# CHAPITRE 1

# La problématique du Big Data

**Contenu du chapitre :**

| Type | Base |
|---|---|
| Colonne | Hbase |
| Clé-valeur | Redis |
| Clé-valeur/CI | Cassandra |
| Document | MongoDB |
| Graphe | Neo4j |
| | Oracle |

**Objectifs du chapitre :**

Après une définition du concept de Big Data, ce chapitre, sans constituer un cours, rassemble des pratiques et des considérations dont la connaissance est souhaitable pour tirer parti des différents frameworks. De même, quelques notions sur les bases de données relationnelles sont rappelées pour permettre de mieux comprendre les bases de données **non** relationnelles à la source des frameworks de Big Data.

| Systèmes | | | | | |
|---|---|---|---|---|---|
| Windows | x | Linux | x | Mac | x |

## 1.1 Introduction

Le terme de bases de données "NoSQL" pour "Not only SQL" regroupe des solutions récentes qui se différencient du modèle SQL par une logique de représentation des données différente. Leurs principaux avantages sont leurs performances et leur capacité à traiter de très grands volumes de données ; en particulier lorsque cela concerne le stockage de données dont la structure varie.

Le terme de base de données NoSQL, gagnerait à être remplacé par le terme de base de données "non relationnelle" ; cela pourrait éviter une certaine confusion dans l'esprit de concepteurs issus du monde relationnel. Les préceptes non relationnels pour le stockage de données (parfois importantes) sont très anciens en informatique : on pense, par exemple, à la structure des répertoires sous Windows ou linux. Toutefois, le terme NoSQL a pris récemment une nouvelle dimension dans le cadre du stockage de très grandes quantités de données : ce qu'on regroupe sous la terminologie "Big Data". Historiquement, la notion de Big Data est apparue avec la collecte de grandes masses d'informations sur Internet.

Le Big Data est une innovation qui se trouve à la frontière entre technologie et management. Même les institutionnels dont le ministère de l'Economie Numérique

tentent d'accompagner ce mouvement en allouant 11.5 millions d'euros des investissements d'avenir à des projets de type Big Data. L'intérêt de l'état se manifeste par un soutien important qui se base sur l'idée que les promesses de croissance autour du traitement de ces grandes masses de données sont jugées suffisamment grandes pour justifier ce type de mesures. La "data-driven company" est une notion qui attire beaucoup d'investisseurs car elle laisse entrevoir des possibilités de réduction des coûts, d'augmentation du chiffre d'affaires ou de conception de nouveaux outils de décision.

### *1.1.1 Définition d'un framework pour le Big Data*

On regroupe sous le terme Big Data des technologies dédiées aux traitements de données de forte volumétrie et qui répondent au critère des 4V :

- **V**olume important de données ;
- **V**ariété des informations (généralement peu ou pas structurées) ;
- **V**élocité exigée dans le traitement due à la fréquence de création et/ou de collecte.
- **V**ariabilité permettant de s'adapter au format changeant des données.

Le critère des **4V** est une extension du critère des **3V** introduit en 2011 par le cabinet de conseil McKinsey and Company. Parfois même on trouve des auteurs parlant du critère des **5V** dans lequel le cinquième **V** correspond à Véracité représentant le fait que la véracité des données est un élément important à vérifier et à intégrer dans un processus de traitement des données.

Afin de se faire une idée sur les enjeux liés au volume important des données, il suffit de savoir que tous les deux ans, il se génère autant de données que depuis le début de l'humanité et qu'en 2010 cela devrait représenter 40 zettaocters de données.

Les frameworks de Big Data recouvrent trois éléments complémentaires (Figure 1-1) :

- un système de bases de données non relationnelles appelées à tort NoSQL ;
- un système de distribution des traitements ;
- un système de stockage de données en mémoire pour accélérer les traitements.

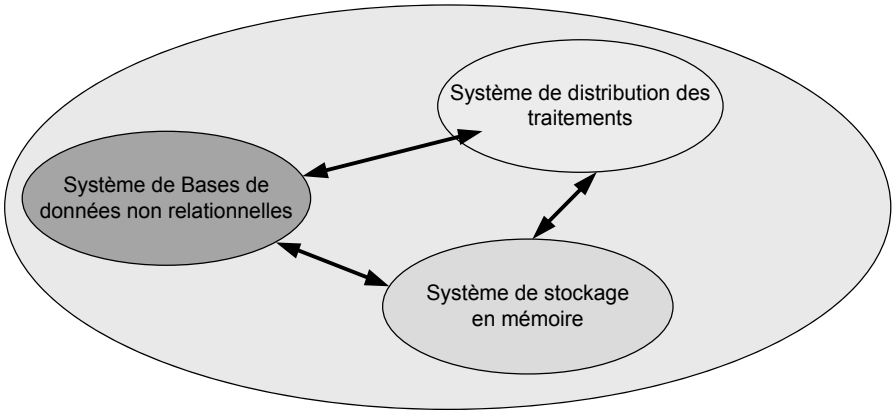


*Figure 1-1. Les trois éléments d'un framework dans le Big Data*

Le plus connu des frameworks de Big Data est sans conteste "Hadoop" auquel un chapitre complet est consacré dans ce livre. Le framework Hadoop comprend le système

de bases de données "Hbase" pour le stockage des données, un système appelé "MapReduce" pour la distribution des traitements et un système de fichiers distribués HDFS qui assure la sauvegarde et l'accès aux données (Figure 1-2).

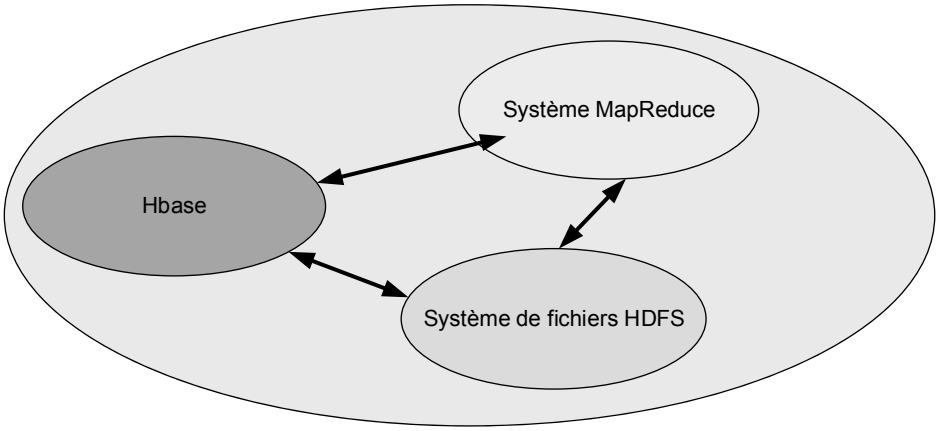


*Figure 1-2. Les trois éléments d'Hadoop*

La représentation d'Hadoop sur la Figure 1-2 ne peut être que schématique. Seuls ses composants principaux y sont mentionnés. Pour chaque framework dédié au Big Data il existe un écosystème (ensemble d'applications prévues pour fonctionner ensemble) qui couvrent à la fois les problématiques de stockage mais aussi de restitution des résultats (tableaux de bord, fouille de données, etc.). Le lecteur trouvera par exemple dans le chapitre 7 de ce livre, une présentation d'Hadoop, et une rapide présentation de son écosystème particulièrement riche comme on pourra facilement s'en convaincre.

### *1.1.2 Spécificités des bases de données relationnelles*

Lorsqu'on réalise l'analyse des données on passe généralement par la conception d'un MCD et d'un MLD qui permettent à terme d'identifier :

- les entités (on peut aussi parler de classes) ;
- les attributs de ces entités ;
- les relations qui existent entre ces entités.

Cela suppose évidemment que les données analysées sont représentatives des données à stocker et il est souhaitable que le format des données en entrée n'évolue pas au cours du temps.

Considérons l'exemple de la gestion d'une bibliothèque et en particulier la gestion des livres. On considère qu'un livre est écrit par un seul auteur et qu'un auteur peut écrire plusieurs livres. On envisage aussi le cas où le nom d'un auteur est stocké dans la base de données alors qu'aucun de ses livres ne figure dans cette bibliothèque.

Après l'analyse et la spécification du problème on retrouve les différentes informations structurées dans les listes ci-dessous (Tableau 1-1 et Tableau 1-2).

**Tableau 1-1. Liste des livres**

| Numéro | Titre | Prix | Auteur |
|---|---|---|---|
| 10101 | aaaaa | 10 | Emilie Castafiore |
| 11111 | ee | 54 | Emilie Chambord |
| 80808 | cccc | 45 | Pierre Dupont |
| 90909 | ddddd | 35 | Roland Momo |
| 202022 | bb | 25 | Sylvie Fabière |

**Tableau 1-2. Liste des auteurs**

| Nom | Prénom | Domicile | Numéro |
|---|---|---|---|
| Castafiore | Emilie | Paris | 85478 |
| Chambord | Emilie | Nice | 3547 |
| Dupont | Pierre | Avignon | 542563 |
| Fabière | Sylvie | Bordeaux | 52136 |
| Momo | Roland | Toulouse | 8547585 |
| Tintin | Thiery | Clermont | 78545 |

L'analyse du problème permet d'identifier deux entités : AUTEUR et LIVRE.

Il existe une relation que l'on peut nommer **Ecrire** entre l'entité LIVRE et l'entité AUTEUR. Le MCD correspondant est proposé sur la Figure 1-3.

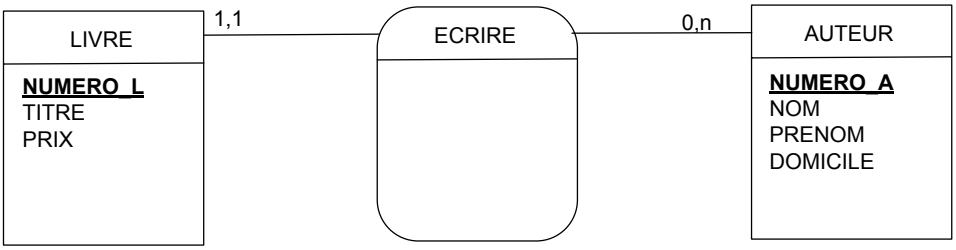


*Figure 1-3. MCD du problème de gestion de la bibliothèque*

Le schéma fait apparaître qu'un livre est écrit par un et un seul auteur alors qu'un auteur est à l'origine de 0 à n livres.

Supposons que le schéma précédent donne lieu à la création d'une base de données MySQL (par exemple) et qu'au bout de trois semaines d'utilisation, la base contienne : 100 000 livres et 105 000 auteurs.

Supposons de plus, qu'à l'issue de la 3ème semaine de mise en œuvre, il est décidé, pour chaque livre, de stocker aussi un résumé du contenu car celui-ci est maintenant fourni directement par les maisons d'éditions. Une telle information stockée dans la base serait de nature à faciliter le développement d'un futur magasin en ligne en autorisant des recherches complexes par mots-clés.

Cela signifie que, concrètement, il faut ajouter un nouveau champ dans la table LIVRE et qu'elle contiendra 100 000 livres dont le champ **RESUME** ne sera pas renseigné (mais existera tout de même) pour l'ensemble des livres présents. On peut raisonnablement imaginer (sauf à effectuer, si cela était techniquement possible, une nouvelle saisie pour les livres déjà présents), que le champ **RESUME** ne sera renseigné qu'à partir du tuple numéro 100 001.

La table LIVRE ressemblera alors à celle de la Figure 1-4. On peut facilement constater que la présence d'une nouvelle information (ici un résumé) oblige à modifier les tuples numéros 1-100 000 par l'ajout d'un nouvel attribut (nouvelle colonne dans la table) alors

même que le résumé n'existe pas pour ces tuples. On voit ici les limites du modèle relationnel classique.

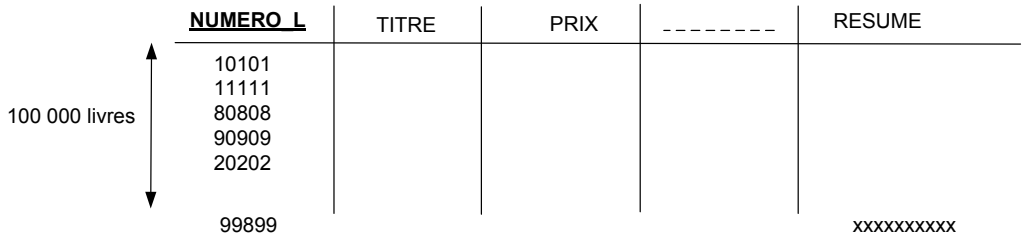


*Figure 1-4. Un exemple de table LIVRE après ajout d'un champ RESUME*

La situation aurait pu être encore plus complexe si à l'issue de la 3ème semaine de mise en œuvre, la législation sur les livres avait changé et si pour les livres édités par la suite, deux taux de TVA s'appliquaient : un taux de 5,50% aux romans et un taux de 10,50% aux livres historiques alors que pour les livres édités avant cette date, un taux unique et usuel de 7,70% continuait à être en vigueur. Avec cette nouvelle conjoncture, il faut modifier profondément le MCD/MLD et introduire une classe CATEGORIE et une relation d'appartenance à une catégorie (Figure 1-5).

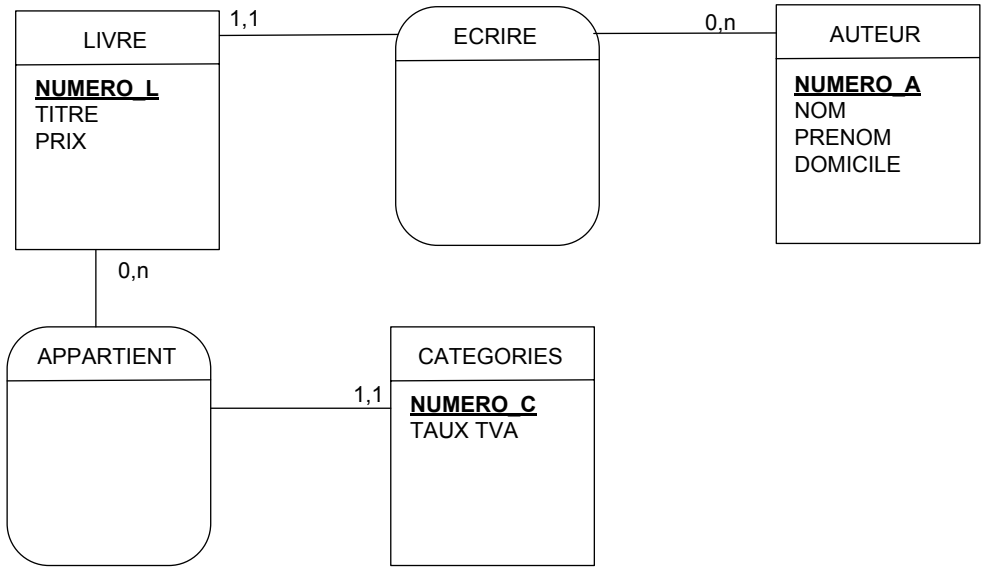


*Figure 1-5. MCD du problème de gestion de la bibliothèque avec les taux de TVA*

Pour éviter l'ajout de nouvelles tables dans le schéma actuel de la base, de nombreux concepteurs pourraient se contenter d'ajouter un champ TAUX TVA directement dans la classe LIVRE ce qui donnerait le schéma de la Figure 1-6.

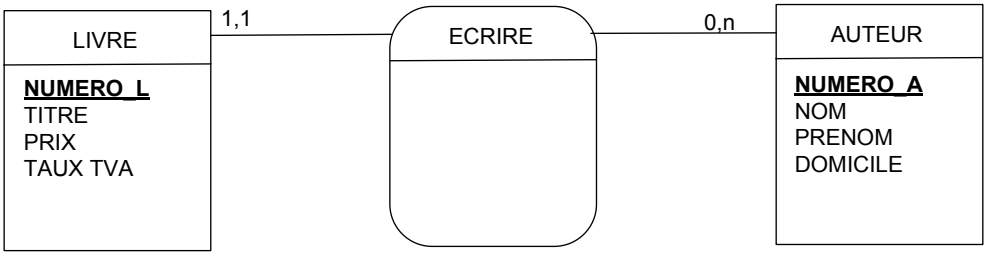


*Figure 1-6. Autre MCD du problème de gestion de la bibliothèque avec les taux de TVA*

### *1.1.3 Avantage des bases de données non relationnelles*

Les bases de données non relationnelles sont parfaitement adaptées aux traitements de données peu structurées et/ou ayant une structure changeante.

Le premier acteur économique d'importance ayant fait la promotion de ce type de base, est le géant Américain Google qui a rapidement mis en place un système dédié à la manipulation de gros volumes de données, basé sur :

- un système de fichiers distribués ;
- un système de bases de données orientée colonnes ;
- un système de parallélisation des traitements nommé MapReduce qui fera l'objet d'une présentation détaillée dans la suite du livre.

Les bases de données NoSQL (non relationnelles) constituent une nouvelle manière de représenter l'information. Elles s'affranchissent des contraintes dites ACID (Atomicité, Cohérence, Isolation, Durabilité) et fournissent une architecture technique où il suffit de rajouter des serveurs pour gagner en performance.

Dans les projets, il ne faut pas opposer ces deux approches mais bien souvent les faire cohabiter et les bases de données NoSQL ne visent pas à remplacer les SGBD traditionnels mais plutôt à les compléter.

## 1.2 Les différents types de bases de données NoSQL

On peut distinguer 5 représentations de base de données NoSQL qui seront détaillées par la suite dans les différents chapitres :

- **Clé-valeur :** il s'agit de la représentation la plus simple. Cette structure est très adaptée à la gestion de caches ou pour fournir un accès rapide aux informations. Elle fonctionne comme un (grand) tableau associatif et retourne une valeur (de nature éventuellement complexe) à partir d'une clé ;
- **Clé-valeur avec contraintes d'intégrité :** une représentation évoluée qui autorise en plus la définition de contraintes d'intégrité telles que celles qu'on retrouve dans les bases de données relationnelles ;

- **Document :** une représentation qui ajoute au modèle **clé-valeur**, l'association d'une valeur "complexe", c'est-à-dire qui nécessiterait un ensemble de jointures en logique relationnelle ;
- **Colonne :** une autre évolution du modèle **clé-valeur** qui permet de disposer de plusieurs valeurs, permettant ainsi de stocker les relations de type **one-to-many** ;
- **Graphe :** une représentation qui permet la modélisation, le stockage et la manipulation de données très "complexes" liées par des relations variées.

Quelques acteurs du monde NoSQL, c'est-à-dire des fournisseurs de solutions non relationnelles sont présentés sur la Figure 1-7. Ils se divisent, comme cela vient d'être précisé, en 5 catégories et émargent sous des licences essentiellement Apache. On peut s'attarder quelques instants sur la solution Hbase dans la mesure où cette solution a été retenue par Facebook, Yahoo, StumberUpon, Hulu, et de nombreux autres acteurs du web.

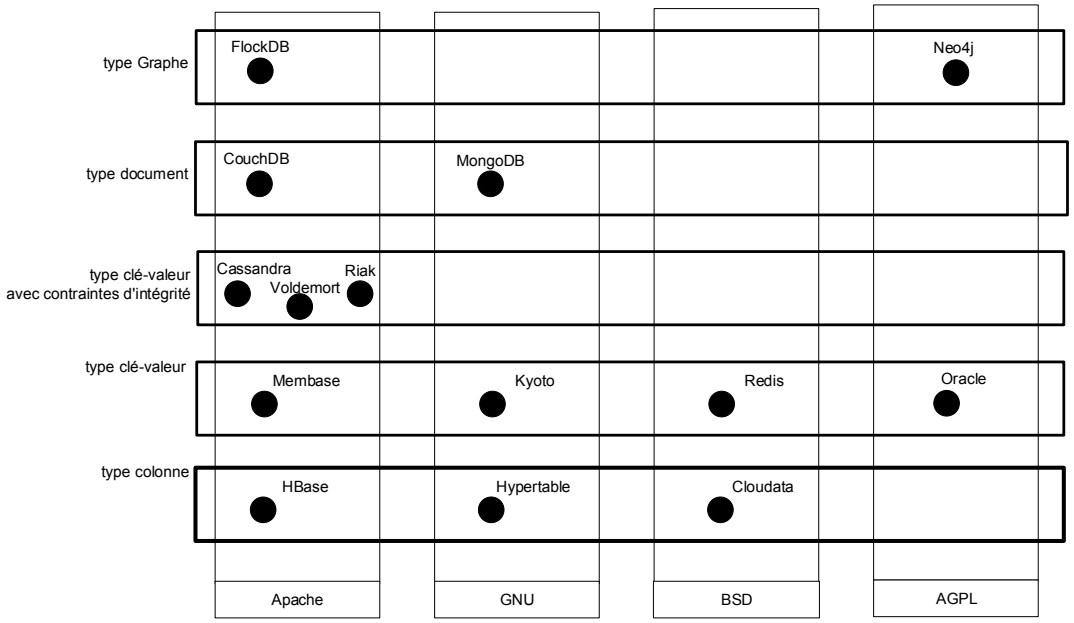


*Figure 1-7. Les différentes licences des principaux acteurs*

Le Tableau 1-1 résume les caractéristiques des différentes solutions et souligne en particulier la présence ou non d'API et de web services ainsi que d'un langage d'interrogation natif. Le lecteur intéressé par une classification exhaustive des systèmes existants peut se reporter au site web http://nosql-database.org/ qui offre le meilleur classement actuellement disponible.

Le lecteur intéressé par une classification des différents types de bases NoSQL peut avantageusement consulter le site suivant qui offre un panorama presque exhaustif :

http://5.freshminutes.it/2010/02/17/introduction-au-nosql-et-de-redis-ou-le-compte-rendu-du-nosql-user-group/

**Tableau 1-1. Caractéristiques des différentes solutions**

| Type | Nom | Langage | Date création | Langage d'interrogation natif | Web Services | Bibliothèque (API) |
|---|---|---|---|---|---|---|
| Colonne | Hbase | Java | 2007 | Non | / | Java |
| | Hypertable | C++ | 2007 | HQL | / | / |
| | Cloudata | Java | 2011 | CQL | REST | JAVA |
| Clé-Valeur | Membase | C et C++ | 2009 | / | / | / |
| | Kyoto | C++ | / | / | / | C, C++, Java, C#, Python, Ruby, Perl… |
| | Redis | C | 2009 | / | / | C, C++, Java, Python, Ruby, … |
| | Oracle | / | 2012 | / | / | Java, C |
| Clé-Valeur/CI | Cassandra | Java | 2008 | / | / | Java, Python, PHP, Ruby… |
| | Voldemort | Java | 2008 | / | / | / |
| | Riak | Erlang, C, Javascript | 2008 | / | / | Java, Ruby, Python, PHP, Javascript |
| Document | CouchDB | C, Javascript | 2005 | / | REST | / |
| | MongoDB | C++ | / | OUI | / | C, C++, Java, C#, Ruby, Javascript... |
| Graphe | Neo4j | Java | 2003 | SPARQL | REST | Java, Python, Ruby, PHP |
| | FlockDB | Scala | 2010 | / | / | / |

## 1.3 Quelles sont les promesses des frameworks dédiés au Big Data

Les solutions traditionnelles de bases de données se retrouvent au cœur du système d'information des entreprises de par leur utilisation :

- comme unité de stockage d'informations gérée par un ERP qui est l'épine dorsale du système d'information ;
- comme unité d'analyse permettant l'exécution de traitement de logiciels "BI", ces deux fonctions étant complémentaires.

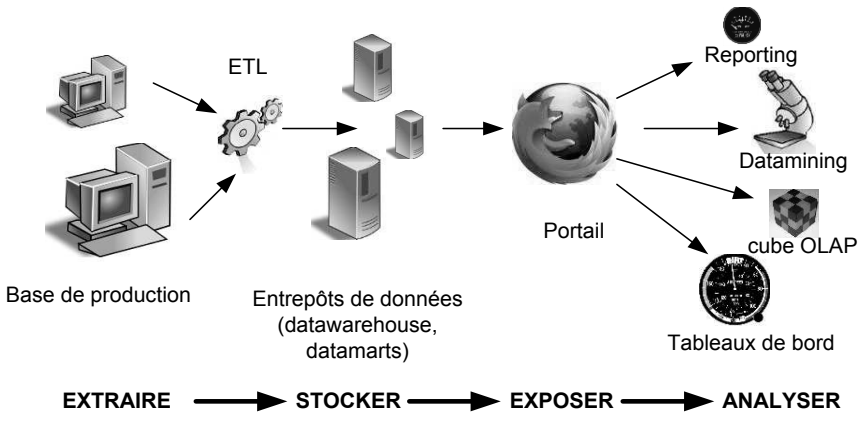


*Figure 1-8. Chaîne de traitement en Business Intelligence*