République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université de 8 Mai 1945 – Guelma -

Faculté des Mathématiques, d'Informatique et des Sciences de la Matière

Département d'Informatique

Mémoire de Fin d'études Master

Filière : Informatique

Option : Sciences et technologie de l'information et de la communication

Thème :

Une approche sémantique pour la gestion de données issues de l'internet des objets dans un contexte de Big Data

Encadré par :

Dr. Lynda Djakhdjakha

Présenté par :

Laiadi Adlene

Octobre 2020

بسم الله الرحمن الرحيم

Remerciements

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes, à qui je voudrais témoigner toute ma gratitude.

Je voudrais tout d'abord adresser toute ma reconnaissance à mon encadreur qui a accepté ma candidature, pour son dévouement et ces remarques minutieuses.

J'adresse mes sincères remerciements à tous mes professeurs, qui m'ont enseigné au sein du département informatique de l'université du 08 Mai 1945.

Je remercie mes très chers parents : ma Mère et mon Père, ainsi tous qui m'ont offert leurs soutiens inconditionnés : mes frères et ma sœur, ma femme, la femme de mon frère Mehdi, les enfants de ma sœur, mes enfants : Mohamed Amine, Adam, Maria et mes amis Malek, Mohamed et Lazhar et Baki.

Un grand merci à mon responsable de travail qui sans lui je n'aurai pas l'occasion d'être parmi vous.

Résumé

Selon plusieurs rapports, l'internet des objets est parmi les domaines les plus générateurs de Big Data, mais aussi l'un des domaines à potentiel économique énorme. Pour en extraire de la valeur, une gestion lucide qui permet l'exploitation efficace de ces données est primordiale.

En outre, Les lacs de données ont été proposés et largement acceptés dans la communauté scientifique comme une solution pour quelques problèmes qui peuvent être soulevés par le Big Data. Basé essentiellement sur le schéma en lecture, les lacs de données s'emblent ne pas offrir d'autres facilités pour l'exploitation efficace de ces données comme : la découverte des données, la gouvernance, provenance …etc.

Compte tenu de ces défis, nous proposons une nouvelle approche pour gérer les métadonnées dans un lac de données, utilisant des techniques du web sémantique édifiées sur une nouvelle ontologie, pour bénéficier de son expressivité et de ses capacités de raisonnement, combinées avec des techniques d'apprentissage automatique, nous tirerons profit du meilleur des deux.

Mots clé : *Internet des Objets, IoT, Big Data, lac de données, Hadoop, Spark, métadonnées, web sémantique, ontologie, raisonnement d'ontologie, Règles SWRL, Apprentissage automatique.*

Abstract

According to many reports, IoT field is one of the biggest Big Data generators, and the one among most promising economic valued fields as well, but its value is constrained by the ability to judiciously manage and exploit generated data.

Data Lake was proposed and accepted within the scientific community as a solution to many problems raised by the big data phenomenon, but not all problems. The schema on read that data lake offers doesn't seem to provide any further facilities regarding problems such: data discovery, governance, lineage …etc.

Taking into account these challenges, we suggest a new approach to handle metadata using semantic web technologies uplifted on a new proposed ontology, to benefit from its expressiveness and reasoning capabilities, combined with machine learning techniques we shall take advantage of the best of the two.

Key words: *IoT, Big Data, Data Lake, Hadoop, spark, metadata, semantic web, ontology, ontology reasoning, SWRL rules, machine learning.*

.

Table des matières

Introduction Générale 6

PARTIE I : Etat de l'art 8

Chapitre 1 : Internet des objets 9
1. Introduction 9
2. Définitions et écosystème Internet des Objets 9
2.1. Le concept Internet des Objets 9
2.2. L'écosystème IoT 10
3. Architecture IoT 11
4. IoT selon la perspective Centrée données 11
4.1. Enrichissement des données IoT 12
4.2. Techniques d'enrichissements 12
5. Domaines d'application [26] 13
6. Conclusion 14

Chapitre 2 : Big Data 15
1. Introduction 15
2. Définition 15
3. Type des traitements des Big Data 15
4. Modèle de données, stockages et de programmation 16
4.1. Modèle de stockage 16
4.2. Modèle de données 17
4.3. Modèle de programmation 17
5. Domaines d'applications 18
6. Architecture des systèmes de Big Data 19
7. Outils et technologies 20
7.1 Systèmes de traitement des Big Data 20
7.2 Systèmes de traitement de graphes à grand échelle 21
7.3 Systèmes de traitement RDF massif 22
7.4 Systèmes de traitement de flux massifs 23
8. Conclusion 24

Chapitre 3 : Lac de Données 25
1. Introduction 25
2. Définition 25
3. Caractéristiques du LD 26
3.1. Entrepôt de données classique vs Lac de Données 26
4. Architecture 28
5. Défis des lacs de données 29
6. Gestion des métadonnées 30
6.1. Caractéristiques de base pour un système de métadonnées 30
6.2. Techniques de stockage des métadonnées 31

6.3. Critères des métadonnées 32
6.4. Typologies des métadonnées 32
7. Conclusion…….. 32
Chapitre 4 : Web Sémantique et ontologie 33
1. Introduction 33
2. Web Sémantique et ontologie 33
3. Représentation des connaissances du Web Sémantique 34
4. Stockage des données RDF 34
5. Raisonnement et règles SWRL 34
6. Conclusion…… 35
Chapitre 5 : Travaux connexes 36
1. Introduction 36
2. Travaux connexes 36
3. Conclusion…….. 40

Partie II : Conception et Implémentation 41

Chapitre 6 : Conception et Implémentation 42
1. Introduction 42
2. Conception 42
2.1. Ontologie proposée 42
2.2. Création des règles SWRL 43
2.3. Système Proposé 44
2.3.1. Diagramme des cas d'utilisation 45
2.3.2. Description textuelle du cas d'utilisation 45
2.3.3 Diagramme de classes 47
3. Implémentation 49
3.1. Environnement de développement 49
3.2. Environnement Opérationnel 50
3.3. Configuration matérielle 50
3.4. Diagramme de composants 51
3.5. Intégration d'ontologie avec le système 52
3.6. Gestion de métadonnées 55
3.7. Inférence du domaine 56
3.8. Architecture système 56
4. Expérimentation 58
5. Conclusion…… 60
Conclusion générale 61
Références…… 63

Table des figures

Figure 1-1 : Architecture IoT norme : ISO/IEC 30141[15],[12]. 11
Figure 2-1 : Taxonomie des modèles de stockage [7]. 16
Figure 2-2 : Modèles de Big Data [7] 17
Figure 2-3 : NIST Big Data Architecture de référence (NBDRA) [38]. 20
Figure 3-1 : LD et les concepts associés [46]. 26
Figure 3-2 : Architecture fonctionnelle de LD [42]. 29
Figure 4-1 : Pile Web Sémantique. 33
Figure 6-1 : Diagramme des cas d'utilisation 45
Figure 6-2. Diagramme de classes métiers. 48
Figure 6-3 : Diagramme de classes services 49
Figure 6-4 : Pile technologique de l'écosystème Hadoop [83] 50
Figure 6-5 : Diagramme de composants 52
Figure 6-6 : Ontologie modélisant les métadonnées d'un dataset [visualisée avec OWLviz] 53
Figure 6-7 : Edition des règles SWRL 53
Figure 6-8 : Classes d'intégration d'ontologie dans le système. 54
Figure 6-9 : Une partie des tests unitaires 55
Figure 6-10 : Extraction des caractéristiques d'apprentissage, basée sur la taxonomie 56
Figure 6-11 : Architecture système en couches 57
Figure 6-12 : Base de connaissances initiale 58
Figure 6-13 : Aperçu sur la détection des capteurs 58
Figure 6-14 : Fenetre de calcul de similarités 59
Figure 6-15 : Base de connaissance après création de métadonnée 59

Table des tableaux

Tableau 1-1 : Caractéristiques des données IoT. 10
Tableau 1-2 : *Domaine d'application d'IoT [26]*. 14
Tableau 2-1 : Taxonomie des modèles de programmation, adapté de [7]. 18
Tableau 2-2 : Domaines d'application des Big Data..... 19
Tableau 2-3 : Systèmes de traitement des Big Data..... 21
Tableau 2-4 : Systèmes de traitement de graphes à grande échelle. 22
Tableau 2-5 : Systèmes de traitement RDF massif. 23
Tableau 2-6 : *Systèmes de traitement de flux massifs.*..... 24
Tableau 3-1 : Entrepôt de données classique vs LD. 28
Tableau 3-2 : Techniques et méthodes de stockage des métadonnées. 31
Tableau 5-1 : Comparaison de différentes approches de gestion des métadonnées..... 40
Tableau 6-1 : Description des concepts de l'ontologie proposée..... 43
Tableau 6-2 : Description textuelle du cas d'utilisation « Ajouter une métadonnée ». 47
Tableau 6-3 : Configuration de l'environnement Hadoop. 51

Table des Abbreviations

IoT Internet **o**f **T**hing

ITU The **I**nternational **T**elecommunication **U**nion's

NIST National **I**nstitute of **S**tandard and **T**echnology.

NBDRA NIST **B**ig **D**ata **R**eference **A**rchitecture

SPL Streams **P**rocessing **L**anguage

LD Lac de **D**onnées

MD Méta**D**onnées

ETL Extract, Transform, **L**oad

RDF Resource **D**escription **F**ramework

OWL Web **O**ntology **L**anguage

SWRL Semantic **W**eb **R**ule **L**anguage

LCS Longest **C**ommon **S**ubsequence

UML Unified **M**odeling Language

Introduction Générale

''Celui qui détient l'information détient le pouvoir, celui qui l'entretien, détient le monde''
Adam Smires

Au cours des dernières années, le monde a connu une nouvelle vague de données caractérisées par de hauts volumes, haute fréquence et variété, et qui a submergé le monde informatique, rendant nos outils traditionnels tels qu'on a connait obsolète face à ce phénomène. Cela nous a obligé de faire un changement radical, non pas seulement au niveau des outils, mais aussi au niveau des méthodes et modèles. Cependant, à l'ombre des grands défis apparaissent les grandes opportunités, les entreprises qui ont pu apercevoir cette réalité se sont ceux qu'on appelle aujourd'hui les géants informatiques.

Ce phénomène d'explosion de la quantité de données est communément référencé par l'appellation « Donnée Massive » ou « Big Data ». Un exemple qui illustre bien ce phénomène est le Boeing 787, qui génèrent 5 giga octets de données de capteurs par seconde [1]. Un autre exemple concerne la firme Google qui est confrontée à gérer plus de vingt milliards de datasets [2]. A cause de cette quantité phénoménale de données, de nouveaux problèmes ont vu le jour : gestion, sécurité, gouvernance, redondance, volume …etc.

Un concept récent, appelé « Lacs de Données » immerge dans le monde informatique, est supposé être une solution pour stocker et gérer les Big Data, mais ce dernier n'impose aucun formalisme pour sa structuration. Les techniques du Web Sémantique à savoir l'ontologie possèdent un potentiel important de formalisation. C'est dans ce contexte qu'intervient notre travail, visant le domaine d'Internet des Objets (IoT) un domaine digne d'être parmi les premiers générateurs de Big Data. On estime selon quelques ressources [3], [4] que d'ici aux 2025 on aura 152 200 objets IoT connectés par minute et qui peuvent générer 79,4 Zeta Byte de données par année. En conséquence, avoir une gestion efficace de tels : volume de données, fréquence, diversité et hétérogénéité s'avère très nécessaire pour qu'en puisse en tirer profils.

A travers ce mémoire, nous proposons une approche de gestion de métadonnées dans un environnement de Big Data, à savoir un Lac de Données, modélisée et orientée par ontologie et les technologies du web sémantique.
L'objectif de ce travail consiste à modéliser les métadonnées des datasets IoT sous forme d'ontologie, identifier les attributs ayant une relation avec les capteurs et classifier ces datasets par domaine.

Ce mémoire est structuré en deux parties comme suit : une première partie portant sur l'état de l'art et une deuxième sur la conception et l'implémentation de notre proposition.

La première partie est répartie en cinq chapitres :
Le premier chapitre introduira le concept d'internet des objets et ses notions de base.
Dans le deuxième et troisième chapitre nous survolerons les concepts des Big Data et du Lac de Données respectivement, qui sont des concepts étroitement liés. En fait, nous présenterons quelques définitions et nous aborderons leurs architectures, les technologies associées et les défis qui en découlent.
Dans le quatrième chapitre, nous entamerons les concepts d'ontologie et le web sémantique où nous métrons l'accent sur leurs utilités, expressivités et les opportunités qu'ils offrent.
Nous clôturerons cette partie par le chapitre cinq, où nous citeront les travaux connexes à le nôtre, en montrant leurs philosophies, modèles et conceptions.

La deuxième partie abordera notre processus qui supporte l'ontologie des métadonnées proposée et expliquera notre proposition d'une manière abstraite via des diagrammes UML afin de faciliter la compréhension de notre implémentation. Elle développera les détails de notre implémentation, les choix technologiques et le produit final qu'a été abouti par notre vision.

Une conclusion clôturera notre travail, où nous mettrons l'accent sur les résultats que nous avons obtenu, nos constats sur les obstacles et difficultés rencontrées et les perspectives entrevues qui peuvent porter sur une éventuelle évolution ou amélioration de notre approche.

PARTIE I : Etat de l’art

Chapitre 1 : Internet des objets

1. Introduction

L'IoT est un concept qu'on entend souvent ces derniers temps. Il marque le début d'une révolution numérique qui a radicalisé notre style de vie.

Dans ce chapitre nous allons définir ce concept, dévoiler les notions qui constituent son écosystème, ses caractéristiques, ses enjeux et citer à la fin quelques domaines d'application de l'IoT.

2. Définitions et écosystème Internet des Objets

2.1. Le concept Internet des Objets

Dans[5], l'IoT est définie comme étant une infrastructure réseau global dynamique avec capacité d'auto-configuration. Cette infrastructure est basée sur des standards définis et des protocoles de communication interopérables, faisant intervenir des objets physiques et virtuels ayant une identité et des attributs, qui sont capables d'utiliser des interfaces intelligentes et peuvent être intégrées comme un réseau d'information.

Le tableau suivant résume les caractéristiques fondamentales de l'IoT [6][7] :

Caractéristique	**Description**
Interconnectivité	Tout objet peut être connecté à l'infrastructure globale d'information et de communication.
Services liés aux objets	La capacité de fournir des services.
Hétérogénéité du flux de données	Le flux de données tend d'être multimodal et hétérogène en matière de formats, sémantiques et vélocités.
Changement dynamique	Les états des dispositifs changent dynamiquement : connecté, non connecté …etc.
Massive	Un nombre important de dispositifs connectés, générant de grande quantité d'information.
Sensibilité à la confidentialité et à la sécurité	Pour les applications IoT, qui collectent et traitent des données personnelles, il faut envisager des techniques d'anonymisation, cryptage …etc.

Variabilité dans la qualité des données	Le flux de données, souvent bruité et incomplet[1], crée une incertitude dans le processus analytique
Nature temps réel	A cause de la haute vélocité pour certaines applications, il est primordial de traiter le flux en temps réel.
Nature temporelle et spatiale	Le flux de données est étiqueté avec des informations temporelles et spatiales qui peuvent influencer sa valeur.
Biais de données	Les données peuvent conduire à un traitement biaisé et une compréhension approfondie est nécessaire avant le déploiement opérationnel.

Tableau 1-1 : *Caractéristiques des données IoT.*

2.2. L'écosystème IoT

Un écosystème est un ensemble de matériels, logiciels et de protocoles de communication qui joue un rôle pour le bon fonctionnement d'un système donné. On peut résumer l'écosystème IoT par l'équation suivante [4]:

$$IoT = Services + Données + Réseaux + Capteurs$$[2]

- Services : les plateformes qui fournissent des fonctionnalités traitant les défis soulevés par les systèmes IoT ; allant des services de base comme la gestion de configuration et de sécurité aux services applicatifs comme l'analytique.
- Données : les données générées par les appareils IoT.
- Réseaux : l'infrastructures réseaux, protocoles de communication et interfaces permettant aux appareils IoT de se communiquer.
- Appareils : ils sont la pierre angulaire de tous systèmes IoT et peuvent être selon leurs capacités classés sous trois classes : (i) la première comprend ceux de basses gammes comme les capteurs et les actionneurs, (ii) la deuxième comprend les appareils de moyennes gammes plus capables que la première, (iii) la dernière comprend les appareils de hautes gammes comme *Raspberry Pi*, ayants leurs propres systèmes d'exploitation avec des capacités de traitement plus sophistiquées[9].

[1] Fréquents dans les applications réelles à cause : des erreurs d'échantillonnage, défaillance matérielle …etc.
[2] L'auteur emploi le terme capteurs (sensors), dans ce contexte on préfère appareils.

3. Architecture IoT

Afin de comprendre un système complexe, il est indispensable d'avoir une vue de haut niveau mettant en évidence les composants qui le constituent, l'interaction entre ceux-ci et les règles qui les régissent. On trouve plusieurs architecture de référence dans la littérature [10],[11] et [12]. Les architectures proposées ont les mêmes buts mais divergent dans leurs implémentations.

Plusieurs tentatives de standardisation ont été initiées[3] par certains organismes, principalement par : ITU[4] [14], IEEE et la norme ISO [15],[16].

La **figure 1-1** présente l'architecture de référence ISO, qui est fondée sur six couches[5]-domaines – et plusieurs couches verticales[6].

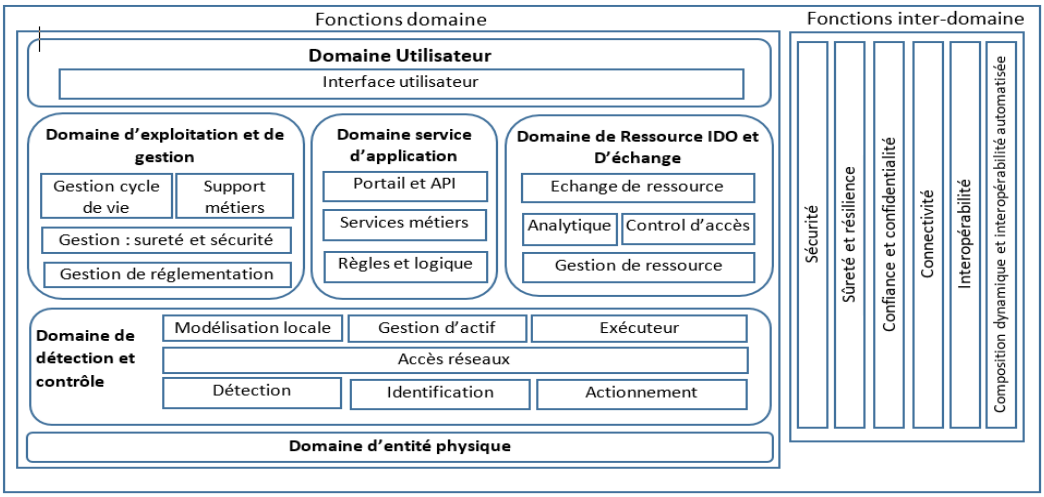

Figure 1-1 : *Architecture IoT norme : ISO/IEC 30141[15],[12]***.**

4. IoT selon la perspective Centrée données

Par nature, les données IoT ont une étroite relation avec le Big Data. Néanmoins, vu qu'ils sont générés par un nombre énorme de capteurs distribués, leurs acquisitions, intégrations, stockages, traitements et exploitations sont devenus un besoin urgent pour les entreprises afin d'atteindre leurs objectifs.

[3] Chaque organisme/institue est biaisé pour résoudre un ou plusieurs défis techniques donnés : la norme IEEE est plus impliquée pour l'aspect connexion tandis que la norme ISO y beaucoup plus pour le vocabulaire[13].
[4] ITU : The International Telecommunication Union's -Telecommunication Standardization - https://www.itu.int/en/Pages/default.aspx
[5] Dans la littérature l'architecture IoT est basée souvent sur trois couches : perception, réseau et applicative[17], d'autres visions existent tel que : architecture orientée services, basé middleware …etc[18].
[6] Une couche verticale est fortement couplée avec toutes les autres couches horizontales

Pour comprendre ces besoins, il est nécessaire d'exposer les caractéristiques des données IoT[7], qui sont fondamentalement [20] :

- Multi-sources, haute hétérogénéité.
- Données dynamiques à grand échelle.
- Données de bas niveau avec faible sémantique.
- Données inexactes.

Bien que le but de traitement de données soit de tirer profit pour les entreprises, la nature brute de ces dernières n'apporte aucun gain immédiat. Le grand défi est, donc, de leur donner un sens ou en d'autres termes de les transformer en information et par la suite en connaissance utilisable pour leur donner une valeur. Ce processus est connu dans la littérature par l'enrichissement des données.

4.1. Enrichissement des données IoT[8]

L'enrichissement ou la sémantisation des données est le formatage des données, avec des balises, des propriétés spéciales, des étiquettes ou d'autres formes. Il fusionne les connaissances du domaine, les informations contextuelles et les données brutes captées afin de faciliter la compréhension et le traitement de celles-ci par la machine.

En d'autres termes, c'est une forme d'organisation de connaissances afin de représenter des relations sémantiques pour rehausser l'interopérabilité entre des sources de données hétérogènes[22]. L'intérêt de ce processus est immédiatement évident en termes d'intégration, d'interopérabilité et de compréhension des données.

4.2. Techniques d'enrichissements

On peut trouver dans la littérature plusieurs techniques d'enrichissement, chaque technique présente des avantages et des inconvénients, se diffère au niveau d'expressivité, formalisme et performance.

Parmi ces techniques [21],[23] on peut citer : Clé valeur, schémas de balises, modélisation graphique[9], orienté objet, modélisation logique, modélisation par ontologie, apprentissage

[7] L'auteur fait référence au Cloud dans son article, la relation cloud/Big data selon[19] :le Big data est le produit, le cloud est le conteneur.

[8] On trouve dans la littérature un concept proche : informations de contexte [21] : ce sont des informations générées par le traitement des données captées brutes, leurs consistance est vérifiée et dont des métadonnées sont ajoutées.

[9] On trouve l'utilisation d'UML ,ORM[24] : Object Role Modeling

automatique, linked stream processing[10], réutilisation des connaissances de domaine[11], raisonnement distribué, système de recommandation.

5. Domaines d'application [26]

On peut classer les domaines d'application de l'IoT en différentes catégories, comme suit :

Catégorie	**Service**	**Connectivité**	**Défis**
Personnel/Domicile	Santé	Wifi, 3G, 4G LTE.	- Hétérogénéité, - Interopérabilité, - Méthodes de contrôles, - Temps réel, - Sécurité, - Qualité de service.
Entreprise	Ville intelligente	Wifi, 3G, 4GLTE Satellite.	- Passage à l'échelle, - Identification et découverte, - Hétérogénéité, - Virtualisation, - Méthodes de contrôles, - Temps réel, - Big Data, - Consommation d'énergie, - Sécurité.
	Environnement intelligent		
	Surveillance vidéo	Wifi, Satellite.	
Mobile	Transport intelligent/ trafic intelligent	WSN, Satellite.	- Identification et découverte, - Méthodes de contrôles, - Passage à l'échelle, - WSN, - Conception et architecture IoT, - Temps réel,

[10] Extension du Modèle de données RDF pour représenter un flux de données des capteurs et des réseaux sociaux[25].

[11] Utilisation des techniques: LOD, LOV, LOR: pour Linked Open : data, Vocabulary, Reasoning respectivement , LOD : ''Le Web des données ouvertes (Linked Open Data ou LOD) s'appuie sur des vocabulaires formels (RDFS, OWL) pour définir les types et propriétés des ressources décrites''[26]

			- Coût.
Utilités	Réseau intelligent	Wifi, Satellite, Cellulaire	- Consommation énergétique, - Ordonnancement et équilibrage de charges, - Temps réel, - Passage à l'échelle, - Méthodes de contrôles, - Coût.
	Energie intelligente		
	Smart water		

Tableau 1-2 *:* *Domaine d'application d'IoT* [27]**.**

6. Conclusion

Dans ce chapitre, nous avons défini le concept de l'IoT, son architecture et ses caractéristiques principales. Aussi nous avons constaté que les données issues d'IoT sont des données hétérogènes, multi-sources, dynamiques et à grand échelle.

En conséquence, on se retrouve face à un domaine en pleine expansion connu sous l'appellation de « *Big Data* », c'est précisément ce que nous allons aborder dans le chapitre suivant.

Chapitre 2 : Big Data

1. Introduction

Le *Big Data* est un phénomène technologique en plein expansion incité par l'avancement et la convergence des technologies de l'information, de communications et la prolifération des technologies IoT. Les Big Data nécessitent le recours aux outils non traditionnels pour leurs traitements et gestion. Nous définissons dans ce chapitre ce concept et nous présentons ses caractéristiques et ses outils afin de mieux le comprendre.

2. Définition

Les systèmes de Big Data concernent les ensembles de données qui ne peuvent pas être perçus, acquis, gérés et traités par les outils matériels et logiciels traditionnels en un temps tolérable[28]. Ils sont caractérisés par les 5V[12] : **V**olume, **V**élocité, **V**ariété , **V**aleur et **V**éracité [30]:

- Volume : les données sont produites d'une manière significativement plus grande que les systèmes classiques ont l'habitude de gérer.
- Vélocité : le rythme de génération des données.
- Variété : les données sont générées depuis des sources différentes et ayant des formats divers.
- Valeur : la transformation des données massives pour extraire l'information.
- Véracité : elle représente l'incertitude et l'inconsistance des données.

3. Type des traitements des Big Data

Vu la nature et la diversité des Big Data, trois grandes catégories de traitement ont vu le jour [31] :

- Le traitement par lots : favorise l'aspect volume de données, il consiste à exécuter des séries de programmes ou de tâches sur des données en entrée. Il utilise les ressources système de façon optimale et efficace grâce à la notion de priorité des tâches.
- Le traitement en temps réel des données : favorise la vélocité, en l'appel aussi traitement

[12] En peut trouver dans la littérature le modèle 7V et même 10V [29].

de flux de données, ou traitement en continue des données en entrée, et opère sous des conditions temporelles strictes.

- Au milieu des deux, une approche hybride s'impose en pratique pour profiter des avantages combinés des deux autres[32].

On trouve aussi, dans la littérature, une classification selon la technique de traitement [33]:

- Données parallèles : les données sont distribuées sur des nœuds de traitements parallèles, où chaque nœud exécute la même tâche sur des segments différents de données distribuées. Ces techniques sont caractérisées généralement par leur tolérance aux erreurs.
- Traitements parallèles : les données sont traitées en parallèle par plusieurs processeurs. En conséquence, le processus de traitement est distribué sur plusieurs nœuds de traitements.
- Graphe parallèle : modélise les traitements comme des vertex qui s'exécutent en parallèle et interagissent le long des arêtes dans un graphe.

4. Modèle de données, stockages et de programmation

Les Big Data ont révolutionnées les systèmes informatiques traditionnels. En fait, de nouveaux modèles de données, de stockage et de mode de programmation ont été émergés.

4.1.Modèle de stockage

Le modèle de stockage est la base sur laquelle reposent tous les systèmes Big Data. On distingue trois modèles principaux : modèle basé sur bloc, basé fichier et basé objet [7]. La **figure 2.1** donne un aperçu sur les plateformes de Big Data selon leur modèle de stockage.

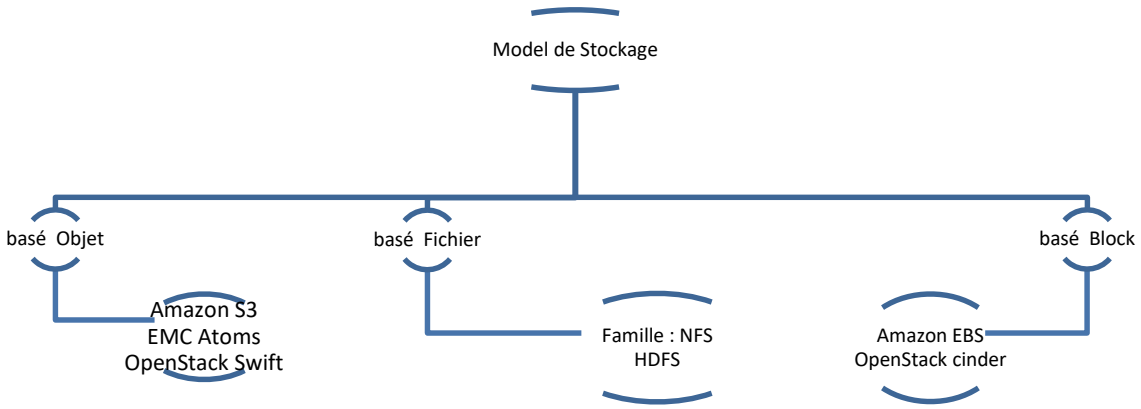

Figure 2-1 : *Taxonomie des modèles de stockage [7].*

4.2. Modèle de données

Traditionnellement le modèle relationnel a dominé les solutions de stockage et de gestion des systèmes d'information. Mais avec l'augmentation colossale des volumes de données ainsi que leurs variétés les systèmes SGBD ont montré leurs limites.

En conséquence, une nouvelle génération de modèle plus scalable et plus flexible a vu le jour, on les nomes NoSQL (Not Only SQL). La **figure 2-2** résume la taxonomie des modèles de données ainsi que les technologies associées[13].

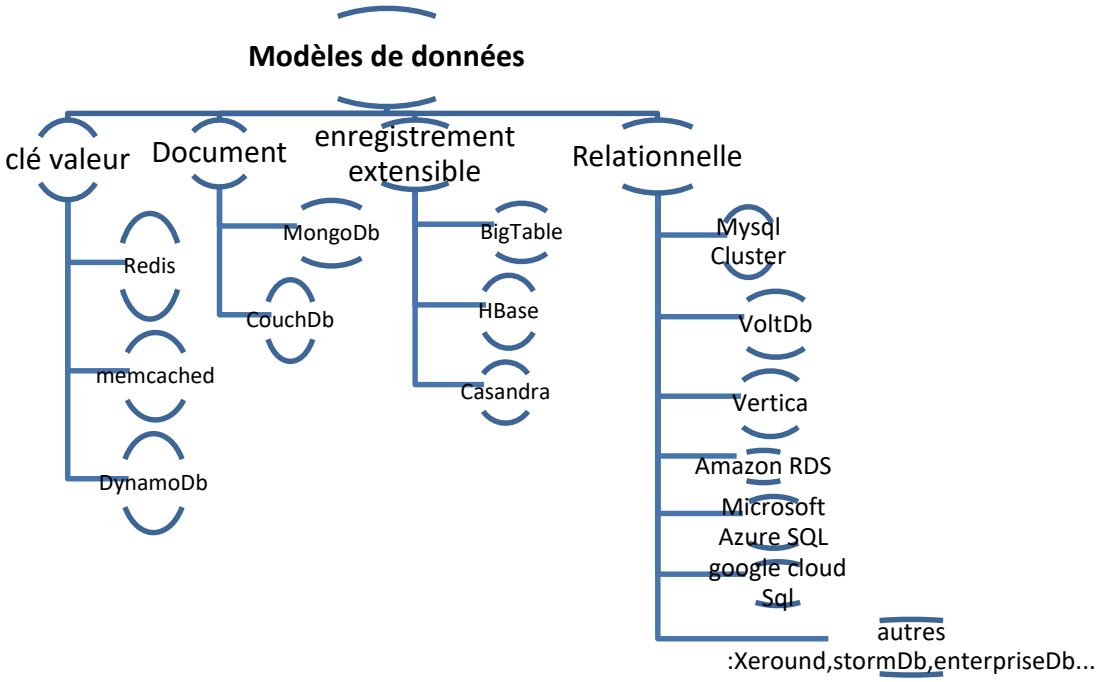

Figure 2-2 : *Modèles de Big Data [7].*

4.3. Modèle de programmation

Après la présentation des modèles de stockage et de données, nous décrirons dans cette section les modèles de programmation qui sont mis à la disposition des développeurs afin de manipuler ces données. Le **tableau 2-1** donne un aperçu sur les modèles de programmation existants et les technologies associées.

Modèle de programmation	**Technologie**
Map-Reduce	MapReduce, Hadoop

[13] On ne vise pas à fournir une liste exhaustive des technologies associées

Fonctionnel	Spark, Flink
Basé SQL	HiveQL, CasandarQL, SparkQL, Drill, Presto, Impala
Basé Acteur	Akka, Storm, S4
Analytiques et statistiques	R, Mahout
Flow de Donnée	Oozie, Dryad
Bulk Synchronous parallel	Giraph, Hamma
DSL Haut Niveau	Pig latin, Crunch, Cascading, LINQ, Trident, Green Marl, AQL, Jql

Tableau 2-1 : *Taxonomie des modèles de programmation, adapté de [7].*

5. Domaines d'applications

Les domaines d'application des Big Data sont résumés dans le tableau suivant [35] :

Industrie	**Cas d'utilisation**
Système de santé	Tendances de qualité de service vis-à-vis des habitudes hygiènes. Système d'aide à la décision Clinique. Surveillance des patients à distance. Profilage des patients. Recherche comparative efficace. Prédiction des maladies.
Marketing	Marketing basé sur la localisation. Analyse du comportement en magasin. Analyse des sentiments.
Fabrication	Opérations basées sur les capteurs. Chaîne d'approvisionnement et gestion des stocks. Valeur de cycle dirigée par raccourcis.
Publique/gouvernement	Identifier les besoins fondamentaux du public. Décongestion du trafic de circulation. Conformité civique pour réduire la pollution sonore atmosphérique et hydrique.
Banque et assurance	Détection de fraude. Prédictions de la clientèle.

	Analyse des risques.
Télécom	Annonces géographiquement ciblées. Gestion des réponses d'urgences. Surveillance à distance des objets personnels. Planification urbaine.
Gaz de pétrole	Perçage à distance. Prédictions pétrolières. Champ pétrolier numérique.

Tableau 2-2 : *Domaines d'application des Big Data.*

6. Architecture des systèmes de Big Data

Dans la littérature, plusieurs architectures ont été proposées pour les Big Data [36], [37]. Une initiative de fonder une architecture de référence a été mise en place par le NIST [14] appelée « *NBDRA*[15] ». La **figure 2-3** montre les composantes clés de cette architecture**.**

La *NBDRA* est composée essentiellement de cinq composants fonctionnels [38]:

- Système Orchestrateur : définit et intègre les activités de données d'application requises dans un système vertical opérationnel.
- Fournisseur de données : introduit les nouveaux flux de données ou d'informations dans le système de Big Data.
- Fournisseur d'applications Big Data : exécute un cycle de vie des données pour respecter les exigences en matière de sécurité et de confidentialité, ainsi que les besoins définis par le système orchestrateur.
- Fournisseur de Framework de Big Data : établit un Framework dans lequel s'exécutent certaines applications de transformation, tout en gardant la confidentialité et l'intégrité des données.

[14] NIST: National Institute of Standard and Technology.
[15] NBDRA: NIST Big Data Reference Architecture

- Consommateur de données : les utilisateurs finaux ou d'autres systèmes qui utilisent les résultats du fournisseur d'applications.

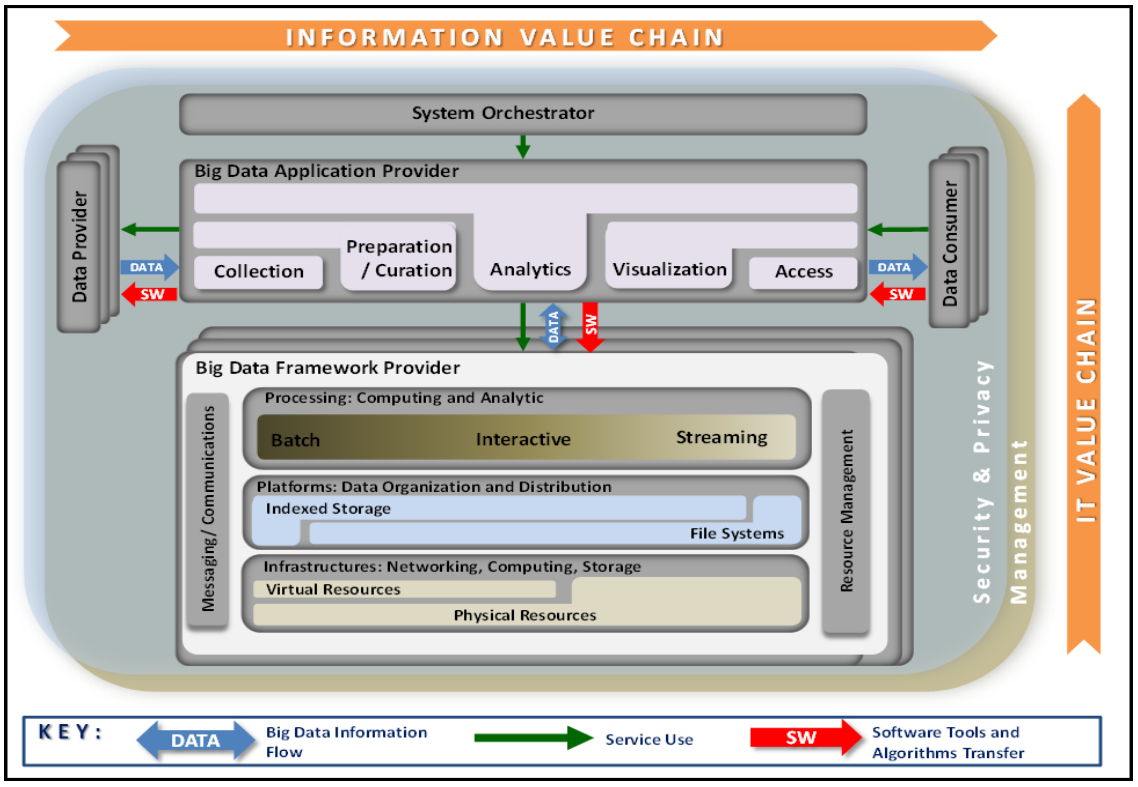

Figure 2-3 : *NIST Big Data Architecture de référence (NBDRA) [38].*

7. Outils et technologies

Dans cette section, on va présenter les outils et les technologies utilisées dans les systèmes de Big Data [16].

7.1 Systèmes de traitement des Big Data

Outil	**Points clef**
Hive	Fournit les concepts du modèle relationnel dans le contexte des Big Data. HiveQL propose du SQL compilé en MapReduce. Supporte les commandes DDL sauf Update et delete.
Impala	Moteur SQL de traitement massif parallèle sur Hadoop. Utilise les mêmes métadonnées et syntaxe SQL que HiveQL. L'exécution des requêtes est optimisée pour le traitement local.
IBM Big SQL	Accès aux données SQL sauvegardées sur Infosphere BigInsight.

[16] Synthétisé à partir[39]

	Requêtes interactives optimisées. Fournit une vue logique des données à partir de métadonnées.
Spark SQL	Traitement relationnel. Offre la possibilité d'exploiter la Machine Learning.
HadoopDB	Combine les avantages de Hadoop avec la performance et l'efficacité des bases de données parallèles. Cluster plusieurs bases de données (PostgreSQL), en utilisant Hadoop comme coordinateur de tâches. Requêtes SQL exécutées d'une façon transparente en MapReduce.
Presto	Moteur SQL open source de Facebook. Requêtes analytiques interactives sur de gros volumes. Requête combinant les données de plusieurs sources.
Apache Tajo	Entrepôt distribué de données pour Hadoop. Vise une latence minimale et des requêtes Ad Hoc scalable. Peut analyser les données sur HDFS, Amazon S3, OpenStack Swift et le system fichier local.
Google Big Query	Système de requêtes Ad Hoc scalable et interactives pour l'analyse des données en lecture seule. Moteur SQL à schéma libre.
Apache Phoenix	Requêtes SQL sur HBase exclusivement. Utilise java JDBC API au lieu HBase API.
Microsoft PolyBase	Permet au SQL server d'exécuter des requêtes sur Hadoop. Permet la création des tables externes qui référencent des données sur hadoop.

Tableau 2-3 : *Systèmes de traitement des Big Data.*

7.2 Systèmes de traitement de graphes à grand échelle

Outils	**Points clef**
Apache Giraph	Implémentation Java sur Hadoop pour Pregel Spécification. Exécute les tâches de traitement de graphe uniquement en Map (pas de Reduce), et utilise HDFS pour les entrées/sorties.

	Utilise Zookeeper pour la coordination. Offre une API de haut niveau, qui masque les détails d'exécution.
Google GPS	Implémentation Java Open source. Extension pour l'API Pregel.
IBM Mizan	Implémentation C++ de Pregel. Repartitionnement basé sur la surveillance des caractéristiques des sommets de graphes pendant l'exécution.
Pregelix	Traitement massif des graphes basé sur les flots de données itératifs orientés ensemble. Traite les messages (Communication par messages) et les états des sommets comme des tuples relationnels avec un schéma bien défini.
Pregel++	Implémentation C++. Vise l'optimisation d'échange de messages.
Giraph++	Propose un paradigme : *Think Like Graph.*
GraphLab	Implémentation C++ open source. Basé sur la notion d'abstraction de mémoire partagée et le modèle de traitement GAS (Gather, Apply, Scatter).
PowerGraph	Vise à combiner les avantages de Pregel et GraphLab en même temps. Offre un mécanisme d'équilibrage de charge pendant le traitement de graphe.
GhraphChi	Implémentation c++. Système centralisé conçu pour le traitement massif des graphes qui résident dans le support de stockage secondaire d'un nœud. Ne gère pas le clustering, ni la tolérance aux fautes.

Tableau 2-4 : *Systèmes de traitement de graphes à grande échelle.*

7.3 Systèmes de traitement RDF[17] massif

Outils	**Points clef**
SparQL[18]	Standard W3C : requête et extraction des données RDF.

[17] Nous aborderons ce concept dans le chapitre 4.
[18] SparQL est diffèrent de Spark SQL

HadoopDB-RDF	Partitionne le graphe RDF sur plusieurs nœuds où chacun exécute un moteur de requêtes centralisé.
Trinity.RDF	Sauvegarde le graphe RDF dans sa forme native. Modèle clef-valeur en mémoire.
H2RDF+	Modèle clef-valeur. Schéma indexé sur HBase. Exécute les requêtes simples sur une seule machine et ceux qui sont complexes sur un nombre fixe de nœuds distants.

Tableau 2-5 : *Systèmes de traitement RDF massif.*

7.4 Systèmes de traitement de flux massifs

Outils	**Points clef**
Twitter Storm	Gestion de flux distribués avec tolérance aux pannes. Concurrence basée sur le modèle de l'acteur[19].
IBM Infosphere Streams	Plateforme analytique pour le développement rapide. Conçue pour gérer le flux émanant de milliers de sources et des millions de messages par seconde[38]. Fournit un modèle de programmation et un IDE pour la définition des sources de données. Utilise le langage de programmation SPL[20] .
Sormy System	Service de gestion de flux de données distribués en continu basée sur les techniques de stockage en Cloud. Utilise les tables de hachages distribuées pour la distribution des requêtes. Haute disponibilité grâce à la réplication des requêtes. Assume qu'une requête peut être entièrement exécutée sur un seul nœud.
Apache S4	Plateforme distribuée, scalable, tolérante aux fautes et enfichable.
Flink Streaming	Extension de Flink API. Gère le flux de données de plusieurs sources.

[19] Modèle de l'acteur : les acteurs sont des composants concurrents qui communiquent par le biais de ports et interagissent selon un motif commun d'interaction [40].

[20] SPL : *(Streams Processing Language) est un langage de composition de flux de données réparties. SPL comporte des types primitifs, des structures de programme et des définitions conçues pour les flux de données en continu* [41].

Apache Samza	Utilise Apache Yarn pour l'allocation et l'ordonnancement distribuée des ressources. Optimiser pour la gestion des messages volumineux distribués.

Tableau 2-6 : *Systèmes de traitement de flux massifs.*

8. Conclusion

Dans ce chapitre nous avons abordé le concept de Big Data, ses caractéristiques inhérentes, son architecture fonctionnelle et les technologies qui orbitent autour de ce concept.
Dans le chapitre suivant nous allons entamer le concept du Lac de Données et découvrir sa relation avec les Big Data.

Chapitre 3 : Lac de Données

1. Introduction

Dans le chapitre précèdent, nous avons constaté que le concept de Big Data est un phénomène imposé par le cours d'évolution des technologies de l'information et de la communication.
Traditionnellement, on a eu recours aux entrepôts de données au sens classique pour le stockage et le traitement des données. Cependant, ils ont rapidement montré leurs limites pour faire face à un tel flot de données accablant, ainsi les Lacs de Données (LD) ont vu le jour.
Dans ce chapitre, nous décrirons le concept de LD, ces caractéristiques et son architecture. Nous définirons aussi le concept de métadonnée, la gestion des métadonnées en tant qu'une solution pour résoudre les problèmes des LD, les caractéristiques et les techniques de stockage de métadonnées.

2. Définition

Le concept de LD est un concept relativement nouveau [42]. Plusieurs définitions existent, chacune met l'accent sur un aspect différent. Plusieurs points de vue peuvent être considérés : les entrées, le processus de traitement, l'ingestion des données, l'architecture, la gouvernance des données, la gestion des métadonnées ou encore le perspectif utilisateur. De plus d'autres mettent en surbrillance le coût économique [43]. Dans la littérature, néanmoins, tous se mettent d'accord que le LD est un entrepôt de stockage des données brutes et dans leurs formats natifs [44].
Un LD est une méthodologie incitée par l'émergence des Big Data, basée sur des technologies à faible coûts, qui améliore : la capture, le raffinement, l'archivage et l'exploitation des données brutes au sein de l'entreprise. Il regroupe des données brutes non structurées ou multi structurées[21], qui ont dans leurs plupart une valeur non encore connue par l'entreprise [45].
La **figure 3-1** nous donne un aperçu sur les concepts qui gravitent autour du LD.

[21] Une autre appellation pour les données semi-structurées.

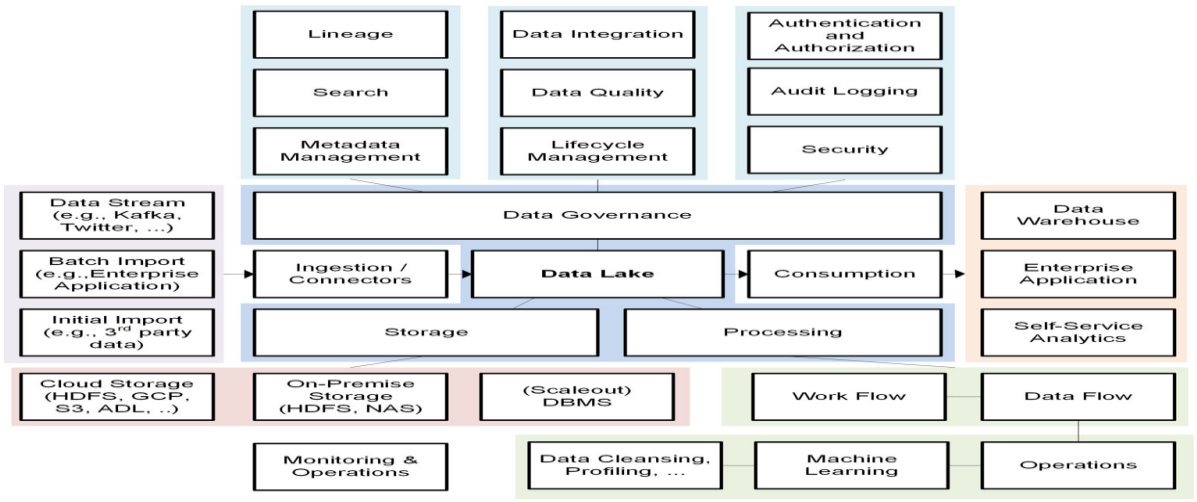

Figure 3-1 : *LD et les concepts associés [46].*

3. Caractéristiques du LD

Les principales caractéristiques du LD sont les suivants,[46] :

- Stockage de données à faible coût dans leurs formats natifs.
- Possibilité de stocker une grande variété de types de données.
- Offre le potentiel d'utiliser les données uniquement lorsque cela est nécessaire, ainsi le coût de modélisation et d'intégration par cette approche est réduit, c'est ce qu'on appelle schéma en lecture.
- Comme les données sont souvent floues, les utilisateurs doivent développer leurs propres méthodes pour consommer les données.
- Des politiques de gouvernance sont mises en œuvre pour identifier, réutiliser et disposer les données.
- Distinguer la provenance des données et garder une trace des sources, les origines, qui les ont modifiés… etc.

3.1. Entrepôt de données classique vs Lac de Données

Pour désambiguïser un concept, il est souvent bénéfique de l'entreposer avec son concept homologue, pour cela nous allons citer les différences entre la notion du LD et l'entrepôt de

données classiques dans le tableau suivant [47],[45] et [48] :

Caractéristique	**Entrepôt de donnée classique**	**Lac de Donnée**
Traitement	Traitement en batch Des centaines voire des milliers d'utilisateurs concurrents. Améliore les performances requêtes Support utilisateurs et analytiques interactifs.	Traitement en batch à l'échelle[22]. Support utilisateur interactif en cours d'amélioration. ETL[23]
Schéma	Schéma en écriture Requiert du travail au début du processus de traitement.	Schéma en lecture Agilité extrême et facilité de capture des données, mais requiert du travail à la fin du processus. Bien adapté pour les données dont le type est inconnu.
Passage à l'échelle	Valable pour des grands volumes avec un coût modéré.	Valable pour des volumes extrêmes avec un coût bas.
Méthodes d'accès	Via SQL et des outils standards	Via un pseudo SQL, des programmes créés par les développeurs et d'autres méthodes.
Complexité	Jointure complexe	Traitement complexe
Données/Stockage	Traitées. Proviennent habituellement des systèmes transactionnels.	Brutes
Chronologies de données	Un temps considérable est voué pour l'analyse des sources variées.	Stocke toutes les données tant que leur stockage est possible.

[22]Il existe des propositions pour le traitement en temps réel[49] [50],[51].
[23] Extract ,Transform, Load

		Stocke les données dont l'utilité est immédiate, mais aussi ceux ayant un futur potentiel. Possibilité de remonter dans le temps pour analyse des données
Utilisateurs	Adéquat pour les utilisateurs opérationnels.	Adéquat pour ceux qui en besoin d'une analyse profonde nécessitant une modélisation prédictive ou statistique .
Coûts	Utilisation efficace CPU/IO	Coût bas de stockage et de traitement.
Avantage	Temps de réponse rapide. Performance consistante. Utilisation facile. Intégration des données. Analyse inter-fonctionnelle. Charger une fois, utiliser plusieurs fois.	Superbe Passage à l'échelle. Support de programmation. Changement radical.

Tableau 3-1 : *Entrepôt de données classique vs LD.*

4. Architecture

Le concept LD est relativement nouveau [42]. Il n'existe pas un standard ou une architecture approuvée dans la littérature[42].

L'architecture des LDs a été concrétisée au cours du temps selon plusieurs visions. Au début, elles ont été basées sur une architecture plate mono-zone, dont les données sont stockées dans leurs formes natives brutes, au cours du temps elles ont évoluées vers une architecture multizone[24][42]. Bien qu'il n'y ait pas un consensus sur le nombre de zones et les caractéristiques de celles-ci, l'idée générale est identique : assigner les données à une couche spécifique selon le degré de traitement de ces dernières [52]. Pour le traitement en temps réel, deux variantes d'architectures émergentes existent :l'architecture ***Lambda*** et ***KAPPA*** [53],[49], dont l'idée générale est de consacrer une zone distincte pour le traitement des données de nature

[24] On trouve souvent le terme anglais ***Pond*** qui signifie : étang, bassin, mare - dictionnaire Larousse -

temps réel.

Pour donner une concrétisation au concept du LD, la **figure 3-2** montre un schéma global d'une architecture fonctionnelle basée sur quatre zones [42].

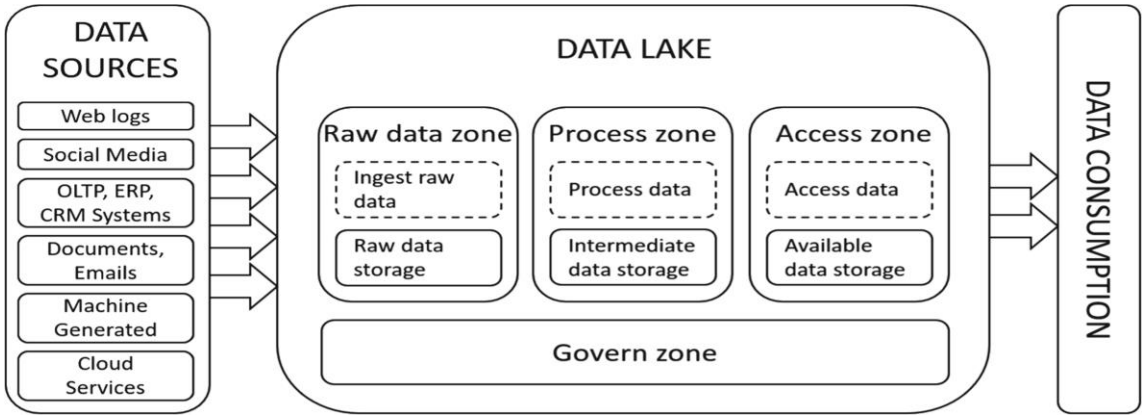

Figure 3-2 : *Architecture fonctionnelle de LD [42].*

- **Zone de données brutes :** tous types de données sont stockés sans traitement dans leurs formats natifs bruts.
- **Zone de traitement :** les données sont transformées selon les besoins des utilisateurs et stockées sous une forme intermédiaire.
- **Zone d'accès :** stocke toutes les données disponibles pour l'analytique.
- **Zone de gouvernance :** la gouvernance des données est appliquée pour toutes les autres zones, elle assure la sécurité, la qualité, le cycle de vie, l'accès aux données et aux métadonnées.

5. Défis des lacs de données

Comme toutes technologies émergentes, des défis se manifestent au fur et à mesure que l'écosystème est mis à l'épreuve, on cite les plus courants comme suit [54]:

- Un ensemble énorme de datasets à gérer et à organiser.
- Une nouvelle technologie en émergence qui nécessite un temps pour arriver au stade de maturité.
- Une collaboration entre les techniciens du domaine et les utilisateurs finaux est nécessaire pour le succès des projets basés sur LD.
- Le manque de structuration des données qui rend leur classification difficile.

- L'implémentation des projets LD est souvent basée sur Hadoop[25], qui évolue rapidement.
- Les données IoT sont parmi les sources qui doivent être gérées par les LD, ils posent ainsi un grand défi en matière de grande variété des protocoles et des connecteurs.
- La gouvernance de données
- Les problèmes de marécage de données (Data Swamp[26])[46] : nécessite une stratégie de gestion de métadonnées et de gouvernance robuste afin que le LD ne devient pas un marécage de donnée.
- La scalabilité d'extraction des métadonnées[27][2]

6. Gestion des métadonnées

Une définition banalisée du concept de métadonnée est que ce sont des données sur les données, ou plus précisément, ce sont des données structurées concernant un objet[28] et supportant des fonctions liées à celui-ci (apparue dans [57],[58]). Le rôle des métadonnées dans LD est primordial pour son fonctionnement[52], à savoir pour son existence.

6.1. Caractéristiques de base pour un système de métadonnées

Un système de métadonnées doit fournir un certain nombre de caractéristiques [56], on peut citer :

- L'enrichissement sémantique[29] : Rendre les datasets plus exploitables et intelligibles via la génération des informations de contexte, généralement réalisé par des bases de connaissances comme les ontologies.
- L'indexation des données : a pour but de garantir l'extraction efficace des données via des structures d'indexation.
- La conservation et génération des liens : intégration des datasets préexistants par exploration des relations et de similarités entre ceux-ci.
- Le polymorphisme des données : Possibilité de relier les données qui ont une représentation multiple à leurs données initiales afin d'éviter le retraitement.
- Le versionnement des données : Capacité de supporter le changement des données en gardant trace des états intermédiaires d'une part, de l'autre part, pour supporter

[25] https://hadoop.apache.org/
[26] Data Swamp : les données deviennent inutiles ou perdent de leurs valeurs pour des raisons variées [55].
[27] Selon[2] le LD Google contenait 20 milliards de datasets le moment ou l'article a été rédigé.
[28] En trouve dans la littérature des appellations équivalentes : unité de données, entités, ensemble de données[56].
[29] Appelé aussi annotation sémantique ou profilage sémantique[56].

l'évolution ramifiée[30] des données.

- Suivi d'utilisation : enregistre les interactions entre le LD et les utilisateurs.

6.2. Techniques de stockage des métadonnées

Les métadonnées sont elles-mêmes des données, elles doivent être persistées pour être exploitées et réutilisées, on présente dans le **tableau 3-2** les principaux techniques et méthodes de stockage [31]:

Technique	**Description**
Métadonnées intégrées	- Encapsulées dans l'objet qu'elles décrivent. - Utilisées pour les objets difficiles à catégoriser : images, vidéo …
Catalogues	- Stockent les métadonnées sous forme d'enregistrements. - Contiennent l'information pour la localisation des données. - Les enregistrements sont souvent organisés en catégories.
Registre	- Type spécial de catalogue qui stocke les métadonnées ayant des caractéristiques communes : même sujet, même propriétaire …etc.
Entrepôt	- A la différence des registres, on enregistre les données elles-mêmes ou une copie avec leurs métadonnées.
Stockage distribué	- Stockage distribué sur réseau fournissant de grande capacité, puissance de calcul et de fiabilité à moindre coût.
Stockage fédéré	- Propose un type de stockage distribué basé sur l'union de plusieurs systèmes de pairs indépendants.
Indexe	- Structure de donnée visant à améliorer la capacité des applications d'analyser, catégoriser, rechercher et extraire l'information numérique. - Se base sur des structures rapide et facile à manipuler. - Utilisé souvent par les moteurs de recherches et les bases de données.
Web Sémantique et ontologie	- Représentation basée sur des standards bien définis. - Stockage sous formes de triple RDF. - Fournit une interface de requête standardisée.

Tableau 3-2 : *Techniques et méthodes de stockage des métadonnées.*

[30] Permettre à une branche de données d'évoluer indépendamment sans mettre en cause l'intégrité des données initiale.

[31] Inspiré et synthétisé depuis [57]

6.3. Critères des métadonnées

Cerner ce que les métadonnées (MD) doivent inclure n'est en aucun cas une tâche facile. Plusieurs vues sont envisageables pour le même objet/dataset dépendant de l'utilisateur final. Néanmoins un minimum doit être garanti pour ne pas compromettre la qualité des métadonnées, nous citons quelques-uns [55]:

- Mode de création : manuelle ou automatique, crée par des experts ou non spécialiste.
- Moment de création.
- Mode de stockage.
- Niveau de structuration.
- But des métadonnées.
- Application : MD pour la découverte des ressources, MD pour l'utilisation des ressources.
- Niveau de standardisation.

6.4. Typologies des métadonnées

Il existe plusieurs types de métadonnées[56] :

- Métadonnées intra-objet : propriétés, sommaires et aperçus, métadonnée sémantique.
- Métadonnées inter-objet : groupage d'objets, liens de similarité, relations père/fils.
- Métadonnées globales : ressources sémantiques, indexes, logs.

7. Conclusion

Dans ce chapitre, nous avons abordé les concepts de LD et celle de métadonnée, leurs définitions et leurs caractéristiques.

Dans le chapitre suivant, nous aborderons l'un des formalismes les plus utilisés pour la représentation des métadonnées à savoir les techniques du web sémantique.

Chapitre 4 : Web Sémantique et ontologie

1. Introduction

Les techniques du web sémantique et en particulier les ontologies sont considérées comme des bonnes solutions pour la représentation des connaissances dans plusieurs domaines. Dans ce chapitre, nous allons mettre le point sur le but du Web Sémantique et les modèles de représentation des connaissances existants.

2. Web Sémantique et ontologie

Le Web Sémantique est une extension du web actuel où on donne à l'information un sens bien défini et permet aux humains et aux machines une meilleure coopération [57].

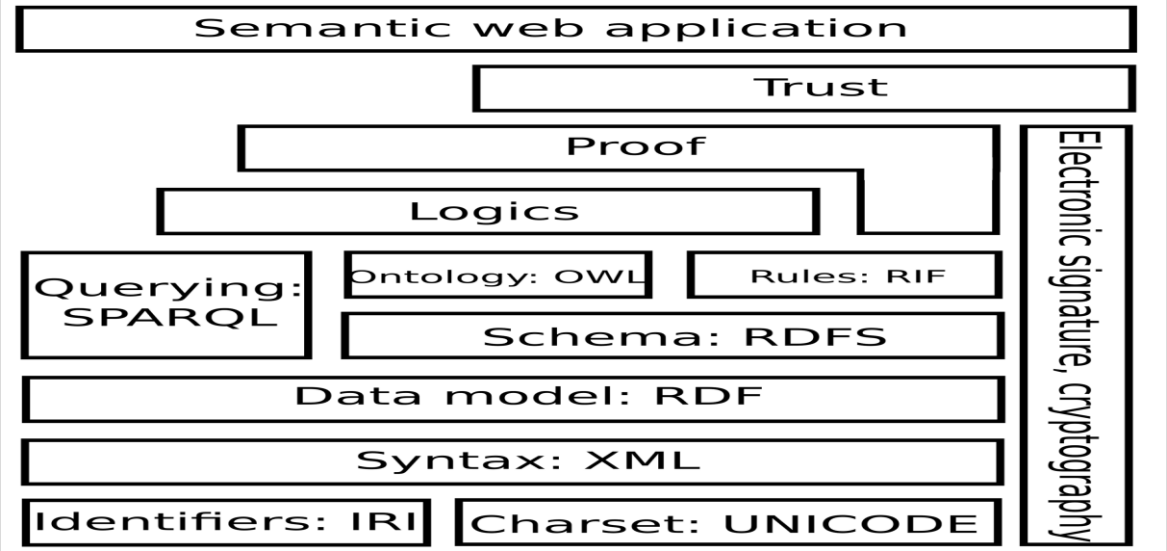

Figure 4-1 : *Pile Web Sémantique.*

Le but principal du Web Sémantique est l'automatisation du traitement des documents web afin de rendre leurs sens explicites.

Une ontologie est une spécification formelle explicite d'une conceptualisation partagée [60], elle est constituée de classes représentant des objets réels, ayant des propriétés et subissent des restrictions, et qui peuvent avoir des règles. Une ontologie ne représente qu'une définition formelle combinée avec des instances. Elles forment ce qu'on appelle une base de connaissances [59].

3. Représentation des connaissances du Web Sémantique

Afin de concrétiser et implémenter le concept du web sémantique notamment une ontologie, plusieurs syntaxes et langages sont apparus avec le temps. La recommandation du W3C[32] pour la représentation de ces concepts est appelé RDF[33], qui est essentiellement un langage de modélisation de données. Il représente les ressources et leurs relations sous forme d'un graphe dirigé [46]. Basé sur RDF, plusieurs syntaxes ont émergé, on peut citer ainsi [61] :

- RDF/XML.
- RDF / XML-Abbrev.
- RDFa.
- RDF / JSON.
- N3 et ses dérivés : N3-XX, N-Triple et son extension N-Quads, Turtle et ses extensions TriX et TriG.
- GRDD.

Pour la description formelle des concepts d'ontologies, on distingue trois familles de langages [62] :

- RDFS : Reconnu comme un langage d'ontologie mais peu expressif.
- OWL /OWL2[34]: famille de langages basés sur la logique de description, éprouve une grande expressivité, on distingue trois sous-langages[62] à expressivité croissante respectivement: OWL Lite, OWL DL, OWL Full[35].
- Langages de règles liés à la programmation logique.

4. Stockage des données RDF

Plusieurs techniques existent pour le stockage des données RDF. Elles sont classées, généralement, sous trois catégories in [65] :

- Stockage natif.
- Stockage basé sur bases de données relationnelles.
- Stockage basé sur le NoSQl.

5. Raisonnement et règles SWRL

Le raisonnement est un processus d'inférence qui permet de déduire des nouvelles informations

[32] https://www.w3.org/
[33] *Resource Description Framework*
[34] *Web Ontology Language*
[35] *OWL 2 Full est indécidable*[64]

depuis un modèle d'ontologie donné. Quelques tâches typiques de raisonnement sont : le contrôle de subsomption, vérification de consistances et classification d'instances [65].

Malgré l'expressivité qu'offre le formalisme OWL, il reste insuffisant pour la représentation des connaissances complexes qui ont besoin d'identifier ce qui se passe dans une situation évolutive. En d'autres termes, l'OWL manque d'expressivité déclarative qu'offrent les règles d'inférence.

En plus, le SWRL [36] est un langage de règles pour le web sémantique recommandé par W3C qui étend OWL pour la représentation déclarative des domaines complexes qui n'auront pas pu être représentés par OWL seul [67].

6. Conclusion

Dans les chapitres précédents, nous avons fait un survol sur le fondement théorique et les technologies qui nous aident à entamer notre conception et par conséquent influencer son implémentation.

Dans le chapitre suivant, nous allons présenter quelques travaux connexes.

[36] Semantic Web Rule Language

Chapitre 5 : Travaux connexes

1. Introduction

Dès que le concept du LD a vu le jour, l'importance d'avoir un mécanisme d'exploiter efficacement les données s'est manifestée rapidement, la gestion des métadonnées est dans le cœur de ce processus.

Dans la littérature, plusieurs chercheurs ont proposé des approches des fois ad hoc et d'autres fois plus au moins formelles ; basées sur des heuristiques dont la plupart se ressemblent conceptuellement mais se divergent sur des particularités imposées souvent par le domaine d'application.

Dans ce chapitre, nous allons citer quelques travaux connexes, nous nous somme limité à ceux qui sont récents et appliqués dans le contexte de LD.

2. Travaux connexes

- Dans [68], l'auteur propose de modéliser les MD par le biais de la technique *ensemble modeling* précisément *Data Vault* [37]qui, selon l'auteur, permet : (i) une évolution facile des schémas de données, (ii)des requêtes en temps acceptable et (iii) intégration des données non structurées et leurs extractions en un temps presque temps réel.

Cette solution se concentre sur l'aspect temps réel et temps d'exécution des requêtes, mais elle n'offre aucune facilité pour la découverte des datasets, aussi son formalisme de représentation est peu accessible aux experts de domaines.

- Dans [69], l'auteur propose un entrepôt des métadonnées, qu'il proclame universel, basé sur un graphe de connaissances et comporte les composants suivants : un moteur de sélection, un composant de résolution de conflits, un composant de renforcement de schéma, un autre pour l'analytique et un gestionnaire d'évènements.

Cette solution a pour but l'extraction efficace et expressive des métadonnées en combinant une vue technique et métier pour la représentation de ces dernières, cependant, elle n'offre aucune possibilité de découverte des datasets.

[37] Data Vault : définis au niveau relationnel et logique comme un ensemble lié de tables normalisées axées sur les détails et le suivi de l'historique et qui prennent en charge un ou plusieurs domaine fonctionnels [68].

- Dans [70], l'auteur propose une approche de découverte de métadonnées basée sur l'analyse exploratoire et échenillage de données. Il identifie pour chaque dataset une liste des colonnes potentielles, il extrait leurs caractéristiques -principalement leurs noms- et une liste des mots-clés. La première sera soumise à un calcul de similarité vis-à-vis des individus de la base de connaissances, la deuxième sera comparée avec une liste des synonymes des autres datasets. Pour les requêtes, il emploie les techniques d'analyse de texte -text mining-. La liste ainsi obtenue avec la liste des mots-clés est comparée avec la liste des caractéristiques et les métadonnées antérieures pour une éventuelle concordance que ce soit partielle ou totale. Par la suite, une étape d'analyse de données est exécutée, pour générer les métadonnées du dataset, celle-ci sera comparée avec les autres métadonnées par mesure de similarité.
Cette approche ne permet pas la classification des datasets, et son formalisme de représentation des métadonnées incite la répétition du processus pour chaque nouveau dataset.

- Dans [71], l'auteur propose une approche holistique de gestion de métadonnées basée sur le web sémantique, essentiellement pour identifier trois types de relation inter-dataset : datasets dupliqués, liés et aberrants. Elle procède par calcul de similarité entre datasets selon un seuil prédéfini, puis applique les techniques du schéma matching et alignement d'ontologies sur les datasets ainsi obtenus.
Cette approche étant une approche globale, elle ne permet pas l'identification au niveau d'attributs.

- Dans [72], l'auteur propose une méthode appliquée dans le domaine des neurosciences et qui suppose d'être applicable à d'autres domaines, basée sur des règles d'inférence qui sont fondées sur les expressions régulières. Ces dernières sont rajoutées aux racines des mots-clés pour augmenter la couverture transitive des mots qui comporte cette racine.
Cette approche est une approche syntaxique, permettant l'identification des libellées à partir d'une racine depuis lequel on peut générer d'autres libellées. Cependant elle ne tient pas compte de l'aspect sémantique et ne permet aucune classification des datasets.

- Dans [73], l'auteur propose un système qui prend en considération l'aspect structurel et sémantique formalisé sous forme d'ontologie, en se concentrant sur les caractéristiques structurelles des métadonnées. Il les modélise sous formes abstraites Data unit et data Unit Template où il identifie deux types de structures à savoir arbres et matrices.
Cette approche se concentre sur l'aspect structurel des métadonnées et n'offre aucune

classification des datasets.

- Dans [74], l'auteur propose un nouveau modèle de métadonnées représenté en réseau et dirigé par la sémantique, permettant selon l'auteur de définir la structure des données non structurées et l'extraction des vues thématiques des sources de données. Pour la structuration des données non structurées, il procède par extraction des mots-clés, depuis ceux-ci il dérive d'autres mots utilisant BabelNet[38]. Il calcule l'intersection de ces dérivations avec les mots-clés pour en déduire la liste de similarités lexicale. Celle-ci sera soumise à un calcul de similarité selon la métrique N-Gramme ainsi et au fur et aux mesures le réseau est construit.

- Dans [75], l'auteur propose un modèle adapté de [73] concentré principalement sur l'évolution structurelle des sources de données hétérogènes en rajoutant quelques concepts comme Type de vélocité et fréquence de changement. L'apport principal de cette solution est le suivi de l'évolution structurelle des datasets dans le temps. Elle n'offre cependant, aucun support pour la classification de ces derniers.

Dans le tableau suivant nous allons résumer les points clés des travaux cités ci-dessous :

Approche	Découvert	Données	Domaine	Modélisation	Concepts clés
[68]	?.	Semi structurée	Corpus d'héritage textiles	Relationnelle Graphe	-Evolution de schémas -Volume de stockage -temps de réponse
[69]	Auto	Relationnelle	Media Service de livraison Blanchiment d'argent Vérification de la confiance des données	Graphe de connaissances	- Résolution de conflits - Lignée de données -Renforcement de schémas

[38] Dictionnaire encyclopédique qui peut être utilisé comme réseau sémantique ou comme base de connaissances https://babelnet.org/

[70]	Semi-auto	-Semi-structurée	Sécurité routière	- ?	-Similarité des Data sources -Similarité d'attributs -Échantillonnage. -Base de connaissances -Text mining
[71]	Auto[39]	Semi-structurée	Véhicule Commerce Sport Santé	Rdf	-Alignement d'ontologies -Profile de données. -Schéma matching -Relations inter-datasets
[72]	Semi-auto	Textuels	Publications	ModelDb[40]	-Règles inférences -Expressions régulières
[73]	?.	Semi structurées	Scientifique	Ontologie	-Schema matching -Structure en arbre et matrice
[74]	Semi-auto	Structurées Semi structurées Non structurées	Environnement Climat	Réseau et graphe	-Structurer les données non structurées -Vues thématiques -BabelNet
[75]	Semi-auto	Structurées Semi structurées Non structurées	Publications	Relationnelle	-Evolutions des sources de données hétérogènes

[39] Automatique pour la découverte des relations inter-datasets.
[40] Corpus de publications dans le domaines des neurosciences [76]

Notre Solution	Semi-auto	Semi-structurée	IoT	Ontologie	-Signature de capteurs -Inférence à partir des abréviations, mots clés et synonymes. -Classification et apprentissage automatique. -Calcul de similarité. -Enrichissement et similarité sémantique.

Tableau 5-1 : *Comparaison de différentes approches de gestion des métadonnées.*

3. Conclusion

Dans ce chapitre, nous avons cité quelques travaux connexes afin d'avoir une vue globale et une meilleure compréhension du problème que nous souhaitons résoudre. Nous avons constaté que ces solutions n'offrent pas une classification adéquate des datasets.

Cependant l'intérêt le plus important des Big Data réside dans leurs exploitabilités par les scientifiques des données. Cette classification ne peut être pertinente sans avoir un formalisme robuste de représentation et un mécanisme efficace d'extraction des métadonnées.

Dans ce but, nous proposons une solution basée sur l'ontologie pour la représentation des métadonnées au sein d'un lac de données appliquées au domaine d'IoT.

Partie II : Conception et Implémentation

Chapitre 6 : Conception et Implémentation

1. Introduction

Dans ce chapitre, nous détaillons l'étape de conception, délimitons ses bornes et son cadre général, et nous fournissons via des diagrammes UML, une abstraction de notre démarche qui facilitera par la suite la compréhension de notre implémentation.
Nous allons décrire aussi les détails d'implémentation, les choix technologiques, l'environnement de développement, l'environnement opérationnel et donner quelques interfaces du système réalisé.

2. Conception

L'objectif principal de ce mémoire est de proposer un système qui vise à faciliter la gestion des métadonnées dans un LD, afin que ce dernier ne se transforme pas en un étang de données et perd en conséquence sa valeur.
Dans notre contexte, nous allons modéliser les métadonnées par les techniques du web sémantique sous forme d'ontologie. Vu leurs expressivités et large acceptation dans la communauté scientifique, outre les possibilités d'inférences et de raisonnements qu'offre une telle modélisation.

2.1. Ontologie proposée

L'objectif de l'ontologie proposée est de représenter un dataset via ses métadonnées. Elle permet de représenter les éléments d'un dataset et les différentes relations entre ceux-ci. Après avoir examiné et analysé plusieurs datasets du domaine publique[41], le tableau suivant donne une description de la liste des concepts de l'ontologie :

[41] https://www.kaggle.com/datasets

Class	Object property	Description	Range
DataSet	HasDomain	Indique que chaque dataset a un domaine : healthcare, weather …etc.	Domain
	IsSimilarTo	Propriété symétrique, pointe sur la liste des datasets similaires, aide dans le raisonnement pour identification et classification des nouveaux datasets.	DataSet
	HasAttributes	Ensemble d'attributs du dataset	Attribute
	hasKeyWords	Ensemble des mots-clés définissant le dataset, aidant dans la détection et la classification de ce dernier.	KeyWord
Domain	hasSubDomain	Propriété transitive qui permet une hiérarchisation des domaines afin de permettre une plus fine spécialisation.	Domain
	hasSensorsNomenclature	Pointe sur le minimum des capteurs qu'un domaine donné généralement inclut, participe au raisonnement pour détection et classification.	Sensor
Attribute	relatedToSensor	Pour les attributs qui représentent une mesure d'un capteur, participants dans le raisonnement.	Sensor
	hasName	Chaque attribut a un libellé	AttributeName
AttributeName	hasSynonymes	Propriétés symétrique et transitive, elle indique que chaque nom d'attributs a un ou plusieurs synonymes	AttributeName
	hasAbbreviation	Indique qu'un attribut peut être exprimé par une abréviation	Abbreviation

Tableau 6-1 : *Description des concepts de l'ontologie proposée.*

2.2. Création des règles SWRL

Afin de compléter l'ontologie proposée et pour plus d'expressivité, on a proposé un ensemble

des règles SWRL :

- ***R1 : si deux attributs ont les mêmes noms alors ils sont tous les deux reliés au même capteur :***

DataSetOntolgy:Attribute(?a1) ^ DataSetOntolgy:Attribute(?a2) ^ DataSetOntolgy:hasName(?a1, ?n1) ^ DataSetOntolgy:hasName(?a2, ?n2) ^ swrlb:equal(?n1, ?n2) ^ DataSetOntolgy:relatedToSensor(?a1, ?s1) ^ DataSetOntolgy:relatedToSensor(?a2, ?s2) -> sameAs(?a1, ?a2) ^ sameAs(?s1, ?s2).

- ***2: si deux attributs sont équivalents alors leurs synonymes sont interchangeables :***

DataSetOntolgy:hasSynonymes(?a1, ?s1) ^ DataSetOntolgy:hasSynonymes(?a2, ?s2) ^ DataSetOntolgy:Attribute(?a2) ^ DataSetOntolgy:Attribute(?a1) ^ attributeLabel(?s1,?L1)^ attributeLabel(?s2,?L2) ^swrlb:equal(?L1, ?L2)->sameAs(?a1,?a2) ^ DataSetOntolgy:hasSynonymes(?a2, ?s1) ^ DataSetOntolgy:hasSynonymes(?a1, ?s2).

- ***R3 : Si deux attributs sont reliés au même capteur alors ils sont équivalents :***

DataSetOntolgy: Attribute(?a1) ^ DataSetOntolgy:Attribute(?a2) ^ DataSetOntolgy:relatedToSensor(?a1, ?s1) ^ DataSetOntolgy:relatedToSensor(?a2, ?s1) -> sameAs(?a1, ?a2).

- ***R4: Si deux datasets ont les mêmes capteurs -nomenclature- donc ils peuvent être considérés du même domaine :***

DataSetOntolgy:DataSet(?ds1) ^ DataSetOntolgy:DataSet(?ds2) ^ DataSetOntolgy:hasDomain(?ds1, ?d1) ^ DataSetOntolgy:hasSensorsNomencalture(?d1, ?sn1) ^ DataSetOntolgy:hasSensorsNomencalture(?d2, ?sn2) ^ DataSetOntolgy:hasSensorsNomencalture(?ds2, ?sn1) -> DataSetOntolgy:hasDomain(?ds2, ?d1) .

- ***R5 : si l'intersection des synonymes de deux attributs n'est pas vide alors ces derniers sont équivalents :***

DataSetOntolgy:AttributeName(?a1) ^ DataSetOntolgy:AttributeName(?a2) ^ DataSetOntolgy:hasSynonymes(?a1, ?s1) ^ DataSetOntolgy:hasSynonymes(?a2, ?s2) ^ DataSetOntolgy:hasSynonymes(?a2, ?s1) -> sameAs(?a1, ?a2)

2.3.Système Proposé

Pour la gestion des métadonnées, nous décrivant dans la suite notre système basé sur la

représentation proposée. Le système permet aux utilisateurs de :

- Ajouter une métadonnée.
- Gérer les synonymes dans un dataset.
- Gérer les mots clés d'un dataset.
- Extraire les attributs d'un dataset.
- Inférer le domaine du dataset.
- Inférer un capteur.

Ces fonctionnalités sont organisées à travers des relations de dépendances dans le diagramme des cas d'utilisation (**Figure 6-1**).

2.3.1. Diagramme des cas d'utilisation

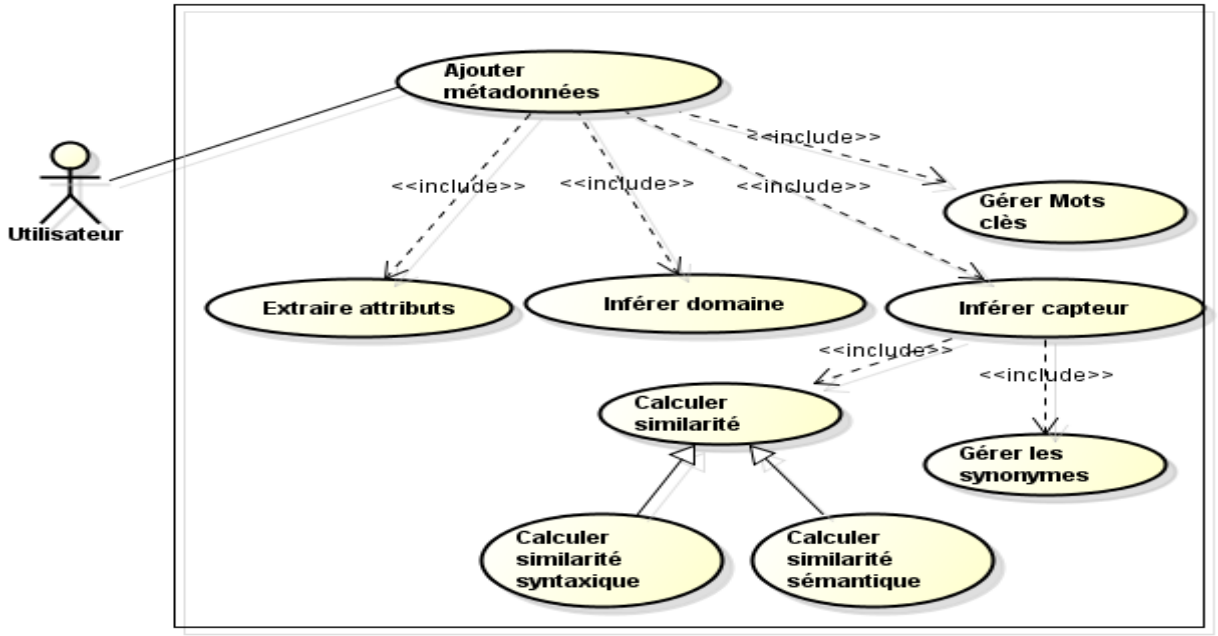

Figure 6-1 *: Diagramme des cas d'utilisation*

2.3.2. Description textuelle du cas d'utilisation

Titre	Ajouter métadonnées
Acteur	Utilisateur
But	Ajouter des métadonnées au dataset
Précondition	Authentification

Postcondition	La nouvelle métadonnée est ajoutée et prêt à être utiliser
Scénario nominal	1. L'utilisateur demande l'ajout de métadonnées 2. Il sélectionne le dataset sous forme brute 3. Le système analyse le dataset 4. Le système extrait les attributs 5. Le système infère le domaine du dataset 6. Le système crée la métadonnée 7. Le système sauvegarde la métadonnée
Scénario alternatif	3. a) Si le type de fichier du dataset n'est pas prise en charge alors déclencher **[Exception1 :formatNonPriseEnCharge]** 4. a. Si l'attribut est non identifié alors déclencher **[Exception2 :attributNonIdentifié]** b. L'utilisateur saisit les informations requises manuellement (y compris les synonymes et les abréviations). c. Reprise à l'étape 5. a. Si le domaine est non identifié alors déclencher **[Exception3 :domaineNonIdentifié]** b. L'utilisateur identifie le domaine du dataset c. Reprise à l'étape 6. 7. Si l'espace du disque est insuffisant alors déclencher **[Exception4 :disqueNonSuffisant]**

Exceptions	**[Exception1 :formatNonPriseEnCharge] :** Un message est affiché à l'écran avisant l'utilisateur que le format du dataset n'est pas prise en charge. **[Exception2 :attributNonIdentifié] :** Un message est affiché à l'écran avisant l'utilisateur que les attributs n'ont pas été identifiés. **[Exception3 :domaineNonIdentifié] :** Un message est affiché à l'écran avisant l'utilisateur que le domaine du dataset n'a pas été identifié **[Exception4 :disqueNonSuffisant] :** Un message est affiché à l'écran avisant l'utilisateur que le l'espace du disque est insuffisant pour la sauvegarde des métadonnées.

Tableau 6-2 : *Description textuelle du cas d'utilisation « Ajouter une métadonnée ».*

2.3.3 Diagramme de classes

Nous présentons dans cette section le diagramme de classes, qui représente le noyau de notre système. Il est illustré par les figures ***Figure 6-2*** et ***Figure 6-3***.

Le premier diagramme représente les objets métiers de notre prototype indépendant de tous détails d'implémentations.

Un objet *Dataset* est associé à un *Domaine* et décrit par des mots-clés, un objet *DataSetMetaData* a pour rôle de décrire les principales caractéristiques et les attributs du dataset.

Un attribut est soit un attribut simple ou composé, chacun a une unité de mesure *MesurementUnit* et est associé à un capteur : *sensor*. Un ensemble de capteurs constituent la *nomenclature*[42] d'un *domaine* identifié par la relation *hasSensorsNomenclature.*

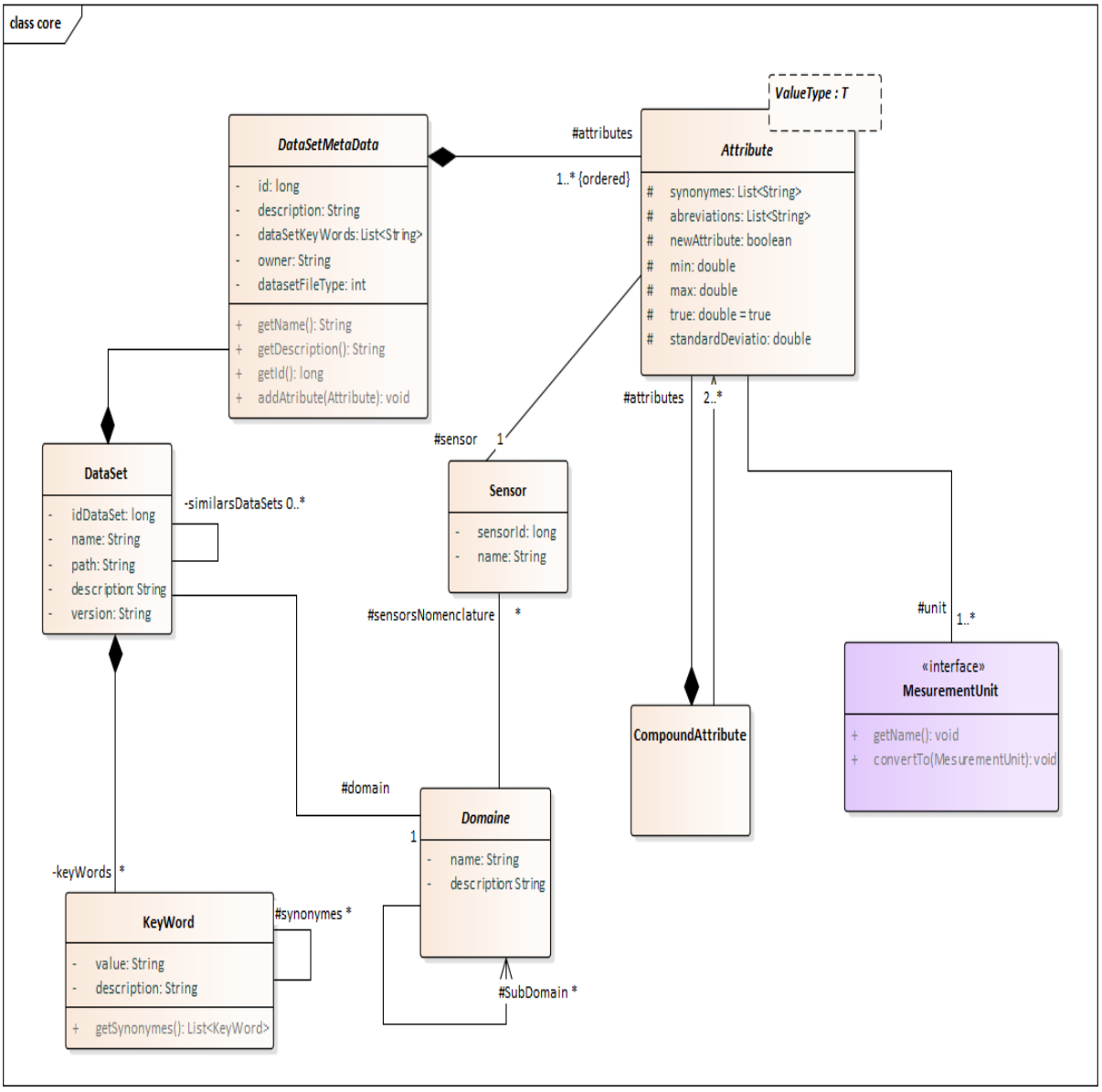

Figure 6-2. *Diagramme de classes métiers.*

[42] Genre de signature de capteurs dont leurs présences est lie à un domaine donné.

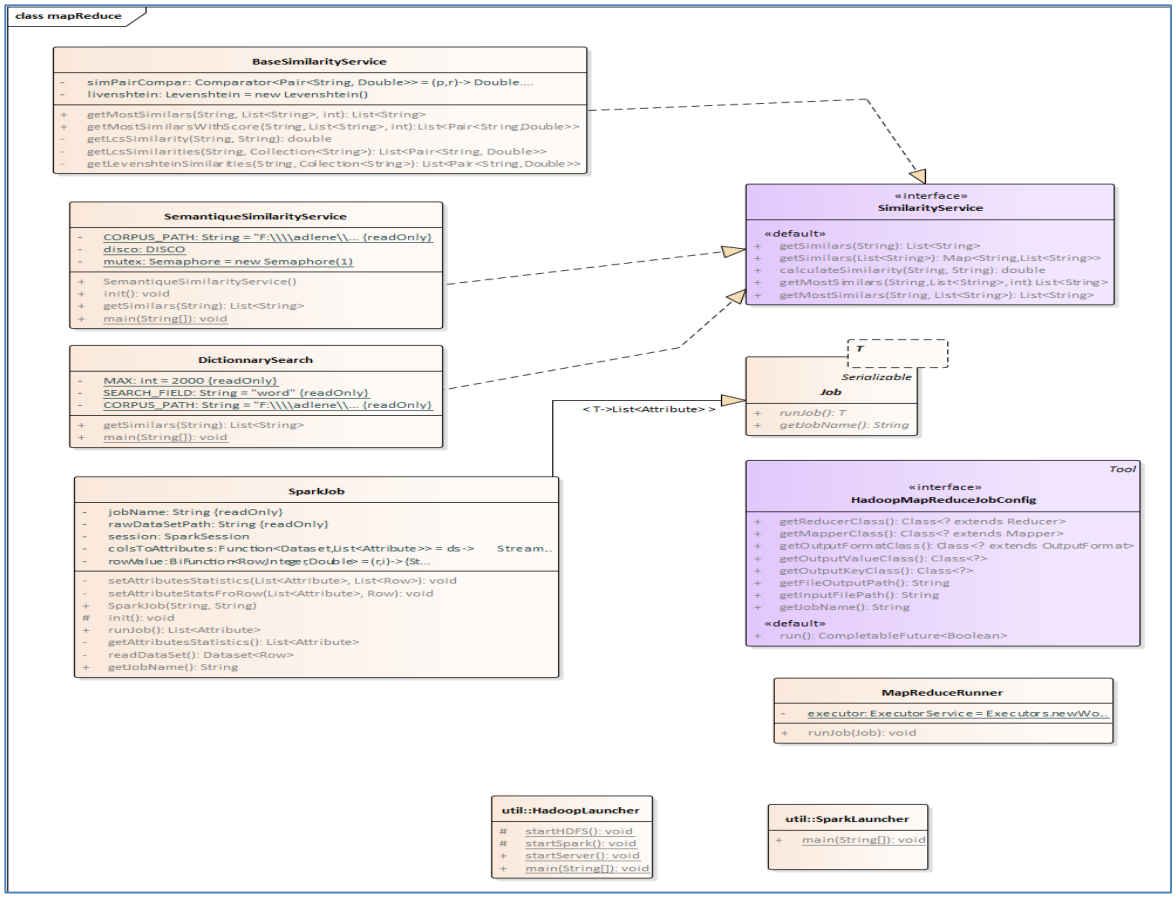

***Figure 6-3** : Diagramme de classes services*

3. Implémentation

3.1. Environnement de développement

Pour l'implémentation du système, nous avons choisi le langage java sous l'IDE Eclipse. Un choix qui, d'une part été basé sur la richesse et la disponibilité des librairies qui facilitent la procédure du traitement, et d'une autre part imposée par l'implémentation du LD qui est basée sur Hadoop.

Cependant il faut noter que la dernière version de Hadoop 3.2 et celle de Spark[43] ne supportent que la version 8 de java[44].

[43] https://spark.apache.org/docs/latest/index.html
[44] https://cwiki.apache.org/confluence/display/HADOOP/Hadoop+Java+Versions

3.2. Environnement Opérationnel

Pour le contexte de Big Data, nous avons utilisé, plus précisément, Spark sous Hadoop. La **figure 6-4** donne un aperçu sur l'écosystème Hadoop. Cependant, nous mentionnons que Spark est plus rapide que Hadoop[45].

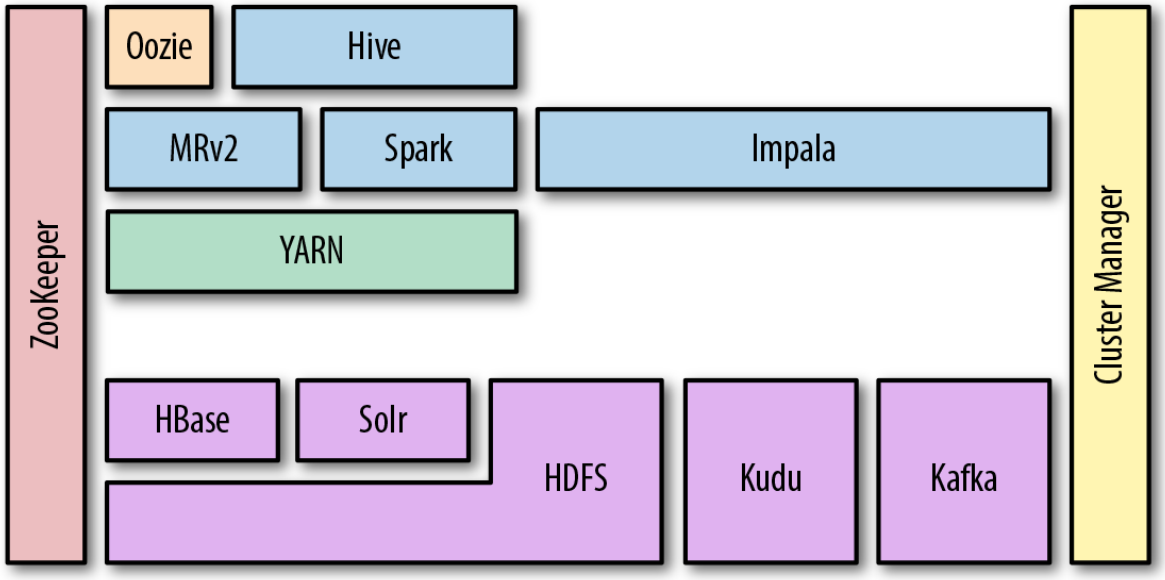

***Figure 6-4** : Pile technologique de l'écosystème Hadoop [83].*

3.3.Configuration matérielle

Pour la configuration matérielle, nous avons utilisé un ordinateur portable i5 de quatrième génération avec 8 Go de RAM vive sous Windows 10.

La configuration de l'environnement est faite par le biais de fichiers de propriétés. Ainsi, il est facile d'ajouter de clusters et de les enlever dynamiquement. Le **Tableau 6-3** montre la configuration d'Hadoop.

Environnement	**Configuration**	**Dépendances**	**Variables d'environnement**
Hadoop 2.7	hadoop-env.cmd core-site.xml	Java 8 winutils.exe[46]	HADOOP_HOME HADOOP_BIN

[45] Spark est 100 fois plus rapide que Hadoop [77]

[46] Obligatoire pour la distribution Windows, il faut le copier dans le répertoire /bin de Hadoop : https://github.com/steveloughran/winutils/blob/master/hadoop-2.7.1/bin/winutils.exe

	hdfs-site.xml mapred-site.xml		JAVA_HOME Path
Spark 2.4.5	docker. Properties fairscheduler.xml log4j.properties metrics. Properties slaves spark-defaults.conf Spark-env	JVM 8 scala[47] Hadoop	SPARK_HOME Path

Tableau 6-3 : *Configuration de l'environnement Hadoop.*

3.4.Diagramme de composants

La **figure 6-5** donne un aperçu global sur les différents composants de notre système.

- Nous avons utilisé la bibliothèque *DSICO*[48] et son corpus de données basé sur *Wikipédia* pour calculer la similarité sémantique. *DISCO* est interopérable avec *Protégé* via un Plugins[49].
- Pour calculer la similarité syntaxique, nous avons utilisé les mesures : LCS[50] et *levenshtein*.
- Pour tous ce qui est calcul des statistiques et apprentissage automatique nous avons utilisé la bibliothèque *Spark MLlib*[51].
- Pour le profilage des datasets, nous avons utilisé Spark Dataframe qui permet d'extraire les attributs avec leurs caractéristiques : min, max, écart type, …etc. Ces attributs nous permettent, par la suite, d'analyser la distribution statistique de ces derniers.

[47] https://www.scala-lang.org/ : langage de programmation base sur JVM.
[48] http://www.linguatools.de/disco/disco_en.html :extracting Distributionally related words using CO-occurrences
[49] http://www.linguatools.de/disco/disco4protege.html
[50] LCS : Longest common subsequence
[51] https://spark.apache.org/mllib/

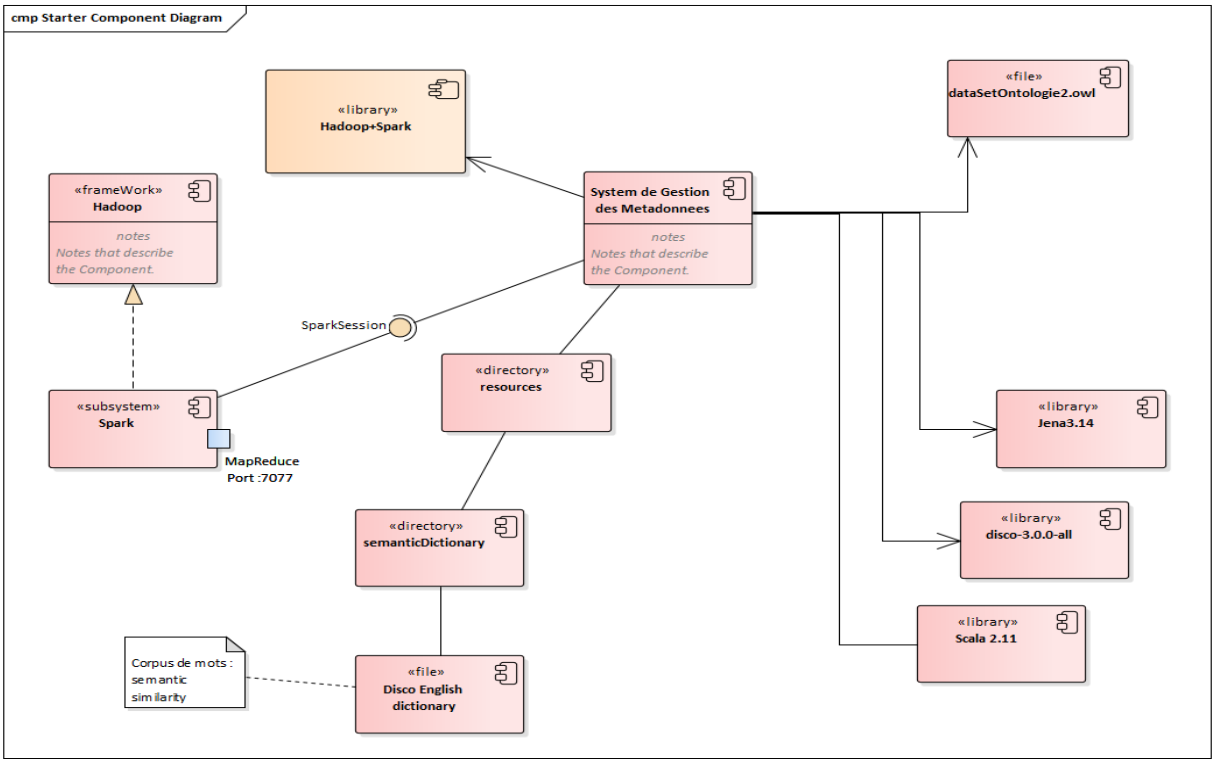

Figure 6-5 : *Diagramme de composants*

3.5. Intégration d'ontologie avec le système

Pour l'édition de l'ontologie et des règles SWRL, nous avons utilisé Protégé (**figure 6-6, figure 6-7)**

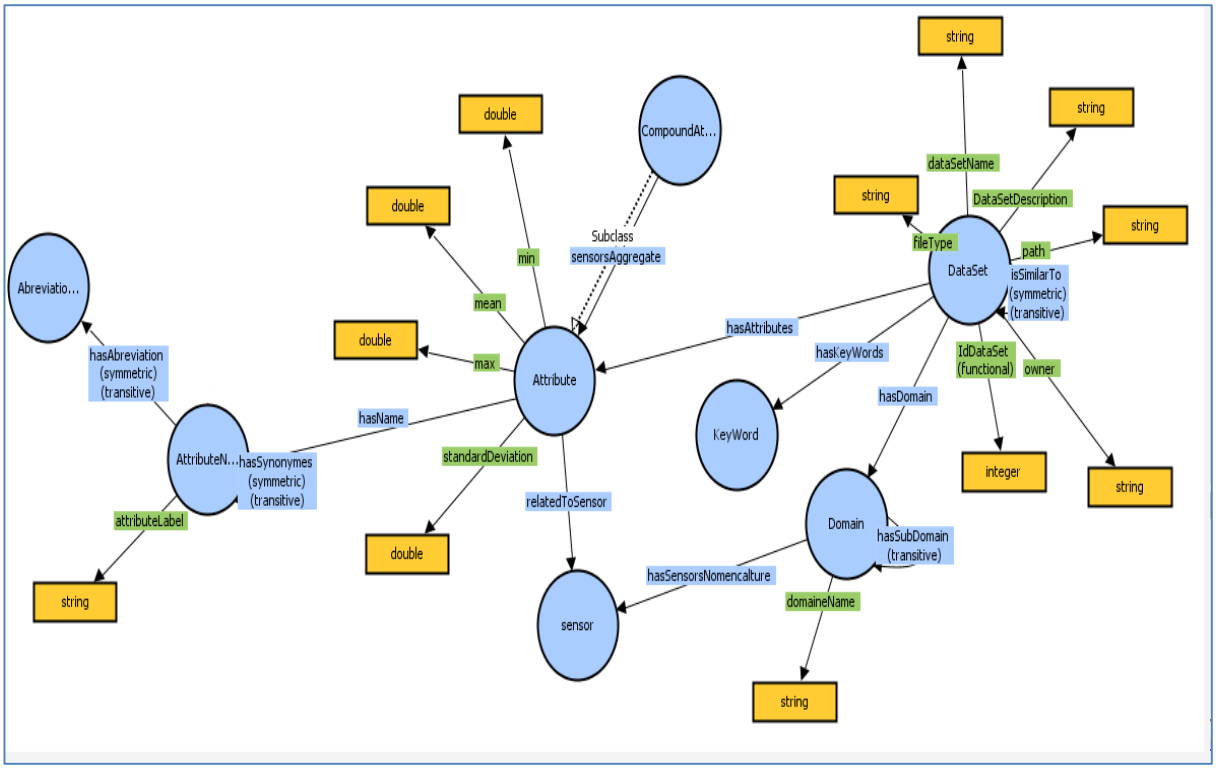

Figure 6-6 *: Ontologie modélisant les métadonnées d'un dataset [visualisée avec OWLviz]*

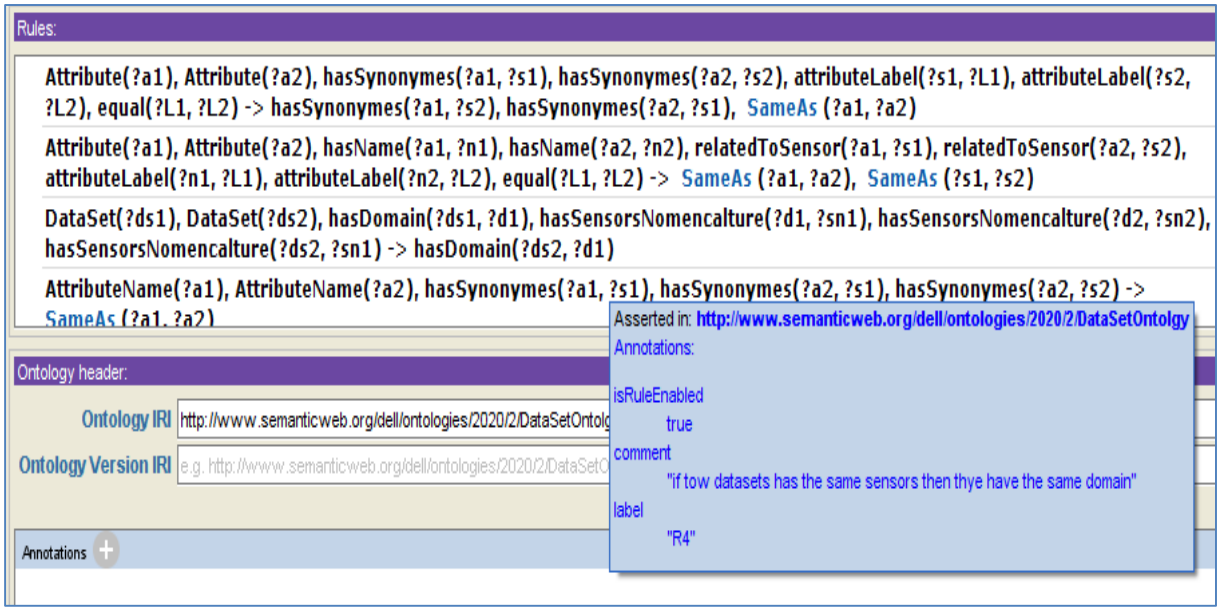

Figure 6-7 : *Edition des règles SWRL*

Pour l'intégration et la manipulation de l'ontologie, nous avons utilisé la bibliothèque Apache Jena[52]. La figure suivante (**Figure 6-8)** montre les classes qui participent à cette intégration.

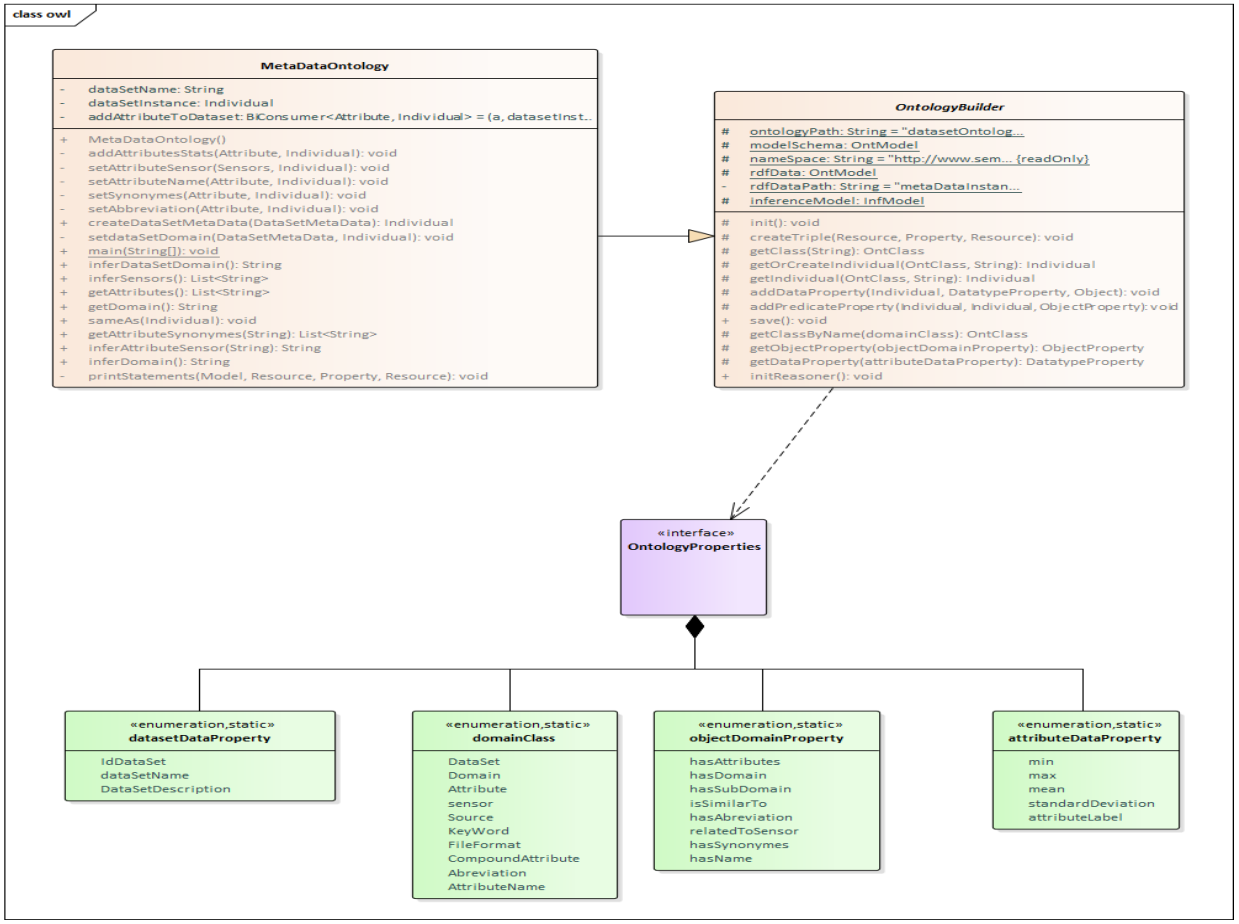

Figure 6-9 : *Classes d'intégration d'ontologie dans le système.*

L'intégration est garantie par la classe *OntolgyBuilder*, responsable de l'instanciation de l'ontologie et qui fournit les fonctionnalités de base pour sa manipulation. Elle dépend de l'interface *OntologyProperties* qui encapsule les propriétés spécifiques de l'implémentation de l'ontologie.

La classe *MetaDataOntology* étend les fonctionnalités de base de la classe *OntolgyBuilder* pour la gestion des métadonnées de l'ontologie.

La **figure 6-9** montre une partie des tests effectués pour assurer une intégration correcte de

[52] https://jena.apache.org/

l'ontologie dans notre système, pour aboutir ce but nous avons synthétisé un dataset fictif sur lequel nous avons testé le processus de raisonnement.

```
46    private Sensors temperatureSensor;
47    private Attribute tempInAttribute;
48    private Attribute tempOutAttribute;
49
50
52⊕    * @throws java.lang.Exception
55⊕   void setUp() throws Exception {
82
84⊕    * Test method for {@link donnee.owl.MetaDataOntology#createDataSetMetaData(metier.DataSetMetaData)}.
87⊕   void testCreateDataSetMetaData() {
92
94⊕   void testDataSetAttributes() {
106
107⊖  @Test
108   void testDataSetAttributeSynonymes() {
109       final List<String>syno1=metaOnto.getAttributeSynonymes( TEMPERATURE_IN_ATT);
110       final List<String>syno2=metaOnto.getAttributeSynonymes( TEMPERATURE_OUT_ATT);
111       assertEquals(NB_SYNONYMES, syno1.size());
112       assertEquals(NB_SYNONYMES, syno2.size());
113       assertTrue(tempInSynonymes.containsAll(syno1));
114       assertTrue(tempOutSynonymes.containsAll(syno2));
115
116   }
118⊕   void testDataSetAttributeEquivalenceBySynonymes() {
122
124⊕    * 1-infer from attributes
127⊕   void testInferDataSetDomain() {
137⊕   void testDataSetDomain() {
148
151⊕    * provide attribute with synonyme name
```

Figure 6-10 *: Une partie des tests unitaires*

3.6. Gestion de métadonnées

Le stockage de données est effectué au niveau de Hadoop sous leurs formes brutes, sans subir aucune modification, pareille pour notre base de connaissances. Une contrainte imposée par le concept du LD d'une part et de l'autre part par le système de gestion des fichiers de Hadoop « HDFS » afin de bénéficier des services offerts par ce dernier comme la réplication des données, et le traitement distribué via le paradigme MapReduce. Le processus d'extraction des métadonnées est découpé en tâches. Pour cela nous avons employé le patron de conception « *chaine de responsabilité* [77] », où la fin de l'exécution d'une tâche déclenche l'exécution de

la tâche suivante, la communication entre ces tâches est asynchrone découplée par le biais du modèle *Publication-Abonnement* [79].

3.7. Inférence du domaine

Pour l'inférence d'un domaine, nous nous somme basés sur la taxonomie des capteurs proposée dans [80]. La **figure 6-10** dévoile la conception du classifieur utilisé, où nous avons pour chaque capteur inscrit le nombre d'occurrences de celui-ci dans le dataset[53], pour la variable de classe nous avons choisi le domaine d'application.

AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	A
deformation	viscosity	flow	load	moisiture	shock	contact	strain	corrosion	electrical cor	oxygen	blood	organ	mental	tissue	domain	
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	agriculture	
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	logistic	
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	plant floor	
0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	transport	
1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	buildings	
0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	environement	
0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	healthcare monito	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	healthcare manage	

Figure 6-11 : *Extraction des caractéristiques d'apprentissage, basée sur la taxonomie*

Nous avons utilisé le classifieur *Forêt d'arbres décisionnels* [87] fourni par la bibliothèque *Spark MLlib* qui permet de minimiser le risque de sur-apprentissage par la combinaison de plusieurs arbres de décision [88].

3.8. Architecture système

Nous avons opté pour notre architecture une implémentation modulaire en couches (**Figure 6-11)**. Cette disposition en couches est basée sur le principe de séparation des concernes facilitant ainsi la maintenance et permet une évolutivité des couches sans influencer les autres. On distingue les couches suivantes :

- **La couche présentation ou d'interface graphique :** comporte toutes les entités d'interaction homme-machine. Similarité sémantique, similarité syntaxique,…etc. Elle est sollicitée par la couche présentation, et dépend des couches inférieures à savoir la couche métier.

[53] Outre les données issues de la taxonomie[80] nous avons analysé et utilisé les informations des datasets suivants :[81],[82],[83],[84],[85],[86].

- **La couche service :** offre les services de base de notre système tel que le calcul de similarité, correction d'orthographe…etc. Elle est sollicitée par la couche présentation, et dépend des couches inférieures à savoir la couche métier, pour le traitement d'ontologie nous avons dédié un module appart pour des raisons de maintenance et d'évolutivité.
- **La couche métier :** représente les objets du domaine d'application comme les datasets, les attributs…etc.
- **La couche infrastructure :** comporte le serveur Hadoop et Spark, communiquant sur leurs ports standards, ainsi les corpus de données nécessaires pour notre implémentation.
- **Couche utilité :** couche verticale dont l'utilisation peut être sollicitée par toutes les autres couches.

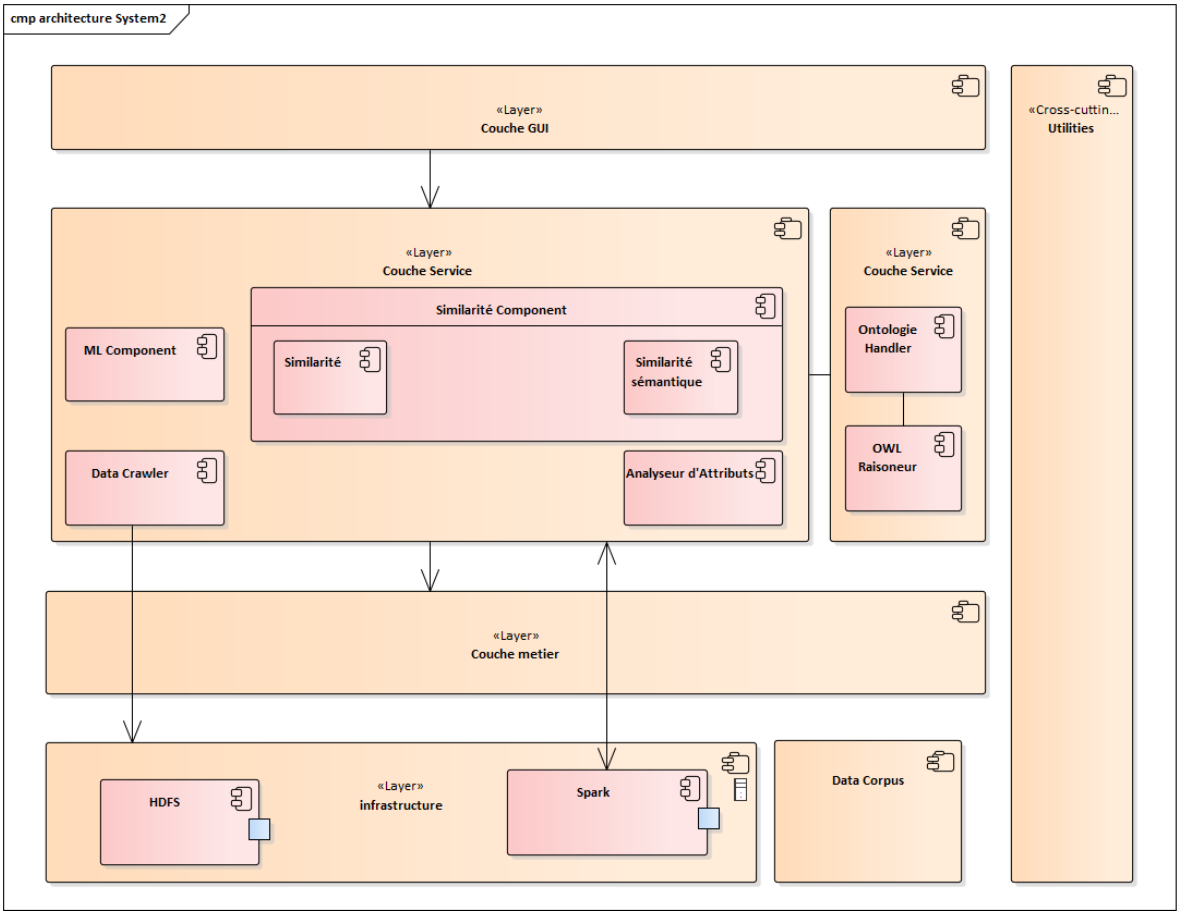

Figure 6-12 : *Architecture système en couches*

4. Expérimentation

Pour concrétiser notre solution, nous allons illustrer sous forme de captures d'écran un exemple de séquence de déroulement lié au cas d'utilisation ‘ *'Ajouter une métadonnée* ‘’ :

1. La **figure 6-12** montre la base de connaissances initiale.

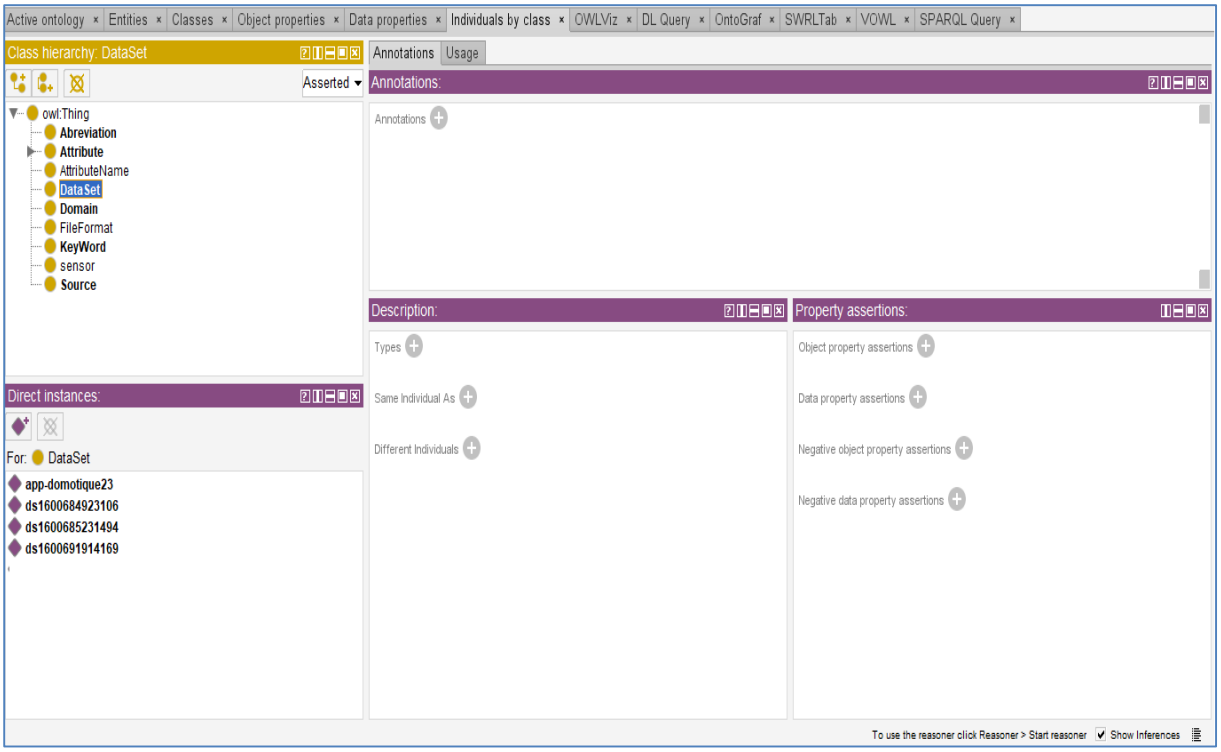

Figure 6-13 *: Base de connaissances initiale*

2. La **figure 6-13** donne un aperçu su la détection des capteurs.

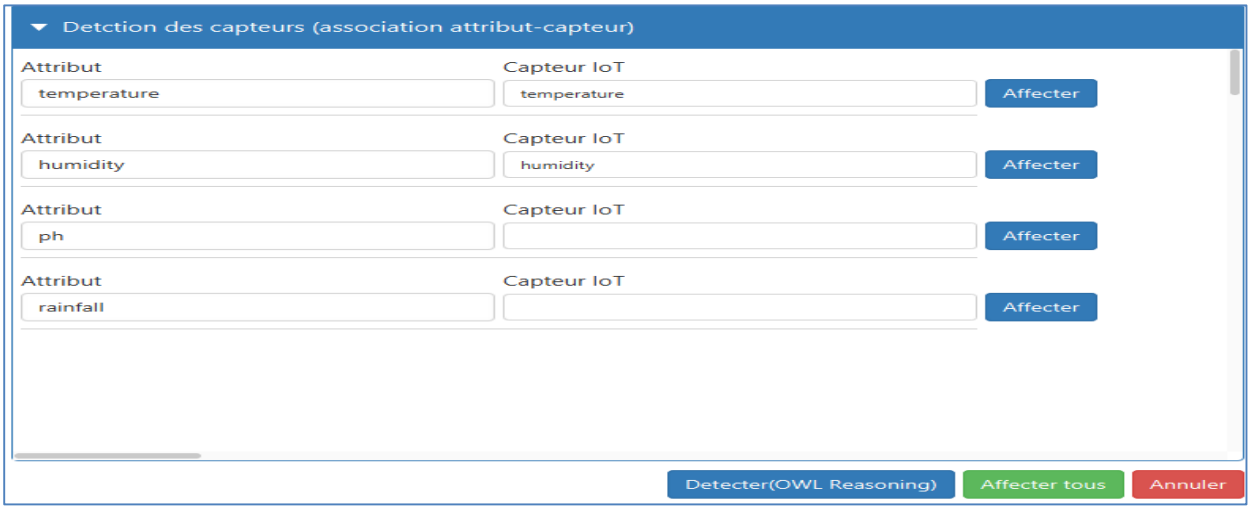

Figure 6-14 *: Aperçu sur la détection des capteurs*

3. La figure **6-14** montre l'étape de calcul de similarité.

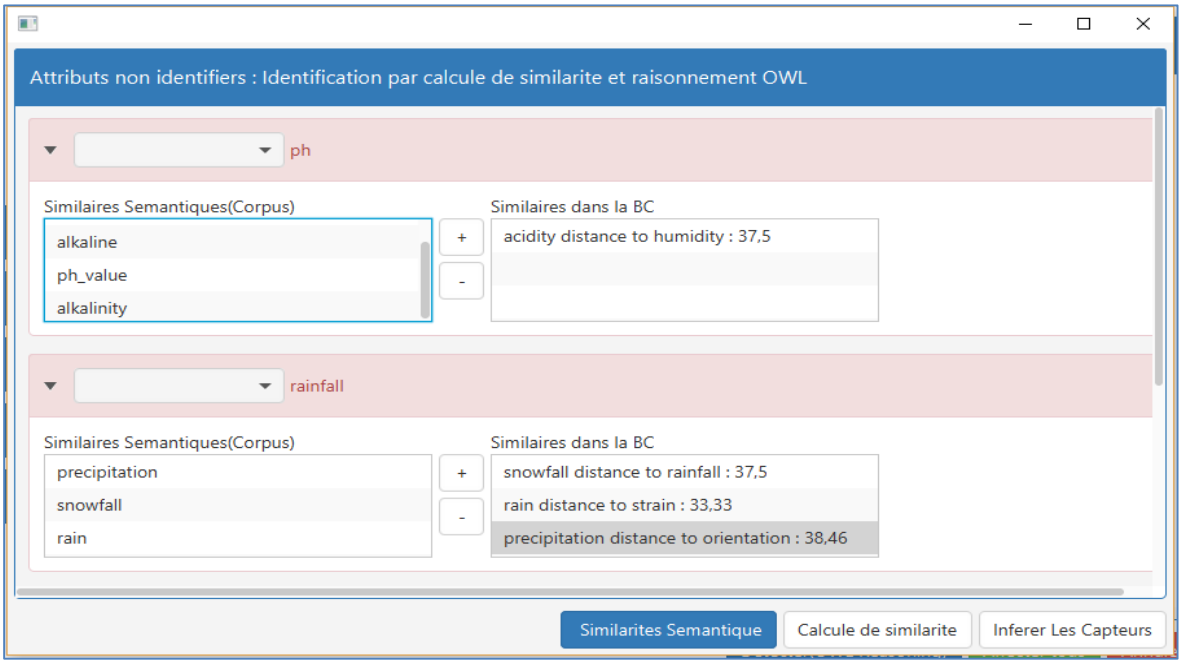

Figure 6-15 *: Fenetre de calcul de similarités*

4. La **figure 6-15** montre la base de connaissances après la création des métadonnées.

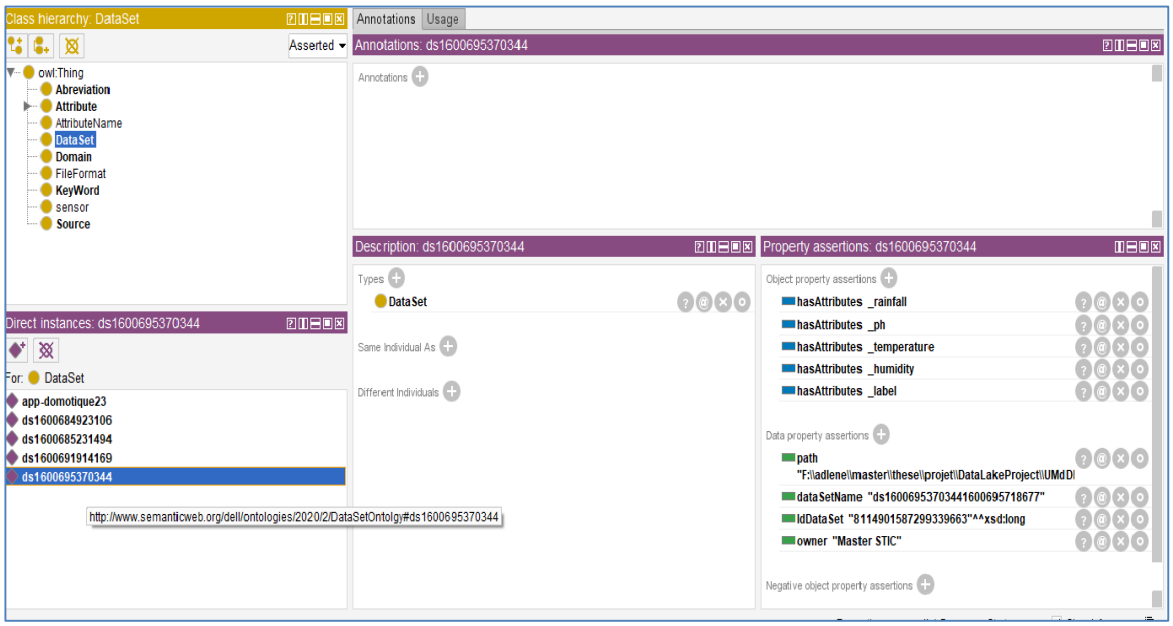

Figure 6-16 *: Base de connaissance après création de métadonnée*

5. Conclusion

Dans ce chapitre, nous avons présenté la conception et l'implémentation d'un système pour faciliter la création des métadonnées dans le contexte des LD. De plus nous avons parlé de l'environnement, des outils et des techniques qui ont forgé notre implémentation basée sur l'écosystème *Hadoop* et le langage Java.

Conclusion générale

Nous avons constaté le long de ce mémoire que dans le contexte des Big Data et des lacs de données, leurs réussites se reposent essentiellement sur leur capacité d'exposer les informations ingérées pour une éventuelle future exploitation. Nous avons vu aussi que, la gestion des métadonnées est dans le cœur de ce processus. Ainsi, nous avons remarqué que ces dernières peuvent être représentées d'une façon élégante et expressive par le biais des techniques du Web Sémantique. Donc, ceux-ci nous ont permet d'inférer les attributs, les capteurs et le domaine d'application à partir d'un raisonnement sur les synonymes d'attributs, les abréviations, les mots-clés et la signature des capteurs.

Pour que ce processus de raisonnement soit efficace, nous avons utilisé les mesures de similarités pour l'alignement des libellées des attributs, des mots clés, des capteurs et des abréviations avec la diversité de ceux-ci dans les datasets en pratique. Au fait, nous avons estimée qu'après un certain temps d'approvisionnement de la base de connaissances, notre système peut converger vers un système quasi-automatique. Sachons que les techniques du Web Sémantique sont des techniques symboliques [89]. Malgré leurs puissances et élégance de formalisation, elles restent limitées à ce que l'expert du domaine a conçu initialement, et ne peuvent en aucun cas, à notre connaissance, couvrir tous les types de datasets qui peuvent exister.

De ce fait, nous avons complémenté notre solution par une technique basée sur les données. Nous avons, également, utilisé l'apprentissage ensembliste qui offre un apprentissage incrémental adapté aux petites datasets[54], dont sa performance s'améliore au fils du temps.

Cependant nous reconnaissons que notre travail peut être amélioré par le perfectionnement des points suivants :

- L'introduction de la signature d'attributs au lieu de la signature de capteurs, car la présence de certains attributs peut balancer la classification des métadonnées d'un domaine à l'autre.
- Les données IoTs ne concernent pas seulement les capteurs, il faut introduire aussi tous les dispositifs de l'écosystème.
- L'utilisation d'un moyen de stockage *RDF* robuste, qui supporte le raisonnement

[54] https://spark.apache.org/docs/latest/mllib-ensembles.html

distribué, et qui garantit un passage à l'échelle adéquate.

- Expérimentation empirique du système en exploitant un grand nombre de datasets et identification des paramètres qui peuvent influencer ces performances et par conséquent les mesures de similarités appropriées.
- Exploiter les statistiques des attributs des datasets, par des techniques d'identification de similarités de population comme la technique Z-test [90], cela nécessite l'étude et l'analyse des distributions des valeurs des attributs IoT. Son implémentation peut être envisagée au niveau des règles SWRL ou au niveau applicatif.
- Implémentation des requêtes inter-datasets.
- Etendre le système pour gérer les flux de données et les datasets structurés.
- Etendre l'identification des attributs par le biais des techniques d'apprentissage automatique.
- Appliquer les techniques de réduction de dimensionnalité sur les attributs d'apprentissages afin de minimiser ces derniers dans le cadre du possible.

Références

[1] K. Jain et S. Mohapatra, « Taxonomy of Edge Computing: Challenges, Opportunities, and Data Reduction Methods », in *Edge Computing*, Springer Nature Switzerland AG 2019 51., F. Al-Turjman, Éd. Cham: Springer International Publishing, 2019, p. 51-69.

[2] A. Halevy *et al.*, « Managing Google's data lake: an overview of the GOODS system », *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, p. 10, 2016.

[3] « driving the digital agenda requires strategic architecture rosen idc ». https://idc-cema.com/dwn/SF_177701/driving_the_digital_agenda_requires_strategic_architecture_rosen_idc.pdf (consulté le avr. 22, 2020).

[4] « DC forecasts connected IoT devices to generate 79.4ZB of data in 2025 », juin 22, 2019. https://futureiot.tech/idc-forecasts-connected-iot-devices-to-generate-79-4zb-of-data-in-2025/ (consulté le avr. 22, 2020).

[5] S. Li, L. D. Xu, et S. Zhao, « The internet of things: a survey », *Inf Syst Front*, vol. 17, nº 2, p. 243-259, avr. 2015, doi: 10.1007/s10796-014-9492-7.

[6] K. K. Patel, S. M. Patel, et P. Scholar, « Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges », p. 10, 2016.

[7] J. Soldatos, « Building Blocks for IoT Analytics », River Publishers Alsbjergvej 10 9260 Gistrup Denmark, 2017, p. 1-294.

[8] S.-L. Peng, S. Pal, et L. Huang, Éd., *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm*, vol. 174. Cham: Springer International Publishing, 2020.

[9] S. Bansal et D. Kumar, « IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication », *Int J Wireless Inf Networks*, févr. 2020, doi: 10.1007/s10776-020-00483-7.

[10] M. Weyrich et C. Ebert, « Reference Architectures for the Internet of Things », *IEEE Softw.*, vol. 33, nº 1, p. 112-116, janv. 2016, doi: 10.1109/MS.2016.20.

[11] A. Torkaman et M. A. Seyyedi, « Analyzing IoT Reference Architecture Models », vol. 5, nº 8, p. 7, 2016.

[12] B. Di Martino, M. Rak, M. Ficco, A. Esposito, S. A. Maisto, et S. Nacchia, « Internet of things reference architectures, security and interoperability: A survey », *Internet of Things*, vol. 1-2, p. 99-112, sept. 2018, doi: 10.1016/j.iot.2018.08.008.

[13] S. Wagle et J. E. Pecero, « Efforts Towards IoT Technical Standardization », in *Ad-Hoc, Mobile, and Wireless Networks*, vol. 11803, M. R. Palattella, S. Scanzio, et S. Coleri Ergen, Éd. Cham: Springer International Publishing, 2019, p. 524-539.

[14] D. Aksu et M. A. Aydin, « A Survey of IoT Architectural Reference Models », in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, Istanbul, Turkey, mars 2019, p. 413-417, doi: 10.1109/SSD.2019.8893170.

[15] ISO, *Information technology – Internet of Things Reference Architecture (IoT RA)*. .

[16] J. Silva, J. Rodrigues, J. Al-Muhtadi, R. Rabêlo, et V. Furtado, « Management Platforms and Protocols for Internet of Things: A Survey », *Sensors*, vol. 19, nº 3, p. 676, févr. 2019, doi: 10.3390/s19030676.

[17] M. A. Jabraeil Jamali, B. Bahrami, A. Heidari, P. Allahverdizadeh, et F. Norouzi, « IoT Architecture », in *Towards the Internet of Things*, Cham: Springer International Publishing, 2020, p. 9-31.

[18] S. Bansal et D. Kumar, « IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication », *Int J Wireless Inf Networks*, févr. 2020, doi: 10.1007/s10776-020-00483-7.

[19] N. Zanoon, A. Al-Haj, et S. M. Khwaldeh, « Cloud Computing and Big Data is there a Relation between the Two: A Study », vol. 12, nº 17, p. 14, 2017.

[20] H. Cai, B. Xu, L. Jiang, et A. V. Vasilakos, « IoT-based Big Data Storage Systems in Cloud Computing: Perspectives and Challenges », *IEEE Internet Things J.*, p. 1-1, 2016, doi: 10.1109/JIOT.2016.2619369.

[21] C. Perera, A. Zaslavsky, P. Christen, et D. Georgakopoulos, « Context Aware Computing for The

Internet of Things: A Survey », *IEEE Commun. Surv. Tutorials*, vol. 16, nº 1, p. 414-454, 2014, doi: 10.1109/SURV.2013.042313.00197.
[22] F. Shi, Q. Li, T. Zhu, et H. Ning, « A Survey of Data Semantization in Internet of Things », *Sensors*, vol. 18, nº 2, p. 313, janv. 2018, doi: 10.3390/s18010313.
[23] M. Serrano et A. Gyrard, « A Review of Tools for IoT Semantics and Data Streaming Analytics », p. 26.
[24] « The ORM Foundation ». http://www.ormfoundation.org/ (consulté le févr. 10, 2020).
[25] D. Le-Phuoc, M. Dao-Tran, M.-D. Pham, P. Boncz, T. Eiter, et M. Fink, « Linked Stream Data Processing Engines: Facts and Figures », in *The Semantic Web – ISWC 2012*, vol. 7650, P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, et E. Blomqvist, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 300-312.
[26] P.-Y. Vandenbussche et B. Vatant, « Linked Open Vocabularies, un écosystème encore fragile », p. 13, 2012.
[27] Z. H., H. A., et M. M., « Internet of Things (IoT): Definitions, Challenges and Recent Research Directions », *IJCA*, vol. 128, nº 1, p. 37-47, oct. 2015, doi: 10.5120/ijca2015906430.
[28] M. Chen, S. Mao, et Y. Liu, « Big Data: A Survey », *Mobile Netw Appl*, vol. 19, nº 2, p. 171-209, avr. 2014, doi: 10.1007/s11036-013-0489-0.
[29] S. Okul, D. Aksu, et M. A. Aydin, « Applications of Deep Learning and Big Data Technologies », in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, Istanbul, Turkey, mars 2019, p. 550-553, doi: 10.1109/SSD.2019.8893261.
[30] A. Jaiswal, V. K. Dwivedi, et O. P. Yadav, « Big Data and its Analyzing Tools : A Perspective », in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, mars 2020, p. 560-565, doi: 10.1109/ICACCS48705.2020.9074222.
[31] Y. N. Malek *et al.*, « On the use of IoT and Big Data Technologies for Real-time Monitoring and Data Processing », *Procedia Computer Science*, vol. 113, p. 429-434, 2017, doi: 10.1016/j.procs.2017.08.281.
[32] J. W. Ansari, « Semantic Profiling in Data Lake », p. 81.
[33] Z. Albert Y et S. Sherif, *Handbook of big data technologies*. New York, NY: Springer Berlin Heidelberg, 2016.
[34] « Albert Y. Zomaya, Sherif Sakr - Handbook of Big Data Technologies-Springer (2017).pdf ». .
[35] K. Venkatram et M. A. Geetha, « Review on Big Data & Analytics – Concepts, Philosophy, Process and Applications », *Cybernetics and Information Technologies*, vol. 17, nº 2, p. 3-27, juin 2017, doi: 10.1515/cait-2017-0013.
[36] NIST Big Data Public Working Group, « NIST Big Data Interoperability Framework: volume 8, reference architecture interfaces », National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 1500-9, juin 2018. doi: 10.6028/NIST.SP.1500-9.
[37] S. M. Borodo, S. M. Shamsuddin, et S. Hasan, « Big Data Platforms and Techniques », *IJEECS*, vol. 1, nº 1, p. 191, janv. 2016, doi: 10.11591/ijeecs.v1.i1.pp191-200.
[38] NIST Big Data Public Working Group Reference Architecture Subgroup, « NIST Big Data Interoperability Framework: Volume 6, Reference Architecture », National Institute of Standards and Technology, NIST SP 1500-6, oct. 2015. doi: 10.6028/NIST.SP.1500-6.
[39] S. Sakr, *Big data 2.0 processing systems*, Springer. New York, NY: Springer Berlin Heidelberg, 2016.
[40] G. Rajesh, L. G. Paul, et K. S. Sandeep, *Formal Methods and Models for System Design*. SPRINGER, 2004.
[41] IBM, « Présentation du langage SPL (Streams Processing Language) », *Présentation du langage SPL (Streams Processing Language)*, janv. 02, 2020. https://www.ibm.com/support/knowledgecenter/fr/SSCRJU_4.2.1/com.ibm.streams.tutorials.doc/doc/tutorial-container.html (consulté le janv. 02, 2020).
[42] F. Ravat et Y. Zhao, « Data Lakes: Trends and Perspectives », in *Database and Expert Systems Applications*, vol. 11706, S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa, et I. Khalil, Éd. Cham: Springer International Publishing, 2019, p. 304-313.
[43] C. Ordonez, I.-Y. Song, G. Kotsis, A. M. Tjoa, et I. Khalil, *Big data analytics and knowledge*

discovery: 21st International Conference, DaWaK 2019, Linz, Austria, August 26-29, 2019, Proceedings. 2019.
[44] F. Ravat et Y. Zhao, « Data Lakes: Trends and Perspectives », in *Database and Expert Systems Applications*, vol. 11706, S. Hartmann, J. Küng, S. Chakravarthy, G. Anderst-Kotsis, A. M. Tjoa, et I. Khalil, Éd. Cham: Springer International Publishing, 2019, p. 304-313.
[45] H. Fang, « Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem, IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), Shenyang », *IEEE*, p. 820-824, 2015, doi: 10.1109/CYBER.2015.7288049.
[46] J. Waheed Ansari, « Semantic Profiling in Data Lake », Master thesis, RWTH Aachen University, RWTH Aachen University | 52056 Aachen | Germany, 2018.
[47] S. Divya Meena et M. Vidhya, « DATA LAKES-A NEW DATA REPOSITORY FOR BIG DATA ANALYTICS WORKLOADS », *ijarcs*, vol. 7, nº 5, oct. 2016.
[48] T. Chomo, « Deploying Data Lake for Big Data ,Master's Thesis,Masaryk University Faculty of Informatics,Czech », *Data Management*, p. 119, 2019.
[49] M. Feick, N. Kleer, et M. Kohn, « Fundamentals of Real-Time Data Processing Architectures Lambda and Kappa », p. 12.
[50] P. P. Khine et Z. S. Wang, « Data lake: a new ideology in big data era », *ITM Web Conf.*, vol. 17, p. 03025, 2018, doi: 10.1051/itmconf/20181703025.
[51] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, et P. C. Arocena, « Data lake management: challenges and opportunities », *Proc. VLDB Endow.*, vol. 12, nº 12, p. 1986-1989, août 2019, doi: 10.14778/3352063.3352116.
[52] C. Giebler, C. Gröger, E. Hoos, H. Schwarz, et B. Mitschang, « Leveraging the Data Lake: Current State and Challenges », in *Big Data Analytics and Knowledge Discovery*, vol. 11708, C. Ordonez, I.-Y. Song, G. Anderst-Kotsis, A. M. Tjoa, et I. Khalil, Éd. Cham: Springer International Publishing, 2019, p. 179-188.
[53] T. John et P. Misra, *Data Lake for enterprises: leveraging Lambda architecture for building Enterprise Data Lake*. Birmingham, UK: Packt Publishing, 2017.
[54] A. Panwar et V. Bhatnagar, « Data Lake Architecture: A New Repository for Data Engineer », *International Journal of Organizational and Collective Intelligence*, vol. 10, nº 1, p. 63-75, janv. 2020, doi: 10.4018/IJOCI.2020010104.
[55] R. K. Eichler, « Metadata Management in the Data Lake Architecture », p. 73.
[56] P. N. Sawadogo, É. Scholly, C. Favre, É. Ferey, S. Loudcher, et J. Darmont, « Metadata Systems for Data Lakes: Models and Features », in *New Trends in Databases and Information Systems*, vol. 1064, T. Welzer, J. Eder, V. Podgorelec, R. Wrembel, M. Ivanović, J. Gamper, M. Morzy, T. Tzouramanis, J. Darmont, et A. Kamišalić Latifić, Éd. Cham: Springer International Publishing, 2019, p. 440-451.
[57] M.-A. Sicilia, Éd., *Handbook of metadata, semantics and ontologies*. Singapore ; Hackensack, N.J: World Scientific, 2014.
[58] H. H. Alrehamy, « Extensible Metadata Management Framework for Personal Data Lake », p. 212.
[59] P. Li, « Semantic Reasoning on the Edge of Internet of Things », University of Oulu, Finland, 2016.
[60] E. Järvenpää, N. Siltala, O. Hylli, et M. Lanz, « The development of an ontology for describing the capabilities of manufacturing resources », *J Intell Manuf*, vol. 30, nº 2, p. 959-978, févr. 2019, doi: 10.1007/s10845-018-1427-6.
[61] mouad banane et A. Belangour, « A Comparative Study of RDF Triple Stores », *SSRN Journal*, févr. 2020, doi: 10.2139/ssrn.3349399.
[62] U. Straccia, « Fuzzy Semantic Web Languages and Beyond », in *Advances in Artificial Intelligence: From Theory to Practice*, vol. 10350, S. Benferhat, K. Tabia, et M. Ali, Éd. Cham: Springer International Publishing, 2017, p. 3-8.
[63] M. Botto-Tobar, M. Zambrano Vizuete, P. Torres-Carrión, S. Montes León, G. Pizarro Vásquez, et B. Durakovic, Éd., *Applied Technologies: First International Conference, ICAT 2019, Quito, Ecuador, December 3–5, 2019, Proceedings, Part I*, vol. 1193. Cham: Springer International Publishing, 2020.

[64] M. Lenzerini, L. Lepore, et A. Poggi, « Metaquerying made practical for OWL2QL ontologies », *Information Systems*, vol. 88, p. 101294, févr. 2020, doi: 10.1016/j.is.2018.02.012.
[65] M. Banane et A. Belangour, « A Survey on RDF Data Store Based on NoSQL Systems for the Semantic Web Applications », in *Advanced Intelligent Systems for Sustainable Development (AI2SD'2018)*, vol. 915, M. Ezziyyani, Éd. Cham: Springer International Publishing, 2019, p. 444-451.
[66] S. Biffl et M. Sabou, Éd., *Semantic Web Technologies for Intelligent Engineering Applications*. Cham: Springer International Publishing, 2016.
[67] A. Lawan et A. Rakib, « The Semantic Web Rule Language Expressiveness Extensions-A Survey », *arXiv:1903.11723 [cs]*, mars 2019, Consulté le: mai 01, 2020. [En ligne]. Disponible sur: http://arxiv.org/abs/1903.11723.
[68] I. D. Nogueira, M. Romdhane, et J. Darmont, « Modeling Data Lake Metadata with a Data Vault », in *Proceedings of the 22nd International Database Engineering & Applications Symposium on - IDEAS 2018*, Villa San Giovanni, Italy, 2018, p. 253-261, doi: 10.1145/3216122.3216130.
[69] N. Abolhassani *et al.*, « Universal Metadata Repository: Integrating Data Profiles Across an Organization », in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, Salt Lake City, UT, juill. 2018, p. 452-459, doi: 10.1109/IRI.2018.00072.
[70] H. Khalid, R. Wrembel, et E. Zimányi, « Metadata Discovery Using Data Sampling and Exploratory Data Analysis », in *Model and Data Engineering*, vol. 11815, K.-D. Schewe et N. K. Singh, Éd. Cham: Springer International Publishing, 2019, p. 106-120.
[71] A. Alserafi, A. Abello, O. Romero, et T. Calders, « Towards Information Profiling: Data Lake Content Metadata Management », in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, Barcelona, Spain, déc. 2016, p. 178-185, doi: 10.1109/ICDMW.2016.0033.
[72] R. A. McDougal, I. Dalal, T. M. Morse, et G. M. Shepherd, « Automated Metadata Suggestion During Repository Submission », *Neuroinform*, vol. 17, nº 3, p. 361-371, juill. 2019, doi: 10.1007/s12021-018-9403-z.
[73] C. Quix, R. Hai, et I. Vatov, « Metadata Extraction and Management in Data LakesWith GEMMS », *CSIMQ*, nº 9, p. 67-83, déc. 2016, doi: 10.7250/csimq.2016-9.04.
[74] C. Diamantini, P. L. Giudice, L. Musarella, D. Potena, E. Storti, et D. Ursino, « A New Metadata Model to Uniformly Handle Heterogeneous Data Lake Sources », in *New Trends in Databases and Information Systems*, vol. 909, A. Benczúr, B. Thalheim, T. Horváth, S. Chiusano, T. Cerquitelli, C. Sidló, et P. Z. Revesz, Éd. Cham: Springer International Publishing, 2018, p. 165-177.
[75] D. Solodovnikova, L. Niedrite, et A. Niedritis, « On Metadata Support for Integrating Evolving Heterogeneous Data Sources », in *New Trends in Databases and Information Systems*, vol. 1064, T. Welzer, J. Eder, V. Podgorelec, R. Wrembel, M. Ivanović, J. Gamper, M. Morzy, T. Tzouramanis, J. Darmont, et A. Kamišalić Latifić, Éd. Cham: Springer International Publishing, 2019, p. 378-390.
[76] « ModelDb Corpus de publications dans le domaine des neurosciences ». https://senselab.med.yale.edu/modeldb/ (consulté le avr. 24, 2019).
[77] A. Singh, A. Khamparia, et A. Kr. Luhach, « Performance comparison of Apache Hadoop and Apache Spark », in *Proceedings of the Third International Conference on Advanced Informatics for Computing Research - ICAICR '19*, Shimla, India, 2019, p. 1-5, doi: 10.1145/3339311.3339329.
[78] E. D. Lavieri, *Hands-on design patterns with Java: learn design patterns that enable the building of large-scale software architectures*. 2019.
[79] H. Washizaki, S. Ogata, A. Hazeyama, et E. B. Fernandez, « Landscape of Architecture and Design Patterns for IoT Systems », *IEEE INTERNET OF THINGS JOURNAL*, p. 12, 2020.
[80] V. Rozsa, M. Denisczwicz, M. Dutra, et P. Ghodous, « An Application Domain-Based Taxonomy for IoT Sensors », p. 11.
[81] A. Vergara, « Chemical gas sensor drift compensation using classifier ensembles », p. 10, 2012.
[82] Y. Zheng, L. Zhang, X. Xie, et W.-Y. Ma, « Mining Interesting Locations and Travel Sequences from GPS Trajectories », *Mobile Web*, p. 10, 2009.
[83] O. Walch, « Motion and heart rate from a wrist-worn wearable and labeled sleep from

polysomnography », *PhysioNet*, oct. 2019, doi: 10.13026/hmhs-py35.
[84] S. M. Bugenhagen, A. W. Cowley, et D. A. Beard, « Identifying physiological origins of baroreflex dysfunction in salt-sensitive hypertension in the Dahl SS rat », *Physiol Genomics*, vol. 42, p. 20.
[85] Z. Toth et J. Tamas, « Miskolc IIS Hybrid IPS: Dataset for Hybrid Indoor Positioning », p. 5.
[86] W. Ugulino, D. Cardador, K. Vega, E. Velloso, R. Milidiú, et H. Fuks, « Wearable Computing: Accelerometers' Data Classification of Body Postures and Movements », in *Advances in Artificial Intelligence - SBIA 2012*, vol. 7589, L. N. Barros, M. Finger, A. T. Pozo, G. A. Gimenénez-Lugo, et M. Castilho, Éd. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 52-61.
[87] A. B. Shaik, « A Brief Survey on Random Forest Ensembles in Classification Model », p. 8.
[88] « Ensembles - RDD-based API ». https://spark.apache.org/docs/latest/mllib-ensembles.html#gradient-boosted-trees-vs-random-forests (consulté le sept. 06, 2020).
[89] d'Amato Claudia, « Machine Learning for the Semantic Web: Lessons learnt and next research directions », *Semantic Web*, vol. 11, nº 1, p. 195-203, janv. 2020.
[90] T. Hailemeskel Abebe, « The Derivation and Choice of Appropriate Test Statistic (Z, t, F and Chi-Square Test) in Research Methodology », *ML*, vol. 5, nº 3, p. 33, 2019, doi: 10.11648/j.ml.20190503.11.