

Bases de données NoSQL

Chapitre 3 : Typologie des BD NoSQL

Pr. ZAIDOUNI Dounia

Filière: ASEDS, S2, INE1
Institut National des Postes et Télécommunications (INPT)

29 Avril 2025

Présentation du syllabus du cours

- Chapitre 1: Introduction aux BD NoSQL
- Chapitre 2: Fondements des BD NoSQL
- Chapitre 3: Typologie des BD NoSQL
- Chapitre 4: MongoDB
- Chapitre 5: Apache Hbase

1 Typologie des BD NoSQL

- 1 Typologie des BD NoSQL
 - BD NoSQL orientées 'agrégats'
 - BD NoSQL orientées 'graphes'

Il existe deux catégories de SGBD NoSQL :

- **Les BD NoSQL orientées 'agrégats' :**

- **Type 'Clés-valeurs' :**

- Exemples : DynamoDB, Redis, Riak, Voldemort, Memcached, ...

- **Type 'Colonne' :**

- HBase, Cassandra, Hypertable, Amazon Keyspaces, ...

- **Type 'Document' :**

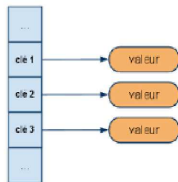
- Exemples : MongoDB, CouchDB, Couchbase, RavenDB, ArangoDB, ...

- **Les BD NoSQL orientées 'graphe' :**

- Exemples : Neo4j, ArangoDB, TigerGraph, OrientDB, Amazon Neptune, ...

BD type 'Clés-Valeurs'

- Ce modèle peut être assimilé à une table de hachage (hashmap) distribuée.
- Les données sont simplement représentées par un couple clé-valeur.
- La valeur: une simple chaîne de caractères, un objet sérialisé, etc.
- Dans ce modèle clé-valeur, on stocke les entités mais pas les relations.
- Ce modèle retourne une valeur dont elle ne connaît pas la structure.
- Les structures de données ne sont pas contraintes par un schéma.
Selon l'implémentation, la clé peut être attribuée ou générée aléatoirement lors de l'ajout de l'enregistrement.
- Chaque objet est identifié par une clé unique seule façon de le requêter.



Les opérations 'CRUD'

- L'exploitation des BD de types clés-valeurs est basée sur 4 opérations 'CRUD' :
 - Create : créer un nouvel objet avec sa clé
 - `create(key, value)`
 - Read : lit un objet à partir de sa clé
 - `read(key)`
 - Update : met à jour la valeur d'un objet à partir de sa clé
 - `update(key, value)`
 - Delete: supprime un objet à partir de sa clé
 - `delete(key)`
- Ces BD de types clés-valeurs 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.

Exemples d'implémentation du modèle 'Clés-Valeurs'

- **Amazon Dynamo :** 
- **Riak :**  **riak**
- **Redis** (projet sponsorisé par VMWare) :  **redis**
- **Voldemort** (développée et utilisée par LinkedIn) : 

BD type 'Colonnes'

- Dans ce modèle, les données sont stockées par colonne, non par ligne.
- Chaque colonne est définie par un couple clé-valeur.
- Modèle proche d'une table dans un SGBDR mais plus flexible :
 - Les colonnes d'une base de données relationnelle sont statiques et présentes pour chaque ligne.
 - Dans une base de données orientée colonnes, les colonnes sont dynamiques et présentes uniquement pour les lignes concernées.
 - Il n'y a pas de colonnes avec comme valeur Null donc le coût de stockage des valeurs Null est 0.
 - Il est possible d'ajouter des colonnes à une ligne à tout moment.

	A	B	C	D	E
1	foo	bar	hello		
2		Tom			
3			java	scala	cobol

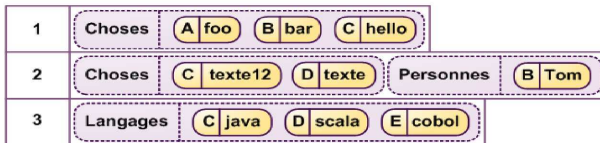
Organisation d'une table dans
une BDD relationnelle

1	A foo	B bar	C hello
2	B Tom		
3	C java	D scala	E cobol


Organisation d'une table dans
une BDD orientée colonnes

Les principaux concepts associés au modèle 'colonnes'

- Une **Colonne** est une entité de base représentant un champ de donnée, chaque colonne est définie par un couple clé-valeur.
 - Les colonnes sont stockées de manière triée sur le disque :
 - Cela permet d'obtenir un intervalle de colonnes avec un nombre réduit d'accès aléatoires sur le disque. Il nécessite par contre de reconstruire l'ensemble de la table de données sur le disque au fur et à mesure des modifications.
- Une **supercolonne** est une colonne contenant d'autres colonnes.
 - L'utilisation des super-colonnes ajoute une dimension supplémentaire au modèle permettant par exemple le stockage de listes de listes.
- Une **famille de colonnes** permet de regrouper plusieurs colonnes (ou supercolonnes).



Organisation d'une table dans une
BDD orientée colonnes avec *super-colonnes*

- **Cassandra** (Initialement créé par Facebook) :  **Cassandra**

- **Hbase** (développé au sein de Hadoop) : 

BD type 'Documents'

- Elles stockent une collection de “documents”, une collection contient plusieurs 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, il est possible aussi de stocker n'importe quel objet, via une sérialisation.
- Les documents ont une structure arborescente, ils contiennent une liste de champs, un champ a une valeur qui peut être une liste de champs.

BD type 'Documents'

- Bien que les documents soient structurés, ces BD n'ont pas de schéma (schemaless), il n'est pas nécessaire de définir au préalable les champs utilisés dans un document :
 - Les documents peuvent voir une structure hétérogène au sein de la BD.
 - Ces BD permettent d'effectuer des requêtes sur le contenu des documents/objets.
- Elles ont généralement une interface d'accès HTTP REST permettant d'effectuer des requêtes.

Exemples d'implémentation

- **CouchDB** (distribué sous licence Apache) :



- **MongoDB** (développé par MongoDB inc) :



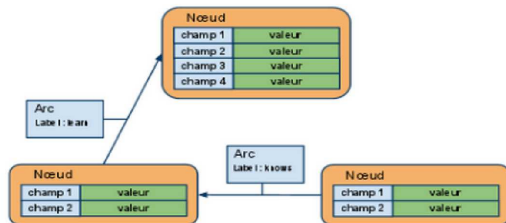
- **Couchbase** (sous licence Apache HTTP Server) :



BD NoSQL orientées 'graphes'

- C'est une BD utilisant la théorie des graphes. Elle est adaptée pour les cas où les relations entre les données sont aussi importantes, voire plus, que les données elles-mêmes.
- Elle correspond à un système de stockage de données dans un graphe, capable de fournir une adjacence entre éléments voisins : chaque voisin d'une entité est accessible grâce à un pointeur physique.
- Les enregistrements sont des noeuds.
- Les noeuds sont connectés entre eux par des relations orientées.
- Les noeuds et les relations peuvent avoir des attributs qu'on appelle 'propriétés'.
- Un 'label' est un nom qui organise des groupes de noeuds.
- Un 'chemin' est un ou plusieurs noeuds connectés par des relations. Il résulte d'une requête ou d'une traversée.

BD NoSQL orientées 'graphes'



Exemple d'implémentation

- **Neo4j** (développé par Neo Technology, Inc) :



- **OrientDB** (Orienté graph et document) :



- **ArangoDB** (Orienté graph et document) :



- Exemples d'utilisation :

- Exploitation des relations entre les données :
 - Par exemple : connaissances entre des personnes, description de l'ensemble des pièces d'une machine industrielle et de la manière dont elles sont liées entre elles.
- Modélisation des données Géo-Spatial : Carte routière et calculs d'itinéraires.
- Elles sont utilisées dans la modélisation des réseaux sociaux.