République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mouloud Mammeri de Tizi-Ouzou

En vue de l'obtention du diplôme de Master Academique
Spécialité : Informatique
Option : Systèmes d'informatiques

Thème

Conception et réalisation d'une base de données NoSQL sous Hbase et Firebase

Cas d'un tremblement de terre

Présenté par :
Narimane BENNABI
Nassima AIT IBRAHIM

Devant le jury composé de :
Président : Mr Yacine Younes
Examinateur : Mr Radja Hakim
Examinateur : Melle Yesli Yasmine
Encadreur : Mme Taouri Dalila

Année universitaire : 2018/2019

Remerciements

D'abord, nous remercions le bon ***DIEU*** *de nous avoir donné santé et courage pour réaliser ce travail.*

Nous tenons à exprimer notre profonde gratitude à notre encadreur ***Mme TAOURI Dalila****, pour nous avoir encadré et guidé et surtout pour ses judicieux conseils qui ont contribué à alimenter notre réflexion.*

Nous remercions également ***Mr ALIANE Abdelouahab Abdenour*** *pour le temps qu'il a consacré à nous aider et orienté dans la partie pratique.*

Nous remercions chaleureusement les membres de jury pour l'honneur qu'ils nous ont fait en acceptant de juger notre travail.

Nos sincères sentiments vont à nos parents qui ont sacrifié jusqu'aujourd'hui et leurs encouragements tout le long de notre parcours.

Narimane, Nassima.

Dédicaces

Je dédie ce modeste travail : A mes très chers parents que dieu les protègent, pour leur aide et leur soutien tout au long de mes études,
A toute ma famille, à mes chers amis,
Enfin à tous ceux qui ont contribué de près ou de loin pour la réalisation de ce travail.

Narimane.

Dédicaces

Je dédie ce modeste travail : A mes très chers parents que dieu les protègent, pour leur aide et leur soutien tout au long de mes études,
A toute ma famille, à mes chers amis,
Enfin à tous ceux qui ont contribué de près ou de loin pour la réalisation de ce travail.

Nassima.

Table des matières

Table des figures

Liste des tableaux

Résumé

Dans le monde d'aujourd'hui de multiples acteurs de la technologie numérique produisent des quantités infinies de données. Capteurs, réseaux sociaux ou e-commerce, ils génèrent tous de l'information qui s'incrémente en temps-réel selon les 3 V de Gartner : en Volume, en Vitesse et en Variété. Afin d'exploiter efficacement et durablement ces données, il est important de trouver un moyen pour les structurer : Trouver un modèle dynamique capable de supporter la diversité de la forme de ses données, la croissance exponentielle de leur quantité et donc la scalabilité et enfin assurer la sémantique de ces données en s'adaptant aux contraintes des outils technologiques qui sont à notre porté.

Mots clés

Big Data, Internet des Objets (IoT), Bases de données NoSQL, Hadoop, Apache HBase, Firebase, Scalabilité, Structurées, Semi-structurées, Non-structurées, Sémantique, Volume, Vitesse, Variété.

Abstract

In today's world, multiple players in digital technology produce infinite amounts of data. Sensors, social networks or e-commerce, they all generate information that is incremented in real time according to Gartner's 3V : in Volume, Velocity, Variety. In order to exploit these data efficiently and durably, it is important to find a way to structure them : To find a dynamic model able to support the diversity of the form of its data, the exponential growth of their quantity and thus the scalability and finally ensure the semantics of these data by adapting to the constraints of the technological tools that are our concern.

Keywords

Big Data, Internet of Things (IoT), NoSQL databases, Hadoop, Apache HBase, Firebase, Scalability, Structured, Semi-structured, Unstructured, Semantics, Volume, Velocity, Variety.

Motivations, Problématique et Objectifs

Depuis la fin des années 1980, Internet a évolué de manière spectaculaire. La dernière étape est l'utilisation de ce réseau mondial pour la communication avec des objets ou entre objets, évolution nommée Internet des Objets (IoT pour Internet of Things). L'Internet des Objets (IoT) rend les objets qui nous entourent intelligents en leur offrant la faculté de communiquer entre eux ou avec le nuage (cloud). L'évolution de l'IoT est rapide : depuis 2014, le nombre d'objets connectés est supérieur au nombre d'humains connectés et il est prévu que plus de 50 milliards d'objets seront connectés en 2025.

L'Internet des objets est une composante fondamentale de la future smart city : pour relever des données et piloter des usages, les villes doivent connecter des objets. Ces technologies offrent des perspectives considérables pour les villes. L'Internet des objets (IoT) relie les appareils électroniques (autres que les ordinateurs et les smartphones) à Internet pour un suivi et une gestion efficace des activités quotidiennes. Dans le domaine du développement urbain, l'IoT est l'un des éléments constitutifs importants de la création d'une infrastructure intelligente pour administrer, servir et accompagner la population urbaine en constante augmentation.

Avec l'essor des grandes plateformes web et la multiplication des objets connectés à travers le monde s'accompagnent d'une croissance exponentielle des données créées sur la toile. En plus du volume des informations, elles sont également très diverses, non structurées au sens où elles ne se présentent pas sous la forme de lignes et de colonnes et les systèmes de gestion de données traditionnels, relationnels et transactionnels, basés sur le langage SQL, ont montré leurs limites. Depuis quelques années, de nouvelles approches du stockage et de la gestion des données sont apparues, qui permettent de s'astreindre de certaines contraintes, en particulier de scalabilité, inhérentes au paradigme relationnel. Regroupées derrière le vocable NoSQL.

Plus que des technologies de remplacement intégral des outils relationnels, le NoSQL correspond le plus souvent à des approches qui complètent les outils existants, pour en combler les faiblesses. Ainsi, on observe de plus en plus souvent la traduction "Not Only SQL" au lieu de "No SQL". Ces technologies, portées par des acteurs majeurs du Web comme Google, Facebook, Twitter, ont rapidement acquis une légitimité réelle dans le domaine de la manipulation de volumétries de données très importantes et ont rapidement gagné tant en maturité qu'en notoriété.

Problématique

C'est dans ce contexte technologique récent et très riche en concepts qu'il nous a été demandé de réfléchir à la conception et à l'implementation d'une base de données NoSQL. Pour ce faire il fallait donc imaginer un cadre de travail qui permettera de mettre en évidence une problématique multiple. On se propose alors d'étudier et d'essayer de répondre aux problèmes suivants :

1. Dans le cadre d'une ville intelligente qui véhicule un gros volume de données on se propose de réfléchir à donner de la valeur aux données utiles à un système de surveillance dans le contexte d'un tremblement de terre.
2. Sachant que les bases de données SQL sont limités concernant le critère de scalabilité, comment se fait le passage à l'échelle dans le cadre des bases de données NoSQL. Il serait donc utile de comparer les deux concepts SQL/NoSQL.
3. Avant de penser au stockage des bases de données NoSQL, il est impératif de réfléchir à la meilleure façon de les structurer sachant qu'à ce niveau se posera le problème de perte de la sémantique des données.
4. Sachant qu'il existe aujourd'hui un nombre important de base de données NoSQL qui sont tous à l'étude chacun avec ses avantages et ses limites, il nous a été confié l'étude et la critique de deux outils NoSQL à savoir HBase et Firebase.
5. Essayer d'implémenter notre base de données NoSQL en s'adaptant aux contraintes des outils technologiques qui sont à notre porté.

Objectifs

Dans cette optique, on s'est fixé en accord avec nos encadreurs d'atteindre les objectifs suivants :

1. Faire une recherche Bibliographique concernant le Big Data, L'IoT, et le NoSQL.
2. Etudier, Analyser et critiquer les outils HBase et Firebase et en rédiger une synthèse.
3. Penser et imaginer une ville intelligente du point de vue données et scénarios d'alerte dans le contexte d'un tremblement de terre.
4. Récolter et structurer ces données en NoSQL et modéliser les scénarios d'alerte.
5. Implémenter notre base de données sous HBase et Firebase.
6. Faire une modélisation E-A de la base de données dans un but comparatif.
7. Essayer de dresser des perspectives.

Dans le cadre de notre projet de fin d'études, nous avons scindé le présent mémoire en trois parties :

1. ***Première partie Etat de l'art :*** Dans cette partie nous allons parler des quatres chapitres qui nous permetterons de mettre en évidence le contexte de ce mémoire.
   - ***Chapitre 1 : Big Data.***
   - ***Chapitre 2 : Internet des Objets (IoT).***
   - ***Chapitre 3 : Les bases de données NoSQL.***
   - ***Chapitre 4 : Apache HBase et Firebase.***
2. ***Deuxième partie Conception :*** Après l'étude du contexte, la phase analyse et conception vient éclaircir les objectifs à atteindre et décrire le fonctionnement futur de notre solution. Ainsi, à travers cette partie nous allons décrire, planifier le fonctionnement en modélisant les différents objectifs à atteindre.
   - ***Chapitre 5 : Conception.***
3. ***Troisième partie Réalisation :*** À travers cette partie nous venons présenté les résultats obtenus après concrétisation de tout ce qui a été planifié, modélisé dans les parties précédentes. En première partie nous parlerons du choix des outils technologiques qui nous ont permis de réaliser la solution, en seconde partie nous allons présenter notre solution.
   - ***Chapitre 6 : Choix des outils techniques.***
   - ***Chapitre 7 : Implémentation.***

Première partie

Etat de l'art

Chapitre 1

Big Data

Introduction

Depuis une vingtaine d'années, les données générées n'ont fait que s'accroître. Nous procréons environ 2,5 trillions d'octets de données tous les jours à travers les réseaux sociaux ou encore le cloud. Nous constatons également une plus grande variété de données qu'auparavant sachant qu'un ensemble de données peut se composer à lui seul de vidéo, audio, texte, données de capteur...etc. Avec une telle quantité accrue de données, nous sommes face à une plus-value potentielle, ce qui a donner naissance au domaine du "Big Data". Il s'agit d'un concept permettant de stocker un nombre indicible d'informations sur une base numérique.

Selon l'étude Data Age 2025 [1] des analystes de IDC [2], cette croissance sera multipliée par 5,3 d'ici 2025 pour atteindre 175 Zettaoctets (Zo) soit 175 milliards de téraoctets.

- **Quelques statistiques sur le Big Data**

  La donnée représente l'or noir du 21 siècle. En effet après le siècle du pétrole, nous entrons dans l'ère de la donnée.

  Le Big Data est promis à un futur radieux. Le marché est en pleine croissance, le volume de données explose et les entreprises sont de plus en plus nombreuses à adopter les technologies analytiques.

  La figure ci-dessous permet de présenter la quantité de données générées en 60 secondes d'Internet entre 2018 et 2019

1. L'étude Data Age 2025 est une étude sur la digitalisation dans le monde réalisée par l'IDC, établi un comparatif de la digitalisation dans quatre régions (Asie/Pacifique, dont Japon, mais hors Chine) ; Chine ; États-Unis ; EMEA (Europe, Moyen-Orient et Afrique).
2. International Data Corporation (IDC) est le premier groupe mondial de conseil et d'études sur les marchés des technologies de l'information.

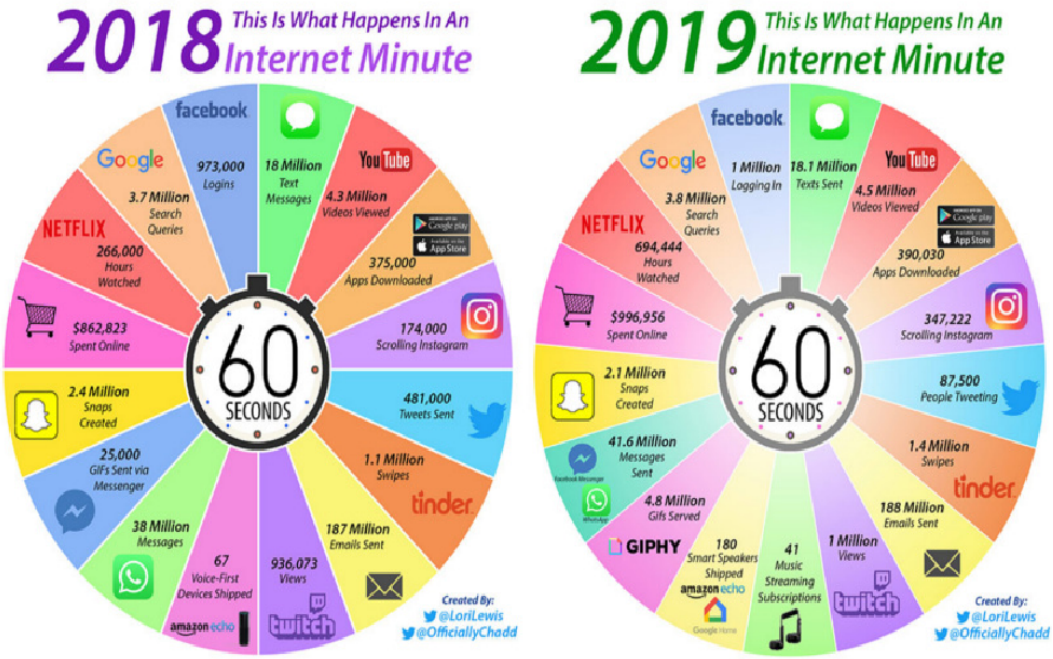

FIGURE 1.1 – 1 minute d'internet

Si nous nous intéressons de plus près à la figure ci-dessus, nous pouvons bien constater que nous nageons dans un océan de données où le niveau de l'eau augmente rapidement. En effet, à chaque minute d'Internet, sont envoyés plus de 38 millions de messages en 2018 et 47 millions en 2019 . Chaque minute, Google enregistre près de 3.7 millions de requêtes différentes sur son moteur de recherche en 2018 et 3.8 millions en 2019. Facebook enregistre près d'un millions de « logins ».

Face à cette croissance exponentielle du volume de données, les entreprises sont confrontées à certaines problématiques qui sont celles de savoir comment collecter, stocker, analyser et exploiter ces grands volumes de données pour créer de la valeur ajoutée. Tout l'enjeu, pour les entreprises et les administrations, consiste à ne pas passer à côté d'informations précieuses noyées dans la masse. C'est là qu'intervient la technologie du "Big Data", qui repose sur une analyse très fine de masses de données.

1.1 Concepts généraux

Comme chaque domaine de connaissance en expansion, la terminologie naissante Big Data a permis l'apparition de nouveaux concepts. Dans cette section nous allons énumerer ces concepts et définitions se rapportant au domaine du "big data", intérêts, contraintes et caractéristiques.

1.1.1 Quelques définitions liées au Big Data

Défition 1. Littéralement, ces deux termes "Big" et "Data" signifient mégadonnées, grosses données ou encore données massives. Ils désignent un ensemble très volumineux de données qu'aucun outil classique de gestion de base de données ou de gestion de l'information ne peut vraiment travailler. Ce sont les informations provenant de partout : messages

que nous nous envoyons sur les médias sociaux, vidéos que nous publions,capteurs utilisés pour collecter les informations climatiques, signaux GPS, enregistrements transactionnels d'achats en ligne et bien d'autres encore.[1]

Définition 2. Le Big Data est une exploration de très vastes d'ensembles de données pour obtenir des renseignements utilisables. Il s'agit d'un ensemble de technologies, et d'outils permettant à une organisation de collecter, stocker et analyser rapidement de larges quantités de données hétérogènes afin d'en extraire des informations pertinentes.[2]

Définition 3. Inventé par les géants du web [3], le Big Data se présente comme un ensemble de solutions dessinées pour permettre à tout le monde d'accéder en temps réel à des bases de données géantes. Il vise à proposer une amélioration aux solutions traditionnelles classiques de bases de données et d'analyse afin de traiter un volume très important de données, en temps réel et avec une très grande diversité de sources et de formats.[1]

> ***Remarque :*** *Cependant, aucune définition précise ou universelle ne peut être donnée au Big Data. Etant un objet complexe polymorphe, sa définition varie selon les communautés qui s'y intéressent en tant qu'usager ou fournisseur de services.*

1.1.2 Intérêts du Big Data

Dans une même entreprise, plusieurs départements peuvent être concernés par la mise en place et l'utilisation du Big Data : Informatique, commercial, marketing.... Les services marketing font le plus appel au Big Data. Ils sont considérés comme précurseurs dans la mise en place de nouvelles stratégies, ainsi les entreprises pourront[3] :

- Améliorer la prise de décision.
- Réduire les coûts d'organistaions
- Développer la réactivité et l'interactivité à l'égard des clients.
- Améliorer les performances opérationnelles.
- Augmenter la productivité.
- Recruter des gens plus intelligents pour des emplois plus intelligents.
- Fidéliser la marque.
- Des données pour les stratégies de vente.
- Mettre en place des prix compétitifs.
- Anticiper les besoins et adapter les campagnes marketing
- Mieux comprendre les comportements des prospects et des clients
- Améliorer l'expérience client

3. Les géants du web ou GAFAM Google, Apple, Facebook, Amazon et Microsoft — qui sont les cinq grandes firmes américaines (fondées entre le dernier quart du xxe siècle et le début du xxie siècle) qui dominent le marché du numérique, parfois également nommées les Big Five, ou encore ≪ The Five ≫. Cet acronyme correspond au sigle GAFA initial, auquel le M signifiant Microsoft a été ajouté.

- Améliorer l'efficacité des campagnes publicitaires, qu'elles soient en ligne ou non
- Affiner le ciblage des prospects et des clients.
- Analyser le comportement des prospects et des clients à 360 : achats en magasin et en ligne, habitudes de navigation sur internet, préférences renseignées sur les réseaux sociaux...

1.1.3 Les contraintes du Big Data

Cette technologie en plein essor, toute brillante et impressionnante offre un nombre considérable d'avantages et marque une avancée révolutionnaire en plus d'être pleine de promesses pour les entreprises. Mais, comme toute avancée technologique, le Big Data a ses limites et cela nous amène à nous questionner sur les risques majeurs potentiels de vulnérabilité du Big Data. On en citera parmis ces risques :[4]

- Les personnes qui ont les mêmes caractéristiques que celles qui ont un comportement à risque peuvent se faire refuser l'accès à des services (prêts, cartes de crédit, etc.), même si leur dossier personnel est impeccable (cela peut aussi toucher les embauches, l'accès aux études supérieures, et bien d'autres domaines où on utilise le Big Data).
- Certaines caractéristiques personnelles peuvent être rendues publiques uniquement grâce à des corrélations. ***Par exemple,*** *les "j'aime" de Facebook ont permis à cette entreprise de déterminer correctement 82 % la religion de ses utilisateurs, et 75 % leur consommation de drogues.*
- Ces renseignements peuvent servir aux fraudeurs en permettant d'identifier les personnes les plus vulnérables. rappelons-nous du scandale « Prism »[4], programme américain de surveillance électronique par la collecte de renseignements à partir d'Internet. La transmission de données de ce programme de surveillance, qui a été découvert et que le service administratif américain minimise, remet totalement en cause le respect de la vie privée des individus.
- L'analyse des données massives peut faire augmenter les prix dans les quartiers pauvres (par exemple quand le volume d'achats faits sur Internet y est moins élevé).

Remarque : *Finalement, nous ne pouvons pas nier que le Big Data peut se révéler dangereux pour les individus et la société en raison des dérives et des failles qu'il peut induire. En revanche, le Big Data marque une avancée incontestable et il convient de renforcer la surveillance de l'exploitation de ces données pour en éviter les dérives.*

4. PRISM, également appelé US-984XN1, est un programme américain de surveillance électronique par la collecte de renseignements à partir d'Internet et d'autres fournisseurs de services électroniques. Ce programme classé, relevant de la National Security Agency (NSA), prévoit le ciblage de personnes vivant hors des États-Unis.

1.1.4 Les caractéristiques du Big Data

Selon le Gartner [5], le Big Data regroupe une famille d'outils qui répondent à une triple problématique de : **Volume**, **Variété** et **vélocité** dite règle des 3V.

Il s'agit notamment d'un **Volume** de données considérable à traiter, une grande **Variété** d'informations (venant de diverses sources, non-structurées, organisées, Open...), et un certain niveau de **Vélocité** à atteindre, autrement dit de fréquence de création, collecte et partage de ces données.

Ces 3V sont les caractéristiques majeures des mégadonnées. Aussi une définition plus pertinente des mégadonnées pourrait alors être : ***" des données qui sont trop volumineuses ou ayant une arrivée trop rapide ou une variété trop grande pour permettre de les ranger directement dans des bases de données traditionnelles ou de les traiter par les algorithmes actuels "***[B1]

Certains auteurs parlent du critère des 4V voir même des 10V à savoir :

- La Véracité.
- La Valeur.
- La Validité.
- La Variabilité.
- La Volatilité.
- La Visualisation.
- La Vulnérabilité.

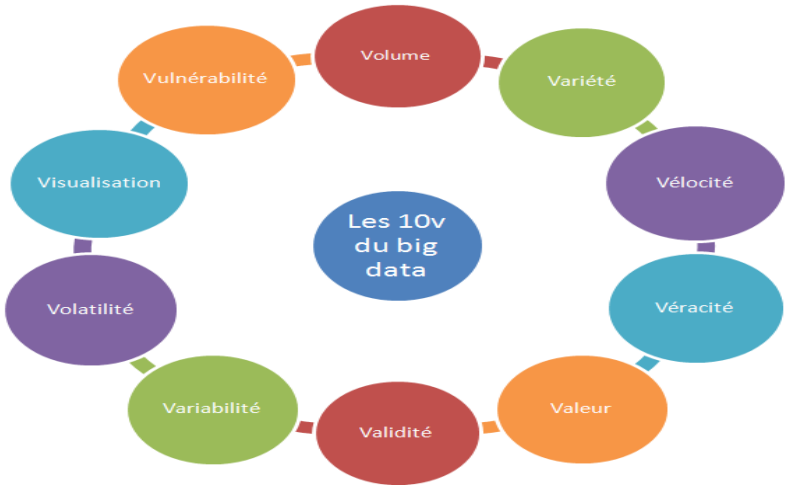

FIGURE 1.2 – Les 10V du big data

5. Le Gartner est une entreprise américaine de conseil et de recherche dans le domaine des techniques avancées. Elle mène des recherches, fournit des services de consultation, tient à jour différentes statistiques et maintient un service de nouvelles spécialisées.

1. **Le Volume :**
   Le caractère volume est certainement celui qui est le mieux décrit par le terme Big de l'expression Big Data. Volume fait référence à la quantité d'informations, trop volumineuse pour être acquise, stockée, traitée, analysée et diffusée par des outils standards. Ce caractère peut s'interpréter comme le traitement d'objets informationnels de grande taille ou de grandes collections d'objets.[B1]

   L'exemple de Facebook : *Facebook compte plus d'utilisateurs que la Chine n'a d'habitants. Chacun de ces utilisateurs y stocke de nombreuses photos. Facebook stocke ainsi environ 250 milliards d'images.*

2. **La Variété :**
   En ce qui concerne le Big Data, nous devons non seulement gérer des données **structurées**, mais également des données **semi-structurées** et surtout **non structurées**. Aujourd'hui, les informations utiles proviennent aussi d'autres sources : documents, courrier électronique, réseaux sociaux... et prennent de nouvelles formes : texte, image, vidéo... L'analyse des "big data" concerne aussi ces données, pour lesquelles les moyens disponibles sont encore en émergence.[5]

   Exemple : Les messages électroniques : *Actuellement on éstime à environs un millions de messages électroniques émis par jour. Aucun de ces messages ne sera exactement comme un autre.*

   Chacun se composera de l'adresse électronique de l'expéditeur, d'un destinataire et d'un horodatage. Chaque message comportera un texte et éventuellement des pièces jointes qui ne sont généralement pas structurés et très variés à savoir : Les photos, vidéos, enregistrements audio, messages électroniques, documents, livres, présentations et bien d'autre encore.

3. **La Vélocité :**
   Dernière dimension, tout aussi importante que les précédentes, la vélocité traduit la capacité à produire rapidement les données et à les transformer en temps utile pour leurs utilisateurs. L'exercice, déjà difficile dans un contexte "classique", prend toute sa valeur lorsqu'il doit être appliqué à d'immenses volumes de données de toutes sortes.[5]

   Exemple : *Google traite en moyenne plus de "40 000 requêtes de recherche par seconde", ce qui représente environ 3,5 milliards de recherches par jour.*

4. **La Véracité :**
   le caractère complémentaire "véracité" fait référence à la provenance ou à la fiabilité de la source de données, à son contexte et à son importance pour l'analyse qui en découle. Il comprend les problèmes de valeurs aberrantes ou manquantes, mais aussi à la confiance que l'on peut avoir dans les données. S'il existe des critères permettant de qualifier la qualité des données, dans le cas de big data, cette vérification de la qualité est rendue difficile voire impossible du fait du volume, de la variété et de la

vélocité spécifiques au Big Data.[B1]

Exemple : *Selon un rapport traitant de la protection des données privées publié en 2015 par la société de cybersécurité Symantec*[6]*, 57 % des européens se déclarent inquiets quant à la sécurité de leurs informations personnelles, 81 % estiment que leurs données ont une valeur supérieure à mille euros et 31 % n'hésitent plus à communiquer aux systèmes de fausses données pour protéger leurs données personnelles. Des applications ont été développées pour créer de fausses données dans le but de tromper les applications Android qui sont parfois ressenties comme trop intrusives par l'utilisateur. Xprivacy est un outil qui permet de nourrir les applications Android avec de faux contacts, de fausses coordonnées géographiques, de faux dictionnaires user, de faux presse-papiers, de faux historiques d'appels, de faux SMS. L'objectif affiché par Xprivacy est de créer de fausses données pour mieux protéger sa vie privée.*

5. **La Valeur :**
   Le caractère "valeur" fait référence à la potentialité des données. C'est le fait de dégager de nouvelles données de nouvelles connaissances à partir des données existantes. en particulier en termes économiques. Il est ainsi associé à l'usage qui peut être fait de ces mégadonnées, de leur analyse, notamment d'un point de vue économique. L'analyse de ces mégadonnées demande une certaine expertise tant liée à des méthodes et techniques en statistique, en analyse de données, que de domaine pour l'interprétation de ces analyses.[B1]

   L'exemple du zoo de Cincinnati[7] ***:*** *Confronté à des difficultés de rentabilité, le zoo américain de Cincinnati s'est orienté vers le traitement big data de ses données clients et des données issues de capteurs déployés au sein des attractions et bâtiments du parc. L'image en temps réel des comportements de la clientèle du zoo a permis d'augmenter de 25 % les dépenses des visiteurs en apportant plus de 350 000 dollars de recettes supplémentaires par an. La compréhension fine des données clients a été appliquée à l'optimisation des ressources humaines du zoo et a libéré du temps pour le personnel désormais disponible sur d'autre postes de rentabilité. Le budget de l'entreprise a ainsi retrouvé son équilibre.*

6. **La Validité :**
   Semblable à la véracité, la validité fait référence à la précision et à la correction des données pour l'usage prévu. Les avantages de l'analyse de données volumineuses ne dépendent que de ses données sous-jacentes. Il est donc conseillé d'adopter de bonnes pratiques de gouvernance des données pour garantir une qualité des données cohérente et une transmission non corrompue des données.[B2]

6. Symantec Corporation NASDAQ : est une société américaine fondée en 1982 spécialisée dans les logiciels informatiques. Elle a son siège social à Mountain View en Californie. Symantec est spécialisé dans l'édition de logiciels utilitaires (notamment liés à la sécurité et à la protection des données) pour PC tournant sur plateforme Microsoft. Elle s'est développée grâce à l'acquisition de sociétés tierces et le développement commercial de ses produits.

7. Le Zoo et Jardin botanique de Cincinnati est un parc zoologique américain situé dans l'Ohio, à Cincinnati. Ouvert en 1875, il est le deuxième plus vieux parc zoologique des États-Unis.

Exemple : *La date d'une transaction est « 02/07/1994 » alors que l'activité de la société a débuté en 2000.*

7. **La Variabilité :**
   La variabilité fait référence à différentes choses. L'une est le nombre d'incohérences dans les données. Celles-ci doivent être détectées par des méthodes de détection d'anomalies et de valeurs aberrantes afin de permettre toute analyse significative.

   Les mégadonnées sont également variables en raison de la multitude de dimensions de données résultant de multiples types et sources de données disparates. Comme elle peut également faire référence à la vitesse incohérente à laquelle les données volumineuses sont chargées dans la base de données.[B2]

   L'exemple du supercalculateur[8] ***:*** *Brian Hopkins, analyste principal de Forrester*[9]*, a cité le supercalculateur Watson*[10] *comme un excellent exemple de cela. Pour participer au jeu télévisé Jeopardy*[11]*, Watson devait "être capable de comprendre l'énoncé des questions, buzzer pour prendre la main,disséquer une réponse dans son sens pour déterminer quelle était la bonne question». Les mots n'ont pas de définitions statiques et leur signification peut varier énormément dans le contexte.*

8. **Volatilité :**
   En raison de la vitesse et du volume des mégadonnées du Big Data, leur volatilité doit être examinée avec soin. Il s'agit de la durée de vie des données générées, autrement dit, pendant combien de temps elles sont valides. En fonction des domaines, la volatilité des Big Data diffère beaucoup. Ce qui tend à en faire un élément important à prendre en compte d'un point de vue opérationnel, mais qui ne les définit pas d'un point de vue théorique. Pour celà il faut définir des règles pour la disponibilité et la mise à jour des données, garantir une récupération rapide des informations en cas de besoin et s'assurer qu'ils sont clairement liés aux besoins et aux processus de l'entreprise.[6]

---

8. Un superordinateur ou supercalculateur est un ordinateur conçu pour atteindre les plus hautes performances possibles avec les techniques connues lors de sa conception, en particulier en ce qui concerne la vitesse de calcul.La science des superordinateurs est appelée « calcul haute performance » (en anglais : High-Performance Computing ou HPC). Cette discipline se divise en deux : la partie hardware (conception électronique de l'outil de calcul) et la partie software (adaptation logicielle du calcul à l'outil). Ces deux parties font appel à des champs de connaissances différents.

9. Forrester Research est une entreprise indépendante qui fournit à ses clients des études de marché sur l'impact des technologies dans le monde des affaires. Forrester Research a 19 implantations géographiques à travers le monde, dont huit centres de recherche.

10. Le super calculateur d'IBM, surnommé «Watson» est un système d'information capable de comprendre le langage naturel, notre langage, et d'appréhender toute la subtilité d'une réflexion humaine. C'est une machine capable de simuler un raisonnement humain et d'engranger un savoir gargantuesque, l'équivalent d'1 million de livres.

11. La participation à Jeopardy : Quatorze ans après la confrontation entre Deep Blue et le champion d'échecs Garry Kasparov, qui avait vu la défaite de ce dernier4, les équipes d'IBM font participer Watson au célèbre jeu télévisé américain Jeopardy !. Pour cela, Watson doit être capable de comprendre l'énoncé des questions, buzzer pour prendre la main, trouver les réponses en quelques secondes, et, grâce à un système de synthèse vocale, énoncer les réponses et de choisir le thème et le montant de la prochaine question, conformément aux règles du jeu.

Par exemple : *certaines entreprises peuvent ne conserver que l'année la plus récente de leurs données et transactions client dans leurs systèmes d'entreprise. Cela garantira une récupération rapide de ces informations si nécessaire. S'ils doivent consulter une année antérieure, l'équipe informatique peut être amenée à restaurer les données d'un stockage hors connexion pour répondre à la demande. Avec le Big Data, ce problème est amplifié. Si le stockage est limité, il faut examiner les sources de données volumineuses pour déterminer ce qu'on doit collecter et combien de temps pour le conserver.*

9. **Visualisation :**
   C'est la partie difficile du Big Data qui rend toute cette énorme quantité de données compréhensible. Avec la bonne analyse et visualisation, les données brutes peuvent être utilisées. Visualisation signifie graphiques complexes pouvant inclure plusieurs variables des données tout en restant compréhensibles et lisibles.[B3]

   Dans le monde du Big Data, les outils et technologies de visualisation de données sont indispensables pour analyser d'énormes volumes d'informations et prendre des décisions en s'appuyant sur les données.

   Prenons l'exemple *du tableau suivant qui fait apparaître deux séries de chiffres : le chiffre d'affaires "France" et le chiffre d'affaires "Reste du monde". La lecture de ce tableau et sa signification ne sont pas immédiates.*

Tableau 1 : ÉVOLUTION DU CA PAR RÉGION

kEur	Janv	Fevr	Mars	Avril	Mai	Juin	Juillet	Août	Sept	Oct	Nov	Déc
France	10000	12000	14000	13000	15444	17028	15000	15804	18000	17500	19000	20958
Reste du monde	3444	3816	4038	3558	3864	4074	3558	2000	3594	3498	3612	4140
	13444	15816	18038	16558	19308	21102	18558	17804	21594	20998	22612	25098

FIGURE 1.3 – Evolution du chiffre d'affaires par région.

Mais si nous représentons les séries de chiffres sous forme graphique (ci-dessous), on comprend en un coup d'œil que le chiffre d'affaires "France" progresse et que le chiffre d'affaires "Reste du monde" stagne.

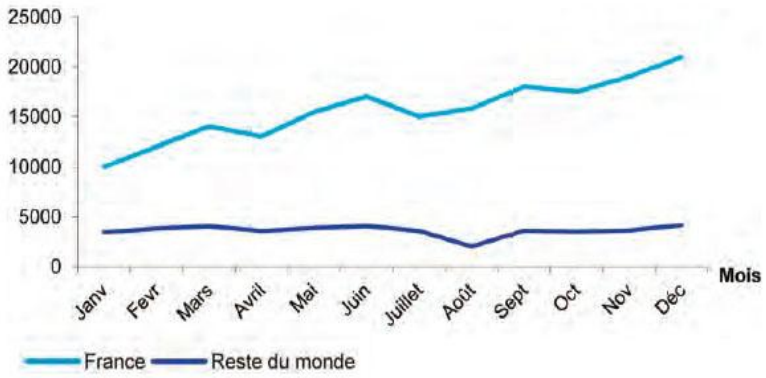

FIGURE 1.4 – Graphe d'évolution du chiffre d'affaires par région.

10. **Vulnérabilité :**
    Le Big Data apporte de nouveaux problèmes de sécurité. Après tout, une violation de données avec le Big Data est une grande violation. Malheureusement, il y a eu beaucoup de violations de données massives.[B2]

    Exemple rapporté par CRN[12] **:** *En Mai 2016, "un pirate informatique appelé Peace a posté des données à vendre sur le dark web*[13]*, qui auraient notamment fourni des informations sur 167 millions de comptes LinkedIn et 360 millions de courriels et de mots de passe pour les utilisateurs de MySpace*[14] *".*

En gros, lorsqu'il est question de Big Data, cela ne comprend pas uniquement l'immense quantités d'information composant des trésors de connaissance. Cela comprend également le travail d'analyse de ces données, la façon dont nous sélectionnons le trésor. Dans l'univers du Big Data, **les données** et **les analyses** sont **interdépendantes** : l'une sans l'autre est insensé, mais **leur puissance combinée est illimitée**.

1.2 Les technologies du Big Data

Avec la venue et la croissance du Big Data, le volume de données à gérer par les applications a explosé, et les systèmes de gestion de données traditionnels, relationnels et transactionnels, basés sur le langage SQL, ont montré leurs limites.

Depuis quelques années, de nouvelles approches de stockage et de la gestion des données sont apparues permettant de s'astreindre de certaines contraintes, en particulier de scalabilité[15], inhérentes au paradigme relationnel.

12. anciennement Computer Reseller News, est un magazine commercial américain traitant d'informatique. Il cible principalement le milieu de la revente informatique
13. Le dark web Internet clandestin ou encore l'Internet sombre est le contenu du World Wide Web qui existe sur les darknets, des réseaux overlay qui utilisent l'internet public mais sont seulement accessibles via des logiciels, des configurations ou des autorisations spécifiques.
14. est un site web de réseautage social qui met gratuitement à disposition de ses membres un espace web personnalisé, permettant de présenter diverses informations personnelles et d'y faire un blog.
15. La scalabilité désigne la capacité d'un produit à s'adapter à un changement d'ordre de grandeur de la demande (montée en charge), en particulier sa capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande.

Ces nouvelles créations technologiques peuvent globalement être catégorisées en deux familles que nous présenterons comme suit :

- **Les technologies de stockage.**
- **Les technologies de traitement.**

1.2.1 Les technologies de stockage

dans une première partie nous verrons, **les technologies de stockage**, portées plus particulièrement par le déploiement du **Cloud Computing**.

1. **Le Cloud Computing**
   Jusqu'ici nous avons bien vu qu'avec le Big Data, les sources de données se sont largement diversifiées et sont devenues relativement hétérogènes et ont été surtout localisées sur Internet, ces informations sont produites de façon continue et à un rythme soutenu. Différentes nouvelles solutions ont vu le jour pour traiter ces volumes d'informations et ces flux de données, toutes ces solutions reposent sur un stockage distribué (partitionné) des données sur les clusters [16].

   (a) **Définition**
   Le nuage (cloud) est un ensemble de matériels, de raccordements réseau et de logiciels fournissant des services sophistiqués que des individus et des collectivités peuvent exploiter à volonté depuis n'importe où. Au lieu d'obtenir de la puissance de calcul par acquisition de matériel et de logiciel, dans le Cloud Computing, le consommateur utilise une puissance de calcul mise à sa disposition sur une architecture d'un fournisseur via Internet.[7]

16. Un cluster est une grappe de serveurs sur un réseau, appelé ferme ou grille de calcul.

(b) **Principe de fonctionnement d'un Cloud**
Pour comprendre le fonctionnement du Cloud dans sa partie plus technique, il faut le segmenter en deux éléments essentiels qui le composent : **Virtualisation** et **Data center**.

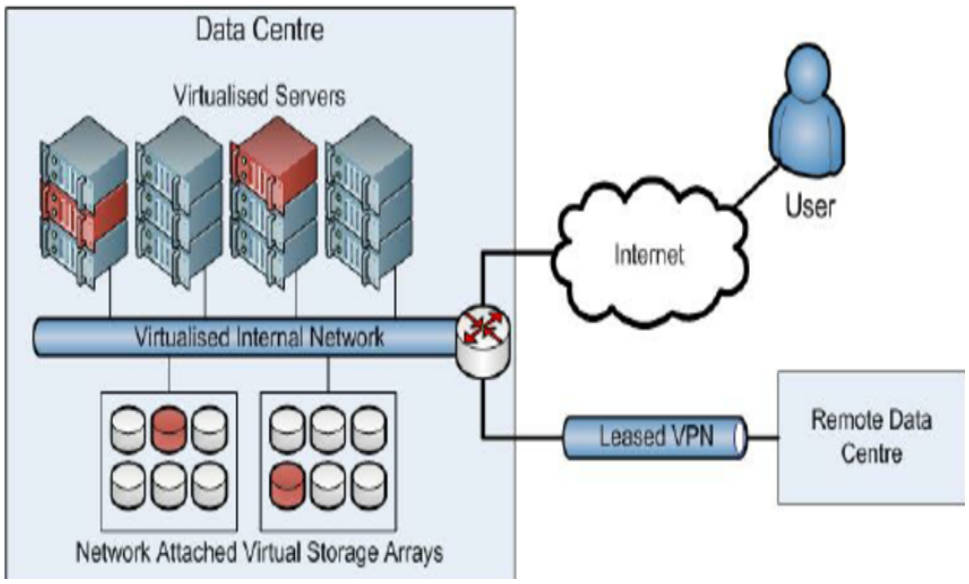

FIGURE 1.5 – La virtualisation des serveurs au cœur d'un data center.

- **La virtualisation :**

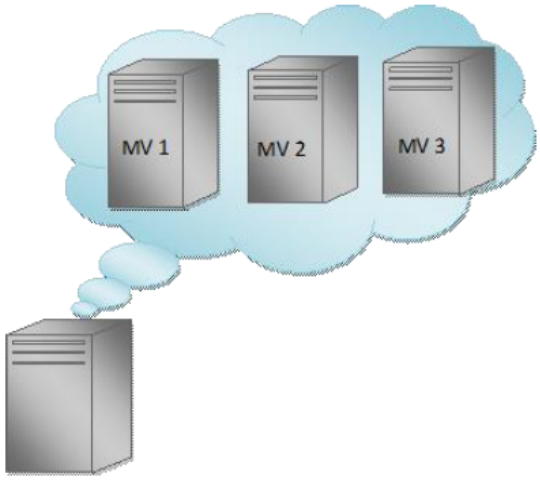

FIGURE 1.6 – La virtualisation d'un serveur.

Même s'il existe plusieurs types de virtualisation, telle que la virtualisation des postes clients ou la virtualisation des super calculateurs, la forme **la plus populaire** de virtualisation dans le cloud est la **virtualisation des serveurs**. Ainsi, la virtualisation consiste à dématérialiser le comportement et les données d'un serveur ou d'une machine, de façon à faire tourner plusieurs de ces instances dématérialisées sur un même serveur physique. De cette façon, les différentes instances créées se partagent les ressources du ser-

veur physique. Cela permet une plus grande modularité dans la répartition des charges, une facilité dans l'administration des serveurs et une réduction des coûts.[8]

- **Les Data Center :**

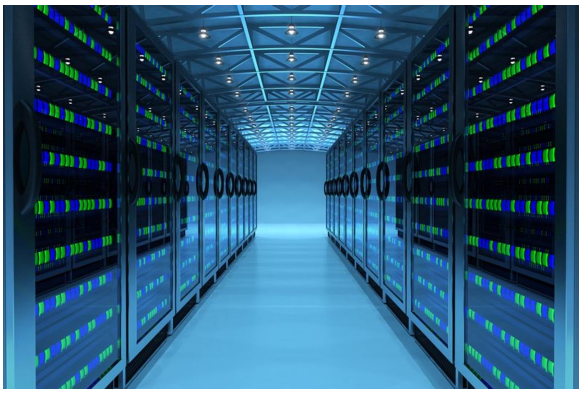

FIGURE 1.7 – Un data center.

Un data center (En français, centre de traitement de données) est un site physique réel sur lequel se trouvent regroupés des équipements constituants le système d'information de l'entreprise (serveurs, baies de stockage, équipements réseaux et de télécommunications, etc). Il peut être interne ou dans des lieux géographiquement éloignés et reliés par des réseaux hauts débits à l'entreprise. Il comprend en général un contrôle sur l'environnement (Système de prévention contre l'incendieune, alimentation d'urgence ainsi qu'une sécurité physique élevée).

(c) **Le stockage dans le Cloud**

Le stockage des mégadonnées dans le cloud a tout son sens. En effet, cette architecture est prévue pour monter facilement en charge ("scalability" en anglais) notamment par une mutualisation de ressources hétérogènes.

Cette standardisation des infrastructures a permis de réaliser des calculs multiprocesseurs, parallèles, sur des machines et des systèmes d'exploitation standard, très faciles à accèder. Ce n'est pas la puissance de calcul des serveurs qui a changé, c'est la façon de monter en charge. En revanche, Si le cloud permet d'appréhender la caractéristique de volume des mégadonnées, les caractéristiques de variété et de vélocité ne le sont pas, le cloud étant davantage un support de stockage qu'une solution de gestion de données.[9]

(d) **Quelques exemples de Cloud Computing**

Le Cloud est désormais ancré dans presque toutes les tâches que nous accomplissons sur ordinateur.

- **Google Drive :** Google Drive est un pur service Cloud Computing. Il propose un stockage en ligne, et fonctionne avec les applications Cloud Google Docs, Google Sheets et Google Slides. Ce service est accessible depuis un ordinateur, depuis une tablette, ou même depuis un smartphone, au même

titre que les applications mobiles Docs et Sheets. La plupart des services Google peuvent d'ailleurs être classés dans la catégorie du Cloud Computing. C'est le cas de Gmail, Google Calendar, et Google Maps par exemple.

- **Amazon Cloud Drive :** Amazon propose essentiellement le stockage de musique au format MP3 et de photos. Les abonnés Amazon Prime bénéficient d'une capacité de stockage illimitée. Amazon Cloud Drive sert également à stocker le contenu acheté pour Kindle. En résumé, cette plateforme sert à stocker tout le contenu numérique acheté auprès d'Amazon.

- **Apple iCloud :** Principalement utilisé pour le stockage en ligne, le backup, Apple iCloud est également utile pour la synchronisation des mails, des contacts, ou encore du calendrier. Toutes les données sont disponibles sur iOS, Mac OS, ou sur les appareils Windows depuis le panneau de contrôle iCloud. Apple propose également des versions Cloud de son traitement de texte Pages, de sa feuille de calcul Spreadsheet, et de son logiciel de présentation Keynote pour tous les utilisateurs d'iCloud.

1.2.2 Les technologies de traitement

En seconde partie nous parlerons de l'arrivée **des technologies de traitement** ajustées, plus spécialement sur la mise au point de modes de calcul à haute performance **(MapReduce)** et la principale plateforme du Big Data **(Hadoop)** et enfin nous clôturons cette section avec le développement de nouvelles bases de données adaptées aux données non-structurées **(NoSQL)**.

1. **Hadoop**
   Hadoop est un framework logiciel open source de la fondation Apache [17] créé en réponse à la nécessité d'une infrastructure logicielle dédiée qui permette d'exécuter le schéma MapReduce de manière distribuée sur un cluster machines. Cette solution offre un espace de stockage massif pour tous les types de données, une immense puissance de traitement et la possibilité de prendre en charge une quantité de tâches virtuellement illimitée.[9]

17. La fondation Apache est une communauté décentralisée de développeurs qui travaillent sur ses projets open source. Les projets Apache sont caractérisés par un mode de développement collaboratif fondé sur le consensus ainsi que par une licence de logiciel ouverte et pragmatique.

Dans son principe, Hadoop se compose essentiellement de :

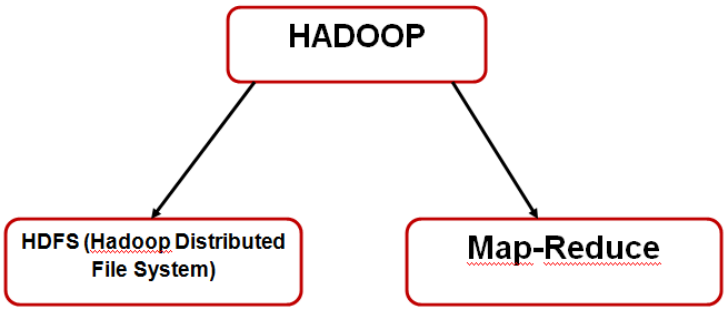

FIGURE 1.8 – Les composants d'Hadoop.

(a) **Système de gestion de fichiers HDFS (Hadoop Distributed File System) :** est un système de fichier distribué permettant de stocker et de récupérer des fichiers en un temps record sur un grand nombre de machines (noeuds) équipées de disques durs banalisés. Ce système augmente les possibilités de gestion de données du cluster HDFS Hadoop et permet donc de traiter le Big Data efficacement. Parmi ses principales fonctionnalités, on compte la possibilité de stocker des terabytes, voire des petabytes de données.[10]

(b) **Modèle de programmation Map-reduce :** Grâce au framework MapReduce, il permet de traiter les immenses quantités de données. Plutôt que de devoir déplacer les données vers un réseau pour procéder au traitement, MapReduce permet de déplacer directement le logiciel de traitement vers les données. Ce modèle fera l'objet de la deuxième technologie que nous allons présenter.

(c) Une collection d'outils spécifiques pour HDFS et Map Reduce : par exemple des API et des frameworks.

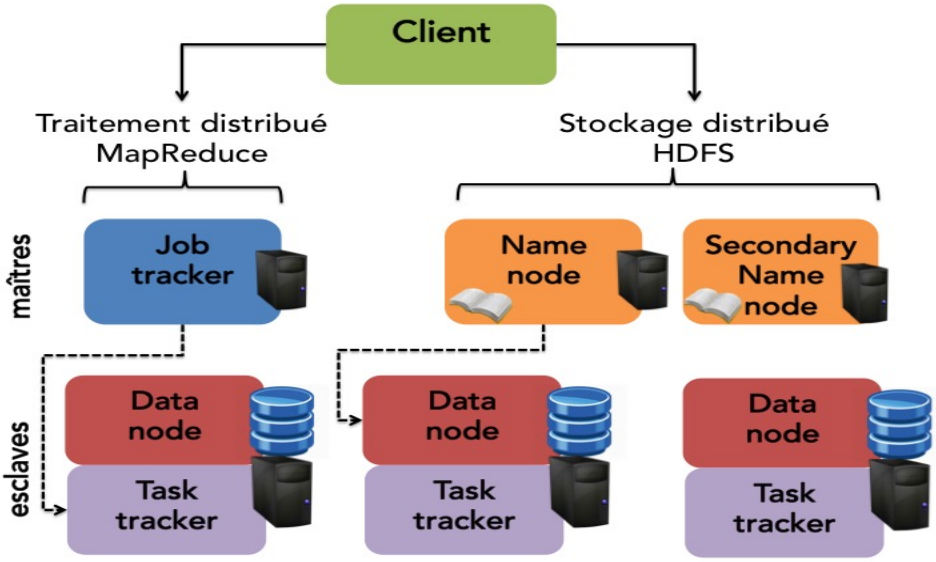

FIGURE 1.9 – L'environnement Hadoop.

2. **MapReduce**
   MapReduce est un modèle de programmation créé par Google pour le traitement et la génération de larges ensembles de données. Il permet le traitement résilient [18] et distribué d'ensembles de données non structurées massifs sur des clusters d'ordinateurs, au sein desquels chaque nœud possède son propre espace de stockage.

   (a) **Principe de MapReduce**
   Concrètement, le framework propose deux fonctionnalités principales. La fonction **Map** qui répartit le travail sur les différents nœuds du cluster. La fonction **Reduce** les organise et réduit les résultats fournis par chaque nœud en une seule réponse cohérente à une requête. Cela est rendu possible grâce au système de fichiers distribués HDFS d'Hadoop.

18. en informatique, la résilience est la capacité d'un système ou d'une architecture réseau à continuer de fonctionner en cas de panne.

Ce Framework est également constitué de plusieurs composants :

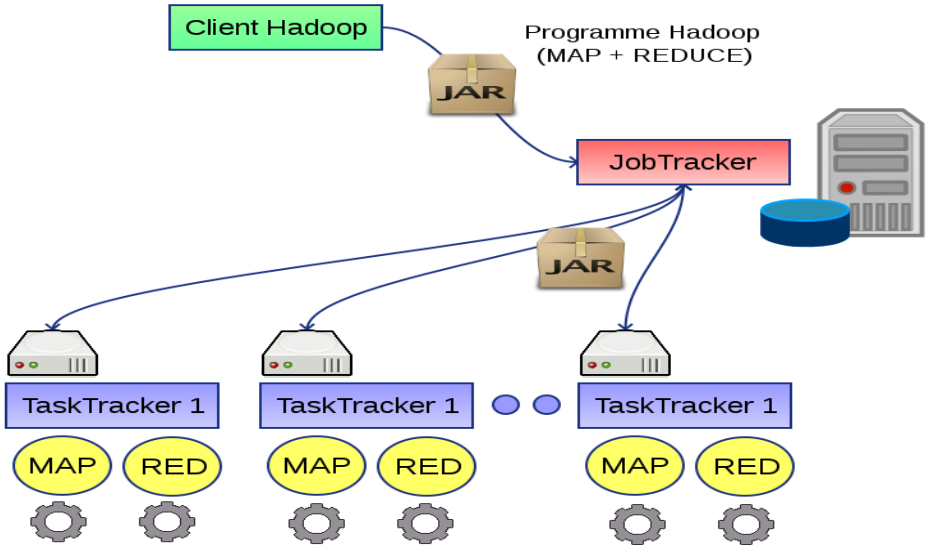

FIGURE 1.10 – Les composants de MapReduce.

- **Job tracker :** Est le nœud principal qui gère toutes les tâches et les ressources d'un cluster.
- **Les TaskTrackers :** Sont les agents déployés sur chaque machine d'un cluster pour lancer la map et réduire les tâches.
- **JobHistoryServer** est un composant permettant de suivre les tâches complétées, généralement déployé comme une fonction séparée ou avec JobTracker.

Exemple : *Dans ce qui suit on vient mettre en clair le fonctionnement de Map-Reduce avec l'exemple typique* ***"Word Count"*** *schématisé afin de mieux cerner le rôle de chaque fonction de ce framework.*

S'il est possible de compter manuellement le nombre de fois qu'un mot apparaît dans un roman, cela prend beaucoup de temps. Si l'on répartit cette tâche entre une vingtaine de personnes, les choses peuvent aller beaucoup plus vite. Chaque personne prend une page du roman et écrit le nombre de fois que le mot apparaît sur la page. Il s'agit de la partie Map de MapReduce. Si une personne s'en va, une autre prend sa place. Cet exemple illustre la tolérance aux erreurs de MapReduce. Lorsque toutes les pages sont traitées, les utilisateurs répartissent tous les mots dans 26 boîtes en fonction de la première lettre de chaque mot. Chaque utilisateur prend une boîte, et classe les mots par ordre alphabétique. Le nombre de pages avec le même mot est un exemple de la partie Reduce de MapReduce.[11]

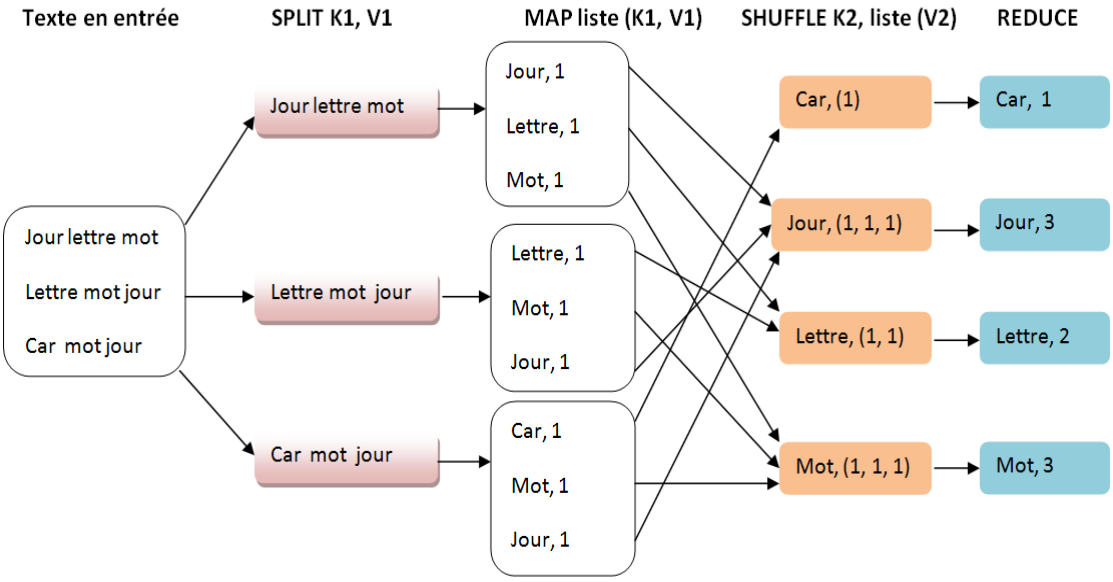

FIGURE 1.11 – Exemple de Word Count.

(b) **Les avantages de MapReduce :**
Parmi les avantages de la programmation MapReduce nous citons :

- La scalabilité.
- La flexibilité.
- La sécurité et l'authentification.
- Le traitement parallèle.
- La disponibilité.
- Un modèle simple de programmation.

3. **Bases NoSQL**
Les bases de données relationnelles ont une philosophie d'organisation des données bien spécifiques, avec notamment le langage d'interrogation SQL, le principe d'intégrité des transactions (ACID) [19], et les lois de normalisation [20].

Avec le succès que l'on connait sur les bases de données relationnelles. Ces dernières années, plusieurs avis nuancés ont été émis concernant le modèle relationnel qui n'est pas du tout adaptées au stockage de très grandes dimension et au traitement ultra rapide. Cela fut une motivation pour la nouvelle vague de bases NoSQL qui vient completer les limitations des systèmes de gestion de base de données classiques, en autorisant la redondance pour mieux servier les besoins en matière de flexibilité, de tolérance aux pannes et d'évolutivité.[B1]

Concrétement une base de données NoSQL est une approche de la conception des bases et de leur administration particulièrement utile pour de très grands ensembles

19. les propriétés ACID (atomicité, cohérence, isolation et durabilité) sont un ensemble de propriétés qui garantissent qu'une transaction informatique est exécutée de façon fiable.
20. La normalisation consiste ; Llrs de la création d'une base de données, à l'organiser en tables de telle sorte que les résultats de son utilisation soient toujours sans ambiguïté et sans erreur. La normalisation entraîne parfois la duplication de données de la base de données et crée souvent des tables supplémentaires.

de données distribuées. Elle englobe une gamme étendue de technologies et d'architectures, afin de résoudre les problèmes de performances en matière d'évolutivité et de Big Data que les bases de données relationnelles ne sont pas conçues pour affronter. De plus elle est particulièrement utile lorsqu'une entreprise doit accéder, à des fins d'analyse, à de grandes quantités de données non structurées ou de données stockées à distance sur plusieurs serveurs virtuels du Cloud.

Pour mieux comprendre les bases de données NoSQL et leur application dans le Big Data, nous allons consacrer par la suite un chapitre où nous détaillons cette technologie.

1.3 Les modes de stockage du Big Data

À l'ère du Big Data, la question se pose souvent de savoir quelles sont les architectures de stockage les mieux adaptées pour soutenir des processus analytiques à grande échelle. En fait, plusieurs technologies rivalisent aujourd'hui dans ce domaine, certaines parfois mieux adaptées à certains types de traitements que d'autres. Chacune a ses propres forces et faiblesses. Et en général, l'usage de l'une n'exclut pas celui d'une autre. En l'état actuel des développements, quatre technologies sont en lice pour les applications Big Data :

- Le stockage massivement distribué.
- Le NAS en mode Scale-out -En français la scalabilité horizentale- (ou NAS [21] en cluster).
- Les baies de stockage 100% Flash.
- Le stockage en mode objet.

1.3.1 Stockage en mode distribué

Les architectures de stockage distribuées sont souvent associées au "grid computing", car elles évoluent en parallèle de l'environnement de calcul. Dans ces architectures, les capacités de traitement sont co-résidentes des capacités de stockage sur les nœuds de la grille. C'est par exemple le cas pour les clusters HDFS (le file system d'Hadoop) ou pour certains clusters à base de la technologie Red Hat Storage[12].

Le principal avantage de ces architectures est leur faible coût, mais aussi la proximité du stockage des capacités de calcul. Cette affinité peut permettre d'accélérer considérablement les traitements si l'on parvient à localiser, les traitements pertinents à certaines données sur les nœuds où sont stockées ces données (ce qui est l'objectif de mapreduce dans Hadoop).

Dans certains cas, les déploiements se font en utilisant les emplacements de disques internes aux serveurs. Dans d'autres cas, on s'appuie sur des JBOD [22] connectés aux différents nœuds.

21. NAS : Network Attached Storage, ou rattachée au réseau

22. JBOD signifie « Just a Bunch Of Disks » (littéralement : juste un paquet de disques). Il s'agit d'une mise à disposition de disques durs, peu importe leur taille, sans aucune gestion du matériel, par un châssis de disques SAS ou SATA via un port externe. En cas de panne d'un des disques, les données situées sur ce dernier sont perdues, mais celles des autres disques restent accessibles.

1.3.2 Stockage en mode Scale-out NAS

est d'une certaine façon une variante du stockage distribué, à ceci prêt qu'à quelques rares exceptions, aucun traitement de calcul n'est effectué sur les nœuds de stockage distribués. La capacité CPU est intégralement dédiée à la gestion des fonctions de stockage.[12]

Les NAS en mode Scale-out ont des caractéristiques de performance uniques puisque l'ajout d'un nœud additionnel ajoute à la fois de la capacité, mais aussi de la connectivité additionnelle – donc de la bande passante -, ce qui les rend très attractifs pour le stockage de quantités massives de données avec des contraintes de performances fortes.

Le Scale-Out NAS est ainsi massivement utilisé dans le monde du HPC [23] et du calcul scientifique. Un autre avantage par rapport aux systèmes de stockage distribués est que les NAS en cluster disposent en général d'un environnement logiciel riche avec des fonctions avancées de gestion du stockage (snapshots, réplication, compression, déduplication de données) qui font encore défaut aux technologies de stockage distribué open source. Autant de fonctions importantes lorsque l'on manipule de grandes quantités de données avec des contraintes de performances, de capacité et de disponibilité fortes.

1.3.3 Stockage en mode Flash

Le stockage Flash est un support de stockage bien adaptées pour les applications Big Data, conçu pour sécuriser électroniquement les données. Le support est conçu pour être effacé et reprogrammé électroniquement. Le Flash représente une véritable mutation dans le secteur informatique ; en supprimant les délais de rotation et de recherche, cette technologie offre des vitesses d'exécution considérablement supérieures à celles des disques durs rotatifs traditionnels. Le stockage Flash constitue une réelle amélioration des performances pour les opérations de stockage (E/S).[13]

Une baie 100 % Flash, également appelée baie à l'état statique, est un stockage de données qui contient plusieurs lecteurs de mémoire Flash. Les pièces du stockage 100 % Flash ne bougent pas, ce qui signifie que beaucoup moins de chaleur est générée. De plus, la baie nécessite moins d'alimentation et moins de maintenance. Les baies 100 % Flash fournissent les bases nécessaires aux applications métiers de nouvelle génération et au datacenter 100 % Flash.

En effet, dans certains cas bien précis, où il est important d'obtenir des résultats d'analyse en quasi temps-réel, les baies 100% Flash sont incontournables. Elles sont par exemple un complément de choix pour les architectures In-Memory.

1.3.4 Stockage en mode objet

Avec l'essor du Cloud Computing, le stockage de données évolue. Le stockage objet, ou stockage basé objet, est une méthode de stockage non hiérarchique généralement utilisée pour le stockage Cloud. Elle consiste à stocker et gérer de très grands volumes de données non structurées sous forme d'unités appelées objets.

23. HPC est un sigle, qui signifie en anglais : High Performance Computer, ordinateurs dont le champ d'application est le high performance computing, soit ≪ calcul à haute performance ≫.

Les objets sont stockés au sein d'un seul bassin de stockage. Chaque objet est doté d'un identifiant qui permet de le retrouver facilement, et est associé à des métadonnées.[14]

1. **Les caractéristiques principales**
   Contrairement aux autres méthodes de stockage de données, le stockage objet ne repose pas sur une hiérarchie de répertoires. Les données sont stockées sous forme d'objets dans un espace d'adressage linéaire permettant de rendre le système évolutif et scalable.

2. **Constitution d'un objet :**

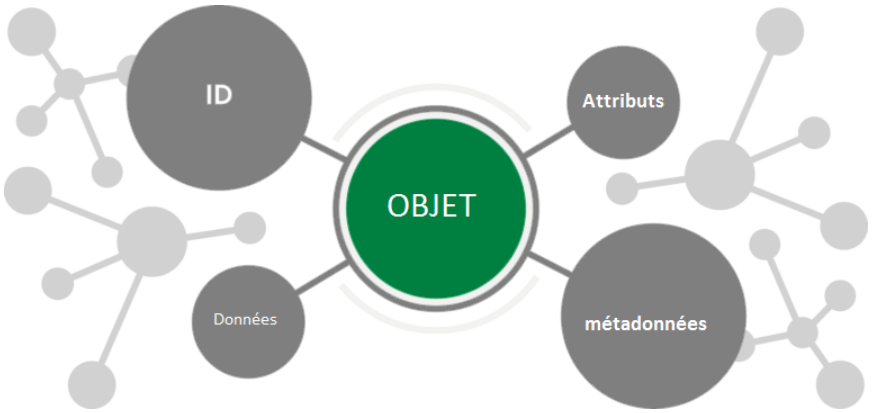

FIGURE 1.12 – Les caractéristiques d'un objet

- **La donnée**
- **Un identifiant unique** : Cet identifiant correspond au chemin d'accès de la donnée. L'utilisateur dispose donc d'une base de référence unique quel que soit l'endroit où se trouve son objet.
- **Des métadonnées** : Il s'agit des informations liées au contexte de l'objet (type de donnée, structure...). Elles permettent de faire le lien entre plusieurs objets qui comportent ces mêmes métadonnées.

1.4 Les sources du Big Data

Le Big Data est utilisé par les organisations uniquement à des fins d'analyse. Toutefois, avant de pouvoir extraire des informations précieuses à partir de données volumineuses, les entreprises doivent avoir la connaissance de plusieurs sources de données disponibles. Comme nous le savons, les données sont volumineuses et existent sous différentes formes. Si elles ne sont pas bien classées ou mal achetées, elles peuvent nous faire perdre un temps et des ressources précieuces. Ainsi, il est important que les entreprises disposent du savoir-faire nécessaire pour faire le tri entre les différentes sources de données disponibles et classer en conséquence leur utilisation et leur pertinence.[15]

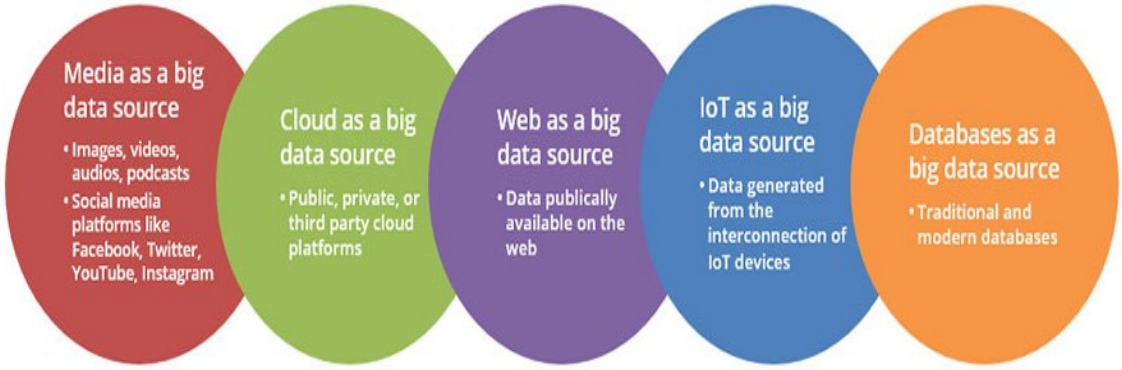

FIGURE 1.13 – Les 5 sources du Big Data.

1.4.1 Les médias

Les médias constituent la source la plus populaire de données volumineuses, car ils fournissent des informations précieuses sur les préférences des consommateurs et les tendances changeantes. Autodiffusés et franchissant toutes les barrières physiques et démographiques, ils constituent le moyen le plus rapide pour les entreprises d'obtenir un aperçu détaillé de leur public cible, d'en dégager des schémas et des conclusions et d'améliorer leur prise de décision.

> ***Exemple :*** *Les médias comprennent les médias sociaux et les plates-formes interactives, telles que Google, Facebook, Twitter, YouTube, Instagram, ainsi que des supports génériques tels que des images, des vidéos, des audios et des podcasts qui fournissent des informations quantitatives et qualitatives sur tous les aspects de l'interaction utilisateur.*

1.4.2 Le Cloud

Aujourd'hui, les entreprises ont pris de l'avance sur les sources de données traditionnelles en transférant leurs données sur le cloud . Le stockage en nuage prend en charge les données structurées et non structurées et fournit aux entreprises des informations en temps réel et des informations à la demande. L'attribut principal de l'informatique en nuage est sa flexibilité et son évolutivité. Les mégadonnées pouvant être stockées et collectées sur des clouds publics ou privés, via des réseaux et des serveurs. Ainsi le cloud constitue une source de données efficace et économique.

1.4.3 Le Web

Le Web public constitue un Big Data répandu et facilement accessible. Les données sur le Web ou sur Internet sont généralement disponibles pour les particuliers et les entreprises. En outre, des services Web tels que Wikipedia fournissent à tous des informations détaillées et gratuites. L'énormité du Web garantit sa convivialité et est particulièrement bénéfique pour les nouvelles entreprises et les PME [24], car elles n'ont pas à attendre pour développer leur propre infrastructure de Big Data et leurs référentiels avant de pouvoir exploiter le Big Data.

24. PME : Petite et moyenne entreprise

1.4.4 Les bases de données

Les entreprises d'aujourd'hui préfèrent utiliser une fusion de bases de données traditionnelles et modernes pour acquérir des données volumineuses pertinentes. Cette intégration ouvre la voie à un modèle de données hybride et nécessite un faible investissement et des coûts d'infrastructure informatique. En outre, ces bases de données sont également déployées à plusieurs fins d'informatique décisionnelle. Ces bases de données peuvent ensuite permettre d'extraire des informations permettant de générer des bénéfices pour l'entreprise.

> ***Exemple :*** *Les bases de données populaires incluent diverses sources de données, telles que MS Access, DB2, Oracle, SQL et Amazon Simple, entre autres.*

Le processus d'extraction et d'analyse de données parmi de nombreuses sources de données volumineuses est un processus complexe qui peut être frustrant et prendre beaucoup de temps. Ces complications peuvent être résolues si les organisations englobent toutes les considérations nécessaires du Big Data, prennent en compte les sources de données pertinentes et les déploient de manière à répondre parfaitement à leurs objectifs organisationnels.

1.4.5 L'Internet Des Objets (IoT)

Les données créées ou générées à partir de l'IoT constituent une source précieuse de données volumineuses. Ces données sont généralement générées à partir des capteurs connectés à des appareils électroniques. La capacité d'approvisionnement dépend de la capacité des capteurs à fournir des informations précises en temps réel. L'IoT prend de l'ampleur et inclut les données volumineuses générées, non seulement à partir d'ordinateurs et de smartphones, mais également de tout appareil pouvant émettre des données. Grâce à l'IoT, les données peuvent désormais provenir

> ***Par exemple,*** *de dispositifs médicaux, de processus véhiculaires, de jeux vidéo, de compteurs, de caméras, d'appareils électroménagers, etc.*

Le chapitre suivant fera l'objet d'étude de cette source et sa relation avec le Big Data.

1.5 Les nouveaux métiers du Big Data

Nous pouvons, pour simplifier, répertorier 4 catégories de métiers non liés directement au Big Data, tout au moins rendus visibles et attractifs grâce à la prise de conscience des enjeux liés aux données du Big Data :

- **Le Chief Data Officer (CDO) :** Autrement appelé directeur de la transformation digitale .
- **Le Data Stewart :** Il s'agit de l'administrateur des données.
- **Le Data Scientist :** On parle ici de celui qui analyse la donnée à l'aide d'outils statistiques et datamining complexes.

- **Le Data Analyst :** A pour rôle d'analyser les données pour ses besoins métiers propres.

- **Le Chief Digital Officer**

  Le rôle du Chief Digital Officer, autrement appelé Directeur de la transformation digitale, est à la fois pilote et moteur, à haut niveau de responsabilité. Il cumule de nombreux savoir faire (marketing, stratégie, IT, business, innovation, logique d'entreprise, process) et savoir être (communication, ouverture aux autres, capacités à convaincre...) pour prendre en charge cette fameuse mutation numérique (il sait répondre à la question : « mais par quel bout on commence ? »).

  Il contribuer à la stratégie de l'entreprise en s'appuyant sur les données, leur gestion, éventuellement en maintenant le niveau de qualité de celles-ci ; Puis diffuser la connaissance en interne des données (dans les grandes entreprises il y a des gisements de données dans toutes les entités, mais personne n'a de vision transversale).[16]

- **Le Data Stewart**

  Ils ont une connaissance de la donnée, et la travaillent quotidiennement, même si ce n'est pas nécessairement un travail à temps plein. C'est le poste le plus bas dans l'organisation de ces nouveaux métiers. Il doit dépendre du CDO, car il fait partie d'une communauté, il n'est pas seul. Dans les grandes entreprises, il y aura un Data Stewart par territoire. Si c'est une organisation matricielle, on y rajoutera une dimension métier. Les Data Stewart sont responsables de la mise en œuvre de la stratégie sur le terrain, ils vont appliquer la gouvernance décidée par le CDO, et veiller à ce qu'elle soit suivie ; de même pour ce qui est des bonnes pratiques et des cycles de vie.[B4]

- **Le Data Scientist**

  Le Data Scientist est celui qui produit la valeur de la donnée. Il part de données fiables (grâce au travail du Data Stewart), et il a les outils pour le faire. Les Big Data sont récentes, mais les problèmes liés aux Big Data existent depuis longtemps. C'est la technologie récente qui permet de traiter les plus gros volumes et aussi de faire ce travail en temps réel. Le Data Scientist est un expert aux multiples compétences. Il maîtrise les outils statistiques et le datamining pour pouvoir manipuler les données à sa guise ; enfin, il est capable de comprendre les finesses d'un processus pour pouvoir se poser les bonnes questions tout en suggérant des pistes de réponses.[B4]

- **Le Data Analyst**

  Le Data Analyst est également quelqu'un qui produit la valeur de la donnée. Il réceptionne une partie du travail du Data Scientist et le rapproche des autres reportings et des autres données qu'il a en sa possession pour pouvoir faire son travail. Il utilise des outils de dashboarding, de visualisation de l'information et d'exploration de l'information assez proches de ceux qu'il utilise pour de la Business Intelligence

mais il les applique différemment en fonction des données qu'on met à sa disposition et des enjeux métiers auxquels il doit faire face.[B4]

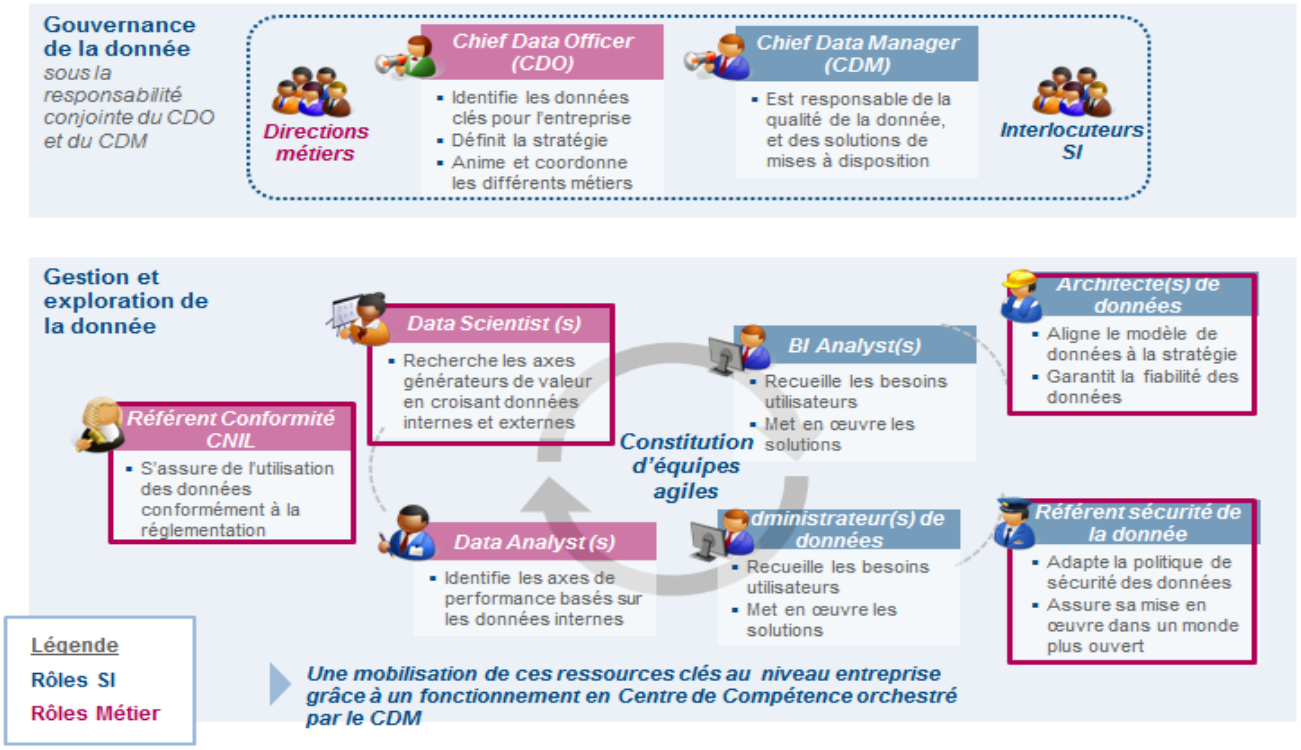

FIGURE 1.14 – La hierarchie entre les métiers du Big Data.

Conclusion

L'arrivée du Big Data est maintenant présentée par de nombreux articles comme une nouvelle révolution industrielle semblable à la découverte de la vapeur (début du 19e siècle), de l'électricité (fin du 19e siècle) et de l'informatique (fin du 20e siècle). D'autres, un peu plus mesurés, qualifient ce phénomène comme étant la dernière étape de la troisième révolution industrielle, laquelle est en fait celle de l'information. Dans tous les cas, le Big Data est considéré comme une source de bouleversement profond de la société.

En effet, Parmis les sources des données traitées par le Big Data on retrouve les objets connéctés. A mesure que le nombre d'objets connecté augmente, le volume de données générées par l'internet des objets prend de plus en plus d'ampleur. Ainsi, pour pouvoir les prendre en charge et les analyser en temps réel, il est nécessaire de s'en remettre aux outils analytiques Big Data.

C'est la raison pour laquelle le Big Data et l'internet des objets (IoT) sont étroitement liés. Toutefois, avant de s'étendre plus en détail sur cette relation, il convient de revenir sur la présentation de cette technologie qui fera l'objet du chapitre suivant.

Chapitre 2

L'Internet des Objets (IoT)

Introduction

Depuis plusieurs années, Internet ne se limite plus aux ordinateurs et autres smartphones. Désormais, presque tous les objets sont connectables à Internet. Les montres, les frigos, les télévisions, les voitures, ou encore les machines industrielles...

L'Internet des objets n'a pas de limites, elle constitue la prochaine étape de la révolution numérique. En effet, selon une récente étude d'IDC [1] cabinet de recherche international dans le domaine des technologies, d'ici 2025 le volume de données stockées atteindra 175 Zetta-octets (Zo) en 2025, soit 5,3 fois plus que ce qui est stocké actuellement.

2.1 Concepts généraux

Dans cette section nous allons d'abord donner une définition au terme d'Internet des objets (IdO) puis évoquer ses composants principaux.

2.1.1 Définitions liées à l'Internet des objet

Définition 1. l'Internet des objets(IdO) ou Internet Of Things(IoT) en anglais désigne selon l'ingénieur Kevin ASHTON [2] un système où les objets physiques sont connectés à Internet et qui sont capables de créer et transmettre des données afin de créer de la valeur pour ses utilisateurs à travers divers services.[1]

Définition 2. l'Internet des objets est aussi selon l'UIT [3] une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physiques ou virtuels) grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution.[B1]

1. International Data Corporation (IDC) est le premier groupe mondial de conseil et d'études sur les marchés des technologies de l'information.

2. Kevin ASHTON est un expert en capteurs de consommation novateur qui a inventé l'expression ≪Internet des objets≫ pour décrire le réseau connectant des objets du monde physique à Internet

3. Union internationale des télécommunications est l'agence des Nations unies pour le développement spécialisé dans les technologies de l'information et de la communication

Définition 3. l'Internet des Objets est un réseau de réseaux qui permet, via des systèmes d'identification électronique normalisés et unifiés, et des dispositifs mobiles sans fil, d'identifier directement et sans ambiguïté des entités numériques et des objets physiques et ainsi de pouvoir récupérer, stocker, transférer et traiter, sans discontinuité entre les mondes physiques et virtuels, les données s'y rattachant.[B2]

2.1.2 Les composants de l'Internet des objets

L'internet des objet se repose principalement sur quatre composants essentiel pour assurer la récollet, la transmission, le traitement et l'exploitation des données. [2]

- Les objets.
- Les données.
- La connectivité.
- Le traitement et exploitation des données.

1. **Les objets**
   Un objet est, avant toute chose, une entité physique pouvant être un livre, une voiture, une machine à café électrique ou un téléphone mobile[B3] conçue sans intégration des systèmes informatiques ni connexion à Internet. Par conséquent, l'intégration d'une connexion Internet à cet objet permet de l'enrichir en terme de fonctionnalité, d'interaction avec son environnement, il devient alors un objet connecté riche.[B4]

   (a) **Les objets connectés (intelligents) :**
   Ce sont des objets peuvent eux être connectés en permanence à internet ou lorsqu'une connexion est disponible à fin d'accomplir des calculs, d'effectuer des mesures sur l'environnement ou d'influer sur celui-ci.

	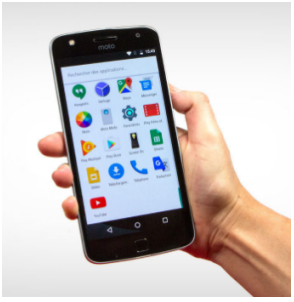
Appareils électroménagers on retrouve les fours, les réfrigérateurs qui on une capacité de se connecter à internet afin de faciliter la vie des utilisateurs.	**Smartphones** sont des appareils qui se connectent à Internet de partout et à n'importe quel moment. Ils accèdent à plusieurs fonctionnalités :GPS, météo, etc.
	
La caméra connectée est dotée d'une connexion à Internet qui permet à l'utilisateur via une application sur son Smartphone de vérifier en temps réel l'état de sa maison (vérifier qu'il n'a pas oublié d'éteindre une lumière).	**Bracelet santé connecté** est un dispositif de surveillance permettant de suivre à tout moment les activités physiques d'une personne en l'informant par exemple : Des calories brûlées, la qualité de son sommeil grâce au suivi cardiaque en continu..

TABLE 2.1 – Exemples d'objets intelligents

(b) **Les capteurs**

Un capteur est un dispositif de prélèvement d'informations qui élabore, à partir d'une grandeur physique,une autre grandeur physique de nature différente (généralement électrique) image de la grandeur prélevée, et utilisable à des fins de mesure. D'une manière plus simple, un capteur est un dispositif qui transforme une grandeur physique en une grandeur électrique (courant, tension ou résistance).

Leur rôle dans l'internet des objets est très important, ils permettent de capter des données en temps réel afin de les exploiter dans des domaines bien précis. Ces capteurs se composent de modules d'énergie, de gestion d'alimentation, de Radio fréquence qui gèrent les communications.

On peut classer ces capteurs on deux catégories selon sa mis en œuvre pour générer le signal de mesure[3].

- **Les capteurs passifs :**
  Un capteur passif est celui qui à besoin d'apport d'énergie extérieure pour fonctionner et fournir l'information.
  Exemple :*les sismomètres*[4] *utilisent des capteur passifs pour des mesures de position de la masse mobile.*

- **Les capteurs actifs :** Un capteur actif transforment directement la grandeur physique en grandeur électrique.
  Exemple : *un thermocouple transforme directement la température en tension électrique*

	
capteur de distance	capteur de gaz
	
capteur de température	capteur de pression

TABLE 2.2 – Exemples de capteurs

2. **Les données :**
   En générale, les données sont ce qui est connu et qui sert de point de départ à un raisonnement ayant pour objet la détermination d'une solution à un problème en relation avec cette donnée[4]. En effet, ces données sont partout, proviennent de différentes sources (web, objets connectes ...), et prennent divers formats. A l'état brute, les données n'ont aucune signification réelle, c'est qu'une fois traiter et analyser qu'elles deviennent utiles.

   On distingue deux types de données, les données structurées et non structurées.

   - **Les données structurées :**
     Les données structurées sont organisées de façon à être comprises par des machines. Elles sont généralement stockées dans des bases de données relationnelles et affichées en colonnes et lignes ce qui facilite leur exploitation.[5]
     Par d'exemple *on souhaite enregistre les champs : code capteur, quantité d'énergie et code route dans une base de données via une application, cependant, on crée des données structurées.*

4. sismomètre est un instrument de mesure équipé d'un capteur des mouvements du sol, le sismomètre, capable de les enregistrer sur un support visuel, le sismogramme.

- **Les données non structurées :**
  Les données non structurées sont des données représentées ou stockées sans format prédéfini, ce dernier entraîne des irrégularités et des ambiguïtés qui peuvent rendre difficile la compréhension des données.[6]

  A titre d'exemple *les données textuelles générées par les courriers électroniques, les données non textuelles telles que les photos, les fichiers audio et des vidéos.*

3. **La connectivité :**
   La connectivité est considéré comme étant un élément de base dans l'IoT, vu que c'est elle qui assure la communication entres les différents objets connéctés. Elle s'organise autour d'une multitude de réseaux qui permettent de couvrir le territoire à des échelles locales et nationales afin de transporter les données issues des objets.[B5]

   A titre d'exemple : *on a le wifi, le bluetooth, le Zigbee utilisés pour la communication à courte porté et le satellite, le SigFox pour la communication à longue porté.*

communication à courte porté	
Technologie	**Caractéristiques**
Wifi	• Transmission sans fil à moyenne portée. • Une fréquence qui varie entre 2.5 GHZ et 5.0 GHZ . • Distance de transmission qui va de 10 m jusqu'à 100 m
Bluetooth	• Transmission sans fil à courte portée. • Une bande de fréquence de 2,4 GHz. • distance maximale jusqu'à 15 m
Zigbee	• Transmission sans fil à courte portée. • Utilise 3 bande de fréquence (2.4 GHz disponible partout, 915 MHz disponible en Amériques et Australie et 868 MHz disponible Europe). • distance maximale jusqu'à 10m

TABLE 2.3 – Exemple de Technologie des réseaux à courte porté

communication à longue porté	
Technologie	**Caractéristiques**
satellite	• De 4 à 8 GHz(C band), de 10 à 18 GHz(Ku band) et de 18 à 31 GHz (Ka band). • Un débit de données de 16 kbps à 155 mbps . • Distance de transmission à l'échelle terrestre.
SigFox	• Fréquence de 902 à 928 MHz. • Un débit de données arrivant jusqu'à 100 bps .

Table 2.4 – Exemple de Technologie des réseaux a longue porté

Remarque :
La communication entre les différents objets peut se faire avec ou sans l'intervention humaine. Cependant, on trouve :

Machine-to-machine (M2M) *désignant l'échange de données entre machines sans intervention humaine en utilisant les différents moyens de communication cité ci-dessus.par exemple : dans la domotique, on trouve un capteur d'intrusion qui est en communication direct avec le capteur déclencheur d'alarme.*
Machine-to-personne (M2P) *Décrit les analyses pour les mégadonnées sous une forme lisible par l'homme, par exemple une personne voulant récupérer des informations d'une base de données il interagi avec la machine afin de lui fournir ces informations.*

4. **Traitement et exploitation des données**
Les données générer par l'IoT prennent diverses formes, certaines sont structurées non-structurées. Afin de mieux les exploiter, il est très importants de les analyser. Cette étape consiste à convertir les données captés de la forme analogiques en informations utiles pouvant être interprétées et utilisées pour une analyse détaillée afin de les stocker dans des bases de données.

Les données transportées à partir des support de transmissions, sont transmises dans des réseaux virtuel tel que le Cloud. Leur traitement et stockage sont assurés par les opérateurs du Cloud dans le but de produire des services.

2.2 Architecture de l'Internet des objets

L'architecture d'un système IoT est composée de plusieurs niveaux qui communiquent entre eux pour relier le monde tangible des objets au monde virtuel des réseaux et du cloud. Tous les projets n'adoptent pas une architecture formellement identique, néanmoins il est possible de schématiser le parcours de la donnée.[B6]

Les niveaux de communication entre les composants de l'IoT :

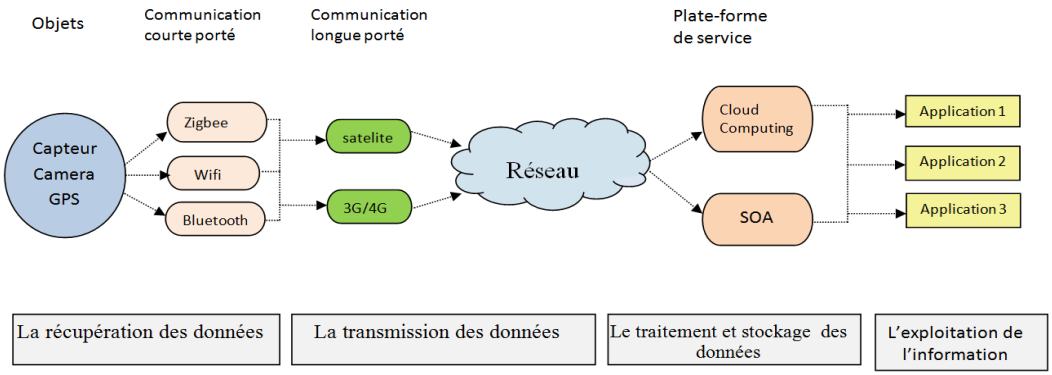

FIGURE 2.1 – L'architecture de l'IoT

1. **La récupération des données :**
   Le premier niveau du système IoT est la collecte de données provenant de différents objets tel que, les capteurs, les caméras, terminaux GPS, etc. Ces données (température, Niveau d'humidité, quantité d'énergie d'un séisme, etc) sont collectées en utilisant des technologies de communication à courte portée (Zigbee, Bluetooth, Wifi, etc) ou à longue portée (Satellite, 3G/4G, etc).

2. **La transmission des données :**
   Dans ce deuxième niveau, les données collectées par la détection doivent être transmises à la plateforme de service à travers le système réseau afin que les applications puissent accéder à ces données. Des méthodes sont nécessaires pour accéder au réseau via passerelles d'accès et technologies hétérogènes.

3. **Traitement et stockage des données :**
   A ce niveau, les données transmises via le réseau, seront analysées, traitées par les plates-forme de service (le Cloud Computing, SOA, etc) et stockées dans leur bases de données. Dans le but de faciliter aux utilisateurs d'application l'accès à ces données, ces plates-forme offrent des services hébergés sur Internet, ***par exemple :*** *les ressources virtualisées(machine virtuelle ou une application), qui peuvent être allouées de manière dynamique sans aucune intervention humaine.*

4. **Exploitation de l'information :**
   L'utilisateur bénéficie, dans la plupart des cas, d'une interface dédiée sur son terminal mobile, type smartphone. Il peut alors piloter son objet, accéder à l'historique des mesures relevées, et utiliser les différentes fonctionnalités prévues par la solution IoT. De part cette interface, l'entreprise peut communiquer avec l'utilisateur de façon directe et personnalisée.

2.3 La sécurité dans l'Internet des objets

Aujourd'hui l'internet des objets permet de faire communiquer le monde industriel et le monde grand public, et dans tous les secteurs il y a des exemples d'attaque ou des

problèmes de sécurité. Avec les objets connectés on observe un élargissement du périmètre d'attaque des cybercriminels.
Dans cette section, nous allons présenter quelques menaces liées à l'IoT ainsi que les recommandations techniques pour minimiser ces menaces.[B7]

2.3.1 Les menaces et les attaques dans l'IoT :

1. **Les attaques liées aux objets connectés eux-même :**
   Du côté matériel, on trouve par exemple, le piratage des composants présents sur les circuits imprimés qui peut servir à dessouder le processeur central. En parallèle, du côté logiciel, la compromission d'un objet connecté par un code malveillant peut entraîner son inaccessibilité pour son utilisateur, la désactivation de fonctions de sécurité voire la compromission d'autres objets connectés communiquant avec l'équipement compromis.

2. **Les attaques réalisées au travers des objets connectés :**
   Ces attaques exploitent les équipements connectés comme moyen d'accès à de plus vastes réseaux tels que Internet. L'équipement compromis sert ainsi de relais dans la réalisation d'une attaque de plus vaste ampleur, comme un déni de service distribué.

   Exemple :
   Une caméra connectée à Internet peut posséder une vulnérabilité permettant à un pirate d'en prendre le contrôle et d'exécuter du code malveillant, et tenter de refaire le même scénario à un maximum de caméras identiques accessibles sur Internet afin de les transformer en autant de machines participant à une attaque de plus vaste ampleur, sans que les propriétaires des caméras ne se rendent compte.[B7]

3. **Les attaques contre les réseaux et protocoles de communication de l'IoT :**
   Les protocoles de communication de l'IoT autorisent la communication entre les différents objets soit à courte ou longue distance, de manière sécurisée ou non. Si un pirate réussit à s'introduire dans ces réseaux, il pourra être en mesure d'intercepter les données échangées entre divers équipements mais aussi d'attaquer d'autres équipements auxquels il n'avait pas accès initialement, grâce à cette interconnexion.

2.3.2 Les recommandations techniques :

1. **La sécurité physique de l'objet :**
   En mesure de protection des données collectées (en particulier les plus sensibles) et stockées par l'objet, il est nécessaire de verrouiller le boîtier de l'objet par (collage, soudure, ...) de façon à empêcher son ouverture normale, de faire le moulage des cartes et composants électroniques dans de la résine pour empêcher l'identification des composants utilisés, d'utiliser des éléments sécurisés [5] pour le stockage des clés et les traitements cryptographiques.

2. **La sécurité des protocoles de communication :**
   Pour mener à bien la transmission des données, il ne suffit pas uniquement de choisir un protocole adapté, mais il est aussi nécessaire d'activer ses fonctions de sécurité et les utiliser convenablement, de chiffrer les communications sensibles, utiliser l'authentification par certificat électronique a fin d'éviter les attaques de type "Man-in-the-Middle" [6].

3. **La sécurité applicative et système :**
   Dans le but de protéger les système intégrés dans des appareils(serveur, smartphone,...) et les logiciels ou applications (site,...), il est recommander d'implémenter des contrôles côté serveur, d'utiliser les sondes HIDS [7], de supprimer les comptes et services non utilisés, d'appliquer des mots de passe forts sur tous les comptes

2.4 Domaines d'application de l'Internet des objets

L'évolution de l'IoT s'accélère de plus en plus. Désormais, il s'imposer dans divers domaines dans le but de répondre a leur besoins et d'améliorer leur mode de travail. Parmi ces domaines, on a la santé, la domotique, l'industrie, l'agriculture et les smart cities.

Exemples :

1. **La santé :**
   les objets connectés impactent très fortement le domaine de la santé en permettant, notamment, un meilleur suivi des maladies chroniques (diabète,asthme, hypertension artérielle, etc.) et une meilleure observance des traitements,l'hospitalisation à distance, notamment pour les soins de suite et de réadaptation.[B8]

2. **L'industrie :**
   L'intégration de l'IOT dans le domaine de l'industrie a permis de renforcer la prévention des incidents et d'améliorer les conditions de travail. Par exemple, les capteurs connectés récoltent et enregistrent les données relatives à la défaillance d'une pièce de manière plus précise ce qui permet de réduire les non-conformités et améliorer la qualité.

5. Élément sécurisé est une plate-forme matérielle inviolable, capable d'accueillir en toute sécurité des applications et stocker des données confidentielles chiffrées :

6. Man-in-the-Middle appelée attaque de l'intercepteur : est une attaque qui a pour but d'intercepter les communications entre deux parties, sans que ni l'une ni l'autre ne puisse se douter que le canal de communication entre elles a été compromis

7. Host-based Intrusion Detection System : est un système de détection d'intrusion capable de surveiller et d'analyser les composants internes d'un système informatique ainsi que les paquets réseau sur ses interfaces réseau,

3. **La domotique :**
   La domotique ou "le maison intelligente" est parmi les domaines les plus affectés par l'émergence de l'IoT. Grâce aux objets domestiques mis sur réseau, le propriétaire d'une maison peut facilement les contrôler à distance en ce servant d'une interface sur son smartphone ou d'une application sur le web.

4. **L'agriculture :**
   Pour faire face à l'explosion démographique, les agriculteurs et les éleveurs doivent accroitre leur production. L'intégration d'objets connectés dans le domaine de l'agriculture, leur a permit d'évoluer cette production. On peut citer : les drones qui permettent de prévenir les agricoles des recoles atteinte maladie pour éliminer, Les colliers trackers pour la surveillance du bétail.[7]

5. **Les smart cities :**
   A l'heure où l'IoT explose, l'urbanisme fait, lui aussi, sa transformation numérique en mettant les objets connectés au service d'une ville plus intelligente. Cette digitalisation de l'espace urbain a un véritable impact sur l'aménagement des territoires. En effet, grâce aux données collectées par les objets connecté, les villes de demain permettent aux habitants de mener une vie plus simple, en prenant exemple de l'éclairage public intelligent qui ne s'allumera qu'en cas de détection de passants et s'adaptera en fonction de la luminosité, et permettra de réduire la consommation énergétique publique, ou bien l'optimisation des trajets des camions et donc la réduction des détours polluants ou encore l'amélioration de la fluidité du trafic urbain.[8]

2.5 Le rôle de l'Internet des objets dans le Big Data

L'horizon de l'Internet des objets et du Big data représente un important relais de croissance économique selon de nombreuses études [8]. Ces deux technologies ouvrent la possibilité de connecter les personnes ou les objets de manière plus pertinente dans un écosystème numérique désormais global, de fournir la bonne information au bon destinataire et au bon moment, ou encore de faire ressortir les informations utiles à la prise de décision.[B8]

Aujourd'hui, les objets connectés couplés à la capacité d'analyse du Big data fournissent aux entreprises, aux particuliers et aux administrations de nouveaux leviers d'action à savoir :

1. **Disposer de plus d'informations pour la prise de décision :**

   Exemple : *les départements marketing disposent d'informations plus détaillées sur les usages qui sont faits de leurs produits et les préférences individuelles de leurs clients, et peuvent mieux répondre aux attentes de ces derniers.*

2. **Mieux anticiper afin d'agir plus tôt et à moindre coût.**

   Exemple : *Dans le domaine de la santé le suivi régulier et précis de critères diagnostiques permet la détection précoce de certaines pathologies et en minimise pour l'avenir les conséquences.*

8. Cisco, McKinsey, Idate, Inspection générale des finances, Gartner Research, Boston Consulting Group, A.T. Kearney, etc.

2.6 Mode de stockage des données de l'internet des objets

L'Internet des Objets (IoT) génère de la valeur ; il produit aussi des données dont le gigantesque volume entraîne une remise en question des modèles de stockage. L'IoT accroît la pression sur l'infrastructure qui doit répondre aux deux conditions que posent le volume des données et son traitement en temps réel. Dans cette section nous allons aborder les technologies utilisées dans le contexte de l'IoT à savoir le Cloud computing, le Edge computing, le Fog computing et le Roof computing.

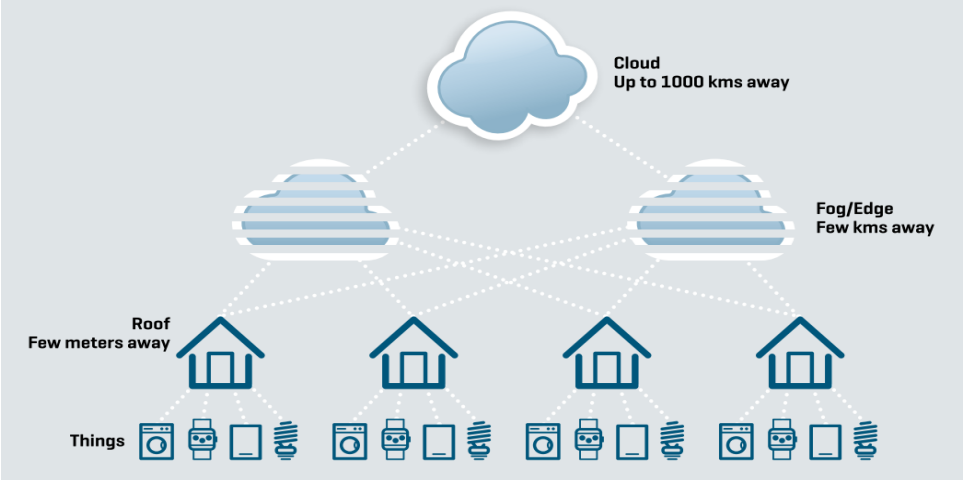

FIGURE 2.2 – Exemple de mode de stockage de l'IoT

2.6.1 Le Cloud Computing

Un paradigme de calcul distribué émergeant dans lequel les données et les services sont disponibles dans des data centers extensibles et peuvent être accédés de manière transparente depuis des appareils(ordinateurs, téléphones, grappes,. . .)connectés par Internet".

2.6.2 Le Edg Computing

Le Edge computing une méthode permettant d'optimiser les systèmes de cloud computing en effectuant un traitement de données à l'extrémité du réseau, près de la source des données, en intégrant les capacités de réseau, de calcul, de stockage et d'application, et en fournissant des services intelligents de périphérie. Il réduit principalement la bande passante de communication nécessaire entre les capteurs et le centre de données central en effectuant des analyses et en générant des connaissances au niveau de la source de données ou à proximité. Cette approche exploite des ressources dont la connexion à un réseau, tel qu'un smartphone, un ordinateur portable, une tablette ou un capteur, n'est pas toujours nécessaire.[B6]

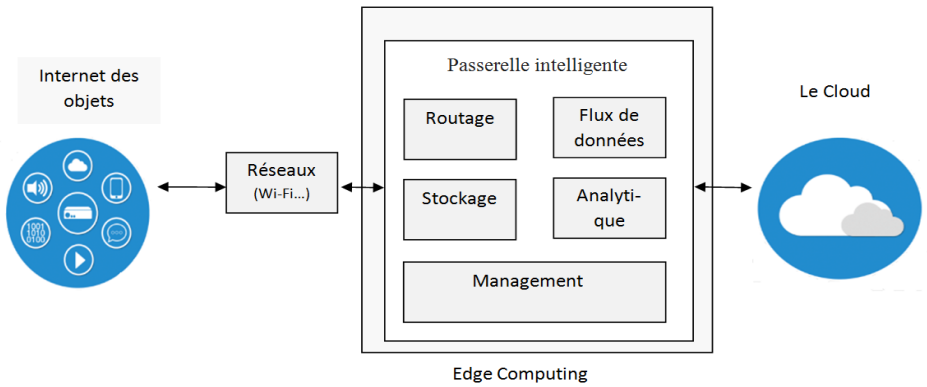

FIGURE 2.3 – l'IoT et le Fog Computing

2.6.3 Le Fog computing

Le Fog computing est considéré comme une extension du paradigme de Cloud computing du cœur du réseau à la périphérie du réseau. Il accélère la prise de conscience et la réaction aux événements en éliminant les allers-retours dans le le Cloud pour analyse. Cela évite des ajouts coûteux en bande passante en déchargeant des gigaoctets de trafic réseau du réseau principal. Il protège également les données IoT sensibles en analysant les données là où elles se trouvent (c'est-à-dire sans les déplacer en dehors de l'entreprise). Il s'agit d'une plate-forme hautement virtualisée fournissant des services de calcul, de stockage et de mise en réseau entre les terminaux et les serveurs cloud traditionnels.

Fog computing constitue la plate-forme appropriée pour un certain nombre de services et d'applications IoT critiques, à savoir les véhicules connectés, les réseaux intelligents, les villes intelligentes et, en général, les réseaux de capteurs et d'actionneurs sans fil. OpenFog définit le brouillard informatique comme une architecture horizontale au niveau du système qui distribue des ressources et des services d'informatique, de stockage, de contrôle et de mise en réseau partout dans le continuum allant du Cloud aux objets.[B6]

2.6.4 Le Roof computing

Le Roof computing est un réseau récemment fédéré et un paradigme informatique pour l'IoT disponible pour la facilitation des opérations sur site en temps réel "ROOF [9]". IL peut être utilisé dans les routeurs domestiques, les passerelles IoT, les téléphones mobiles, les plates-formes informatiques personnelles et autres plates-formes intégrées pouvant servir de proxy pour la connectivité au cloud et au réseau.

Dans cette architecture, le Roof computing est un élément de réseau fixe toujours accessible en tant que premier saut du canal de communication. Il permet une architecture réseau sécurisée forte et améliore la connectivité robuste vers le cloud en fournissant aux objets les capacités de stockage et de traitement requises. Il établit un pont entre le monde physique et le monde virtuel en connectant les objets et les systèmes au Cloud computing et en construisant un système virtuel pour les systèmes physiques respectifs, créant ainsi des systèmes cyber physiques [10] extrêmement évolutifs pour l'IdO.[B6]

9. ROOF : Real-time Onsite Operations Facilitation

10. systèmes cyber physiques CPS : sont des systèmes formés d'entités collaboratives,dotées de capacité de calcul, qui sont en connexion intensive avec le monde physique environnant et les phénomènes s'y déroulant, fournissant et utilisant à la fois les services de mise à disposition et de traitement de données disponibles

Conclusion

L'internet des objets n'existerait pas sans bien sûr, les inventions technologiques. En effet, plus les nouvelles technologies augmente plus l'IoT évolue et prend de la place dans nos vies. Cette liaison nous amène à un monde de plus en plus connecté avec la réalité, la fusion du réel au virtuel nous facilite notre quotidien en nous offrons des solutions toujours plus innovantes.

La valorisation des données issues de l'IoT est totalement compatible avec la démarche et les outils Big Data. Pour autant, l'industrialisation de l'IoT (déploiement et gestion des flottes, contractualisation avec les fournisseurs de réseaux LPWAN pour assurer la connectivité, gestion des mises à jour logicielles à distance, etc.) demeure un sujet à part entière et capital sur lequel il est nécessaire d'investir de manière décorrélée d'une approche Big Data pour faire atterrir les cas d'usage.

On constate cependant en entreprise des filières distinctes « IoT » et « Big Data » pour traiter chacun de ces projets : il devient primordial que ces dernières se réorganisent afin de concilier les deux approches. La chaine de traitement de la donnée IoT doit être considérée dans son ensemble, de l'objet « communiquant » jusqu'au service mis à disposition et consommé par le client (et donc en principe, via les services Big Data). Ainsi, le sujet n'est pas seulement de surmonter les complexités technologiques pour unifier les solutions de traitement mais bien de fusionner les initiatives et les compétences au sein des programmes et des organisations.

sur le réseau

Chapitre 3

Les bases de données NoSQL

Introduction

À la naissance de l'informatique, plusieurs modèles de stockage de l'information sont explorés. Mais c'est finalement le modèle relationnel qui l'emporte dans les années 1970 car il permet de mieux assurer le contrôle de l'intégrité des données, grâce à un modèle théorique puissant et simple.

Au cours de la dernière décennie de nombreuses grandes sociétés, notamment les géants du WEB s'interrogent et recherchent des techniques leur permettant de monter en charge [1] ou de trouver des solutions à leurs problèmes de traitement de données massives que les systèmes de gestion de base de données relationnels ne peuvent gérer, En effet le langage SQL a atteint ses limites dans certaines situations que nous verrons plus en détail dans ce chapitre, dans ce contexte le mot "NoSQL" est apparu.

Tout au long de ce chapitre nous allons présenter d'une part les SGBDR [2], leurs contraintes, leurs points forts ainsi que leurs limites qui ont été le point de départ du mouvement NoSQL que nous allons aussi détailler. D'autre part la comparaison entre ces deux notions s'impose afin de mieux comprendre l'apport du Nosql par rapport au SQL. Enfin, nous achevons ce chapitre en citant quelques exemples des bases de données NoSQL les plus utilisées avec leur caratéristiques et avantages.

3.1 Brève histoire des systèmes de gestion de bases de données

A partir du moment où les utilisateurs ont disposé d'ordinateurs, on s'est immédiatement posé la question de savoir comment les données pouvaient être archivées et extraites des organes périphériques des ordinateurs. La façon dont nous nous représentons ces ordinateurs est une métaphore du cerveau humain. Il nous est évident que l'élément central du fonctionnement intellectuel est la mémoire.

Au début, les données ont été stockées sur des bandes magnétiques où elles étaient regroupées en enregistrements physiques. Dans la mesure où ces derniers représentaient

1. La montée en charge ou en anglais Scalability désigne la capacité d'un produit à s'adapter à un changement d'ordre de grandeur de la demande, en particulier sa capacité à maintenir ses fonctionnalités et ses performances en cas de forte demande.
2. SGBDR acronyme de Système de Gestion de Base de Données Relationnels

des objets de même nature qui se répétaient un certain nombre de fois, on en faisait une collection pour constituer un fichier. Mais ces fichiers, qui ont été construits pour répondre aux besoins des applications informatiques, possèdent la plupart du temps des éléments communs, des asscoiations qui ne sont pas exploitées du fait qu'ils sont utilisés isolément et indépendamment les uns des autres.[B1]

Face à ses limitations, le concept de base de données est apparu. Pour pouvoir interagir avec cette dernière on utilise un système de gestion de base de données (SGBD) qui est un logiciel permettant principalement d'organiser les données sur les supports périphériques et de fournir les procédures de description, manipulation et de contrôle de ces données.

3.2 Les systèmes de gestion de base de données relationnels

Un système de gestion de base de données relationnels est un type particulier des SGBD basé sur le modèl relationnel introduit par Edgar Frank Codd [3].

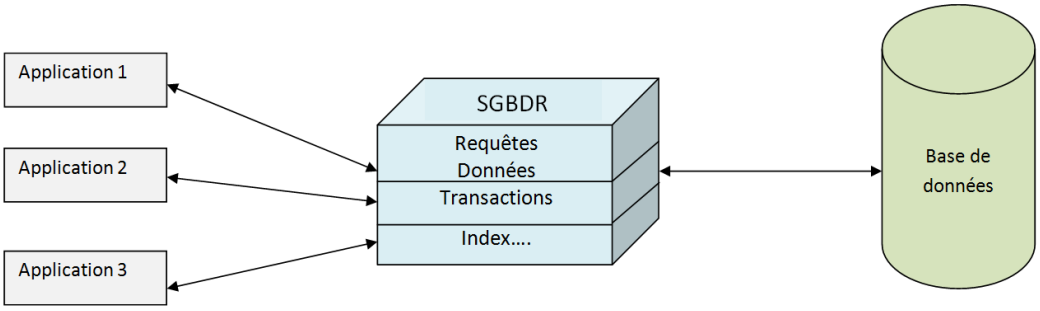

FIGURE 3.1 – Schèma d'un SGBDR

Dans cette section, nous expliquerons qu'est ce qu'un modèle relationnel, les règles et les contraintes d'une transaction dans les SGBDs relationnels. Nous allons également voir leurs points fort qui ont fait qu'ils sont les sytèmes les plus dominant. Enfin, nous citrons leurs limites à l'égard du Big data.

3.2.1 Le modèle relationnel

Tout d'abord, le modèle relationnel s'appuie sur le présupposé qu'un modèle mathématique peut servir de base à une représentation des données dans un système informatique. Cela se retrouve dans la conception même des entités que sont les tables des bases de données relationnelles.[B2]

3. Edgar Frank « Ted » Codd est un informaticien britannique. Il est considéré comme l'inventeur du modèle relationnel des SGBDR.

les bases de données relationnelles mettent à la disposition des développeurs un certains nombre d'opérations pour représenter les relations entre les tables :

- un système de jointure entre les tables permettant de construire des requêtes complexes faisant intervenir plusieurs entités (les tables en l'occurrence).
- un système d'intégrité référentielle permettant de s'assurer que les liens entre les entités sont valides.

3.2.2 Les régles de Codd

Les 12 règles de CODD ont été conçues pour définir ce qui est exigé d'un système de gestion de base de données afin qu'il puisse être considéré comme relationnel. Ces règles sont apparues dans deux articles de vulgarisation du magazine Computerworld (octobre 1985) : ***Is your DBMS really relational ? et Does your DBMS run by the rules ?*** Elles constituent les fondements des bases de données relationnelles et permettent de connaître le niveau relationnel du produit d'un éditeur. Ne pas s'y contraindre impliquera plus d'inconvénients que d'avantages. Tout bon système de bases de données relationnel devrait respecter ces régles suivantes à la lettre :[1]

- **Régle 0.** Toutes les fonctionnalités du SGBDR doivent être disponibles à travers le modèle relationnel et le langage d'interrogation.
- **Régle 1. Unicité :** Toute l'information dans la base de données est représentée d'une et une seule manière, à savoir par des valeurs dans des champs de colonnes de tables.
- **Régle 2. Garantie d'accès :** Toutes les données doivent être accessibles sans ambiguïté. Cette règle est essentiellement un ajustement de la condition fondamentale pour des clefs primaires. Elle indique que chaque valeur scalaire individuelle dans la base de données doit être logiquement accessible en indiquant le nom de la table contenante, le nom de la colonne contenante et la valeur principale primaire de la rangée contenante.
- **Régle 3. Traitement des valeurs nulles :** Une cellule peut ne pas contenir de valeur, ou exprimer que la valeur est inconnue, à l'aide du marqueur NULL. Il s'agit d'un indicateur spécial, distinct de toute valeur et traité de façon particulière.
- **Régle 4. Catalogue lui-même relationnel :** Le système doit supporter un catalogue en ligne, intégré, relationnel, accessible aux utilisateurs autorisés au moyen de leur langage d'interrogation régulier. Les utilisateurs doivent donc pouvoir accéder à la structure de la base de données (catalogue) employant le même langage d'interrogation qu'ils emploient pour accéder aux données de la base de données.

- **Régle 5. Sous-langage de données :** Le SGBDR doit implémenter un langage relationnel qui supporte des fonctionnalités de manipulation des données et des métadonnées, de définition de contraintes de sécurité et la gestion des transactions.

- **Régle 6. Mise à jour des vues :** Toutes les vues pouvant théoriquement être mises à jour doivent pouvoir l'être par le système.

- **Régle 7. Insertion, mise à jour, et effacement de haut niveau :** Le système doit supporter les opérations par lot d'insertion, de mise à jour et de suppression. Ceci signifie que des données peuvent être extraites d'une base de données relationnelle dans des ensembles constitués par des données issues de plusieurs tuples et/ou de multiples table. Cette règle explique que l'insertion, la mise à jour, et les opérations d'effacement devraient être supportées aussi bien pour des lots de tuples issues de plusieurs tables que juste pour un tuple unique issu d'une table unique.

- **Régle 8. Indépendance physique :** Les modifications au niveau physique (comment les données sont stockées, si dans les rangées ou les listes liées, etc.) ne nécessitent pas un changement d'une application basée sur les structures.

- **Régle 9. Indépendance logique :** Les changements au niveau logique (tables, colonnes, rangées, etc) ne doivent pas exiger un changement dans l'application basée sur les structures. L'indépendance de données logiques est plus difficile à atteindre que l'indépendance de donnée physique.

- **Régle 10. Indépendance d'intégrité :** les contraintes d'intégrité doivent être indépendantes des programmes clients et doivent être stockées dans le catalogue du SGBDR. On doit pouvoir modifier ces contraintes sans affecter les programmes clients.

- **Régle 11. Indépendance de distribution :** la distribution ou le partitionnement des données ne doivent avoir aucun impact sur les programmes clients.

- **Régle 12. Règle de non-subversion :** aucune interface de bas niveau ne doit permettre de contourner les règles édictées. Dans les faits, cette règle implique qu'il n'est possible d'interroger et de manipuler le SGBDR qu'à travers son langage relationnel.

L'ensemble de ces règles indique la voie à suivre pour les systèmes de gestion de bases de données relationnelles. Elles ne sont jamais totalement implémentées, à cause des difficultés techniques que cela représente.

3.2.3 Les contraintes des SGBDs relationnels

la plupart des moteurs de SGBDs relationnels se base sur le concept de la transaction [4] qui est une unité d'action qui leur impose le respect des contraintes **ACID** acronyme de : **A**tomicité **C**onsistance **I**solation **D**urabilité et qu'on nomme donc l'acidité de la transaction :[B3]

- **Atomicité :** Cela signifie que les mises à jour de la base de données doivent être atomiques, c'est-à-dire qu'elles doivent être totalement réalisées ou totalement annulées (c'est tout ou rien).

  A titre d'exemple, *sur 5000 lignes devant être modifiées au sein d'une même transaction, si la modification d'une seule échoue, alors la transaction entière doit être annulée. C'est primordial, car chaque ligne modifiée peut dépendre du contexte de modification d'une autre, et toute rupture de ce contexte pourrait engendrer une incohérence des données de la base.*

- **Cohérence :** Ou encore ce qu'on appelle la consistance. Cela signifie que les modifications apportées à la base doivent être valides, en accord avec l'ensemble de la base et de ses contraintes d'intégrité. Si un changement enfreint l'intégrité des données, alors soit le système doit modifier les données dépendantes, comme dans le cas classique d'une suppression en cascade, soit la transaction doit être interdite.

  Prenons l'example suivant : *Un système doit être capable de reconnaître qu'une facture est liée à un client et aux éléments facturés. Il doit être capable d'éviter, par exemple, la suppression d'un client s'il existe encore des factures pour ce client, et la suppression d'une facture qui a des éléments associés.*

- **Isolation :** Cela signifie que les transactions lancées au même moment ne doivent jamais interférer entre elles, ni même agir selon le fonctionnement de chacune.

  Par exemple, *si une requête est lancée alors qu'une transaction est en cours, le résultat de celle-ci ne peut montrer que l'état original ou final d'une donnée, mais pas l'état intermédiaire. De fait, les transactions doivent s'enchaîner les unes à la suite des autres, et non de manière concurrentielle.*

- **Durabilité :** Cela signifie que toutes les transactions sont lancées de manière définitive. Une base ne doit pas afficher le succès d'une transaction, pour ensuite remettre les données modifiées dans leur état initial. Pour ce faire, toute transaction est sauvegardée dans un fichier journal, de sorte que si un problème survient empêchant sa validation complète, la transaction pourra être correctement terminée lors de la disponibilité du système.

4. Au sein d'une base de données, le terme de "transaction" désigne les opérations apportant des modifications aux données. Par exemple, un virement bancaire provoquant le débit du compte de l'émetteur et le crédit du compte du bénéficiaire est une transaction.

3.2.4 Les forces des SGBDs relationnels

Le succés des SGBDs relationnels est dû essentiellement aux 12 régles de CODD et le modèle relationnel qui a permis de développer une vision de ce qu'est ou doit être une base de données. Parmis ces nombreux avantages nous citons :

1. **Cohérence transactionnelle forte :** garantissant la gestion de la concurrence, l'isolation entre les utilisateurs, la reprise sur panne.[B4]

2. **Séparation logique et physique :** il y a une forte indépendance entre :[B4]

   (a) Modèle de données et structures de stockage.

   (b) Requêtes déclaratives et exécution.

3. **Structuration forte des données :** Simplicité de présentation, les données sont sauvegardées dans des tables.

4. Capacité à gérer des requêtes très complexes.

5. Optimisation très fine des requêtes, index permettant un accès rapide aux données.

6. Logiciels mûrs, stables, efficaces, riches en fonctionnalités et en interfaces.

7. Contraintes d'intégrité permettant d'assurer des invariants sur les données.

3.2.5 Les limites des SGBDs relationnels

Bien que les SGBDs relationnels ont fait succès depuis plus de 20 ans. Néanmoins, un certain nombre de limitations importantes sont apparues. Les premiers acteurs qui ont rencontré ces limites furent les géants du web. Le constat était simple : ***les SGBDs relationnels ne sont plus une solution pour un environnement distribué requis par les volumes gigantesques de données, ce n'est pas tant le langage SQL en lui même qui est inadapté mais les grands principes sur lesquels il a été construit (le modèle relationnel et transactionnel que nous avons vu tout en haut).***

En effet, En Big Data, non seulement on doit gèrer des quantités de données très importantes , mais aussi on veut un résultat très rapidement. Or Les SGBDs relationnels traditionnels atteignent ici leurs limites à savoir :[2]

1. **Les structures de données :** sont devenues de plus en plus complexes avec en contrepartie des moteurs de stockage évoluant peu. Le principal point faible des modèles relationnels est l'absence de gestion d'objets hétérogènes ainsi que le besoin de déclarer au préalable l'ensemble des champs représentant un objet. Pour répondre au besoin des ces objets non pré-déclarés, on a vu apparaître des modélisations complexes sur le plan algorithmique comme le modèle Entity-Attribute-Value [5] permettant de séparer un objet et ses champs.[B3]

2. **un système de jointure entre les tables :** Concrètement un abus des jointures risque d'entrainer un traffic réseau conséquent accompagné d'un pic de consommation de la mémoire.

   En effet, Le stockage des données dans des HDFS entraine un split [6] des jeux de données en petits blocs de données afin de les répartir sur différents noeuds (fichiers distribués). Lorsque le data engineer [7] envoie une requête pour réaliser des jointures de table, nous assistons à une sollicitation importante des serveurs gérant tous les blocs de données nécessaires pour la réconciliation des données. Il faut donc limiter le plus possible les jointures de table très gourmandes en mémoire.[3]

3. **Contraintes d'intégrité :** les SGBD relationnels sont limités à des contrôles de cohérence simples. Il est donc difficile de modéliser des contraintes réelles correspondant aux données d'une entreprise. Ce problème n'est que partiellement résolu avec SQL-2 [8] qui permet d exprimer des contraintes dans la partie langage de définition.[B5]

4. **Surcharge sémantique :** le modèle relationnel s'appuie sur un seul concept (la relation) pour modéliser à la fois les entités et les associations (ou relations) entre ces entités. Il existe donc un décalage entre la réalité et sa représentation abstraite.[B5]

5. Un modèle EAV est un modèle dans lequel nous déclarons des objets applicatifs sous forme de trois éléments : Une Entité (Un produit Magento par exemple), des attributs (La description par exemple), et des valeurs typées (le lipsum de la description). Il déclare une table (entités) ou chaque attribut serait un champ et la valeur serait la donnée stockée.

6. Split est la première étape du modèle de programmation MapReduce. Cette étape est à la charge du framework qui se base sur le format des données. La taille de chaque découpage est fixe et généralement identique à la taille d'un bloc HDFS (64 Mo par défaut).

7. Data engineer -En français Ingénieur de données- est un travailleur dont les principales tâches consistent à préparer des données pour des utilisations analytiques ou opérationnelles. Les tâches spécifiques gérées par les ingénieurs de données peuvent varier d'une organisation à l'autre, mais comprennent généralement la création de pipelines de données pour rassembler des informations provenant de différents systèmes sources ; intégrer, consolider et nettoyer les données ; et en le structurant pour une utilisation dans des applications d'analyse individuelles.

8. SQL-2 est une amélioration du langage SQL-1 ; il est sous-divisé en trois niveaux, respectivement entrée, intermédiaire et complet. Le niveau entrée peut être perçu comme une amélioration de SQL1, alors que les niveaux intermédiaire et complet permettent de supporter totalement le modèle relationnel avec des domaines variés, tels date et temps.

5. **Scalabilité limitée :** nous distinguons deux types de scalabilité que les SGBDs relationnels ne peuvent gérer :

   (a) **la scalabilité des traitements :** représente la capacité à distribuer les traitements sur un nombre de machines important afin d'être en mesure d'absorber des charges très importantes.

   (b) **la scalabilité des données :** représente la capacité à répartir les données entre un nombre important de machines afin d'être en mesure de stocker de très grands volumes de données.

6. **Les contraintes ACID en milieu distribué :** dans le cas des systèmes distribués, il est nécessaire de distribuer les traitements de données entre différents serveurs. Il devient alors difficile de maintenir les contraintes ACID à l'échelle du système distribué entier tout en maintenant des performances correctes.

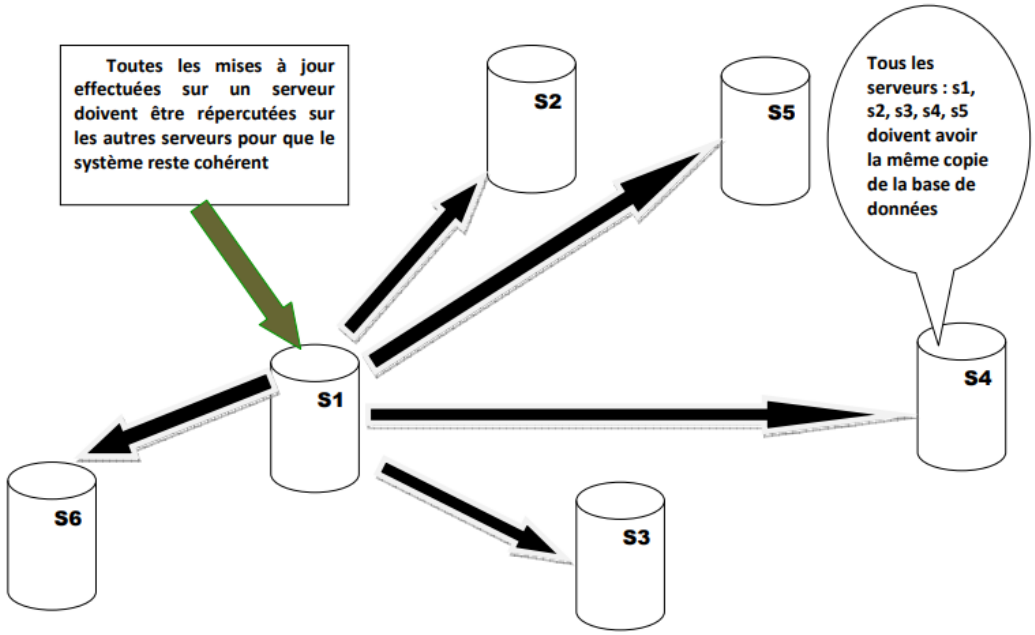

FIGURE 3.2 – Limites liées aux propriétés ACID

3.3 Les bases de données NoSQL

A l'heure où le big data explose, le mouvement NoSQL voie apparaitre le jour afin de proposer une alternative ou compléter les fonctionnalités des SGBDs relationnels pour donner des solutions plus intéressantes dans certains contextes à savoir la scalabilité.

Dans cette section nous exposons les concepts du NoSQL ; les différents types des bases NoSQL, le théorème CAP ainsi que leurs avantages et limites.

3.3.1 Definitions

Définition 1. NoSQL signifie ≪ Not Only SQL ≫ littéralement "pas seulement SQL". La définition apporte un début de réponse à la question "Est ce que le NoSQL va tuer les bases relationnelles ?". En effet, Ce n'est pas seulement du SQL et ce n'est pas du relationnel. Comme son nom l'indique, ce n'est pas un remplacent des SGBDR, mais il vient en complément de ces derniers.[4]

Définition 2. Le NoSQL regroupe de nombreuses bases de données, récentes pour la plupart, qui se caractérisent par une logique de représentation de données non relationnelles et qui n'offrent donc pas une interface de requês en SQL.[5]

Définition 3. Le terme NoSQL est pour la première fois proposé par Carlo Strozzi en 1998, lors de la présentation de son SGBDs relationnels open source. Il l'a appelé ainsi à cause de l'absence d'interface SQL pour interagir avec les bases de données. Les bases NoSQL peuvent bien entendu être utilisées avec du SQL, c'est pourquoi Strozzi précise qu'il est plus pertinent d'utiliser le terme ≪ NoREL ≫ car ces dernières s'abstraient du côté relationnel des données. Le mot réapparu en 2009 lorsqu'Eric Evans l'utilisa pour caractériser le nombre grandissant de bases de données non-relationnelles lors d'une présentation sur les bases de données distribuées open source.

3.3.2 Le théorème CAP

Le théorème **CAP** acronyme de ≪ **C**onsistency, **A**vailability and **P**artition tolerance ≫,qui désigne en français : ≪ Cohérence, Disponibilité et Résistance au partitionnement ≫ à été inventé par Eric Brewer [9] en 2000. Il s'agit d'une réflexion générale sur la conception de systèmes distribués, ce qui intéresse spécialement le monde du NoSQL pour les besoins de calcul distribué.[B3]

En 2002 Seth Gilbert et Nancy Lych [10], affirment qu'un système d'information à calcul distribué ne peut satisfaire que deux, parmi les trois contraintes suivantes :

- **Cohérence (Consistency) :** tous les noeuds du système voient exactement les mêmes données au même moment.

- **Disponibilité (Availability) :** Les requêtes d'écriture et de lecture sont toujours satisfaites c'est à dire ; la perte de noeuds n'empêche pas les survivants de continuer à fonctionner correctement.

9. est un scientifique américain, professeur émérite d'informatique à l'Université de Californie à Berkeley et vice-président de l'infrastructure de la société Google. Ses domaines de recherche comprennent les systèmes d'exploitation, l'informatique distribuée, le cloud computing et les réseaux de capteurs.

10. est une chercheuse américaine en informatique, professeur au MIT. Elle dirige le groupe de recherche sur la théorie des systèmes distribués. Elle a obtenu le prix Knuth en 2007 et le prix Dijkstra en 2001 et 2007

- **Résistance au partitionnement (Partition tolerance) :** La seule raison qui pousse un système à l'arrêt est la coupure totale du réseau. Autrement dit, si une partie du réseau n'est pas opérationnelle, cela n'empêche pas le système de répondre (ou encore : en cas de morcellement en sous-réseaux, chacun doit pouvoir fonctionner de manière autonome).[B3]

Afin de créer une architecture distribuée correcte, on est amené à choisir deux de ces propriétés qu'on vient de citer, laissant ainsi trois designs possibles comme illustrer dans le schéma ci-dessous.

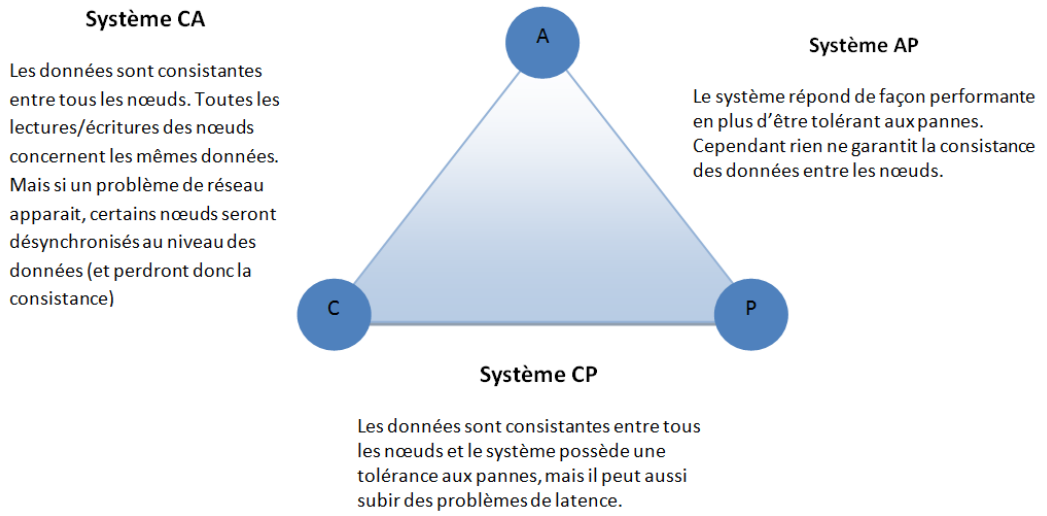

FIGURE 3.3 – Le théorème CAP

3.3.3 Les types des bases NoSQL

Dans la mouvance NoSQL, il existe une diversité importante d'approches que nous classons en quatre grandes catégories :

1. Les bases orientées clé-valeur.
2. Les bases orientées colonnes.
3. Les bases orientées documents.
4. Les bases orientées graphes.

3.3.3.1 Les bases de données orientées clé-valeur

Ce type est probablement le plus connu et le plus basique que comporte la mouvance NoSQL. Son principe est extrêmement simple : chaque objet (valeur) est identifié par une clé unique. Celle-ci représente la seule manière de solliciter l'objet.[B6]

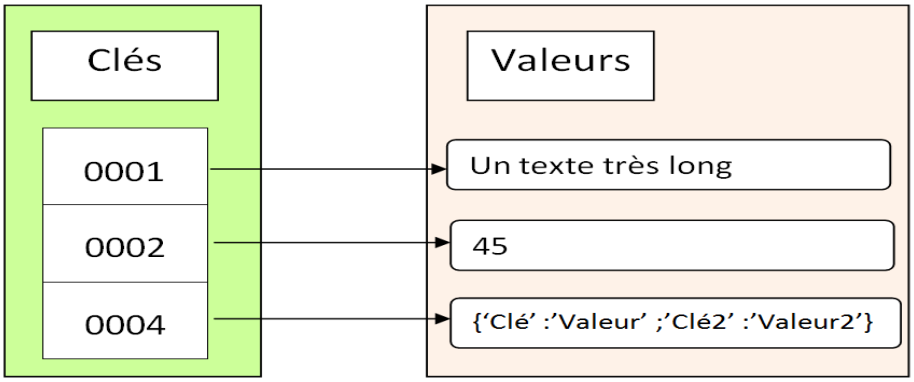

FIGURE 3.4 – Base de données orientée clé-valeur

La structure de l'objet est libre et le plus souvent laissé à la charge du développeur de l'application (XML, JSON, ...), la base ne gérant généralement que des chaînes d'octets. A fin de communiquer avec cette base, on se résume généralement qu'a quatre opérations Create Read Update Delete (CRUD)[B3] :

- **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).

La plupart des bases de données de type clé/valeur disposent d'une interface HTTP REST qui permet de procéder très facilement à des requêtes depuis n'importe quel langage de développement. Ces systèmes affichent des performances exceptionnellement élevées en lecture et en écriture, ainsi qu'une scalabilité horizontale [11] étendue, cela vient du fait que ces types de bases sont réduits à un simple accès disque.[B6]

3.3.3.2 Les bases de données orientées colonnes

Ce modèle ressemble à première vue à une table dans un SGBD relationnel à la différence que le nombre de colonnes est dynamique. En effet, le nombre de colonnes peut varier d'un enregistrement à un autre ce qui évite de retrouver des colonnes ayant des valeurs NULL.[B7]

Par ailleurs, c'est le modèle le plus complexe à comprendre parmi la mouvance NoSQL. Bien que l'on obtienne au final un schéma relativement proche des bases documentaires, l'organisation sous-jacente des données permet à ces bases d'exceller dans les traitements d'analyse de données et dans les traitements massifs (notamment via des opérations de type Map-Reduce [12]).

11. La scalabilité horizontale consiste à rajouter en parallèle des machines supplémentaires. C'est pourquoi on parle de croissance externe. On part d'une machine et on rajoute au fur et à mesure l'augmentation de charge de nouvelles machines identiques. Parmis ces plus grand avantage c'est qu'elle rend le système plus flexible.
12. Voir le chapitre 1 section : Les technologies du Big Data (Technologie de traitement)

Bien que l'on trouve des variations en fonction de la base considérée, les concepts essentiels sont les suivants :

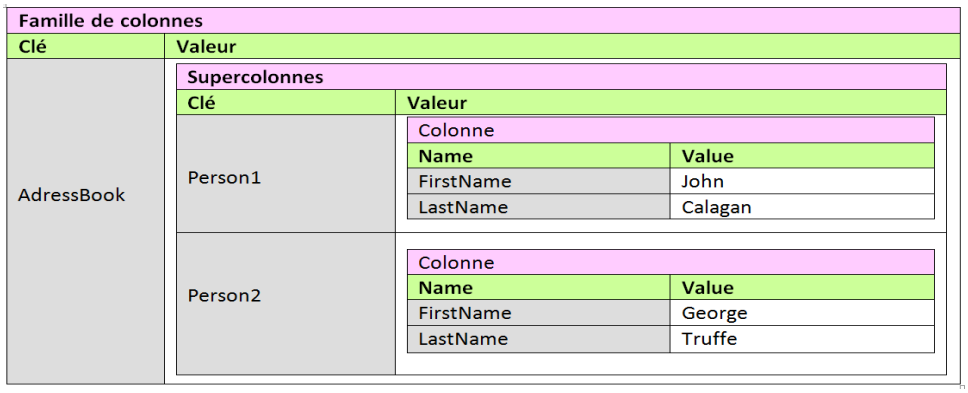

FIGURE 3.5 – Base de données orientée colonne

- **Colonne :** entité de base représentant un champ de donnée. Chaque colonne est définie par un couple clé-valeur.
- **Super colonne :** représente une colonne qui contient d'autres colonnes.
- **Famille de colonne :** il s'agit d'un conteneur permettant de regrouper plusieurs colonnes ou super-colonnes. Les colonnes sont regroupées par ligne et chaque ligne est identifiée par un identifiant unique.
- **Clé :** c'est l'identifiant unique de chaque ligne de colonne.
- **Valeur :** représente le contenu d'une colonne qui peut être aussi une elle même une colonne.

Exemple d'implémentation :

HBase : Utilise un API Java. Adopte un design CA. C'est un exemple qui sera détaillé dans la dernière section de ce chapitre.

3.3.3.3 Les bases de données orientées documents

Les systèmes de type documentaire sont composés de collections de documents. La représentation en document est une sorte d'extension du concept clé/valeur, qui se compose de champs et des valeurs associées. Ces dernières peuvent être, soit d'un type simple(entier, chaine de caractère, date, ...), soit elles-mêmes composées de plusieurs couples clé/valeur.[B3]

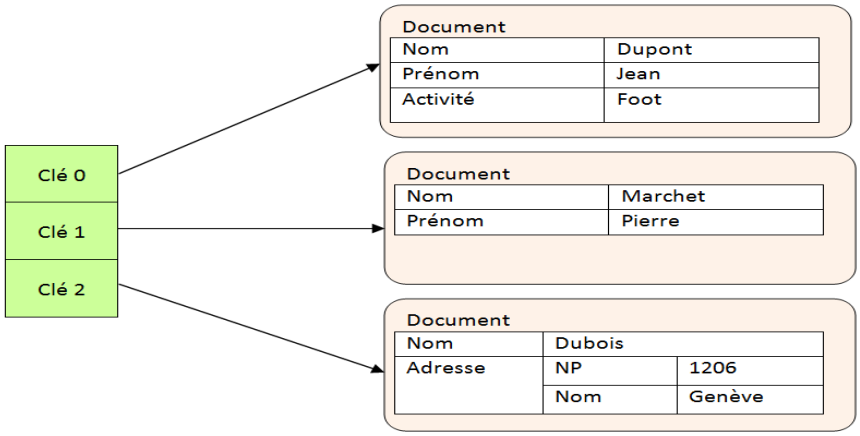

FIGURE 3.6 – Base de données orientée document

Par ailleurs, les documents sont organisés de manière hiérarchique à l'image de ce que permettent les fichiers de type XML ou JSON, elles sont dites « Schemaless » ou en français « sans schéma défini » ce qui signifie : il n'est pas nécessaire de définir au préalable les champs dans le document[B6]. De plus, le stockage structuré des documents leur confère des fonctionnalités simples dont la plus évidente est la capacité à effectuer des requêtes sur le contenu des objets.

3.3.3.4 Les bases de données orientées graphes

Les bases orientées graphe sont les moins connues de la mouvance NoSQL. Ces bases permettent la modélisation, le stockage ainsi que le traitement de données complexes reliées par des relations. Elle se compose de deux concepts ; un moteur de stockage pour les objets qui est une base documentaire où chaque entité de cette base est nommé nœud, et un mécanisme qui décrit les arcs : c'est les relations entre les objets, elles contiennent des propriétés de type simple (integer, string, date..).[B6]

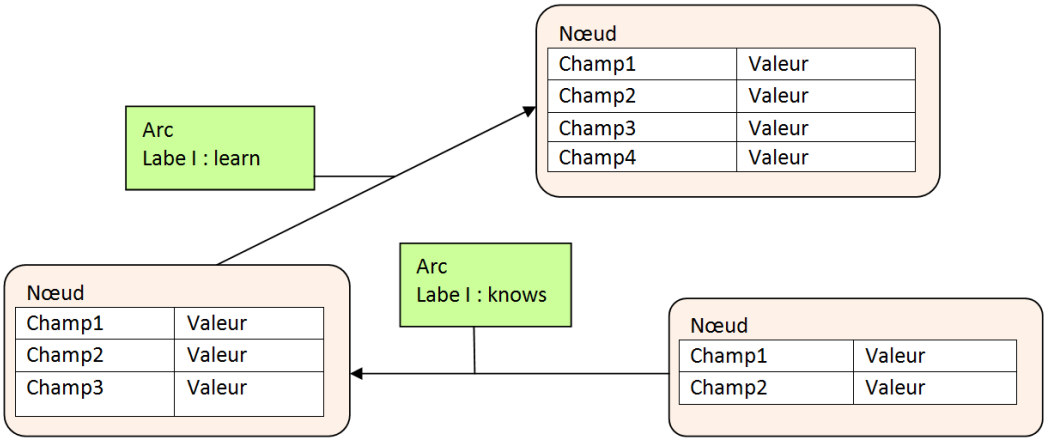

FIGURE 3.7 – Base de données orientée graphe

Exemple d'implémentation :

Neo4J : Développé en Java, supporte beaucoup de langages, propriétés ACID possibles, utilisé dans le monde des réseaux sociaux (Ex : amis sur Facebook) ; Langage de requêtes personnalisé « Cypher ».

3.3.4 L'apport du NoSQL

Les bases de données NoSQL reposent essentiellement sur plusieurs aspects qui font leurs forces et justifient leur usage aujourd'hui par les géants du web. Parmis ces points forts nous citons :

- **Scalabilité :** La scalabilité est le terme utilisé pour définir l'aptitude d'un système à maintenir un même niveau de performance face à l'augmentation de charge ou de volumétrie de données, par augmentation des ressources matérielles[B6].

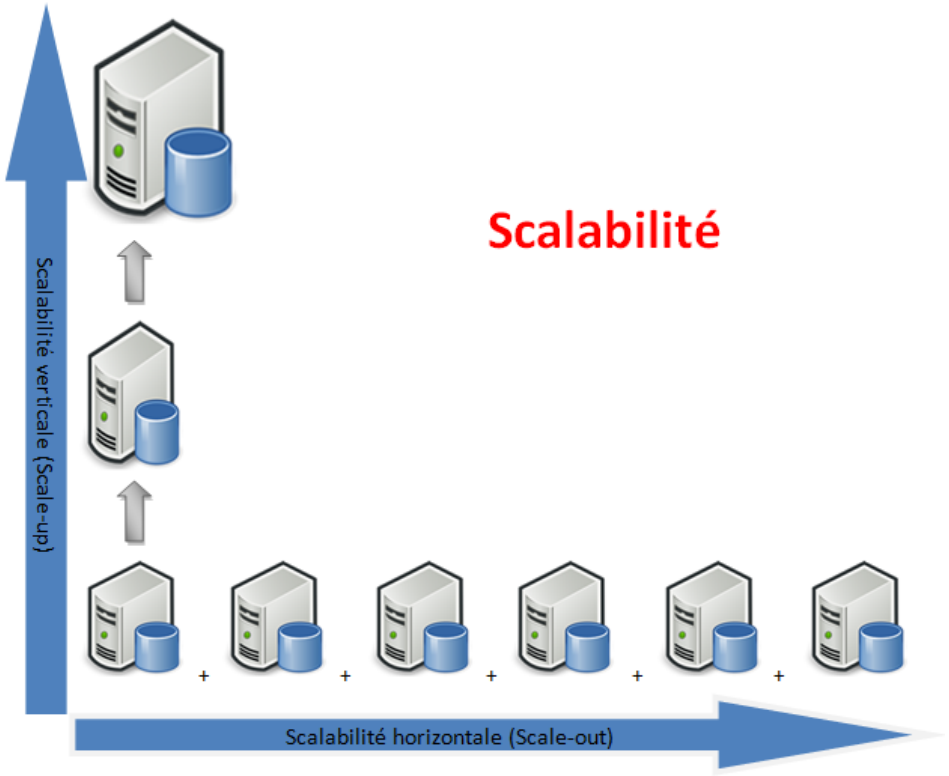

FIGURE 3.8 – La scalabilité verticale et horizontale

L'extensibilité est résolue de deux manières différentes :

- **Scalabilité verticale (interne) :** consiste à rajouter des ressources supplémentaires à la machine(CPU, RAM, disque, carte réseau, ...). C'est pourquoi on parle de croissance interne. On part d'une machine conçue pour évoluer et on rajoute au fur et à mesure de l'augmentation de la charge des ressources spécifiques.
- **Scalabilité horizontale (externe) :** consiste à rajouter en parallèle des machines supplémentaires. C'est pourquoi on parle de croissance externe. On part

d'une machine et on rajoute au fur et à mesure l'augmentation de charge de nouvelles machines identiques. Parmis ces plus grand avantage c'est qu'elle rend le système plus flexible.[B8]

- **Gestion de gros volume de données :** Le NoSQL est conçu pour supporter un nombre important d'opérations provenant d'un très grand nombre d'utilisateurs.

- **Performance en écriture :** Celle-ci à un intérêt principalement pour les géants : Google, Facebook, Twitter, gérant des masses de données qui augmentent considérablement chaque année, et exigeant un temps très important pour stocker ces données. En conséquence, l'écriture se doit être distribuée sur un cluster de machines, ce qui implique l'utilisation du MapReduce, la réplication, la tolérance aux pannes et la cohérence.[B8]

- **Performances en lecture** Les données sont distribuées sur plusieurs machines de ce fait on évite les goulots d'étranglements [13] lors de la récupération des données.[B8]

- **Type de données flexibles :** Parmis les innovations majeures, c'est le fait que les solutions NoSQL supportent de nouveaux types de données. Les objets complexes peuvent être mappés sans trop de problèmes avec la bonne solution.

3.3.5 Les inconvenients

Les avantages apportés par les bases de données NoSQL ne sont pas sans contreparties. En effet comme toute avancée technologique les bases de données NoSQL ont aussi des inconvénients, on en citera :

- **La maturité :** Les SGBD relationnels sont là depuis plus de 30 ans, ainsi, cette pérennité est un signe de réconfort pour les responsables vu qu'ils sont de nature stable et riche en fonctionnalités. En comparaison, la plupart des SGBD NoSQL sont en pré-production et ont encore beaucoup de fonctionnalités futures à implémenter.[B6]

- **Fonctionnalités réduites :** Les bases de données NoSQL sont principalement conçues pour stocker des données et offrent très peu de fonctionnalités autres que celle-là. Lorsque les transactions entrent dans l'équation, les bases de données relationnelles restent le meilleur choix. Voilà pourquoi les bases NoSQL ne pourront jamais remplacer entièrement SQL (ce n'est pas le but d'ailleurs).[B9]

- **L'administration :** Un des buts des bases de données NoSQL était d'en simplifier l'administration, mais la réalité est différente. Aujourd'hui, les solutions NoSQL demandent beaucoup de compétences pour leur installation ainsi que des efforts conséquents lors de leur maintenance.[B6]

13. Un goulot d'étranglement est un point d'un système limitant les performances globales d'un flux de production d'une entreprise. Le goulot d'étranglement, qu'on appelle aussi « ressource goulot », est défini par l'étape de production qui a la plus faible cadence dans un flux de production.

- **L'expertise :** On retrouve dans le monde des millions de développeurs, opérant, dans différents businesses, qui sont familiers des concepts ainsi que de la programmation dans un environnement de bases de données relationnelles. Dans le monde NoSQL, presque tous les développeurs sont en apprentissage avec la technologie.[B6]

3.4 Des SGBDR au NoSQL

Après avoir définit et bien étudier les SGBDs relationnels et les bases de données Nosql et ayant exposé leurs points fort et limites dans les deux sections précédentes ; nous venons ici faire une comparaison entre ces deux derniers afin de mieux comprendre l'apport du Nosql en regard des SGBDs relationnels et également voir les inconvénients des systèmes Nosql par rapport aux SGBDs relationnels.

• ***Les structures de données***	
SGBDR	**NoSQL**
• Le stockage de données est organisé sur le principe de tables reliées entre elles. • La structure et les types des données sont rigides, c'est-à-dire fixés à l'avance avant d'implémenter une logique métier. • Les attributs sont fortement typés, et selon les principes édictés par Codd, la normalisation correcte des structures implique qu'il n'y ait aucune redondance des données. • Les tables SQL imposent un modèle de données strictes, donc il est difficile de faire des erreurs.	• Le noSQL repousse les limites, il est beaucoup plus flexible. • Les données ne sont pas typées ni limitées. • On trouve moins de contraintes. • On est plus proche d'un agrégat que d'une table, parce que la clé revêt une plus grande importance.

• ***Le modèle et la logique***	
SGBDR	**NoSQL**
• Pas d'ajout de données tant qu'on ne définitc pas les tables et les types de champs dans ce que l'on appelle un schéma • Le schéma SQL contient d'autres informations : Clés primaires – index, contraintes, fonction, procédures stockées...	• Les données peuvent être ajoutées n'importe où et à tout moment. • Il n'est pas nécessaire de spécifier une conception de document ou même une collection à l'avance.

• *Les jointures*	
SGBDR	**NoSQL**
• Les requêtes SQL offrent une puissante clause JOIN. • Nous pouvons obtenir des données reliées dans plusieurs tables en utilisant une seule instruction SQL.	• NoSQL n'a pas toujours d'équivalent de JOIN.

• *Le partitionnement de données*	
SGBDR	**NoSQL**
• L'un des problèmes de la normalisation dans un SGBDR concerne la distribution des données et du traitement. ***Exemple :*** *Si on stocke des données qui ont un rapport entre elles, comme des clients,des commandes, des factures, des lignes de facture, etc., dans des tables différentes, on rencontrera un problème lorsque du partitionnement ces données. On doit alors s'assurer que les données en rapport les unes avec les autres se trouvent sur le même serveur.*	• La réplication est essentiellement destinée à pallier les pannes en dupliquant les données sur plusieurs serveurs et en permettant donc qu'un serveur prenne la relève quand un autre vient à faillir. • Le fait de disposer des mêmes données sur plusieurs serveurs par réplication ouvre également la voie à la distribution de la charge (en recherche, en insertion) et donc à la scalabilité.

• *L'integrité des données*	
SGBDR	**NoSQL**
• La plupart des bases de données SQL nous permettent d'appliquer des règles d'intégrité de données à l'aide de contraintes de clés étrangères. • Elles peuvent valider nos données et nos mises à jour (définition de contraintes), mais elles les rendent du coup très strictes.	• Ces options d'intégrité ne sont pas disponibles dans les bases de données NoSQL. • Elles sont plus flexibles mais ne valident pas les données. • Il en revient au développeur d'ajouter des contrôles dans la logique du programme car la base de données elle-même n'empêchera pas les erreurs. • Vous pouvez stocker ce que vous voulez indépendamment de tout autre document. • Idéalement, un seul document sera la seule source de toutes les informations sur un élément.

Scalabilité	
SGBDR	**NoSQL**
Il est souvent constaté que Le load balancing La répartition de charge (load balancing en anglais, littéralement équilibrage de charge) est une technique utilisée en informatique pour distribuer un travail entre plusieurs serveurs sur un serveur SQL est complexe car dû à la nature même dont les données sont stockées, organisées et reliées entre elles. Pour régler les problèmes de montées en charge, on admettra le plus souvent le principe d'un puissant serveur adossé à une infra de type failover Le Failover est une opération de sauvegarde dans laquelle les fonctions d'un système (processeur, serveur, réseau ou base de données, par exemple) sont assurées par les composants d'un système secondaire, lorsque le premier devient indisponible, soit à la suite d'une panne, soit en raison d'une interruption planifiée. plutôt que de « clusterer » les instances SQL. Cela a un impact direct sur les licences et donc le coût de ce type d'infrastructure.	Les modèles de données NoSQL peuvent rendre le processus plus facile et beaucoup d'entre eux ont été conçus nativement avec des fonctionnalités de « scalabilité élastique ». L'organisation des données en documents et la « denormalization » des collections permettent le partitionnement et autorise une montée en charge de la base de données sur le matériel courant déployé sur site ou dans le Cloud. Cela permet une croissance pratiquement illimitée.

• ***Transaction***	
SGBDR	**NoSQL**
• Plusieurs mises à jour peuvent être exécutées au sein d'une même transaction afin de garantir le succès ou l'échec de l'exécution du code SQL.	• La modification d'un document unique est atomique (si vous mettez à jour trois valeurs dans un document, les trois sont mis à jour avec succès ou ils restent inchangés. • Il n'y a pas vraiment d'équivalent de la transaction SQL pour les mises à jour de plusieurs documents. • Il semble alors clair qu'il faille gérer la notion d'atomicité dans le code.

• ***Performance***	
SGBDR	**NoSQL**
• Le SQL est moins préformant en revanche, la conception du globale projet et les exigences en matière d'organisation de données seront déterminant avant même le choix du SGBD • Une base de données SQL bien conçue sera certainement supérieure à celle d'un équivalent NoSQL mal conçu et vice versa.	• NoSQL est plus rapide que SQL. • Le principe de ≪ denormalization ≫ induit une représentation plus simple et permet donc de récupérer toutes les informations sur un élément spécifique dans une seule requête, il n'y a donc pas besoin de liens JOIN ou de requêtes SQL complexes. • la redondance des informations alourdie considérablement les opérations de mise à jour.

TABLE 3.1 – SGBDR vs NoSQL

3.5 Exemples de bases de données NoSQL

Comme nous l'avons vu, la famille des bases de données NoSQL se compose de plusieurs catégories. Dans cette section nous allons présenter les plus utilisées et les plus populaires sur le marché à savoir :

1. Google BigTable.
2. Apache Cassandra.
3. MangoDB.
4. Apache HBase.
5. Firebase.

3.5.1 Google BigTable

1. **Définition**
   BigTable est un système de gestion de base de données compressées, haute performance, propriétaire, développé et exploité par Google. C'est une base de données orientée colonnes, dont se sont inspirés plusieurs projets libres, comme HBase, Cassandra ou Hypertable.[6]

FIGURE 3.9 – Google BigTable.

2. **Caractéristiques de Google BigTable**
   BigTable a été conçue pour répondre à de grandes montées en charge tout en gardant une latence faible même avec d'importants débits de données. Parmis ses caractéristiques nous citons :

   - Bigtable offre de faibles latences et un haut débit quel que soit le niveau et le type d'application. Elle est utiliser comme moteur de stockage à grande échelle ou pour des applications nécessitant une grande réactivité et une grande capacité de traitement des données.
   - Bigtable provisionne et découpe automatiquement des centaines de pétaoctets de données et peut facilement gérer des millions d'opérations par seconde. La configuration du déploiement est instantanée et en cas de modifications les répercussions sont immédiates. Les nœuds de cluster Bigtable peuvent être ajoutés et supprimés dynamiquement. Les modifications peuvent être faites en production, sans redémarrage.
   - Bigtable s'intègre facilement avec la plupart des outils de Big Data comme Hadoop et Spark, de la même manière qu'avec un ensemble d'outils de la plate-forme Google Cloud tel que Dataflow [14], BigQuery [15], ou Dataproc [16]. Bigtable utilise la même API que HBase, la base de données native Hadoop, ce qui permet la portabilité des applications entre HBase et Bigtable.

14. DataFlow Group a été fondé en 2006 et son siège social est situé à Dubaï. La société dispose d'un réseau de 60 000 autorités émettrices réparties dans plus de 200 pays, ainsi que de 620 experts et chercheurs au travers de ses sites mondiaux, DataFlow Group sert une base de données de clients couvrant 200 pays.Parmi les clients de la société figurent des organisations gouvernementales, quasi gouvernementales, réglementaires et multinationales à travers le monde.

15. BigQuery est un service web RESTful qui permet l'analyse interactive massive de grands ensembles de données en collaboration avec l'espace de stockage Google. C'est un logiciel en tant que service (SaaS) qui peut être utilisé en complément de MapReduce.

16. Google Cloud Dataproc est un service géré Spark et Hadoop en nuage proposé sur la plate-forme Google Cloud. Il utilise de nombreuses technologies Google Cloud Platform telles que Google Compute Engine et Google Cloud Storage pour offrir des clusters entièrement gérés exécutant des infrastructures de traitement de données populairestelles que Apache Hadoop et Apache Spark.

3. **Les principaux avantages de Google BigTable**
   Parmi les avantages d'une solution Google BigTable nous avons :

   - Elle répond en moins d'un centième de seconde dans 99% des cas, soit 50 fois plus rapide que la plupart des solutions du marché.
   - Elle possède une gestion des droits d'accès avec des ACL, Access Controle Liste.
   - Toutes les données sont cryptées à la fois à la volée et au repos.
   - La construction des index est très puissante.
   - Elle bénéficie de la performance de Google Cloud Platform.

3.5.2 Apache Cassandra

1. **Définition**
   Apache Cassandra est une base de données NoSQL orienté colonnes. Initialement créé par Facebook, ce système est désormais open source. Cassandra compte parmi les bases de données NoSQL les plus efficientes et les plus utilisées pour le stockage et le traitement de larges volumes de données.[7]

FIGURE 3.10 – Apache Cassandra.

2. **Caractéristiques d'Apache Cassandra**
   Apache Cassandra peut être définie par plusieurs caractéristiques essentielles, on en citera :

   - Il s'agit d'une base de données orientée colonne.
   - Elle est hautement consistante, tolérante aux erreurs et scalable.
   - Le modèle de données est basé sur Google Bigtable.
   - Prise en charge de propriétés telles que l'atomicité, la cohérence, l'isolation et la durabilité (ACID).
   - Le modèle de réplication de données reprend celui de Amazon Dynamo [17], mais y apporte des améliorations par le biais de son modèle de données orienté colonne.

17. Amazon DynamoDB est une base de données de clés-valeurs et de documents, offrant des performances de latence de l'ordre de quelques millisecondes, quelle que soit l'échelle. Il s'agit d'une base de données multi-région et multi-maître entièrement gérée avec un système intégré de sécurité, de sauvegarde et de restauration et de mise en cache en mémoire pour les applications à l'échelle d'Internet.

3. **Les principaux avantages d'Apache Cassandra**
   Cassandra est utilisée par certaines des plus grandes entreprises du monde : **Facebook**, **Netflix**, **Twitter**, **Cisco** voici certains des avantages qui lui permettent de se distinguer de la concurrence :

   - La capacité de prendre en charge les données structurées, non structurées ou semi-structurées. Elle est également capable de supporter les changements dynamiques apportés aux structures de données pour s'adapter aux besoins changeants.
   - Son architecture est scalable de façon linéaire. Il suffit d'ajouter de noeuds pour l'adapter à une hausse de la demande. En outre, les données peuvent être distribuées de façon homogène sur de multiples centres de données par le biais d'un processus de réplication des données.
   - Cette base de données est également très fiable, car les éventuelles défaillances de noeuds n'affectent par les performances générales. Cassandra se distingue aussi par son impressionnante vitesse d'écriture de données.

3.5.3 MangoDB

1. **Définition**
   MongoDB est la plus connue des bases de données NoSQL. Il s'agit d'une base de données Open-Source orientée document développé par MongoDB Inc depuis 2007. MongoDB est une base de données évolutive et accessible. Elle peut également être utilisé comme système de fichiers. Dans MongoDB, JavaScript peut être utilisé comme langage de requête.[8]

FIGURE 3.11 – MangoDB.

2. **Caractéristiques de MangoDB**

   - Fournit un rendement élevé.
   - Exécution sur plusieurs serveurs.
   - Garantit la scalabilté horizontale (réplication et sharding).
   - Prise en charge de la réplication maître-esclave.
   - Les données sont stockées sous forme de documents de style JSON.
   - indexer n'importe quel champ d'un document.
   - MangoDB a une configuration d'équilibrage de charge automatique en raison des données placées dans des tessons.
   - Facile à administrer en cas d'échec.

3. **Principaux avantages de MangoDB**

   - Facile à installer.
   - MongoDB Inc. offre un soutien professionnel à ses clients.
   - Prise en charge des requêtes ad hoc.
   - Base de données haute vitesse.
   - Base de données sans schéma.
   - Base de données évolutive horizontalement.
   - La performance est très élevée.

3.5.4 Apache Hbase

1. **Définition**
   HBase est une base de données distribuée et non relationnelle, reprenant le concept et les fonctionnalités de Google BigTable. La différence est que HBase est Open Source. L'un des principaux objectifs de HBase est d'héberger des milliards de lignes et des millions de colonnes. On peut ajouter des serveurs à tout moment pour augmenter la capacité et de multiples nœuds maîtres assureront une haute disponibilité des données. Cette base de données s'exécute généralement sur le système de fichiers distribués Hadoop (HDFS). Elle offre un accès d'écriture et de lecture en temps réel, aléatoire et cohérent.[8]

FIGURE 3.12 – Apache Hbase.

2. **Caractéristiques d'Apache Hbase**

   - Prise en charge de l'échec automatique.
   - Elle est de type orientée colonnes.
   - Linéairement évolutif.
   - Permet la réplication des données.
   - S'intègre à Hadoop, à la fois comme source et comme destination. Les tables de cette base de données peuvent servir d'entrée pour les tâches MapReduce dans l'écosystème Hadoop, et peuvent aussi servir de sortie après traitement des données par MapReduce.

3. **Principaux avantages d'Apache Hbase**

   - Fournit des recherches rapides pour les grandes tables.
   - Fournit un accès à faible latence à des rangées individuelles à partir de milliards d'enregistrements.
   - API Java facile pour le client
   - Auto-sharding.
   - C'est une base de données sans licence.
   - Gére de grands ensembles de données sur le dessus du stockage de fichiers HDFS.
   - Flexible sur la conception des schémas.
   - Haute vitesse.

3.5.5 Firebase

1. **Définition**
   Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application (Android, iOS, Javascript, Node.js, Java, Unity, PHP, C++ ...). Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de l'authentification sociale (Google, Facebook, Twitter et Github), et des notifications, ou encore des services, tel que par exemple un serveur de communication temps réel.[9]

FIGURE 3.13 – Firebase.

2. **Caractéristiques de Firebase**
   La plateforme firebase, se compose de deux types de bases de données NoSQL qui sont **Firebase Realtime Database** et **Firebase Firestore**.

   - **Firebase Realtime Database :**
     Est la base de données originale. Il s'agit d'une solution efficace à faible temps de latence pour les applications mobiles qui nécessitent des états synchronisés entre les clients en temps réel. Les données sont stockées sous forme de JSON [18] et synchronisées en temps réel sur chaque client connecté.

     Ce mode de stockage implique l'absence de tables et d'enregistrement, car les données sont représentées en tant que nœud dans l'arborescence JSON dans le cloud. Chaque nœud a un identifiant de clé unique qu'on peut nous même fournir ou laisser à Firebase le faire.

18. JSON : JavaScript Object Notation, format d'échange de données en texte lisible. Utilisé pour représenter des structures de données.

- **Firebase Firestore :**
  Est la nouvelle base de données phare de Firebase pour le développement d'applications mobiles. C'est aussi une base de données NoSQL, hébergée sur le cloud de Google (Firebase) et orientée Document. Les données collectées sont sauvegarder dans des documents qui sont organisés en collections. Chaque document contient un ensemble de paires clé-valeur.

Remarque : *Dans le cadre de notre travail, nous nous intéresserons plus particulièrement aux deux bases de données* ***Apache Hbase*** *et* ***Firebase*** *qui seront étudiées dans le chapitre suivant.*

Conclusion

En conclusion, nous pouvons dire que le NoSQL est une technologie en plein essor qui, en plus de ses performances techniques avérées, constitue un véritable marché qui profite aux entreprises du secteur. Il est important de noter que le NoSQL n'est en aucun cas une négation du SQL, ni un remplaçant du SQL. En ce qui concerne l'avenir du NoSQL nous pouvons affirmer que cette technologie cohabitera avec le SQL qui demeure pour l'heure le langage le plus utilisé.

Dans la foulée des solutions NoSQL, de nombreux projets libres ont vu le jour. Même si une grande diversité règne, plusieurs caractéristiques fréquentes semblent émerger : absence de schéma, partitionnement horizontal sur un grand nombre de noeuds, dé-normalisation, réplication automatique, etc.

Comme nous l'avons vu, la plupart des SGBD de la mouvance NoSQL ont donc été construits en faisant fi des contraintes ACID, quitte à ne pas proposer des fonctionnalités transactionnelles.

Le groupe NoSQL, vise en premier lieu à démontrer qu'il existe désormais des solutions alternatives utilisées au sein de systèmes exigeants. L'objectif est évidemment d'obtenir une meilleure disponibilité des données, au moyen de capacités de partitionnement étendues mais au prix d'un relâchement des contraintes des propriétés ACID.

Chapitre 4

Apache Hbase et Firebase

Introduction

Chaque seconde le nombre de données générées augmente. Face à cette augmentation il est devenu plus difficile que jamais d'extraire des informations précieuses à partir de cet ensemble de données volumineuses. Les scientifiques du secteur de l'information doivent maintenir et assembler de multiples plates-formes hétérogènes afin de pouvoir gérer des tâches d'analyse de données volumineuses afin de réduire les niveaux d'expertise et d'effort requis pour que les Data Scientist puissent travailler avec le Big Data. Nous étudierons dans ce chapitre Apache Hbase et Firebase afin de montrer leurs avantages et inconvénients et la façon dont ils répondent aux caractéristiques suivantes :

- **La scalabilité.**
- **Le temps de réponse des requêtes.**
- **La sémantique des données.**

Tout au long de ce chapitre, il convient de commencer par la présentation des concepts de base de ces deux bases de données en mettant en avant leur mode de fonctionnement, caratéristiques, le modèle de données...etc

4.1 Apache HBase

4.1.1 Présentation

Apache HBase est une base de données NoSQL orienté colonne reprenant le concept et les fonctionnalités de Google BigTable [1]. La différence est que HBase est Open Source. Cette base de données s'exécute généralement sur le système de fichiers distribués Hadoop (HDFS). Elle offre à des tables contenant des milliards de lignes et des millions de colonnes : un accès d'écriture et de lecture en temps réel, aléatoire et une forte cohérence pour de vastes quantités de données non structurées et semi-structurées, dans une base de données sans schéma.[1]

1. BigTable est un système de gestion de base de données compressées, haute performance, propriétaire, développé et exploité par Google. C'est une base de données orientée colonnes, dont se sont inspirés plusieurs projets libres, comme HBase, Cassandra ou Hypertable.

Il s'agit d'un très bon choix pour le stockage de données multi-structurel. Il est aussi possible d'effectuer des requêtes pour un repère temporel spécifique, ce qui permet de réaliser des requêtes « flashback »[2].[2]

FIGURE 4.1 – Apache Hbase.

4.1.2 Concepts de base

La base de données HBase est régie par les concepts suivants :

- **Map :** Le stockage se fait dans une map. Cette dernière est basée sur le principe de clé/valeur. Chaque valeur (tableau de bytes) est identifiée par une clé (tableau de bytes). L'accès à une valeur par sa clé est très rapide.
- **Map trié :** La map est triée par ordre lexicographique. Cette fonctionnalité de tri est très importante car elle permet de récupérer les valeurs par intervalle de clés.
- **Multidimensionnel :** La clé dans la map est une structure composée de row-key, column family, column, et d'un timestamp.

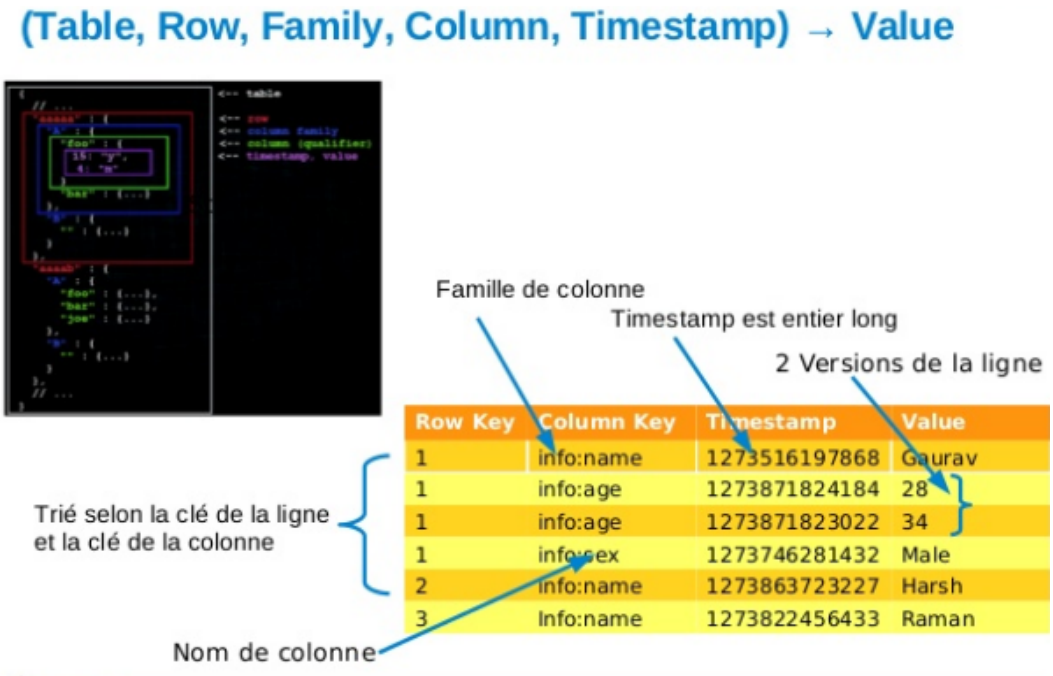

FIGURE 4.2 – Structure d'une clé dans la map.

2. Le principe du FlashBack Query consiste à requêter une table via une clause SELECT + une clause "AS OF" permettant de préciser à quelle date (Time Stamp) ou à quel Numero SCN on souhaiterait voir l'image de la table.

- **Null (Sparse) :** Contrairement aux bases de données relationnelles, une colonne qui n'a pas de valeur n'est pas matérialisée (aucun stockage n'est nécessaire en cas d'une valeur null pour une colonne).
- **Persistance :** Les données stockées dans la map sont sauvegardées durablement sur disque.
- **Consistance :** Toutes les modifications sont atomiques et les lectures se font toujours sur la dernière valeur validée (commit).
- **Système distribué :** Le système de base de données est construit sur un systèmes de fichiers distribués afin que le stockage de fichiers sous-jacent soit réparti sur un ensemble de machines d'un cluster. Les données sont répliquées sur un certain nombre de nœuds permettant ainsi une tolérance aux pannes.

4.1.3 Architecture

HBase est un système de gestion des données pour un environnement distribué qui est couplé au gestionnaire de fichiers d'Hadoop où : **Hbase** gère la **partie logique**, tandis qu'**Hadoop** gère la **partie physique**.

1. **Aspect logique (Avec Hbase)**
   Le modèle se base sur six concepts, qui sont :

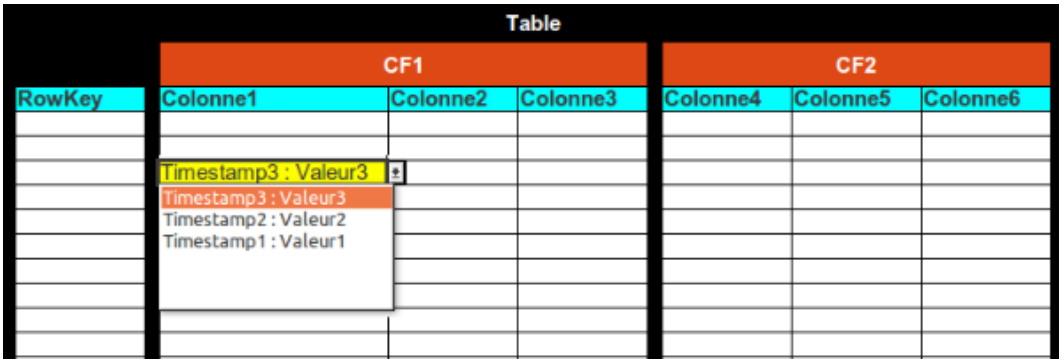

FIGURE 4.3 – Vue synthétique du modèle de données dans HBase.

- **Htable :** Les données sont organisées au sein de grandes tables. Les noms de tables sont des chaînes de caractères.
- **Row (Lignes) :** dans chaque table les données sont organisées dans des lignes. Une ligne est identifiée par une clé unique (RowKey). La Rowkey n'a pas un type, elle est traité comme un tableau d'octets.
- **Column Family (Familles de colonnes) :** Les données au sein d'une ligne sont regroupées par column family. Chaque ligne de la table a les mêmes column family, qui peuvent être peuplées ou pas. Les column family sont définit à la création de la table dans HBase, et chaque famille de colonne est divisée en sous colonnes.
- **Column qualifier :** -les colonnes- L'accès aux données au sein d'une column family se fait via le column qualifier ou column. Ce dernier n'est pas spécifié à la création de la table mais plus tôt à l'insertion de la donnée. Comme les rowkeys, le column qualifier n'est pas typé, il est traité comme un tableau d'octets.

- **Cell :**La combinaison du RowKey, de la Column Family ainsi que la Column qualifier identifie d'une manière unique une cellule. Les données stockées dans une cellule sont appelée les valeurs de cette cellule. Les valeurs n'ont pas de type, ils sont toujours considérés comme tableau d'octets.
- **Version :** Les valeurs au sein d'une cellule sont versionnés. Les versions sont identifiés par leur timestamp (de type long). Le nombre de versions est configuré via la Column Family. Par défaut, ce nombre est égale à trois.

Du fait que toutes les données sont logiquement placées dans la même table, les jointures sont inutiles.

2. **Aspect physique (Avec Hadoop)**
HBase est basé sur la même architecture physique qu'Hadoop puisque ce dernier gère le stockage physique. On retrouve donc une configuration distribuée en cluster de plusieurs noeuds qui peuvent être hétérogènes au niveau hardware.[3]

Le principe est donc le même que celui d'Hadoop. Seuls les noms diffèrents. La figure suivante montre les principaux composants d'une architecture HBase :

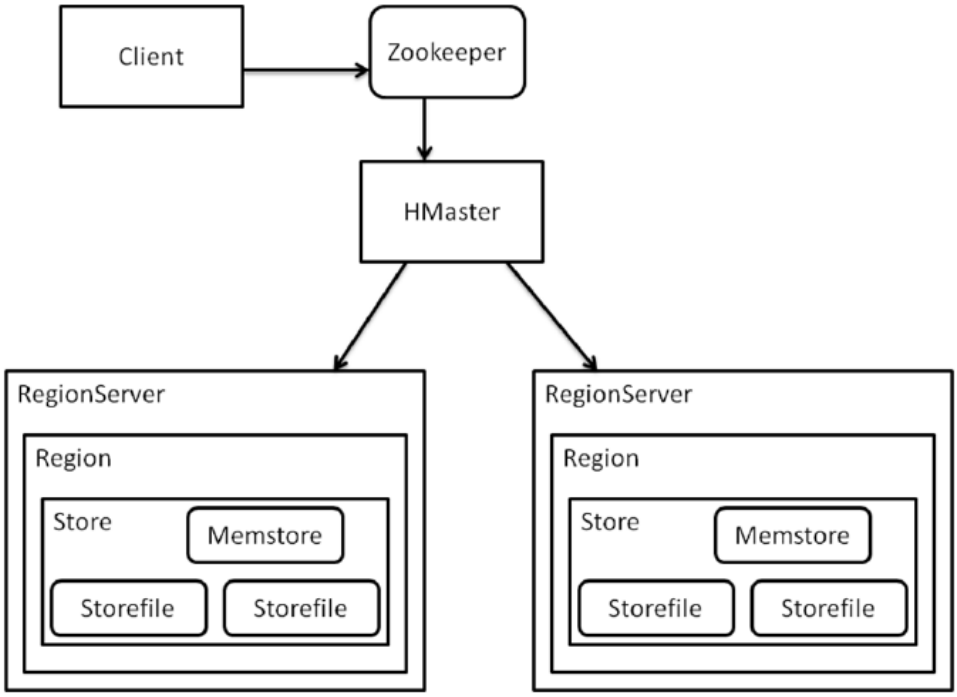

FIGURE 4.4 – Architecture HBase.

(a) **RegionServer**
Le RegionServer est le point d'entrée pour l'accès à la donnée. C'est le serveur maître où les données sont stockées il gère d'une part plusieurs processus et d'autre part plusieurs composants :

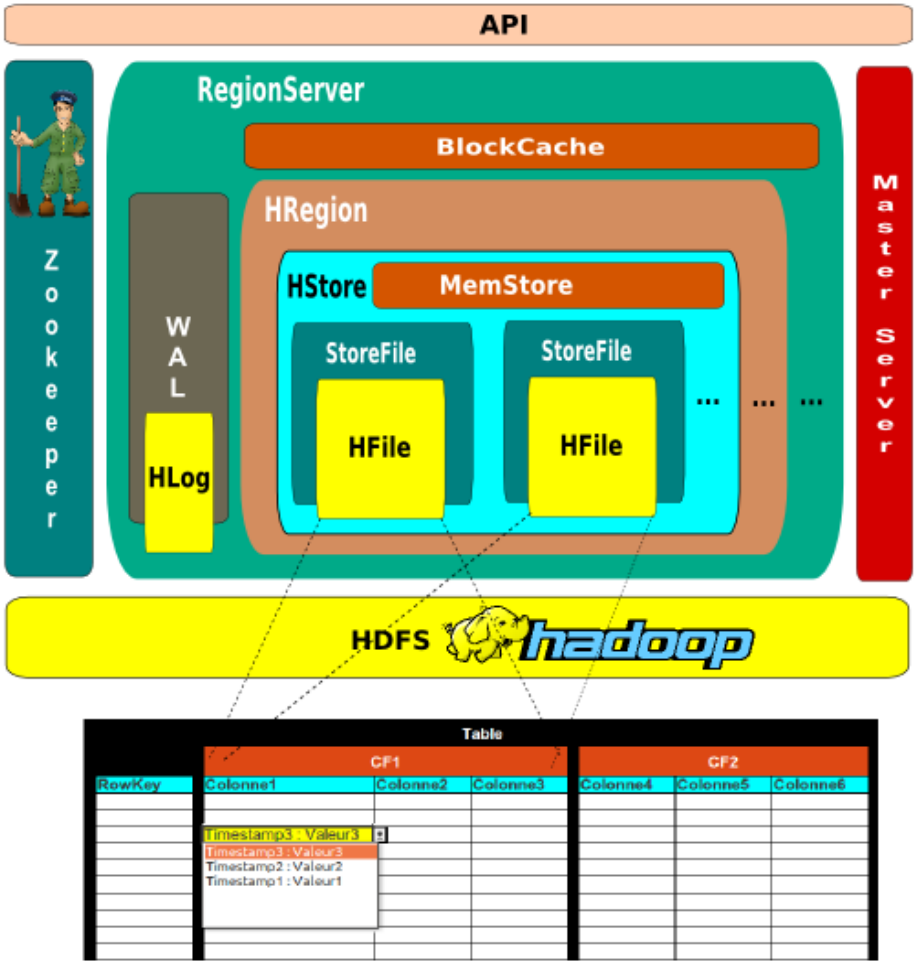

FIGURE 4.5 – Anatomie de la base de données NoSQL HBase.

(b) **La répartition des données :**

- Les lignes d'une HTable sont scindées en paquets réguliers par le regionServer.
- Chaque paquet est dirigé vers un serveur-esclave où il est de nouveau divisé par le memstore selon les colonnes de la HTable.

(c) **La gestion d'un WAL, un BlockCache et plusieurs Régions :**

- **BlockCache :** Le BlockCache est un cache LRU. Il est activé par défaut pour toutes les tables, ce qui signifie que toute opération de lecture sont chargées dans le cache LRU.

- **Write Ahead Log (WAL) :** Chaque ajoute ou mises à jour dans un RegionServer sont systématiquement écrit dans write-ahead log (WAL) en premier lieu. Avant d'être répliquer dans le MemStore. Ce mécanisme garantit la durabilité de la donnée en cas de défaillance du serveur. Le processus WAL écrit dans le fichier Hlog qui est placé dans HDFS. Pour chaque instance RegionServer il y a une instance de WAL.

- **Region :** C'est l'élément de base dans le stockage et la distribution de la donnée dans HBase, dont l'implémentation est HRegion. Chaque Region gère un sous ensemble d'une table HBase (une partition). Une Region est composé de :
  - **Store :** Chaque partition d'une ColomnFamily est gérée par un store. HStore est l'implémentation d'un Store, il est compose de plusieurs StoreFiles et d'un MemStore.
  - **MemStore :** C'est un cache en mémoire. Il stocke toutes les écritures et les mises à jours relatives à une partition.
  - **Hfile :** fichier physique sur le quel les données sont sauvegardées.

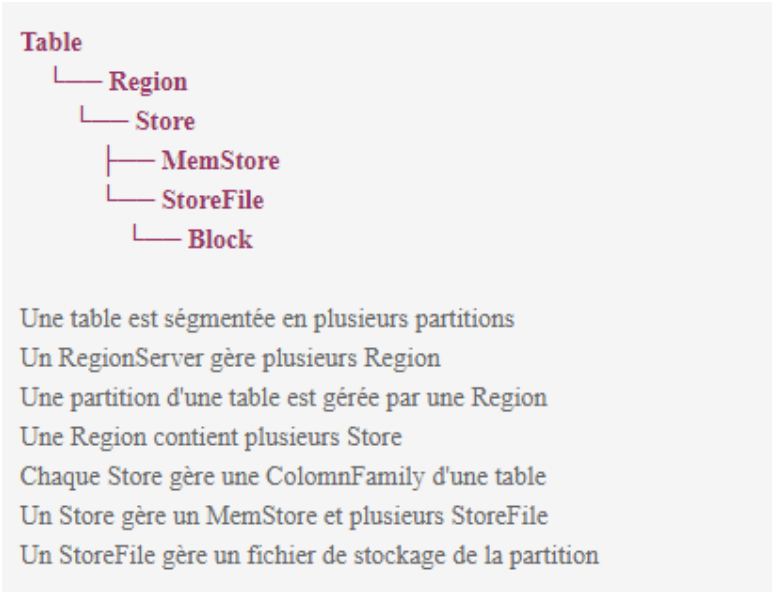

FIGURE 4.6 – Recapitulatif du rôle de Region.

(d) **Zookeeper**

Le projet open-source Apache ZooKeeper est un système open-source de synchronisation et de coordination des systèmes distribués, il permet de maintenir des informations de configuration. Il propose aussi des services synchronisés et des services de groupe pour une large variété d'applications distribuées. Cette technologie est déployée sur un cluster Hadoop pour administrer l'infrastructure.[4]

ZooKeeper facilite la synchronisation entre les process en maintenant un statut sur les serveurs ZooKeeper qui stocke l'information sur des fichiers de log locaux. Les serveurs ZooKeeper sont en mesure de supporter un large cluster Hadoop. Chaque machine client communique avec l'un des serveurs pour retrouve l'information.

Exemple : *Soit un cluster Hadoop constitué de plus de 500 serveurs. Compte tenu du nombre de serveurs, une gestion centralisée du cluster s'impose pour les services de noms, de synchronisation ou pour la configuration et bien plus encore. En utilisant ZooKeeper, il est possible d'éviter d'avoir à développer des services de synchronisation en partant de zéro.*

En effet il surveille l'état du cluster et informe régulièrement le MasterServer des différents états. De plus, il stocke les informations critiques du cluster, dans l'emplacement de la table système -ROOT-.

- **ROOT :** contient la liste des tables .META. et leurs emplacements.
- **META :** contient la liste des Regions et leurs emplacements.

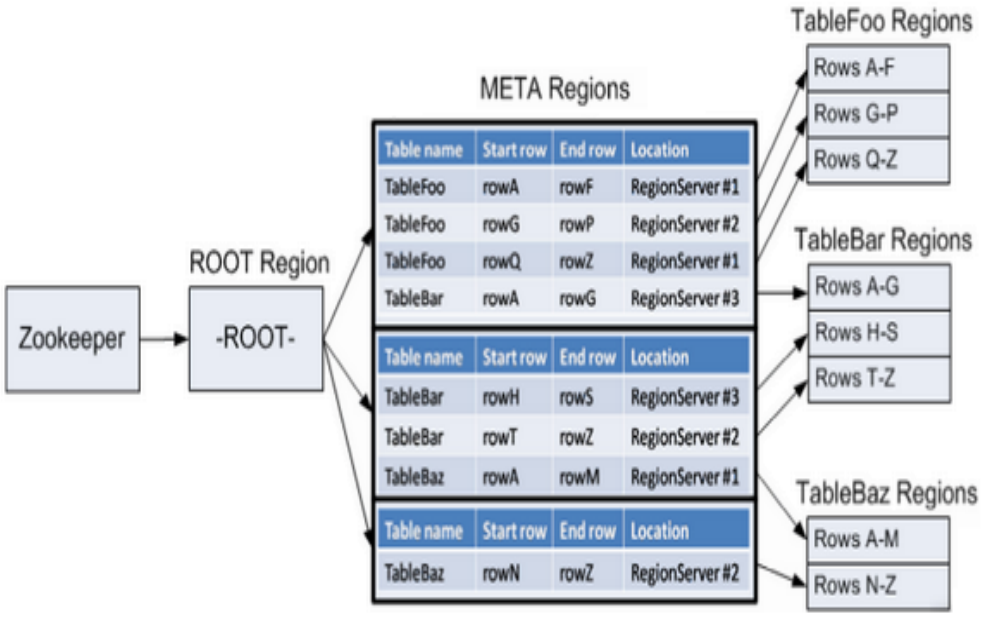

FIGURE 4.7 – Gestion des métadonnées avec ZooKeeper.

3. **MasterServer**
   HMaster est l'implémentation du Master Server. Le Master Server est chargé de coordonner et de surveiller toutes les instances de RegionServer du cluster. Il en charge de la répétition des Region(s) sur les nœuds du cluster. Les changements dans les tables metadonnées passe par le serveur Master Server.[5]

4.1.4 Lecture/Écriture dans HBase

La lecture et l'écriture des données dans HBase passent par plusieurs étapes comme le montre la figure suivante :

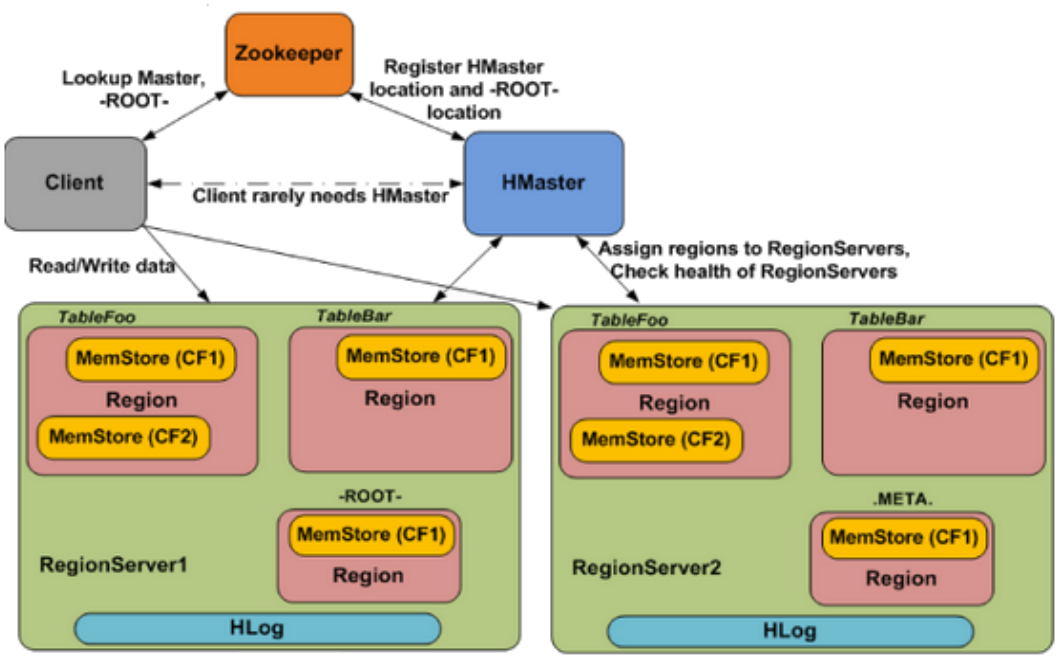

FIGURE 4.8 – Interaction entre le client HBase et les différents composant du cluster HBase.

1. **La lecture**
   En premier lieu le client repère la Region qui est responsable de la partition souhaitée. Pour se faire, le client contacte ZooKeeper afin de récupérer la table ROOT. A partir de cette dernière une fois reçu, le client récupéré l'emplacement de la tables META. Cette dernière fournie au client la localisation de la Region qu'il cherche, ensuite il envoi directement une requête de type « get » à ce Server Region qui contient l'information souhaitée.[5]

2. **L'écriture**
   Les données envoyées par le client Region Server sont stockées d'une manière temporaire dans le memstore, ensuite elles sont triées et indexées d'une manière continue. Lorsque le memstore atteint un certain seuil, l'ensemble des données triées sont écrites dans un nouveau fichier HFile dans HDFS. Régulièrement ces nouveaux fichiers sont fusionnés comme le montre la figure ci-dessus.[5]

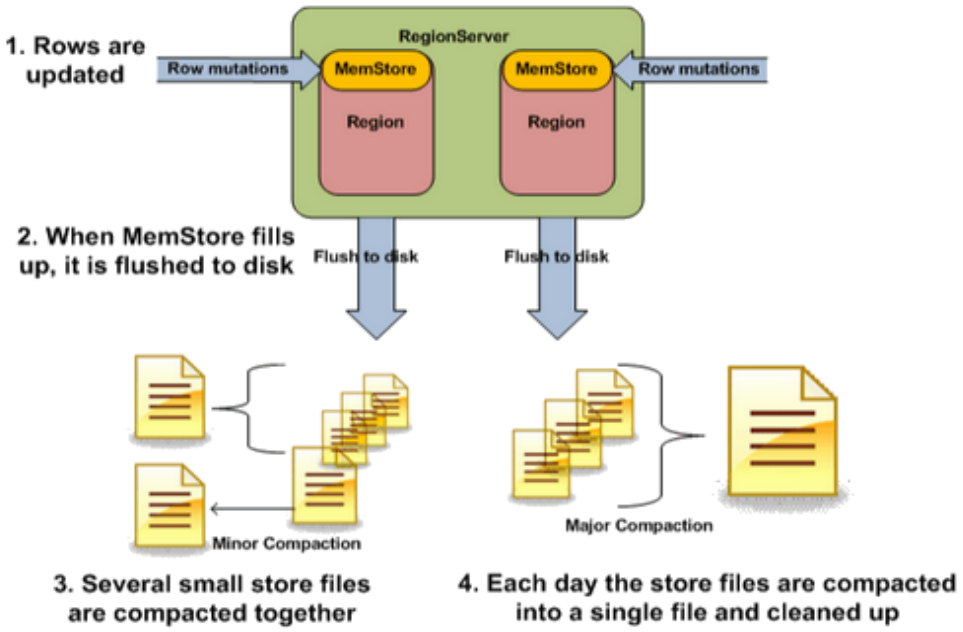

FIGURE 4.9 – Fusion des fichiers HBase.

Bien que l'écriture des données dans le memestore est efficace mais néanmoins elle reste risqué.

Afin de palier à ce risque que HBase commence par écrire les données dans le write-ahead log (WAL) avant de les stocker dans le memstore. Cette façon de faire, permet de récupérer les données en cas d'un dysfonctionnement dans le Region Server, à partir de la sauvegarde réalisées dans le WAL.

Une Region Server sert de nombreuses régions, mais ne dispose que d'un fichier WAL pour toutes ces Region(s). Comme le fichier WAL grossie avec le temps, arrivé à un certain seuil, il est fermé et archivé. L'archivage des fichiers est géré par un système de roulement.

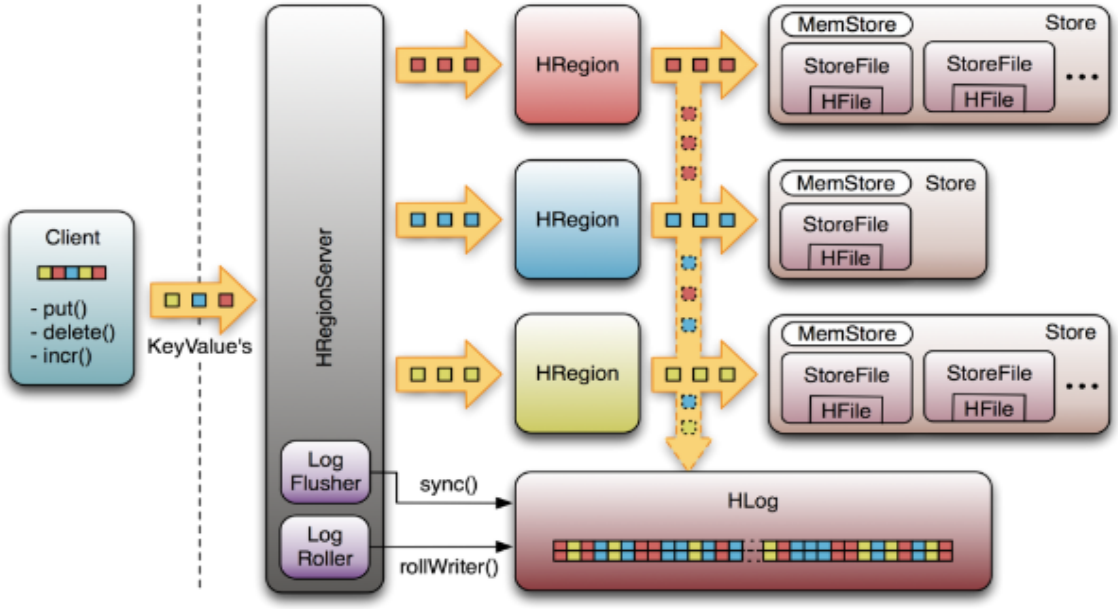

FIGURE 4.10 – Schématisation des écritures dans HBase.

4.1.5 Avantages de HBase

Les avantages de HBase sont nombreux, en plus de ceux que nous avons énoncé dans le chapitre précédent nous citons :

- **Assure l'extensibilité et la scalabilité.**
- **Importante tolérance aux pannes :** Les données sont répliquées sur les différents serveurs du Data Center, et un failover (basculement) automatique garantit une haute disponibilité.
- **Rapidité :** Il propose des lookups en temps réel ou presque, et un processing server side [3] par le biais de filtres et de coprocesseurs. Un caching in-memory est également proposé.
- **Distribution transparente de la donnée :** Répartition de la charge est faite par le système lui même.
- **Les langages de programmation supportés :** Hbase supporte beaucoup de langage de programmation : C, C#, C++, Groovy, Java, PHP, Python, Scala.
- ***Utile pour les données de capteurs :*** *HBase est très adapté aux données collectées de façon incrémentielle à partir de diverses sources. Cela inclut l'analytique sociale, les séries chronologiques, la mise à jour des tableaux de bord interactifs avec les tendances et les compteurs, et la gestion de systèmes de journal d'audit.*

3. L'expression server-side (côté serveur) fait référence à des opérations qui sont effectuées par le serveur dans la communication entre client et serveur dans un réseau informatique.

4.1.6 Inconvénients de HBase

- Ne supporte pas la transaction.
- Pas de permissions ou d'authentification intégrée.
- L'indexation et le tri se fait uniquement sur clé.
- Point de défaillance unique (lorsqu'un seul HMaster est utilisé).
- Ne supporte pas la structure SQL et XML.
- Problèmes de mémoire sur le cluster : Il y a une très faible capacité en mémoire.
- HBase ne vérifie pas le respect des contraintes d'intégrité référentielle et sémantiques.

4.2 Firebase

4.2.1 Concepts généraux

1. **Définitions**

Définition.1 Firebase est une plateforme mobile créée en 2011 par James Tamplin et Andrew Lee [4] sous le nom d'Envolve, puis rachetée par Google en 2014 pour être intégrée à leur offre de services Cloud (Google Cloud Platform). Son objectif premier est de libérer les utilisateurs de la complexité de création et de la maintenance d'une architecture serveur, tout en garantissant une scalabilité à toute épreuve (plusieurs milliards d'utilisateurs) et une simplicité dans l'utilisation.[6]

FIGURE 4.11 – Firebase.

Définition.2 Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application (Android, iOS, Javascript, Node.js, Java, Unity, PHP, C++ ...). Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de l'authentification sociale (Google, Facebook, Twitter et Github), et des notifications, ou encore des services, tel que par exemple un serveur de communication temps réel.[7]

4. James Tamplin et Andrew Lee : les fondateur de Firebase, ils ont construits une société appelée Envolve, qui offrait des outils de discussion en temps réel sous forme de widgets pouvant être intégrés à un site Web

2. **Les produits de Firebase :**
   Firebase est conçu dans le but de libérer les utilisateurs de la complexité de création et de la maintenance d'une architecture serveur, tout en garantissant une scalabilité à toute épreuve (plusieurs milliards d'utilisateurs) et une simplicité dans l'utilisation.[6]

   Pour cela, Firebase a été décomposée en plusieurs produits extrêmement riches et adaptés au monde du mobile dont nous allons citer quelques uns.

   - **Cloud Firestore :** Base de données NoSQL orientée documents de Firebase, permettant de facilement stocker, synchroniser et récupérer des données distantes pour une application mobile.

   - **Storage :** Espace de stockage de Firebase dédié au stockage et à la récupération de fichiers propres à l'utilisateur comme des photos ou des vidéos.

   - **Authentication :** Solution permettant de créer et gérer facilement des moyens d'authentification variés (Google, Facebook, Email, etc...) dans le but de sécuriser l'accès à une application mobile et authentifier les utilisateurs.

   - **Cloud Messaging :** Fournit un flux de communication fiable et économe en batterie entre le serveur (Firebase) et les appareils distants (où l'application est installée) dans l'objectif d'envoyer et recevoir des messages de notifications.

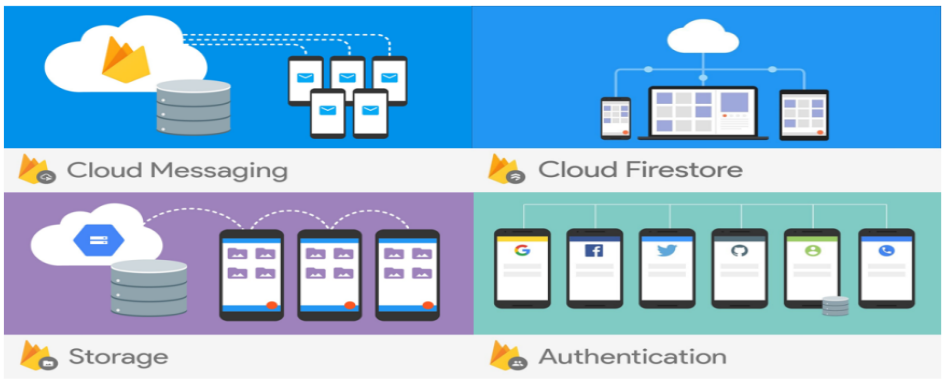

FIGURE 4.12 – Quelques produits Firebase.

4.2.2 Les bases de données de Firebase

La plateforme firebase, se compose de deux types de bases de données NoSQL qui sont **Firebase Realtime Database** et **Firebase Firestore**.

1. **Firebase Realtime Database :**
   Est la base de données originale. Il s'agit d'une solution efficace à faible temps de latence pour les applications mobiles qui nécessitent des états synchronisés entre les clients en temps réel. Les données sont stockées sous forme de JSON [5] et synchronisées en temps réel sur chaque client connecté.[8]

   Ce mode de stockage implique l'absence de tables et d'enregistrement, car les données sont représentées en tant que nœud dans l'arborescence JSON dans le cloud. Chaque nœud a un identifiant de clé unique qu'on peut nous même fournir ou laisser à Firebase le faire.[7]

   Pour la base de données Firebase Realtime Database nous allons parler de :

   (a) La structuration des données.
   (b) Les caractéristiques de Firebase Realtime Database.
   (c) Les avantages de Firebase Realtime Database.

   (a) **Structuration de données :**
   La construction d'une base de données nécessite une planification de la manière dont les données vont être sauvegardées puis récupérées pour rendre ce processus aussi simple que possible.

   Toutes les données de la base de données Firebase Realtime sont stockées en tant qu'objets JSON. En effet quand on ajoute des données à l'arborescence JSON, cela devient un nœud dans la structure JSON existante avec une clé associée. Comme nous pouvons fournir nos propres clés, telles que des identifiants d'utilisateur ou des noms sémantiques.

   Par exemple : *considérons une application de discussion qui permet aux utilisateurs de stocker un profil de base et une liste de contacts. Un profil utilisateur typique se trouve sur un chemin, tel que / users / $ uid. L'utilisateur utilisateur1 peut avoir une entrée de base de données qui ressemble à ceci :*

```
{
  "users": {
    "alovelace": {
      "name": "Ada Lovelace",
      "contacts": { "ghopper": true },
    },
    "ghopper": { ... },
    "eclarke": { ... }
  }
}
```

FIGURE 4.13 – Structure d'une arboresence JSON

5. JSON : JavaScript Object Notation, format d'échange de données en texte lisible. Utilisé pour représenter des structures de données.

(b) **Les caractéristiques de Realtime Database**

- **Synchronisation en temps réel :** la base de données en temps réel de Firebase utilise la synchronisation des données. Ainsi, chaque fois que les données changent, tous les clients connectés sont mis à jour, aucune demande HTPP traditionnelle n'est nécessaire.

- **Prise en charge automatique hors ligne :** le SDK offre la persistance du disque pour une synchronisation gratuite et automatique lorsque l'application est remise en ligne. Celà nous permet de ne plus se soucier de l'état hors connexion. En effet on peut ajouter, modifier et supprimer des données de manière transparente lorsqu'on est hors ligne, et le SDK se chargera de tout synchroniser correctement lorsque l'application sera de nouveau en ligne.

(c) **Avantages de la base de données Firebase Realtime database**
Cette base de données comporte un ensembles d'avantages, dont nous citons quelques uns :

- C'est une base de données NoSQL stockée dans le cloud de Google, donc disponible partout.
- Mise à jour en temps réel sur plusieurs plates-formes et plusieurs utilisateurs.
- Base de données multiplateforme.
- Google héberge les données afin de ne plus se soucier du matériel ni d'autres choses.

2. **Firebase Firestore :**
Est la nouvelle base de données phare de Firebase pour le développement d'applications mobiles[6]. C'est aussi une base de données NoSQL, hébergée sur le cloud de Google (Firebase) et orientée Document. Les données collectées sont sauvegarder dans des documents qui sont organisés en collections. Chaque document contient un ensemble de paires clé-valeur. [9]

Pour mieux comprendre les base de données Firebase Firestore nous nous intéresserons aux :

(a) Les elements de structuration des données.
(b) Les types de données supportées.
(c) L'architecture des données dans Firestore.

(a) **Les elements de structuration des données :**
La structuration des données dans Firestore se distingue par les 3 elements suivants :

- **La donnée brute (DATA) :**représente la donnée qu'on souhaite sauvegarder dans une base de donnée, elle prend toutes les formes que la base firestore peut supporter, par exemple un entier, une chaine de caractère, un tableau, etc.

- **Le document (DOCUMENT) :**les données brute déjà générer, ne seront pas sauvegardées au hasard et n'importe oú, mais elles sont obligatoirement rattachées à un document, sous forme de propriétés/champs.

- **La collection (COLLECTION) :** Permet d'organiser sous forme de liste les documents. En effet, elle peut contenir un à plusieurs documents.

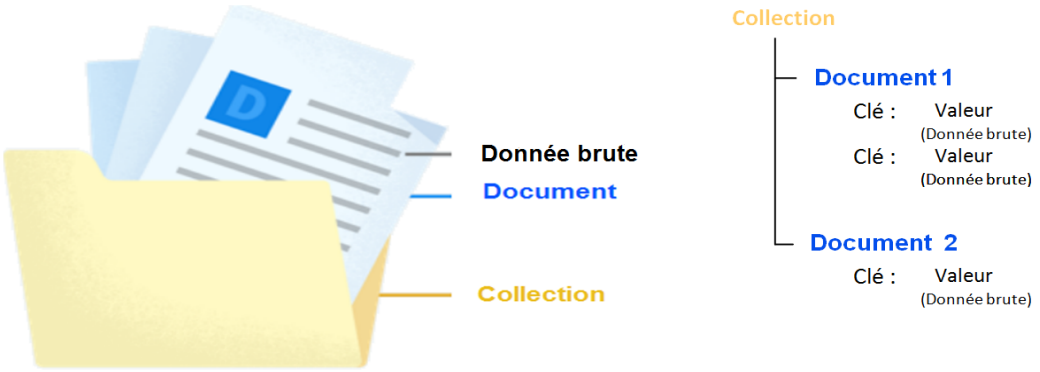

FIGURE 4.14 – Structure de données sous Firestore

(b) **Les types de données supportées :**
Le cloud Firestore prend en charge un certain nombre limité de types de données, et décrit aussi l'ordre de tri utilisé lors de la comparaison de valeurs du même type.[10]

Le tableau ci-dessous illustre les différents types de données :

Type de données	**Ordre de tri**	**Remarques**
Array	par valeurs d'élément	**Exemple :** [1, 2, 3] ¡[1, 2, 3, 1] ¡[2]. Le tableau [2] a la plus grande valeur de premier élément. Le tableau [1, 2, 3] a des éléments égaux aux trois premiers éléments de [1, 2, 3, 1] mais il est plus courte en longueur.
Boolean	true /false	-
Octets	Ordre des octets	Jusqu'à 1 048 487 octets (1 Mo - 89 octets). Seuls les 1 500 premiers octets sont considérés par les requêtes.
Date et heure	Chronologique	Une fois stocké dans Cloud Firestore, précisez-le uniquement à la microseconde ; toute précision supplémentaire est arrondie.
Nombre à virgule flottante	Numérique	double précision, 64 bits, IEEE 754.
Point géographique	Par latitude puis longitude	-
Entier	Numérique	64 bits, signé
Carte	par clés, puis par valeur	**Exemple :** si vous écrivez (c : "foo", a : "bar", b : "qux"), la carte est triée par clé et enregistrée comme (a : "foo", b : "bar", c : " qux ").
Null	Aucun	-
Référence	Par éléments de chemin (collection, ID de document, collection, ID de document ...)	**Exemple :** projects/[PROJECT-ID]/databases/[DATABASE-ID]/documents/[DOCUMENT-PATH]
Chaîne de texte	Ordre des octets codés UTF-8	Jusqu'à 1 048 487 octets (1 Mo - 89 octets). Seuls les 1500 premiers octets de la représentation UTF-8 sont pris en compte par les requêtes.

Table 4.1 – Types de données supportées

(c) **Architecture des données dans firestore**
Afin d'organiser les données, Firestore nous offre trois architectures possible pour mener à bien notre structure, à savoir les données imbriquées dans les documents, les sous-collections et les collections de niveau racine.[11]

- **Les données imbriquées dans les documents :** ou bien ≪ Nested data in documents ≫ en anglais, est une architecture qui permet d'imbriquer des objets complexes tels que des tableaux ou des cartes dans des documents. Facilite la configuration et la rationalisation de la structure des données si un utilisateur se dispose d'une liste simple et fixe et qu'il veut conserver ces données dans son document. En parallèle, si les données se développent avec le temps, la liste et le document augmentent eu aussi en croissance, ce

qui permet d'avoir une latence dans la récupération des documents.

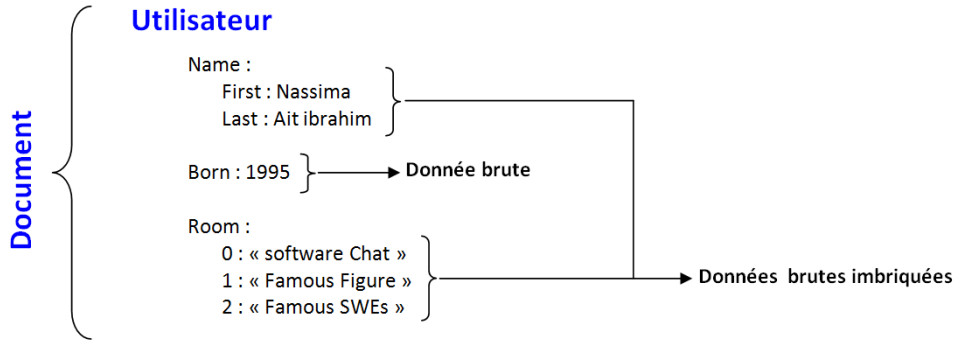

FIGURE 4.15 – Les données imbriquées dans les documents

Exemple : *Dans une application de discussion, on peut stocker les 3 dernières salles de discussion visitées par un utilisateur en tant que liste imbriquée dans son profil.*

- **les Sous-collections :** ou bien « Subcollections » en anglais, est l'architecture qui permet à un utilisateur de créer des collections dans des documents lorsque il y a des données capable de se développer avec le temps. Son avantage est que si une liste s'allonge, la taille du document parent reste inchangée, aussi, elle offre des fonctionnalités de requête complètes sur les sous-groupes. Comme elle apporte un inconvénient sur la difficulté de supprimer un sous-groupe.

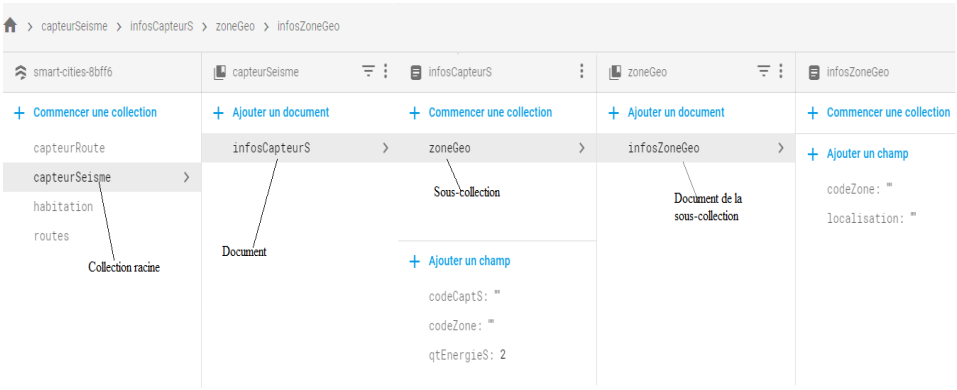

FIGURE 4.16 – Sous-collections

Exemple : *Dans la même application de discussion, on peut créer des collections d'utilisateurs ou des messages dans des documents de salle de discussion.*

- **les collections de niveau racine :** ou « Root-level collections » en anglais, est le type d'architecture qui offre la possibilité de créer des collections au niveau racine de la base de données pour organiser des ensembles de données différentes. Parmi ses avantages on retrouve : une grande flexibilité et évolutivité, et des requêtes puissantes au sein de chaque collection. Mais cela ne l'empêche pas d'avoir des limites, a savoir la complexité d'obtenir des données naturellement hiérarchique si a base de données devient plus grande.

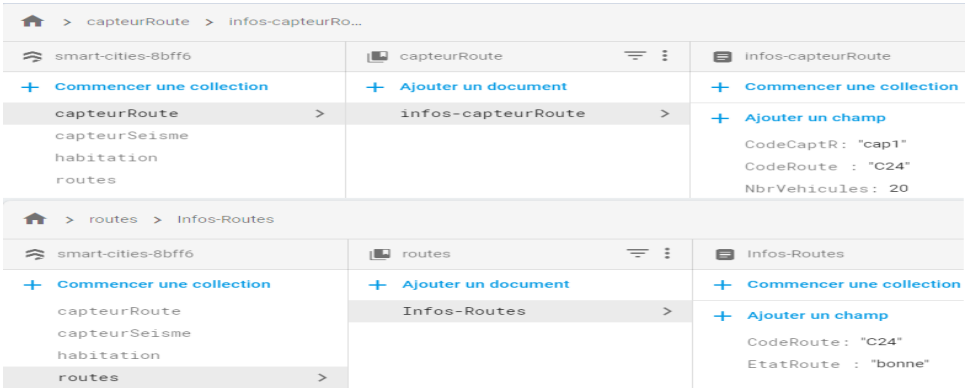

FIGURE 4.17 – collections de niveau racine

Exemple : *Dans la même application de discussion, on peut créer une collection pour les utilisateurs et une autre pour les salles et les messages.*

4.2.3 Comparaison entre les bases de données de Firebase

Très souvent on se voit confronté à choisir ce qui est le plus convenable et adaptable à notre projet. Pour celà une étude comparative entre les bases de données de Firestore est nécessaire afin de mieux voir la différence entre les deux et pouvoir situer notre projet, dans le tableau ci-dessous nous avons effectué une comparaison selon des caratéristiques essentielles :

• ***Modèle de données***	
Realtime Database	**Firestore**
les données sont stocké sous la forme d'une grande arborescence JSON. Cependant, les données simple sont faciles à stocker contrairement au données complexes et hiérarchiques qui sont plus difficiles à organiser à grande échelle	Les données simples sont faciles à stocker dans des documents, elles sont similaires à JSON et grâce aux sous-collections au sein des documents, les données complexes et hiérarchiques sont plus faciles à organiser à l'échelle.

• *Support en temps réel et hors ligne*	
Realtime Database	**Firestore**
Prise en charge hors ligne pour les clients mobiles sur iOS et Android uniquement.	Prise en charge hors ligne pour les clients iOS, Android et Web.

• *Interrogation*	
Realtime Database	**Firestore**
Requêtes approfondies avec une fonctionnalité de tri et de filtrage limitée(trier ou filtrer une propriété dans une seule requête).	Requêtes indexées avec tri et filtrage composés.

• *Écriture et transactions*	
Realtime Database	**Firestore**
Opérations d'écriture et de transaction de base car les données sont écrite en tant qu'opération individuelle et Les transactions dans les SDK natifs nécessitent un rappel d'achèvement.	Les Opérations d'écriture et de transaction atomiques, car les opérations sont par lots et les compléter de manière atomique et Les transactions se répètent automatiquement jusqu'à ce qu'elles soient terminées.

• *Fiabilité et performance*	
Realtime Database	**Firestore**
Realtime Database est une solution régionale grâce a sa très faible latence et la limitation des bases de données à la disponibilité de zone dans une seule région	Cloud Firestore est une solution multi-régions qui évolue automatiquement.

• *L'évolutivité*	
Realtime Database	**Firestore**
La mise à l'échelle nécessite le sharding.	La mise à l'échelle est automatique.

• *Sécurité*	
Realtime Database	**Firestore**
Se base sur des règles en cascade qui nécessitant une validation séparée, elle sont la seule option de sécurité. La lecture et l'écriture des règles en cascade est obligatoire .la validation se fait séparément avec les règles de validations.	Une sécurité plus simple et plus puissante pour les applications mobiles, Web et SDK de serveur, il n'utilise pas des règles en cascade, et la validation des données se fait automatiquement .

• ***Prix***	
Realtime Database	**Firestore**
Ne facture que la bande passante et le stockage, mais à un taux plus élevé	Une charge sur les opérations effectuer (lecture,...) et un taux inférieur su la bande passante et le stockage.

TABLE 4.2 – Realtime Database vs Firestore

4.2.4 Inconvénients de Firebase

Après avoir vu les avantages des deux bases de données de Firebase nous venons ici parler de leurs inconvénients :

- Les bases de données de Firebase sont disponibles uniquement autant que service Cloud.
- Absence de prise en charge de SQL.
- Certaines formes de traitement de données au format XML, telles que la prise en charge de structures de données XML et/ou la prise en charge de XPath, XQuery ou XSLT ne sont pas supportés.
- N'offrent pas une API pour les méthodes de mappage/réduction définies par l'utilisateur.

Conclusion

À travers ce chapitre, nous avons expliqué comment Apache Hbase et Firebase peuvent être utilisés pour stocker et manipuler des données sémantiques récoltées à partir d'ensembles de données à grande échelle. Nous avons présenté pour chacun le modèle de données utilisé, l'architecture physique et logique ainsi que la façon dont les données sont structurées. Enfin nous avons parlé des points forts et inconvénients de ces deux bases de données.

Deuxième partie

Conception

Chapitre 5

Etude et Conception

Introduction

Face à la croissance de l'activité sismique et aux catastrophes engendrées, le besoin de se protèger s'accroît. Ainsi plusieurs systèmes d'alerte précoce sismique sont aujourd'hui adoptés comme étant un outil efficace pour réduire le risque sismique. De ce fait, plusieurs systèmes sont aujourd'hui opérationnels, en cours de construction ou en projet dans de nombreuses régions du globe : Mexique, Romanie, Californie, Japon, Taïwan, Turquie, Grèce, Italie, Lituanie comme le montre la figure suivante :

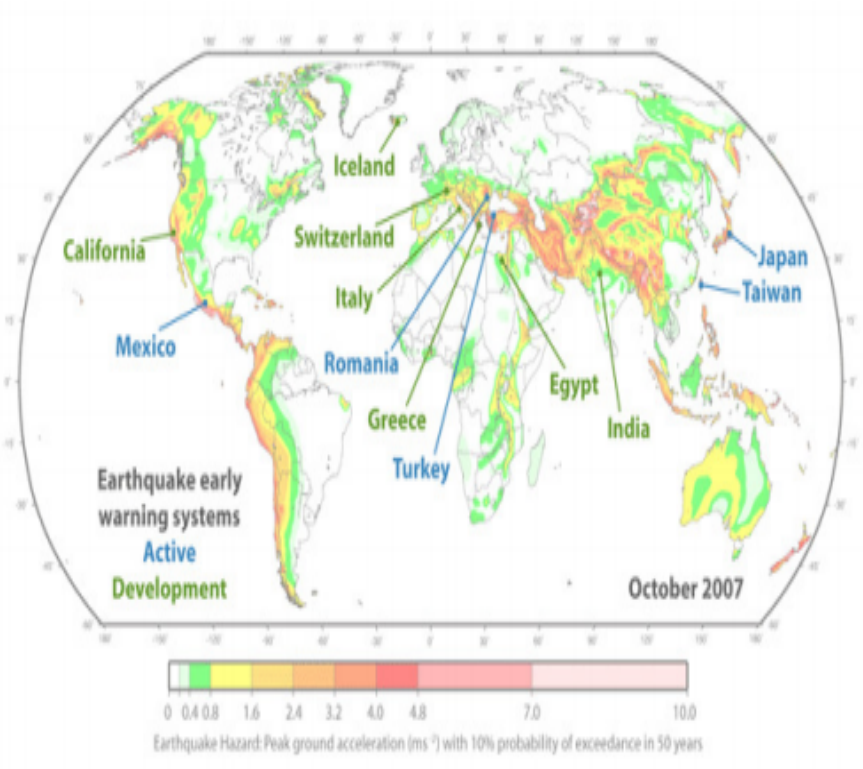

FIGURE 5.1 – Carte globale d'aléa sismique figurant les régions dotées de systèmes d'alerte sismique précoce(bleu), et celles en développement (vert).

Dans cette partie, nous allons procéder à la phase la plus cruciale qui est l'étude du contexte. En effet, nous allons mener une étude préliminaire sur les systèmes d'alerte précoce sismique déja existants. Cette dernière nous permet d'avoir une idée claire et bien précise sur le fonctionnement, les interventions et les actions possibles en cas d'alerte. Enfin nous nous intéresserons à la description des scénarios susceptibles de générer de la valeur et permettant d'anticiper les futurs cas d'utilisation pour pouvoir ainsi structurer toutes les données relatives à notre contexte.

5.1 Concepts de base sur les Smart Cities

Smart city -en français ville intelligente- est un nouveau concept de développement urbain. Il s'agit d'améliorer la qualité de vie des citadins en rendant la ville plus adaptative et efficace, à l'aide de nouvelles technologies qui s'appuient sur un écosystème d'objets et de services. Le périmètre couvrant ce nouveau mode de gestion des villes inclut notamment : infrastructures publiques (bâtiments, mobiliers urbains, domotique, etc.), réseaux (eau, électricité, gaz, télécoms) ; transports (transports publics, routes et voitures intelligentes, covoiturage, mobilités dites douces, etc.) ; les e-services et e-administrations.[1]

L'implémentation de ce concept se confronte à plusieurs problèmes. Néanmoins, grâce au Big Data et à l'Internet des Objets, ce projet utopique pourrait enfin aboutir.

5.1.1 Le rôle du citoyen dans la Smart City

On retrouve dans la notion de smart city l'ensemble des secteurs qui composent la ville d'aujourd'hui. Mais au-delà de la "technique", le citoyen de la smart city a vocation à devenir acteur de sa ville. Il serait désormais inconcevable de penser à la ville de demain sans son citoyen. A travers les réseaux sociaux, les applications, les forums ou autres messageries, le citoyen crée de la donnée, utilise de la donnée, gère à son échelle de la donnée. La smart city devra permettre à chaque citoyen de connaître, d'analyser et d'influer sur les données de son environnement.

Une ville qui se réinvente intelligemment doit commencer par mettre en place tous les outils possibles pour impliquer davantage ses citoyens dans l'administration. Pour que cela soit possible, il faut rendre publiques les données (open data [1]), ce qui permet à ceux qui savent (start up, entreprises) de travailler directement sur les informations mises à leur disposition par les services publics et les entreprises.[2]

Prenons les deux exemples suivant dans notre cas d'étude, *grâce à la géolocalisation et à la détection de la présence humaine, On peut facilement savoir le nombre de personnes présentes dans une zone publique à risque ou dans les grands immeubles.*

1. L'open data ou données ouvertes sont des données numériques dont l'accès et l'usage sont laissés libres aux usagers. Elles peuvent être d'origine publique ou privée, produites notamment par une collectivité, un service public (éventuellement délégué) ou une entreprise.

Tout comme on peut surveiller les réseaux routiers afin de s'informer sur les routes les plus empreintées et les plus exposées aux embouteillages en comptant le nombre de véhicules présents (véhicules munis d'un GPS) et l'implémentation des capteurs sur les routes.

5.1.2 Les caractéristiques d'une Smart City

Pour se transformer en smart city, une ville dispose de multiples domaines où agir. Ainsi chaque ville est différente. Chaque ville doit donc satisfaire des besoins et relever des défis particuliers. Chaque ville développe par conséquent sa propre vision de ville intelligente. Six directions fondamentales d'action lui permettent d'orienter ses objectifs dans cette voie, selon la définition des smart cities de Boyd Cohen [2], un chercheur en développement urbain :[3]

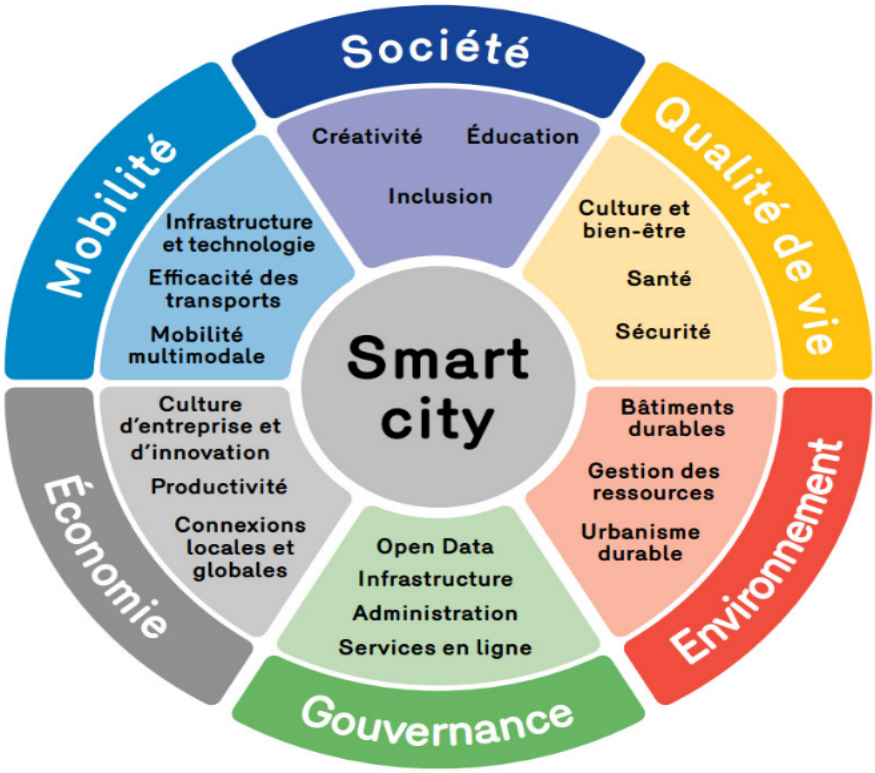

FIGURE 5.2 – Les 6 domaines d'action des Smart Cities.

- **La smart économie :** c'est la ville qui veut se positionner comme une capitale de la nouvelle économie et de l'innovation ainsi que pôle d'attraction.
- **La smart mobilité :** c'est la ville qui s'organise pour offrir une alternative à la congestion et à la pollution automobiles en favorisant l'efficacité des moyens de déplacement collectifs et durables.
- **La smart population :** c'est la ville qui privilégie le développement de ses citoyens en résorbant les inégalités et les poussant à se former.

2. Boyd Cohen est un stratège urbain et climatique travaillant dans le domaine du développement durable et des villes intelligentes.

- **Les smarts conditions de vie :** c'est la ville qui se hisse au meilleur niveau en termes de santé ou de sécurité par exemple.
- **La smart gouvernance :** c'est la ville dont les services publics sont entrés dans l'ère numérique via des services en ligne efficaces, le wifi ou encore l'exploitation des données numériques produites dans la ville.

  L'exemple de l'Open Data : *Les Open Data, ou données ouvertes, sont des données auxquelles l'accès est totalement public et libre de droit, au même titre que l'exploitation et la réutilisation. Ces données offrent de nombreuses opportunités pour étendre le savoir humain et créer de nouveaux produits et services de qualité. Par exemple pour la surveillance du trafic routier on peut publier certaines données pour : l'affichage des temps de parcours, la détection automatique des congestions ou simplement le renseignement de l'état du trafic. Ici, les capteurs sont déployés en grand nombre afin d'obtenir un maillage dense, ils se doivent donc d'être simples, économiques, ne nécessitant qu'une maintenance réduite et associés à un système de transmission de données performant.*

- **Le smart environnement :** c'est la ville qui concilie ses fonctions d'habitat, de mobilité, de pôle économique... tout en réduisant son empreinte sur la planète (consommation réduite d'énergie et de ressources naturelles et réduction des émissions polluantes).

  L'Exemple de l'urbanisme durable : *L'urbanisme durable se veut plus solide face aux séismes en utilisant de nouvelles méthodes de constructions, de nouveaux matériaux, de nouveaux modes de déplacements.*

5.2 Réflexion sur la modélisation d'une Smart City dans le contexte d'un tremblement de terre

Dans le contexte des systèmes d'alerte précoce un tremblement de terre se manifeste sous forme de trois phases à savoir :

- **La phase Pre-sismique**
- **La phase Co-sismique**
- **La phase Post-sismique**

Le schéma suivant donne un bref résumé sur l'enchaînement et un apreçu du rôle de ces phases.

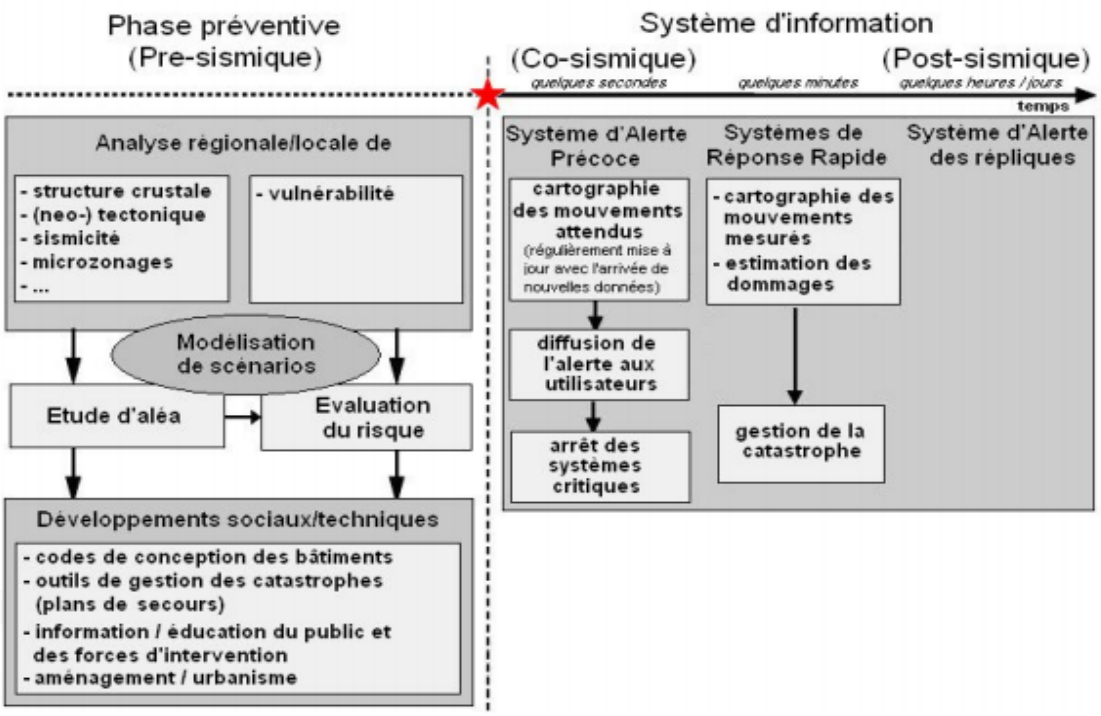

FIGURE 5.3 – Contexte et principes des Systèmes d'Alerte Précoce.

5.2.1 La phase Pre-sismique

La phase Pre-sismique est préventive elle consiste à analyser la région, étudier sa vulnérabilité, sa structure crustale ainsi que d'evaluer les risques.

1. **Présentation de la ville choisie**
   Le choix de notre étude s'est porté essentiellement sur la ville de Tokyo qui est au carrefour de trois plaques tectoniques (la plaque philippine, la plaque eurasienne, et la plaque nord-américaine), ce qui constitue l'une des zones sismiques les plus actives du monde.

2. **Reflexion sur les informations utiles relatives à la ville de Tokyo dans le contexte d'un tremblement de terre**
   Dans ce qui suit nous allons présenter l'infrastructure de la ville de Tokyo, à savoir : le transport, les routes, les immeubles, **les structures étatiques de secours** (les casernes de pompiers, les départements de la sécurité publique, les hopitaux etc...) et de **gouvernace** (les centrales nucleaires, les banques).

FIGURE 5.4 – Emplacement de Tokyo.

(a) **Transport :** Tokyo possède deux compagnies aériennes internationales, un réseau routier. L'une des spécificités du transport japonais est son transport en commun composé de trains, métros, tramways, monorails et bus.

- **Aéroports :** Tokyo dispose de deux aéroports internationaux tout comme de nombreuses grandes métropoles dans le monde, le premier à avoir été construit est « **L'aéroport international Haneda de tokyo** », le second est « **l'aéroport international de Narita**».[4]

- **Métros :** Le réseau métropolitain de Tokyo est exploité par deux grandes sociétés (Tokyo Metro et Toei) gérant en tout **13 lignes** qui totalisent 307 kilomètres pour environ **290 stations**.[5]

- **Trains :** Tokyo tatolise 4 gares de train : la principale est «**La gare de Tokyo**» ; la plus grande et plus fréquentée est celle de «**Shinjuku**» et enfin les deux dernières «**Shibuya et Ikebukuro**».[6]

- **Tramway :** Il existe deux grandes ligne à Tokyo.**la ligne Toden Arakawa** est gérée par la compagnie Toei tandis que **la Ligne Setagaya** est exploitée par la compagnie Tōkyū.[7]

(b) **Les routes :** Tokyo se composent de plusieurs types de routes à savoir :

- **Autoroutes :** Les autoroutes de Tokyo sont en général construites en hauteur pour remédier au manque de place et sont payantes selon un système de tarif fixe.

- **Routes :** La plupart des petites rues dans la capitale n'ont pas de noms. Les adresses sont composées des numéros des pâtés de maisons. Cependant il existe plusieurs routes circulaires et nationales de tailles différentes qui sont numérotées.[8]

(c) **Les habitations :** Tokyo est une ville extrêmement peuplée, ce qui explique la multiplication continue de nombre de résidénces, bâtiments d'habitations, immeubles, écoles, gratte-ciels.

(d) **Les structures gouvernementales de secours :**
Parmis les structures gouvernementales de secours les plus impliquées lors d'un tremblement de terre nous citons :

i. **Les casernes de pompiers :** Le service d' incendie de Tokyo est chargé de la protection de la région métropolitaine de Tokyo . Il est le plus grand service d'incendie urbain au monde. Couvrant les 23 arrondissements de Tokyo et certaines parties de l' ouest de Tokyo, il fournit une assistance en cas d'incendies, de risques biologiques, chimiques et radioactifs, ainsi que de tremblements de terre et inondations.[9]

L'ensemble des stations dans la région métropolitaine de Tokyo est diviser sur tout le territoire de la manière suivant :

- Divisions de casernes de pompiers : 3.
- Casernes de pompiers : 81.
- Branches de casernes de pompiers : 207.

ii. **Les hôpitaux :** L'hôpital a pour ambition de prendre en charge les patients et de contribuer à leur état de santé et ainsi de répondre à toutes les détresses de tous les citoyens.

Exemple : *Dans le cas d'un tremble de terre, le département des urgences est très actif pour les premiers secours que ça soit au sein de l'hôpital ou bien en déplacement vers les zones en détresses, ainsi ils procurent des fournitures médicales supplémentaires, des tentes, des lits, du matériel de secours et des ambulances.*

iii. **Les départements de la sécurité publique :**
Regroupent le département de la police et de la gendarmerie, ils interviennent lors d'un tremblement de terre pour évacuer les zones à risque et porter assistance aux personnes éxposées aux dangers, d'orienter les passants perdus.

(e) **Les sites à risque :**
Parmis les sites à risque dans une ville nous citrons les plus sensibles lors d'un tremblement de terre.

i. **Les centrales nucléaires :**
Une centrale nucléaire est un site industriel destiné à la production d'électricité et dont la chaudière est constituée d'un ou plusieurs réacteurs nucléaires ayant pour source d'énergie un combustible nucléaire. [3]

Par exemple, *Dans le cas d'un tremblement de terre, les centrales nucléaires représentent un danger majeur. En effet, la dispersion de matériaux radioactifs dans l'environnement peut nuire celui-ci et la santé de l'homme.*

ii. **Les banques :**
Une banque est un établissement de crédit qui à pour rôle d'assurer la gestion des moyens de paiement, de dépôt, d'investissement et de financement. De ce fait, elle doit se munir d'un système de haute sécurité pour la protéction des biens du publiques vis-à-vis des effractions.

5.2.2 La phase Co-sismique

Le principe du système d'alerte précoce sismique consiste, lorsque qu'un séisme vient juste de survenir, à fournir une alerte dès les premières secondes de manière à pouvoir mettre en œuvre des procédures permettant de minimiser les dommages. Si, dans l'état actuel des connaissances, la prédiction des séismes n'est pas possible à l'échelle de temps des contingences humaines, le principe d'alerte précoce sismique offre en revanche une alternative intéressante permettant de limiter l'exposition des enjeux à l'aléa sismique.

Au niveau du public, et moyennant une information et des formations adaptées, l'alerte précoce peut être un outil utile permettant d'adopter des mesures protectrices en mesure de réduire significativement les pertes humaines. Au niveau organisationnel, des mesures automatiques de mise en sécurité peuvent également être instaurées, permettant de limiter les dégâts lors de l'arrivée des mouvements forts, ainsi que les suraccidents. Ce système, basé sur l'analyse automatique en temps réel des enregistrements des mouvements sismiques, s'inscrit donc dans la démarche générale de réduction du risque sismique, comme étant le seul outil ≪ co-sismique ≫.[B1]

1. **Réflexion sur le système d'alerte sismique à mettre en place**
Le Japon est parmis les pays les plus exposés au séisme sur la planète, il dispose d'un système d'alerte sismique rapide aux tremblements de terre le plus avancé au monde. Développé par l'Agence Japonaise de Météorologie (JMA) [4] et qui couvre l'ensemble

3. Le combustible nucléaire est le produit qui, contenant des matières fissiles (uranium, plutonium...), fournit l'énergie dans le cœur d'un réacteur nucléaire en entretenant la réaction nucléaire en chaîne de fission nucléaire.

4. JMA est une agence spécialisée dans la météorologie localisée au Japon. En plus de relever les données terrestres et maritimes du temps, elle lance et exploite des satellites météorologiques, ainsi que de la détection des séismes dans la région de l'océan Pacifique ouest en utilisant l'échelle de Shindo d'intensité sismique de 0 à 7, basée sur l'amplitude des accélérations mesurées.

du pays. Ce système, initialement conçu pour un usage restreint à certains utilisateurs (institutions publiques, centrales nucléaires, compagnies ferroviaires), a été étendu en 2007 à toute la population, pour devenir le plus vaste système d'alerte existant au monde. Ce système national en ligne détecte les tremblements et prévoit leur destination, calcule l'épicentre d'un séisme et envoie de brefs avertissements à partir de plus de 2 000 sismographes [5] disséminés dans tout le pays.[B1]

2. **Le fonctionnement de ce système**
   La première preuve d'un tremblement de terre est les deux ondes P (pour primaire) et S (pour secondaire) détectées à quelques secondes par le sismographe le plus près de l'épicentre. les premieres ondes P, ont des longueurs d'onde courtes et rapides et causent peu de dégâts. Celles-ci sont suivies généralement quelques secondes plus tard par les ondes S destructives avec des longueurs d'onde plus longues.[B2]

   La bonne nouvelle est que les ondes P arrivent toujours avant les ondes S afin de pouvoir envoyer des alertes et se préparer quelques secondes avant le tremblement. Le délai d'alerte est alors défini comme le temps qui sépare l'arrivée des ondes P de l'arrivée des ondes S. Ces avertissements sont principalement émis par l'Agence météorologique japonaise (JMA), avec des indications sur la manière de réagir[10]

   Il existe deux niveaux d'alerte en fonction de la puissance du séisme :

   - Dans le cas de séismes de magnitude modérée très fréquents au Japon, émission d'alertes restreintes.
   - Lors de puissants séismes, diffusion d'alertes publiques via les chaînes de télévision et de radio, par internet, ainsi que par des haut-parleurs situés dans les lieux publics.
   - Dans tous les cas, le public peut être averti en s'équipant de récepteurs individuels proposés par des entreprises spécialisées, et qui utilisent les alertes du JMA émises sur internet. **A titre d'exemple**, la société SunShine Co. Ltd propose le système EQGuard, qui procède à un décompte du temps restant avant l'arrivée des ondes S.

3. **Réflexion sur les scenarios possibles en cas d'alerte**
   Le système a été mis au point pour minimiser les dégâts causés par les tremblements de terre et permettre aux personnes de s'abriter ou d'évacuer des zones dangereuses avant l'arrivée de ses fortes vagues de surface.

   - Arrêt de l'alimentation des trains à grande vitesse de manière à les ralentir afin qu'ils n'atteignent pas la zone épicentrale où la ligne est potentiellement endommagée.

5. Un sismographe est un instrument de mesure équipé d'un capteur des mouvements du sol, le sismomètre, capable de les enregistrer sur un support visuel, le sismogramme.Pour obtenir le mouvement tridimensionnel de l'onde sismique, il est nécessaire d'enregistrer trois directions différentes formant un trièdre (en général, une direction verticale, et deux directions horizontales perpendiculaires).

- L'appareil qui detecte peut être connecté aux portes de garage de la caserne de pompiers, ce qui les ouvrira automatiquement.

- Arrêt de la circulation et fermeture de réseaux de transport sensibles.

- Fermeture de réseaux sensibles ***(réseau de distribution du gaz)***.

- Arrêt des systèmes informatiques sensibles et mise en sécurité des données.***(cas des systèmes bancaires)***.

- Alerte des aéroports de manière à éviter tout atterrissage.

- Alerte des centres opérationnels et des centres de secours de manière à permettre une meilleure organisation des secours grâce à l'anticipation.

- Alerte des établissements de santé de manière à suspendre l'activité des blocs opératoires.

- Alerte de la population de manière à évacuer les lieux les plus exposés et se mettre en position de sécurité.

L'efficacité de l'avertissement dépend de la position du récepteur. Après avoir reçu un avertissement, une personne dispose de quelques secondes à une minute ou plus pour agir.

La figure ci-dessous représente quelques actions principales lors d'une alerte d'un tremblement de terre.

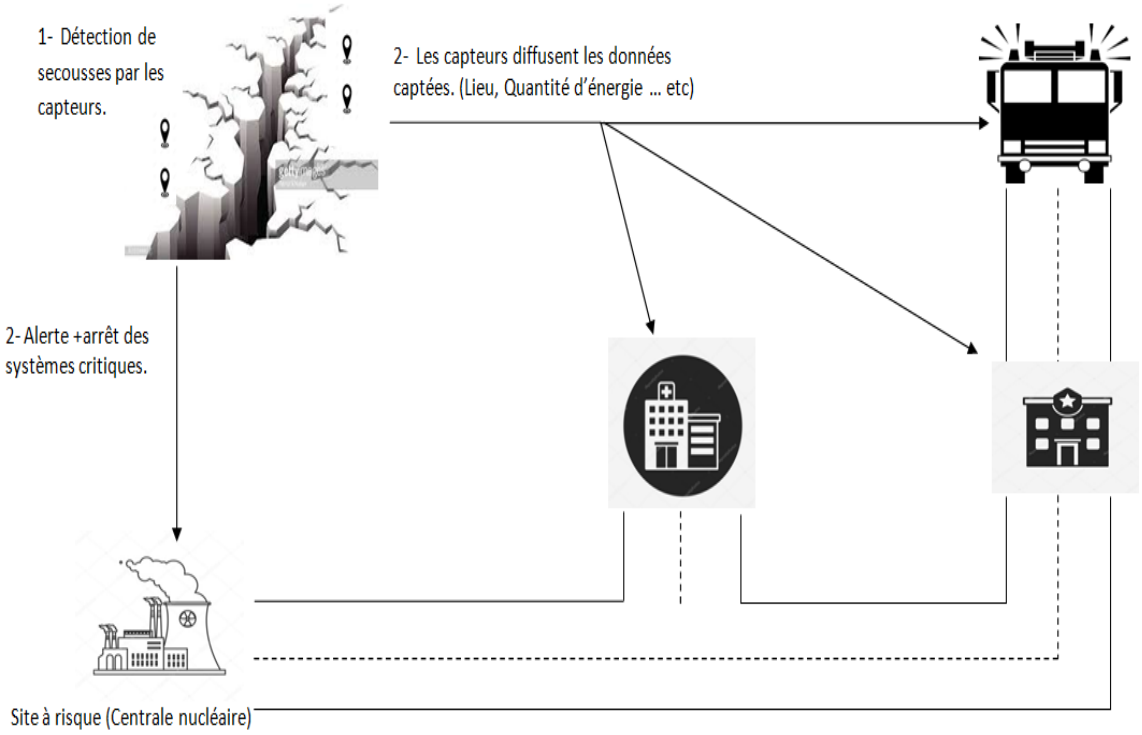

FIGURE 5.5 – Scénario d'avertissement des centrales nucléaires et des structures gouvernementales.

5.2.3 La phase Post-sismique

La phase post-sismique est caractérisée par des répliques et des réajustements viscoélastiques [6]. Pour celà il est important d'appréhendre quelques scénarios pour l'évacuation des zones touchées par le séisme ainsi que la protection des vies humaines en danger que nous présenterons dans ce qui suit :

- **Scénario d'évacuation des habitations.**
- **Scénario de secours pour une effraction dans une banque.**
- **Scénario d'évacuation des sites à risque.**

6. La viscoélasticité est la propriété de matériaux qui présentent des caractéristiques à la fois visqueuses et élastiques, lorsqu'ils subissent une déformation.

1. **Scénario d'évacuation des habitations :**

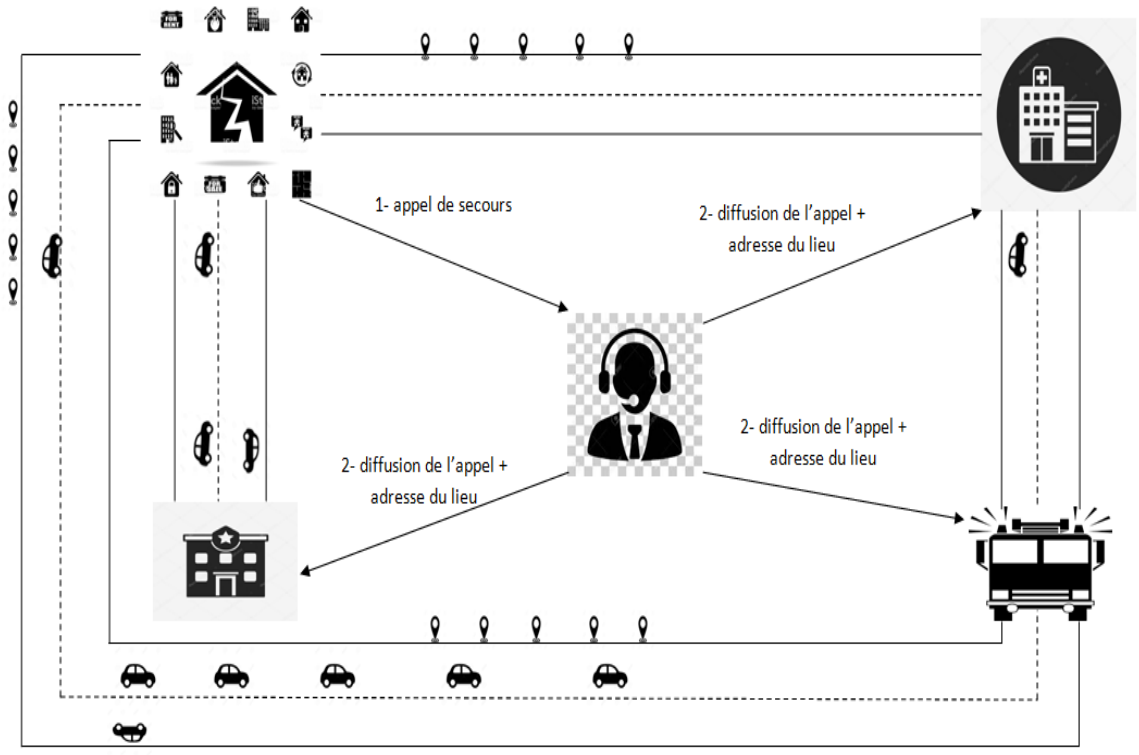

FIGURE 5.6 – Scénario d'évacuation des habitations.

2. **Scénario de secours pour une effraction dans une banque :**

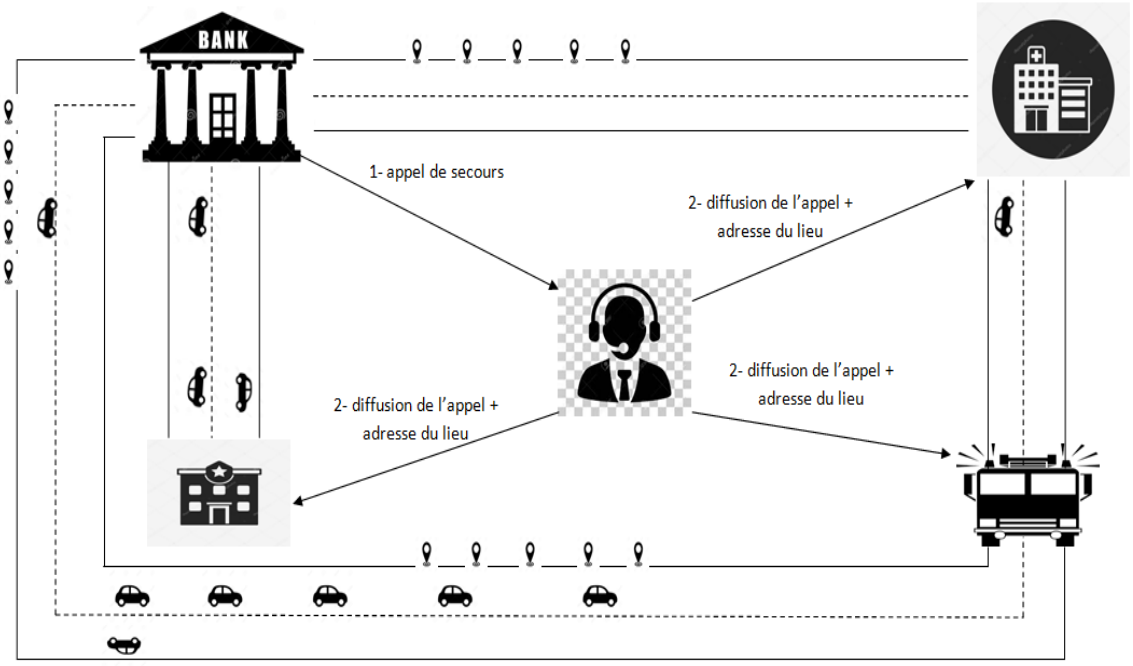

FIGURE 5.7 – Scénario de secours pour une effraction dans une banque.

3. **Scénario d'évacuation des sites à risque :**

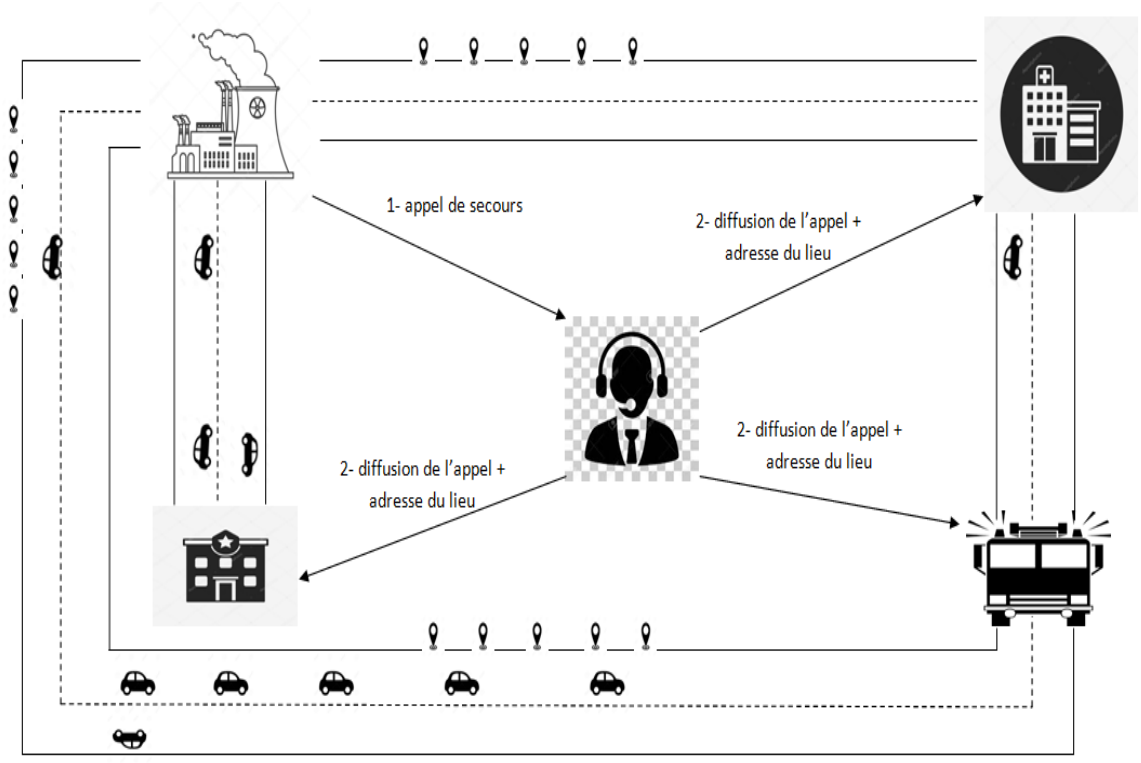

FIGURE 5.8 – Scénario d'évacuation des zones à risque(cas d'une centrale nucléaire).

5.3 La Collecte des données relatives à ce système

Après avoir délimiter notre projet selon nos besoins et étudier le contexte ainsi que la description des scénarios susceptibles de générer de la valeur et permettant d'anticiper les futurs cas d'utilisation. Nous venons ici collecter ces données relatives à cette ville.

Par collecte de données, on entend l'approche systématique qui consiste à réunir et à mesurer des informations en provenance de sources variées, afin d'obtenir une vue complète et précise d'un domaine d'intérêt. La collecte des données permet à une personne ou à une entreprise de répondre à des questions pertinentes, d'évaluer des résultats et de mieux anticiper les probabilités et les tendances à venir[11]. Dans le cadre de notre travail on va classer ces données en deux catégories à savoir :

- **Les données fixes :** Une donnée fixe est en général immuable c'est-à-dire son état ne peut être modifier au fil du temps.

- **Les données évolutives :** On dit qu'une donnée est évolutive si dans un même contexte, sa valeur change au fil du temps.

Selon notre contexte à étudier, Les données sur lesquelles nous informe cette ville sont regroupées dans le tableau suivant :

Catégorie		**Caractéristiques**	
		Données fixes	**Données évolutives**
Routes	Autoroutes et Routes circulaires	• Code route.	• Etat de la route. • Localisation
Habitations	Immeubles (Ecoles, immeubles de bureaux, Maisons...)	• Code habitation. • N° Tél habitation. • Adresse habitation.	
Sites à risque	Centrales nucléaires	• Code centrale nucléaire. • N° Tél centrale nucléaire. • Adresse centrale nucléaire.	
	Banques	• Code banque. • N° Tél banque. • Adresse banque.	

Structures de secours	Hôpitaux	• Code hôpital. • N° Tél hôpital. • Adresse hôpital.	
	SAMU	• Code SAMU. • N° Tél SAMU. • adresse SAMU.	
	Casernes de pompiers	• Code caserne. • N° Tél. • Adresse caserne.	
Structures gouvernementales	Centre de gendarmerie	• Code gendarmerie. • N° Tél gendarmerie. • Adresse centre de gendarmerie.	
	Poste de police	• Code poste. • N° Tél police. • Adresse du poste.	

Transport	Aéroports	• code AITA et AOCI. • N° Tél. • Adresse aéroport.	
	Station de métros	• Code stationM. • Nombres de lignes • N° Tél • Adresse stationM.	
	Station de tramway	• Code stationT. • Nombre de lignes • N° Tél. • Adresse stationT	
	Gare de trains	• Code gare. • Nombre de lignes • N° Tél. • Adresse gare.	
Capteurs séisme		• Code capteurS. • Localisation.	• Quantité d'énergie du séisme.
Capteurs route		• Code capteurR. • Localisation.	• Etat de la route.

TABLE 5.1 – Exemple de données collectées.

5.4 Structuration des données récoltées pour Hbase et Firebase

1. **Structruration pour Hbase**
   Le modèle utilisé pour structurer nos données pour HBase est le modèle orientée colonnes.

Famille de colonnes			
Clé	Valeur		
CodeCaptS	Colonne		
	Nom	Valeur	
	CodeZone		
	Localisation		
CodeCaptR	Super colonne		
	Clé	Valeur	
	CodeRoute	Colonne	
		Nom	Valeur
		CodeHab	
		AdresseHab	
		NumTelHab	
		Colonne	
		Nom	Valeur
		CodeSR	
		AdresseSR	
		NumTelSR	
		Colonne	
		Nom	Valeur
		CodeSG	
		AdresseSG	
		NumTelSG	
		Colonne	
		Nom	Valeur
		CodeUS	
		AdresseUS	
		NumTelUS	

FIGURE 5.9 – Structuration des données sous HBase.

2. **Structruration pour Firebase** Le modèle utilisé pour structurer nos données pour Cloud Firestore est le modèle orientée documents.

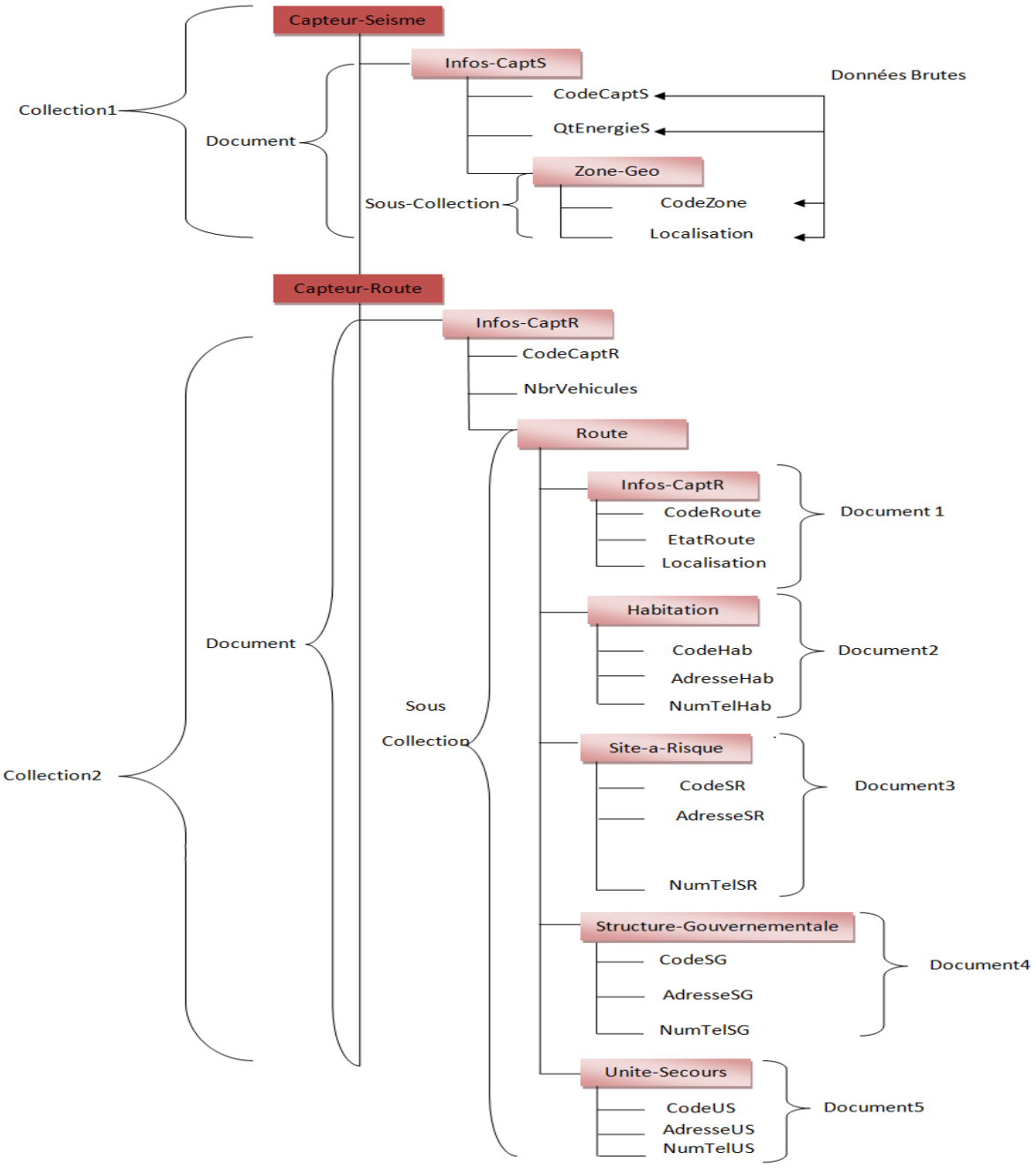

FIGURE 5.10 – Structuration des données sous Firebase.

5.5 Architecture applicative du système à implémenté

Dans le but d'exploiter les données récoltées, les scénarios imaginés vont être implémenté sous forme de modules :

1. Module de récolte des données capteurs.
2. Modules de structuration pour les scénarios d'évacuation.

1. **Module de récolte des données capteurs**

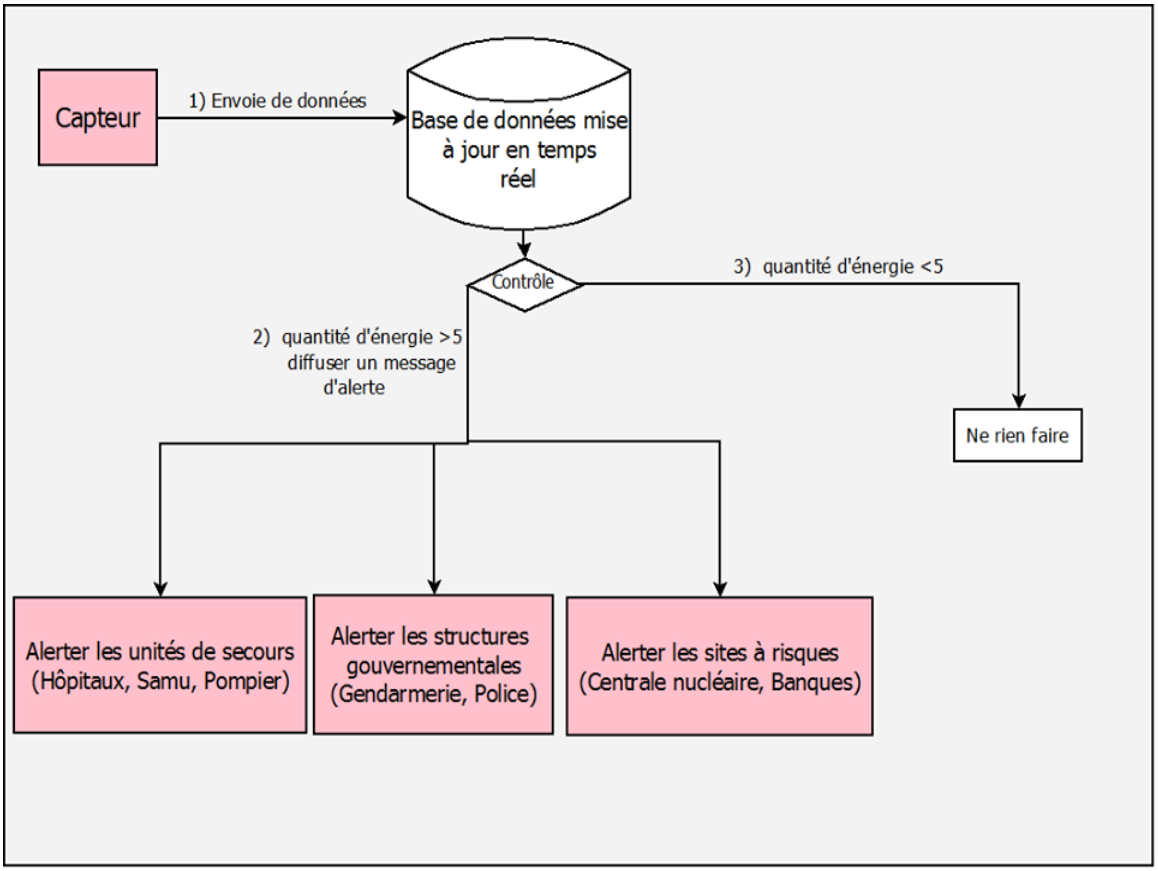

FIGURE 5.11 – Récolte des données via capteur.

2. **Modules de structuration pour les scénarios d'évacuation**
   Ici nous venons modéliser les 3 scénarios d'évacuation des habitations, des sites à risque proposés précédemment :

   (a) **Modélisation pour l'évacuation des habitations**

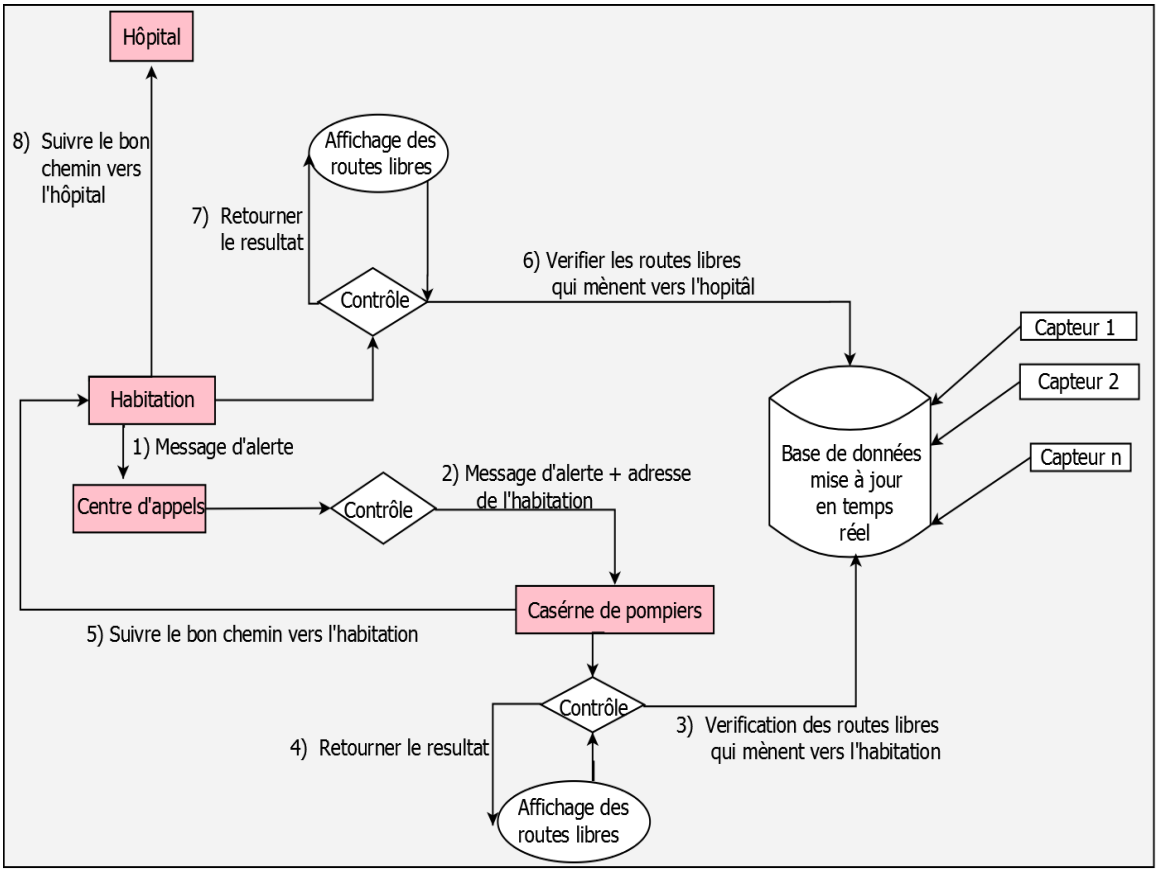

FIGURE 5.12 – Evacuation des habitations.

(b) **Modélisation pour l'évacuation des centrales nucléaires**

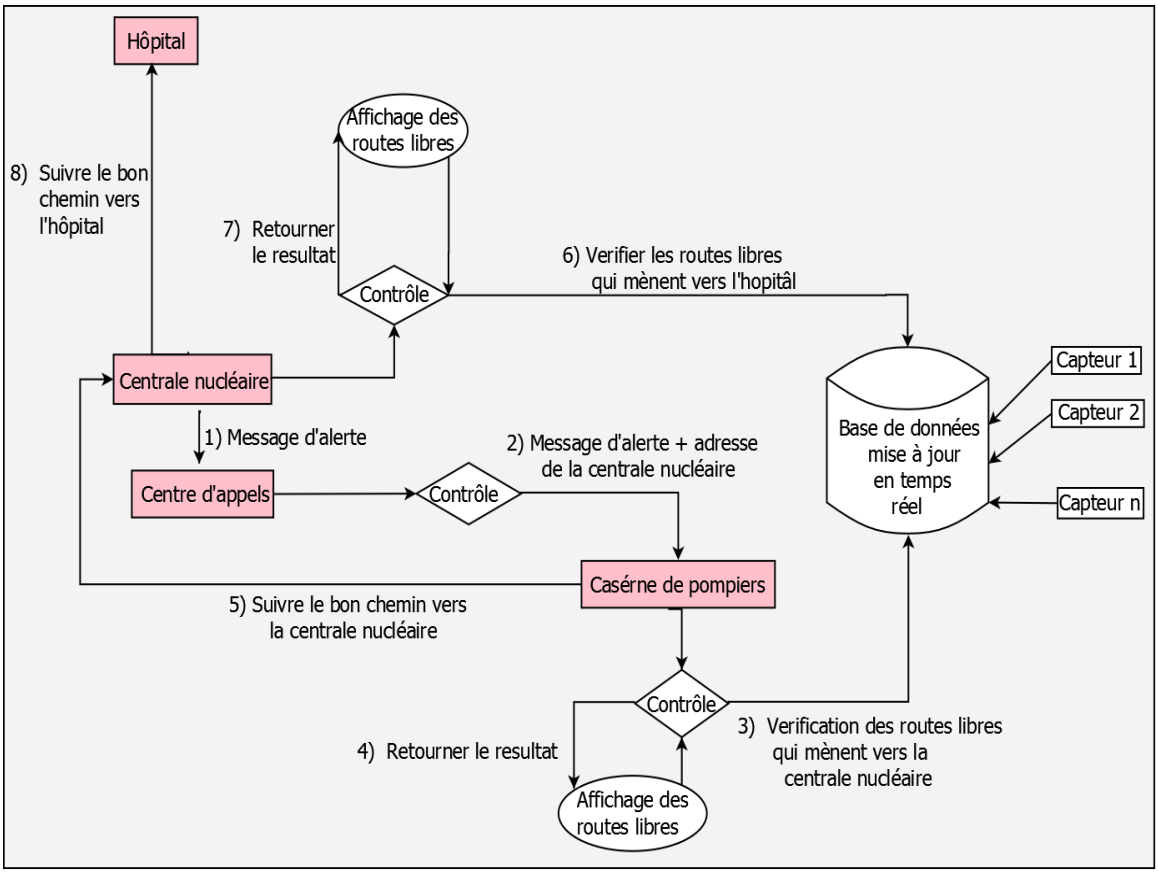

FIGURE 5.13 – Evacuation des centrales nucléaires.

(c) **Modélisation pour l'évacuation des banques**

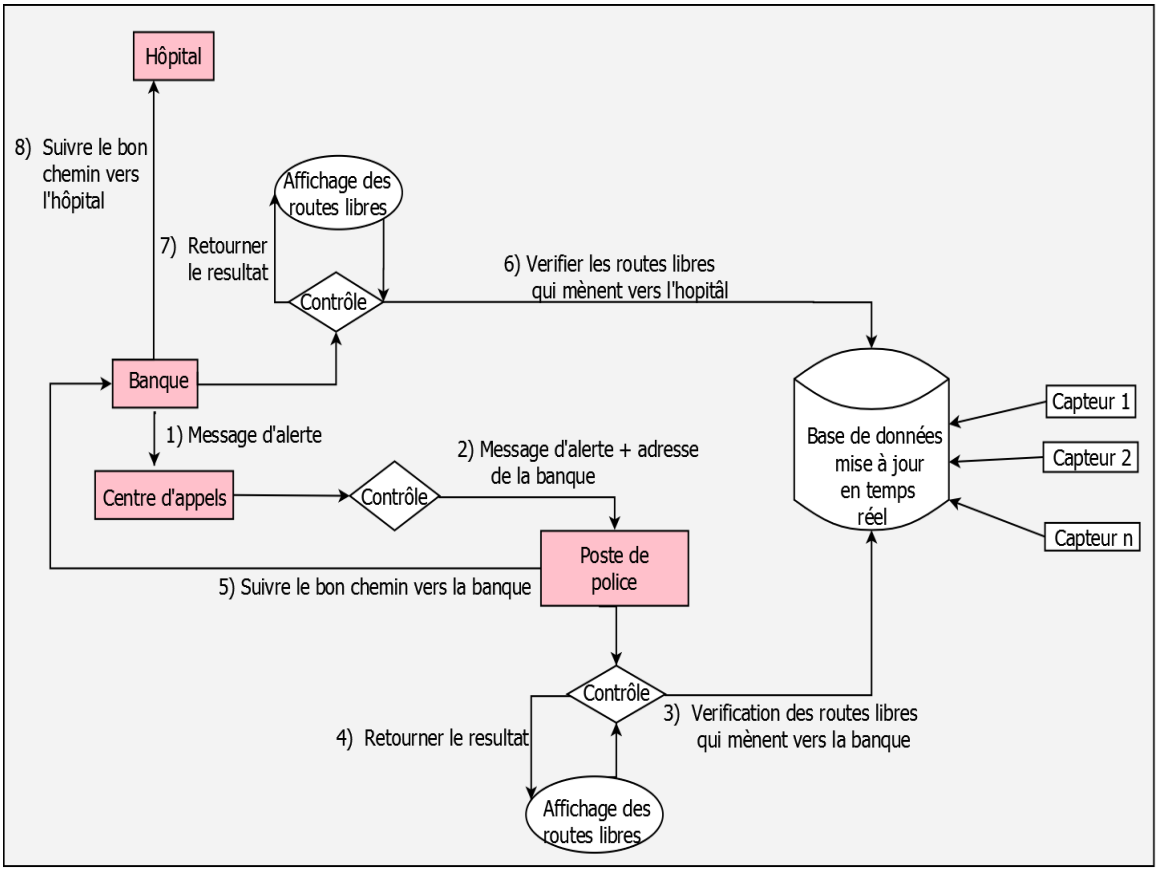

FIGURE 5.14 – Evacuation des banques.

Nous présenterons au final une architecture globale de notre système comme le montre la figure suivante :

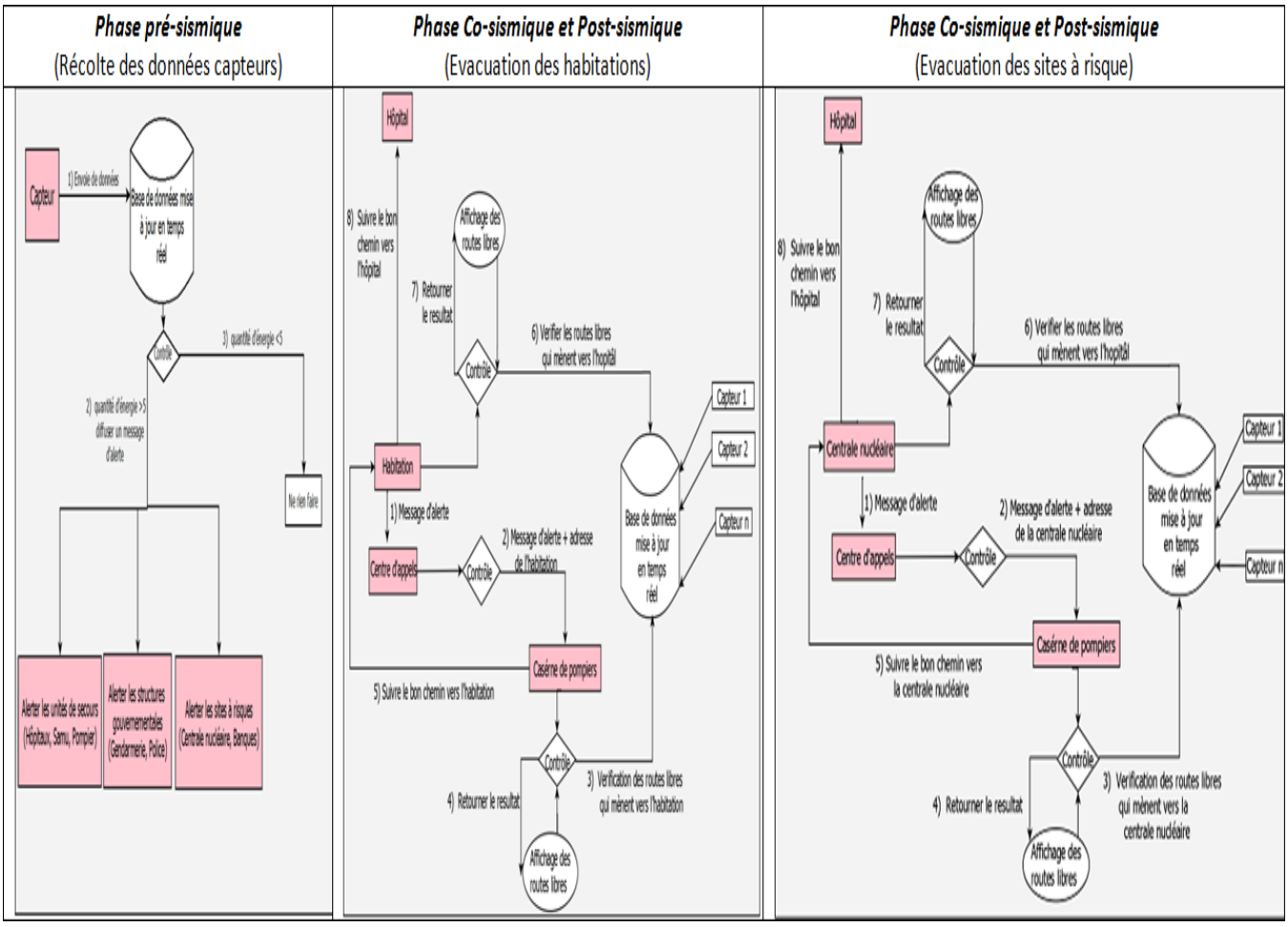

FIGURE 5.15 – Architecture globale du système à implémenté.

- Dans un but d'analyse et de comparaison (SQL vs NoSQL), il nous a été demandé de réaliser le modèle E-A relative à notre cas d'étude que nous présenterons comme suit :

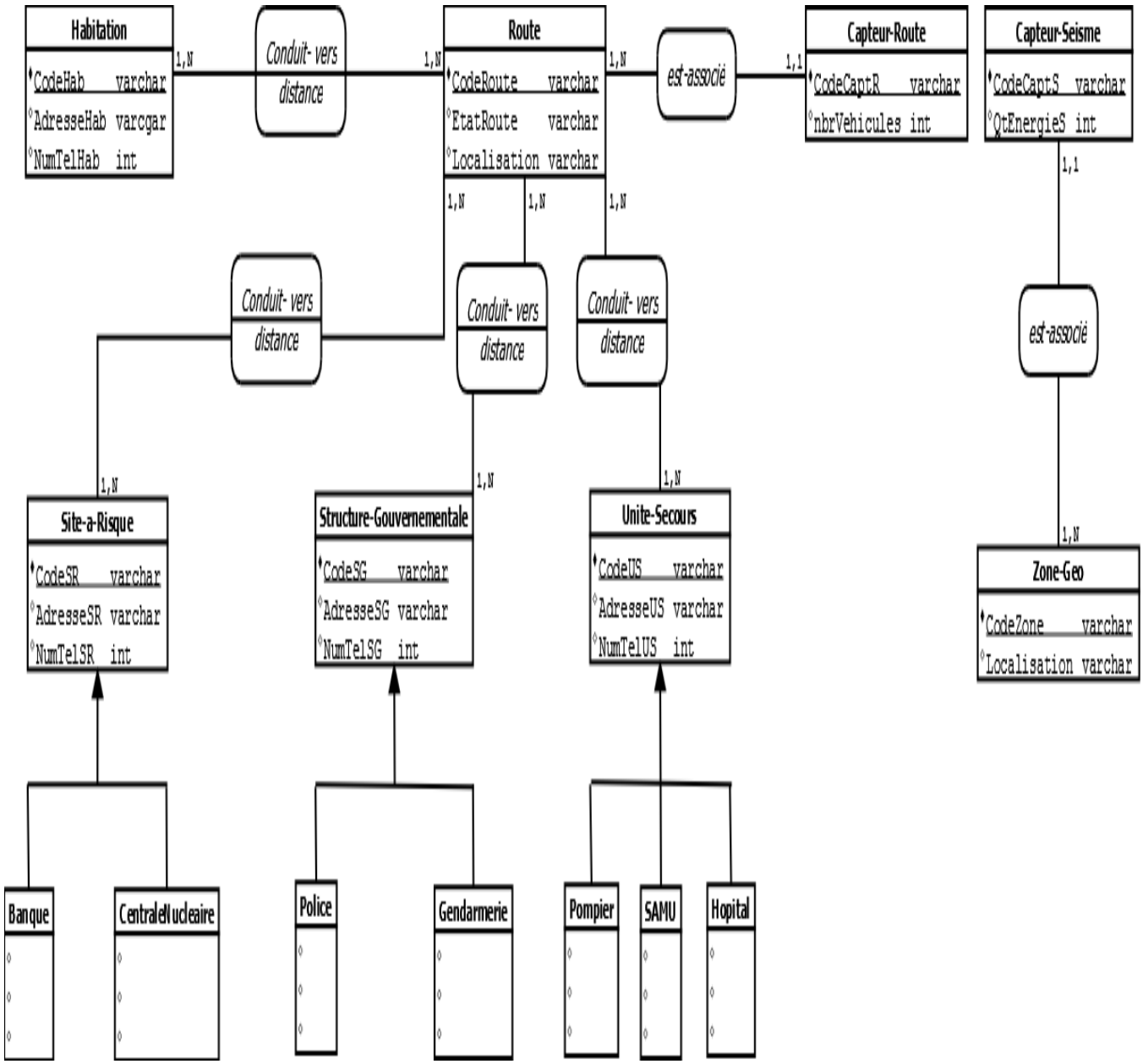

FIGURE 5.16 – Le modèle Entité-Association.

- **Le modèle relationnel :** Le modèle relationnel est une manière de représenter les relations existantes entre plusieurs informations, et de les ordonner entre elles.

  A partir du modèle E-A précédent on peut faire un passage vers le modèle relationnel en respectant les règles de passage et on obtient le modèle relationnel suivant :

Route (CodeRoute, EtatRoute, Localisation) ;

Capteur-Route (CodeCaptR, NbrVehicules, CodeRoute) ;

Habitation (CodeHab, AdresseHab, NumTelHab) ;

Rconduit-versH (CodeRoute, CodeHab, distance) ;

Banque (CodeSR, AdresseSR, NumTelSR) ;

Centrale-Nucleaire (CodeSR, AdresseSR, NumTelSR) ;

Rconduit-versSR (CodeRoute, CodeSR, distance) ;

Police (CodeSG, AdresseSG, NumTelSG) ;

Gendarmerie (CodeSG, AdresseSG, NumTelSG) ;

Rconduit-versSG (CodeRoute, CodeSG , distance) ;

Pompier (CodeUS, AdresseUS, NumTelUS) ;

SAMU (CodeUS, AdresseUS, NumTelUS) ;

Hopital (CodeUS, AdresseUS, NumTelUS) ;

Rconduit-versUS (CodeRoute, CodeUS , distance) ;

Zone-Geo (CodeZone, localisation) ;

Capteur-Seisme (CodeCaptS, QtEnergieS, CodeZone) ;

FIGURE 5.17 – Le modèle relationnel.

Conclusion

Dans un premier temps nous avons abordé le nouveau concept de **"Smart Cities"** et ses différents domaines d'action où nous avons mis l'accent plus particulièrement sur les deux domaines de **gouvernance** et d'**environnement** en relation avec notre contexte. Ensuite nous avons commencé par une présentation relative aux systèmes précoce de détection de tremblement de terre dans le but de mieux comprendre le principe de fonctionnement ainsi que de connaitre les differentes technologies auxquels ces systèmes font recours. Enfin nous avons élaboré des scénarios qui nous ont aidés à avoir une vision partielle sur les réactions envisageables avant et après une alerte sismique tout en captant les informations essentielles qui ont fait le sujet du dernier point de ce chapitre où nous avons procédé à la structuration de ces informations tout en réspectant les régles imposées par nos trois bases de données à savoir La base de données **SQL** et les deux bases de données NoSQL : **Hbase** et **Firebase**.

Troisième partie

Réalisation

Chapitre 6

Choix des outils technologiques

6.1 Technologies utilisées

Pour la mise en œuvre de notre travail nous avons utilisé les technologies : **Hadoop**, **Hbase** et **Firebase** comme SGBD pour nos bases de données NoSQL et le SGBD MySQL pour **la base de données relationnelles**. Tout au long de cette partie nous présenterons les étapes d'installation de ces technologies ainsi que les outils de développement utilisés pour développer un simulateur afin de reproduire les 4 scénarios qui reprennent la situation de secours avant, pendant et après un tremblement de terre.

6.1.1 Hadoop

Hadoop est un framwork libre et open source écrit en Java destiné à la création d'applications distribuées et scalables. Hadoop a été inspiré par les publications de MapReduce et GoogleFS de Google et fait partie des projets de la fondation Apache depuis 2009. Hadoop est le framework Big Data le plus largement utilisé dans le monde. Il regroupe un ensemble d'outils répondant à un très grand nombre de cas d'usages tels que le stockage et la gestion des données, la sécurité, l'analyse, etc.[1]

Mais comme nous l'avons déja mentionné, Dans notre cas Hadoop n'est qu'un outil utilisé afin de parachever notre travail sous Hbase. En effet Hbase est basé sur la même architecture physique qu'Hadoop puisque ce dernier gère le stockage physique dans Hbase.

6.1.2 Hbase

HBase est un système de stockage efficace pour des données très volumineuses. Il permet d'accéder aux données très rapidement même quand elles sont gigantesques. Les données sont stockées dans des tables sous forme de colonnes identifiées par une clé. A chaque fois que la table atteint une certaine limite, elle se fait découpé en deux régions au milieu des clés. ce processus s'applique pour les régions quand elles sont trop volumineuses. Chaque région est géré par un serveur de région qui à son tour peut s'occuper de plusieurs région d'une même table.[B1]

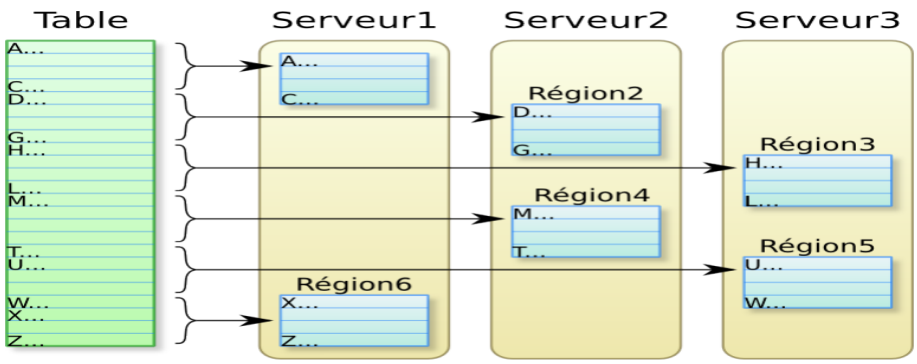

FIGURE 6.1 – Table, région et serveur de région

6.1.3 Firebase

Firebase est une plateforme de développement d'applications pour le web ou pour mobiles. Elle fournit des outils sous forme de services pour la création d'applications mobiles. Comme elle propose deux types de bases de données, Raltime Database pour stocker et synchroniser les données utilisateurs en temps réel sous forme d'arborescence JSON, et Cloud firestore permettant de stocker les données dans des documents (un ensemble de paires clé-valeur) et une collection (une collection de documents).[2]

6.1.4 MySQL

MySQL est une base de données relationnelle libre qui a vu le jour en 1995 et très employée sur le Web, souvent en association avec PHP (langage) et Apache (serveur web). MySql fonctionne indifféremment sur tous les systèmes d'exploitation (Windows, Linux, Mac OS notamment). Son principe est d'enregistrer les informations dans des tables, qui représentent des regroupements de données par sujets (table des routes, table des capteurs, table des habitations, par exemple). Les tables sont reliées entre elles par des relations(clé étrangère). Le langage SQL (acronyme de Structured Query Language) est un langage universellement reconnu par MySql et les autres bases de données et permettant d'interroger et de modifier le contenu d'une base de données. Les autres bases de données utilisées en informatique sont essentiellement Microsoft SQL Server et Oracle.[3]

6.1.5 Google Cloud Platforme

Google Cloud Platform est une plateforme de cloud computing fournie par Google, proposant un hébergement sur la même infrastructure que celle que Google utilise en interne pour des produits tels que son moteur de recherche. Cloud Platform fournit aux développeurs des produits permettant de construire une gamme de programmes allant de simples sites web à des applications complexes.

Google Cloud Platform fait partie d'un ensemble de solutions pour les entreprises appelé Google Cloud, et fournit des services modulaires basés sur le cloud, tels que le stockage d'informations, le calcul, des applications de traduction et de prévision, etc.[4]

6.1.6 Cloudera

Cloudera est une start-up de la Silicon Valley, qui se consacre au développement de logiciels de type Big Data basées sur le framework Hadoop. La firme Cloudera se consacre au développement de logiciels fondés sur Apache Hadoop, permettant l'exploitation de Big Data, à savoir des bases de données accumulant plusieurs pétaoctets. En effet, elle contient les principaux éléments, de base du framework Hadoop (MapReduce et HDFS), ainsi que d'autres composants orientés vers les entreprises qui assurent la sécurité, la haute disponibilité, et l'intégration avec le matériel et les autres logiciels (HDFS, MapReduce, Impala, Apache Spark, HBase, Apache Kafka).[5]

6.2 Installation des technologies utilisées

Pour **Hadoop** et **Hbase** nous avons choisi de les faire tourner sur le système d'exploitation **Linux** distribution **Ubuntu (version 18.04 64 bits)**. En ce qui concerne **Firebase** aucune installation n'est requise tout ce fait sur la plateforme Firebase.

6.2.1 Installation d'Hadoop

Apache Hadoop peut être installé dans différents modes selon l'exigence. Ces différents modes sont configurés lors de l'installation. Nous avons :

- Le mode local (mode Standalone).
- Le mode pseudo-distribué.
- Le mode totalement distribué.

Pour notre travail, le mode totalement distribué est le plus recommandé afin d'avoir des résultats meilleurs et en un temps oppurtun. Pour des raisons d'indisponibilité de machines performantes (Plusieurs machines nœud de 16GB de RAM minimum) nous avons fait recours au mode pseudo-distribué.

Pour ce faire, les instructions suivantes montrent les étapes d'installation :

1. **Installation de Java et SSH :**

   Avant toutes installations de nouveaux paquets, on fait d'abord une mise à jour pour le cache des paquets. La commande suivante téléchargera la nouvelle liste des paquets proposés par le dépôt :

   ```
   $ sudo apt-get update
   ```

   - Installation de Java :

     ```
     $ sudo apt-get install openjdk-8-jdk
     ```

   - Installation de SSH serveur : Hadoop nécessite un accès SSH pour gérer les différents nœuds. Bien que nous soyons dans une configuration pseudo distribué, nous avons besoin de configurer l'accès vers localhost.

     ```
     $ sudo apt-get install openssh-server
     ```

   - Génération d'une paire de clés publique et privée SSH :

     ```
     $ ssh-keygen -t rsa -P ""
     ```

   - Ajout de la clé publique aux authorised_keys :

     ```
     $ chat $ HOME /. ssh / id_rsa. pub >> $ HOME /. ssh / registered_keys
     ```

2. **Installation de Hadoop :**
   Nous avons commencer par télécharger Hadoop depuis l'archive officiel d'apache : https ://archive.apache.org/dist/hadoop/core , en choisissant la version 2.7.1

   Après la fin du téléchargement on accède au répertoire où le fichier est sauvegardé, on l'extrait puis on le déplace vers l'emplacement /usr/local/hadoop , grâce aux commandes suivantes :

   ```
   $ cd Téléchargements
   $ tar xzf hadoop-2.7.1.tar.gz
   $ sudo mv hadoop-2.7.1 /usr/local/hadoop
   ```

   Ici :
   -x est pour extraire le drapeau.
   -z est pour décompresser le fichier.
   -f spécifie l'extraction du fichier.

3. **Configuration d'Hadoop :**

   Cette étape consiste à configurer un fichier **.bashrc** pour ajouter le chemin Hadoop et Java, puis cinq autres fichiers relatifs à Hadoop seront configurés.

   - **Modification du fichier .bashrc :**

     Ouvrir le fichier puis ajouter ces lignes à la fin du fichier :

```
$ sudo gedit .bashrc
```

     Les lignes à rajouter :

```
# chemin d'hadoop

    export HADOOP_HOME=/usr/local/hadoop
    export HADOOP_CONF_DIR=/usr/local/hadoop/etc/hadoop
    export PATH=$PATH:$HADOOP_HOME/bin
    export PATH=$PATH:$HADOOP_HOME/sbin
    export HADOOP_MAPRED_HOME=$HADOOP_HOME
    export HADOOP_COMMON_HOME=$HADOOP_HOME
    export HADOOP_HDFS_HOME=$HADOOP_HOME
    export YARN_HOME=$HADOOP_HOME
    export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
    export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

#chemin java

export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_201
export PATH=/usr/lib/jvm/jdk1.8.0_201/bin:$PATH
```

   - **Modification du fichier hadoop-env.sh :**
     Tout d'abord il faut changer de répertoire puis ouvrir le fichier afin d'ajouter l'emplacement de java :

```
$ cd /usr/local/hadoop/etc/hadoop
$ sudo gedit hadoop-env.sh
```

     Les lignes à ajouter dans ce fichier :

```
#export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_201
```

   - **Modification du fichier core-site.xml :**
     Pour le faire il suffit d'ouvrir le fichier et d'ajouter les lignes ci-dessous :

```
projet@nassima-Inspiron-3558:/usr/local/hadoop/etc/hadoop$ sudo gedit core-site.xml
```

Les lignes à ajouter dans ce fichier :

```
<configuration>

    <property>
        <name>fs.default.name</name>
        <value>hdfs://localhost:9000</value>
    </property>

</configuration>
```

- **Modification du fichier hdfs-site.xml :**
  Ouvrir le fichier puis ajouter les lignes suivantes :

```
projet@nassima-Inspiron-3558:/usr/local/hadoop/etc/hadoop$ sudo gedit hdfs-site.xml
```

Les lignes à ajouter dans ce fichier :

```
<configuration>
    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>

    <property>
        <name>dfs.namenode.name.dir</name>
        <value>file:/usr/local/hdfsf/namenode</value>
    </property>

    <property>
        <name>dfs.datanode.data.dir</name>
        <value>file:/usr/local/hdfsf/datanode</value>
    </property>
</configuration>
```

- **Modification du fichier yarn-site.xml :**
  Ouvrir le fichier puis ajouter les lignes suivantes :

```
projet@nassima-Inspiron-3558:/usr/local/hadoop/etc/hadoop$ sudo gedit yarn-site.xml
```

Les lignes à ajouter dans ce fichier :

```
<configuration>

   <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>

    <property>
        <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
        <value>org.apache.hadoop.mapred.ShuffleHandler</value>
    </property>

</configuration>
```

- **Modification du fichier mapred-site.xml :**
  Il faut d'abord copier le fichier mapred-site.xml.template dans le fichier mapred-site.xml, puis ajouter les lignes suivantes :

```
projet@nassima-Inspiron-3558:/usr/local/hadoop/etc/hadoop$ sudo cp mapred-site.xml.template mapred-site.xml
```

Ouvrir le fichier et ajouter les lignes suivantes :

```
projet@nassima-Inspiron-3558:/usr/local/hadoop/etc/hadoop$ sudo gedit mapred-site.xml
```

```
<configuration>

    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

</configuration>
```

4. **Formater le namenode :** La commande ci-dessous permet de formater le namenode pour mettre en forme les méta-données liées aux nœuds de données.

```
projet@nassima-Inspiron-3558:~$ hdfs namenode -format
```

Le résultat retourné par cette commande est :

```
projet@nassima-Inspiron-3558:~$ hdfs namenode -format
19/07/21 18:50:03 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = nassima-Inspiron-3558/127.0.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.7.1
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop:/usr/local/hadoop/share/
hadoop/common/lib/commons-io-2.4.jar:/usr/local/hadoop/share/hadoop/common/lib/a
ctivation-1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/avro-1.7.4.jar:/usr/
local/hadoop/share/hadoop/common/lib/jaxb-impl-2.2.3-1.jar:/usr/local/hadoop/sha
re/hadoop/common/lib/jets3t-0.9.0.jar:/usr/local/hadoop/share/hadoop/common/lib/
httpclient-4.2.5.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-xc-1.9.13
.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-beanutils-core-1.8.0.jar:
19/07/21 18:50:19 INFO namenode.FSImage: Allocated new BlockPoolId: BP-821135479
-127.0.1.1-1563735018955
19/07/21 18:50:19 INFO common.Storage: Storage directory /usr/local/hdfsf/nameno
de has been successfully formatted.
19/07/21 18:50:19 INFO namenode.NNStorageRetentionManager: Going to retain 1 ima
ges with txid >= 0
19/07/21 18:50:19 INFO util.ExitUtil: Exiting with status 0
19/07/21 18:50:19 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at nassima-Inspiron-3558/127.0.1.1
************************************************************/
```

5. **Lancer et arrêter Hadoop :**
   Pour démarrer Hadoop, il faut démarrer le système de fichiers HDFS et le serveur MapReduce afin d'utiliser des jobs MapReduce.

```
$ start-dfs.sh
                    ou bien        $start-all.sh
$ start-yarn.sh
```

Pour s'assurer que tout fonctionne, on utilise l'outil jps pour lister les processus Java en cours d'exécution :

```
projet@nassima-Inspiron-3558:~$  jps
3347 NameNode
4164 NodeManager
4484 Jps
3766 SecondaryNameNode
3996 ResourceManager
projet@nassima-Inspiron-3558:~$
```

Enfin, pour arrêter un serveur Hadoop, on utilise la première commande pour interrompre le serveur de fichiers HDFS et la deuxième pour interrompre le serveur MapReduce ou bien arrêter tout à la fois avec la dernière commande.

```
$ stop-dfs.sh
                    ou bien     $ stop-all.sh
$ stop-yarn.sh
```

6.2.2 Installation de Hbase

Pour l'installation de Hbase sous Ubuntu, il est très important de vérifier si Java et Hadoop ont été bien installés sur le système.

1. **Téléchargement de Hbase :**
   Nous avons commencer par télécharger Hbase à partir dee son archive officiel : https ://archive.apache.org/dist/hbase/stable/ et choisir la version 1.4.9 .

2. **Configuration de Hbase :**
   Cette étape consiste à faire des modifications sur les fichiers suivant :

   - **.bashrc** pour ajouter le chemin de l'emplacement Hbase.
   - **hbase-env.sh** pour ajouter le chemin de Java.
   - **hbase-site.xml** pour spécifier le répertoire sur le système de fichiers local où HBase et ZooKeeper écrivent des données.

   - **Modifier le fichier .bashrc :**
     Il suffit d'ouvrir le terminal et de taper la commande suivante :

     ```
     $ sudo gedit .bashrc
     ```

     Puis ajouter les lignes suivantes :

     ```
     #chemin hbase

     export HBASE_HOME=/usr/local/hbase
     export PATH=$PATH:/usr/local/hbase/bin
     ```

   - **Modifier le fichier hbase-env.sh :**
     Pour accéder au fichier, il faut changer de répertoire en tapant les commandes suivantes dans le terminal :

     ```
     $ sudo gedit hbase-env.sh
     ```

     Puis ajouter les lignes suivantes :

     ```
     export JAVA_HOME=/usr/lib/jvm/jdk1.8.0_201

     export HBASE_MANAGES_ZK=true
     ```

   - **Modifier le fichier hbase-site.xml :**
     On accède au fichier grâce à la commande :

     ```
     projet@nassima-Inspiron-3558:/usr/local/hbase/conf$ sudo gedit hbase-site.xml
     ```

Puis ajouter les lignes suivantes :

```
<configuration>

       //Here you have to set the path where you want HBase to store its files.
        <property>
           <name>hbase.rootdir</name>
           <value>hdfs://localhost:9000/hbase</value>
        </property>

       //Here you have to set the path where you want HBase to store its built in zookeeper  files.
        <property>
           <name>hbase.zookeeper.property.dataDir</name>
           <value>/home/hbasezk/zookeeper</value>
        </property>
        <property>
            <name>hbase.cluster.distributed</name>
            <value>true</value>
        </property>

</configuration>
```

3. **Accéder au shell de Hbase :**
   Hbase dispose d'un shell qui permet à un utilisateur de créer des tables et de les manipuler grâce aux commandes qu'il fournit. Pour y accéder il suffit de taper la commande ci-dessous dans le terminal :

```
$ hbase shell
```

   Le résultat de cette commande est comme suit :

```
projet@nassima-Inspiron-3558:/usr/local/hbase/bin$ hbase shell
2019-07-21 19:06:37,960 WARN  [main] util.NativeCodeLoader: Unable to load nativ
e-hadoop library for your platform... using builtin-java classes where applicabl
e
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase/lib/slf4j-log4j12-1.7.10.jar!
/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4
j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 1.4.9, rd625b212e46d01cb17db9ac2e9e927fdb201afa1, Wed Dec  5 11:54:10 PS
T 2018

hbase(main):001:0>
```

4. **Lancer et arrêter Hbase :**
   Les commandes ci-dessous permettent de lancer et d'arrêter Hbase (selon cet ordre) :

```
$ start-hbase.sh

$ stop-hbase.sh
```

   Pour s'assurer que tout fonctionne, on utilise l'outil jps pour lister les processus Java en cours d'exécution :

```
projet@nassima-Inspiron-3558:~$ jps
3347 NameNode
7827 HMaster
4164 NodeManager
3766 SecondaryNameNode
8072 Jps
3996 ResourceManager
7950 HRegionServer
projet@nassima-Inspiron-3558:~$
```

Après avoir configuré tout les fichiers nécessaires pour Hadoop et Hbase nous avons été confronté aux problèmes suivants :

- **Absence du DataNode :** Le DataNode d'Hadoop relatif au stockage des données n'a pas été lancé.
- **Absence de Zookeeper :** L'absence de Zookeeper ne nous permet pas d'assurer la synchronisation et la coordination entre Hbase et le DataNode d'Hadoop.

Pour celà on s'est retourné vers les services qu'offre la plateforme Google Cloud où on a commencé à créer notre base de données Hbase sur le service Cloud Shell :

- Création de la table principale HTable 'my-table' et les deux familles de colonnes 'US' pour unités de secours et 'SR' pour site à risque.

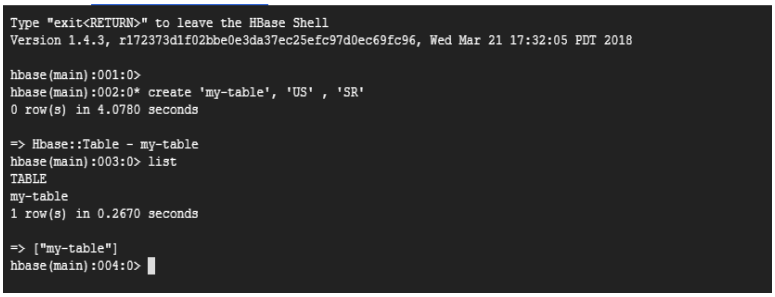

FIGURE 6.2 – Création de la table Htable.

- Création d'une clé de ligne 'r1' pour la famille de colonne 'US' qui a comme colonne 'C1 :COLUMN1'.

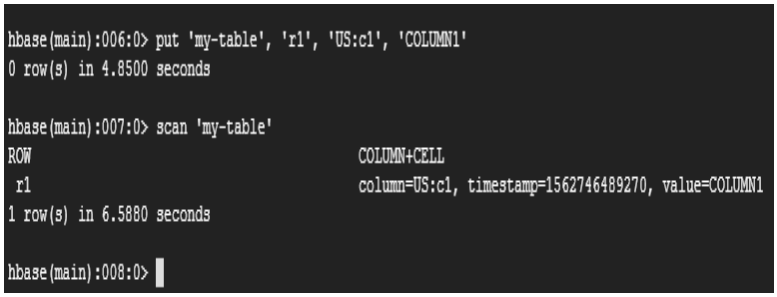

FIGURE 6.3 – Création d'une nouvelle ligne.

- Pour la même clé de ligne 'r1' on crée une colonne 'C2 :COLUMNC2' pour la famille de colonne 'SR'.

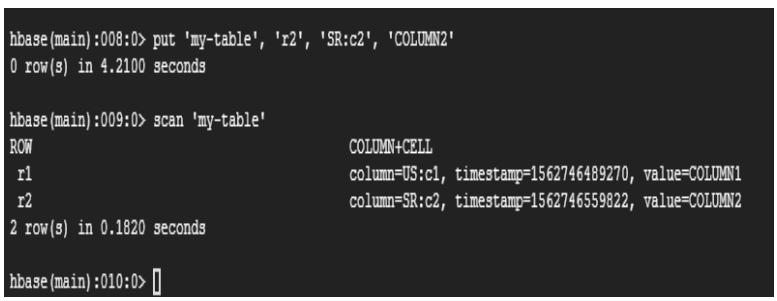

FIGURE 6.4 – Création d'une nouvelle ligne.

- Résultat de la création de la table.

```
hbase(main):005:0> list
TABLE
my-table
1 row(s) in 0.2690 seconds
```

FIGURE 6.5 – Apércu de la table.

Arriver à cet étape le temps qui nous à été offert pour l'accès aux ressources s'est achevé et le lancement d'une nouvelle tentative nécéssite de refaire tout le travail puisque les ressources sur lesquelles on a travaillé on étaient détruites.

6.2.3 Présentation de la plateforme Firebase

Afin d'accéder à la console de Firebase pour créer des bases de données , il est recommander d'avoir un compte Gmail.

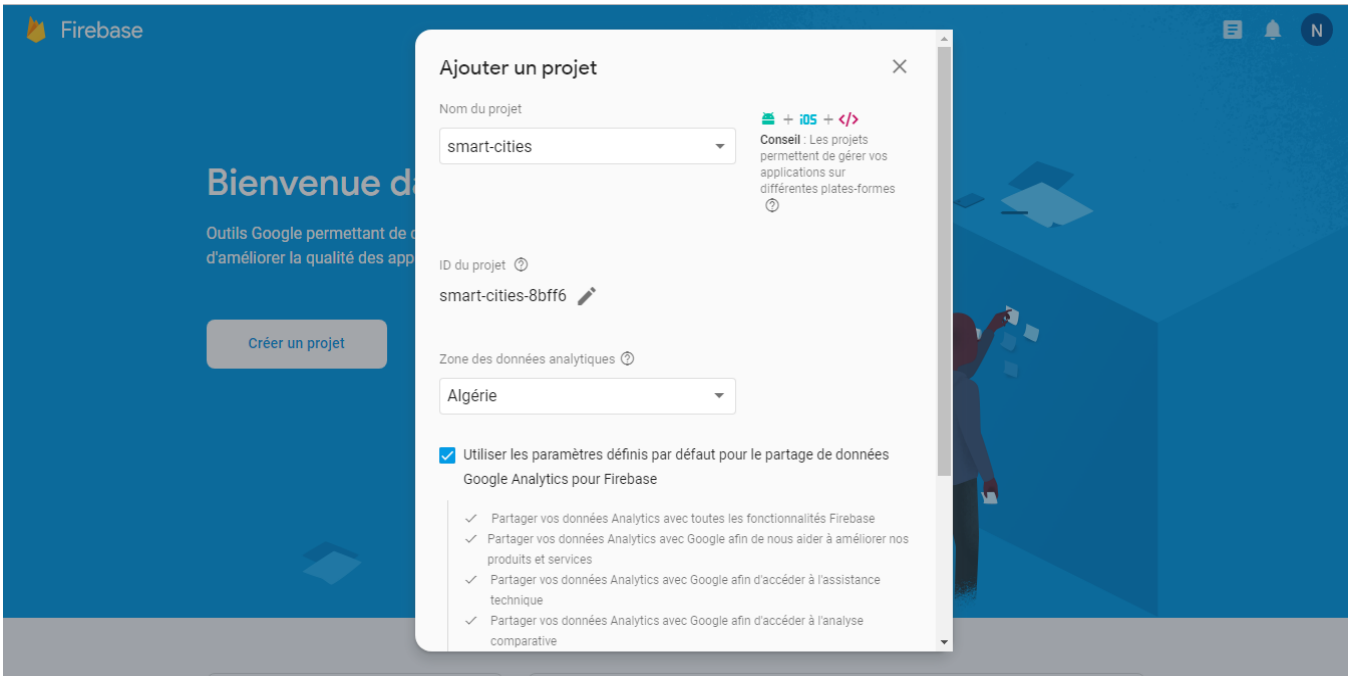

FIGURE 6.6 – Console de Firebase

Pour notre travaille nous avons choisi de le réaliser avec la base de données Cloud Firestore comme le montre la figure ci-dessous.

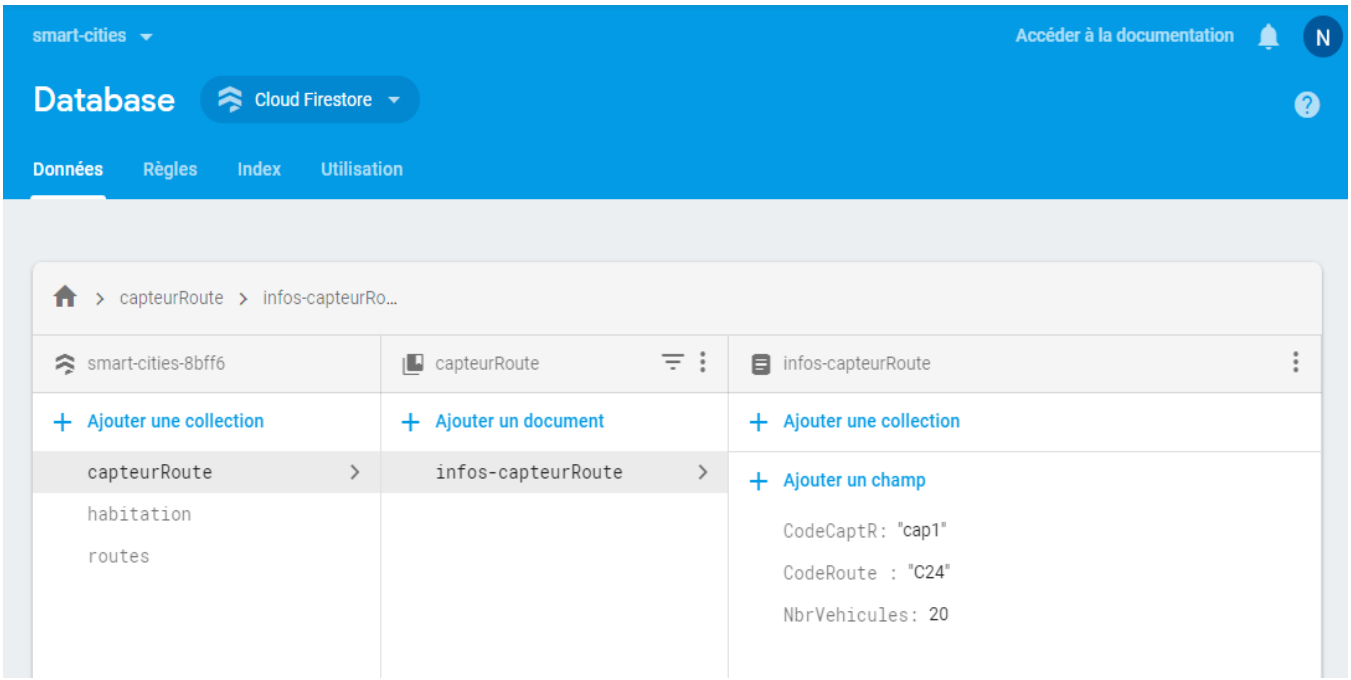

FIGURE 6.7 – Base de données Could Firestore

6.3 Langages et outils de développement

Pour développer notre propre simulateur nous avons utilisés les outils et les langages de programmation suivant :

- Le langage de programmation Java.
- Le langage de requêtes SQL.
- Les deux langages HTML5 et CSS3.
- La plateforme Java Entreprise Edition (J2EE).
- Le concept de programmation Asynchronous JavaScript et XML (AJAX).
- Le WampServer.
- VMware Workstation Pro.

6.3.1 Le langage de programmation Java

Java est un langage de programmation inspiré du langage C++, avec un modèle de programmation orienté objet, développé par Sun Microsystems en 19982 puis racheté en 2009 par la société Oracle. Il est conçu pour permettre aux développeurs d'applications d'écrire une fois le code et de l'exécuter sur toutes les plates-formes prenant en charge Java sans nécessiter de recompilation (il est portable).[6]

6.3.2 Le langage de requêtes SQL

SQL est un sigle désignant en anglais ≪Structured Query Language≫, en français ≪langage de requête structurée≫ est un langage informatique normalisé servant à exploiter des bases de données relationnelles. Il permet de façon générale la définition, la manipulation et le contrôle de sécurité de données. Dans la pratique, le langage SQL est utilisé pour créer des tables, ajouter des enregistrements sous forme de lignes, interroger une base de données, la mettre à jour, ou encore gérer les droits d'utilisateurs de cette base de données.[7]

6.3.3 Les deux langages HTML et CSS

- **Le langage HTML :** HTML est un sigle qui désigne en anglais ≪HyperText Markup Language≫, il est un langage de description de données permettant de créer des pages web pouvant être lues dans des navigateurs, permet également de structurer sémantiquement et logiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias dont des images, des formulaires de saisie, et des programmes informatiques.[8]

- **Le langage CSS :** CSS est un sigle qui désigne ≪Cascading Style Sheets≫ qui veut dire feuilles de styles en cascade, servent à mettre en forme des documents web, type page HTML ou XML. Par l'intermédiaire de propriétés d'apparence (couleurs, bordures, polices, etc.) et de placement (largeur, hauteur, côte à côte, dessus-dessous, etc.), le rendu d'une page web peut être intégralement modifié sans aucun code supplémentaire dans la page web. Les feuilles de styles ont d'ailleurs pour objectif principal de dissocier le contenu de la page de son apparence visuelle.[B2]

6.3.4 La plateforme Java Entreprise Edition (J2EE)

Java Enterprise Edition, ou Java EE (anciennement J2EE), est une spécification est une spécification pour la plate-forme Java d'Oracle, destinée aux applications d'entreprise.Elle étend Java Platform, Standard Edition (Java SE) en fournissant une API de mapping objet-relationnel, des architectures distribuées et multitiers, et des services web3. Cette plateforme se dispose d'une liste de composants contribuant à la réalisation d'une application dont on site : les servlets, les JSP [1] les JDBC [2] etc.[9]

6.3.5 Le concept de programmation Asynchronous JavaScript et XML (AJAX)

AJAX est un concept de programmation Web reposant sur plusieurs technologies comme le JavaScript et le XML. Le objectif d'AJAX est de faire communiquer une page Web avec un serveur Web sans occasionner le rechargement de la page. C'est la raison pour laquelle JavaScript est utilisé, car c'est lui qui va se charger d'établir la connexion entre la page Web et le serveur.[10]

1. JavaServer Pages(JSP) : Framework Web
2. JDBC : API de connexion à des bases de données

6.3.6 Le WampServer

WampServer (Anciennement WAMP5) est une plateforme de développement Web de type WAMP, permettant de faire fonctionner localement (sans avoir à se connecter à un serveur externe) des scripts PHP. WampServer n'est pas en soi un logiciel, mais un environnement comprenant trois serveurs (Apache, MySQL et MariaDB), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL.[11]

Il dispose d'une interface d'administration permettant de gérer et d'administrer ses serveurs.

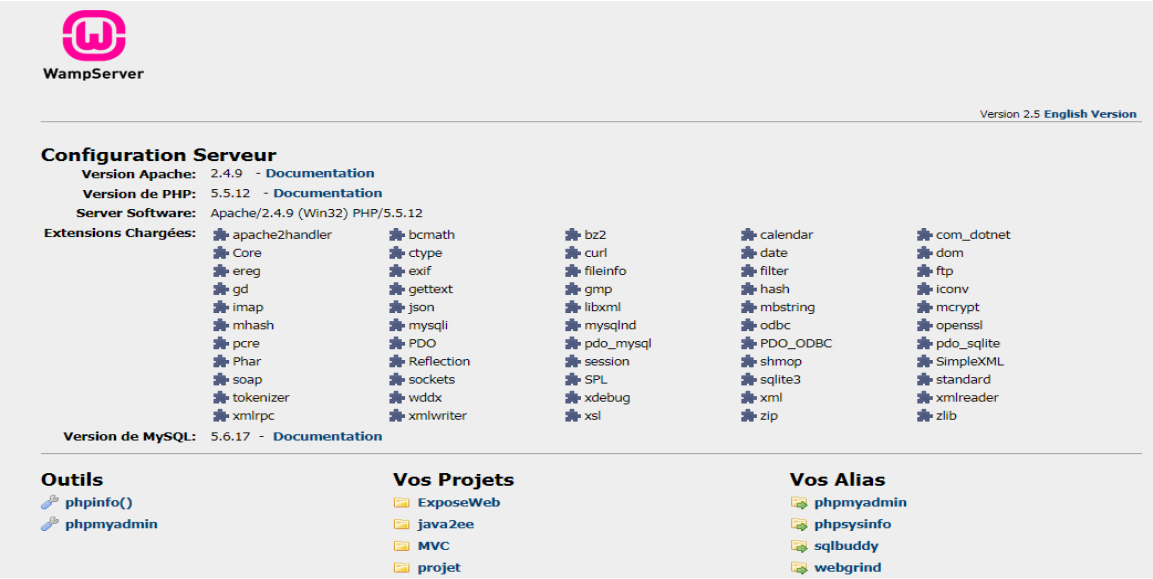

FIGURE 6.8 – Interface wampServer

6.3.7 VMware Workstation Pro

Pour faire fonctionner Cloudera nous avons décidé de le faire sous la machine virtuelle VMware Workstation Pro.
une machine virtuelle (anglais virtual machine, abr. VM) est une illusion d'un appareil informatique créée par un logiciel d'émulation ou instanciée sur un hyperviseur. Le logiciel d'émulation simule la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée. Dans notre cas nous avons utilisé la version 15.1.0.

Chapitre 7

Implémentation

7.1 Présentation des tables

7.1.1 Création de la base de données Apache Hbase

Pour la création de notre base de données HBase nous devons accéder à l'emplacement du fichier 'bin' de HBase pour pouvoir lancer le shell. Les figures ci-dessous illustent les étapes de création :

FIGURE 7.1 – Accéder au fichier HBase.

```
[cloudera@quickstart bin]$ hbase shell
2019-07-23 03:50:23,484 INFO  [main] Configuration.deprecation: hadoop.native.li
b is deprecated. Instead, use io.native.lib.available
HBase Shell; enter 'help<RETURN>' for list of supported commands.
Type "exit<RETURN>" to leave the HBase Shell
Version 1.2.0-cdh5.13.0, rUnknown, Wed Oct  4 11:16:18 PDT 2017

hbase(main):001:0> list
TABLE
0 row(s) in 0.8590 seconds
```

FIGURE 7.2 – Lancer le shell d'HBase.

```
hbase(main):002:0> create 'ville' , 'ZoneGeo'
0 row(s) in 1.4500 seconds

=> Hbase::Table - ville
hbase(main):003:0> list
TABLE
ville
1 row(s) in 0.0100 seconds

=> ["ville"]
hbase(main):004:0> scan 'ville'
ROW                                   COLUMN+CELL
0 row(s) in 0.3220 seconds

hbase(main):005:0> put 'ville' , 'CodeCaptS' , 'ZoneGeo:CodeZone' , 'cz1'
0 row(s) in 0.1100 seconds

hbase(main):006:0> scan 'ville'
ROW                                   COLUMN+CELL
 CodeCaptS                            column=ZoneGeo:CodeZone, timestamp=1563879616054, value=cz1
1 row(s) in 0.0300 seconds

hbase(main):007:0> put 'ville' , 'CodeCaptS' , 'ZoneGeo:Localisation' , 'zoneA'
0 row(s) in 0.0100 seconds

hbase(main):008:0> scan 'ville'
ROW                                   COLUMN+CELL
 CodeCaptS                            column=ZoneGeo:CodeZone, timestamp=1563879616054, value=cz1
 CodeCaptS                            column=ZoneGeo:Localisation, timestamp=1563879654292, value=zoneA
1 row(s) in 0.0230 seconds
```

FIGURE 7.3 – Création de la table 'Ville'.

7.1.2 Création de la base de données Cloud Firestore

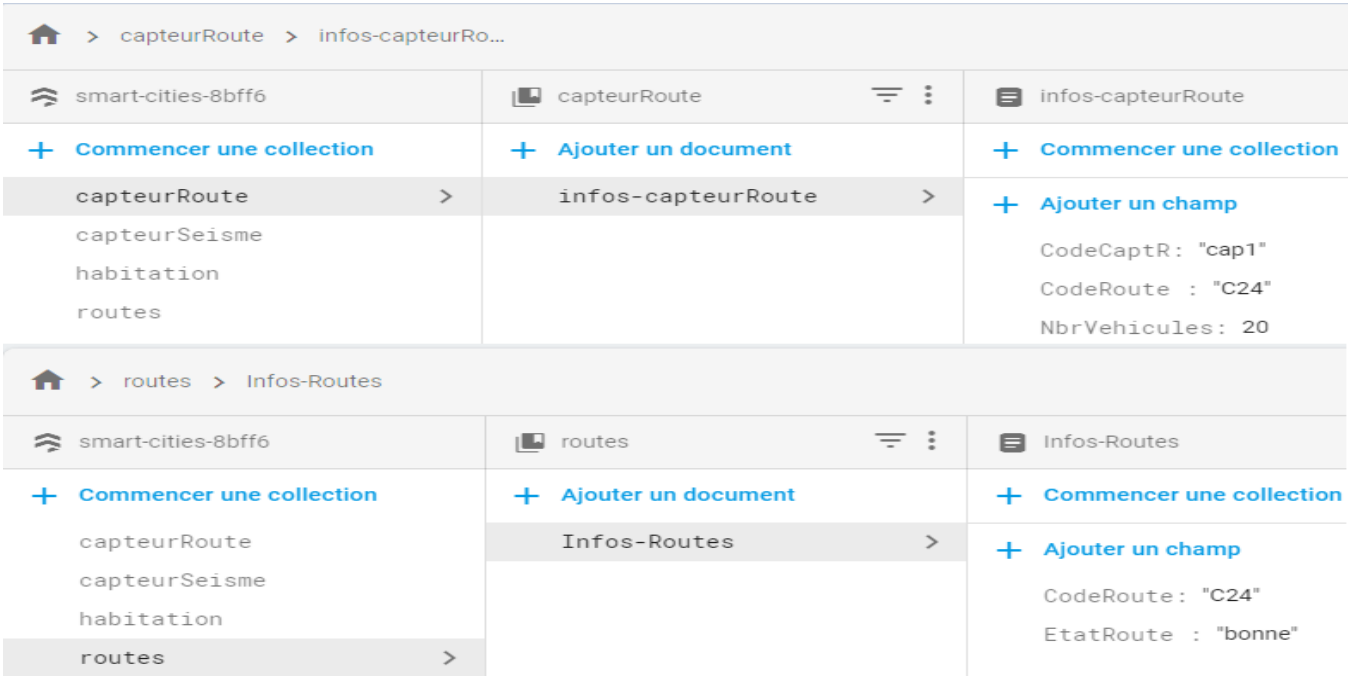

FIGURE 7.4 – Base de données Cloud Firestore

7.2 Présentation du simulateur

7.2.1 L'interface d'alerte

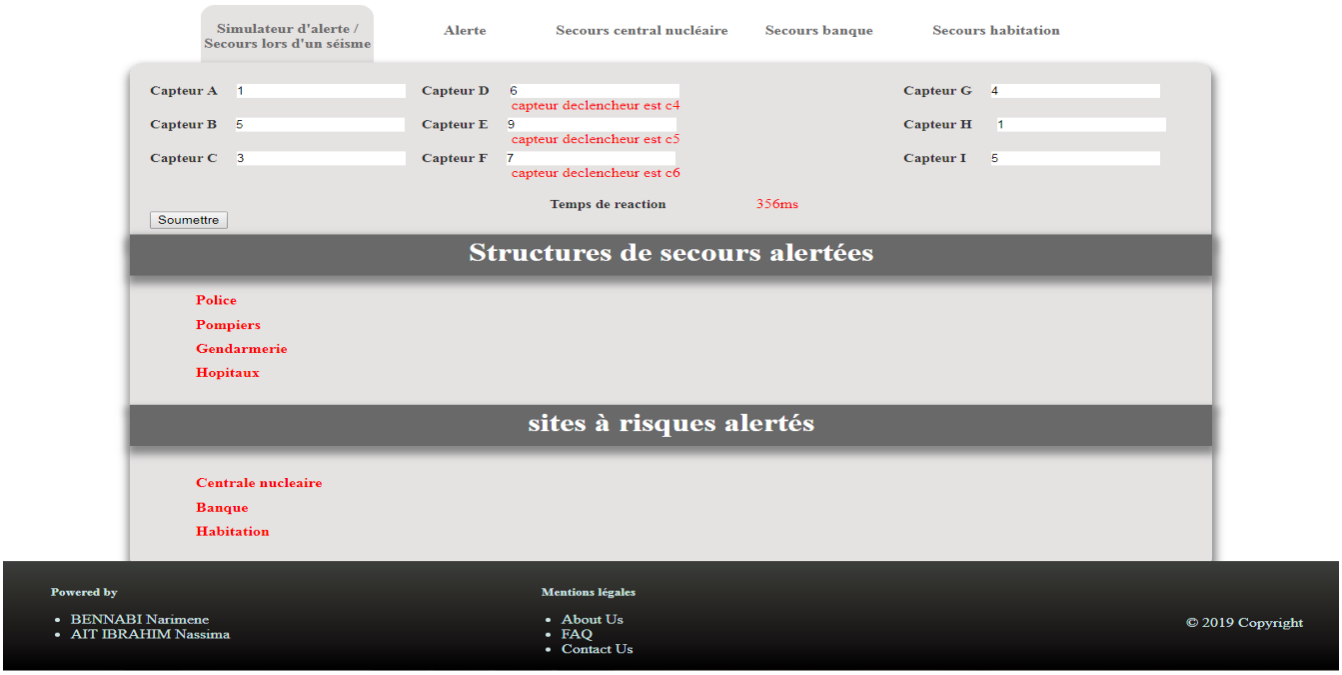

FIGURE 7.5 – Interface d'alerte des structures de la ville.

7.2.2 L'interface de secours des habitations

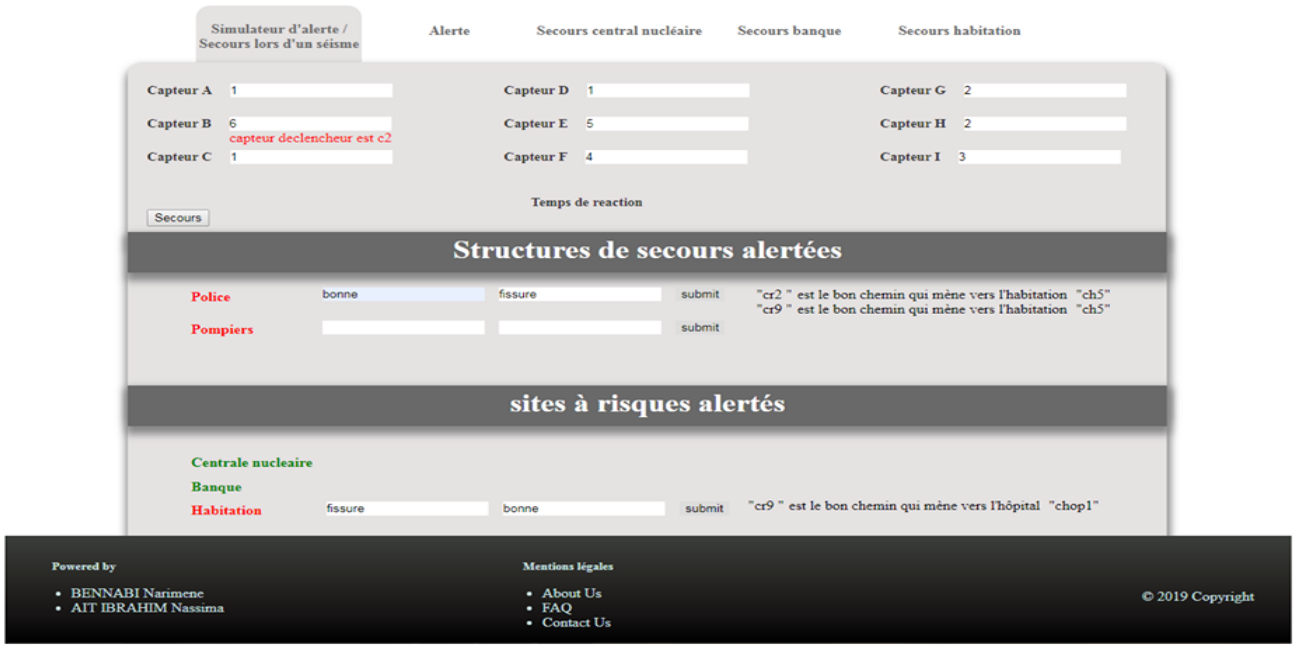

FIGURE 7.6 – Interface de secours des habitations.

7.2.3 L'interface de secours des centrales nucléaires

FIGURE 7.7 – Interface de secours des centrales nucléaires.

7.2.4 L'interface de secours des banques

FIGURE 7.8 – Interface de secours des banques.

Conclusion générale

Avant de dresser les différents points de conclusion, rappelons tout d'abord les objectifs fixés pour essayer de répondre à la problématique fixée au départ.

1. Faire une recherche Bibliographique pour comprendre le contexte du Big Data, IoT, et Base de données NoSQL.
2. Imaginer le contexte d'implémentation d'une base de données NoSQL pour une ville intelligente dans le cas d'un tremblement de terre.
3. Etudier deux outils HBase et Firebase :
   (a) Faire une analyse du fonctionnement.
   (b) Elaborer une critique en dressant les avantages et inconvénients.
   (c) Rédiger une synthèse.
4. Imaginer les données utiles à ce système, les récolter et les structurer en base de données NoSQL.
5. Imaginer les scénarios pour exploiter ces données.
6. Essayer d'entrevoir des perspectives.

A l'issue de ce travail nous pouvons dire que :

1. Suite à notre travail de recherche Bibliographique nous avons enrichie et approfondie nos connaissances dans les domaines Big Data, IoT, et les bases de données NoSQL et nous avons aussi amélioré notre style de rédaction.
2. Nous avons délimiter un cadre de travail à notre porté dans lequel nous avons pu imaginer, collecter et structurer les données en mode NoSQL.
3. Elaborer une synthèse sur HBase et Firebase inspiré d'un article de recherche (Voir Annexe) ce qui nous a permi d'améliorer notre analyse pour la synthèse.
4. Implémenter tout de même notre base de données NoSQL sous HBase et Firebase malgré le problème de limites matérielles.

Perspectives

En ce qui concerne les problèmes d'enrichissement de la sémantique, de scalabilité seront des points à étudier avec HBase et Firebase sous réserve de disposer de moyens pour implémenter la base de données NoSQL dans des conditions optimums.

Annexe

Webographie

Partie 1 : *Etat de l'art*

Chapitre 1 : *Big Data*

[1] https ://www.lebigdata.fr/definition-big-data

[2] https ://www.redsen-consulting.com/fr/inspired/data-analyse/big-data

[3] https ://www.commentcamarche.net/faq/43812-le-big-data-les-avantages-pour-l-entreprise

[4] http ://fonderie-infocom.net/blognumerique/index.php/2018/03/22/le-big-data-represente-t-il-un-danger/

[5] http ://cestpasmonidee.blogspot.com/2011/07/gartner-les-3-dimensions-des-big-data.html

[6] https ://www.lemagit.fr/definition/Big-Data

[7] https ://www.delphisoft-group.com/servicescloud/

[8] https ://www.ivision.fr/cloud-et-virtualisation-les-differences/

[9] https ://www.lebigdata.fr/hadoop

[10] https ://www.lebigdata.fr/hdfs-fonctionnement-avantages.

[11] https ://www.lebigdata.fr/mapreduce-tout-savoir

[12] https ://www.lemagit.fr/tutoriel/Comprendre-quelles-architectures-de-stockage-pour-le-Big-Data

[13] https ://www.dellemc.com/fr-fr/storage/discover-flash-storage/definitions.htm

[14] https ://www.scalair.fr/blog/stockage-objet

[15] https ://www.allerin.com/blog/top-5-sources-of-big-data

[16] https ://www.elaee.com/fiches-metiers/chief-digital-officer-cdo.

Chapitre 2 : *L'Internet des Objets (IoT)*

[1] https ://www.digora.com/fr/blog/definition-iot-et-strategie-iot.

[2] https ://www.synox.io/4-choses-a-savoir-sur-linternet-des-objets/

[3]https ://www.electronique-mixte.fr/formation-pdf/formation-pdf-aop-ampli/cours-en-electronique/electronique-analogique/capteurs-et-detecteurs/

[4] https ://fr.wikipedia.org/wiki/Donnée

[5] http ://www.globalsecuritymag.fr/Donnees-non-structurees-pourquoi,20180726,80096.html

[6] https ://fr.wikipedia.org/wiki/Informations_non_structurées

[7] https ://www.globalsign.fr/fr/blog/l-iot-ameliore-l-agriculture-avec-le-betail-connecte-et-les-drones.

[8] https ://www.matooma.com/fr/s-informer/actualites-iot-m2m/smart-cities-liot-villes-de-demain.

Chapitre 3 : *Les bases de données NoSQL*

[1] https ://sqlpro.developpez.com/SGBDR/ReglesCodd/.

[2] https ://www.createursdemondes.fr/2016/05/limites-des-sgbdr-pour-le-big-data-nosql-et-newsql/.

[3] https ://www.linkedin.com/pulse/big-data-les-3-principaux-probl%C3%A8mes-rencontr%C3%A9s-lon-dun-ingrid-phd.

[4] https ://blog.deimos.fr/categories/sql/page/5/

[5] https ://stph.scenari-community.org/idl-bd/idl-nosql/co/nos01.html

[6] https ://fracademic.com/dic.nsf/frwiki/213226

[7] https ://www.lebigdata.fr/apache-cassandra-definition

[8] https ://www.ambient-it.net/top-meilleures-db-nosql-2019/

[9]https ://news.icodia.com/securite/firebase-decouverte-dune-faille-sur-le-service-dhebergement-dapplications-de-google

Chapitre 4 : *Apache Hbase et Firebase*

[1] https ://docs.microsoft.com/fr-fr/azure/hdinsight/hbase/apache-hbase-overview.

[2] https ://www.lebigdata.fr/apache-hbase-tout-savoir

[3] https ://fr.slideshare.net/larbret/hadoop-hbase

[4] https ://www.lebigdata.fr/apache-zookeeper-tout-savoir-sur-le-systeme-de-coordination

[5] http ://www.linusnova.com/2013/09/hbase-la-base-nosql-de-hadoop/

[6] https ://openclassrooms.com/fr/courses/4872916-creez-un-backend-scalable-et-performant-sur-firebase

[8] https ://firebase.google.com/docs/database/rtdb-vs-firestore

[9] https ://firebase.google.com/docs/firestore/data-model

[10] https ://firebase.google.com/docs/firestore/manage-data/data-types

[11] https ://firebase.google.com/docs/firestore/manage-data/structure-data

Partie 2 : *Conception*

Chapitre 5 : *Etude et conception*

[1] https ://www.cnil.fr/fr/definition/smart-city

[2] https ://smartcity.brussels/le-projet-definition

[3] http ://www.mbadmb.com/2017/02/11/le-citoyen-au-centre-de-la-smart-city/

[4] https ://fr.wikipedia.org/wiki/A%C3%A9roport_international_de_Narita

[5] https ://fr.wikipedia.org/wiki/M%C3%A9tro_de_Tokyo

[6] https ://fr.wikipedia.org/wiki/Gare_de_Tokyo

[7] https ://www.kanpai.fr/voyage-japon/guide-train-metro-tokyo

[8] https ://www.vivrelejapon.com/location-voiture/japon-en-voiture/les-types-de-route-au-japon

[9] https ://fr.wikipedia.org/wiki/D%C3%A9partement_de_la_Police_M%C3%A9tropolitaine_de_Tokyo

[10] http ://content.time.com/time/world/article/0,8599,2059780,00.html

[11] https ://whatis.techtarget.com/fr/definition/collecte-de-donnees

Partie 3 : *Réalisation*

Chapitre 6 : *Choix des outils technologiques*

[1] https ://www.formation-bigdata.com/java-pour-hadoop/

[2] https ://www.joel-douillet.com/google-firebase/

[3]http ://www.mosaique-info.fr/glossaire-web-referencement-infographie-multimedia-informatique/m-glossaire-informatique-et-multimedia/448-mysql-definition.html

[4] https ://fr.wikipedia.org/wiki/Google_Cloud_Platform

[5] https ://fr.wikipedia.org/wiki/Cloudera

[6] https ://fr.wikipedia.org/wiki/Java_(langage)

[7] https ://fr.wikipedia.org/wiki/Structured_Query_Language

[8] https ://fr.wikipedia.org/wiki/Hypertext_Markup_Language

[9] https ://fr.wikipedia.org/wiki/Jakarta_EE

[10]http ://docplayer.fr/10428654-Ajax-est-l-acronyme-d-asynchronous-javascript-and-xml-autrement-dit-javascript-et-xml-asynchrones.html

[11] https ://fr.wikipedia.org/wiki/WampServer

Bibliographie

Partie 1 : *Etat de l'art*

Chapitre 1 : *Big Data*

[B1] http ://www.lsis.org/espinasseb/Supports/BD/Article-BigData-TI-2016

[B2] https ://tdwi.org/articles/2017/02/08/10-vs-of-big-data.aspx

[B3] https ://thesai.org/Downloads/Volume7No3/Paper_37-Extract_Five_Categories_CPIVW.pdf

[B4] http ://www.lsis.org/espinasseb/Supports/BD/Article-BigData-TI-2016

Chapitre 2 : *L'Internet des Objets (IoT)*

[B1] Présentation générale de l'Internet des objets, ITU-T Y.2060, 2012-06-15

[B2] Objets connectés, un 360 pour bien les comprendre, CIGREF, Rapport, décembre 2016.

[B3] Thèse de doctorat, Système de gestion de flux pour l'Internet des objets intelligents, École doctorale Sciences et Technologies de Versailles (STV), 2015.

[B4] Thèse de doctorat, Imad Saleh,Internet des Objets (IdO) : Concepts, Enjeux, Défis et Perspectives , 2018.

[B5] PRÉPARER LA RÉVOLUTION DE L'INTERNET DES OBJETS , Imad Saleh, PRÉPARER LA RÉVOLUTION DE L'INTERNET DES OBJETS , 2016.

[B6] INTERNET OF THINGS IoT, ILNAS/ANEC, INTERNET OF THINGS (IoT) , july 2018.

[B7] IoT : la sécurité de l'internet des objets, la sécurité de l'internet des objets, 2015.

[B8] Big Data et objets connectés, L'Institut Montaigne, Big Data et objets connectés ,avril 2015.

Chapitre 3 : *Les bases de données NoSQL*

[B1] Bases de données et systèmes relationnels, Claude DELOBEL et Michel ADIBA,DUNOD,bases de données et systèmes relationnels,1985.

[B2] Les bases de données NoSQL et le Big Data,Rudi Bruchez, EYROLLES,les bases de données NoSQL et le Big Data, 1985.

[B3] NoSQL : les meilleures solutions open source,Collaborateurs Smile, SMILE,NoSQL : les meilleures solutions open source", 2011.

[B4] Fondamentaux pour le Big Data , Pierre Senellart,Télécom ParisTech,Limites des

systèmes classiques de gestion de bases de données

[B5] Le NoSQL- Cassandra,Xavier Maletras, thèse professionnelle, Université Paris, 2012

[B6] Adoption d'une solution NoSQL dans l'entreprise, Matteo DI MAGLIE, En vue de l'obtention du Bachelor HES, Carouge, 12 septembre 2012.

[B7] Approche de migration d'une base de données relationnelle vers une base de données NoSQL orientée colonne, KOUEDI Emmanuel, En vue de l'obtention du diplôme de MASTER 2, Mai 2012.

[B8] http ://dspace.univ-tlemcen.dz/bitstream/112/8812/1/Etude-comparative-des-bases-de-donnees-NoSQL.pdf

[B9] Synthèse d'étude et projets d'intergiciels : bases NoSQL, Mathieu Roger

Partie 2 : *Conception*

Chapitre 5 : *Etude et Conception*

[B1] http ://www.planseisme.fr/IMG/pdf/RP-56663-FR.pdf

[B2] http ://www.i-tradeonline.com/CATALOGUEFRN.pdf

Partie 3 : *Réalisation*

Chapitre 6 : *Choix des outils technologiques*

[B1] https ://perso.univ-rennes1.fr/pierre.nerzic/Hadoop/semaine8.pdf

[B2] https ://fr.wikibooks.org/wiki/Le_langage_CSS