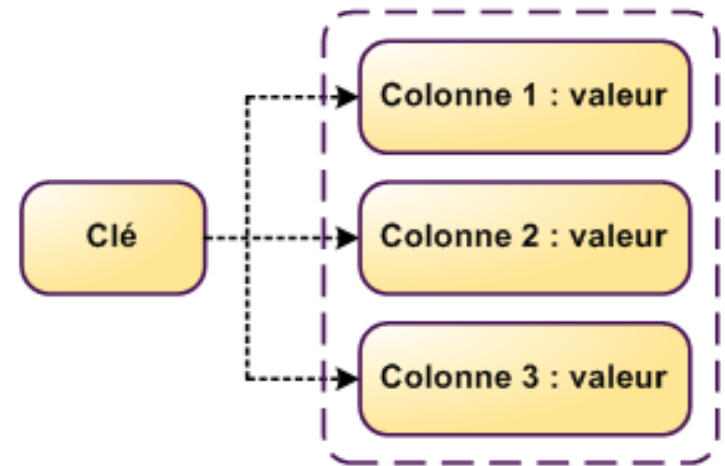
Clé-valeur

- Forces :
 - ✓ Modèle de données simple .
 - ✓ Bonne mise à l'échelle horizontale pour les lectures et écritures :
 - Evolutivité (scalable).
 - Disponibilité.
 - Pas de maintenances requises lors d'ajout/suppression de colonnes.
- Faiblesses :
 - ✓ Modèle de données TROP simple :
 - performances trop faible pour les données complexes.
 - Interrogation seulement sur clé.

Les familles No SQL

Orientées colonnes

- Proche du relationnel, mais le stockage se fait par colonne non pas par ligne
- Ajout de colonnes facile et dynamique
- Une super-colonne est une colonne contenant d'autres colonnes
- Modèle proche d'une table dans un SGBDR mais ici le nombre de colonnes:
 - ✓ est dynamique.
 - ✓ Peut varier d'un enregistrement à un autre ce qui évite de retrouver des colonnes ayant des valeurs NULL



BDD Orientée colonnes

Les familles No SQL

Orientées colonnes

- **Opérations**

- ✓ Les requêtes doivent être prédéfinies en fonction de l'organisation en colonnes choisie.

- **Cas d'utilisation**

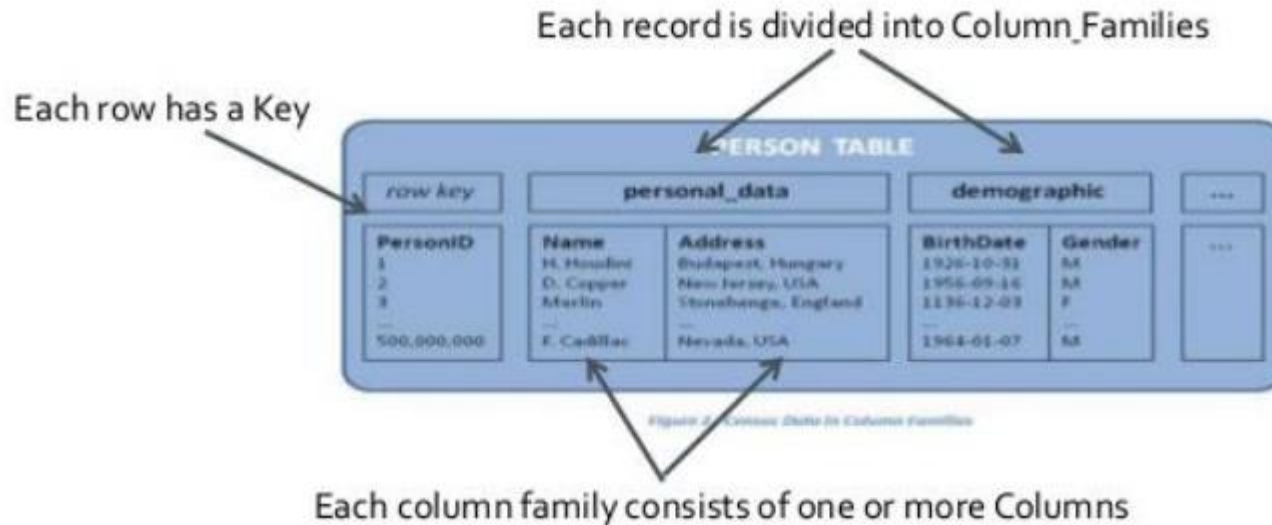
- ✓ Analyse de données
- ✓ traitement analytique en ligne (On Line Analytical Processing (OLAP))
- ✓ Stockage de listes (messages, posts, commentaires,...)
- ✓ Entrepôt de données (datawarehouse)
- ✓ Ex.:Netflix(logging et analyse de sa clientèle), eBay Inc.(optimisation de la recherche), Adobe Systems Incorporated...

- **Logiciels**

- ✓ Cassandra
- ✓ Hbase
- ✓ BigTable

Les familles No SQL Orientées colonnes

- **Exemple**

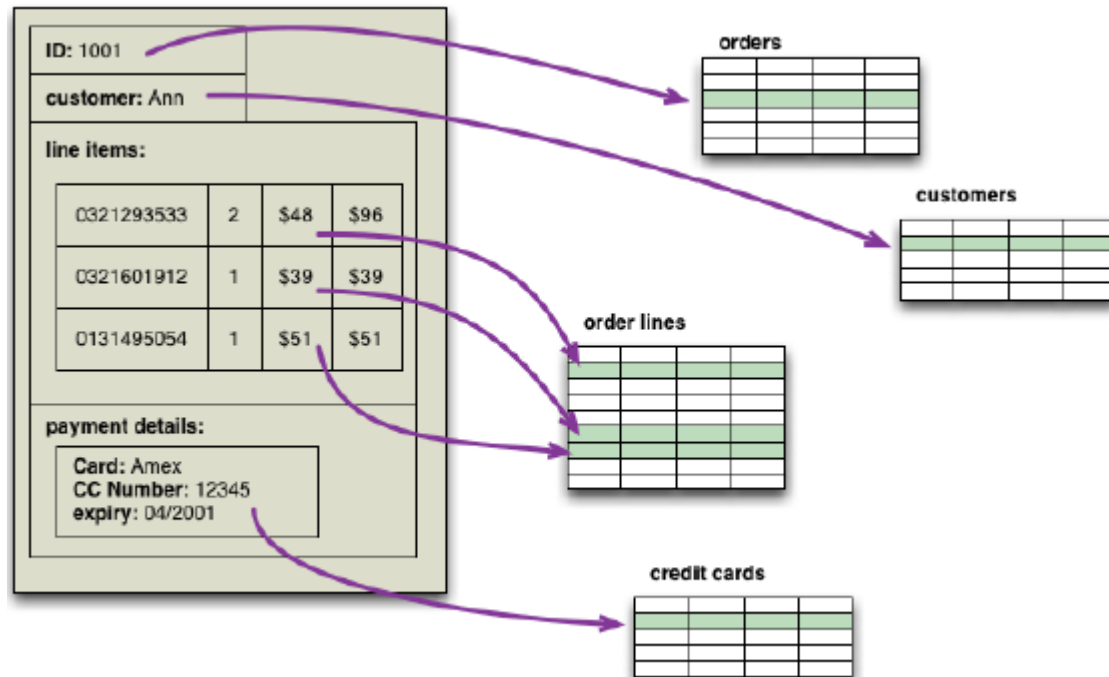


Les familles No SQL

Orientées colonnes

- **Exemple**

- Une colonne pourrait rassembler plusieurs données stockées dans des lignes qui s'étendent sur plusieurs tables d'une base de données relationnelle.



Les familles No SQL

Orientées colonnes

- **Relationnel vs orientée colonne**

Relationnel	Colonne
schéma	Keyspace
table	Famille de colonne
Clef primaire	Idem
Nom de colonne	Idem
Valeur de colonne	Idem

- Elles sont les plus complexes à appréhender des BD NoSQL, même si au final on a un schéma assez proche des bases documentaires.
- Elles offrent plus de flexibilité que les BDR :
 - il est possible d'ajouter une colonne ou une super colonne à n'importe quelle ligne d'une famille de colonnes, colonnes ou super-colonne à tout instant

Les familles No SQL

Orientées colonnes

- Forces :
 - ✓ Supporter des données semi-structurées.
 - ✓ Indexation (colonnes).
 - ✓ Bonne mise à l'échelle à l'horizontale. (utilisation de MapReduce)
- Faiblesses :
 - ✓ Modèle de données TROP simple : `a éviter pour des données interconnectés
 - ✓ A éviter pour des données complexes.
 - ✓ Exige de la maintenance - lors de l'ajout / suppression de colonnes et leur regroupements.

Les familles No SQL

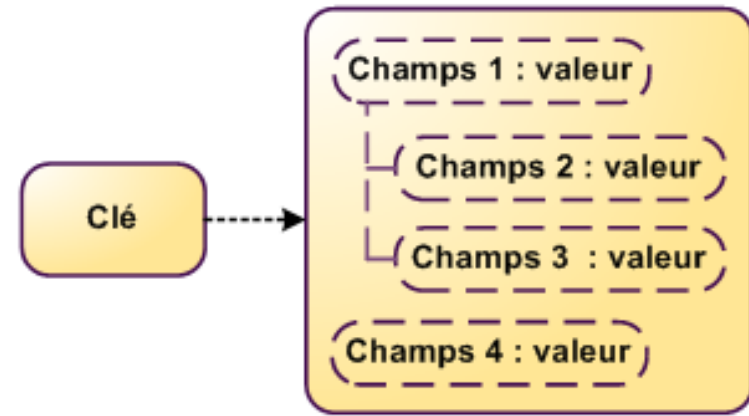
Orientées colonnes

- Les BD NoSQL orientées colonnes sont principalement utilisées pour :
 - Netflix : l'utilise notamment pour le logging et l'analyse de sa clientèle.
 - Ebay : l'utilise pour l'optimisation de la recherche.
 - Adobe : l'utilise pour le traitement des données structurées et de Business Intelligence (BI).
 - Des sociétés de TV l'utilisent pour cerner leur audience et gérer le vote des spectateurs (nombre élevé d'écritures rapides et analyse de base en temps réel (Cassandra)).
 - Peuvent être de bons magasins d'analyse des données semi-structurées

Les familles No SQL

Orientées document

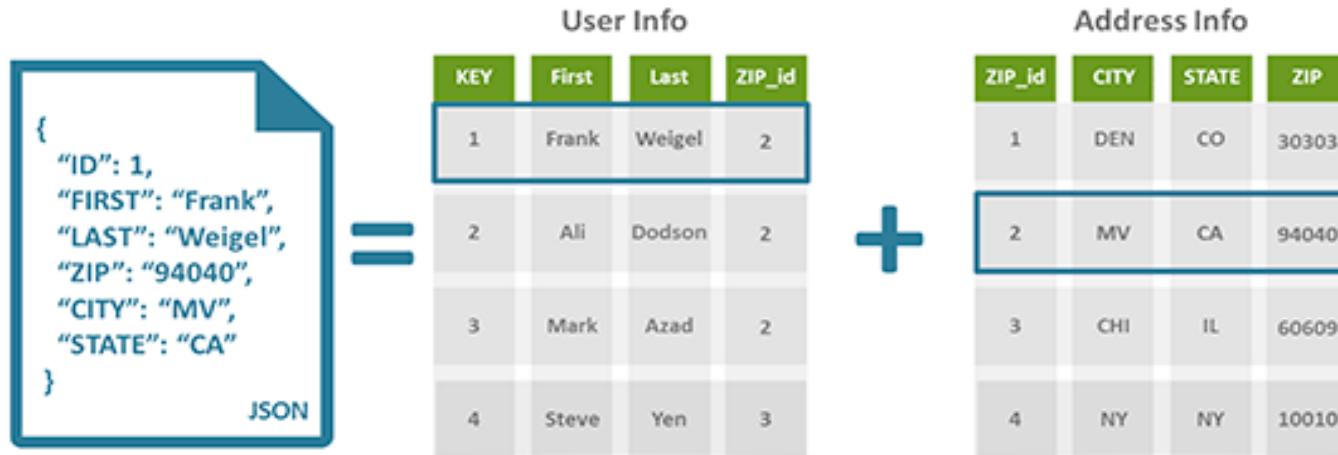
- La BD est une collection de documents.
- La représentation en document est particulièrement adaptée au monde du Web.
- Il s'agit d'une extension du concept de clé-valeur qui représente la valeur sous la forme d'un document.
- Les documents peuvent être très hétérogènes au sein de la BD
- Un document contient des données organisées de manière hiérarchique à l'image de ce que permettent XML , JSON, BSON
- Pouvoir de récupérer, via une seule clé, un ensemble d'informations structurées de manière hiérarchique
- Dans les bases relationnelles, cela impliquerait plusieurs jointures.



BDD Orientée document

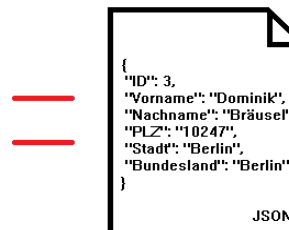
Les familles No SQL

Orientées document



Key	Vorname	Nachname	PLZ_id
1	Dennis	Bossmann	1
2	Fabian	Buch	2
3	Dominik	Bräusel	2
4	Linda	Dräusdorf	4

PLZ_id	Stadt	Bundesland	PLZ
1	Hannover	Niedersachsen	30159
2	Berlin	Berlin	10247
3	München	Bayern	80331
4	Köln	NRW	50667



Les familles No SQL

Orientées document

- **Cas d'utilisation**

- ✓ Outils de gestion de contenu (Content Management System (CMS))
- ✓ catalogues de produits
- ✓ web analytique enregistrement d'événements
- ✓ Systèmes d'exploitation
- ✓ Gestion de données semi-structurées

- **Logiciels**

- ✓ CouchDB
- ✓ RavenDB
- ✓ MongoDB
- ✓ Terrastore

Les familles No SQL

Orientées document

- **Avantages du format JSON:**
 - ✓ Format abstrait pour une représentation simplifiée dans les différents langages.
 - ✓ Indépendant du langage de programmation.
 - ✓ Simple et complet pour la représentation des objets.
 - ✓ Utilise des types de données connus et simples à décrire.
 - ✓ Une bonne lisibilité de la syntaxe.
 - ✓ Temps de traitement très proche de celui des fichiers XML.
 - ✓ Moins volumineux en terme de stockage.

Les familles No SQL

Orientées document

Relationnel	Orienté document
schéma	database
table	Collection
colonne	Champ
ligne	Document

Les familles No SQL

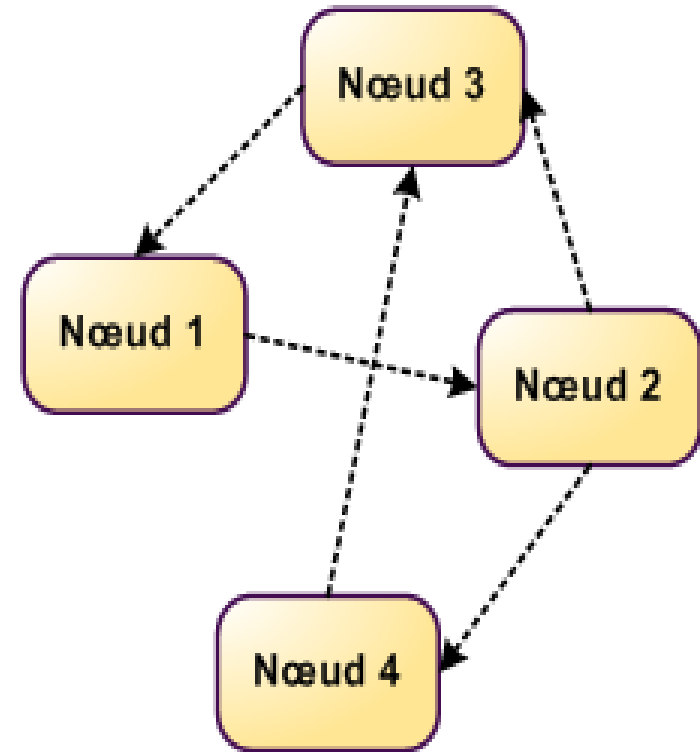
Orientées document

- Forces :
 - ✓ Bonne mise à l'échelle.
 - ✓ Pas de maintenance de la BD requise pour ajouter/supprimer des colonnes
- Faiblesses :
 - ✓ Peut alors être lent pour les grandes requêtes (avec MapReduce)

Les familles No SQL

Orientées Graphe

- Une base de données de type graphe stocke les données dans un graphe. (Des nœuds pour les entités (entreprise, personne...), des arcs pour les relations entre les entités.)
- Elle est basée sur les théories des graphes.
- Elles permettent la modélisation, le stockage et la manipulation de données complexes liées par des relations non-triviales ou variables.
- Adapté aux traitements des données des organisés en réseaux (réseaux sociaux, réseaux de transport ...)



BDD Orientée graphe

Les familles No SQL

Orientées Graphe

- **Opérations**

- ✓ SPARQL pour les SGBD No SQL :GrapheRDF
- ✓ API et langages spécialisés de programmation et de requêtes sur les graphes

- **Cas d'utilisation**

- ✓ Moteurs de recommandation
- ✓ informatique décisionnelle
- ✓ Données géo spatiales
- ✓ Réseaux sociaux
- ✓ Réseaux de transport
- ✓ Services de routage et d'expédition
- ✓ Services financiers

- **Logiciels**

- ✓ Neo4J
- ✓ OrientDB
- ✓ Titan

Les familles No SQL Orientées Graphe

- Forces :
 - ✓ Rapide et puissant
 - ✓ Indexation (colonnes).
 - ✓ Bonne mise à l'échelle à l'horizontale. (utilisation de MapReduce)
- Faiblesses :
 - ✓ La répartition de données ne convient pas parfois avec des jeux de données volumineux.
 - ✓ Sharding (fragmentation)

Les familles No SQL

Orientées Graphe

Relationnel	Graphe
schéma	database
table	Ensemble des nœuds
colonne	Attribut
ligne	Nœud défini par une propriété

Les familles No SQL Orientées Graphe

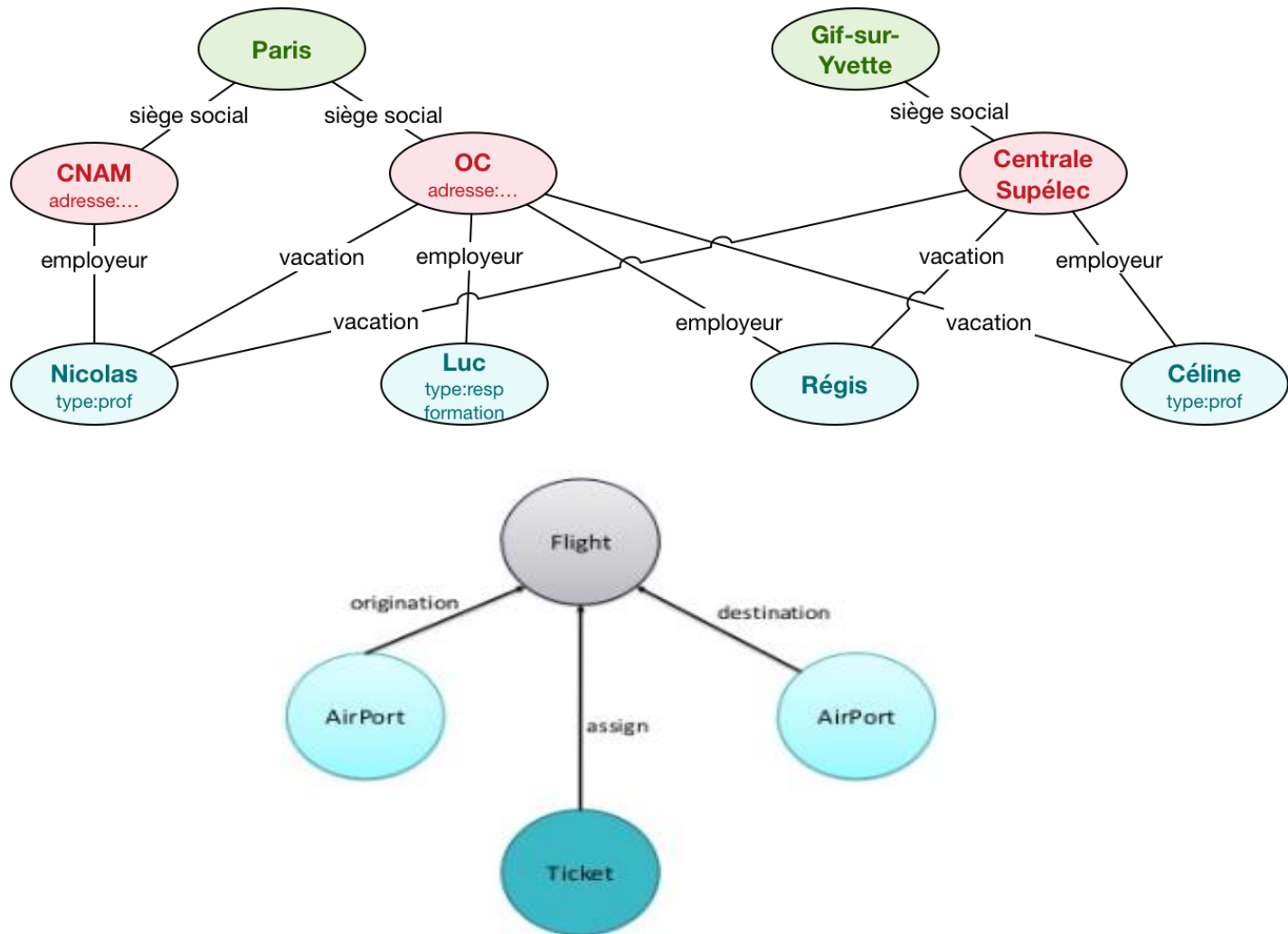


Illustration des modèles NoSQL

Exemple : Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Représentation de ventes en colonne

Family Sales

#ticket	date	books
1	01/01/16	2212121504 2212141556
2	01/01/16	2212141556

Family Book

#isbn	title	a-surname	a-firstname
2212121504	Scenari		
2212141556	NoSQL	Bruchez	Rudi

Illustration des modèles NoSQL

Exemple : Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Représentation de ventes en document

Collection Sales

#oid	
4d0407660766b236450b45a3	<pre> { "ticket" : 1 "date" : "01/01/16" "books" : [{ "isbn" : 2212121504 "title" : "Scenari" } { "isbn" : 2212141556 "title" : "NoSQL" "author" : { "surname" : Bruchez "firstname" : Rudi } }] }</pre>
4d0407660766b236450b45a4	<pre> { "ticket" : 2 "date" : "01/01/16" "books" : [{ "isbn" : 2212141556 "title" : "NoSQL" "author" : { "surname" : Bruchez "firstname" : Rudi } }] }</pre>

Illustration des modèles NoSQL

Exemple : Représentation de ventes en relationnel

Table Sales

#ticket	#date	#book
1	01/01/16	2212121504
1	01/01/16	2212141556
2	01/01/16	2212141556

Table Book

#isbn	#title	#author
2212121504	Scenari	1
2212141556	NoSQL	2

Table Author

#id	surname	firstname
1		
2	Bruchez	Rudi

Représentation de ventes en graphe

Classe Sales

#oid	
4d0407660766b236450b45a3	<i>property</i> ticket : 1
	<i>property</i> date : 01/01/16
	<i>relation</i> book : 4d0407660766b236450b45a5
	<i>relation</i> book : 4d0407660766b236450b45a6
4d0407660766b236450b45a4	<i>property</i> ticket : 2
	<i>property</i> date : 01/01/16
	<i>relation</i> book : 4d0407660766b236450b45a6

Classe Book

#oid	
4d0407660766b236450b45a5	<i>property</i> title : Scenari
4d0407660766b236450b45a6	<i>property</i> title : NoSQL
	<i>relation</i> author : 4d0407660766b236450b45a8

Classe Author

#oid	
4d0407660766b236450b45a8	<i>property</i> surname : Bruchez
	<i>property</i> firstnam : Rudi