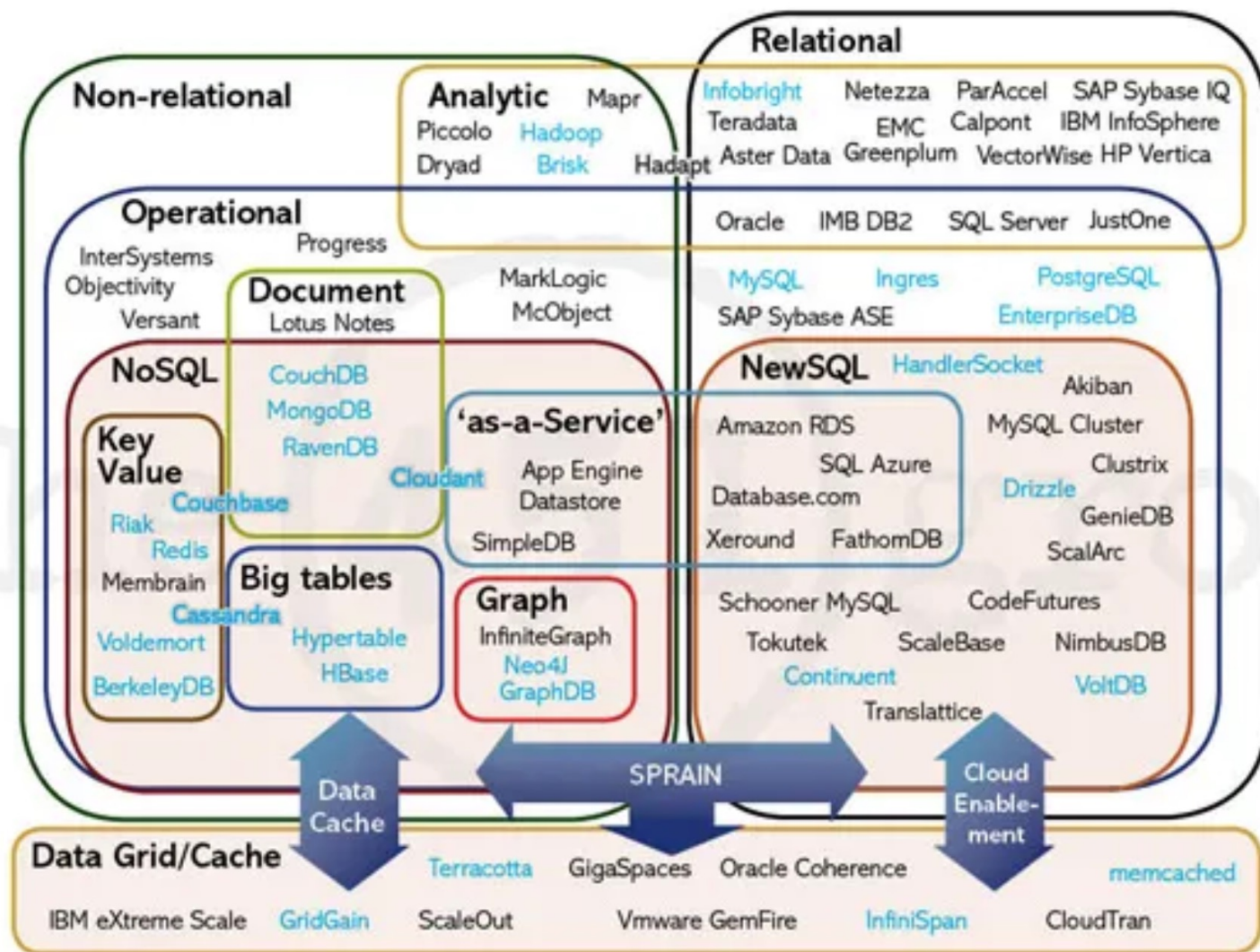


Base de données

Chapitre 8 : Introduction aux bases de données NoSQL

Stéphane Malandain – sbd – 2024

Généralités



Généralités

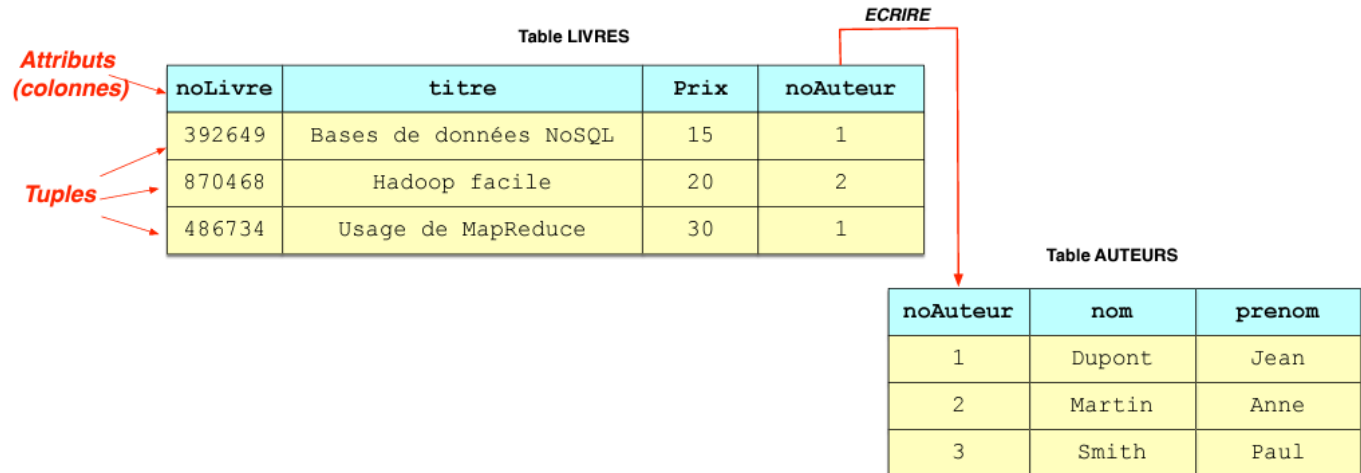
420 systems in ranking, May 2024

Rank			DBMS	Database Model	Score		
May 2024	Apr 2024	May 2023			May 2024	Apr 2024	May 2023
1.	1.	1.	Oracle +	Relational, Multi-model T	1236.29	+2.02	+3.66
2.	2.	2.	MySQL +	Relational, Multi-model T	1083.74	-3.99	-88.72
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model T	824.29	-5.50	-95.80
4.	4.	4.	PostgreSQL +	Relational, Multi-model T	645.54	+0.49	+27.64
5.	5.	5.	MongoDB +	Document, Multi-model T	421.65	-2.31	-14.96
6.	6.	6.	Redis +	Key-value, Multi-model T	157.80	+1.36	-10.33
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model T	135.35	+0.57	-6.28
8.	8.	↓ 7.	IBM Db2	Relational, Multi-model T	128.46	+0.97	-14.56
9.	9.	↑ 11.	Snowflake +	Relational	121.33	-1.87	+9.61
10.	10.	↓ 9.	SQLite +	Relational	114.32	-1.69	-19.54
11.	11.	↓ 10.	Microsoft Access	Relational	104.92	-0.49	-26.26
12.	12.	12.	Cassandra +	Wide column, Multi-model T	101.89	-1.97	-9.25
13.	13.	13.	MariaDB +	Relational, Multi-model T	93.21	-0.60	-3.66
14.	14.	14.	Splunk	Search engine	86.45	-2.26	-0.19
15.	↑ 17.	↑ 18.	Databricks +	Multi-model T	78.61	+2.28	+14.66
16.	↓ 15.	16.	Microsoft Azure SQL Database	Relational, Multi-model T	77.99	-0.41	-1.21
17.	↓ 16.	↓ 15.	Amazon DynamoDB +	Multi-model T	74.07	-3.50	-7.04
18.	18.	↓ 17.	Hive	Relational	61.17	-1.41	-12.44
19.	19.	↑ 20.	Google BigQuery +	Relational	60.38	-1.52	+5.51
20.	20.	↑ 21.	FileMaker	Relational	48.20	-1.53	-3.80
21.	21.	↓ 19.	Teradata	Relational, Multi-model T	45.33	-2.52	-17.39
22.	22.	↑ 23.	SAP HANA +	Relational, Multi-model T	44.69	-1.14	-5.67
23.	23.	↓ 22.	Neo4j +	Graph	44.46	-0.01	-6.65
24.	24.	24.	Solr	Search engine, Multi-model T	42.91	-1.37	-6.85
25.	25.	25.	SAP Adaptive Server	Relational, Multi-model T	36.31	-0.81	-6.78
26.	26.	26.	HBase	Wide column	30.50	-0.74	-8.09
27.	27.	27.	Microsoft Azure Cosmos DB +	Multi-model T	29.04	-0.81	-6.95
28.	28.	↑ 29.	InfluxDB +	Time Series, Multi-model T	25.83	-0.74	-4.08

SGBD relationnel : caractéristiques

- Généralement transactionnel : propriétés ACID qui implique que toutes les opérations de mise à jour des données (insertion, modification, suppression) sont prises en compte et sérialisées tout en préservant l'intégrité des données
- Basés sur un 'contexte' mathématique (Algèbre relationnel)
- Langage de requêtes standard (SQL)
- Très largement implantées
- Grande maturité (bientôt 50 ans)
- Système de jointure entre les tables permettant de construire des requêtes complexes
- Système d'intégrité référentielle assurant que les liens entre les entités sont valides

Exemple :



SGBD relationnel : le 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 le modèle ACID (Atomicity, Consistency, Isolation, Durability) assurant toujours l'intégrité de la BDD

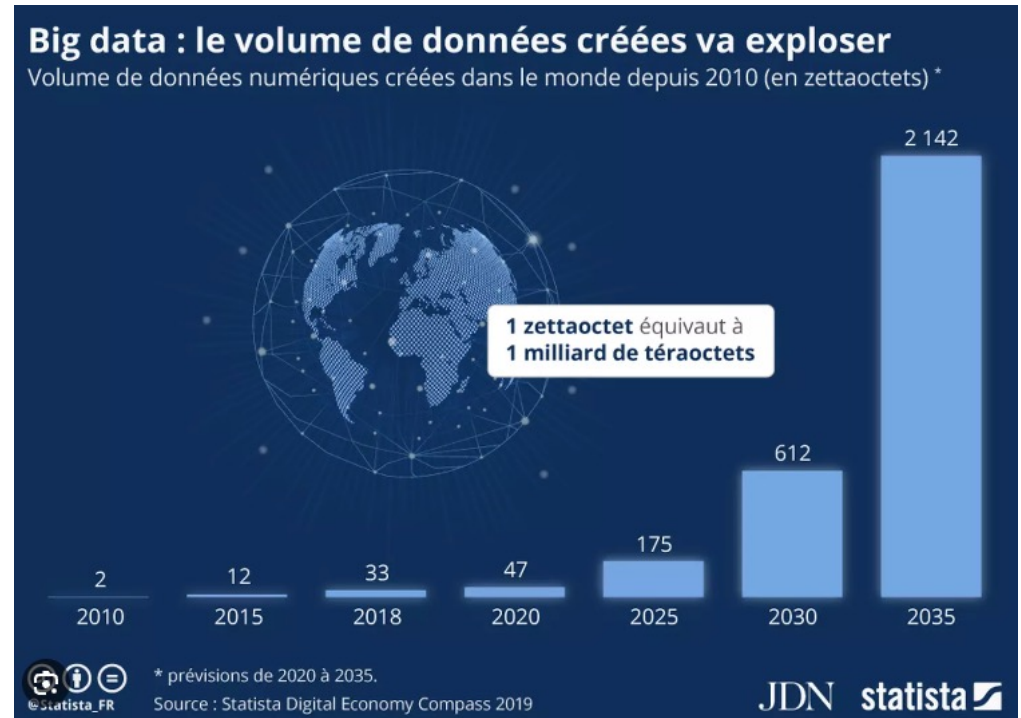
Exemple : le distributeur de billets

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 des données

- Essor des très grandes plateformes et applications Web (Google, Fb, Twitter, LinkedIn, Amazon, etc...) avec des mégas mégas données
- Le volume à gérer par ces applications est gigantesque
- Les données de ces applications sont principalement hétérogènes et peu structurées



SGBD relationnel : limites liées au volume

- Le modèle ACID ralenti considérablement l'ensemble
- Les jointures sont coûteuses et les requêtes peuvent être très complexes
- Il est difficile d'effectuer une mise à l'échelle horizontale
- Moins efficace pour des données non structurées
- ...

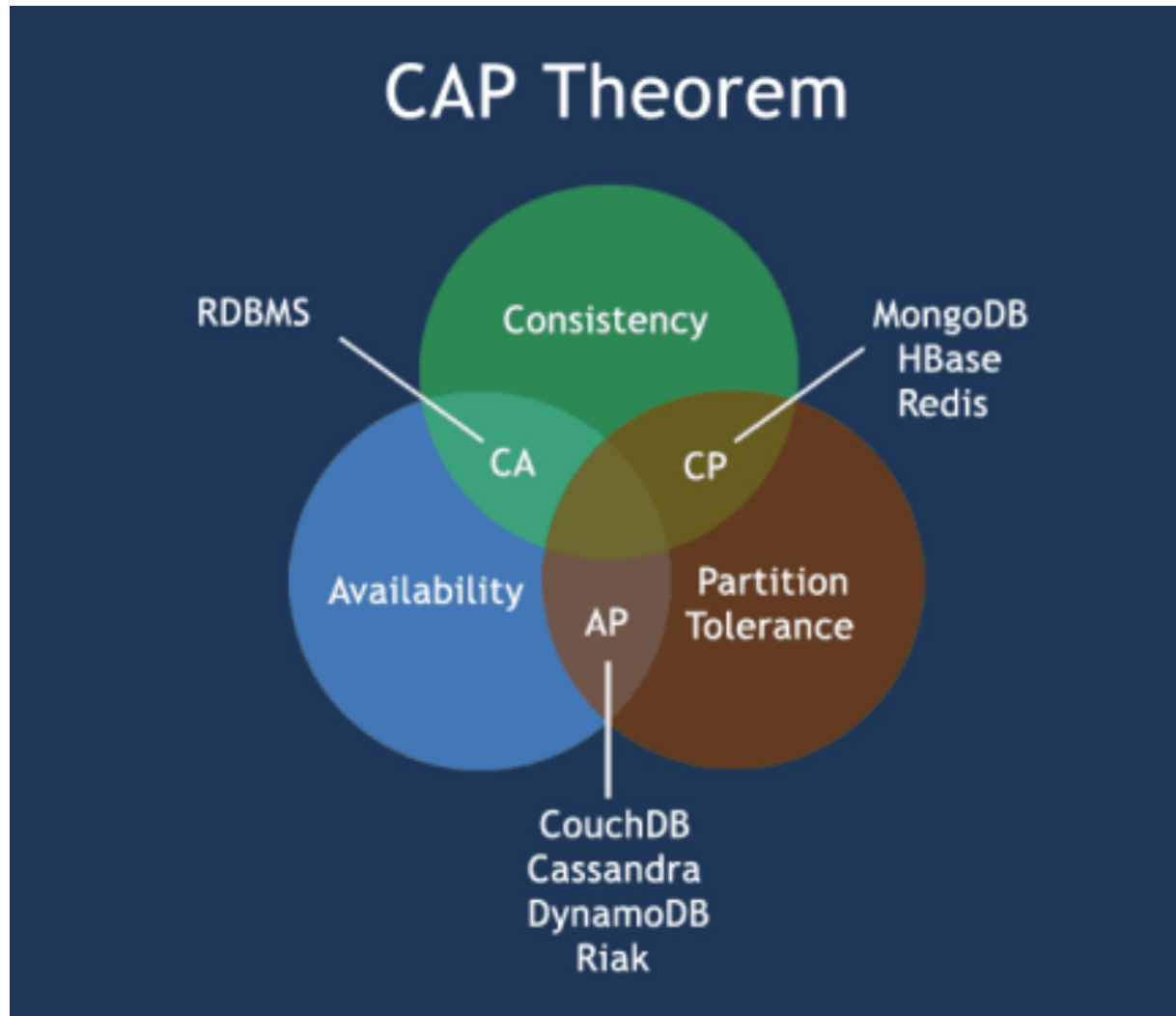
SGBD relationnel : limites liées à l'hétérogénéité des données

- Le modèle relationnel est fortement structuré et rigide
- Les mégadonnées et les applications Web concernent des données souvent peu structurées et très hétérogènes (textes, images)
- Recherche de nouveaux modèles de données de stockage plus **flexibles** (voir pas de modèle)

Limites liées au volume : distribution

- Les mécanismes permettant une gestion des transactions garantissent le maintien des propriétés ACID (Atomicité, Cohérence, Isolation et Durabilité)
- MAIS avec le volume, le contexte fortement distribué implique des mécanismes très coûteux
- Avec la plupart des SGBD relationnels, les données d'un BDD liées entre elles sont placées sur le même nœud du serveur.
- Si le nombre de liens est important, il est de plus en plus difficile de placer les données sur des nœuds différents
- Ceci implique un 'Relâchement' des propriétés ACID (théorème de CAP)

Théorème de CAP



Théorème de CAP

3 propriétés fondamentales pour les systèmes distribués :

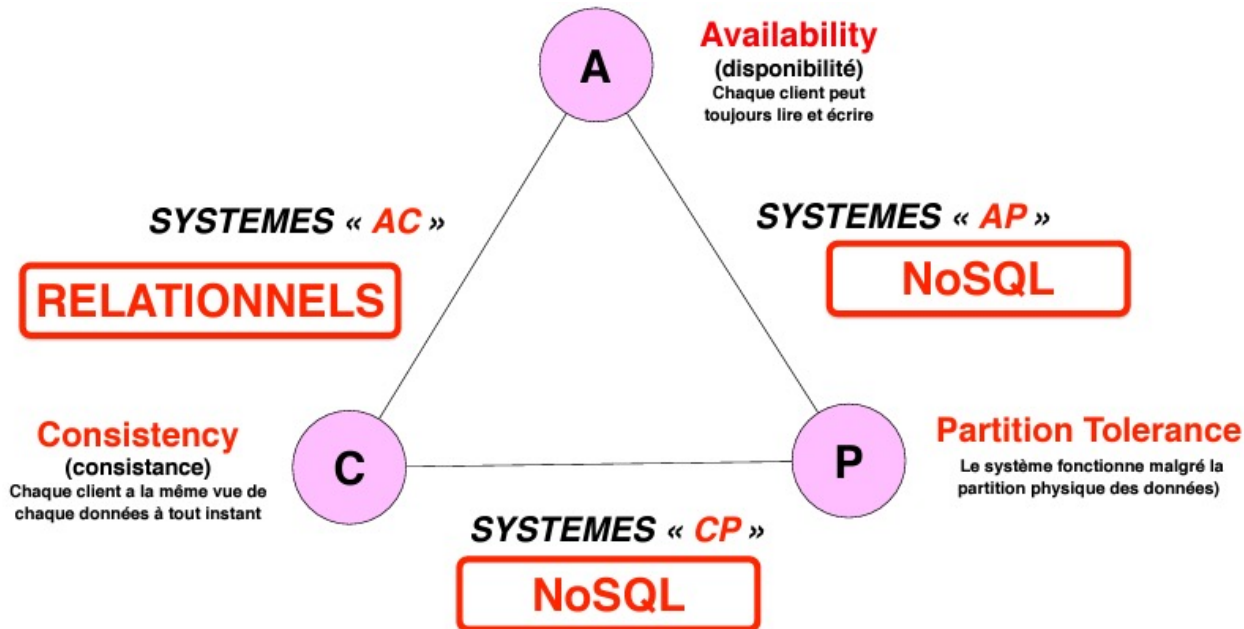
- **C**oherence ou Consistance : tous les nœuds du système voient exactement les mêmes données au même moment
- **A**vailability ou Disponibilité : la perte de nœuds n'empêche pas les survivants de continuer à fonctionner correctement, les données restent accessibles
- **P**artition tolerance ou Résistance 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

Théorème de "CAP" (Brewer, 2000) :

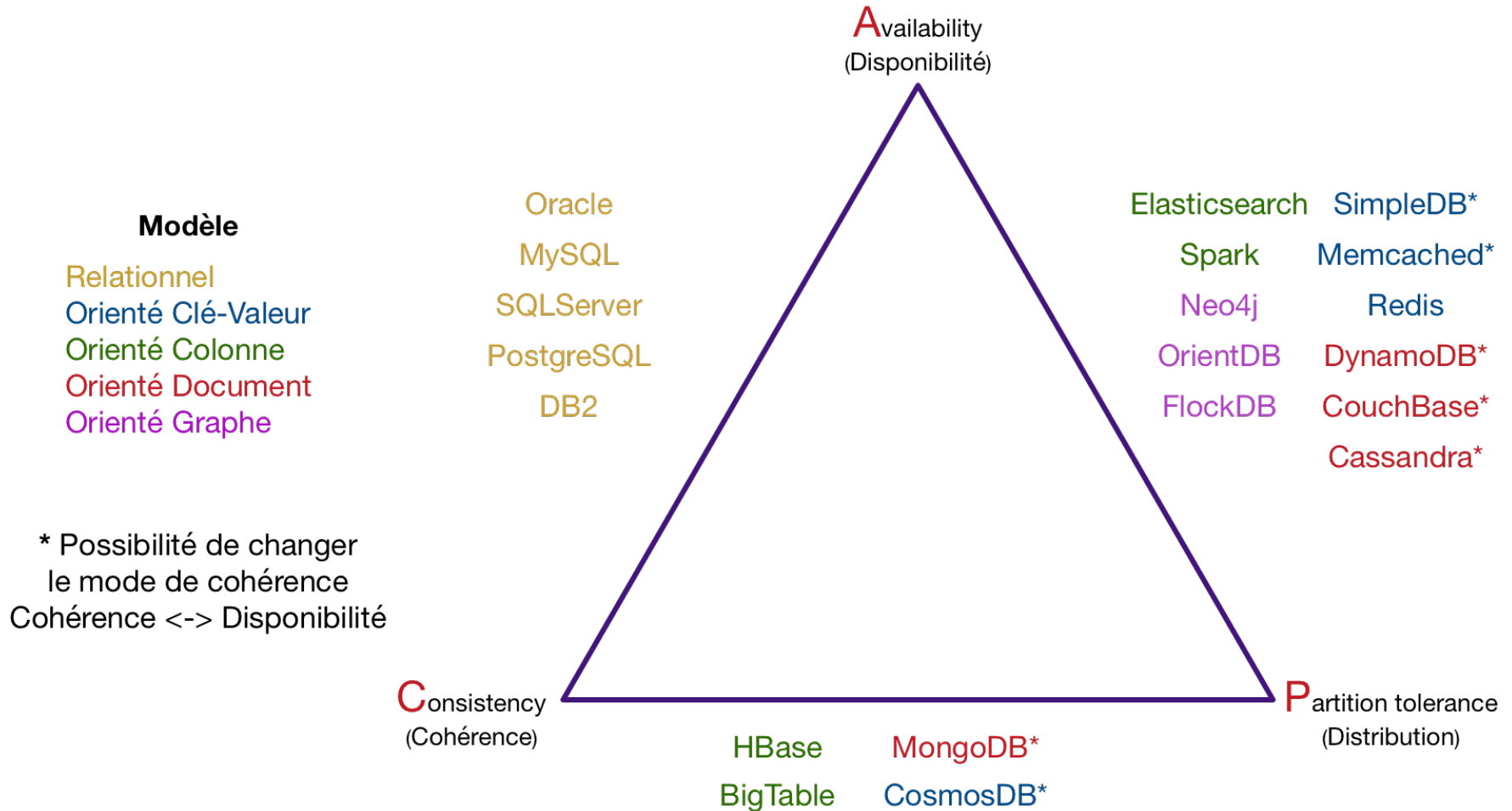
Dans un système distribué, il est impossible d'obtenir ces 3 propriétés en même temps, il faut en choisir 2 parmi les 3

Théorème de CAP



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

Le triangle de CAP et les bases de données



Les nouvelles solutions : BD NoSQL

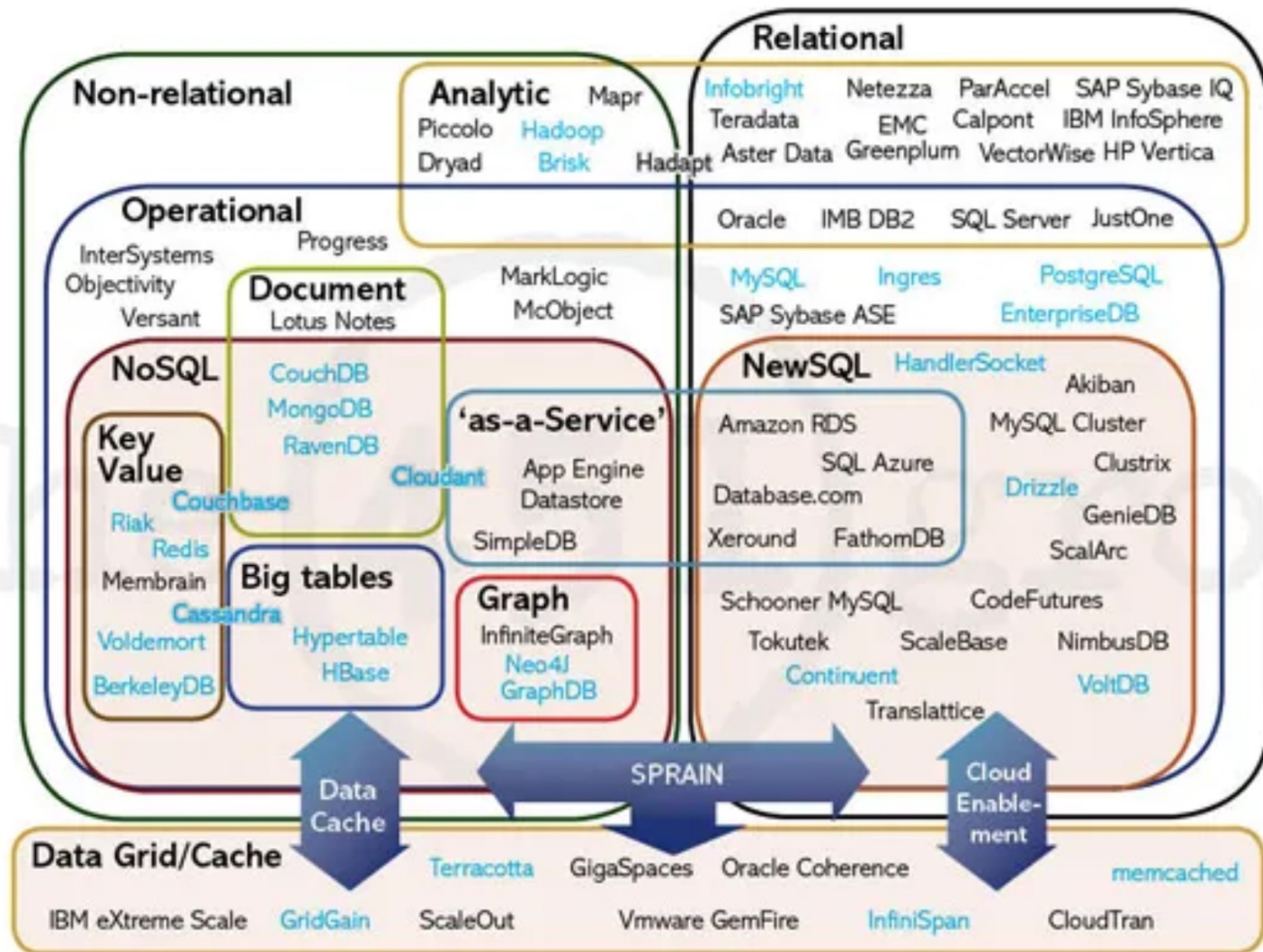
- Face aux limites des BD relationnelles de nouvelles solutions sont apparues. Elles permettent :
 - Une meilleure scalabilité dans des contextes fortement distribués (cluster de serveurs, data centers)
 - Une gestion d'objets complexes et hétérogènes sans avoir à déclarer au préalable l'ensemble des champs représentant un objet (modèles de données flexibles)
- Ces nouvelles solutions sont regroupées derrière le terme NoSQL (Carl Strozzi)
- Ces solutions ne remplacent pas les SGBD Relationnels traditionnels mais sont complémentaires et comblent leurs faiblesses dans certains contextes d'applicatifs spécifiques (notamment mégadonnées et applications Web)

Les nouvelles approches

Face aux limites des SBD relationnels différentes solutions à architectures distribuées émergent :

- **NoSQL SBD** : SBD avec schéma dynamique (voir même sans schéma), SBD clé-valeur, SBD de documents, SBD de graphes de donnée etc. . Ces solutions existent déjà.
- **NewSQL BD** : Amélioration des performances des SBD relationnels grâce à de nouveaux moteurs de stockage, de nouvelles technologies de fragmentation et de nouveaux logiciels et matériels. Ces solutions relèvent encore de la Ra&d.
- **Data Grid/Cache Products** : Amélioration des performances des application et le la BD par stockage des données en mémoire : Données persistantes en cache, replaication des données distribuées, calcul exploitant le Grid, etc. . Ces solutions relèvent encore de la Ra&d.

Le grand paysage des bases de données



SBD NoSQL : Du modèle ACID vers le modèle BASE

Du modèle ACID vers le modèle BASE

Thérème de CAP : NoSQL = Systèmes AP ou CP

- Notions de transaction généralement absente.
- Priorité de la disponibilité sur la cohérence.

Modèle BASE

- **Basically Available** : le système garantit la **disponibilité** des données et répondra à toute demande
- **Soft State** : L'état du système peut changer au fil du temps. La base NoSQL n'a pas à être cohérente à tout instant.
- **Eventually consistent** : à terme, la base atteindra un état cohérent, les données se propageront partout où elles devraient se trouver.



Du modèle ACID vers le modèle BASE : Un exemple

Exemple : Le 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 PAS une transaction ACID

Soit :

- Les opérations de lecture/écriture sont disponibles autant que possible sur tous les nœuds
- Les réponses approximatives deviennent, au fil du temps, de plus en plus cohérentes
- Au final, le système converge vers des valeurs 100 % cohérentes.

Modèle de données NoSQL

Instance de TABLE (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, „productName”: „iPod Touch”
    },
    {
      „productId”:1020, „price”: 199.55, „productName”: „Monster Beat”
    }
  ],
  „shippingAddress”: [{ „street”:„Chaumontweg”, „city”:„Spiegel”, „postCode”:„3095”}]
}
```

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

Caractéristiques des BD NoSQL

Modèle :

- Non relationnel, pas de schéma pour les données (parfois 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 nœuds.

Propriétés et usages :

- Privilégient la Disponibilité (A) à la Cohérence (C) (théorème de CAP)
- Modèle "BASE"
- Pas de gestion de transactions
- Mode d'utilisation : Peu d'écritures, beaucoup de lectures

Hadoop et MapReduce

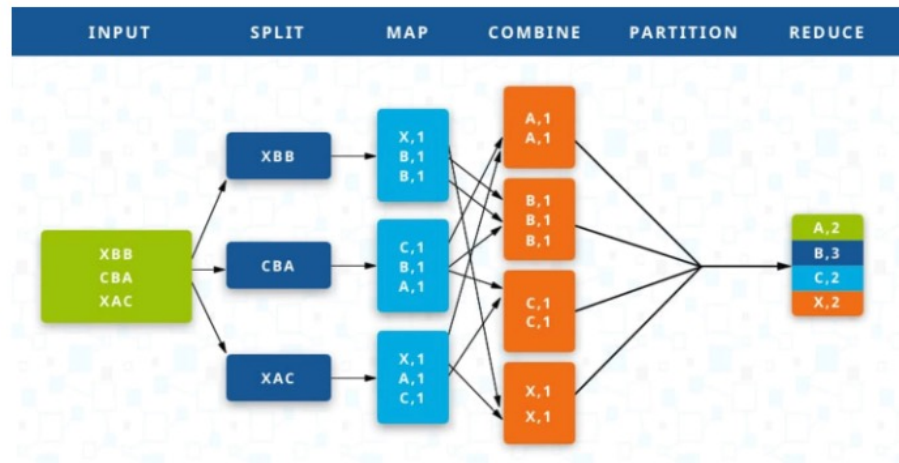
MapReduce est un modèle (ou structure) de programmation disponible dans les environnements [Hadoop](#) qui est utilisé pour accéder aux big data stockées dans le Hadoop File System (HDFS). MapReduce est un élément essentiel et fait partie intégrante du fonctionnement de l'environnement Hadoop.

MapReduce facilite les traitements concurrents en divisant les péta-octets de données en volumes plus petits et en les traitant en parallèle sur des serveurs standard dédiés à Hadoop. Pour résumer, MapReduce agrège les données de plusieurs serveurs et renvoie un résultat consolidé à l'application.

MapReduce – Principes de base

Au cœur de MapReduce se trouvent deux fonctions, Map et Reduce, qui sont séquencées l'une après l'autre.

- La fonction **Map** transforme les entrées du disque en paires <key,value>, les traite et génère un autre ensemble de paires <key,value> intermédiaires en sortie.
- La fonction **Reduce** transforme également les entrées en paires <key,value> et génère une des paires <key,value> en sortie.

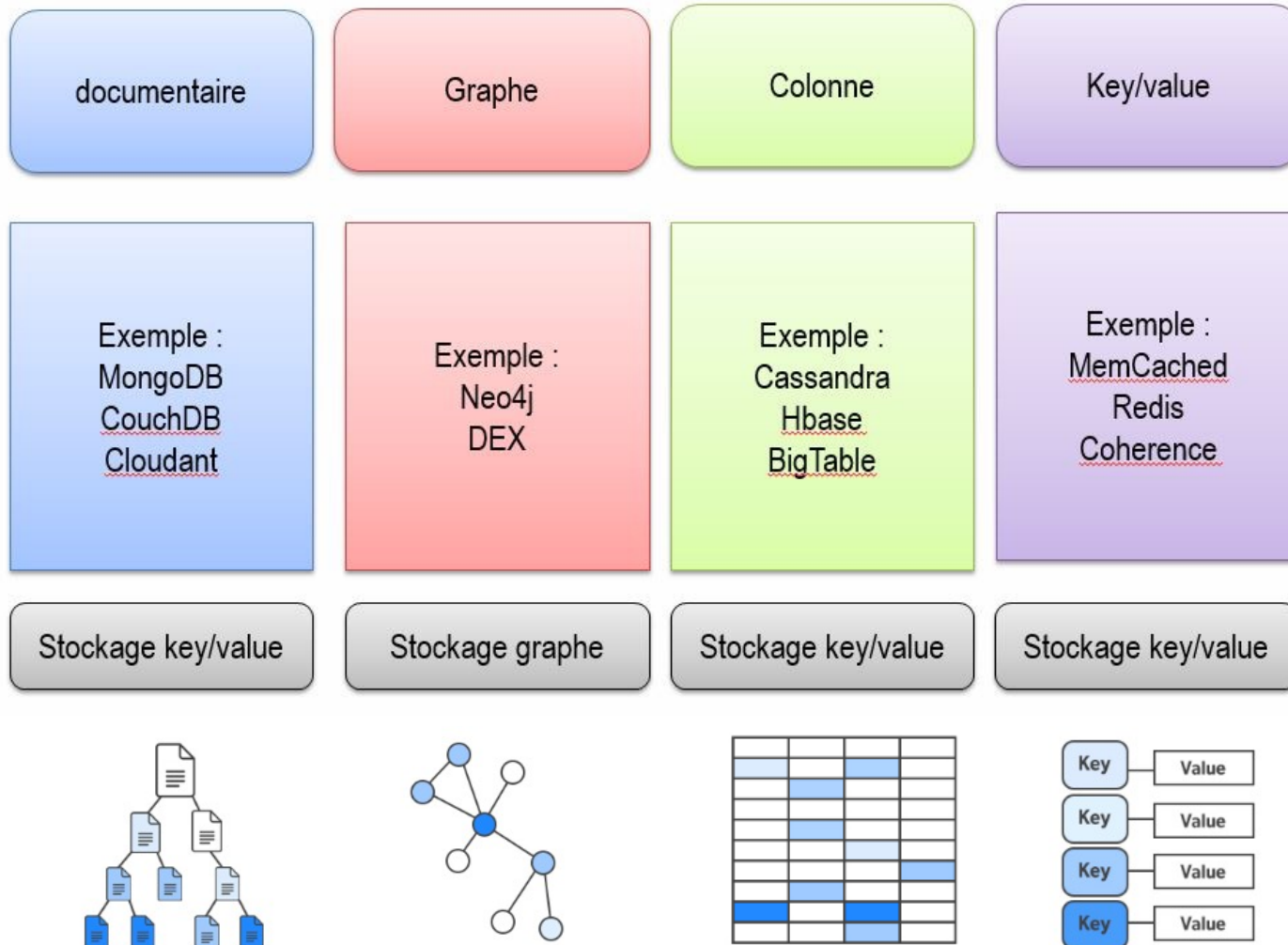


Les différents types de BD NoSQL

Le but est de stocker les informations de la façon la mieux adaptée à leur représentation. Nous avons ainsi 4 types de BD NoSQL :

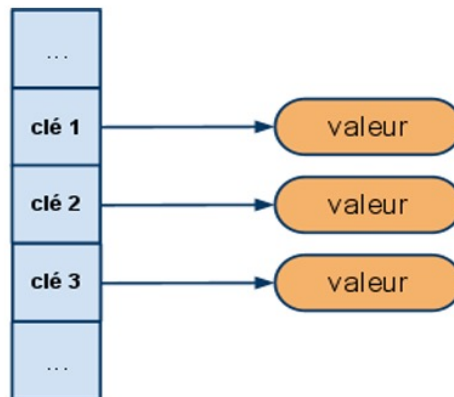
- **Type Clé-Valeur (Key-Value)** : Basique, chaque objet est identifié par une clé unique qui constitue la seule manière de requêter.
- **Type Colonne (Column)** : Permet de disposer d'un très grand nombre de valeurs sur une même ligne, de stocker des relations « one-to-many », d'effectuer des requêtes par clé (adaptés au stockage de listes : messages, posts, commentaires, ...)
- **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)
- **Type Graphe** : pour gérer des relations multiples entre les objets (adaptés aux données issues de réseaux sociaux, ...)

Les différents types de BD NoSQL



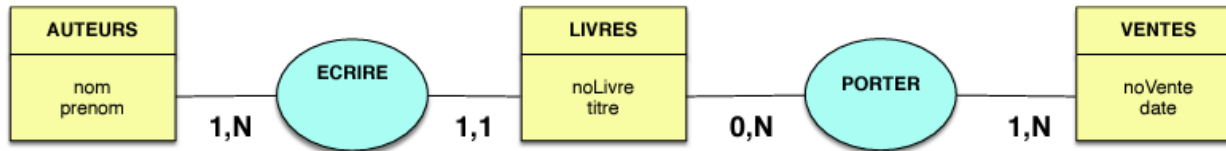
BD NoSQL modèle "Clé-Valeur"

- 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 implique que toute l'intelligence portée auparavant par les requêtes SQL devra être portée par l'applicatif qui interroge la BD.
- Implémentations connues : *Amazon Dynamo*, *Redis* (VMWARE), *Voldemort* (LinkedIn)
- 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

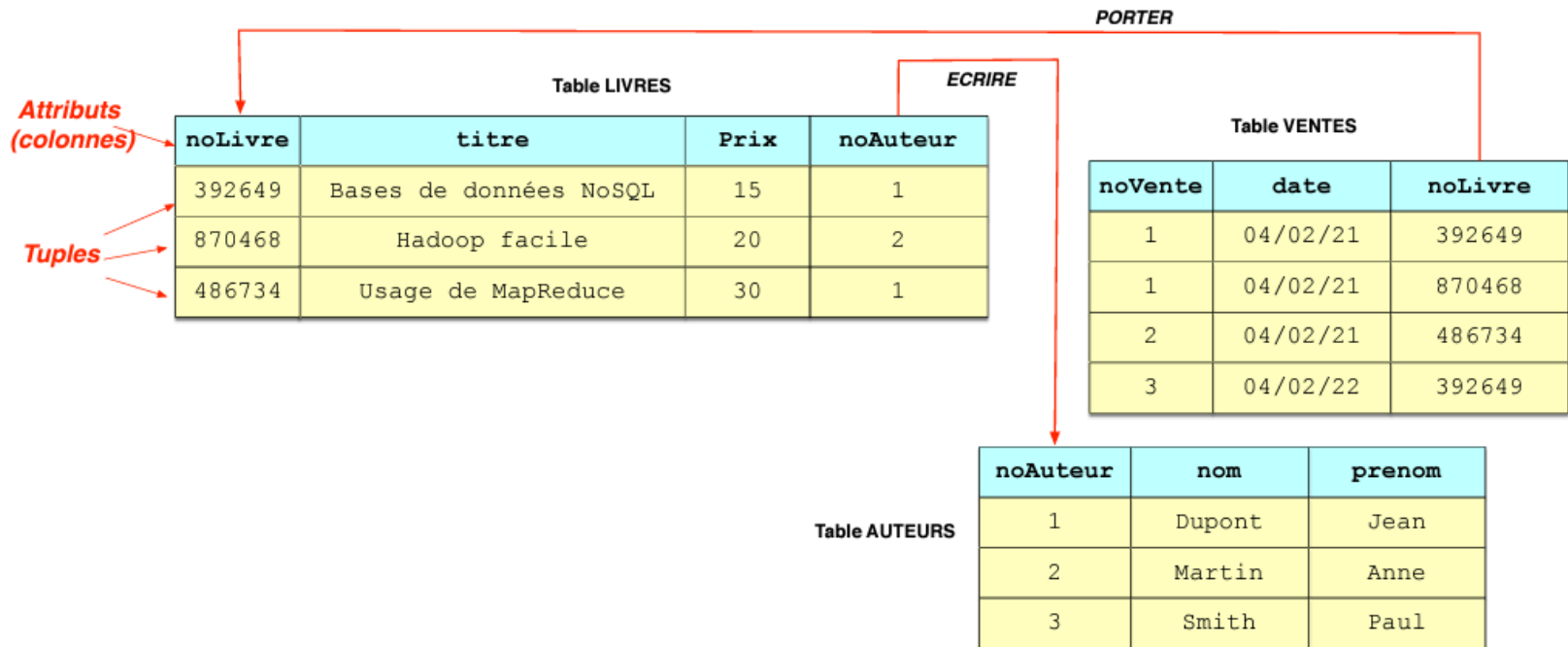


BD NoSQL modèle "Clé-Valeur"

MCD (Entité-Relation) :



Une extension de la BD relationnelle associée :



BD NoSQL modèle "Clé-Valeur"

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 modèle "Clé-Valeur"

Forces :

- Modèle de données simple
- Bonne mise à l'échelle horizontale

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

BD NoSQL modèle "Clé-Valeur"

Utilisations principales :

- 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 capteurs
- Les logs
- ...

BD NoSQL "orientées colonnes"

- 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), *Cassandra* (Apache Foundation), *SimpleDB* (Amazon)

BD NoSQL "orientées colonnes"

Les principaux concepts associés sont :

- 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.

BD NoSQL "orientées colonnes"

Famille de colonnes

Colonne

champ1	valeur
champ2	valeur

Super colonne

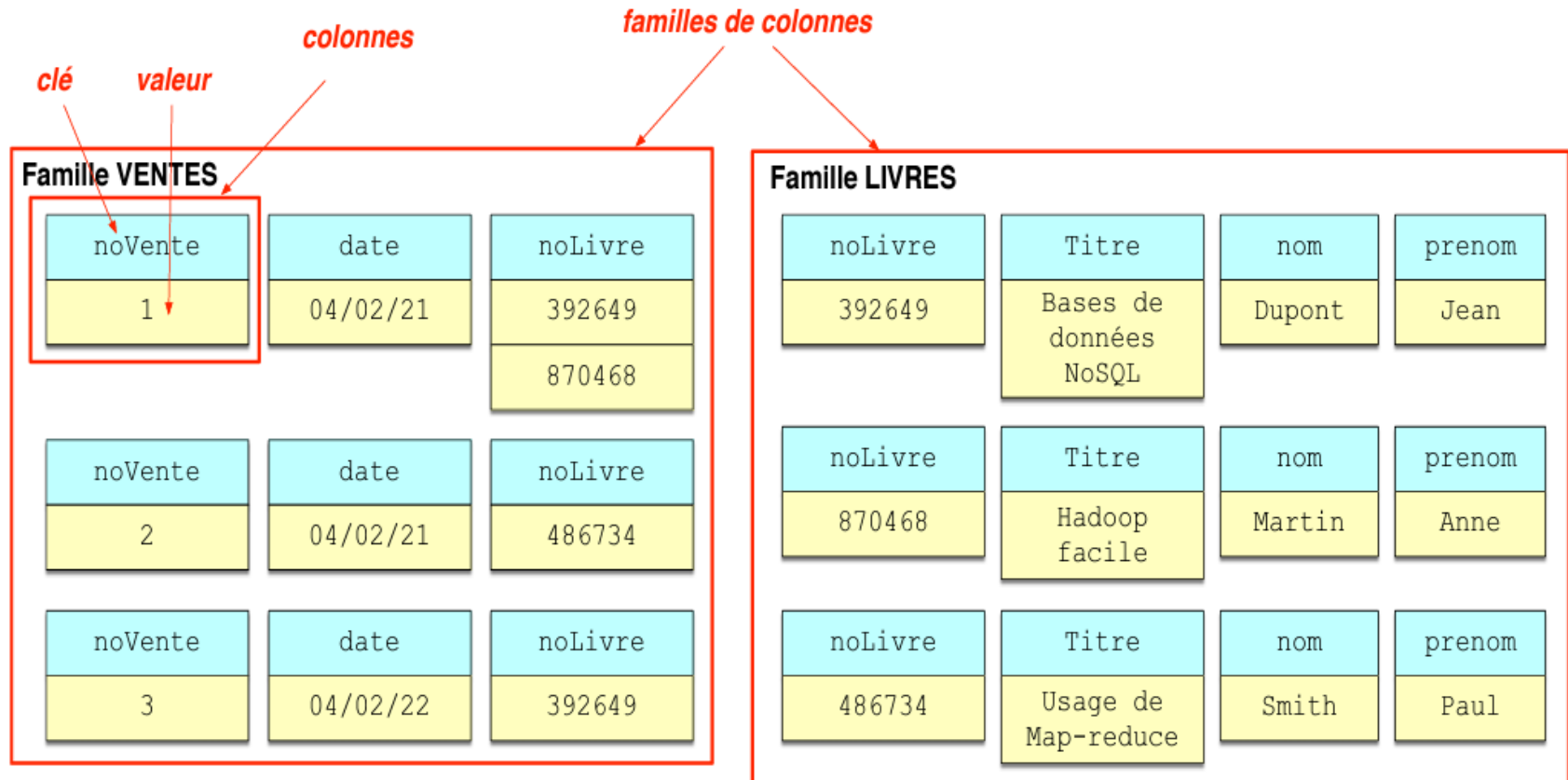
Colonne

champ1	valeur
champ2	valeur

Colonne

champ1	valeur
champ2	valeur

BD NoSQL "orientées colonnes"



BD NoSQL "orientées colonnes"

- 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 "orientées colonnes"

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 leurs regroupements,
- les requêtes doivent être pré-écrites => pas de requêtes ad-hoc définies "à la volée"

BD NoSQL "orientées colonnes"

Utilisations principales :

- Analyse de données, traitement analytique en ligne (OnLine Analytical Processing (OLAP))
- 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))
- utilisé pour la journalisation des événements et pour des compteurs
- ...

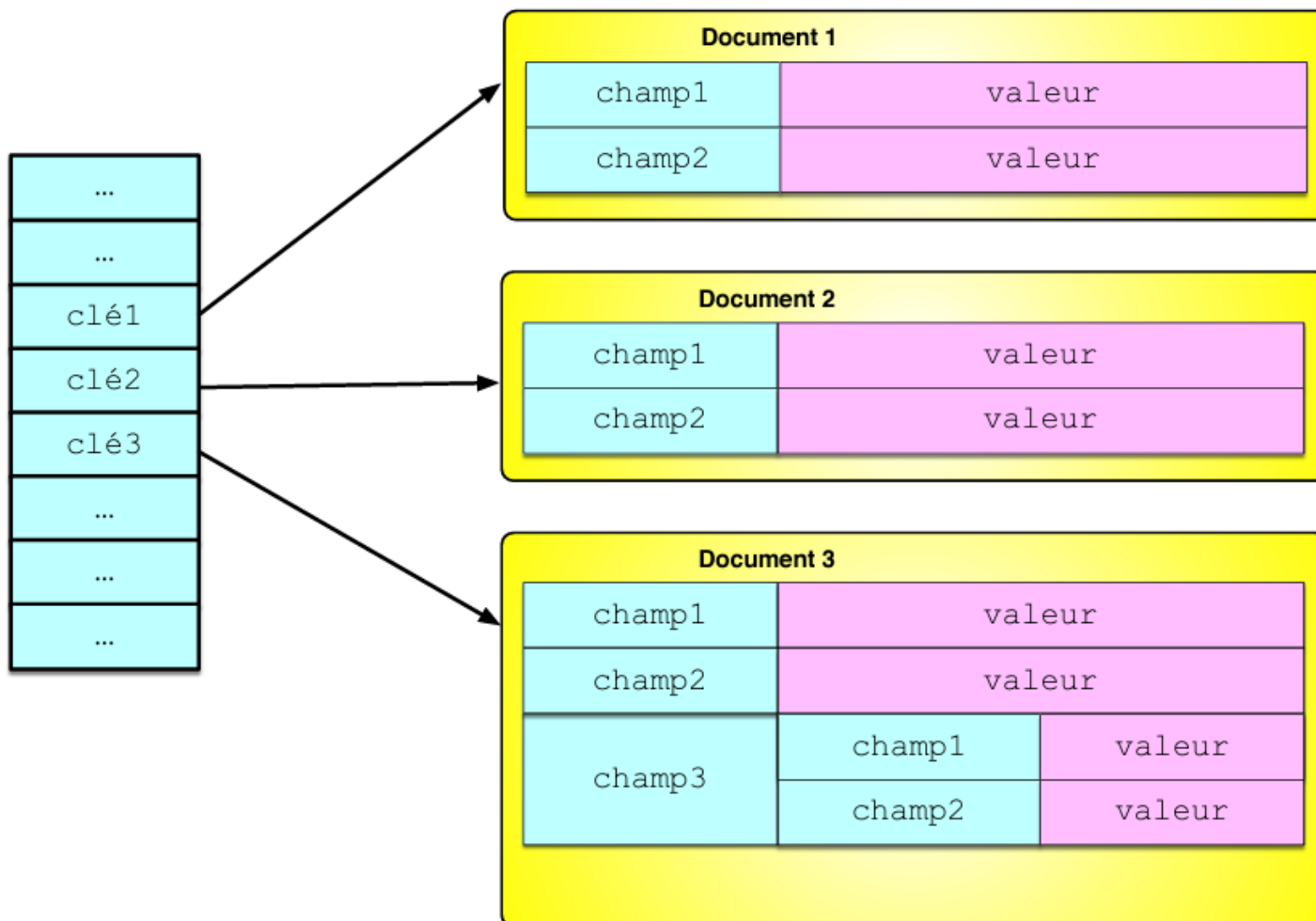
BD NoSQL modèle "Document"

- 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*, *MongoDB*, *Terrastore*, *ElasticSearch*

BD NoSQL modèle "Document"

- Un document est composé de champs et des valeurs associées
- Ces valeurs :
 - peuvent être requêtées
 - sont soit d'un type simple (entier, chaîne 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 modèle "Document"



BD NoSQL modèle "Document"

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 modèle "Document"

Collection VENTES :

OID	
5d5gj6ksrg8b45	<pre>"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 } }</pre>
8gd5gty6u6a34	<pre>"novente" : 2 "date" : "04/02/21" "livres" : [{ "noLivre" : 486734 "titre" : "Usage de MapReduce" "auteur" : { "nom" : Schmit "prenom" : Paul } }]</pre>

BD NoSQL modèle "Document"

Forces :

- Modèle de données simple mais puissant (expression de structures imbriquées)
- Bonne mise à l'échelle
- 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 modèle "Document"

Utilisations principales :

- Outils de gestion de contenu (Content Management System (CMS)),
- Catalogues de produits,
- Web analytique, analyse temps réel,
- Enregistrement d'événements, stockage de profils utilisateurs,
- Systèmes d'exploitation,
- Gestion de données semi-structurées
- ...

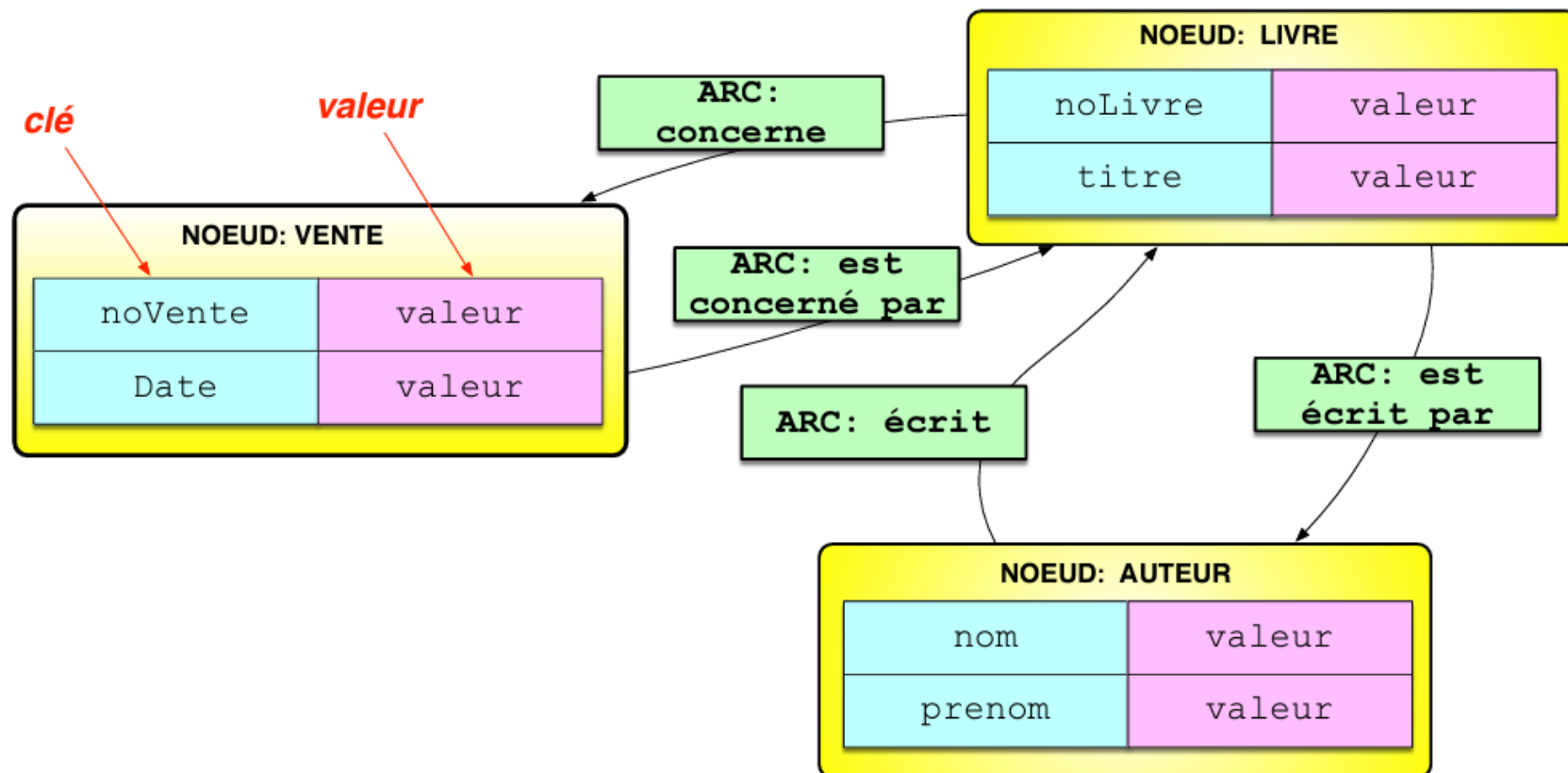
BD NoSQL "orientées graphes"

- 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 "orientées graphes"

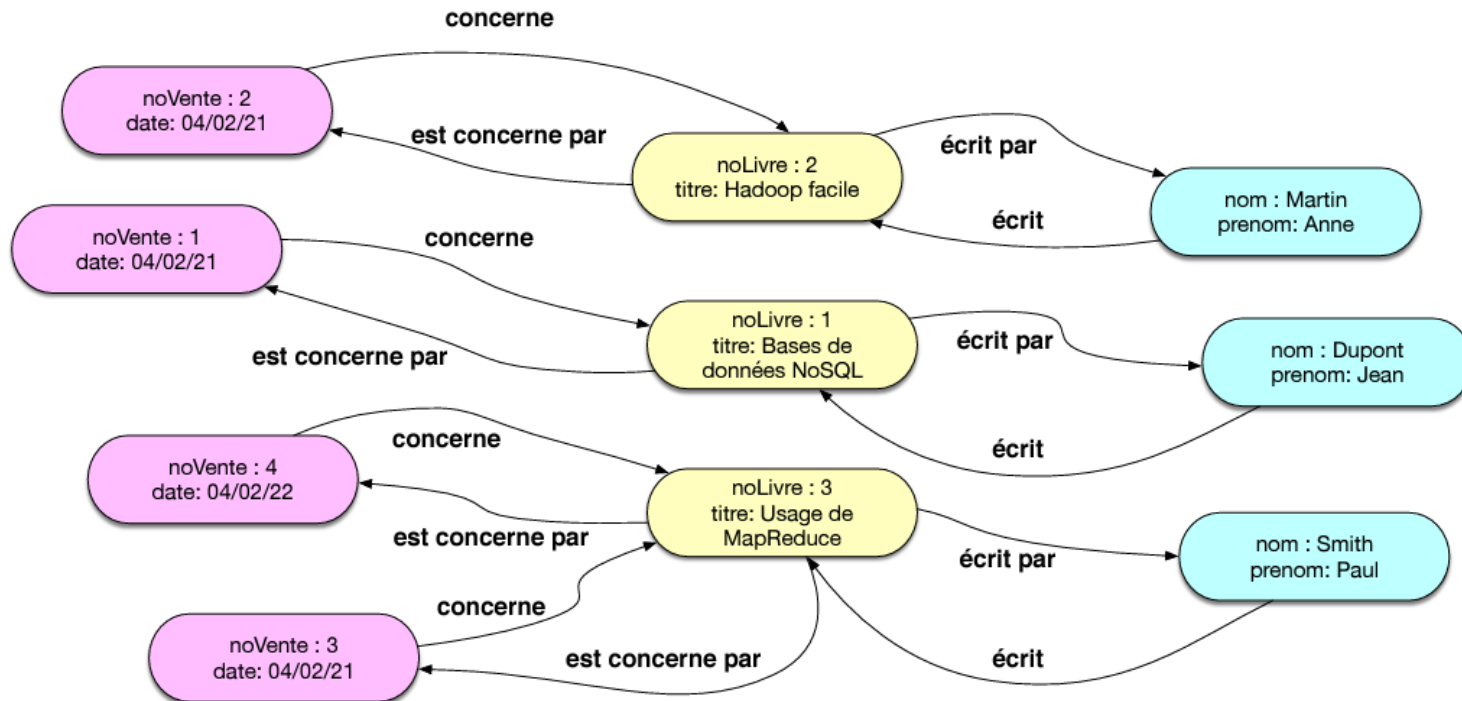
- Elles utilisent :
 - un moteur de stockage pour les objets (similaire à une base documentaire, chaque entité de cette base étant nommée nœud)
 - un mécanisme de description d'arcs (relations entre les objets), arcs orientés et avec propriétés (nom, date, ...)
- Elles sont bien plus efficaces que les 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 "orientées graphes"



BD NoSQL "orientées graphes"

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 "orientées graphes"

Classe VENTE :

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

Classe LIVRE :

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

Classe AUTEUR :

OID		
87d4hj6ksrg834	<i>property</i>	nom : Dupont
	<i>relation</i>	prenom : Jean
37b87hte623b57	<i>property</i>	nom : Martin
	<i>relation</i>	prenom : Anne
67kp87hte623b5	<i>property</i>	nom : Schmit
	<i>relation</i>	prenom : Paul

BD NoSQL "orientées graphes"

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 parfois même problématique

BD NoSQL "orientées graphes"

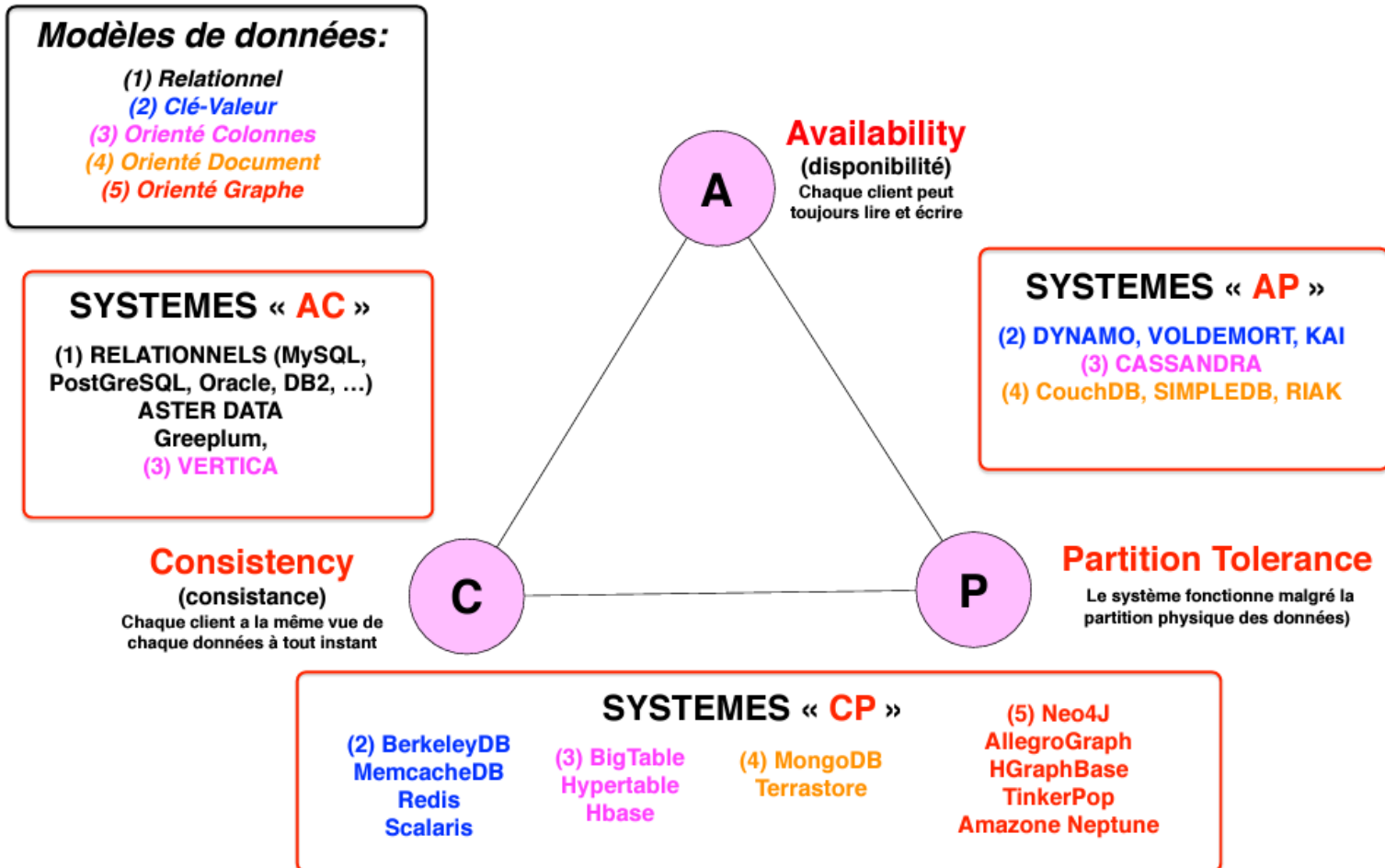
Utilisations principales :

- 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 "orientées graphes" 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, ...*

Les bases de données et le théorème de CAP



Bilan sur les BD NoSQL

Conception, modélisation d'une BD NoSQL : Encore délicate

- Pas de tables, seulement des paires de « cles-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...)
- ⇒ Il n'y a 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 avec 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 qui émerge
- Devraient permettre :
 - une interrogation des données via SQL
 - tout en garantissant des performances et un passage à l'échelle similaire aux bases de données NoSQL.
 - et conserver 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, ...