Introduction aux bases de données NoSQL

Bernard ESPINASSE
Professeur à Aix-Marseille Université (AMU)
Ecole Polytechnique Universitaire de Marseille

Septembre 2021

1. **Des BD relationnelles au BD NoSQL :**
   Limites des bases de données relationnelles ; Principales caractéristiques des BD NoSQL ; types de BD NoSQL
2. **BD NoSQL type « clé-valeur » :** Modèle, forces, faiblesse, usages
3. **BD NoSQL type « colonnes » :** Modèle, forces, faiblesse, usages
4. **BD NoSQL type « documents » :** Modèle, forces, faiblesse, usages
5. **BD NoSQL type « graphes » :** Modèle, forces, faiblesse, usages
6. **Conclusion**

Plan

Principales sources du cours

Documents :

- C. Strauch, « Nosql databases », Lecture Notes, Stuttgart Media University, 2011.
- A. Foucret, « Livre blanc sur NoSQL », par Smile (http://www.smile.fr/Livres-blancs/Culture-du-web/NoSQL).
- S-K. Gajendran, « A Survey on NoSQL Databases ».
- S. Abiteboul, I. Manolescu, P. Rigaux, M-C Rousset, P. Senellart, « Web Data Management », Cambridge University Press 2011 (en ligne, la 3ème partie : http://webdam.inria.fr/Jorge/?action=chapters).
- J. Dean and S. Ghemawat, « MapReduce: Simplified Data Processing on Large Clusters », OSDI 2004.
- B. Espinasse, P. Bellot, « Introduction au Big-Data: opportunité, stockage et analyse des mégadonnées », in Dossiers Techniques de l'Ingénieur (DTI), Ref. H6040, 2017.
- …

Présentations :

- F. Duchateau, « Les SGBD Non-relationnels », Univ. Lyon 1, 2014.
- P. Selmer, « NOSQL stores and Data analytics tools », Advances in Data Management, 2012.
- A.-C. Caron, « NoSQL », Université de Lille 1.
- M. Jaffré, P. Rauzy, « MapReduce », ENS, 2010.
- K. Tannir, « MapReduce : Algorithme de parallélisations des traitements », 2011, http://**blog.khaledtannir.net**/wp-content/.../KT-Presentation-MapReduce.pdf
- …

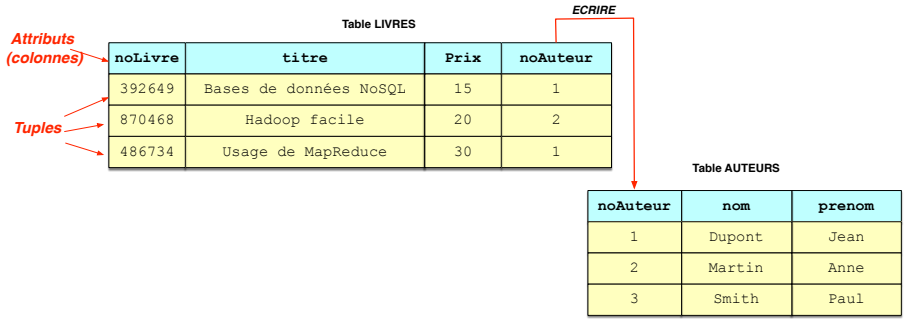
1. Des BD Relationnelles aux BD NoSQL

- **Limites des bases de données relationnelles**
- **Principales caractéristiques des BD NoSQL**
- **Types de BD NoSQL**

Bases de données Relationnelles

- Très **largement implantées, incontournables dans tous les SI**
- **Grande maturité** : bientôt 50 ans.
- Collection de **tables** (fichiers) inter-reliées
- **Système de jointure** entre les tables permettant de construire des **requêtes complexes** impliquant plusieurs entités avec un **langage de requête (SQL)**
- **Système d'intégrité référentielle** assurant que les **liens** entre les entités sont **valides**

Exemple de BD relationnelle :



BD Relationnelles et transactions : modèle ACID

- ***Transaction*** = traitement permettant le passage de la BD d'un état cohérent à un autre état cohérent
- SGBD relationnels assurent une *gestion des transactions* respectant les **le modèle ACID** (Atomicity, Consistency, Isolation, Durability) assurant toujours l'intégrité de la BD

Exemple : le distributeur de billets

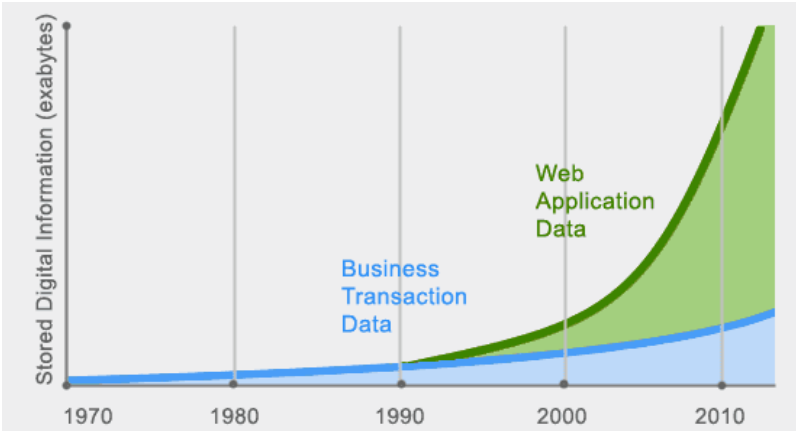
Atomicity *[atomicité]* Accomplie entièrement ou pas du tout	Lors d'un retrait, la transaction doit s'accomplir intégralement ou pas du tout
Consistency *[cohérence]* Passage d'un état valide à un autre état valide	La somme des billets retirée doit correspondre au débit mémorisé dans la base
Isolation *[indépendance]* Indépendance des transactions concurrentes	Deux opérations simultanées de deux personnes sur un compte commun
Durability [persistance] Résultat d'une transaction stocké de manière durable	Le retrait de billet doit correspondre à un débit mémorisé dans la base

Nouveaux besoins en gestion de données

Constat :

- Essor des **très grandes plateformes** et **applications Web (**Google, Facebook, Twitter, LinkedIn, Amazon, …) trainant des mégadonnées.
- **VOLUME : volume considérable de données** à gérer par ces applications
  - *Facebook : 890 millions d'utilisateur en moyenne par jour*
  - *Twitter : 284 millions d'utilisateur actifs*
- **VARIETE** : données principalement **peu structurées** et **hétérogènes**

Evolution des volumes de données d'entreprise versus Web



Évolution des volumes de données d'entreprise versus web

Limites des BD relationnelles

Limites liées au VOLUME des données

- Les BD relationnelles peuvent gérer de très gros volumes de données : usage de *machine bases de données* pour des entrepôts de données *(par distribution des données sur de nombreux disques permettant une parallélisation des requêtes).*
- Mais ces machines ne peuvent gérer des mégadonnées au delà d'un certain volume.
  - ⇨ ***Stockage et traitement fortement distribués** des données sur des serveurs (Data Centers)*
  - ⇨ *Map-Reduce et Hadoop …*

Limites liées à la VARIETE des données

- Le modèle relationnel est fortement structuré et rigide (tables de données, formes normales …)
- Les mégadonnées et les applications Web concernent des données souvent peu structurées et hétérogènes (textes, images, …) mal adaptées
  - ⇨ *Recherche de nouveaux **modèles de données de stockage plus flexibles** (voir pas de modèle du tout)*

Limites liées au VOLUME : distribution

SGBD Relationnels :

- système de jointure entre les tables permettant de construire des **requêtes complexes** impliquant plusieurs entités avec un **langage de requête (SQL)**
- système d'intégrité référentielle permettant de s'assurer que les **liens** entre les entités sont **valides**

⇨ *Mécanismes permettant une **gestion des transactions** garantissant le maintien des **propriétés ACID** (Atomicité, Cohérence, Isolation et Durabilité)*

Contexte fortement distribué => mécanismes très couteux :

- avec la plupart des SGBD relationnels, les **données d'une BD liées entre elles,** sont placées sur le **même nœud** du serveur
- si le **nombre de liens important**, il est de plus en plus **difficile de placer les données sur des nœuds différents**.

⇨ ***Relâchement des propriétés ACID (Théorème de CAP)***

Théorème de CAP (1)

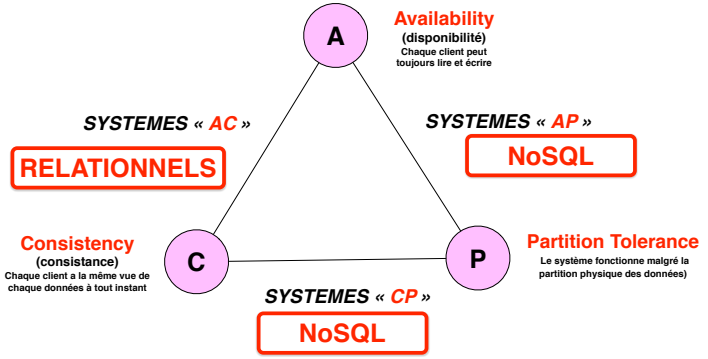
3 propriétés fondamentales pour les **systèmes distribués** :

- **C oherence** *ou Consistance* : tous les nœuds du système voient exactement les mêmes données au même moment
- **A vailability** *ou Disponibilité* : la perte de nœuds n'empêche pas les survivants de continuer à fonctionner correctement, les données restent accessibles
- **P artition tolerance** *ou Résistance au partitionnement* : le système étant partitionné, aucune panne moins importante qu'une coupure totale du réseau ne doit l'empêcher de répondre correctement : le système continue à fonctionner malgré les défaillances d'une partie des nœuds

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

Théorème de CAP (2)



- Les **SGBD RELATIONNELS** 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)*

⇨ ***Conséquence** : les nouvelles solutions de stockage (**NoSQL**) n'assurent pas les propriétés ACID : **relâchement** de ces propriétés est nécessaire.*

Limites liées à la VARIETE : modèles de données

Modèle relationnel :

- les données sont organisées en **lignes** (tuples – rows) avec des **colonnes** (attributs, columns) **prédéfinies**
- **modèle d'exploitation : algèbre relationnelle** base d'un **langage** de requête et de mise à jour (**SQL**)
- contraintes :
  - **pas de tuples emboîtés autorisés**
  - **pas de liste de valeurs autorisés**

⇨ *Modèle de données rigide mais possédant un algèbre et un langage puissant de requête et mise à jour dérivé (SQL)*

Mégadonnées et applications Web utilisent principalement :

- une **collection d'objets reliés**
- qui doivent être **traités ensemble.**

⇨ *Besoin de modèles de données plus souples (modèle d'agrégats) ou PAS de modèle du tout, mais généralement SANS langage de requête*

Nouvelles solutions : BD NoSQL

- Face aux limites des BD relationnelles de nouvelles solutions ont vu le jour permettant :
  - une **meilleure scalabilité** dans des **contextes fortement distribués** (cluster de serveurs, data centers)
  - une **gestion d'objets complexes et hétérogènes** sans avoir à déclarer au préalable l'ensemble des champs représentant un objet (modèles de données flexibles ou pas de modèle du tout)
- Solutions regroupées derrière le terme **NoSQL** (Carl Strozzi)
- Solutions **ne se substituant pas** aux SGBD Relationnels mais les complétant en comblant leurs faiblesses (Not Only SQL) dans certains contextes d'applicatifs spécifiques (mégadonnées et applications Web).

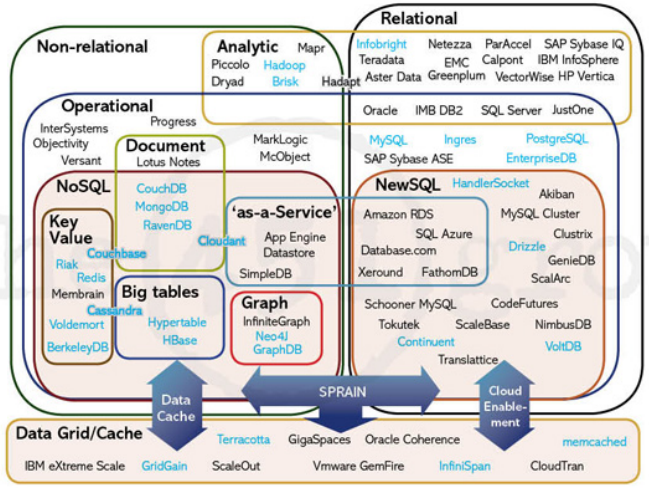
Le grand paysage des bases de données (1)

Face aux limites des SGBDR à gérer des mégadonnées, **différentes solutions à architectures distribuées** émergent :

- **NoSQL BD :** BD avec **schéma dynamique ou sans schéma**, BD magasins **de clés-valeur**, BD **de documents** et de **graphes de données, … :** *solutions déjà mises en œuvre*
- **NewSQL DB** : amélioration des **performances** des BD Relationnelles grâce à de **nouveaux moteurs de stockage**, de nouvelles technologies de **fragmentation**, de **nouveaux logiciels et matériels** : *solutions relevant encore de la R&D*
- **Data Grid/Cache Products :** amélioration des **performances** des applications et de la BD par stockage des données en **mémoire** : **données persistantes en cache**, **réplication des données distribuées**, et **calcul exploitant le Grid, … :** *solutions relevant encore de la R&D*

Le grand paysage des bases de données (2)

Report « NoSQL, NewSQL and Beyond: The answer to SPRAINed relational databases », 451 Group, April 15th, 2011 :



BD NoSQL : du modèle ACID au modèle BASE

Théorème de CAP => BD NoSQL = systèmes AP ou CP :

- Notion de *transaction* généralement **absente.**
- Priorité de la *disponibilité* sur la *cohérence*.

Modèle **BASE** (remplace le modèle **ACID**) :

- **Basically Available** : le système garantit la ***disponibilité*** des données et répondra à toute demande,
  - ⇨ *MAIS cette réponse peut être un "échec" dans l'obtention des données demandées ou les données peuvent être incohérentes …*
- **Soft state** : ***l'état du système peut changer au fil du temps***, de sorte que même pendant les périodes sans saisie, des changements peuvent se produire en raison d'une "éventuelle cohérence" : l'état du système est toujours "***soft***".
- **Eventual consistency** : ***tôt ou tard le système deviendra cohérent***, les données se propageront partout où elles devraient se trouver
  - ⇨ *MAIS le système continuera à recevoir des données SANS vérifier la cohérence de chaque transaction avant de passer à la suivante.*

BD NoSQL : modèle BASE - exemple

Exemple : post d'un tweet

Quand un tweet est posté :

- il peut arriver que différents utilisateurs, à différents endroits dans le monde *ne le voient pas arriver exactement en même temps (BD de Twitter est distribuée => délais de propagation).*
- les incohérences temporaires ne sont pas un problème, le respect des contraintes BASE suffit.
  - ⇨ *poster un tweet n'est donc PAS une transaction ACID*

Soit :

- les *opérations de lecture et d'écriture sont disponibles* autant que possible *sur tous les nœuds*,
- *réponses approximatives* (parfois non cohérentes à 100%) : au fil du temps, les *réponses ont tendance à être de plus en plus cohérentes*,
- *au final* cela doit converger vers des *valeurs 100% cohérentes*.

Modèle de données NoSQL: Tables VS Agrégats (1)

Instance de TABLES (Relationnel)

CUSTOMER

ID	NAME
1	Guido

PRODUCT

ID	NAME
1000	IPod Touch
1020	Monster Beat

BILLING_ADDRESS

ID	CUSTOMER_ID	ADDRESS_ID
1	1	55

ADDRESS

ID	STREET	CITY	POST_CODE
55	Chaumontweg	Spiegel	3095

ORDER

ID	CUSTOMER_ID	SHIPPING_ADDRESS_ID
90	1	55

ORDER_ITEM

ID	ORDER_ID	PRODUCT_ID	PRICE
1	90	1000	250.55
1	90	1020	199.55

Instance d'Agrégat (NoSQL)

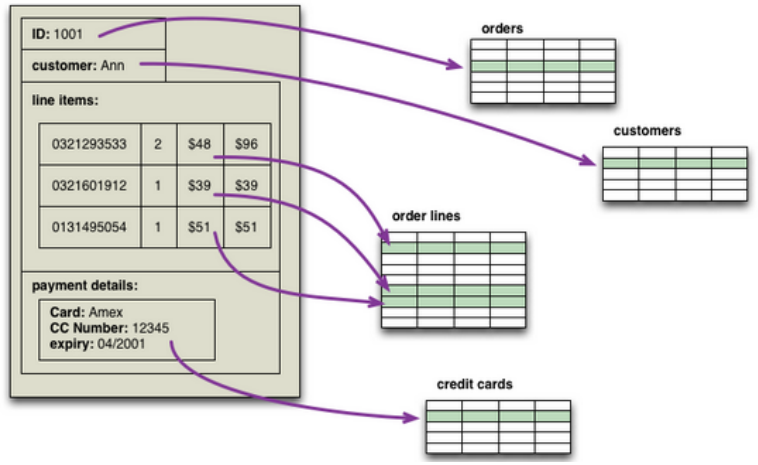
```
{
„id":1,
„name":"Guido",
„billingAddress":[{„street":"Chaumontweg","city":"Spiegel","postCode":"3095"}]
}

{
„id":90,
„customerId":1,
„orderItems":[
 {
 „productId":1000,"price": 250.55, „produtName": „iPod Touch"
 },
 {
 „productId":1020,"price": 199.55, „produtName": „Monster Beat"
 }],
„sippingAddress":[{„street":"Chaumontweg","city":"Spiegel","postCode":"3095"}]
}
```

(Martin Fowler, Aggregate Oriented Database, 19 January 2012)

Modèle de données NoSQL: Tables VS Agrégats (2)

Agrégat (NoSQL) **Tables (Relationnel)**



(Martin Fowler, Aggregate Oriented Database, 19 January 2012)

Caractéristiques générales des BD NoSQL

Modèle :

- *non relationnelle,* ***pas de schéma*** pour les données ou ***schéma dynamique***
- données de structures ***complexes*** ou ***imbriquées***

Stockage distribué :

- ***partitionnement horizontal*** des données sur plusieurs nœuds (serveurs)
- usage de ***Hadoop*** et ***MapReduce***
- ***réplication*** des données sur plusieurs noeuds

Propriétés et usages :

- privilégient la ***Disponibilité (A)*** à la Cohérence (C) (théorème de CAP)
- **compromis sur le caractère « ACID »** pour plus de scalabilité horizontale et d'évolutivité : **modèle « BASE »**
- en conséquence ont ***rarement de gestion de transactions***
- mode d'utilisation : ***peu d'écritures, beaucoup de lectures***

Typologie des BD NoSQL

Stocker les informations de la **façon la mieux adaptée** à leur représentation => **différents types de BD NoSQL :**

- **type « Clé-valeur / Key-value » :** basique, chaque objet est identifié par une clé unique constituant la seule manière de le requêter
  - ⇨ ***Systèmes : Voldemort, Redis, Riak, …***
- **type « Colonne / Column »** : permet de disposer d'un très grand nb de valeurs sur une même ligne, de stocker des relations « one-to-many », d'effectuer des requêtes par clé (adaptés au stockage de listes : messages, posts, commentaires, ...)
  - ⇨ ***Systèmes : HBase, Cassandra, Hypertable, …***
- **type « Document »** : pour la gestion de collections de documents, composés chacun de champs et de valeurs associées, valeurs pouvant être requêtées (adaptées au stockage de profils utilisateur)
  - ⇨ ***Systèmes : MongoDBn CouchDB, Couchbase, …***
- **type « Graphe »** : pour gérer des relations multiples entre les objets (adaptés au données issues de réseaux sociaux, …)
  - ⇨ ***Systèmes : Neo4j, OrientDB, …***

SGBD NoSQL disponibles (2019)

	Key/Value Store	Column Store	Document Store	Graph Store
Design	Key/Value pairs; indexed by Key	Columns and Column Families. Directly accesses the column values	Multiple Key/Value pairs form a document. Values may be nested documents or lists as well as scalar values	Focus on the **connections** between data and fast navigation through these connections
Scalability / Performance	+++	+++	++	++
Aggregate-Oriented	Yes	Yes	Yes	No
Complexity	+	++	++	+++
Inspiration / Relation	Berkley DB, Memcached, Distributed Hashmaps	SAP Sybase IQ, Google BigTable	Lotus Notes	Graph Theory
NOSQL Products	Voldemort Redis Riak	HBase Cassandra Hypertable	MongoDB CouchDB Couchbase	Neo4j OrientDB DEX InfiniteGraph [Triple and Quad Stores]

2. BD NoSQL « clé-valeur »

- **Modèles et systèmes**
- **Forces et faiblesses**
- **Usages**

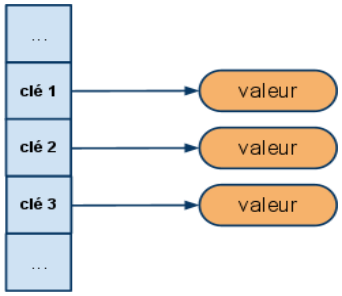
BD NOSQL modèle « Clé-Valeur » (1)

- Les données sont simplement **représentées** par un **couple clé/valeur**
- La valeur peut être une **simple chaîne de caractères**, ou un objet sérialisé…
  - ⇨ *Cette absence de structure ou de typage ont un impact important sur le requêtage : toute l'intelligence portée auparavant par les requêtes SQL devra être portée par **l'applicatif** qui interroge la BD.*
- Implémentations les plus connues :
  - **Amazon Dynamo** (**Riak** en est l'implémentation Open Source)
  - **Redis** (projet sponsorisé par VMWare)
  - **Voldemort** (développé par LinkedIn en interne puis passage en open source).

BD NOSQL « Clé-Valeur » (2)

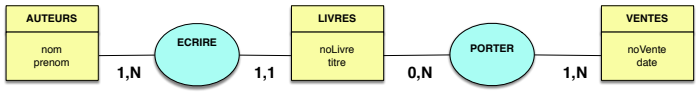
- Chaque **objet** est identifié par une **clé unique** seule façon de le requêter :

  Clé (de l'objet) --------> **Valeur**(s)
- La **structure de l'objet est libre**, souvent laissé à la charge du développeur de l'application (XML, JSON, ...), la base ne gérant généralement que des chaînes d'octets

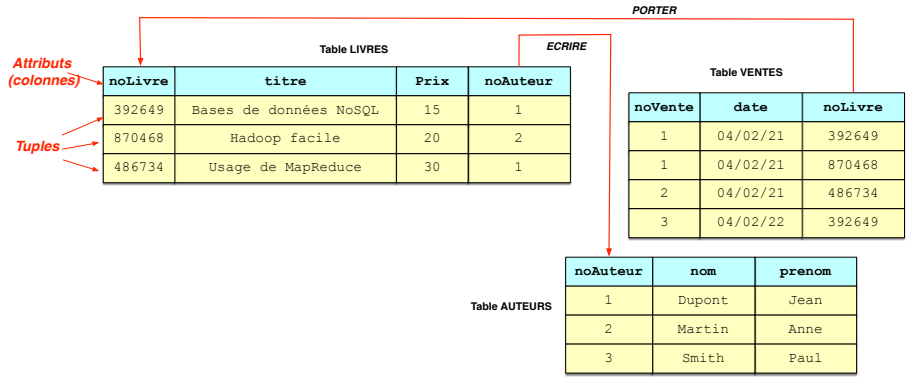


Une base de données relationnelle

MCD (Entité-Relation) :



Une extension de la BD relationnelle associée :



BD NOSQL « Clé-Valeur » : illustration

Clé	Valeur
nom-auteur1	Dupont
prenom-auteur1	Jean
nom-auteur2	Martin
prenom-auteur2	Anne
nom-auteur3	Smith
prenom-auteur3	Paul
titre-livre392649	Bases de données NoSQL
prix-livre392649	15
nom-livre392649-auteur1	Dupont
…	…

BD NOSQL « Clé-Valeur » : exploitation - CRUD

- Leur exploitation est basée sur **4 opérations (CRUD):**
  - **C reate** : créer un nouvel objet avec sa clé → create(key, value)
  - **R ead** : lit un objet à partir de sa clé → read(key)
  - **U pdate** : met à jour la valeur d'un objet à partir de sa clé → update(key, value)
  - **D elete**: supprime un objet à partir de sa clé → delete(key)
- Elles disposent généralement d'une simple **interface de requêtage HTTP REST** accessible depuis n'importe quel langage de développement,
- Elles ont des **performances très élevées** en lecture et en écriture et une **scalabilité horizontale considérable,**
- Le besoin en **scalabilité verticale est faible** du fait de la simplicité des opérations effectuées.

BD NOSQL « Clé-Valeur » : Forces & faiblesses

Forces :

- **modèle de données simple**
- **bonne mise à l'échelle horizontale** pour les lectures et écritures :
  - **évolutivité** (scalable)
  - **disponibilité**
  - **pas de maintenance requise** lors d'ajout/suppression de colonnes

Faiblesses :

- **modèle de données TROP simple :**
  - **pauvre** pour les **données complexes**
  - **interrogation seulement** sur **clé**
  - **déporte** une grande partie de la **complexité** de l'application sur la **couche application** elle-même

BD NOSQL « Clé-Valeur » : utilisations principales

Utilisations principales des BD NoSQL type « Clés-Valeurs » :

- dépôt de données avec besoins de requêtage très simples,
- système de stockage de cache ou d'information de sessions distribuées (quand l'intégrité relationnelle des données est non significative),
- les profils, préférences d'utilisateur,
- les données de panier d'achat,
- les données de capteur,
- les logs de données,
- …

3. BD NoSQL « orientée colonnes »

- **Modèles et systèmes**
- **Forces et faiblesses**
- **Usages**

BD NOSQL modèle « Colonne » (1)

- Les données sont stockées par **colonne**, et non par ligne
- On peut facilement **ajouter des colonnes** aux tables, par contre l'insertion d'une ligne est plus coûteuse
- Quand les données d'une colonne se ressemblent, on peut facilement compresser la colonne
- 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.
- Implémentations les plus connues :
  - **HBase** (Open Source de **BigTable** de Google utilisé pour l'indexation des pages web, Google Earth, Google analytics, ...)
  - **Cassandra** (fondation Apache qui respecte l'architecture distribuée de Dynamo d'Amazon, projet né de chez Facebook)
  - **SimpleDB** de Amazon.

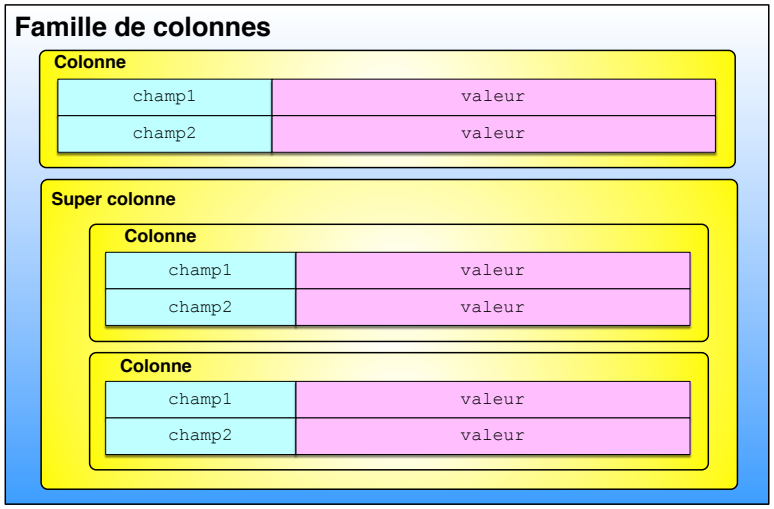
BD NOSQL « Colonne » (2)

Les **principaux concepts** associés sont les suivants :

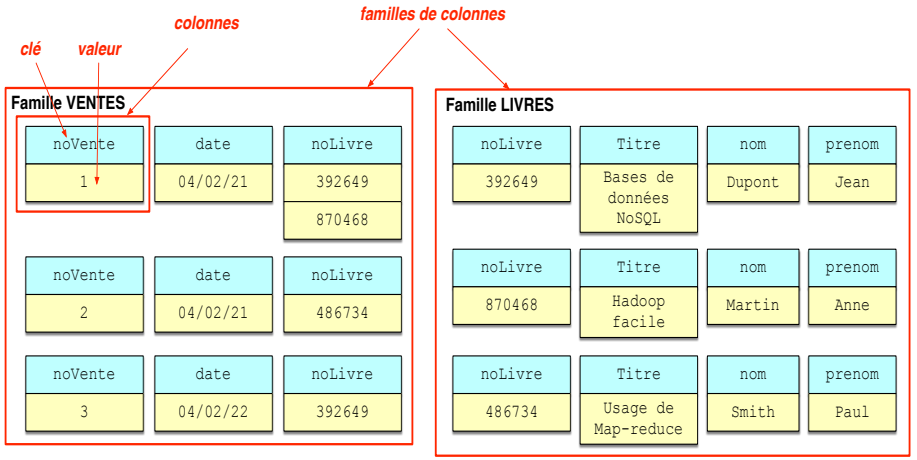
- **Colonne :**
  - entité de base représentant un **champ de donnée**
  - chaque colonne est **définie par un couple clé / valeur**
  - une colonne contenant d'autres colonnes est nommée ***super-colonne***.
- **Famille de colonnes :**
  - permettent de **regrouper plusieurs colonnes** (ou super-colonnes)
  - les **colonnes sont regroupées par ligne**
  - **chaque ligne est identifiée par un identifiant unique** (assimilées aux tables dans le modèle relationnel) et sont **identifiées par un nom unique**
- **Super-colonnes :**
  - situées dans les familles de colonnes sont **souvent utilisées comme les lignes d'une table de jointure** dans le modèle relationnel.

Remarque : Ne pas confondre avec des BD Relationnelles orientées colonnes, qui sérialisent les données par colonne (e.g., MonetDB, Vertica)

BD NOSQL « orientée colonne » : illustration (1)



BD NOSQL « orientée colonne » : illustration (2)



BD NOSQL « Colonne » (3)

- BD assez **complexes** à appréhender (conception et exploitation)
- Très utilisées pour les **traitements d'analyse de données** et dans les **traitements massifs** (notamment via des opérations de type *MapReduce*).
- Elle offrent **plus de flexibilité** que les BD relationnelles:
  - 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.

BD NOSQL « Colonne » : Forces & faiblesses

Forces :

- **Modèle de données** supportant des **données semi-structurées** (clairsemées)
- **naturellement indexé** (colonnes)
- **bonne mise à l'échelle à l'horizontale**
- **MapReduce** souvent utilisé en **scaling horizontal**,
- on peut voir les résultats de requêtes en temps réel

Faiblesses :

- **A éviter pour des données interconnectées** : si les relations entre les données sont aussi importantes que les données elles-mêmes (comme distance ou calculs de la trajectoire),
- **à éviter pour les lectures de données complexes,**
- **exige de la maintenance** - lors de l'ajout / suppression de colonnes et leur regroupements,
- **les requêtes doivent être pré-écrites** => pas de requêtes ad-hoc définies "à la volée"

BD NOSQL « Colonne » : Utilisations principales

- **BD NoSQL type « Colonne » 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*** (nb élevé d'écritures rapides et analyse de base en temps réel (Cassandra)
  - peuvent être de *bons magasins d'analyse des données semi-structurées*
  - utilisé pour la *journalisation des événements* et pour des *compteurs*
  - …

4. BD NoSQL « orientée documents »

- **Modèles et systèmes**
- **Forces et faiblesses**
- **Usages**

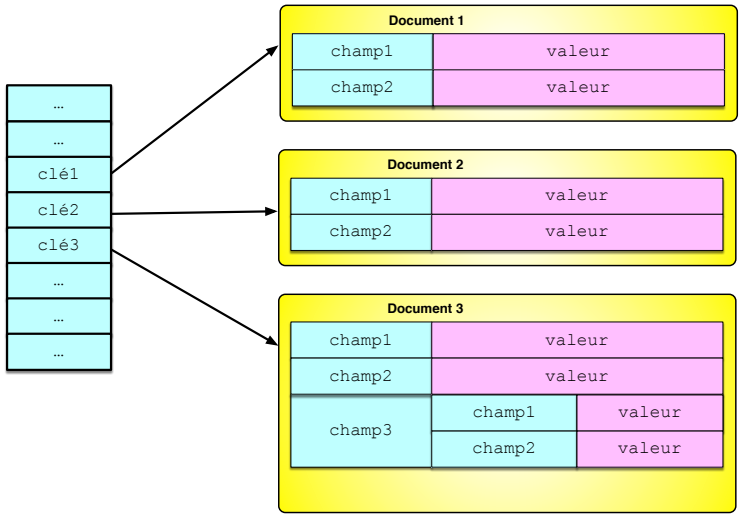
BD NOSQL modèle « Document » (1)

- Elles stockent une collection de "**documents**"
- Elles sont basées sur le modèle « clé-valeur » mais la valeur est un **document** en **format semi-structuré hiérarchique** de type **JSON** ou **XML** (possible aussi de stocker n'importe quel objet, via une sérialisation)
- Les **documents** n'ont **pas de schéma,** mais une **structure arborescente** : ils contiennent une liste de champs, un champ a une valeur qui peut être une liste de champs, ...
- Elles ont généralement une **interface d'accès HTTP REST** permettant d'effectuer des requêtes (plus complexe que l'interface CRUD des BD clés/valeurs)
- Implémentations les plus connues :
  - **CouchDB** (fondation Apache)
  - **RavenDB** (pour plateformes « .NET/Windows » - LINQ)
  - **MongoDB, Terrastore, …**

BD NOSQL « Document » (2)

- Un **document** est composé de **champs** et des **valeurs associées**
- Ces **valeurs** :
  - peuvent être **requêtées**
  - sont soit d'un **type simple** (entier, chaine de caractère, date, ...)
  - soit elles-mêmes **composées** de plusieurs couples clé/valeur.
- Bien que les documents soient structurés, ces BD sont dites "**schemaless**" : il n'est **pas nécessaire de définir au préalable les champs** utilisés dans un document.
- Les documents peuvent être très **hétérogènes** au sein de la BD.
- Permettent d'**effectuer des requêtes sur le contenu** des documents/objets : pas possible avec les BD clés/valeurs simples.
- Elles sont principalement utilisées dans le **développement de CMS** (Content Management System - outils de gestion de contenus).

BD NOSQL « orientée documents : illustration (1)



BD NOSQL « orientée documents : illustration (2)

Collection VENTES :

OID	
5d5gj6ksrg8b45	"novente" : 1 "date" : "04/02/21" "livres" : ["noLivre" : 392649 "noLivre" : 870468]
8gd5gty6u6a34	"novente" : 2 "date" : "04/02/21" "livres" : ["noLivre" : 486734]

Collection LIVRES :

OID	

8jfr56gh5837h7f	"nolivre" : 392649
	"titre" : "Bases de données NoSQL"
	"auteur" : {
	"nom" : Dupont
	"prenom" : Jean
	}

3j67hyr67De67	"nolivre" : 870468
	"titre" : "Hadoop facile"
	"auteur" : {
	"nom" : Martin
	"prenom" : Anne
	}

8hjt7hyr67fg67	"nolivre" : 486734
	"titre" : "Usage de MapReduce"
	"auteur" : {
	"nom" : Schmit
	"prenom" : Paul
	}

BD NOSQL «orientée document » avec imbrication

Collection VENTES :

OID	

5d5gj6ksrg8b45	"novente" : 1
	"date" : "04/02/21"
	"livres" : [
	"noLivre" : 392649
	"titre" : "Bases de données NoSQL"
	"auteur" : {
	"nom" : Dupont
	"prenom" : Jean
	}
	"noLivre" : 870468
	"titre" : "Hadoop facile"
	"auteur" : {
	"nom" : Martin
	"prenom" : Anne
	}
	}
	]

8gd5gty6u6a34	"novente" : 2
	"date" : "04/02/21"
	"livres" : [
	{
	"noLivre" : 486734
	"titre" : "Usage de MapReduce"
	"auteur" : {
	"nom" : Schmit
	"prenom" : Paul
	}
	}
	]

BD NOSQL « Document » : Forces & faiblesses

Forces :

- **Modèle de données simple mais puissant** (expression de structures imbriquées)
- **Bonne mise à l'échelle** (surtout si sharding pris en charge)
- **Pas de maintenance de la BD requise** pour ajouter/supprimer des «colonnes»
- **Forte expressivité de requêtage** (requêtes assez complexes sur des structures imbriquées)

Faiblesses :

- **Inadaptée pour les données interconnectées**
- **Modèle de requête limitée à des clés** (et indexes)
- Peut alors être **lent pour les grandes requêtes** (avec MapReduce)

BD NOSQL « Document » : Utilisations principales

Les BD NoSQL type « Document » principalement utilisées pour :

- Enregistrement d'événements
- Systèmes de gestion de contenu
- Web analytique ou analytique temps-réel
- Catalogue de produits
- …

5. BD NoSQL « orientées graphes »

- **Modèles et systèmes**
- **Forces et faiblesses**
- **Usages**

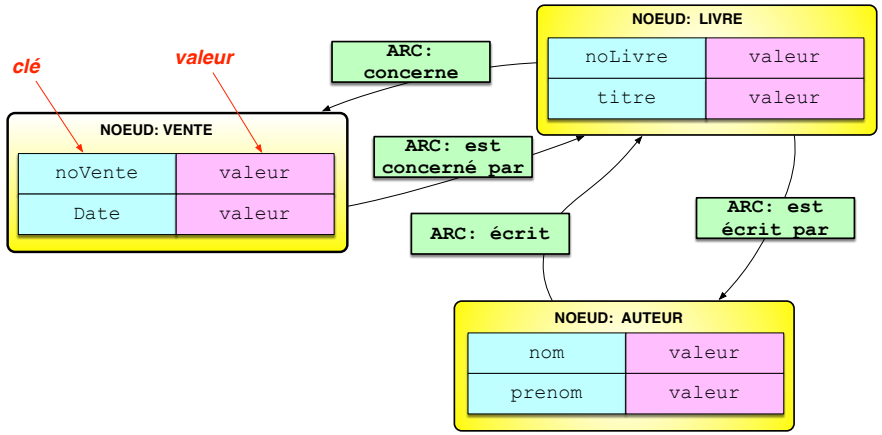
BD NOSQL modèle « Graphe » (1)

- Elles permettent la **modélisation**, le **stockage** et la **manipulation** de **données complexes liées par des relations** non-triviales ou variables
- Modèle de représentation des données basé sur la **théorie des graphes**
- S'appuie sur les notions de ***noeuds***, de ***relations*** et de ***propriétés*** qui leur sont rattachées.
- Implémentations les plus connues :
  - **Neo4J**
  - **OrientDB** (fondation Apache)
  - **…**

BD NOSQL « Graphe » (2)

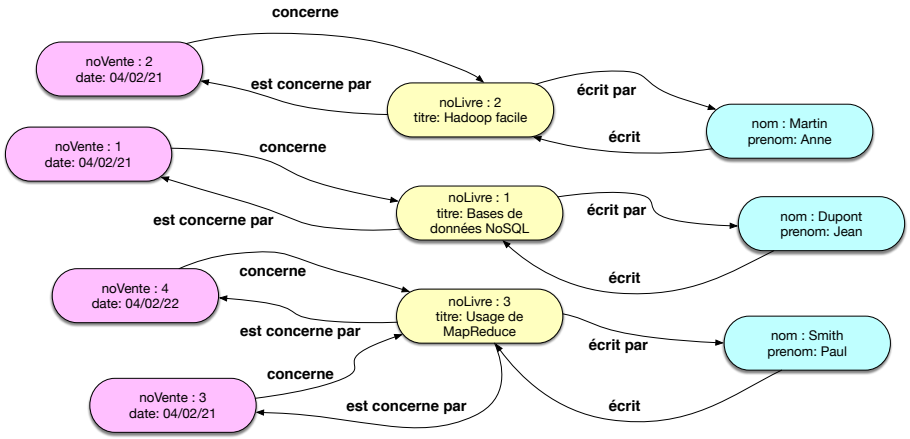
- **Elles utilisent :**
  - un **moteur de stockage** pour les **objets** (similaire à une base documentaire, chaque entité de cette base étant nommée nœud)
  - un **mécanisme de description d'arcs** (relations entre les objets), arcs orientés et avec propriétés (nom, date, ...)
- Elles sont **bien plus efficaces que les BDR** pour traiter les problématiques liées aux réseaux (cartographie, relations entre personnes, ...)
- Elles sont adaptées à la manipulation **d'objets complexes organisés en réseaux** : **cartographie**, **réseaux sociaux**, ..

BD NOSQL « Graphe » : illustration (1)



BD NOSQL « orientée graphe » : illustration (2)

Avec notre illustration, on a par exemple le graphe suivant :



Rappel : un graphe peut être représenté sous forme d'une collection de triplets …

BD NOSQL « orientée graphe » : illustration (3)

Classe VENTE :

OID		
3d5gj6ksrg8b50	*property*	novente : 1
	property	date : "04/02/21"
	relation	livre : 2ft74hj6ksrg834
	relation	livre : 7ht87hte623b56
9gd5gty6u6a56	*property*	novente : 2
	property	date : "04/02/21"
	relation	livre : 37kp87hte623b4
85d5gty6u6a56	*property*	novente : 3
	property	date : "04/02/21"
	relation	livre : 37kp87hte623b4
235qgty6u6a56	*property*	novente : 4
	property	date : "04/02/21"
	relation	livre : 2ft74hj6ksrg834

Classe LIVRE :

OID		
2ft74hj6ksrg834	*property*	titre : Bases de données NoSQL
	relation	auteur : 87d4hj6ksrg834
7ht87hte623b56	*property*	titre : Hadoop facile
	relation	auteur : 37b87hte623b57
37kp87hte623b4	*property*	titre : Usage de MapReduce
	relation	auteur : 87d4hj6ksrg834

Classe AUTEUR :

OID		
87d4hj6ksrg834	*property*	nom : Dupont
	relation	prenom : Jean
37b87hte623b57	*property*	nom : Martin
	relation	prenom : Anne
67kp87hte623b5	*property*	nom : Schmit
	relation	prenom : Paul

BD NOSQL « Graphe » : Forces & faiblesses

Forces :

- **Modèle de données puissant**
- Rapide pour les **données liées**, bien **plus rapide que SGBDR**
- **Modèles d'interrogation (langages) bien établis et performants** : notamment SPARQL (Web Sémantique) et Cypher

Faiblesses :

- **Fragmentation** (sharding) :
  - Plus délicate
  - Cependant parfois fractionnement possible.

BD NOSQL « Graphe » : Utilisations principales

BD NoSQL type « Graphe » principalement utilisées pour :

- Moteurs de recommandation
- Business Intelligence (BI)
- Web Sémantique
- Social computing
- Données géospatiales
- Généalogie
- Web of Things (IoT)
- Catalogue de produits
- Sciences de la Vie et calcul scientifique (bio-informatique, …)
- Données liées, données hiérarchiques
- Services de routage, d'expédition et de géolocalisation
- Services financiers : chaîne de financement, dépendances, gestion des risques, détection des fraudes, …

BD NoSQL « Graphe » et Web Sémantique

Graphe = ensemble de triplets

Magasins de triplets RDF (Triple Stores) : ontologie, base de connaissances - Web Sémantique

- Triplet = arête du graphe = «**nœud-lien-noeud**» (sujet-prédicat-objet)
- Possibilité de **joindre** des graphes ensemble automatiquement en faisant correspondre les identifiants des nœuds
- Possibilité de **fusion automatique** de 2 graphes

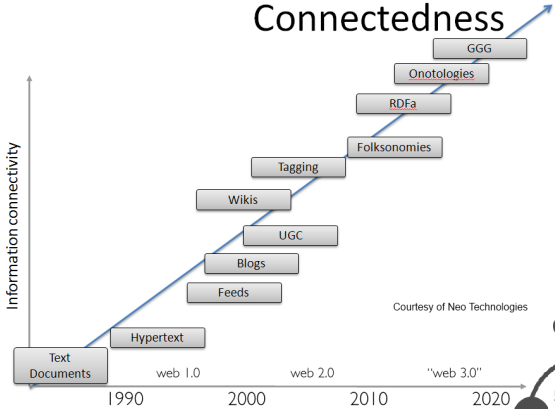
  Ex: le graphe 1 a le noeud A relié à B et le graphe 2 le nœud B relié à C, l'union de ces graphes montre une relation de A à C.
- Les **données RDF** interrogées via le protocole/langage de requête **SPARQL** normalisé permettant l'inférence *(Groupe W3C RDF Data Access de travail)*

Exemple de Triple Stores: Virtuoso, Sesame, Jena, …

BD NoSQL « Graphe » : le futur ?

- Internet : réseau d'ordinateurs
- Word Wide Web : Web de documents
- (GGG) Giant Global Graph : graph of metadata

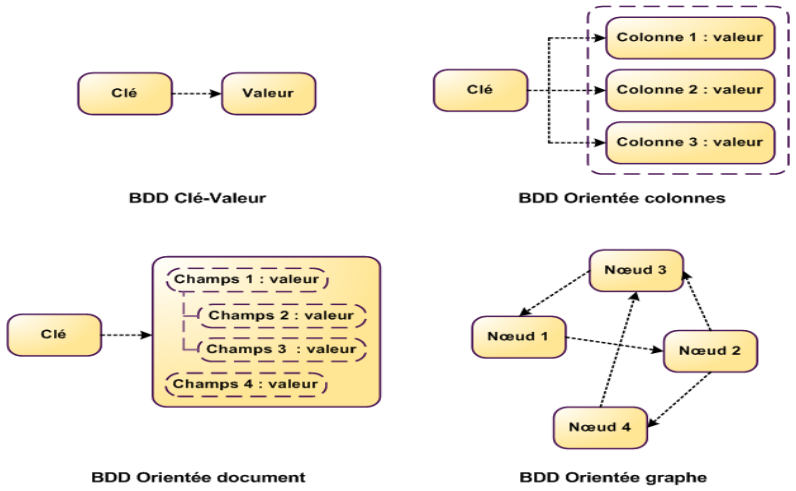
"I called this graph the Semantic Web, but maybe it should have been Giant Global Graph." - Tim Berners-Lee – 2007



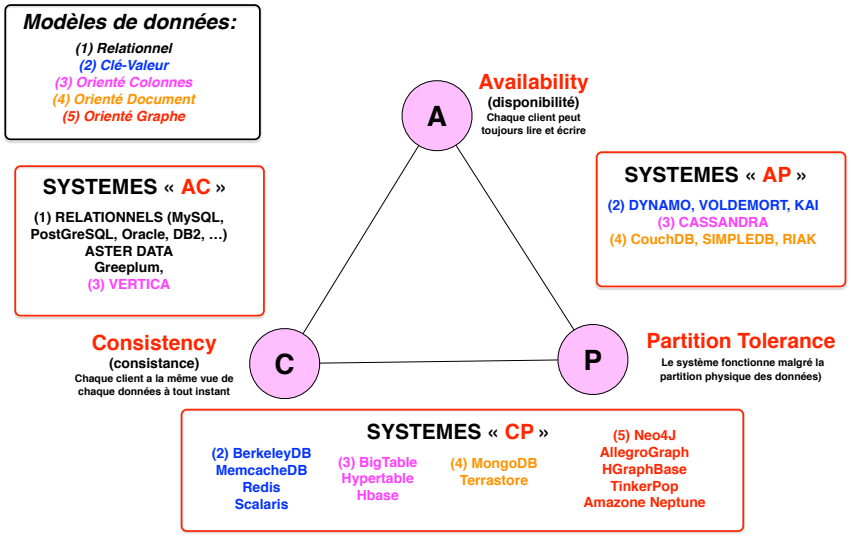
6. Conclusion

- **Résumé des 4 types de BD NoSQL**
- **Systèmes NoSQL et Théorème de CAP**
- **Bilan**

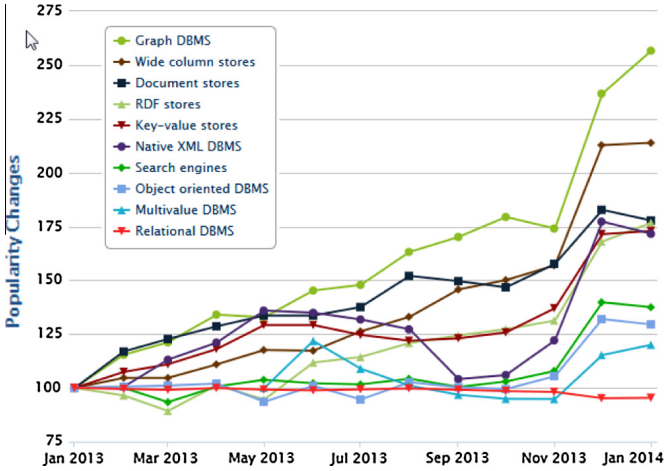
Résumé des 4 types de BD NoSQL :



Systèmes NoSQL et Théorème de CAP



Popularité des bases de données NoSQL



Source : Databases popularity trend (M. Gelbmann, Graph DBMSs are gaining in popularity faster than any other database category, 2014.)

Bilan sur les BD NoSQL

Conception, modélisation d'une BD NoSQL :

Encore délicate :

- Pas de tables, seulement des paires de « clés-valeurs »
- Tous les accès se font avec une clé …
- On peut ajouter un attribut/colonne à tout moment
- Une « valeur » peut être un objet complexe (liste, document, ens. de valeurs…)

⇨ *Pas encore de modèles et méthodologie de conception …*

Problèmes liés aux BD NoSQL

- **Complexité des traitements** : pas de langage puissant de requêtage et d'exploitation comme SQL, mais des langages propriétaire (sauf BD graphes : *Sparql*, *Cypher*)
- **Relâchement de la Cohérence** (ACID) :
  - permet un grain de performances et un passage à l'échelle facilité
  - mais peut être critique pour certaines applications (ex : opérations financières) et alors nécessiter le développement de couches logicielles supplémentaires
- **Technologie peu familière** et une **communauté encore réduite**
- **Beaucoup de solution open-source** : encore très peu de support client.

⇨ ***Vers le NewSQL ?***

Du NoSQL au NewSQL ?

NewSQL :

- **Pallier les limitations du NoSQL et réconcilier les mondes SQL et NoSQL**
- **Nouvelle architecture de stockage des données en émergeance**
- **Devraient permettre :**
  - une **interrogation des données via SQL**
  - tout en garantissant des performances et un **passage à l'échelle** similaires aux bases de données NoSQL.
  - tout en **conservent les propriétés ACID**
- **2 convergences distinctes :**
  - des éditeurs de SGBD NoSQL se tournant vers le relationnel (Ex : MemSQL)
  - des éditeurs de SGBD Relationnels traditionnels intégrant certains des concepts du NoSQL (Ex : Microsoft SQL Server, PostGreSQL, …).
- **Systèmes NewSQL :**
  - Clustrix23, MemSQL, NuoDB, …