Bases de données NoSQL

Mihaela JUGANARU
mihaela.juganaru@emse.fr

École d'Été, Bénin

octobre 2022

Bibliographie :

- Gaurav Vaish (2013) **Getting Started with NoSQL. Your guide to the world and technology of NoSQL**, Packt Publishing
- Rudi Bruchez (2015) **Les bases de données NoSQL et le Big Data. Comprendre et mettre en oeuvre**, Eyrolles
- ... le site web du chaque éditeur NoSQL

Not Only SQL

NoSQL désigne les systèmes de gestion de base de données **qui ne s'appuient plus**, du fait des volumétries et de la variété des données contenues, **sur une architecture relationnelle et transactionnelle**.

Notion apparue en 1998, en 2010 le premier congres international NoSQL.

Utilisé par les grands géant du web (Google, Amazon, Ebay, Twitter, LinkedIn, etc.)

La réalité NoSQL

Très grande hétérogénéité des modèles de données, des façons de d'implémenter et de consulter les données, des fonctionnalités proposées.
Caractéristiques communes aux produits NoSQL :

- adaptés (aussi) à la gestion des données issues et pour le Web
- basés sur de protocoles et des langages ouverts, sans toutefois de proposer un quelconque standard
- "plus simples" à utiliser et à administrer que les produits SGBDR
- pas la même couverture que les SGBDR

Limites du relationnel

Limitations du modèle relationnel :

- mal adapté à la modélisation par objet et les données semi-structurées (valeur NULL)
- performances dépendantes de produit
- schéma à définir de manière globale et ensuite les applications à bâtir. Strictement déconseillé à modifier le schéma d'un BDR déjà en production.
- centralisation du schéma

Les plus : langage SQL, certains produits extrêmement robustes et scalable, paradigme ACID et transaction.
Le moins : coût (avec des excellentes performances Oracle DB, DB2 d'IBM, SQL Server) ou performances limitées pour les produits gratuits et très répandus (MySQL)

NoSQL et Big Data

Le NoSQL a deux grands avantages :

- capables de traiter des grands volumes grâce à l'usage des serveurs distribués (et parfois le paradigme MapReduce)
- le schéma de la base est plus facile à modifier (parfois on dit qu'il n'y a pas de schéma)

Besoin typiques pour l'usage de NoSQL : pas forcément besoin de transactions, structure de la base adaptative, beaucoup d'écritures complètes et de lectures sans trop de mises à jour (update et delete), pas de jointures.

Distribution

La distribution se réalise via des serveurs distribués qui traitent une même masse de données.

Une donnée se trouve répliquées sur plusieurs serveurs. La replication se fait par **partitions** (**shard**).

Les serveurs peuvent être en grille ou en cluster. Aussi de plus en plus de services NoSQL sous la forme DBaaS (DataBase as Service) dans le cloud.

Distribution

Distribution peut être :

- **synchrone** : les réplicas sont tous alimentés avec la même version avant de réaliser une écriture, puis la mise à jour s'accompagne d'une diffusion de cette transformation sur tous les replicas.
- **asynchrone** : pas de garantie de cohérence des réplicas.

La synchronisation des réplicas est très couteuse et impossible à réaliser dans un système tolérant aux pannes.

Distribution

Deux modes essentiels :

- architecture centralisée : modèle maître-esclave
- architecture (fortement) distribuée (le maximum est *sharing nothing*)

Transaction distribuée

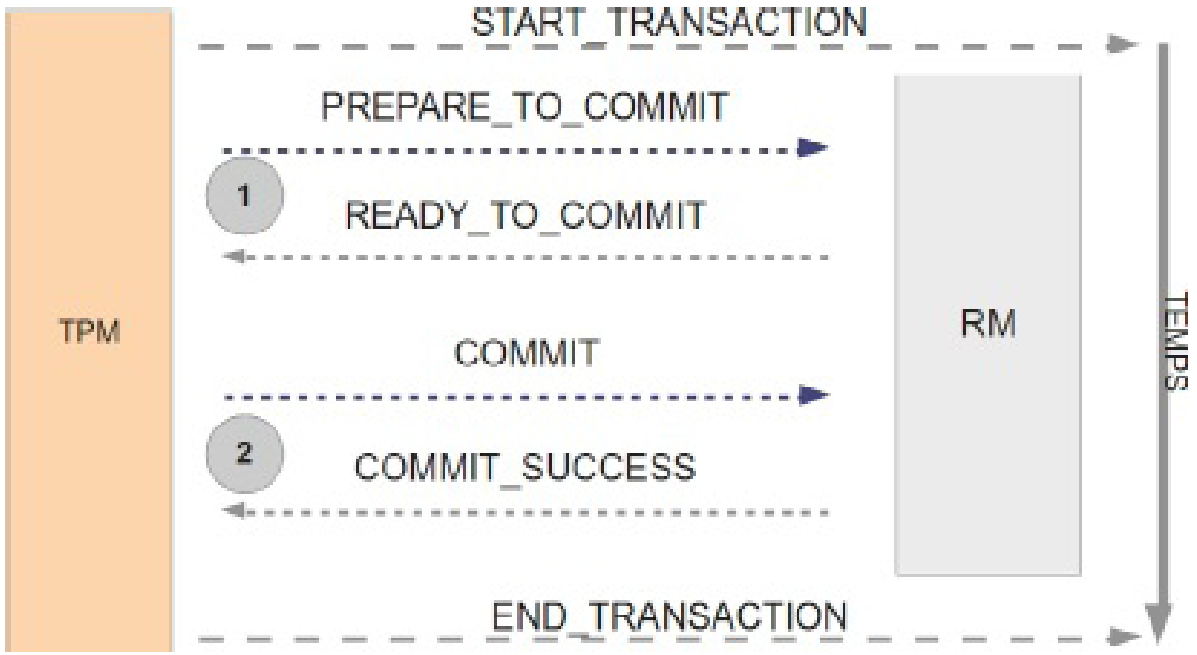
La transaction distribuée se réalise en deux phases :

- préparation à la validation et transmission de ce possible acquittement de la part de tous les sites concernés via le RM
- attente de tous les acquittements

Les principaux acteurs sont : Transaction Processing Monitor (TPM) et Replication Manager (RM).

Le RM calcule les serveurs qui contiennent une donnée (une partition) selon une clé de hachage attachée à la donnée.

Transaction distribuée



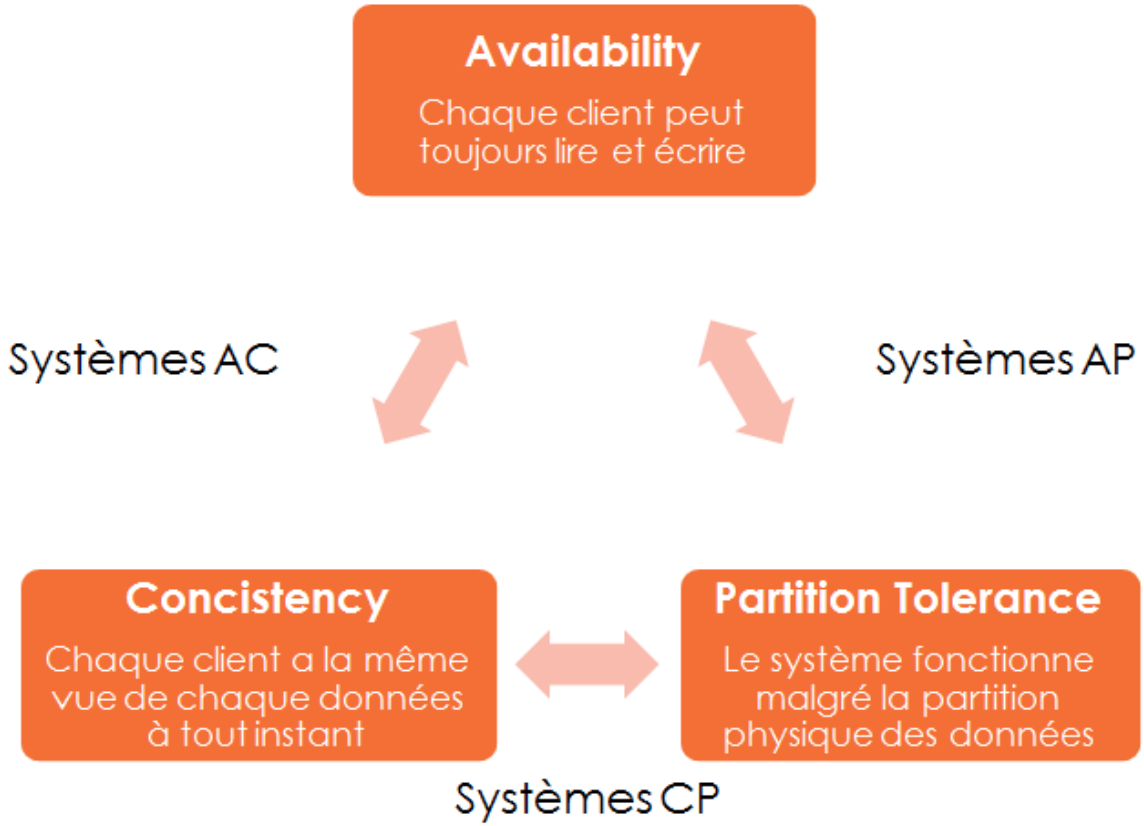
Implémentation des transactions

Certains NoSQL implémente les transactions (CouchDB et Neo4j, HBase pour une ligne).

Autre solution serait de passer par programme en appelant une couche système et un outils de type ZooKeeper (Apache).

Théorème CAP

Trois propriétés idéales pour une BD distribuées : consistence, disponibilité et tolérance au pannes.



(source : Livre Blanc du Big Data, SMILE, 2015)

Théorème CAP

Théorème CAP (Eric Brewer, 2000) : Il est impossible sur un système informatique de calcul distribué de garantir en même temps les trois contraintes suivantes :

- Cohérence (*Consistency*) : tous les nœuds du système voient exactement les mêmes données au même moment ;
- Disponibilité (*Availability*) : garantie que toutes les requêtes reçoivent une réponse ;
- Tolérance au partitionnement (*Partition Tolerance*) : 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).

Propriétés relâchées qui remplacent le paradigme ACID : **BASE**.

BASE : basically available ; soft state, eventually consistent

Intégration

Le monde NoSQL est conçu pour et à partir du Web.

La plupart des bases NoSQL sont consultables à travers le réseau avec des protocoles de type Thirft, CURL ou REST.

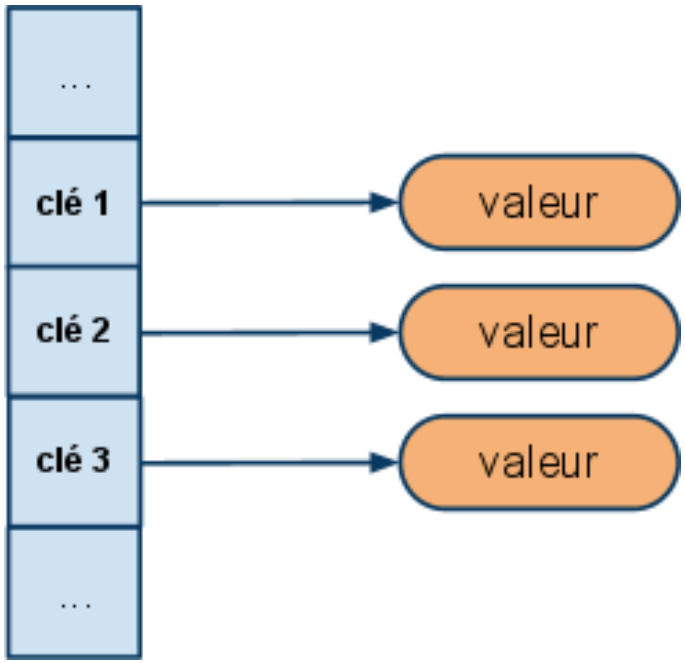
Chaque produit est accompagné des API permettant l'interfaçage avec des langages de type Java, C++, Python, Julia.

Quatre modèles pour les BD NoSQL :

- **clé - valeur** : Oracle NoSQL, projet Voldemort, CouchDB, Dynamo, FoundationDB, HyperDex, MemcacheDB, **Redis**, Riak, FairCom c-treeACE, Aerospike, OrientDB, MUMPS
- **orienté colonne** : Accumulo, Cassandra, Druid, **HBase**, Vertica
- **document** : Clusterpoint, Apache CouchDB, Couchbase, DocumentDB, HyperDex, Lotus Notes, MarkLogic, **MongoDB**, OrientDB, Qizx, eXist
- **graphe** : Allegro, Neo4J, InfiniteGraph, OrientDB, Virtuoso, Stardog
- **multimodel** : OrientDB, FoundationDB, ArangoDB

Modèle clé - valeur

Le modèle clé-valeur (key-value) consiste dans une énorme table de hachage. Le hachage se fait selon la valeur de la clé qui est unique dans toute la base.



La valeur pointée peut avoir une structure complexe, d'autres composants sont identifiés selon une partie de la clé (qui est composée à son tour) dite mineure.

Modèle clé - valeur

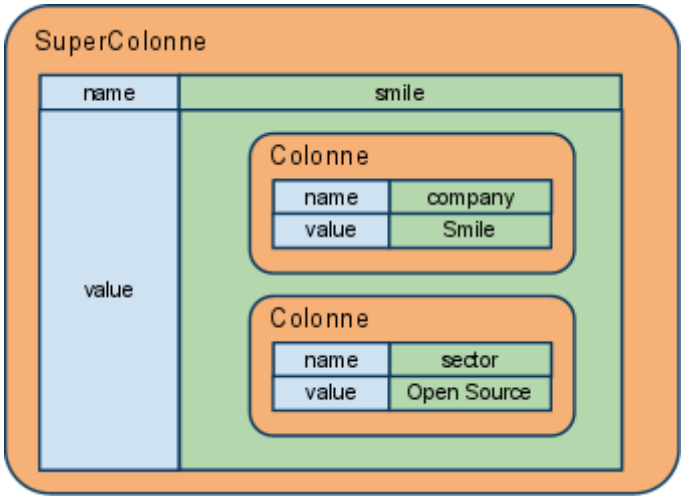
Ce modèle est parfois repris dans les bases orientées colonne et dans celles orientées document.

Les opérations de base assurées sont : Create, Update, Read, Delete (CURD).

Modèle orienté colonne

Les données sont stockées par colonnes et non par lignes, ce qui permet plus de souplesse pour les schéma dynamique et évite aussi de stockée les valeurs NULL.

Une colonne peut se composer des plusieurs colonnes, on a une **super colonne**.



On peut aussi compresser une colonne s'il y a des valeurs qui se répètent.

Modèle orienté document

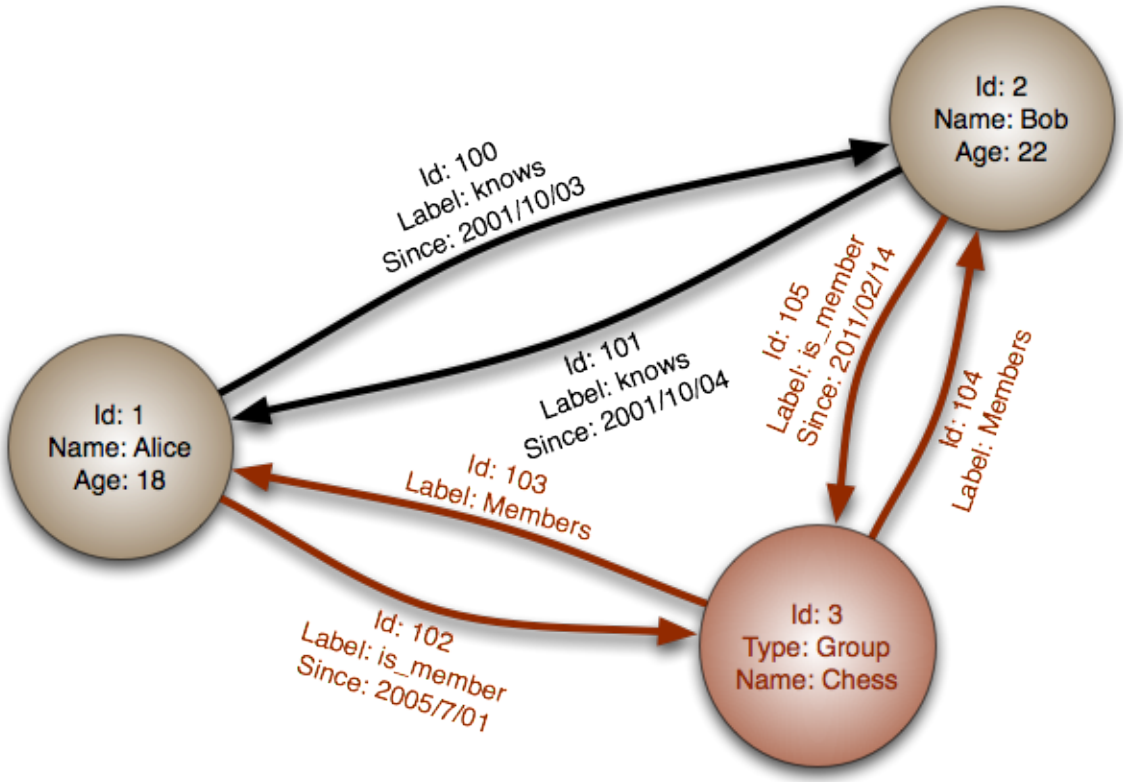
Le concept central est le **document** qui est regroupé en **collections**.

Souvent il s'agit des documents en format XML et JSON.
Un document peut avoir des méta-informations et des clés définies autant au niveau du document que de ses composants.

Il est aussi possible de voir implémentées les langages de requêtes spécifiques comme XPath et XQuery pour XML.

Modèle graphe

On considère qu'on stocke sous la forme les noueds sous la forme des documents et aussi les arcs (orientés) éventuellement avec des propriétés.

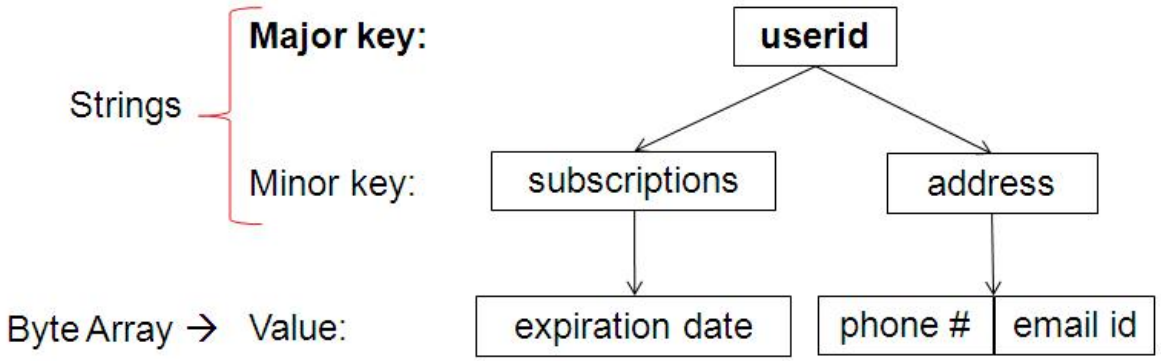


Fort utile pour les graphes des réseaux sociaux et pour des graphes

Oracle NoSQL Database

Produit Oracle depuis 2011 (dernière version stable - octobre 2015).

Implémente le modèle key-value et les données sont structurées logiquement dans des tables qui peuvent se concevoir comme expression d'une modèle à objet. La clé se décompose en majeure et mineure :



Oracle NoSQL Database

Chaque entité gardée est éventuellement sérialisée (json, par exemple) et assignée à une partition selon sa **sharding key** qui peut être différente de la **primary key**.

Supporte le passage à l'échelle et les transactions ACID. On peut le configurer en mode C/P ou A/P (par rapport au référentiel CAP).

Il est possible d'introduire des indexes secondaires. Protocole SSL pour assurer la sécurité (version Entreprise).

Cassandra

Produit open source (depuis 2008) basé sur le modèle colonne et clé valeur.

Une base est appelée **keyspace** et une colonne fait partie d'une **column family** appelée aussi **table**.

Décentralisée, la replication peut être réalisé aléatoirement (un couple clé-valeur est utilisé pseudo-aléatoirement pour désigner une partition) *RandomPartitioner* ou selon un principe de localité (si les clés sont proches les données seront stockées dans la même partition) *OrderPreservingPartitioner*.

Un langage d'interrogation CQL (Cassanda Query Language) adapté de SQL.

MongoDB

Produit sous licence actuellement sous Server Side Public License (SSPL) développé à partir de 2007 en C++(puis Go, Python, Javascript).

Utilisé par NewYork Times, Alibaba (Baba), etc.

Un SGBD NoSQL basé document, un document étant représenté par un par un code en JSON et stocké en BSON (Binary JSON).

Une collection est un ensemble de documents qui n'ont pas tous le même format. Une collection supporte la notion de tableau (`array`) de documents ainsi que la notion de document imbriqué (`nested`).

MongoDB

Indexation : une clé principale, le ID du document et des clés secondaires. Un filtrage efficace par type.
Temporalité : on peut indiquer que certaines collections expirerons après un laps de temps, certaines collections peuvent être de taille fixe, on peut indiquer aussi de faire une indexation partielle, uniquement sur des documents récents.
Distribution : réplication et partitionnement (shard). Un FS dédié à la manipulation de la data.

MongoDB

Agrégation : partielle ou complète
Optimisation
Interrogation : via un shell ou un langage de programmation.
Code embarqué dans la base (JavaScript)