

Mémoire de fin d'étude

Pour l'obtention du diplôme d'ingénieur d'état en Informatique

Option : Systèmes d'Information et Technologies

Thème

**Exploitation des données et métadonnées du Linked
Open Data pour la construction d'un entrepôt de données
sémantique**

Réalisée par

- Aouimeur Yasmine

Encadré par

- Mme. Khouri Selma (ESI)

Promotion : 2018/2019

Dédicaces

*À mes chers parents, pour tous leurs sacrifices, leur amour, leur tendresse,
leur soutien et leurs prières tout au long de mes études,*

*À mes chères frères pour leurs encouragements permanents, et leur
soutien moral, pour leur appui et leur encouragement*

*À toute ma famille pour leur soutien tout au long de mon parcours
universitaire,*

*Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de
votre soutien infaillible,*

Merci d'être toujours là pour moi.

Remerciements

Premièrement et avant toute chose, je rends grâce à Allah, le tout puissant, de m'avoir permis de suivre le chemin du savoir, et donné le courage d'achever ce travail.

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon projet de fin d'étude et qui m'ont aidée lors de la rédaction de ce mémoire.

*Je voudrais dans un premier temps exprimer mes remerciements les plus profonds et ma profonde gratitude à **Mme Selma KHOURI**, Maître de Conférences à l'Ecole Nationale Supérieure d'Informatique, mon encadrante de mémoire et mon maître de stage, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.*

*Je tiens à remercier sincèrement **Mme. AIT ALI YAHIA Dahbia** qui est toujours disponible pour nous orienter.*

Enfin, je remercie l'ensemble du corps professionnel de l'Ecole Nationale Supérieure d'Informatique pour la qualité de leur enseignement et pour les valeurs qu'ils nous ont inculquées et tous ceux qui ont contribué de près ou de loin à la concrétisation de ce travail.

En espérant que ce modeste travail soit à la hauteur et reflète ce que j'ai avons pu acquérir pendant cette recherche.

Yasmine

Résumé

Les systèmes décisionnels de nos jours ont pris une énorme importance chez les entreprises. Ils représentent les méthodes, techniques et outils utilisés par les décideurs afin d'améliorer les différentes stratégies internes et externes de l'entreprise. Ils définissent principalement les systèmes de gestion de données qui sont indispensables à toute organisation afin d'analyser leurs données et en tirer profit.

Cependant, la croissance spectaculaire d'Internet et son adoption généralisée par les entreprises mondiales ont amplifié la quantité de données présentes sur le web. En effet, elle génère des milliards de données chaque année. Le défi est de gérer et exploiter efficacement ce nombre de données disponibles afin d'enrichir et de compléter les données internes des organisations. L'ultime but des décideurs, derrière l'emploi de cette démarche, est d'en tirer une valeur ajoutée pour améliorer les systèmes décisionnels de l'entreprise.

Ces dernières années avec l'évolution du web et l'apparition du web 3.0, le Web Sémantique et les LODs (Linked Open Data) deviennent une nouvelle source de données pertinentes pour les systèmes décisionnels particulièrement les ED (Entrepôts de données). Elle apporte de nouveaux axes d'analyse et enrichit ainsi le processus décisionnel.

Ce dernier se base sur deux systèmes primordiaux qui sont : le système source et le système cible. Le système source représente l'ensemble des sources de données internes et externes traitées par l'ED. On remarque cependant, que les sources externes de données sont très hétérogènes et distribuées, il y a donc une grande nécessité de faire une sélection de ces données par besoins des décideurs et par la fiabilité de leur provenance et leur validité grâce à une exploitation efficace de leurs métadonnées. Le système cible représente la base commune qui unifie les sources de données internes et externes, il représente dans notre cas l'ED de l'entreprise.

Notre travail s'inscrit dans la perspective d'apporter une nouvelle façon d'exploiter les données externes hétérogènes pour créer davantage de valeur aux systèmes décisionnels. Le projet vise donc à proposer un processus d'intégration des données et des métadonnées hétérogènes du LOD, tout en suivant l'approche conceptuel d'ED orienté besoins. Notre contribution permet d'une part de faire une sélection des données externes qui s'accordent aux besoins des décideurs ce qui permet d'une part, d'apporter une épuration stratégique des sources de données avant de passer par le processus d'intégration. D'autre part, nous assurons l'intégration des données afin de les unifier dans une source de données commune, et d'intégrer les métadonnées afin de permettre au concepteur d'ED de s'assurer de la fiabilité et la pertinence des sources de données qu'il utilise dans l'ED.

Mots clés :

Linked Open Data, Métadonnées, Entrepôt de Données Sémantique, Système d'intégration, Approche orienté besoins.

Abstract

Today's decision-making systems have become enormously important to businesses. They represent the methods, techniques and tools used by decision makers to improve the different internal and external strategies of the company. They primarily define the data management systems that are essential for any organization to analyze and benefit from their data.

However, the spectacular growth of the Internet and its widespread adoption by global companies have amplified the amount of data on the web. Indeed, it generates billions of data each year. The challenge is to manage and efficiently use this amount of data available to enrich and supplement the internal data of organizations. The ultimate goal of decision makers behind the use of this approach, is to derive added value to improve the decision-making systems of the company.

In recent years with the evolution of the web and the advent of Web 3.0, the Semantic Web and Linked Open Data (LODs) have become a new source of data that is relevant for decision-making systems, especially DWs (Data Warehouses). It brings new lines of analysis and thus enriches the decision-making process.

The latter is based on two primordial systems which are: the source system and the target system. The source system represents all the internal and external data sources processed by the DW. It should be noted, however, that external sources of data are very heterogeneous and distributed, so there is a great need to make a selection of these data according to the needs of the decision-makers and the reliability of their sources and their validity through an efficient exploitation of data and metadata.

Our work is part of a new way of harnessing heterogeneous external data to create more value for decision-making systems. The project aims to propose a process of integration of heterogeneous data and metadata of the LOD, while following the conceptual approach of ED oriented needs. Our contribution makes it possible, on the one hand, to make a selection of external data that matches the needs of the decision-makers. On the one hand, this allows for a strategic purification of the data sources before going through the integration process. On the other hand, we provide data integration to unify them into a common data source, and integrate the metadata to allow the ED designer to ensure the reliability and relevance of the sources of data, data it uses in the ED.

Keywords :

Linked Open Data, Metadata, Semantic Data Warehouse, Integration System, Needs Oriented Approach.

ملخص

أصبحت أنظمة صنع القرار اليوم مهمة للغاية للشركات. وهي تمثل الطرق والتكتيكات والأدوات المستخدمة من قبل صناع القرار لتحسين الاستراتيجيات الداخلية والخارجية المختلفة للشركة. إنها تحدد في المقام الأول أنظمة إدارة البيانات الضرورية لأي مؤسسة لتحليل بياناتها والاستفادة منها.

ومع ذلك، فإن النمو المذهل للإنترنت واعتمادها على نطاق واسع من قبل الشركات العالمية قد زاد من حجم البيانات على شبكة الإنترنت. في الواقع، فإنه يولد مليارات البيانات كل عام. يمكن التحدي في إدارة هذا الكم من البيانات المتاحة واستخدامه بكفاءة لإثراء واستكمال البيانات الداخلية للمنظمات. الهدف النهائي لصناعة القرار، وراء استخدام هذا النهج، هو استخلاص قيمة مضافة لتحسين أنظمة صنع القرار في الشركة.

في السنوات الأخيرة مع تطور الويب وظهور الويب 3.0 أصبحت الويب الدلالي والبيانات المفتوحة المرتبطة مصدرًا جديداً (مستودعات البيانات). إنه يجلب خطوطاً جديدة للتحليل، وبالتالي يُثري للبيانات ذات الصلة بنظم صنع القرار، وخاصة عملية صنع القرار.

يعتمد الأخير على نظامين أساسيين هما: نظام المصدر والنظام المستهدف. يمثل نظام المصدر جميع مصادر البيانات الداخلية والخارجية التي يتم معالجتها بواسطة مستودعات البيانات. ومع ذلك، تجدر الإشارة إلى أن المصادر الخارجية للبيانات غير متجانسة للغاية وموزعة، لذلك هناك حاجة كبيرة لاختيار هذه البيانات وفقاً لاحتياجات صناع القرار وموثوقية مصدرها وصلاحتها من خلال استغلال فعال للبيانات والبيانات الوصفية الخاصة بهم.

يمثل علينا جزءاً من طريقة جديدة لتسخير البيانات الخارجية غير المتجانسة لخلق قيمة أكبر لأنظمة صنع القرار. والبيانات الوصفية للبيانات المفتوحة، تتبع مساعمتنا، من ناحية، إمكانية اختيار مجموعة من البيانات الخارجية التي تلبى احتياجات صناع القرار، فمن ناحية، يسمح ذلك بالتطهير الاستراتيجي لمصادر البيانات قبل الدخول في عملية التكامل. من ناحية أخرى، نحن نقدم تكامل البيانات لتوحيدها في مصدر بيانات مشترك، ودمج البيانات الوصفية للسماح لمصمم البيانات الإلكترونية بضمان موثوقية مصادر البيانات وأهميتها.

الكلمات الأساسية:

البيانات المفتوحة، وصف البيانات، مستودع البيانات الدلالي، نظام التكامل، المشكلة الموجه نحو الاحتياجات.

Table des matières

Dédicaces	i
Remerciements.....	ii
Résumé.....	iii
Abstract.....	iv
ملخص	v
Table des matières	vi
Table des figures.....	xii
Liste des tableaux.....	xiii
Liste des abréviations.....	xiv
Introduction Générale	1
Partie 1 : Etat de l'art	4
Chapitre 1 : Entrepôt de données (ED)	5
Introduction.....	5
1 Historique	5
2 Définitions	6
2.1 Entrepôt de données (Datawarehouse)	6
2.2 Magasin de données (Datamart).....	7
2.3 Modélisation multidimensionnel	7
2.3.1 Fait	8
2.3.2 Dimensions	8
3 Objectifs d'un ED.....	8
3.1 Lacunes des systèmes traditionnels d'aide à la décision (SIAD)	8
3.1.1 Définition :	8
3.1.2 Lacunes des SIAD traditionnels :	9
3.2 Objectifs des ED pour remédier aux lacunes des SIAD traditionnels	10
3.2.1 Objectifs à court terme.....	10
3.2.2 Objectifs à long terme	11
4 Exploitation des ED pour l'analyse décisionnel	11
5 Architecture d'un ED	12
5.1 Systèmes sources opérationnels (Operational Source Systems).....	13
5.2 Zone de mise à disposition des données (Data Staging Area).....	13
5.3 Zone de présentation des données (Data Presentation Area).....	14
5.4 Outils d'accès aux données (Data Access Tools).....	14
6 Conception d'un ED	14

6.1	Méthodes de conception d'un ED	14
6.1.1	Conception Top-Down.....	15
6.1.2	Conception Bottom-Up.....	16
6.1.3	Conception Hybride	17
6.1.4	Comment bien choisir ?	18
6.2	Cycle de conception d'un ED.....	18
6.2.1	Identification des besoins.....	19
6.2.2	Modélisation conceptuelle	20
6.2.3	Modélisation logique	20
6.2.4	Processus ETL (Extract, Transform, Load)	20
6.2.5	Modélisation physique	21
7	Métadonnées des ED	21
7.1	Généralités sur les métadonnées.....	21
7.1.1	Définition générale :	21
7.1.2	Les métadonnées des ED :	21
7.2	Typologies des métadonnées dans les ED.....	22
8	Conclusion.....	23
	Chapitre 2: Web sémantique et Linked Open Data (LOD).....	24
	Introduction.....	24
1	Web sémantique	24
1.1	Historique et évolution du WEB	24
1.1.1	Web 1.0.....	24
1.1.2	Web 2.0.....	24
1.1.3	Web 3.0.....	24
1.1.4	Web 4.0.....	25
1.2	Définitions	25
1.2.1	Le World Wide Web Consortium (W3C)	25
1.2.2	Web Sémantique	25
1.3	Standards et langages du Web Sémantique	26
1.3.1	Langages existants sur lesquels le web sémantique s'est basé	26
1.3.2	Les couches du Web Sémantique.....	27
1.3.3	RDF (Resource Description Framework)	28
1.3.4	RDFS (Resource Description Framework Schéma)	30
1.3.5	OWL	31
1.3.6	Langage d'interrogation des données sémantiques (SPARQL).....	33
	Conclusion	34
1.4	Les ontologies.....	34

1.4.1	Définitions.....	34
1.4.2	Les notions clés dans une ontologie.....	35
1.4.3	Taxonomies des ontologies.....	36
1.4.4	Outils de construction d'ontologies	37
	Conclusion	38
2	Linked Open Data (LOD).....	39
2.1	Définitions	39
2.2	Evolution des LOD.....	39
2.3	Principes du LOD	41
2.4	Les portails LOD	42
	Conclusion	42
3	Métadonnées du LOD.....	43
3.1	Définitions	43
3.2	Typologies des métadonnées du LOD.....	43
3.2.1	Selon les types.....	43
3.2.2	Selon les normes	44
3.3	Les niveaux de définition des métadonnées du LOD	44
3.3.1	Définition au niveau sémantique.....	44
3.3.2	Définition au niveau schéma.....	45
	Conclusion	49
	Chapitre 3 : Entrepôt de données sémantique	50
	Introduction.....	50
1	Les entrepôts de données à base ontologique	50
1.1	Définition.....	50
1.2	Intérêt des ontologies pour les ED.....	51
2	Les systèmes d'intégration à base ontologique	51
2.1	Structure à base d'une ontologie unique.....	51
2.2	Structure à base d'ontologies multiples.....	51
2.3	Structure à base d'ontologies partagées.....	51
	Conclusion	52
	Partie 2 : Conception de la solution	52
	Chapitre 1 : Contexte et présentation du projet.....	53
	Introduction.....	53
1	Contexte du projet	53
2	Problématiques	53
3	Objectifs	54
	Chapitre 2 : Etude de l'existant.....	55

1	Présentation de l'approche orientée besoins.....	55
1.1	Les entrées de l'approche	56
1.2	Le processus de traitement	56
1.3	Le résultat de l'approche	57
2	Outil existant	58
2.1	Fonctionnalités et interface de l'outil existant.....	58
2.2	Les problèmes rencontrés :	58
2.3	Pistes à compléter et vision de la solution.....	59
2.3.1	Ajout de la couche intégration des données et des métadonnées	59
2.3.2	Refonte de l'outil existant et implémentation de l'approche orientée besoins ...	59
	Conclusion	60
	Chapitre 3 : Conception	61
	Introduction.....	61
1	Approche et démarche de conception.....	61
2	Expression et analyse des besoins	62
2.1	Identification des acteurs	62
2.2	Identification des spécifications fonctionnelles.....	63
2.2.1	Spécifications relatives à la gestion des ontologies internes et externes.....	63
2.2.2	Spécifications relatives à la gestion des besoins	63
2.2.3	Spécifications relatives à a gestion des mappings.....	63
2.2.4	Spécifications relative à la gestion du processus d'intégration.....	64
2.2.5	Spécifications relatives à la gestion du tableau de bord.....	65
2.3	Identification des spécifications non fonctionnels.....	66
3	Fonctionnement générale de la solution	66
4	Architecture globale du système	68
4.1	Couche client.....	71
4.1.1	Modélisation des fonctionnalités du système.....	71
4.1.2	Modélisation des interfaces graphiques	78
4.2	Couche persistance	81
4.2.1	Modèle de données général.....	81
4.2.2	Schémas des sources	83
4.2.3	Schémas de la cible.....	87
4.2.4	Schéma des mappings	89
4.2.5	Schéma des besoins.....	91
4.3	Couche traitement.....	92
4.3.1	Module de la gestion des sources.....	92
4.3.2	Module de gestion des mappings	93

4.3.3	Module de gestion des besoins.....	95
4.3.4	Module du processus d'intégration de données et de métadonnées.....	96
4.3.5	Module de gestion du tableau de bord	104
4.4	Couche service	104
4.5	Diagramme de classes général.....	105
	Conclusion	106
	Partie 3 : Réalisation	107
	Chapitre 1 : Réalisation du système.....	108
	Introduction.....	108
1	Langages, outils et autres technologies utilisées	108
1.1	Langages et formats de données.....	108
1.2	Outils	109
1.3	Framework et API	109
2	Implémentation du système	111
2.1	Couche persistance	111
2.1.1	Manipulation des sources.....	111
2.1.2	Création du méta-schéma cible	113
2.2	Couche traitement.....	114
2.2.1	Module de gestion des sources.....	114
2.2.2	Module de gestion des besoins.....	114
2.2.3	Module de gestion des mappings	114
2.2.4	Module du processus d'intégration	115
2.2.5	Module de gestion du tableau de bord	115
2.3	Couche Client	116
2.3.1	Interface de la gestion des sources.....	116
2.3.2	Interface de la gestion des mappings	116
2.3.3	Interface de la gestion des besoins	117
2.3.4	Interface de la gestion du processus ETL	117
2.3.5	Interface du tableau de bord.....	119
	Conclusion	120
	Chapitre 2 : Expérimentation et évaluation du système.....	121
	Introduction.....	121
1	Démarche et scénario de test	121
2	Statistiques et métriques d'évaluation	122
2.1	Volume global des sources internes et externes	122
2.2	Métriques de description	122
2.2.1	Ressources internes et externes.....	122

2.2.2	Instances inférées et insérées	123
2.2.3	Description des Métadonnées	123
2.3	Métriques de qualité	124
2.3.1	Cohérence des données:	124
2.3.2	Performance :	124
2.3.3	Audibilité :	125
	Conclusion	126
	Conclusion	126
	Partie 4 : Gestion de projet.....	127
1	Encadrement du projet.....	128
2	Présentation des prototypes	128
3	Planning du projet.....	129
4	Principales phases du projet	129
	Conclusion générale.....	131
	Bibliographie	133

Table des figures

Figure 1 : Schéma en étoile (Fait et dimensions).	8
Figure 2 : Les éléments basic d'un ED.....	13
Figure 3 : L'approche descendante par Bill Inmon.	15
Figure 4 : L'approche ascendante ou par bus définit par Kimball.	16
Figure 5 : .L'approche hybride.	17
Figure 6 : Processus de conception d'un ED - Approche par cycle de vie.....	18
Figure 7 : Les principales phases du cycle de vie d'un ED.	19
Figure 8 : Les couches du Web Sémantique. Source : [J. Jaffe, 2016]	27
Figure 9 : Graphe RDF (Ensemble de triplets). Source : [J. Charlet et al., 2003]	29
Figure 10 : Exemple d'un triplet RDF.....	30
Figure 11 : Représentation schématique de RDFS. Source : [H. Melhem, 2017].....	30
Figure 12 : Exécuter une requête SPARQL sur «Virtuoso SPARQL Query Editor ».	33
Figure 13 : Le résultat d'une requête SPARQL sur «Virtuoso SPARQL Query Editor ».	34
Figure 14 : Modèle en oignon. Source [G. Pierra, 2003]	37
Figure 15 : Représentation en carte heuristique des relations entre les données ouvertes.....	40
Figure 16 : l'évolution des relations entre DBpedia et divers autres projets du Web.....	41
Figure 17 : Le même représentation précédente, mais pour février 2017.....	41
Figure 18 : Structure des différents modèles de représentation des métadonnées	45
Figure 19 : Représentation TTL des métadonnées de format NGRAPHS et la requête SPARQL.	46
Figure 20 : Représentation TTL des métadonnées de format RDF Reification et la requête	46
Figure 21 : Représentation TTL des métadonnées de format Nary-Relation et la requête	47
Figure 22 : Représentation TTL des métadonnées de format Singleton Property et la requête.....	48
Figure 23 : Représentation TTL des métadonnées de format Compagnion Property et la requête	
Figure 24 : Etapes d'intégration dans un ED.	50
Figure 25 : Approches d'intégration à base d'ontologie conceptuelle. Source : [S. Khouri, 2013]	52
Figure 26 : Le processus de fonctionnement de l'approche orienté besoins.....	55
Figure 27 : Démarche de conception (Prototypage évolutif)	62
Figure 28 : Schéma de la solution générale	67
Figure 29 : L'architecture globale de la solution.....	69
Figure 30 : Le modèle "MVC".....	70
Figure 31 : Diagramme de cas d'utilisations de la gestion des ontologies sources	72
Figure 32 : Cas d'utilisations de la gestion des besoins	73
Figure 33 : Diagramme de cas d'utilisations de la gestion des mappings	74
Figure 34 : Cas d'utilisation de la gestion de l'extraction des instances à partir des sources	75
Figure 35 : Cas d'utilisation de la phase de transformation	76
Figure 36 : Cas d'utilisation de la phase de chargement dans l'EDS.....	77
Figure 37 : Cas d'utilisation de la phase de consultation de l'EDS	78
Figure 38 : Maquette de la page de chargement des sources internes	79

Figure 39 : Maquette de la page de gestion des besoins	79
Figure 40 : Maquette de la page de consultation des mappings existants.....	80
Figure 41 : Maquette de la page d'extraction des instances	80
Figure 42 : Maquette de la page de transformation et unification des données et métadonnées	81
Figure 43 : Maquette de la page de chargement des données et métadonnées transformés	81
Figure 44 : Diagramme de classes du système	82
Figure 45 : Tabelau représentant la description de sdifferentes classes de la solution globale.	83
Figure 46 : Le diagramme de classe de l'ontologie interne. Source : [F. Irini, 2016]	84
Figure 47 : L'ensemble des classes de l'ontologie interne E-commerce	85
Figure 48 : Les différentes instances de la classe Product	85
Figure 49 : L'ensemble des annotations (Données + Méta-Données) de 'instance « Product1»	85
Figure 50 : Les instances des classes externes "ExternalClass"	86
Figure 51 : Les instances des différents formats de données	86
Figure 52 : Schéma des classes de données et de métadonnées de l'ontologie cible de l'EDS.....	89
Figure 53 : Schéma représentant les mappings de données	90
Figure 54: L'ensemble des propriétés de l'instance de mapping "ETL2"	90
Figure 55 : Schéma représentant les méta-mappings.....	91
Figure 56 : L'ensemble des propriétés de l'instance "Requirement_5"	91
Figure 57 : Diagramme de séquence de la gestion des sources internes.....	92
Figure 58 : Diagramme de classes montrons les interactions entre le contrôleur « OntologyCont » et les composants des ontologies sources.	93
Figure 59 : Diagramme de séquence de la simulation de l'exécution d'un mapping.....	94
Figure 60 : Diagramme de classes montrant l'interactions entre les entités et le contrôleur.....	94
Figure 61 : Diagramme de séquence de l'affection des mappings aux besoins.	95
Figure 62 : Diagramme de classes des interactions entre le module des besoins et mappings	96
Figure 63 : Diagramme d'activité du processus d'intégration de données et de métadonnée	97
Figure 64 : Diagramme de classes interactions du module de gestion du processus d'intégration.	98
Figure 65 : Schéma représentant les six formalismes de métadonnées et le format générique	99
Figure 66 : Processus d'extraction générique des données et des métadonnées.	99
Figure 67 : Diagramme d'activités de l'unification des formats STD Reification et Nary.....	101
Figure 68 : Unification du format STDREIF et Nary sous le format générique	102
Figure 69 : Architecture du service Jena API	105
Figure 70 : Diagramme de classes représentant les différentes couches du modèle MVC.....	105
Figure 71 : Interface d'importation de l'ontologie globale	116
Figure 72 : Interface de gestion des mappings.....	117
Figure 73 : interface de gestion des besoins	117
Figure 74 : Interface d'extraction des sources.....	118
Figure 75 : Interface de transformation des données et des métadonnées	118
Figure 76 : Interface de chargement dans le schéma cible de l'EDS	119
Figure 77 : Interface du tableau de bord	119
Figure 78 : La distribution des ressources internes et externes par rapport aux besoins	123
Figure 79 : La distribution des données insérées et inférées par rapport aux besoins	123
Figure 80 : Nombre de concepts et propriétés impliquées dans les mappings de chaque besoin .	125
Figure 81 : Diagramme de Gantt du projet	129

Liste des tableaux

Tableau 1 : Exemples de portails LOD. Source : [https://lod-cloud.net/].....	42
Tableau 2 : Tableau représentant les détails des sources externes.....	111

Tableau 3 : Méta-propriétés de la source interne avec les méta-classes cibles correspondantes..	113
Tableau 4 : Méta-propriétés des sources externes et leurs méta-classes cibles correspondantes..	113
Tableau 5 : Tableau représentant le volume des sources internes et externes	122
Tableau 6 : Tableau représentant les résultats des métriques de description des métadonnées....	124
Tableau 7 : Tableau représentant les résultats des métriques de qualité.....	124
Tableau 8 : Métriques représentant les résultats des métriques de performances.....	125

Liste des abréviations

ED	Entrepôt de Données
EDS	Entrepôt de Données Sémantiques
SIAD	Système d'Information d'Aide à la Décision
LOD	Linked Open Data
WS	Web Sémantique
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
OWL	Web Ontology Language
TTL	Turtle
MVC	Model-View-Controller
ETL	Extraction-Transformation-Load
API	Application Programming Interface
BI	Business Intelligence
BI&A	Business Intelligence & Analytics
IRI	Internationalized Resource Identifier
MD	Multidimensionnel/ Multidimensional
OLAP	Online Analytical Processing
UML	Unified Modeling Language
Web 3.0	Web des données
XML	EXtended Markup Language

Introduction Générale

L'industrie des technologies des données génère des milliards de dollars chaque année pour gérer efficacement les données disponibles. Ces dernières années, cette industrie fait face au phénomène d'ouverture des données où de nombreuses institutions (comme les gouvernements, les universités, les administrations, les entreprises commerciales, etc.) sont appelées à partager certaines de leurs données sur le web selon les principes du Linked Open Data, afin de donner un accès gratuit à l'information et au savoir dans divers secteurs.

Le mouvement Linked Open Data (LOD), est un ensemble de principes de conception pour partager les données certifiées et à licence libre sur le Web, en utilisant les normes du Web Sémantique. Ce mouvement tente d'utiliser le Web comme ressource pour connecter les données et réduire les obstacles à leur liaison. Ce mouvement a pour objectif d'assurer une exploitation efficace de la quantité énorme de données circulant dans le web et créer davantage de valeur et de connaissance pour les différentes institutions. Plusieurs portails LOD existent dont, DBPEDIA une base de connaissances sémantique qui contient une version structurée et normalisée du contenu de Wikipedia, et travaille à l'interconnecter avec d'autres ensembles de données ouvertes provenant du Web.

Les données sont donc omniprésentes, et les utilisateurs doivent pouvoir les exploiter pour extraire des connaissances, obtenir des informations et prendre des décisions éclairées. La valeur des connaissances découvertes à partir du LOD pourrait être supérieure, si elles sont bien exploitées.

À cet effet, de nombreux systèmes tirent profit de l'environnement du LOD afin de créer une valeur ajoutée. Les systèmes décisionnels, principalement les entrepôts de données (ED), sont un exemple pertinent exprimant des résultats concrets à l'exploitation efficace des LOD dans la prise de décision. La force des ED se situe dans leur capacité à intégrer les différentes sources de données hétérogènes, dans une source commune et normalisée, offrant une vue unifiée des données. Au cours de la dernière décennie, la technologie d'entrepôt de données a été appliquée avec succès dans plusieurs domaines tels que les télécommunications, la vente au détail, la finance et de nombreux autres les industries. Il prend en charge une large gamme d'applications dans tout organisme. Les ED ont été largement utilisés pour proposer des solutions durables aux entreprises.

La quantité de sources de données hétérogènes et distribuées présente sur le web, est le résultat de la croissance spectaculaire de l'Internet et son adoption généralisée par les entreprises mondiales. Pour faciliter l'accès à ces quantités de données, et rendre ces sources interopérables. Une solution innovante qui consiste à faire la combinaison entre la technologie de *l'entreposage de données* et celles du *web sémantique*, formant ainsi les Entrepôts de Données Sémantiques (EDS). Les EDS sont des systèmes d'ED permettant l'intégration de données sémantiques, généralement formalisées selon les standards du web sémantique comme le langage OWL ou le langage RDF.

Dans le cas traditionnel, les entreprises se contentent des sources de données internes pour alimenter un projet d'entreposage de données. Cependant, lors de ces dernières années, de nombreux projets proposent de s'intéresser également aux sources externes du type LOD, car elles permettent d'alimenter l'ED existant avec de nouvelles données venant compléter les données internes. Elles apportent de nouveaux axes d'analyse et enrichissent ainsi le processus décisionnel. Dans ce contexte, trois problématiques se posent : en premier lieu, malgré le fait que les entreprises maîtrisent souvent leurs données internes, et que la qualité et la fiabilité de celles-ci est assurée. Il y a cependant une nécessité de faire l'intégration des données internes qui nécessitent des transformations pour principalement les unifier et les agréger avec les données externes.

En deuxième lieu, il existe une quantité importante de sources externes de données, elles peuvent représenter une valeur ajoutée surprenante. Pour cela, il y a une nécessité de sélectionner les sources externes selon les besoins des décideurs uniquement. En troisième lieu, pour assurer la qualité des résultats, l'intégration des données internes uniquement s'avère insuffisante. Une sélection des sources externes pas uniquement selon le besoin mais aussi selon leur provenance ainsi que leur fiabilité est essentielle pour assurer la qualité des données cibles. Une première étape qui permet au concepteur d'un

EDS d'avoir une vision globale de la qualité des données sémantiques externes manipulées, consiste à analyser leurs métadonnées qui fournissent des informations sur la provenance de ces données, leur création, leur validité, leur description, etc.

Les métadonnées, selon la plus répandue de ses définitions représentent « les données sur les données ». Ils contiennent les informations nécessaires pour comprendre et utiliser efficacement les données. Cela inclut la documentation du contenu, du contexte, de la provenance, de la qualité, de la structure et de l'accessibilité de l'ensemble des données. Les métadonnées sont porteuses d'informations pertinentes pour faciliter la vérification de la qualité des données par un concepteur. Cependant, les métadonnées sont aussi hétérogènes : elles se trouvent sous des *formats* différents et des *sémantiques* différentes. Une intégration des métadonnées est donc nécessaire. En effet, l'exploitation efficace des métadonnées aide à disposer d'un système décisionnel de meilleure qualité, bien documenté, fiable et archivé de manière cohérente.

Notre travail fait partie d'un projet d'entreposage de données sémantiques, entamé au laboratoire LCSI (ESI) pour lequel une démarche de conception orientée par les besoins pour la construction d'un EDS, a été proposée. Nous nous intéressons en 1^{er} lieu dans ce travail, à cette démarche de conception d'un ED qui met l'accent sur l'importance des besoins utilisateurs pour mener l'intégration des sources hétérogènes. L'intérêt de considérer les besoins facilite l'identification des fragments de données externes dans la grande masse de données externes existantes, à considérer pour l'entrepôt de l'entreprise. Dans ce cadre, un premier prototype existant a déjà été réalisé, nous l'avons analysé et y avons constaté plusieurs lacunes.

Nous nous intéressant en 2^{ème} lieu à l'exploitation et à l'intégration des différentes formes de données et métadonnées du LOD, dans un environnement décisionnel tournant autour d'un entrepôt de données sémantique. Dans ce contexte, l'entrepôt de données doit implémenter une couche d'intégration des données puis de leurs métadonnées respectives. Le rajout d'une couche d'intégration des métadonnées permet d'assurer la fiabilité des sources de données externes et faire une sélection pertinente des sources, suivant les besoins des utilisateurs. Le concepteur, aura grâce à cela, une vision unifiée de l'ensemble des données qu'il doit intégrer et peut valider plus facilement les fragments à matérialiser dans l'ED cible.

Dans notre solution, il y a nécessité d'orienter notre attention vers ces points essentiels :

- i) La proposition d'un processus d'intégration des données et des métadonnées de sources sémantiques internes et externes du type LOD,
- ii) Le chargement des données et métadonnées unifiées, respectivement dans le modèle et méta-modèle de l'EDS cible,
- iii) L'intégration de ce processus dans la démarche de conception orientée besoins proposée,
- iv) La mise en œuvre de l'approche de conception globale définie.

On note, que notre projet global initié au laboratoire est d'assurer la gestion de la qualité de l'EDS. Cependant, notre objectif dans ce PFE consiste à fournir le processus d'intégration et de l'automatiser et non pas d'assurer le processus de validation de la qualité des données. Enfin, le but de ce travail consiste à une refonte du prototype existant implémentant l'approche par besoin et l'intégration des données et des métadonnées sources.

Dans ce présent mémoire, nous allons exploiter les données et métadonnées du LOD pour la construction d'un système d'intégration adossé sur un entrepôt de données sémantique. Nous allons donc discuter dans ce présent document, les différentes phases d'analyse des besoins, de la conception et enfin de l'implémentation et l'évaluation du système. Le rapport se compose de trois parties principales la première représente l'état de l'art, où nous avons fait une synthèse bibliographique des différentes notions importantes pour notre projet. Elle est composée de trois chapitres complémentaires, le premier traite les entrepôts de données, le second, les données ouvertes et liées, le dernier, l'association entre ces deux précédents, produisant ainsi les Entrepôts de données Sémantiques. La deuxième partie comporte la conception de la solution. Elle est composée de trois principaux chapitres organisés de façons à permettre au lecteur de comprendre l'acheminement logique des phases de travail.

Le premier chapitre expose le contexte, les problématiques et les différents objectifs du projet. Le second chapitre comporte l'étude de l'existant où nous avons présenté l'approche existante sur laquelle une grande partie de notre solution se repose. Dans le dernier chapitre, nous avons détaillé la conception et la modélisation des différentes couches de la solution. Nous avons partagé ce chapitre en plusieurs points expliquant la démarche et les étapes de conception suivies, nous avons ensuite défini les besoins techniques et fonctionnels exprimés par les utilisateurs de notre système. Les points restants expliquent le fonctionnement et l'architecture de la solution. La troisième partie de notre mémoire comporte la phase de réalisation et d'implémentation du système. Dans celle-ci, nous verrons les différents outils, langages et technologies utilisés mais aussi les différentes fonctionnalités réalisées par système dans les trois couches client, traitement, service et persistance. Le deuxième chapitre de cette partie est l'évaluation du résultat de l'implémentation sous formes de métriques, testant les performances du système. Dans la dernière partie du mémoire nous décrirons les différentes étapes de gestion de projet, les tâches réalisées et l'échéance de chacune.

Nous allons clôturer ce présent mémoire par une conclusion générale, où nous allons présenter les différentes perspectives vers lesquelles s'orienteront les travaux futurs.

Partie 1 : Etat de l'art

Pour pouvoir comprendre le contexte de notre travail et pour mieux cerner la problématique et les objectifs de notre étude, nous allons commencer par la partie de l'état de l'art où nous allons aborder différents concepts en liaison avec notre projet, en se basant principalement sur des recherches bibliographiques sur des livres, des articles de recherche et différentes expériences précédentes dans le domaine.

Cette partie est divisée en trois grands chapitres. Dans le 1^{er} chapitre nous aborderont la conception d'Entrepôts de Données et leurs fonctionnements. Nous allons ensuite parler dans le deuxième chapitre à propos du *web sémantique* et des *données ouvertes et liées* - répondant sous leur nom en anglais *Linked Open Data (LOD)*. Enfin nous allons aborder dans le dernier chapitre la combinaison entre le système décisionnel qui est l'ED avec les ontologies du web sémantique, formant alors ce qu'on appelle un *Entrepôt de données sémantique* (EDS). Dans ce dernier chapitre nous allons mettre l'accent sur le processus d'intégration des sources de données.

Chapitre 1 : Entrepôt de données (ED)

Introduction

Les organisations sont aujourd'hui confrontées à des défis de plus en plus complexes en matière de gestion et de résolution de problèmes afin d'atteindre leurs objectifs opérationnels. Cette situation oblige les membres de ces organisations à utiliser des outils d'analyse permettant de mieux prendre en charge leurs décisions, appelés les systèmes d'aide à la décision (SIAD). L'informatique décisionnelle comprend un ensemble de méthodologies, de processus, d'architectures et de technologies permettant de transformer des données brutes en informations utiles à la prise de décision. Les systèmes d'aide à la décision aident les responsables de différents niveaux de l'organisation à analyser les informations stratégiques. Ces systèmes collectent de grandes quantités de données et les réduisent à une forme pouvant être utilisée pour analyser le comportement organisationnel. Cette transformation de données comprend un ensemble de tâches qui extraient les données des sources et, via des processus d'extraction, de transformation, d'intégration et de nettoyage, stockent les données dans un référentiel commun appelé entrepôt de données. Les entrepôts de données ont été développés et déployés en tant que partie intégrante des systèmes d'aide à la décision afin de fournir une infrastructure permettant aux utilisateurs d'obtenir des réponses efficaces et précises à des requêtes complexes.

1 Historique

Depuis la fin des années 1970, la technologie des bases de données relationnelles a été adoptée par la plupart des organisations pour stocker leurs données essentielles. Cependant, de nos jours, les besoins de ces organisations ne sont plus les mêmes qu'avant. D'un côté, l'accroissement de la dynamique et de la compétitivité du marché a conduit à la nécessité d'avoir la bonne information au bon moment. Les gestionnaires doivent être bien informés de ce qui suit afin de prendre les décisions qui s'imposent pour suivre le rythme des affaires avec succès. D'autre part, les données détenues par les organisations sont généralement dispersées entre différents systèmes, chacun conçu pour un type d'activité particulier. En outre, ces systèmes peuvent également être répartis géographiquement dans différentes branches de l'organisation. Les systèmes de base de données traditionnels ne sont pas bien adaptés à ces nouvelles exigences, car ils ont été conçus pour prendre en charge les opérations quotidiennes plutôt que pour l'analyse des données et la prise de décision. En conséquence, de nouvelles technologies de base de données pour ces tâches spécifiques sont apparues dans les années 90.

En effet, au début des années 90, à la suite d'un monde de plus en plus concurrentiel et en rapide mutation, les entreprises ont compris qu'elles devaient analyser les données de manière sophistiquée pour appuyer leurs processus décisionnels. Les bases de données opérationnelles ou transactionnelles traditionnelles ne répondent pas aux exigences de l'analyse des données, puisqu'elles étaient conçues et optimisées pour supporter les opérations quotidiennes de l'entreprise, et leur préoccupation première était d'assurer un accès simultané par plusieurs utilisateurs et, en même temps, de fournir des techniques de récupération pour garantir la cohérence des données. Les bases de données opérationnelles classiques contiennent des données détaillées, n'incluent pas les données historiques et sont peu performantes lors de l'exécution de requêtes complexes impliquant de nombreuses tables ou agrégant de gros volumes de données. En outre, lorsque les utilisateurs doivent analyser le comportement d'une organisation dans son ensemble, les données de plusieurs systèmes opérationnels différents doivent être intégrées. Cela peut être une tâche difficile à accomplir en raison des différences de définition et de contenu des données. Par conséquent, les entrepôts de données ont été proposés comme une solution aux demandes croissantes des utilisateurs décisionnaires.

2 Définitions

Dans cette partie nous allons définir certains concepts d'entreposage de données.

Tout d'abord on précise que l'Entrepôt de données est plus répandu sous son nom en anglais dit « Data Warehouse ». On utilisera le terme « Datawarehouse » uniquement dans cette partie de notre travail pour qualifier un « Entrepôt de données ».

2.1 *Entrepôt de données (Datawarehouse)*

Comme nous l'avons vu précédemment un entrepôt de données est la base de l'activité analytique et décisionnelle de toute organisation. Dans cette partie, nous allons présenter trois définitions de l'entrepôt de données selon trois sources différentes nécessaires pour la compréhension de notre travail.

- Le terme d'entrepôt de données a été créé par William H. Inmon, connu comme le « père du Data Warehouse ». La définition classique de l'entrepôt de données, donnée par Inmon dans son livre "Building the Data Warehouse", est : « Un entrepôt de données (data warehouse) est une collection de données thématiques, intégrées, non volatiles et historisées pour la prise de décisions » [Inmon, 2002].
- Ralph Kimball, définit l'entrepôt de données comme « une copie des données de transaction spécifiquement structurée pour les requêtes et l'analyse ».
- On trouve aussi la définition suivante : « Un entrepôt de données est une base de données construite par copie et réorganisation de multiples sources (dont principalement le système transactionnel de l'entreprise), afin de servir de source de données à des applications décisionnelles. L'entrepôt de données agrège de nombreuses données de l'entreprise (intégration), Il mémorise les données dans le temps (historisation) et les organise pour faciliter les requêtes de prise de décision (optimisation). » [Goglin, 2001]

Les définitions ci-dessus se complètent. En effet, Inmon définit le but d'un entrepôt de données et met l'accent sur certaines caractéristiques essentielles d'un entrepôt de données comme suit :

- **Orienté par sujet**, signifie qu'un entrepôt de données cible un ou plusieurs sujets d'analyse en fonction des besoins analytiques des responsables à différents niveaux du processus de prise de décision.
- **Intégré**, exprime le fait que le contenu d'un entrepôt de données résulte de l'intégration de données provenant de divers systèmes opérationnels et externes.
- **Non volatile**, indique qu'un entrepôt de données accumule les données des systèmes opérationnels pendant une longue période. Ainsi, la modification et la suppression des données ne sont pas autorisées dans les entrepôts de données et la seule opération autorisée est la purge des données obsolètes qui ne sont plus nécessaires.
- **La variation temporelle**, souligne le fait qu'un entrepôt de données garde une trace de l'évolution de ses données, par exemple pour connaître l'évolution des ventes au cours des derniers mois ou des dernières années.

Kimball met l'accent sur l'utilité analytique des entrepôts de données, et la dernière définition parle de la multiplicité des sources de données d'un entrepôt de données, et de l'utilisation de celui-ci.

2.2 ***Magasin de données (Datamart)***

Le Data Warehouse est connue pour représenter un référentiel organisationnel de données centralisé au sein de l'entreprise, recouvrant ainsi de nombreux domaines d'activité. Or, un stockage de données dans un emplacement commun pour deux domaines d'activité complètement différents et utilisant des données quasiment indépendantes, s'avère encombrant et diminue ainsi le temps de réponse et donc la performance des Data Warehouse.

Pour cela une architecture de stockage plus spécialisée a été conçue et ajoutée à l'architecture initiale d'un Data Warehouse, cette architecture est appelée datamarts. Un datamart est « La couche d'accès de l'environnement de l'entrepôt de données utilisé pour transmettre les données aux utilisateurs. C'est un sous-ensemble de l'entrepôt de données qui est généralement orienté vers un sujet spécifique (finances, marketing, ventes) » [J. Serra, 2013]. L'auteur ajoute « La combinaison logique de tous les datamarts est un entrepôt de données ». Kimball propose une définition plus brève en disant « Dans sa forme la plus simpliste, un datamart présente les données d'un seul processus de gestion. Ces processus de gestion traversent les frontières des fonctions organisationnelles. »

En bref, un entrepôt de données contient de nombreux domaines, et un datamart n'en contient qu'un seul.

2.3 ***Modélisation multidimensionnel***

La modélisation multidimensionnelle c'est d'analyser des indicateurs numériques (par exemple chiffre d'affaires, nombre d'individus, ratios, etc.) dans un contexte précisé par le croisement de plusieurs dimensions d'analyse (par exemple temps, géographie, organisation, produits, etc.), généralement présentées sous forme d'arbres hiérarchiques.

Le modèle de données multidimensionnel fait donc un stockage de données sous la forme d'un cube de données.

Une base de données multidimensionnelle aide à fournir des réponses rapides et précises aux requêtes métiers complexes.

Les entrepôts de données et les outils de traitement analytique en ligne (OLAP) (Voir point 4 du chapitre 1) sont basés sur un modèle de données multidimensionnel. OLAP dans l'entreposage de données permet aux utilisateurs d'afficher des données sous différents angles et dimensions.

Les schémas pour le modèle de données multidimensionnel sont les suivants: -

- Schéma en étoile
- Schéma des flocons de neige
- Schéma des constellations de faits

Le schéma le plus communément utilisé c'est le schéma en étoile. Ce schéma est composé principalement de tables de faits et de dimensions. L'un des principaux objectifs du schéma en étoile est de simplifier un ensemble complexe de tables normalisées et de consolider les données (éventuellement de différents systèmes) dans une structure de base de données pouvant être interrogée de manière très efficace.

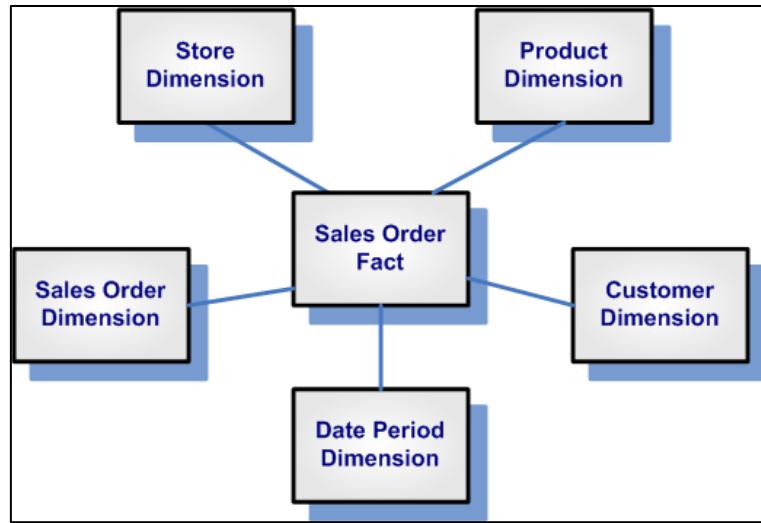


Figure 1 : Schéma en étoile (Fait et dimensions). / Source : <http://bi-insider.com/posts/dimensional-modeling-and-data-warehouses/>

2.3.1 Fait

Dans un schéma en étoile, les faits représentent le sujet principal et l'occupation de l'entreprise. Il contient toutes les clés primaires des dimensions et les faits et les mesures associés (une propriété sur laquelle des calculs peuvent être effectués), tels que la quantité vendue, le montant vendu et les ventes moyennes. Par exemple, les ventes du magasin représentent un fait.

2.3.2 Dimensions

Les dimensions sont des entités vis-à-vis desquelles l'entreprise souhaite conserver des enregistrements. Par exemple, dans l'enregistrement des ventes du magasin, les dimensions permettent au magasin de suivre des événements tels que branches et emplacements.

3 Objectifs d'un ED

Dans cette partie, nous examinerons les systèmes traditionnels d'aide à la décision (SIAD) et les raisons pour lesquelles ils n'ont pas fourni des renseignements complets, exacts et opportuns à l'organisation. Ensuite, nous décrirons comment les objectifs à court et à long terme des entrepôts de données permettent de combler les lacunes des environnements traditionnels du SAD.

3.1 Lacunes des systèmes traditionnels d'aide à la décision (SIAD)

3.1.1 Définition :

On définit un SIAD comme suit « Le SIAD est un outil d'observation et de description qui vise, à partir de données de gestion et/ou de statistiques, à donner aux managers d'une entreprise les moyens d'identifier des alertes de gestion, de suivre l'évolution de l'activité et de disposer d'outils d'investigation de sujets ou phénomènes particuliers. Il ne fournit pas les explications ni les commentaires qui relèvent d'une phase de travail postérieure à l'observation ». [M.Volle, 2001]

Les outils fournis par le SIAD pour remplir ces divers objectifs sont, [M.Volle, 2001] :

- Le tableau de bord comportant des alertes.

- Des tableaux préformatés contenant l'essentiel de la statistique d'activité et d'environnement.
- Analyses et statistiques dans Excel.
- Des tableaux et graphiques restituant les résultats d'interrogations en utilisant la technologie " hypercubes ".

3.1.2 Lacunes des SIAD traditionnels :

Dans ce qui suit nous allons donner quelques exemples des lacunes des SIAD traditionnels les plus courantes [S. Adelman et al., 2000] :

- **Les données ne sont pas comprises :** En effet, la principale cause de cela est que les concepteurs et les utilisateurs originaux du SAD construisent un système sur mesure donc c'est les seuls qui peuvent comprendre réellement des données. Cependant, les nouveaux utilisateurs qui souhaitent utiliser les données n'ont souvent pas la même compréhension de celles-ci.
- **Les rapports sont incohérents :** Les utilisateurs possèdent des interprétations et des vues différentes des données dans les SAD. Par conséquent, plusieurs rapports différents et incohérents peuvent être générés, cela cause une perte de temps et baisse de qualité des résultats attendus.
- **Les données ne sont pas nettoyées :** Les données présentes dans les SAD, sont mal gérées ce qui induit la présence d'enregistrements inutile et d'autres incohérences ce qui induit aux erreurs dans les résultats. Le nettoyage de ces données se fait manuellement, ce qui cause une paresse dans les équipes IT, qui eux se présume capable de corriger les résultats obtenus.
- **Les données de sont pas intégrées :** Les unités opérationnelles construisent leurs systèmes de façon autonome et en fonction de leurs vues individuelles, donc même s'il y avait une volonté de permettre à d'autres utilisateurs de partager leurs données, les données entre systèmes ne seront pas intégrées. Par conséquent, l'accès aux données sur plusieurs systèmes implique souvent une solution longue et coûteuse et l'écriture de ponts compliqués entre les différents systèmes.
- **Les données historiques ne sont pas disponibles :** L'accent mis sur la prise de décision tactique et stratégique, apporte une nouvelle exigence au SAD : la capacité de comparer les données entre les périodes de temps, les régions géographiques et d'autres dimensions opérationnelles. Étant donné que le SAD traditionnel ne stocke généralement pas les données historiques de la même manière que les données actuelles, les analystes finissent par créer de nouveaux systèmes élaborés pour effectuer leurs analyses, ce qui s'avère long et fastidieux.

Sachant que les besoins des utilisateurs augmentent et leurs demandes d'informations devient de plus en plus importante, il devient difficile pour les professionnels de la technologie d'information (IT) de répondre à leurs demandes. Ceci même s'il y a de nombreux projets SAD qui se déroulent en parallèle pour satisfaire les différentes unités opérationnelles d'une organisation. Ces systèmes parallèles ne constituent pas une solution de gestion des données acceptable en raison de leurs lacunes inhérentes car elles ne sont ni coordonnées ni intégrées, elles produisent des systèmes SAD autonomes.

3.2 Objectifs des ED pour remédier aux lacunes des SIAD traditionnels

Nous avons identifié ci-dessus, les principales lacunes des systèmes traditionnels d'aide à la décision. Nous pouvons à présent parler du rôle qu'occupe un ED dans le cycle de vie décisionnel. Un entrepôt de données n'est pas seulement une autre base de données du SAD. « Il s'agit d'un environnement composé d'une ou de plusieurs bases de données conçues pour fournir des renseignements commerciaux cohérents et conciliés à toutes les unités opérationnelles de l'organisation. » [S. Adelman et al., 2000]

Selon [S. Adelman et al., 2000], il y a deux types d'objectifs d'un ED, les objectifs à court terme et les objectifs à long terme.

3.2.1 Objectifs à court terme

Les objectifs à court terme représentent les avantages qu'on peut atteindre dans les utilisations immédiates d'un ED elles apparaissent avec chaque itération de celui-ci.

Vous trouverez ci-dessous, les principaux objectifs à court terme des ED :

- **Améliorer la qualité des données :** Le but ultime d'un ED est d'arriver à nettoyer les données convenablement, Le nettoyage des données est un vrai problème dans l'entreposage des données. Un entrepôt de données lors de sa construction passe par un processus ETL qui fait l'extraction, la transformation et le nettoyage des données, le but de ce processus est de fournir "des données propres, intégrées, cohérentes et rapprochées provenant de sources multiples".
- **Minimiser les rapports incohérents :** Les rapports incohérents sont principalement causés par une mauvaise utilisation des données, et la principale raison de la mauvaise utilisation des données est le désaccord ou la mauvaise compréhension de la signification ou du contenu des données. La correction de ce problème est une autre difficulté de l'entreposage de données, qui a pour but d'unifier les données du même domaine mais venant de sources différentes, pour ainsi créer une cohérence dans ces données, et une complémentarité entre leurs sources pour en fin avoir des rapport plus justes, compréhensibles et logiques reflétant ainsi les résultats sous forme de rapport cohérents.
- **Capturez et donnez accès aux métadonnées :** Les métadonnées sont indispensables pour le partage des données et la navigation dans les données.
 - **Partage des données :** Les définitions et le contenu convenus pour les éléments de données originaux et les nouveaux éléments de données ajoutés sont documentés dans un référentiel de métadonnées et sont mis à la disposition de tous les utilisateurs d'une organisation.
 - **Navigation dans les données :** Une fois que les données sources ont été nettoyées, transformées, agrégées et résumées, il est quasiment impossible que les utilisateurs les retrouvent dans l'entrepôt de données sans l'aide des métadonnées. Pour permettre une navigation de qualité il est nécessaire de capturer des métadonnées, « c'est-à-dire les définitions des données, les domaines, les algorithmes de transformation des données sources, les colonnes et les tableaux dans lesquels résident les données résultantes et tous les autres composants techniques » [S. Adelman et al., 2000] et rendre ces métadonnées facilement accessibles et utiles aux utilisateurs.
- **Intégrer des données provenant de sources multiples :** Il s'agit là d'un autre objectif principal pour tous les entrepôts de données, car il s'agit d'une lacune importante du SAD traditionnel. Le

processus ETL joue un rôle primordial dans l'intégration et l'unification des données provenant de sources différentes.

- **Fusionner les données historiques avec les données actuelles :** Un objectif typique de l'entrepôt de données est de stocker l'historique. Cet objectif s'accompagne de ses propres défis. Les données historiques sont rarement conservées dans les systèmes opérationnels car les données historiques ne sont pas aussi utiles au traitement opérationnel quotidien d'une fonction opérationnelle qu'à l'aide à la décision. L'ED gère l'historique des données avec plusieurs méthodes on cite alors la méthode Point-In-Time qui signifie qu'un enregistrement est écrit dans le fichier chaque fois qu'une transaction (modification) a lieu. Et la méthode Periodic Snapshot qui elle signifie qu'un enregistrement est écrit dans le fichier une fois pour chaque période (quotidienne, mensuelle, etc.), quel que soit le nombre de transactions effectuées pendant cette période.

3.2.2 Objectifs à long terme

La concentration sur les objectifs à court terme lors des itérations de l'entrepôt de données, les objectifs à long terme seront presque certainement atteints. Voici quelques exemples d'objectifs à long terme en matière d'entrepôt de données vue :

- **Rapprocher des vues différentes des mêmes données :** Un Entrepôt de données grandit à chaque itération, de plus en plus de vue différentes sur les mêmes données seront abordés et résolus sur chaque itération et l'organisation pourras alors atteindre le plus haut niveau de maturité en matière de gestion des données.
- **Fournir une image consolidée des données de l'entreprise :** Au fur et à mesure que de nouvelles données et de nouvelles exigences s'ajoutent à l'entrepôt de données dans les prochaines itérations, le modèle logique de données sera développé. Avec le temps, ce modèle se transformera en une image consolidée des données de l'entreprise dans le cadre de l'entrepôt de données.
- **Créer un environnement de données virtuel à « guichet unique » :** à guichet unique signifie que :
 - Toutes les données dans l'environnement de l'entrepôt de données sont accessibles via une interface commune ou "point d'entrée".
 - Une suite d'outils d'interrogation et de rapport standard est disponible et facile à utiliser
 - L'emplacement physique des données est transparent pour les utilisateurs
 - Les données sont intégrées, propres et cohérentes, ou du moins conciliaires
 - Pour atteindre cet objectif à long terme, deux types d'architectures doivent être très bien conçues et redessinées à chaque itération d'entrepôt de données : l'architecture technique et l'architecture des données.

4 Exploitation des ED pour l'analyse décisionnel

Le but initial de l'entreposage de données est la mise en œuvre d'un système adéquat qui permet d'avoir des résultats rapide et pertinent sur des requêtes complexes pour garder l'historique et l'archivage des données mais aussi aider l'interprétation et la visualisation des données décisionnelles grâce à la production de rapports et l'analyse des tendances. Pour cela, une grande variété de systèmes et d'outils peut être utilisée pour accéder, analyser et exploiter les données contenues dans les entrepôts de données.

- **Les systèmes OLAP :** Dès les débuts de l'entreposage de données, le mécanisme typique et principal pour ces tâches et l'analyse des résultats était le traitement analytique en ligne

(OLAP). En effet, « les systèmes OLAP permettent aux utilisateurs d'interroger de manière interactive et d'agréger automatiquement les données contenues dans un entrepôt de données. De cette manière, les décideurs peuvent facilement accéder aux informations requises et les analyser à différents niveaux de détail. » [A. Vaisman et al., 2014].

- Les outils de Data Mining : Sont apparu juste après les outils OLAP et ont pris une place considérable dans la fouille et l'interprétation des données. « Les outils de Data Mining sont également utilisés depuis les années 1990 pour déduire et extraire des connaissances intéressantes cachées dans des entrepôts de données. » [A. Vaisman et al., 2014].
- L'analyse de données : Un grand nombre de nouvelles techniques de la Business Intelligence ont été développées et utilisées pour faciliter la prise de décision. « Le marché du business intelligence est en train de changer pour fournir des outils d'analyse sophistiqués allant au-delà des techniques de navigation des données qui ont popularisé le paradigme OLAP. Ce nouveau paradigme est généralement appelé analyse de données. » [A. Vaisman et al., 2014]

Les techniques Business Intelligence utilisées pour exploiter un entrepôt de données peuvent être résumées comme suit : [A. Vaisman et al., 2014]

- Rapports, tels que des tableaux de bord et des alertes.
- Gestion de la performance, telle que mesures, indicateurs de performance clés (KPI), et des tableaux de bord.
- Des analyses, telles qu'OLAP, l'exploration de données, l'analyse de séries chronologiques, l'exploration de texte, l'analyse Web et la visualisation de données avancée.

5 Architecture d'un ED

Après avoir identifié les lacunes des SAD traditionnel et mis en avant les objectifs courts et à long terme des ED pour venir corriger les systèmes traditionnels d'aide à la décision, examinons les composants qui composent un environnement d'entreposage complet. Chaque composant de l'ED remplit une fonction spécifique. Nous allons donc présenter dans cette partie l'importance stratégique de chaque composant et comment l'utiliser efficacement pour gagner le jeu de l'entreposage de données vues selon Kimball dans son livre « The Data Warehouse Toolkit » [R. Kimball et al., 2008].

La figure 2 proposé par Kimball décrit qu'il y a quatre composantes distinctes à prendre en considération lors de l'exploitation de l'environnement de l'entrepôt de données. Nous trouvons alors les composants suivants : systèmes source opérationnels, la zone de rassemblement des données, la zone de présentation des données et les outils d'accès aux données, que nous analyserons ci-dessous :

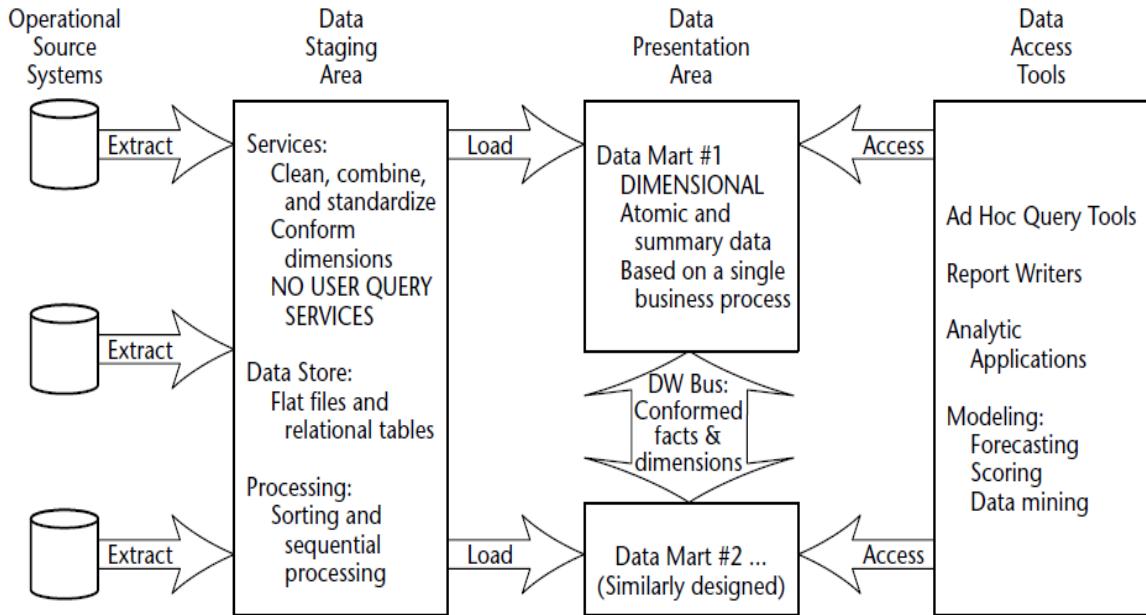


Figure 2 : Les éléments basic d'un ED. / Source : [R. Kimball et al., 2008]

5.1 Systèmes sources opérationnels (*Operational Source Systems*)

Ils représentent les différentes sources de données opérationnelles avec lesquels on peuple l'ED. « Il s'agit des systèmes d'enregistrement opérationnels qui saisissent les transactions de l'entreprise. » [R. Kimball, 2008]. Les systèmes sources se trouvent à l'extérieur de l'entrepôt de données car il n'y a peu ou pas de contrôle sur le contenu et le format des données dans ces systèmes opérationnels existants. Leurs principales priorités sont la performance et la disponibilité des traitements, ils conservent peu de données historiques et dans ces systèmes peu d'investissements ont été faits pour partager les ressources communes de données. Un travail qui devrait être fait pour faciliter la tâche de conception des entrepôts de données.

5.2 Zone de mise à disposition des données (*Data Staging Area*)

C'est l'un des éléments les plus essentiels dans un ED, c'est « la zone de mise à disposition des données de l'entrepôt de données, c'est aussi à la fois une zone de stockage et un ensemble de processus communément appelés extraction-transformation-chargé (ETL). » [R. Kimball, 2008]. Dans l'entrepôt de données, les données opérationnelles brutes sont extraites, transformées en données propres et cohérentes et chargées en données livrables dans l'entrepôt pour l'interrogation et la consommation des utilisateurs.

- **L'extraction :** elle représente la première étape dans le processus d'intégration des données. « Extraire signifie lire et comprendre les données source et copier les données nécessaires à l'entrepôt de données dans la zone de mise à disposition pour une manipulation ultérieure. » [R. Kimball, 2008].
- **La transformation :** Une fois les données extraites des systèmes sources opérationnels et une fois placées dans la zone de mise à disposition, « il existe de nombreuses transformations potentielles, telles que le nettoyage des données (correction des fautes d'orthographe, résolution des conflits de domaine, traitement des éléments manquants ou analyse dans des formats

standard), la combinaison de données provenant de sources multiples, la déduplication des données et l'attribution des clés. » [R. Kimball, 2008]

- **Le chargement :** Une fois les transformations faites, le processus de chargement des données dans la zone de présentation de l'entrepôt de données se lance. Les données seront alors mises à la disposition des utilisateurs pour les interroger.

5.3 Zone de présentation des données (*Data Presentation Area*)

La zone de présentation des données est l'endroit où les données sont organisées, stockées et mises à la disposition des utilisateurs, des rédacteurs de rapports et d'autres applications analytiques pour interrogation directe.

A ce sujet, Kimball présente deux enjeux dans le domaine des datamarts de zone de présentation la première c'est l'utilisation de la modélisation dimensionnelle dans la couche de présentation. En effet, « nous insistons pour que les données soient présentées, stockées et accessibles dans des schémas dimensionnels ». Il ajoute « La modélisation dimensionnelle est la technique la plus viable pour fournir des données aux utilisateurs d'entrepôts de données. »

En second degré, il ajoute que dans le domaine des datamarts la zone de présentation doit contenir des données atomiques détaillées. « Les données atomiques sont nécessaires pour résister aux assauts des requêtes imprévisibles des utilisateurs ad hoc. Bien que les tableaux de données puissent aussi contenir des données sommaires ou des agrégats améliorant le rendement, il ne suffit pas de fournir ces sommaires sans les données granulaires sous-jacentes sous une forme dimensionnelle. » [R. Kimball, 2008]. Dans la zone de présentation, il faut mettre en évidence données les plus fines afin que les utilisateurs puissent interroger l'ED de la façon la plus précise possible.

5.4 Outils d'accès aux données (*Data Access Tools*)

La dernière composante majeure de l'environnement de l'entrepôt de données est le ou les outils d'accès aux données. Elle représente la variété de capacités qui peuvent être fournies aux utilisateurs afin de tirer parti de la zone de présentation pour la prise de décision analytique. Kimball les décrit comme suit « Un outil d'accès aux données peut être aussi simple qu'un outil de requête ad hoc ou aussi complexe qu'une application sophistiquée d'exploration ou de modélisation de données. »

6 Conception d'un ED

Dans cette partie nous allons voir deux volets essentiels à connaître pour comprendre l'entreposage de données. Nous allons tout d'abord présenter les trois approches de conception des ED. Nous verrons ensuite les étapes de modélisation et de déploiement d'un ED.

6.1 Méthodes de conception d'un ED

Les approches de conception des entrepôts de données sont un aspect très important dans tout projet d'entreposage de données. La sélection de la bonne conception de l'entrepôt de données pourrait faire gagner beaucoup de temps et réduire les coûts des projets.

Il existe deux principales approches de conception d'entrepôt de données prise comme référence dans l'étape de conception d'une solution décisionnelle. Celle-ci dépend des exigences du projet, et le choix entre ces deux approches se fait selon celle qui convient le mieux au scénario voulu dans le projet. Ces méthodologies sont le résultat des recherches de Bill Inmon et Ralph Kimball.

Cependant, nous allons citer aussi une troisième approche dite *Hybride* qui associe les deux approches précédentes. Dans ce qui suit nous allons nous reposer principalement sur le travail de Wayne Eckerson, dans [W. Eckerson, 2003]. Dans son article il compare et oppose les approches les plus couramment utilisées pour créer un entrepôt de données.

« Les gestionnaires d'entrepôts de données doivent être au courant de ces méthodologies, mais ils ne doivent pas s'y tenir à la lettre », dit-il. « Ces méthodologies ont façonné le débat sur les meilleures pratiques en matière d'entreposage de données et constituent les éléments constitutifs des méthodologies élaborées par des consultants praticiens. » [W. Eckerson, 2003].

6.1.1 Conception Top-Down

L'approche Top-Down aussi appelée l'approche d'Inmon ou l'architecture Hub-and-spoke est expliquée par plusieurs leaders de la Business Intelligence de la façon suivante :

« L'approche descendante considère l'entrepôt de données comme le pivot de tout l'environnement analytique. L'entrepôt de données contient des données atomiques ou transactionnelles qui sont extraites d'un ou de plusieurs systèmes sources et intégrées dans un modèle de données d'entreprise normalisé. À partir de là, les données sont comprimées, dimensionnées et réparties dans un ou plusieurs datamarts "dépendants". Ces data marts sont "dépendants" parce qu'ils tirent toutes leurs données d'un entrepôt de données centralisé. » [Wayne Eckerson]

On trouve aussi Sansu George dans [Sansu George, 2012] qui décrit l'approche comme suit : « Inmon définit un entrepôt de données comme un référentiel centralisé pour l'ensemble de l'entreprise. Un entrepôt de données stocke les données "atomiques" au niveau de détail le plus bas. Les datamarts dimensionnels ne sont créés qu'après la création de l'entrepôt de données complet. Ainsi, l'entrepôt de données est au centre de l'usine d'information d'entreprise, qui fournit un cadre logique pour la fourniture de renseignements commerciaux. »

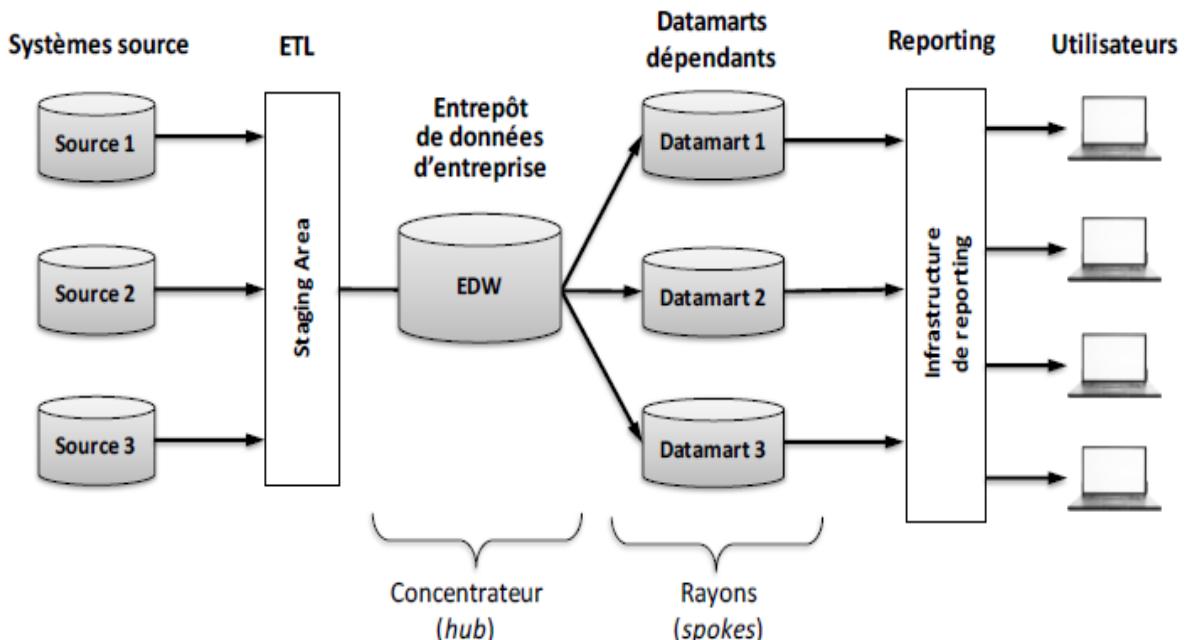


Figure 3 : L'approche descendante par Bill Inmon. / Source : [B. Inmon, 2002]

6.1.2 Conception Bottom-Up

L'approche Bottom-Up de Kimball est définie comme suit :

Wayne Eckerson l'explique de la façon suivante : « Dans une approche ascendante, l'objectif est de créer de la valeur pour l'entreprise en déployant des datamarts dimensionnels aussi rapidement que possible. Contrairement à l'approche descendante, ces datamarts contiennent toutes les données - à la fois atomiques et sommaires - que les utilisateurs peuvent vouloir ou avoir besoin, maintenant ou à l'avenir. Les données sont modélisées selon un schéma en étoile afin d'optimiser la convivialité et la performance des requêtes. Chaque data mart s'appuie sur le suivant, réutilisant les dimensions et les faits afin que les utilisateurs puissent interroger les data marts, si désiré, pour obtenir une version unique de la vérité ainsi que des données à la fois sommaires et atomiques. » [Wayne Eckerson]

Tandis que Sansu George donne une définition beaucoup plus simple en mettant directement l'accent sur la création des datamarts en premier grâce à la prise en compte de l'aspect business et des différents services de l'organisation. Elle déclare « Dans l'approche de conception dimensionnelle de Ralph Kimball (la conception ascendante), les datamarts facilitant les rapports et l'analyse sont créés en premier lieu ; Celles-ci sont ensuite combinées pour créer un vaste entrepôt de données. L'architecture d'entreposage de données de Kimball est également connue sous le nom de bus d'entrepôt de données (BUS). »

Elle ajoute : « La modélisation dimensionnelle met l'accent sur la facilité d'accès de l'utilisateur final et fournit un niveau de performance élevé à l'entrepôt de données. »

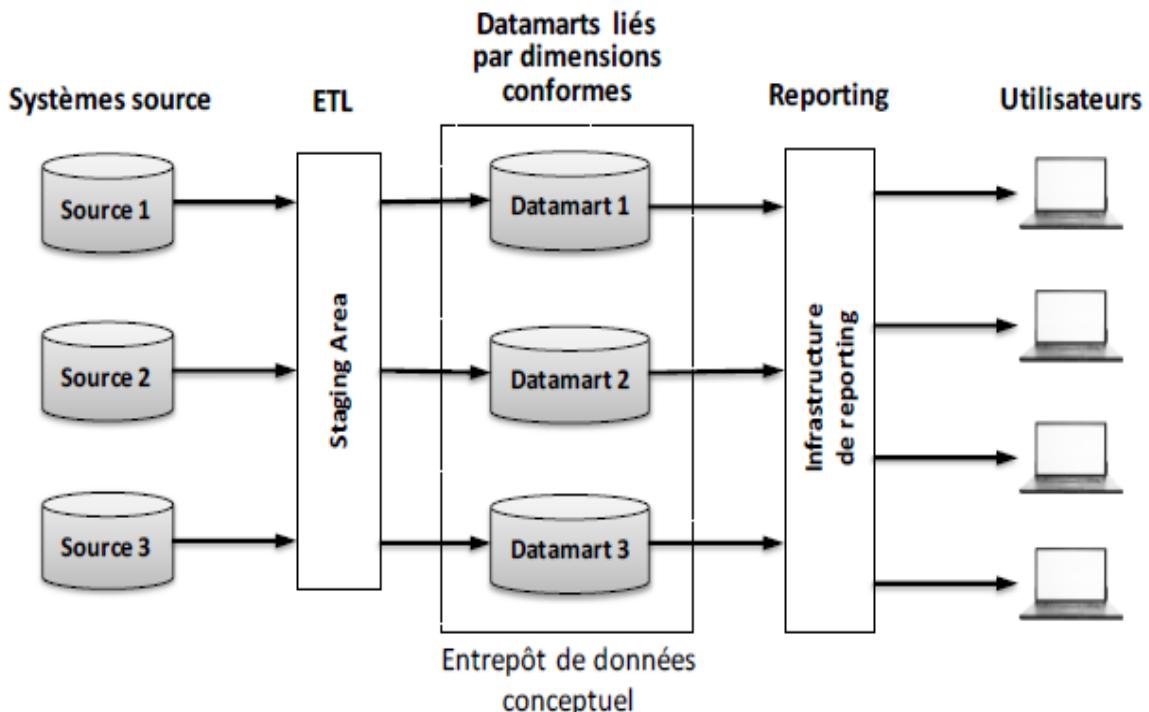


Figure 4 : L'approche ascendante ou par bus définie par Kimball. / Source : [R. Kimball et al., 2008]

6.1.3 Conception Hybride

L'approche hybride dérive des deux approches Top-Dow et Bottom-Up proposées par Inmon et Kimball respectivement. Pieter Mimmo, un consultant indépendant qui enseigne aux TDWI Conferences, est actuellement le plus ardent partisan de cette approche.

Wayne Eckerson, dans [W. Eckerson, 2003] la définit comme suit : « L'approche hybride tente de combiner le meilleur des approches Top-Down et Bottom-Up. Il tente de capitaliser sur la rapidité et l'orientation utilisateur de l'approche Bottom-Up sans sacrifier l'intégration imposée par un data warehouse dans une approche Top-Down. »

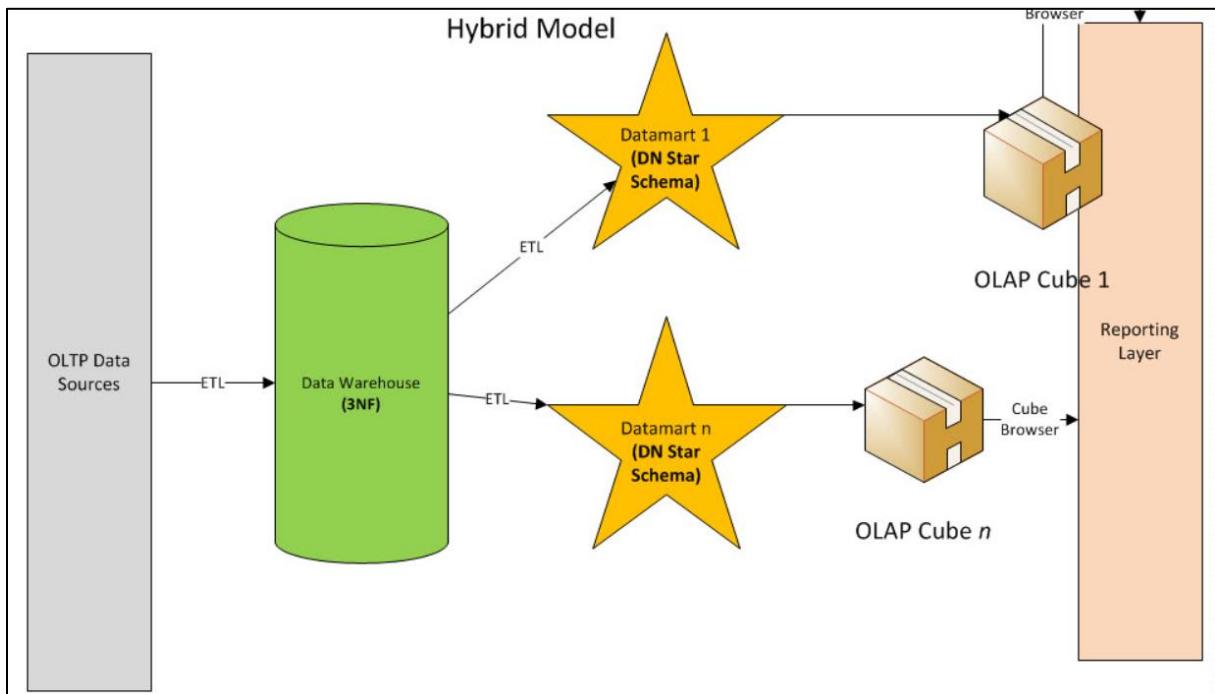


Figure 5 : L'approche hybride. | Source : [J. Serra, 2013]

Il nous explique que l'approche hybride repose sur un outil d'extraction, de transformation et de chargement (ETL) pour stocker et gérer l'entreprise et les modèles locaux dans les data marts et synchroniser les différences entre eux. Cela permet aux groupes locaux, par exemple, de développer leurs propres définitions ou règles pour les éléments de données dérivés du modèle d'entreprise sans sacrifier l'intégration à long terme. Les organisations utilisent également l'outil ETL pour extraire et charger les données des systèmes sources dans les data marts dimensionnels, tant au niveau atomique que sommaire.

Il ajoute qu'après avoir déployé les premiers data marts "dépendants", une organisation remplit ensuite un entrepôt de données derrière les data marts, instanciant la version "étoffée" du modèle de données de l'entreprise. L'organisation transfère ensuite les données atomiques des data marts vers l'entrepôt de données et consolide les flux de données redondants, économisant ainsi du temps, de l'argent et des ressources de traitement à l'organisation. Les entreprises remplissent généralement un entrepôt de données une fois que les utilisateurs professionnels demandent des vues de données atomiques sur plusieurs data marts.

6.1.4 Comment bien choisir ?

Les trois approches qu'on a décrites ci-dessus représentent les principales méthodologies d'entreposage de données et les plus dominantes. Dans n'importe quelle procédure de construction d'un entrepôt de données la connaissance de ces méthodologies est primordiale, sans autant y être fidèle à cent pour cent. Ces méthodologies ont façonné le débat sur les meilleures pratiques en matière d'entreposage des données et constituent les éléments constitutifs des méthodologies élaborées par les consultants praticiens.

Étant donné que chaque organisation doit répondre à des besoins et à des conditions commerciales uniques et différentes, le fait de disposer d'une base de modèles de pratiques exemplaires pour commencer promet des résultats positifs.

Ce qui nous mène à dire que le choix d'une de ses approches dépend fortement des objectifs commerciaux d'une organisation, de la nature des activités, du temps et des coûts impliqués, et du niveau de dépendance entre les différentes fonctions. Il est aussi à souligner que malgré une opposition entre ces modèles, aucun d'entre eux n'est à privilégier et le choix de la meilleure approche dépend de l'activité de l'entreprise ainsi que de ses objectifs à long et court terme sur le plan décisionnel.

Dans le cadre d'un nouveau projet décisionnel, il est nécessaire de mettre en place une solution rapidement. La démarche préconisée dans ce cas est de se focaliser sur la réalisation d'un premier Datamart, soit une approche orientée restitution très proche de celle proposée par Kimball [W. Eckerson, 2003].

6.2 Cycle de conception d'un ED

Plus tard dans notre travail, nous serons amenés à travailler sur des cubes de données, pour cela nous avons choisi de détailler les étapes de conception d'un ED selon l'approche Bottom-Up proposée par Kimball, c'est une approche qui met principalement en avant la modélisation multidimensionnelle des données. Par conséquent, nous allons présenter plus en détails le cycle de vie dimensionnel d'un ED selon l'approche dite « par cycle de vie » proposé par R. Kimball (Voir figure 6).

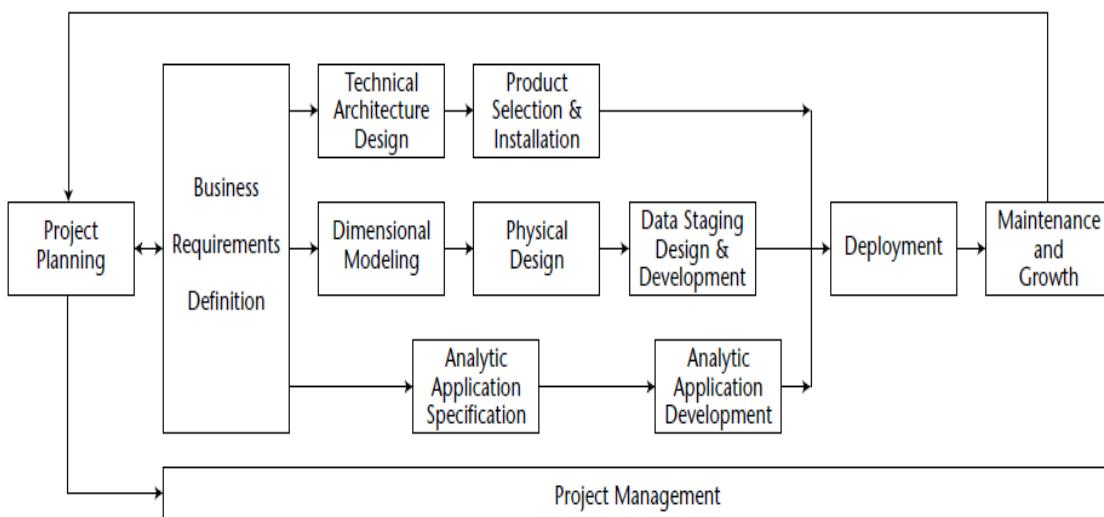


Figure 6 : Processus de conception d'un ED - Approche par cycle de vie. / Source : [M. Golfarelli et al., 2009]

La figure 6 représente le schéma global du cycle de vie d'un ED proposé par R. Kimball. Or, nous allons uniquement nous focaliser sur les parties qui nous intéressent dans la conception d'un ED, dont : La définition des besoins, la modélisation dimensionnelle, la conception physique et logique. Pour cela, nous avons proposé le schéma suivant (voir figure 7) qui représente les principales phases du cycle de vie d'un entrepôt de données. [M. Golfarelli et al., 2009].

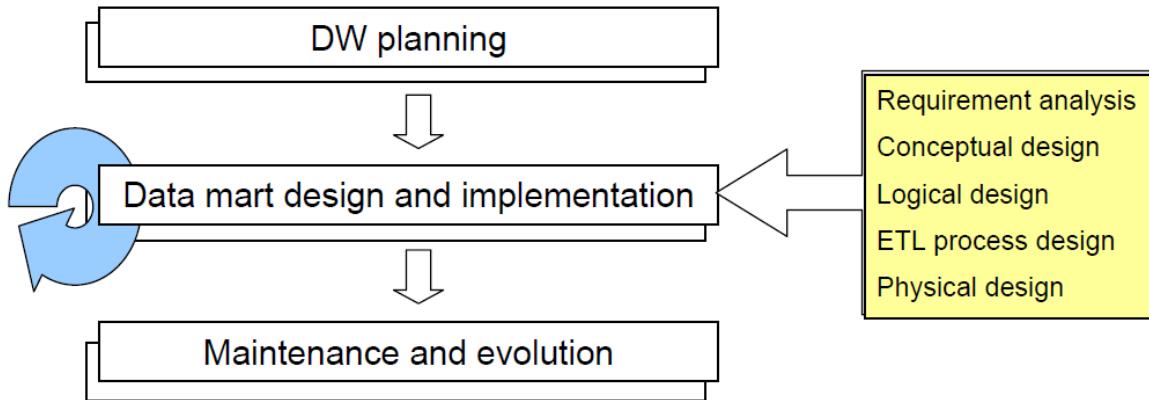


Figure 7 : Les principales phases du cycle de vie d'un ED. Source : [M. Golfarelli et al., 2009]

Bien que l'on ait beaucoup écrit sur la façon de concevoir un ED il n'y a pas encore de consensus sur une méthode de conception. La plupart des méthodes s'accordent sur l'opportunité de distinguer les phases suivantes [M. Golfarelli et al., 2009] :

6.2.1 Identification des besoins

L'analyse et l'identification des besoins représente l'étape la plus importante de tout projet d'entreposage de données. Elle permet de mieux identifier les sources internes et externes qui peuvent être utiles et répondre le mieux aux besoins des utilisateurs.

La définition des besoins consiste à définir de manière détaillée, l'ensemble des besoins et exigences des utilisateurs relatifs aux données et aux traitements. Parallèlement à la spécification des exigences relatives aux données, il est nécessaire de définir les exigences (fonctionnelles et non fonctionnelles) précisant l'ensemble des opérations et traitements que devra supporter l'application de l'ED. Cette étape consiste aussi à identifier les informations pertinentes pour le processus décisionnel en tenant compte soit des besoins des utilisateurs, soit de la disponibilité réelle des données dans les sources opérationnelles.

Les besoins sont généralement formulés sous forme de buts car ils expriment les objectifs des décideurs qui sont les principaux utilisateurs de l'ED.

La phase de définition des besoins comprend les quatre étapes suivantes : [I. Sommerville et al., 1998]

- La collecte des besoins : les besoins sont collectés auprès des utilisateurs finaux ou des clients du système.
- La spécification des besoins : consiste à formaliser les besoins et à identifier leurs caractéristiques attendues. L'étape de collecte des besoins fournit généralement un premier ensemble de besoins informels, inconsistants ou incomplets.

- La validation des besoins : consiste à valider le modèle obtenu lors de l'étape de spécification des besoins par les experts du domaine et les utilisateurs. Cette phase permet d'éviter la propagation des erreurs ou inconsistances des besoins durant les étapes de conception et d'implémentation de la base, qui peuvent s'avérer très coûteuses à corriger par la suite.
- Le suivie et la gestion des besoins : inclut toutes les activités permettant d'établir et de maintenir l'intégrité des besoins pendant que l'application de BD évolue.

6.2.2 Modélisation conceptuelle

Après avoir définis les besoins des décideurs, on passe à l'étape de modélisation de ces besoins sous format d'un schéma conceptuel d'ED. Cette phase prend comme entrée la spécification des besoins collectés auprès des utilisateurs du système. Elle est assurée par les analystes et concepteurs du système.

Cette étape vise donc à dériver un schéma conceptuel indépendant de la mise en œuvre d'un ED, en fonction du modèle conceptuel choisi. Celui-ci est amené à suivre le modèle multidimensionnel. Ce modèle identifie la granularité de la table des faits, les dimensions associées, les attributs ainsi que leur hiérarchisation.

6.2.3 Modélisation logique

Après avoir définis le schéma conceptuel, il est repris dans l'étape de modélisation logique pour créer un schéma logique correspondant. Alors qu'aujourd'hui la plupart des systèmes d'ED sont basés sur le modèle logique relationnel (ROLAP), un nombre croissant de fournisseurs de logiciels proposent également des solutions multidimensionnelles pures ou mixtes comprenant le Multidimensional Online Analytical Processing (MOLAP) et Hybrid Online Analytical Processing (HOLAP). Définis comme suit :

- **ROLAP** : fonctionne directement avec les bases de données relationnelles et ne nécessite pas de calcul préalable. Les données de base et les tables de dimension sont stockées sous forme de tables relationnelles et de nouvelles tables sont créées pour contenir les informations agrégées. Cela dépend d'une conception de schéma spécialisée. Cette méthodologie repose sur la manipulation des données stockées dans la base de données relationnelle pour donner l'apparence des fonctionnalités de découpage et de découpage d'OLAP traditionnelles.
- **MOLAP** : (traitement analytique en ligne multidimensionnel) est la forme classique d'OLAP et est parfois appelé OLAP uniquement. MOLAP stocke ces données dans un stockage multidimensionnel optimisé, plutôt que dans une base de données relationnelle.
- **HOLAP** : Le compromis indésirable entre le coût ETL supplémentaire et les performances de requête lentes a permis à la plupart des outils commerciaux OLAP d'utiliser désormais une approche "OLAP hybride" (HOLAP), qui permet au concepteur de modèle de décider quelle partie des données sera stockée dans MOLAP et quelle partie dans ROLAP.

6.2.4 Processus ETL (Extract, Transform, Load)

Les systèmes de gestion des données de l'entreprise sont hétérogènes autant sur le plan technique que sur le plan logique. Les données à collecter sont en effet stockées dans des systèmes de natures différentes, sous des formats différents, selon des structures différentes. Selon l'"histoire" du SI, les systèmes de stockage proviennent d'éditeurs différents. [A. Fernandez, 2017]

Il s'agit de la phase la plus essentiel dans un projet d'entreposage de données, le processus ETL se charge de faire l'intégration des sources hétérogènes et de l'unification des données sources dans un espace commun et uniforme afin d'assurer résultats de qualité et une exploitation efficace des données.

Le processus ETL utilise un ensemble de mappings composés d'une suite logique d'opérateurs d'*extraction*, de *transformation* et de *chargement*. Leur exécution permet d'intégrer et de transformer les données nécessaires selon un enchainement définis par les concepteurs des ED.

- Extraction :

Lors de l'extraction, les données souhaitées sont identifiées et extraites de nombreuses sources différentes, y compris des systèmes de base de données et des applications. Très souvent, il n'est pas possible d'identifier le sous-ensemble d'intérêts spécifique; par conséquent, il faut extraire plus de données que nécessaire, de sorte que l'identification des données pertinentes sera effectuée ultérieurement.

- Transformation :

Toutes les données ne sont pas utilisables telles quelles. Elle doit être vérifiées, reformatées, nettoyées afin d'éliminer les valeurs aberrantes, celles extérieures à la plage de vraisemblance et les doublons. Puis elles sont consolidées. Il s'agit aussi d'accorder un traitement particulier aux données manquantes.

- Chargement :

Insérer les données dans le Data Warehouse ou le Data Mart. Elles sont ensuite disponibles pour les différents outils d'analyse et de présentation que sont le Data Mining, l'analyse multimensionnelle OLAP, les analyses géographiques, les requêteurs et autres reportings et bien sûr les tableaux de bord.

6.2.5 Modélisation physique

La modélisation physique représente la dernière étape des principales phases du cycle de vie d'un ED. Cette étape aborde toutes les questions spécifiquement liées à la suite d'outils choisis pour la mise en œuvre - comme l'indexation et l'allocation.

7 Métadonnées des ED

7.1 Généralités sur les métadonnées

Nous allons présenter dans cette parties quelques définitions en relation avec les métadonnées :

7.1.1 Définition générale :

Les métadonnées sont souvent définies comme « les données sur des données » [E. Duval, 2001]. Il s'agit d'informations structurées qui décrivent, expliquent, localisent ou facilitent la recherche, l'utilisation ou la gestion d'une ressource d'information, particulièrement dans un environnement réseau distribué comme Internet ou une organisation [D.C. Moura, 1998]. Un bon exemple de métadonnées est le système de catalogage que l'on trouve dans les bibliothèques, qui enregistre par exemple l'auteur, le titre, le sujet et l'emplacement sur l'étagère d'une ressource.

7.1.2 Les métadonnées des ED :

Dans un ED, les métadonnées peuvent contenir des informations sur les sources et leurs contenus, mais aussi sur le schéma de l'ED lui-même, sur les règles de rafraîchissement et sur les profils et groupes d'usagers, etc.

7.2 Typologies des métadonnées dans les ED

Dans les ED, les métadonnées représentent une base primordiale à la construction de celui-ci, car sans eux il est difficile pour les utilisateurs de découvrir l'ensemble des sources de données, d'évaluer leur utilité et leur potentiel de réutilisation, et finalement de les réutiliser. On distingue aussi différents types de métadonnées dans les ED :

Kimball dans [R. Kimball et al., 2008] se réfère à trois grandes catégories de métadonnées :

- **Les métadonnées techniques (Technical metadata)** définissent les objets et les processus d'un système DW/BI, d'un point de vue technique. Les métadonnées techniques comprennent les métadonnées du système, qui définissent les structures de données telles que les tables, les champs, les types de données, les index et les partitions du moteur relationnel, ainsi que les bases de données, les dimensions, les mesures et les modèles de data mining.
- **Les métadonnées métier (Business metadata)** sont le contenu de l'entrepôt de données décrit en termes plus conviviaux. Les métadonnées métier vous indiquent les données dont vous disposez, d'où elles proviennent, ce qu'elles signifient et quelle est leur relation avec les autres données de l'entrepôt de données. Les métadonnées métier peuvent également servir de documentation pour le système DW/BI. Les utilisateurs qui naviguent dans l'entrepôt de données consultent principalement les métadonnées métier.
- **Les métadonnées de processus (Process metadata)** sont utilisées pour décrire les résultats de diverses opérations dans l'entrepôt de données. Dans le processus ETL, toutes les données clés des tâches sont enregistrées lors de l'exécution. Ceci inclut l'heure de début, l'heure de fin, les secondes CPU utilisées, les lectures de disque, les écritures de disque et les lignes traitées.

Tandis que Thomas Stöhr en décris deux qu'on juge très pertinent à citer [T. Stöhr et al., 1999]:

- **Les métadonnées techniques** : décrivant principalement le modèle physique et la couche technique des sources de données (ex : DBMS) et de l'ED lui-même.
- **Les métadonnées sémantiques** : Fondamentalement, les métadonnées sémantiques visent à fournir une description orientée métier du contenu de l'entrepôt de données. Cette description doit être compréhensible pour les utilisateurs qui ne sont pas familiers avec les descriptions de données techniques ou les langages de requête tels que SQL.

8 Conclusion

Dans ce chapitre nous avons abordé différentes notions autour de l'un des piliers de l'analyse décisionnel qui est l'entreposage de données. Les ED ont vite pris une place considérable dans les entreprises, vue l'intérêt énorme qu'il sollicite aux près des décideurs. Dans ce chapitre nous avons présenté certaines généralités autour de cette technologie, nous avons ensuite mis en évidence son architecture et sa conception qui comprend plusieurs phases dont principalement l'analyse des besoins des utilisateurs et les besoins en sources de données. La phase d'intégration représente une étape très importante dans l'entreposage de données. En effet, après avoir choisis les bonnes sources les sources doivent être extraite transformée ensuite chargée dans l'ED afin de les exploiter.

Ces différentes sources de données qu'elles soient internes ou externes, étaient traditionnellement de nature relationnelle (Ex : Base de données) ou des fichiers plats (Ex : CSV). Cependant, l'évolution et l'ampleur qu'a pris le web dans nos jours et la quantité énorme de données qu'il contient a pousser à l'apparition de nouvelles sources (notamment sémantiques) qui peuvent être considérées car elles sont de plus en plus disponibles sous forme de LOD qui se base sur les standards du web sémantique.

Pour cela, nous allons aborder dans le second chapitre le web sémantique et les Linked Open Data (LOD).

Chapitre 2: Web sémantique et Linked Open Data (LOD)

Introduction

Le World Wide Web (WEB) depuis son apparition à connue une évolution surprenante. Il est considéré par son créateur Tim Burners-Lee en 1989 comme la plus grande construction d'informations transformables. Le WEB est une idée qui lui revient d'origine mais beaucoup de progrès ont été réalisés sur le Web et les technologies connexes au cours des deux dernières décennies. Le Web 1.0 en tant que réseau de connaissance, le Web 2.0 en tant que réseau de communication, le Web 3.0 en tant que réseau de coopération et le Web 4.0 en tant que réseau d'intégration sont introduits tels que quatre générations du Web depuis son apparition. Le Web 3.0 a suscité notre attention dans ce travail, il est autrement connu par le web sémantique. Le web sémantique est une extension du web, il est constitué de standards visant à produire un format de données uniifié, il se base principalement sur le protocole Ressource Description Framework (RDF). « Le Web sémantique fournit un cadre commun qui permet de partager et de réutiliser les données entre les applications, les entreprises et les communautés ».

La naissance de la technologie du Web Sémantique a encouragé un très grand nombre de contributeurs de partager à licence libre, des une grande masse de données, suivant les règles du Web Sémantique. C'est l'apparition des données ouvertes et liées ou communément dit ; Linked Open Data (LOD).

Nous verrons dans cette partie les différentes notions liées au Web Sémantique et au LODs.

1 Web sémantique

1.1 Historique et évolution du WEB

Avant de parler du web sémantique, nous allons d'abord donner un aperçu de l'historique et de l'évolution du web depuis son apparition. Le World Wide Web (communément appelé le Web) n'est pas synonyme d'Internet, c'est plutôt « la partie la plus importante d'Internet qui peut être définie comme un système techno-social d'interaction humaine basé sur des réseaux technologiques. » [C. Fuchs et al., 2010].

Quatre génération du WEB on fait surface depuis l'apparition de celui-ci :

1.1.1 Web 1.0

C'est la première génération du web qui, selon Berners-Lee, pourrait être considérée comme le web en lecture seule et aussi comme un système de connaissance [B. Getting, 2007]. Le Web 1.0 a commencé comme un lieu d'information où les entreprises pouvaient diffuser leurs informations aux gens. Les premiers sites Web fournissaient un nombre limité d'interactions avec les utilisateurs ou de contributions au contenu et ne permettaient que d'effectuer des recherches et de lire l'information.

1.1.2 Web 2.0

Il a été défini par Dale Dougherty en 2004 comme un Web en lecture-écriture [T. Berners-Lee, 1998]. Les technologies du web 2.0 permettent de rassembler et de gérer de grandes foules mondiales ayant des intérêts communs dans les interactions sociales.

1.1.3 Web 3.0

Aussi connue par le **web sémantique**, a été suggéré par John Markoff du New York Times comme troisième génération du web en 2006.

On sait que le web actuel est un immense espace d'informations, dans lequel l'individu aura certainement besoin des machines pour l'aider à retrouver les informations qu'il recherche. Or la structure actuelle du Web ne facilite pas cette tâche, étant donné qu'elle avantage plus la compréhension humaine au détriment des machines [M. Mahdi Malik, 2002].

L'idée de base du Web 3.0 est de définir les données de structure et de les relier entre elles afin d'améliorer la découverte, l'automatisation, l'intégration et la réutilisation dans diverses applications [O. Nykänen, 2003]. Le Web 3.0 tente de relier, intégrer et analyser les données de divers ensembles de données pour obtenir de nouvelles informations. Il est capable d'améliorer la gestion des données, d'encourager l'accessibilité d'Internet mobile, de stimuler la créativité et de stimuler l'innovation, de favoriser la mondialisation et la collaboration dans les réseaux sociaux et de renforcer les clients pour les satisfaire [B. Getting, 2007].

1.1.4 Web 4.0

Représentera dans l'avenir proche un service Web à accès en lecture-écriture-exécution-concurrence comportant des interactions intelligentes, mais il n'existe toujours pas de définition exacte. Le Web 4.0 est également connu sous le nom de Web symbiotique dans lequel l'esprit humain et les machines peuvent interagir en symbiose [C. Fuchs et al., 2010].

1.2 Définitions

1.2.1 Le World Wide Web Consortium (W3C)

Tim Berners-Lee, inventeur du Web en 1989 et directeur du W3C définit le Consortium avec les mots suivants : « Nous avons créé un environnement neutre, capable de servir les intérêts de tous, depuis l'individu jusqu'aux plus grandes entreprises et aux états. La communauté industrielle en particulier a compris qu'il est de son intérêt de disposer d'un Web stable et évolutif, fondé sur un accord commun »

Le W3C (World Wide Web Consortium) est une équipe de chercheurs qui contient plus de 520 membres, il est intégré dans trois grandes institutions [M. Mahdi Malik, 2002] :

- L'INRIA en Europe.
- Le MIT en Amérique du Nord.
- L'Université de Keio en Asie.

Il possède en plus une dizaine de bureaux à travers le monde, et en particulier six au sein de la communauté européenne : Allemagne, Angleterre, Grèce, Italie, Pays-Bas et Suède. Ces bureaux sont extrêmement importants pour la promotion du W3C et de ses activités. Ils ont un rôle dans l'organisation de séminaires ou les experts de l'équipe technique du W3C sont invités à venir présenter les technologies du W3C. À noter que l'Europe toute seule regroupe plus du tiers de l'équipe technique du W3C [C. Rigaud-Alfano et al., 2001].

1.2.2 Web Sémantique

Le web sémantique a été conçu par Tim Berners-Lee, inventeur du World Wide Web. Il existe un travail important au sein du W3C pour l'amélioration, l'extension et la normalisation du web sémantique. Dans cette partie, nous allons présenter différentes définitions du web sémantique selon des points de vue différents.

Tim Berners-Lee dans l'un de ses articles définit le web sémantique de la manière suivante « Le Web sémantique est une technologie de partage de données, tout comme le Web hypertexte est une technologie de partage de documents. » [T. Berners-Lee et al., 2006]

On trouve aussi une définition émise par Brian Getting comme suit : « Le web sémantique est un web qui peut démontrer des choses que l'ordinateur peut comprendre. Le but principal du web sémantique est de rendre le web lisible par les machines et pas seulement par les humains ». Il le définit aussi comme « Le web sémantique peut être défini comme le web de données » [B. Getting, 2007].

Dans [C. Chiaramonti, 2001] on trouve la définition suivante, « C'est un concept dans lequel le web comporte des données définies et reliées entre elles à l'aide de règles d'inférence. Ces données doivent être utilisées par des machines non seulement à des fins d'affichage et de présentation, mais pour automatiser, intégrer et réutiliser ces données entre applications ».

On trouve enfin la définition suivante « L'expression Web sémantique, due à Tim Berners-Lee au sein du W3C, fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés » [J. Charlet, P. Laublet, C. Reynaud, 2003].

On constate que les définitions se rejoignent dans deux concepts fondamentaux du web sémantique, le partage et la liaison de données d'un côté et l'automatisation du contenu du web pour le rendre compréhensible par la machine de l'autre.

1.3 *Standards et langages du Web Sémantique*

Dans le Web Sémantique, l'interopérabilité des différentes ressources disponibles en ligne joue un grand rôle dans la conception de celui-ci. Réaliser ces interopérabilités est la définition de standards permettant d'offrir des langages formels communs à toutes les applications du WS.

Pour permettre à la machine de traduire le contenu du web à partir du web sémantique, il est nécessaire de traduire les données du web et de les coder avec des langages compréhensibles par la machine tout en leur donnant une sémantique. Dans ce qui suit nous allons voir deux grandes catégories de langages les langages existants sur lesquels le web sémantique s'est appuyé et les langages résultants des travaux du web sémantique.

1.3.1 *Langages existants sur lesquels le web sémantique s'est basé*

Le web sémantique s'est basé sur certains langages existants qui ont joué le rôle d'une base d'inspiration aux concepteurs du web sémantique. On trouve trois types de langages classifiés de la manière suivante :

- XML (Extensible Markup Language)

C'est considéré comme langage de base. Il a l'avantage d'être fait pour la communication en réseau et de disposer de nombreux outils. Il est donc naturellement utilisé pour encoder les langages du Web sémantique. Mais il a surtout la propriété d'être un métalangage (une description de type de document, DTD, permet de décrire la grammaire des documents admissibles). Par ailleurs, ceci ne permet pas à une machine de manipuler sémantiquement un document. Mais cela a la vertu de permettre une manipulation syntaxique de tous les documents. C'est cette propriété qui permet d'insérer des éléments du Dublin-core dans une ontologie et d'annoter des documents à l'aide de la connaissance formalisée [J. Charlet, P. Laublet, C. Reynaud, 2003].

- La représentation de connaissances et notamment les langages de représentation de connaissance

Représentent la seconde source d'inspiration des langages du web sémantique. On parle des connaissances que sont les logiques de descriptions et les réseaux sémantiques (que nous considérerons sous leur aspect plus avancé des graphes conceptuels). Ces langages permettent d'exprimer la connaissance de nature ontologique (décrire des classes d'entités, les relier par spécialisation, décrire et typer leurs attributs) ou assertionale (décrire l'état du monde par des individus en relations entre eux, individus et relations étant décrits dans l'ontologie) [J. Charlet, et al.,, 2003].

- Les langages de description de “Workflow”

Représentent les dernières sources d'inspiration. Ces langages permettant d'exprimer de manière abstraite des activités (ou tâches) et leurs dépendances (séquence, parallélisme, synchronisation...). Ces langages sont en général destinés à être supervisés par les humains qui exécutent les tâches du workflow, ils doivent donc acquérir plus de rigueur dans la description des tâches pour pouvoir être manipulés par des machines dans le cadre du Web sémantique [J. Charlet, P. Laublet, C. Reynaud, 2003].

1.3.2 Les couches du Web Sémantique

Pour mieux se bâtir une identité, le web sémantique s'est créé de nouveaux langages suivants un certain nombre de standards pour pouvoir arriver à la fin ultime de la 3ème génération du web. En effet, différents consortiums et organismes mettent donc des acteurs en action pour définir les langages à utiliser dans le Web sémantique. L'intérêt de cette approche de standardisation est bien sûr d'assurer des traitements uniformes sur l'ensemble des documents écrits dans ces langages.

Nous décrirons ici les standards du web sémantique aussi appelé les couches de l'architecture du web sémantique proposé par Tim Berners-Lee. La (figure 8) illustre les différentes couches de l'architecture et on précise que les couches du milieu dont RDF, RDFS, OWL et SPARQL représentent les langages du web sémantique.

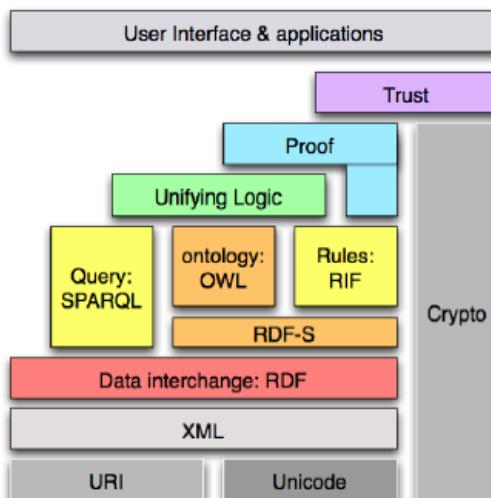


Figure 8 : Les couches du Web Sémantique. Source : [J. Jaffe, 2016]

Nous avons réparti les couches du web sémantique en trois catégories, la 1^{ère} représente les couches de base, la 2^{ème} représente les langages du web sémantique et la 3^{ème} représente les couches logiques.

- Les couches de base :
 - *Unicode* : C'est un encodage utilisé pour représenter n'importe quel caractère de façon unique quel que soit le caractère écrit et par n'importe quel langage [S. Aghaei, M.A Nematbakhsh, H.K. Farsani, 2012].
 - *URI (Uniform Resource Identifier)* : C'est un identificateur unique pour tous les types de ressources. La fonctionnalité de l'Unicode et de l'URI pourrait être décrite comme la fourniture d'un mécanisme d'identification unique au sein de la pile de langages pour le web sémantique. [S. Aghaei, M.A Nematbakhsh, H.K. Farsani, 2012]
 - *XML (Extensible Markup Language)* : Comme décrit dans la section précédente, XML est utilisé comme syntaxe de base pour d'autres technologies développées pour les couches supérieures du web sémantique. Plus précisément, XML et ses normes connexes, comme les espaces de noms (NS) et les schémas, sont utilisés pour former un moyen commun de structurer les données sur le Web. Ces espaces de noms (NS) sont utilisés pour identifier et distinguer différents éléments XML de différents vocabulaires [J. Charlet, P. Laublet, C. Reynaud, 2003].
- Les langages du web sémantique :

Dans cette partie nous allons citer brièvement les différents langages du web sémantique. Nous allons ensuite les détailler dans les points (1.3.3), (1.3.4), (1.3.5) et (1.3.6) du chapitre 2.

- RDF (Resource Description Framework)
- RDFS (Resource Description Framework Schéma)
- OWL (Web Ontology Language)
- SPARQL (Sparql Protocol and Rdf Query Langage)
- Les couches logiques :
 - *Logique et preuve* : Cette couche se trouve au-dessus de la structure ontologique pour faire de nouvelles inférences par un système de raisonnement automatique. Les agents peuvent déduire si des ressources particulières répondent à leurs besoins en utilisant de tels systèmes de raisonnement [J. Greenberg et al., 2003].
 - *Confiance* : La dernière couche de la pile traite de la confiance afin de fournir une assurance de la qualité de l'information sur le Web et un degré de confiance dans la ressource qui fournit cette information [S. Aghaei, M.A Nematbakhsh, H.K. Farsani, 2012].

Nous avons présenté ci-dessus une vue générale des couches et standard du web sémantique. Nous verrons plus en détails dans les points suivants les différents langages du web sémantique.

1.3.3 RDF (Resource Description Framework)

C'est un modèle de description de données simple qui utilise les URI pour identifier les ressources Web et décrit les relations entre les ressources en termes de propriétés et de valeurs nommées. Il sera utilisé pour annoter des documents écrits dans des langages non structurés, ou comme une interface pour des documents écrits dans des langages ayant une sémantique équivalente [J. Charlet et al., 2003]. En général, la famille RDF supporte l'interopérabilité au niveau sémantique. Les développements RDF sont

constitués du langage web de base, de sorte que les agents sont capables de faire des inférences logiques pour exécuter des fonctions basées sur les métadonnées [S. Aghaei, et al., 2012].

Un document RDF décrit des données dans lequel toute ressource est identifiée par une URI, et où l'on peut faire des assertions ou déclarations sur ces ressources sous la forme de triplets de la forme <sujet, prédicat, objet>. Dans ce triplet, le sujet et le prédicat sont toujours exprimés par des URIs (Universal Resource Identifiers), L'objet peut être exprimé sous la forme d'une URI ou d'une chaîne de caractère (littéral). Un ensemble de triplets RDF qui décrivent une ressource ou un ensemble de ressources composent un graphe.

Cet ensemble de triplets peut être représenté de façon naturelle par un graphe, où les éléments apparaissant comme sujet ou objet sont les sommets, et chaque triplet est représenté par un arc dont l'origine est son sujet et la destination son objet. Ce document sera codé en machine par un document RDF/XML, mais est souvent représenté sous une forme graphique (voir figure 9) [J. Charlet et al., 2003].

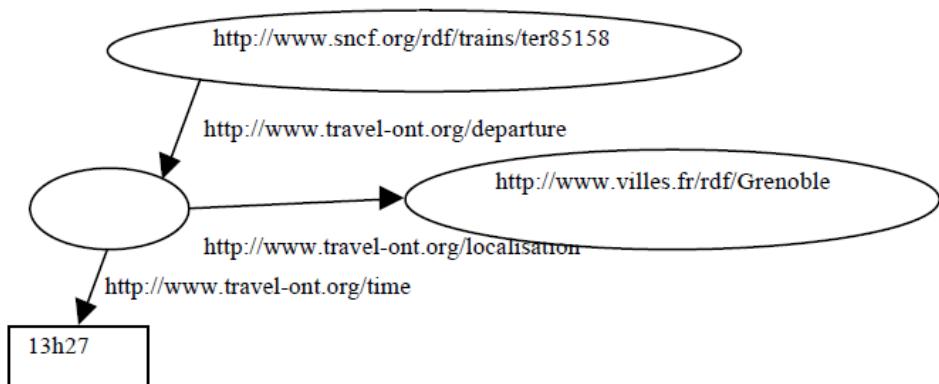


Figure 9 : Graphe RDF (Ensemble de triplets). Source : [J. Charlet et al., 2003]

La (figure 9) représente un exemple d'un graphe RDF qui décrit une partie d'un document RDF. Les termes de la forme http://... sont des URI qui identifient des ressources définies de façon unique. Notant que les URIs peuvent appartenir à des espaces de noms différents. Cette différence on peut la remarquer sur la représentation des URIs dans la (figure 9) On remarque que certaines ressources sont spécifiques à la SNCF (le train), et que d'autres (departure...) sont issus d'une ontologie dédiée aux voyages. Les objets d'un triplet qui sont des littéraux sont représentés dans un rectangle (ici, 13h27). Le sommet non étiqueté représente une variable. Intuitivement, ce graphe peut se comprendre comme « le train TER 85158 part de Grenoble à 13h27 ». Cette sémantique « intuitive » ne suffisant pas à un traitement automatique, il faut munir les documents RDF d'une sémantique formelle [J. Charlet et al., 2003].

Pour une visualisation plus simple nous proposons la représentation suivante d'un triplet :

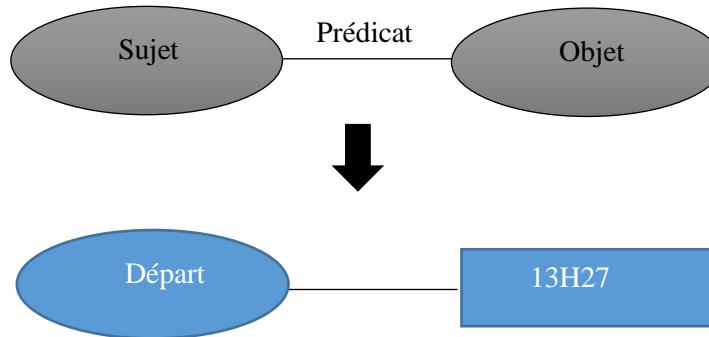


Figure 10 : Exemple d'un triplet RDF

1.3.4 RDFS (Resource Description Framework Schéma)

RDFS (Brickley et Guha, 1999; 2003) est un étendu et une extension de l'RDF de base. RDFS est le langage de description des vocabulaires associé à RDF, il permet de spécifier des ontologies dites « légères » [Gandon et al., 2012]. Plus précisément, il décrit les classes et les propriétés des ressources dans le modèle RDF de base. RDFS fournit un cadre de raisonnement simple pour déduire les types de ressources [S. Aghaei, et al., 2012]. Il a pour but d'étendre le langage RDF en décrivant plus précisément les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme permettant de spécifier les classes dont les ressources seront des instances, comme les propriétés.

Les classes permettent de définir la nature des ressources et les propriétés correspondent aux prédictats dans les triplets et permettent d'exprimer les relations entre les ressources [Bermès et al., 2013]. Autrement dit, RDFS est un ensemble de triplets RDF dont les données sont liées entre elles pour former un schéma comme illustré dans la (figure 11) [H. Melhem, 2017].

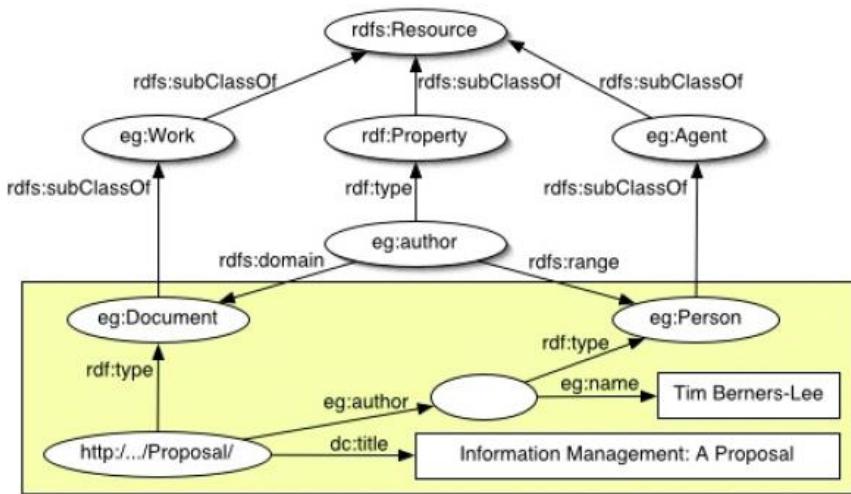


Figure 11 : Représentation schématique de RDFS. / Source : [H. Melhem, 2017]

RDFS nous permet de définir de nouveaux mots-clés. Pour les spécifier, nous allons les introduire dans l'exemple vue précédemment comme suit [J. Charlet et al., 2003] :

- *<ex:Vehicule rdf:type rdfs:Class>* la ressource ex:Vehicule a pour type rdfs:Class, et est donc une classe.
- *<sncf:TER8153 rdf:type ex:Vehicule>* la ressource sncf:TER8153 est une instance de la classe ex:Vehicule que nous avons définie.
- *<sncf:Train rdfs:subClassOf ex:Vehicule>* la classe sncf:Train est une sousclasse de ex:Vehicule, toutes les instances de sncf:Train sont donc des instances de ex:Vehicule.
- *<ex:localisation rdf:type rdfs:Property>* affirme que ex:localisation est une propriété (une ressource utilisable pour étiqueter les arcs).
- *<ex:localisation rdfs:range ex:Ville>* affirme que toute ressource utilisée comme extrémité d'un arc étiqueté par ex:localisation sera une instance de la classe ex:Ville.

RDFS permet de construire un langage primitif de description d'ontologies [Brunet et Xuan, 2010], tandis qu'une définition formelle et plus exhaustive des ressources relève des ontologies recommandées sous l'appellation OWL [H. Melhem, 2017].

1.3.5 OWL

Le langage OWL est utilisé dans la couche ontologie. Le langage OWL est survenu suite aux besoins de spécifier davantage les classes et remplir les lacunes du modèle RDFS. OWL vient alors pour spécifier détailler et compléter le langage RDFS. Il est dédié aux définitions de classes et de types de propriétés, et donc à la définition d'ontologies.

Le langage OWL est une recommandation du w3c permettant de définir des ontologies web. Il est basé sur la syntaxe de RDF/XML, OWL sert à décrire un modèle de données représentatif de l'ensemble des concepts dans un domaine donné [H. Melhem, 2017]. La particularité de ce modèle tient au fait qu'il est exploitable par les machines et les humains. Le langage OWL par opposition à RDF Schema permet de définir un cadre formel et détaillé d'un domaine de connaissances en utilisant des contraintes appliquées aux classes et propriétés [Bermès et al., 2013].

Inspiré des logiques de descriptions, il fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les propriétés des classes définies. La rançon de cette expressivité est l'indécidabilité du langage obtenu en considérant l'ensemble de ces constructeurs. C'est pour cela que OWL a été fractionné en trois langages distincts :

OWL permet, grâce à sa sémantique formelle basée sur une fondation des logiques descriptives, de fournir un grand nombre de constructeurs permettant d'exprimer de façon très fine les propriétés des classes définies. Il permet de définir des associations plus complexes des ressources. Cette forte expressivité de OWL mène à l'indécidabilité du langage obtenu en considérant l'ensemble de ces constructeurs. C'est pour cela que OWL a été fractionné en trois sous langages, du moins expressif au plus expressif : OWL-Lite, OWL-DL et OWL-Full décrits comme suis selon [J. Charlet et al., 2003].

- OWL LITE :

Il ne contient qu'un sous-ensemble réduit des constructeurs disponibles, mais son utilisation assure que la comparaison de types pourra être calculée (un problème de NP, donc « simple » en représentation de connaissances).

Nous donnons ici l'ensemble des constructeurs utilisés dans OWL LITE [J. Charlet et al., 2003] :

- Reprend tous les constructeurs de RDF
- Utilise les mots-clés de RDFS (rdfs:subClassOf, rdfs:Property, rdfs:subPropertyOf, rdfs:range, rdfs:domain), avec la même sémantique.

- Permet de définir une nouvelle classe (owl:Class) comme étant plus spécifique ou équivalente à une intersection d'autres classes
- owl:sameIndividualAs et owl:differentIndividualFrom permettent d'affirmer que deux individus sont égaux ou différents.
- Des mots-clés permettent d'exprimer les caractéristiques des propriétés : owl:inverseOf sert à affirmer qu'une propriété p est l'inverse de p' (dans ce cas, le triplet <s p o> a pour conséquence <o p' s>) ; d'autres caractéristiques sont par exemple la transitivité (owl:TransitiveProperty), la symétrie (owl:SymmetricProperty).
- owl:allValuesFrom associe une classe C à une propriété P. Ceci définit la classe des objets x tels que si <x P y> est une relation, alors la classe de y est C (quantification universelle de rôle en logique de descriptions). owl:someValuesFrom encode la quantification existentielle de rôle.
- owl:minCardinality (resp. owl:maxCardinality) associe une classe C, une propriété P, et un nombre entier n. Ceci définit la classe des objets x tels qu'il existe au moins (resp. Au plus) n instances différentes y de C avec <x P y>. Pour des raisons d'efficacité algorithmique, OWL LITE ne permet d'utiliser que des entiers égaux à 0 ou 1. Cette restriction est levée dans OWL DL.

▪ OWL DL :

Il contient l'ensemble des constructeurs, mais avec des contraintes particulières sur leur utilisation qui assurent la décidabilité de la comparaison de types. Par contre, la grande complexité de ce langage (un de ses fragments est P-SPACE-complet) semble rendre nécessaire une approche heuristique.

Nous donnons ici l'ensemble des constructeurs utilisés dans OWL DL [J. Charlet et al., 2003] :

- Reprend tous les constructeurs d'OWL LITE.
- Permet tout entier positif dans les contraintes de cardinalité,
- owl:oneOf permet de décrire une classe en extension par la liste de ses instances.
- owl:hasValue affirme qu'une propriété doit avoir comme objet un certain individu.
- owl:disjointWith permet d'affirmer que deux classes n'ont aucune instance commune.
- owl:unionOf et owl:complementOf permettent de définir une classe comme l'union de deux classes, ou le complémentaire d'une autre classe.

▪ OWL FUL :

Sans aucune contrainte, pour lequel le problème de comparaison de types est vraisemblablement indécidable.

Nous donnons ici l'ensemble des constructeurs utilisés dans OWL FULL [J. Charlet et al., 2003] :

- Reprend tous les constructeurs d'OWL DL,
- Reprend tout RDF Schema,
- Permet d'utiliser une classe en position d'individu dans les constructeurs.

1.3.6 Langage d'interrogation des données sémantiques (SPARQL)

SPARQL (Sparql Protocol and Rdf Query Langage) est un protocole et un langage de requête pour les données RDF. C'est une recommandation conçue par le W3C datant de janvier 2008 et une des technologies du web sémantique. La puissance de ce langage d'interrogation réside dans sa capacité à interroger plusieurs bases de données. SPARQL prend également en charge l'agrégation, les sous requêtes, la négation, la création de valeurs par des expressions, le test de valeur extensible et la contrainte des requêtes RDF [H. Melhem, 2017]. Des terminaux sous forme de services web, connue par SPARQL ENDPOINT ont été créés pour permettre l'exécution des requêtes SPARQL sur des bases de données sémantiques comme DBPEDIA.

Selon Tim Berners-Lee, SPARQL facilite l'interrogation des informations issues des bases de données et d'autres sources répandues sur le Web. Selon W3C, SPARQL est conçu pour être utilisé à l'échelle du Web et donc faciliter les interrogations des sources de données distribuées, quel que soit leur format. Un point fort de ce langage d'interrogation est sa capacité à faire une recherche en se basant sur les liens décrits par les triplets RDF. Une recherche peut amener l'utilisateur à découvrir des informations auxquelles il n'avait pas pensé, cela grâce aux règles d'inférences [H. Melhem, 2017].

Pour mieux illustrer le protocole SPARQL, nous avons pris un exemple de requête SPARQL qui interroge la base de données sémantique DBPEDIA, comme suit : `SELECT DISTINCT ?predicat ?objet WHERE {<http://dbpedia.org/ontology/architect> ?predicat ?objet}`.

Pour ce faire, nous avons utilisé « Virtuoso SPARQL Query Editor », un service web en ligne représentant un SPARQL endpoint qui nous permet d'exécuter des requêtes sur des BDD sémantique (voir figure 12).

Figure 12 : Exécuter une requête SPARQL sur «Virtuoso SPARQL Query Editor ». Source : [\[https://dbpedia.org/sparql\]](https://dbpedia.org/sparql)

La requête exécutée ci-dessus, nous permet d'afficher tous les prédictats et les objets dont le sujet est « *Architect* » représenté par la ressource (URI) suivante : `<http://dbpedia.org/ontology/architect>`.

En lançant la requête, nous obtiendrons le résultat illustré dans la figure ci-après.

predicat	objet
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/1999/02/22-rdf-syntax-ns#Property
http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/2002/07/owl#ObjectProperty
http://www.w3.org/2000/01/rdf-schema#subPropertyOf	http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#coparticipatesWith
http://www.w3.org/2002/07/owl#equivalentProperty	http://www.wikidata.org/entity/P84
http://www.w3.org/2000/01/rdf-schema#label	"architecte"@fr
http://www.w3.org/2000/01/rdf-schema#label	"Architekt"
http://www.w3.org/2000/01/rdf-schema#label	"architett" @ga
http://www.w3.org/2000/01/rdf-schema#label	"architect" @en
http://www.w3.org/2000/01/rdf-schema#label	"architect" @nl
http://www.w3.org/2000/01/rdf-schema#label	"architekt" @pl
http://www.w3.org/2000/01/rdf-schema#label	"architetto"
http://www.w3.org/2000/01/rdf-schema#label	"архитекторъ" @el
http://www.w3.org/2000/01/rdf-schema#label	"архитекта" @sr
http://www.w3.org/2000/01/rdf-schema#label	"архитектор" @ru
http://www.w3.org/ns/prov#wasDerivedFrom	http://mappings.dbpedia.org/index.php/OntologyProperty:architect
http://www.w3.org/2000/01/rdf-schema#domain	http://dbpedia.org/ontology/ArchitecturalStructure
http://www.w3.org/2000/01/rdf-schema#range	http://dbpedia.org/ontology/Architect

Figure 13 : Le résultat d'une requête SPARQL sur « Virtuoso SPARQL Query Editor ». Source : [\[https://dbpedia.org/sparql\]](https://dbpedia.org/sparql)

Conclusion

Dans le web sémantique il y a les données représentées par les différents langages du Web Sémantique et il y a les modèles de données, qui eux donne une certaine structure à ces données, pour les exploiter plus facilement. Afin de décrire le modèle (d'un domaine, d'une application, etc.), de façon partagée tout en présentant un consensus d'experts, nous définissons des « *Ontologies* ».

1.4 Les ontologies

Le terme ontologie est très rependu dans nos jours, que ce soit dans le domaine informatique dans l'ingénierie des connaissances ou dans la philosophie. Nous nous intéressant de façon très approfondie aux ontologies du web sémantique, on les considère comme une des bases de notre travail. Dans cette partie nous allons voir les principales notions des ontologies.

1.4.1 Définitions

La notion d'ontologie semble ambiguë pour certains, donc pour comprendre son sens et son utilisation dans le domaine du web des données principalement, nous avons proposé certaines définitions qui se rapportent au sujet de notre travail.

La 1^{ère} définition qu'on a jugé intéressante à présenter définit l'ontologie comme une « représentation des connaissances terminologiques relatives à un domaine, agréée par une communauté de personnes et sensée en faciliter le partage. Une ontologie a une taxonomie et un ensemble de règles d'inférence (raffinement, décomposition, prédiction, relativité, similarité) »¹

On note que la communauté de représentation des connaissances a adopté le terme d'ontologie dans leur domaine pour désigner l'objet issu d'un processus de modélisation de connaissances permettant de représenter d'une façon formelle un domaine donné [H. Melhem, 2017]. Cette définition cite des caractéristiques importantes des ontologies qui sont : les taxonomies et les règles d'inférences. On décrira ces deux caractéristiques plus en détails dans les points suivants.

La 2^{ème} définition d'une ontologie est une définition consensuelle introduite par Gruber [Gruber, 1993] et étendue par Borst [Borst, 1997] « une ontologie est une spécification formelle explicite d'une conceptualisation partagée d'un domaine donné ».

¹ Cacaly, Serge et al. (2004). Dictionnaire de l'information. Paris : Armans Colin

Cette définition s'appuie sur deux dimensions : la conceptualisation qui permet de décrire d'une manière explicite les vocabulaires d'un domaine donné, et la spécification qui consiste à utiliser un langage formel pour cette description [H. Melhem, 2017].

La 3^{ème} définition vient compléter, synthétiser les définitions précédentes en se focalisant sur les ontologies du web sémantique. On constate alors que selon [G. Pierra, 2003] une ontologie est « une représentation formelle, explicite, référençable et consensuelle de l'ensemble des concepts partagés d'un domaine sous forme de classes, de propriétés et de relations qui les lient »

Cette définition souligne des points importants, spécifiant ainsi les caractéristiques d'une ontologie :

- **Formelle** : exprimée dans un langage de syntaxe et de sémantique formalisé (RDF Schéma (Brickley et al., 2002), DAML+OIL (Connolly et al., 2001), OWL (Bechhofer et al., 2004), PLIB(ISO-13584-42, 1998), etc.) permettant ainsi des raisonnements automatiques ayant pour objet soit d'effectuer des vérifications de consistance, soit d'inférer de nouveaux faits.
- **Explicite** : l'ensemble des concepts et propriétés d'une ontologie sont spécifiés explicitement indépendamment d'un point de vue particulier ou d'un contexte implicite.
- **Référençable** : signifie que tout concept de l'ontologie peut être référencé de manière unique afin d'expliquer la sémantique de l'élément référencé.
- **Consensuelle** : signifie que l'ontologie est admise et acceptée par l'ensemble des acteurs d'une communauté.

La 4^{ème} et la dernière définition qu'on va présenter met l'accent sur deux des notions clés dans une ontologie, elle « inclut une organisation hiérarchique des concepts pertinents et des relations qui existent entre ces concepts » [Gandon, 2012 p. 84].

Il s'agit donc de représenter un domaine de connaissance par un vocabulaire spécifique. Ce vocabulaire est formé d'un ensemble de concepts et des relations qui les unissent. De ce fait, la composition d'une ontologie dépendrait du choix des concepts [H. Melhem, 2017].

1.4.2 Les notions clés dans une ontologie

Comme cité ci-dessus, une ontologie possède des notions clés qui la constituent. En effet, malgré la multitude de modèles ontologiques existants, ils se reposent tous sur les mêmes notions suivantes [S. Khouri, 2013] :

- **Le concept** : est définie comme « une idée générale désignée par un mot » [H. Melhem, 2017]. Il est composé de trois éléments distincts :
 - **Le terme** : exprimant le concept en langage naturel.
 - **Notion ou intension du concept** : la signification du concept.
 - **Les objets dénotés par le concept** : appelés également réalisations ou extensions du concept.
- **Les classes** : décrivent les concepts d'un domaine. Une classe peut avoir des sous-classes et des instances représentant les extensions du concept.
- **Les attributs** : décrivent les propriétés des classes et des instances de l'ontologie.
- **Les relations** : désignent les associations définies entre les concepts de l'ontologie.
- **Les axiomes** : ce sont des assertions dites vraies dans le domaine en question. Les axiomes et les règles permettent de vérifier la cohérence d'une ontologie, et aussi d'inférer de nouvelles connaissances.

1.4.3 Taxonomies des ontologies

Une taxonomie représente une classification dans un domaine précis. Dans cette section nous allons présenter les typologies et classifications des ontologies. Les ontologies peuvent être classées selon le type de concepts ou selon leur objectif de conceptualisation.

1. Classes d'ontologies selon le type des concepts :

L'usage des ontologies diffère d'une communauté à une autre selon les domaines de spécialité. Elle a initialement été utilisée par la communauté de l'intelligence artificielle pour la gestion des connaissances, les ontologies ont ensuite été exploitées dans le domaine de la linguistique et celui des bases de données [G. Pierra, 2003]. L'usage des ontologies diffère d'une communauté à une autre selon la façon de modéliser un domaine. On a divisé alors les ontologies selon la manière de les conceptualiser comme suit :

- **Dans le domaine de la linguistique**, les ontologies décrivent et manipulent des mots et les relations entre ces mots tels que la synonymie et l'antonymie. Ces ontologies sont dites ontologies linguistiques (OL) [G. Pierra, 2003]. Wordnet² est l'une des OL les plus connues.
- **Dans le domaine des bases de données et celui de l'intelligence artificielle**, on s'intéresse aux ontologies dites conceptuelles qui manipulent des concepts et non des mots. [Gruber, 1993] distingue deux types de concepts dans une ontologie conceptuelle :
 - **Concepts primitifs (ontologies conceptuelles canoniques)** : qui représentent des concepts ne pouvant être définis par une définition axiomatique complète [Gruber, 1993]. Les concepts primitifs, sont utilisés dans le domaine des bases de données. Ces concepts, possèdent une représentation unique. Ils sont utiles pour la conception des BD où les redondances sont exclues, et également dans les formats d'échange où il doit exister une seule représentation possible pour chaque information échangée [S. Khouri, 2013]. Les ontologies ne contenant que des concepts primitifs sont appelées **ontologies conceptuelles canoniques (OCC)** [G.Pierra, 2003]. Par exemple, on peut considérer une OCC avec une classe Personne caractérisée par deux propriétés Nom et Sexe (masculin ou féminin) [S. Khouri, 2013].
 - **Concepts définis (non canoniques)** : qui sont définis à base d'autres concepts canoniques ou non canoniques (primitifs ou définis) par des relations d'équivalence entre concepts [Gruber, 1993]. Les concepts définis sont utilisés dans le domaine de l'intelligence artificielle, afin d'avoir la possibilité de faire des déductions [S. Khouri, 2013]. Les ontologies contenant des concepts primitifs et définis sont appelées **ontologies conceptuelles non canoniques (OCNC)** [G.Pierra, 2003]. En revenant à l'exemple précédent du concept primitif Personne qu'on spécialise par les deux concepts définis Homme (Personne de sexe masculin) et Femme (Personne de sexe féminin), nous obtenons les concepts d'une OCNC [S. Khouri, 2013].

Cet usage différent des ontologies a été matérialisé dans une taxonomie en couches proposée dans [G. Pierra, 2003] illustrée par le modèle en oignon dans la (figure 14).

² <http://wordnet.princeton.edu/>

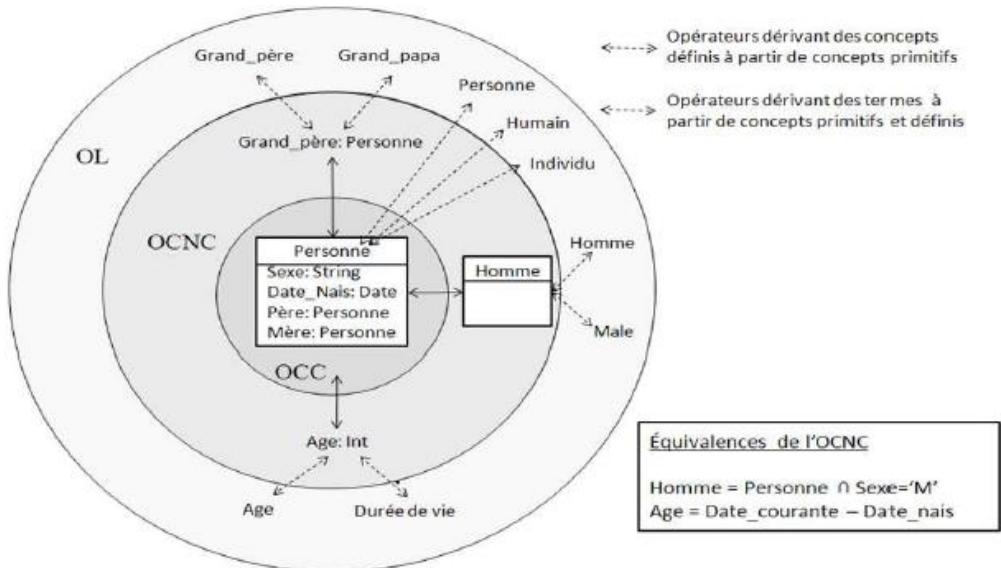


Figure 14 : Modèle en oignon. Source [G. Pierra, 2003]

Les relations entre les deux couches conceptuelle et linguistique sont définies dans ce modèle, où à chaque concept de la couche conceptuelle, on peut correspondre plusieurs termes de la couche linguistique (dans la même langue ou dans différentes langues) [Z. Djilani, 2017].

2. Classes d'ontologies selon l'objet de conceptualisation

Cette partie concerne les ontologies conceptuelles. Les ontologies peuvent être classées selon l'objet de conceptualisation en six types présenté dans [H. BENHMIDI et al., 2011] : ontologie de représentation des connaissances, ontologie supérieure ou de haut niveau, ontologie générique, ontologie de domaine, ontologie de tâches et ontologie d'application. Nous nous intéressons dans notre travail aux ontologies de domaine, qui sont définies comme suit :

Ontologie du domaine (Domain ontology) Cette ontologie exprime des conceptualisations spécifiques à un domaine, elle est pour plusieurs applications de ce domaine [H. BENHMIDI et al., 2011]. Ces ontologies sont alors définies comme des ontologies qui décrivent le vocabulaire correspondant à un domaine particulier, en spécialisant par exemple les concepts d'une ontologie de haut niveau. L'ontologie du domaine caractérise la connaissance du domaine où la tâche est réalisée [R. Mizoguchi et al., 2000].

1.4.4 Outils de construction d'ontologies

Après l'apparition des ontologies, plusieurs outils et éditeurs ont été créés pour manipuler les ontologies. Le plus connus sont : Protégé, ODE et WebODE, OntoEdit, WebOnto et beaucoup d'autres.

Dans notre travail nous serons amenés à utiliser *Protégé*³ comme outil de construction et d'édition d'ontologies. *Protégé* est une interface modulaire permettant l'édition, la visualisation, le contrôle d'ontologie, l'extraction d'ontologies à partir de sources textuelles, et la fusion semi-automatique d'ontologies. Il autorise la définition de métaclasses, dont les instances sont des classes, ce qui permet de créer son propre modèle de connaissances avant de bâtir une ontologie [H. BENHMIDI et al., 2011].

³ <https://protege.stanford.edu/>

Conclusion

Comme vue dans cette section le Web Sémantique est représenté en tant qu'extension du Web dans laquelle les données, en plus d'avoir une signification pour les humains, ont également une signification pour les machines de manière à ce qu'elles puissent être recherchées, trouvées, interprétées, partagés et réutilisés entre applications, organisations et communautés.

Le Web sémantique possède plusieurs concepts, l'un des plus importants est les Linked Open Data (LOD) Linked Open Data (LOD). Dans la section suivante nous allons voir plus en détails les LOD.

2 **Linked Open Data (LOD)**

2.1 **Définitions**

Le Linked Open Data ou Données Ouvertes Liée consiste à rendre libre de toute licence des bases de données préalablement liées entre elles selon le modèle du Linked Data de Tim Berners-Lee. Nous verrons ci-dessous les définitions des concepts primaires sur lesquels s'est basé les LOD :

- Open Data :

Les *Open Data* ou *données ouvertes* sont l'idée que certaines données devraient être librement accessibles à tous pour être utilisées et republiées à leur guise, sans restrictions du droit d'auteur, des brevets ou d'autres mécanismes de contrôle. Les objectifs du mouvement de données open-source sont similaires à ceux d'autres mouvements "open-source" tels que les logiciels open-source. [S. Auer, 2007]

- Linked Data :

C'est une donnée structurée qui est interconnectée avec d'autres données afin de devenir plus utile par le biais de requêtes sémantiques. Il s'appuie sur les technologies Web standard telles que HTTP, RDF et les URI, mais plutôt que de les utiliser pour servir des pages Web uniquement à des lecteurs humains, il les étend pour partager des informations de manière à pouvoir être lues automatiquement par des ordinateurs. Une partie de la vision des données liées consiste pour Internet à devenir une base de données mondiale. [C. Bizer et al., 2011]

- Linked Open Data :

Définit une vision des données Internet globalement accessibles et liées, basée sur les standards RDF du Web sémantique. LOD est souvent considéré comme un nuage de données virtuel dans lequel n'importe qui peut accéder à toutes les données qu'il est autorisé à consulter et peut également ajouter des données sans perturber la source de données d'origine. Cela fournit un environnement ouvert où les données peuvent être créées, connectées et utilisées à l'échelle Internet. Une théorie de base du niveau de détail est que les données ont plus de valeur si elles peuvent être connectées à d'autres données. Les données, dans ce contexte, sont toute information structurée basée sur le Web [T. Berners-Lee, 2006].

Par exemple, DBPEDIA⁴ représente un des principaux portails LOD qui fournit la masse d'informations, c'est une base de données sémantique contenant des données ouvertes et liées qui regroupe une grande partie du contenu de Wikipédia.

2.2 **Evolution des LOD**

Le linked open data est étroitement lié à l'essor du web sémantique. Ce dernier est à la base une idée de l'inventeur même du web, Tim Berners-Lee, qui publie en 2001 un article présentant le concept de web sémantique. L'idée présentée peine à percer et à se concrétiser, sans doute à cause de la complexité technique sous-jacente. En 2006, Berners-Lee publie un second article, intitulé simplement Linked Data [T. Berners-Lee, 2006]. Le concept commence alors à se populariser et gagne en visibilité.

C'est après que le projet Linking Open Data community à vue le jour. L'objectif du groupe W3C, initiateur du projet, est d'étendre le Web avec une communauté de données communes en publiant divers ensembles de données ouvertes sous forme de données RDF sur le Web et en établissant des liens RDF entre des éléments de données provenant de différentes sources.

⁴ <https://wiki.dbpedia.org/>

En octobre 2007, les ensembles de données comprenaient plus de deux milliards de triples RDF, qui étaient reliés entre eux par plus de deux millions de liens RDF (Voir figure 15). En septembre 2011, ce chiffre était passé à 31 milliards de triples RDF, reliés par environ 504 millions de liens RDF. Une ventilation statistique détaillée a été publiée en 2014⁵.

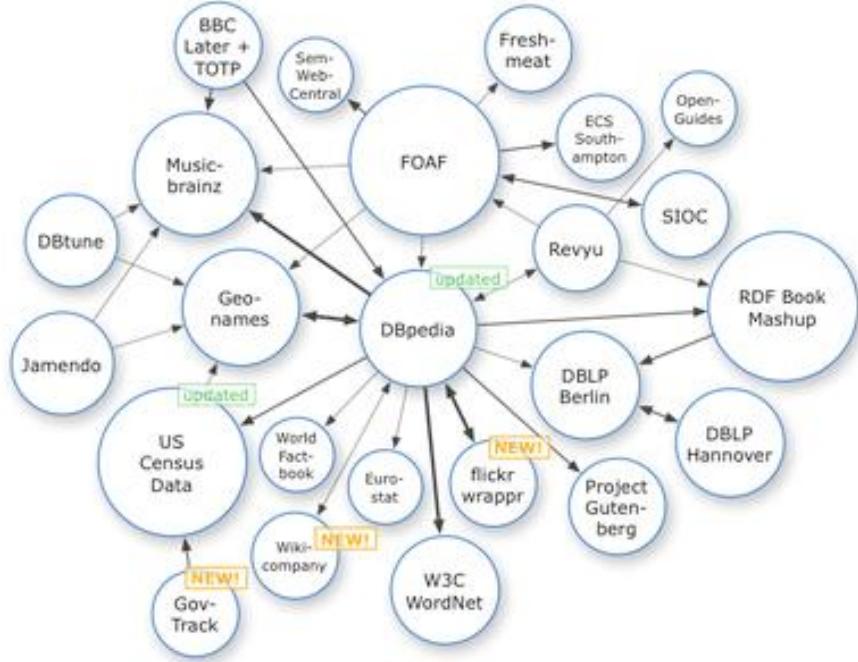
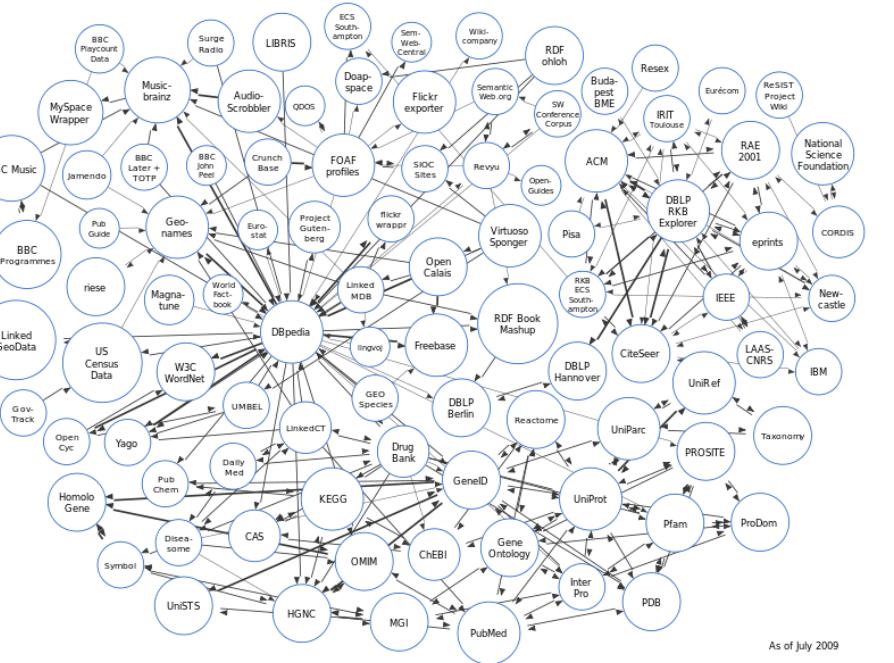


Figure 15 : Représentation en carte heuristique des relations entre les données ouvertes de DBpedia et divers autres.
Source : <https://lod-cloud.net/>.



⁵ <http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/>

Figure 16 : l'évolution des relations entre DBpedia et divers autres projets du Web de la représentation précédente en 2009. Source : <https://lod-cloud.net/>.

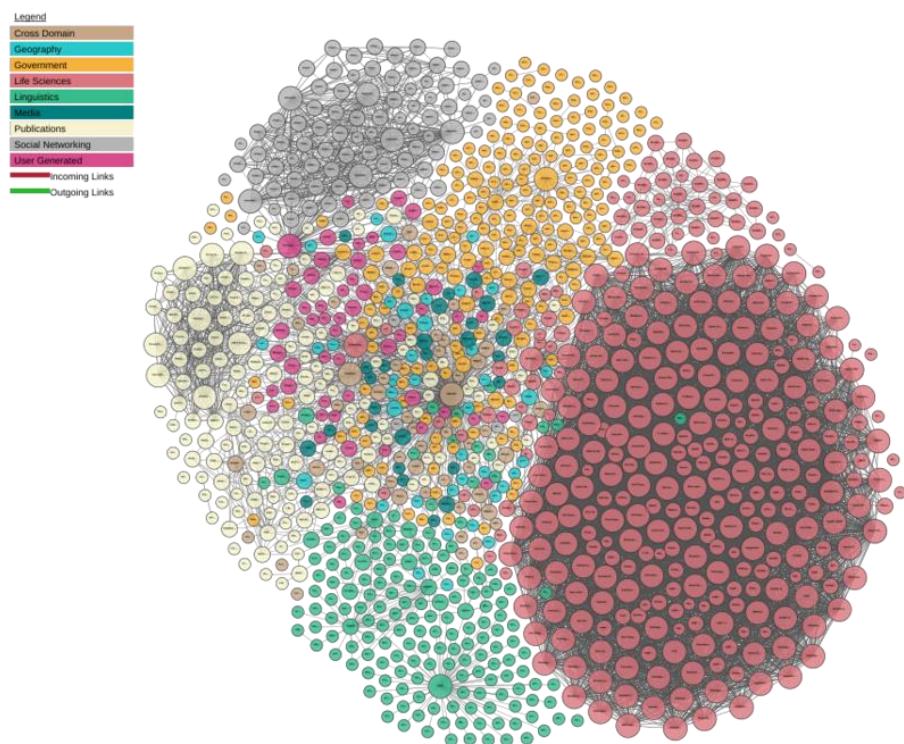


Figure 17 : Le même représentation précédente, mais pour février 2017, montrant la croissance en seulement quelques années. Source : <http://lod-cloud.net/>

Les figures ci-dessus représentent l'évolution des datasets LOD à partir de leur apparition en 2007, puis leur évolution en 2009 et ensuite en 2017. On note que le web de données et les relations entre différents datasets LOD sont en évolution continue. En effet, ce domaine prend de plus en plus d'ampleur.

2.3 Principes du LOD

Le Linked Open Data repose sur 4 principes de base essentiels⁶, pour qualifier une source de données qu'elle est de nature LOD. On présente donc ces caractéristiques comme suit⁷ :

- **Principe 1 :** Utiliser des URI comme noms de choses. Toute données doit être précédé de l'URI complet. Si on utilise pas l'ensemble de symboles URI universel, nous ne l'appelons pas Web sémantique.
- **Principe 2 :** Utiliser des URI HTTP pour que les utilisateurs puissent rechercher ces noms. Impliquent généralement de ne pas vouloir s'engager dans le système de noms de domaine (DNS) établi pour la délégation de pouvoirs, mais de construire quelque chose sous un contrôle séparé.

⁶ <https://vimeo.com/36752317>

⁷ <https://www.w3.org/DesignIssues/LinkedData.html>

- **Principe 3 :** Utiliser les standards (RDF *, SPARQL). Chaque lien utilise le modèle Ressource Description Framework (RDF). Un RDF triplet est une association triple de liens soi : sujet-prédicat-objet. Ce modèle permet de décrire formellement les ressources Web et leurs métadonnées, afin d'en permettre un traitement automatique et une certaine interopérabilité.
- **Principe 4 :** Inclure des liens vers d'autres URI. Afin qu'ils puissent découvrir plus de choses.

2.4 *Les portails LOD*

Nous présentant dans le tableau ci-dessous quelques exemples des portails LOD présent dans le cloud⁸ :

Tableau 1 : Exemples de portails LOD. Source : [https://lod-cloud.net/]

Dataset	Taille	Description
DBPEDIA	9,500,000,000 triples	DBpedia.org est un effort de la communauté pour extraire des informations structurées de Wikipedia et les rendre disponibles sur le Web. DBpedia vous permet de poser des questions complexes sur Wikipedia et de lier d'autres jeux de données sur le Web à des données Wikipedia.
UNESCO	10,400,000 triples	Données pour les objectifs de développement durable
Bio2RDF::Sider	17,627,864 triples	SIDER contient des informations sur les médicaments commercialisés et leurs réactions indésirables enregistrées. Les informations sont extraites de documents publics et de notices.

Conclusion

Dans les points précédents nous avons présenté le web sémantique et les LODs. Dans la suite de ce chapitre nous allons présenter les métadonnées du LOD car c'est l'une des principales préoccupations de notre projet.

⁸ <https://lod-cloud.net/>

3 *Métadonnées du LOD*

Les LOD sont présentes de plus en plus sur le Web, il y a une masse de données LOD considérable. Pour cela il est nécessaire de les organiser et de les documenter pour faciliter leur compréhension. Le processus des Linked Open Data pourrait être beaucoup plus amélioré en fournissant des métadonnées structurées considérables.

3.1 *Définitions*

Diverses définitions de métadonnées sont disponibles [A.Zuiderwijk, 2012] :

Souvent, les métadonnées sont simplement définies comme des données sur des données) ou des informations sur des informations

Par exemple, la National Information Standards Organization définit les métadonnées comme « des informations structurées qui décrivent, expliquent, localisent ou facilitent la récupération, l'utilisation ou la gestion d'une ressource d'information »⁹

Gilliland¹⁰ indique que les métadonnées dans la discipline des systèmes d'informatique et d'information font référence à « un ensemble de normes sectorielles ou disciplinaires ainsi qu'à une documentation interne et externe supplémentaire et à d'autres données nécessaires à l'identification, représentation, interopérabilité, gestion technique, performance et utilisation des données contenues dans un système d'information ».

3.2 *Typologies des métadonnées du LOD*

Plusieurs classifications de métadonnées ont été proposées par plusieurs chercheurs et communautés, chacun selon son domaine d'activité. Dans la section suivante nous allons en citer quelques-unes.

3.2.1 *Selon les types*

On distingue plusieurs types de métadonnées dans les LOD, dont [A. Zuiderwijk et al., 2012]:

- **Les métadonnées de découverte (à plat)** sont descriptives et de navigation et permettent la découverte de données ouvertes pertinentes par navigation ou requête. Les exemples de métadonnées de découverte incluent des données sur l'identifiant, le titre, le créateur, l'éditeur, le pays, la source, le type, le format, la langue, le secteur, les sujets, etc. mots-clés, système d'information relatif, date de validité (de - à), public cible, cadre juridique, statut, ressources pertinentes et ensembles de données liés.
- **Les métadonnées contextuelles** sont descriptives, restrictives et de navigation. Ils permettent une information riche sur les personnes, les organisations, les projets, les publications et de nombreux autres aspects associés à l'ensemble de données. De plus, les métadonnées contextuelles permettent l'interopérabilité en ingérant et en générant des formats de métadonnées courants utilisés dans les données ouvertes. Les métadonnées contextuelles sont des données sur des organisations, des personnes, des projets, des financements, des installations, des équipements, des services et des pointeurs vers des métadonnées détaillées.

⁹ <https://www.niso.org/>

¹⁰ (2008, <http://www.getty.edu>)

- **Les métadonnées détaillées** couvrent les métadonnées de schéma, ainsi que des métadonnées supplémentaires pour assurer la qualité, telles que des contraintes sur les valeurs d'attribut et les valeurs d'un attribut dépendant d'un autre. Les métadonnées détaillées sont généralement spécifiques à un domaine (par exemple, la santé) ou à un ensemble de données spécifique. Les métadonnées détaillées peuvent être des données sur la qualité (précision, précision, étalonnage et autres paramètres) et des paramètres spécifiques à un domaine ou à un ensemble de données utilisés par les logiciels accédant et traitant le jeu de données.

3.2.2 Selon les normes

"Deux normes de métadonnées ont été utilisées, pour examiner la faisabilité de l'approche proposée dans le modèle de passage sémantique utilisant RDF [N. Y. Chen et al., 2015] :

- **EAD** qui représente une norme de métadonnées pour documenter la description archivistique.
- **CDWA** qui représente une norme de métadonnées qui a été adopté pour la description d'artefacts de musée

3.3 *Les niveaux de définition des métadonnées du LOD*

Il faut noter qu'il y a différents niveaux de normalisation des métadonnées dans les LOD :

3.3.1 Définition au niveau sémantique

C'est principalement d'unifier la sémantique des terminologies, par exemple les normes du Dublin Core¹¹ qui ont fait une unification sémantique terminologique des mots et réduire les conflits de sens et de sémantique. Au niveau sémantique on répond à comment représenter et utiliser les normes existantes imposées par les collectivités LOD.

Pour les collectivités du web sémantique, ce qui prime pour l'instant et qui constitue un des principaux axes de travail, consiste donc en « une normalisation des catalogues de données des portails d'Open data ». La proposition portée par elles, n'est jamais qu'un équivalent tardif (et nécessaire) de ce qu'a été le Dublin Core pour le Web. Il s'agit de mettre en place une fusion des différents catalogues selon le format DCAT (Data Catalog Vocabulary) du W3C. [P. Y. Vandenbussche et al., 2017].

Les points ci-dessous donnent un aperçu de quelques normes de métadonnées disponibles dans le web des données :

- **Friend of a Friend (FOAF)** : consiste à créer un réseau de pages d'accueil lisibles par machine décrivant les gens, les liens entre eux et les choses qu'ils créent et font.
- **Dublin Core (DC)** : C'est norme interopérable de métadonnées en ligne axée sur les ressources en réseau.

¹¹ <https://www.bnf.fr/fr/dublin-core>

3.3.2 Définition au niveau schéma

La définition au niveau schéma décrit le modèle de représentation de métadonnées. Il existe plusieurs définitions hétérogènes. Il y a donc nécessité d'unifier le schéma de représentation des métadonnées dans des triplets RDF. Il existe six façons différentes de décrire un triplet RDF $\langle s, p, o \rangle$ de métadonnées, grâce à une clé de métadonnées et une paire de valeurs.

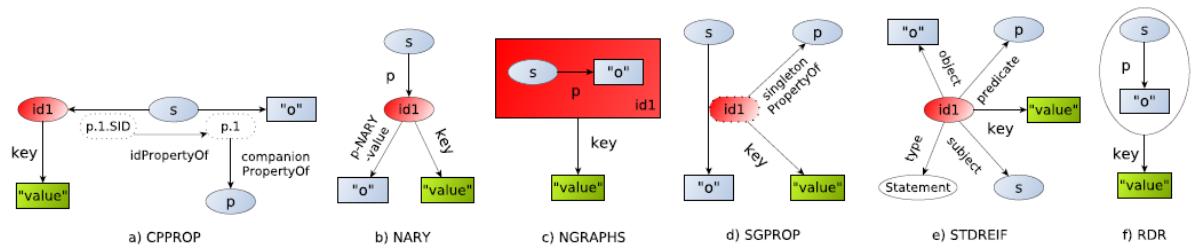


Figure 18 : Structure des différents modèles de représentation des métadonnées au niveau schéma. Source : [J. Frey, 2017]

Il existe principalement six modèles de représentation des métadonnées (Voir figure 18). Ces normes sont différentes et hétérogène, pour cela des efforts sur ce niveau doivent être fournis pour unifier les normes et proposer un schéma homogène. Au niveau schéma on répond à comment représenter les métadonnées dans un schéma unifié.

Nous allons présenter dans ce qui suit, les différents formats grâce à un exemple, dans lequel les entités sont présentées avec les valeurs d'année de naissance pour les personnes *p1* et *p2* à l'aide de l'attribut *dbo:birthYear* pour chaque métadonnée d'instruction RDF concernant la date de dernière modification (*dc: modified*). La requête présentée recherche la date de naissance la plus récente pour chaque personne distincte [J. Frey, 2017].

a) Named graphs (ngraphs)

La fonctionnalité de ngraphs, qui est prise en charge par de nombreux moteurs de graphe, permet l'attribution d'un IRI pour un ou plusieurs triplets en tant qu'identificateur de graphe. Le même IRI peut ensuite être utilisé comme sujet pour une entité de métadonnées, qui peut elle-même stocker les métadonnées sur le ou les triples associés sous forme de prédictats et d'objets.

```

# data with birth year values for
# Person p1 and p2 (trig notation)
<g1> { <p1> dbo:birthYear "1981" . }
<g2> { <p1> dbo:birthYear "1983" . }
<g3> { <p2> dbo:birthYear "1982" . }

# metadata:
meta:dbpedia { <g1> dc:modified "2016-11^^xsd:gYearMonth .
<g2> dc:modified "2014-12^^xsd:gYearMonth .
<g3> dc:modified "2012-01^^xsd:gYearMonth . }

# query
SELECT ?person ?birth {
  { GRAPH ?g {?person dbo:birthYear ?birth} .
    GRAPH meta:dbpedia {?g dc:modified ?modified}
  } FILTER NOT EXISTS {
    GRAPH ?g2 {?person2 dbo:birthYear ?birth2} .
    GRAPH meta:dbpedia {?g2 dc:modified ?modified2}
    FILTER (?person = ?person2 && ?modified2 > ?modified)
  }
}

```

Figure 19 : Représentation TTL des métadonnées de format NGRAPHS et la requête SPARQL correspondante. Source : [J. Frey, 2017]

Le gros inconvénient du modèle ngraphs est le fait qu'il utilise l'IRI de graphe nommé en tant qu'URI pour une ressource de métadonnées, qui stocke elle-même un ensemble de métadonnées basées sur la valeur clé. Si l'ensemble de données d'origine utilise un graphe nommé pour stocker les données, un MRM ngraphs ne peut pas être appliqué de manière rétro compatible.

b) RDF reification

Comme spécifié dans le standard RDF, il est possible de créer une ressource qui décrit un triple et son sujet, son prédicat et son objet. La ressource IRI peut ensuite être utilisée pour connecter une métainformation de provenance ou une métainformation au triple.

```

# birth year values for
# Person p1 and p2 (turtle notation)
<stmt-56e8> rdf:type rdf:Statement ;
  rdf:subject <p1> ;
  rdf:predicate dbo:birthYear ;
  rdf:object "1981" .
<stmt-4f83> rdf:type rdf:Statement ;
  rdf:subject <p1> ;
  rdf:predicate dbo:birthYear ;
  rdf:object "1983" .
<stmt-4327> rdf:type rdf:Statement ;
  rdf:subject <p2> ;
  rdf:predicate dbo:birthYear ;
  rdf:object "1982" .

# metadata:
<stmt-56e8> dc:modified "2016-11^^xsd:gYearMonth .
<stmt-4f83> dc:modified "2014-12^^xsd:gYearMonth .
<stmt-4327> dc:modified "2012-01^^xsd:gYearMonth .

# query
SELECT ?person ?birth {
  ( ?g rdf:subject ?person; rdf:predicate dbo:birthYear;
    rdf:object ?birth ; a rdf:Statement . )
  (?g dc:modified ?modified . )
} FILTER NOT EXISTS {
  (?g2 rdf:subject ?person2; rdf:predicate dbo:birthYear;
    rdf:object ?birth2 ; a rdf:Statement . )
  (?g2 dc:modified ?modified2 )
  FILTER (?person = ?person2 && ?modified2 > ?modified )
}

```

Figure 20 : Représentation TTL des métadonnées de format RDF Reification et la requête SPARQL correspondante. Source : [J. Frey, 2017]

Par rapport aux ngraphs, il n'est pas possible de réutiliser des requêtes de données existantes prêtées à l'emploi. Pour qu'ils fonctionnent, un mécanisme de raisonnement personnalisé devrait être appliqué.

De plus, chaque triple de données d'origine doit être représenté par une entité ressource, elle-même composée de quatre déclarations (rdf: subject, rdf: predicate, rdf: object et rdf: Statement). Cela n'augmente pas seulement la taille du jeu de données, mais ajoute plus de motifs triples aux requêtes de recherche potentielles.

c) Nary relation (naryrel)

Une instance de relation est créée en tant que ressource de la paire sujet-prédicat au lieu de l'objet du triple. L'objet est connecté à la ressource de relation à l'aide d'une version renommée du prédicat (suffixe désigné ajouté). La même ressource relationnelle est utilisée pour relier les métadonnées à la déclaration.

```
# birthyear values for
#Person p1 and p2 (turtle notation)
<p1> dbo:birthYear <rel-56e8> .
<rel-56e8> dbo:birthYear-value "1981" .
<p1> dbo:birthYear <rel-4f83> .
<rel-4f83> dbo:birthYear-value "1983" .
<p2> dbo:birthYear <rel-4327> .
<rel-4327> dbo:birthYear-value "1982" .

# metadata:
<rel-56e8> dc:modified "2016-11"^^xsd:gYearMonth .
<rel-4f83> dc:modified "2014-12"^^xsd:gYearMonth .
<rel-4327> dc:modified "2012-01"^^xsd:gYearMonth .

# query
SELECT ?person ?birth {
  { (?person dbo:birthYear ?g. ?g dbo:birthYear-value ?birth .) .
    (?g dc:modified ?modified .)
  } FILTER NOT EXISTS {
    { ?person2 dbo:birthYear ?g2.
      ?g2 dbo:birthYear-value ?birth2 . }

    (?g2 dc:modified ?modified2 )
  } FILTER ( ?person = ?person2 && ?modified2 > ?modified )
}
```

Figure 21 : Représentation TTL des métadonnées de format Nary-Relation et la requête SPARQL correspondante.
Source : [J. Frey, 2017]

Semblable à la réification standard, un IRI peut être utilisé pour accéder à des données et métadonnées. Dans le cas du MRM à nary relation, l'IRI est une ressource de relation IRI. En raison de l'introduction de la ressource relation IRI, une déclaration supplémentaire par valeur triple doit être ajoutée. Les requêtes de données existantes ne peuvent pas être réutilisées, mais par rapport à la réification standard, il faut moins de modèles triples pour accéder aux valeurs de données ou aux métadonnées.

d) Singleton properties (sgprop)

Le schéma de la Singleton Properties utilise une propriété unique pour chaque triple avec les métadonnées associées. Cette propriété unique consiste en un identificateur triple, qui peut être utilisé pour décrire la déclaration avec des métadonnées. Pour pouvoir reconstruire la propriété d'origine de l'instruction, chaque propriété singleton est liée à son prédicat d'origine à l'aide d'une relation rdf: singletonPropertyOf.

```

# birth year values for
# Person p1 and p2 (turtle notation)
<p1> <56e8-783f-9cd3> "1981" .
<p1> <4f83-6cd5-88da> "1983" .
<p2> <4327-367e-439b> "1982" .

# property reconstruction information
<56e8-783f-9cd3> rdf:singletonPropertyOf dbo:birthYear .
<4f83-6cd5-88da> rdf:singletonPropertyOf dbo:birthYear .
<4327-367e-439b> rdf:singletonPropertyOf dbo:birthYear .

# metadata:
<56e8-783f-9cd3> dc:modified "2016-11"^^xsd:gYearMonth .
<4f83-6cd5-88da> dc:modified "2014-12"^^xsd:gYearMonth .
<4327-367e-439b> dc:modified "2012-01"^^xsd:gYearMonth .

# query
SELECT ?person ?birth {
  { ?person ?g ?birth .
    ?g rdf:singletonPropertyOf dbo:birthYear . }
  { ?g dc:modified ?modified . }
} FILTER NOT EXISTS {
  {?person2 ?g2 ?birth2.
    ?g2 rdf:singletonPropertyOf dbo:birthYear. }
  {?g2 dc:modified ?modified2 }
} FILTER ( ?person = ?person2 && ?modified2 > ?modified )
}
}

```

Figure 22 : Représentation TTL des métadonnées de format Singleton Property et la requête SPARQL correspondante.
Source : [J. Frey, 2017]

Cette propriété unique peut être vue comme une ressource de prédicat IRI. La ressource de prédicat IRI peut être utilisée pour accéder aux métadonnées et au type de prédicat d'origine. Étant donné que la ressource de prédicat a la relation rdf: singletonPropertyOf, il est possible d'utiliser des règles d'implication RDFS pour déduire les instructions d'origine. Lorsque vous examinez les triples de jeu de données, il n'est pas possible de déduire la signification directe d'un prédicat triple.

e) Companion property (cpprop)

Le modèle de représentation de Companion Property souffre du fait qu'il crée une nouvelle propriété pour chaque instruction afin de créer des propriétés globalement uniques. Cela se traduit par une distribution uniforme de propriétés très inhabituelle et, par conséquent, par une augmentation du temps d'interrogation.

```

# birth year values for
# Person p1 and p2 (turtle notation)
<p1> dbo:birthYear.1 "1981" ;
      dbo:birthYear.1.SID <sid-56e8> .
<p1> dbo:birthYear.2 "1983" ;
      dbo:birthYear.2.SID <sid-4f83> .
<p2> dbo:birthYear.1 "1982" ;
      dbo:birthYear.1.SID <sid-4327> .

# property reconstruction and SID association vocabulary
dbo:birthYear.1.SID rdf:idPropertyOf dbo:birthYear.1 .
dbo:birthYear.1 rdf:companionPropertyOf dbo:birthYear .
dbo:birthYear.2.SID rdf:idPropertyOf dbo:birthYear.2 .
dbo:birthYear.2 rdf:companionPropertyOf dbo:birthYear .

# metadata:
<sid-56e8> dc:modified "2016-11^^xsd:gYearMonth" .
<sid-4f83> dc:modified "2014-12^^xsd:gYearMonth" .
<sid-4327> dc:modified "2012-01^^xsd:gYearMonth" .

# inference "rule"
rdf:companionPropertyOf rdfs:subPropertyOf rdfs:subPropertyOf .

# query
SELECT ?person ?birth {
  { ?person ?cp ?birth; ?cpid ?g.
    ?cp rdf:companionPropertyOf dbo:birthYear.
    ?cpid rdf:idPropertyOf ?cp . } .

  (?g dc:modified ?modified . )
} FILTER NOT EXISTS {
  { ?person2 ?cp2 ?birth2; ?cpid2 ?g2.
    ?cp2 rdf:companionPropertyOf dbo:birthYear.
    ?cpid2 rdf:idPropertyOf ?cp2. } .
  (?g2 dc:modified ?modified2 )
} FILTER ( ?person = ?person2 && ?modified2 > ?modified )
}
}

```

Figure 23 : Représentation TTL des métadonnées de format Compagnion Property et la requête SPARQL correspondante. Source : [J. Frey, 2017].

Contrairement aux autres représentations à base de triplets, les identifiants ne sont pas nécessaires pour reconstruire le triple d'origine. Ainsi, cela permet également aux graphiques de partager des identifiants d'instruction et plusieurs identificateurs par instruction. Cela permet aux propriétés associées de prendre en charge différents niveaux de granularité.

Conclusion

Dans ce chapitre nous avons présenté les différentes notions relatives au web sémantique et aux Linked Open Data (LOD). Nous avons aussi mis l'accent sur les métadonnées du LOD, qui joue un rôle très important dans notre étude.

Après avoir vue les notions des entrepôts de données et du web sémantique, nous verrons par la suite la combinaison entre les deux technologies, qui nous servira de base pour notre étude.

Chapitre 3 : Entrepôt de données sémantique

Introduction

L'utilisation importante du Web et de l'internet a créé un besoin d'échanges et de partage d'informations ce qui a augmenté de façon considérable la masse de données traitées. Pour réaliser ces tâches, connaître le sens d'une donnée est devenu un enjeu important pour les entreprises, les gouvernements, les applications, les services, etc. Si nous nous penchons sur la technologie des bases de données dans sa première génération, les concepteurs construisaient des bases de données pour répondre à un besoin particulier, mais sans se soucier si ces bases de données seraient intégrées et utilisées par des personnes en dehors de leur entreprise. Lorsque cette dernière souhaite s'ouvrir et partager ses données, elle se heurte au problème d'hétérogénéité structurelle et sémantique des données.

Afin de répondre à ce problème, une combinaison de deux technologies a été proposée. D'une part, les EDs remplissent le rôle d'un système d'intégration des données hétérogènes grâce au processus ETL. Les ontologies quant à elles sont largement utilisées pour expliciter le sens des concepts et des propriétés d'un domaine donné pour permettre une unification sémantique. Cette association crée ce qu'on appelle un Entrepôt de Données Sémantique (EDS).

1 Les entrepôts de données à base ontologique

1.1 Définition

L'entreposage sémantique ou l'entrepôt à base ontologique est une méthode d'entreposage de données utilisant l'un des standards du web sémantique qui sont les ontologies. Les ED sémantiques, sont donc des ED à bases ontologiques dans leur processus de construction, et plus particulièrement dans le processus d'intégration des données.

Lors de la construction d'un ED comme d'un EDS, on distingue deux niveaux de construction. Le premier niveau c'est la construction des sources de données ou bien le système source, et la deuxième c'est la construction de l'ED ou bien du système cible.

Le processus de construction d'un système d'intégration en ED est décomposé en quatre étapes principales, qui sont [M. Hacid, 2004]:

- a) L'extraction des données des sources de données opérationnelles,
- b) La transformation des données aux niveaux structurelle et sémantique,
- c) L'intégration des données (Extraction –Transformation- Chargement)
- d) Le stockage des données intégrées dans le système cible.

La (figure 24) résume l'enchaînement de ces étapes de traitement.

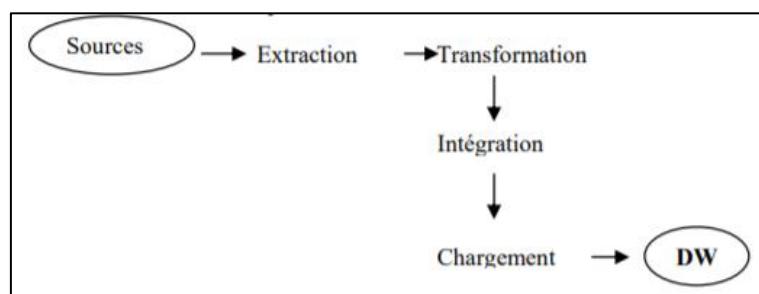


Figure 24 : Etapes d'intégration dans un ED. Source : [M. Hacid, 2004]

1.2 *Intérêt des ontologies pour les ED*

La maturité des ontologies a motivé plusieurs chercheurs et industriels à utiliser pour répondre à plusieurs problèmes, la conception des entrepôts de données, etc. D'une part, les ontologies ont largement fait leur preuve pour faciliter et pour automatiser l'intégration des sources de données hétérogènes en facilitant la résolution des conflits syntaxiques et sémantiques entre ces sources. D'autre part, les ED peuvent être vus comme des systèmes d'intégration matérialisant les données des sources. Les ontologies ont donc naturellement été introduites pour faciliter la phase ETL, qui est la phase responsable de l'intégration des données issues des sources. Les ED à base ontologique se sont largement inspirés des fondements des systèmes d'intégration à base ontologique que nous allons exposer dans ce qui suit.

2 *Les systèmes d'intégration à base ontologique*

Trois principales structures présentées dans la (figure 25) sont utilisées dans un système d'intégration à base ontologique [H. Wache, 2001] :

2.1 *Structure à base d'une ontologie unique*

Dans les systèmes d'intégration à base d'une seule ontologie, chaque source de données est liée à une unique ontologie. Ces approches peuvent être utilisées uniquement lorsque les sources à intégrer fournissent globalement la même vue du domaine, autrement il sera difficile de trouver un vocabulaire minimal commun partagé entre les sources. De plus, des changements au niveau des sources peuvent affecter la conceptualisation du domaine représenté par l'ontologie [H. Wache, 2001]. Cette structure présente certaines contraintes. Les sources de données ne peuvent pas évoluer indépendamment, leur autonomie schématique est donc restreinte. La scalabilité du système d'intégration est également limitée par la portée de l'ontologie ou l'ajout de nouvelles sources peut nécessiter la redéfinition de l'ontologie globale.

2.2 *Structure à base d'ontologies multiples*

Une autre structure de systèmes d'intégration à base d'ontologies multiples a été proposée pour venir en tant que solution pour les inconvénients de la première structure. Dans ces systèmes, chaque source de données définit sa sémantique par une ontologie locale. Les ontologies locales sont mises en correspondance deux par deux. Le principal inconvénient de cette approche est la difficulté d'effectuer les mappings [S. Khouri, 2013].

2.3 *Structure à base d'ontologies partagées*

Dans la troisième structure des systèmes d'intégration à base d'une ontologie partagée, on retrouve les notions d'ontologie globale et d'ontologies locales. Chaque source est décrite sémantiquement par sa propre ontologie, qu'on appelle ontologie locale. Afin de faciliter le mapping entre les ontologies locales, elles sont mises en correspondance avec une ontologie partagée modélisant un domaine particulier, qu'on appelle ontologie globale [S. Khouri, 2013].

L'intégration dans cette structure passe par les étapes suivantes [L. Bellatreche et al., 2006] :

- La définition de l'ontologie locale pour chaque source.
- La mise en correspondance ou l'alignement entre les ontologies locales des sources et l'ontologie partagée en établissant des relations sémantiques entre leurs concepts.

- L'alignement entre les ontologies locales et l'ontologie partagée permet d'identifier les relations entre les entités des différentes ontologies.

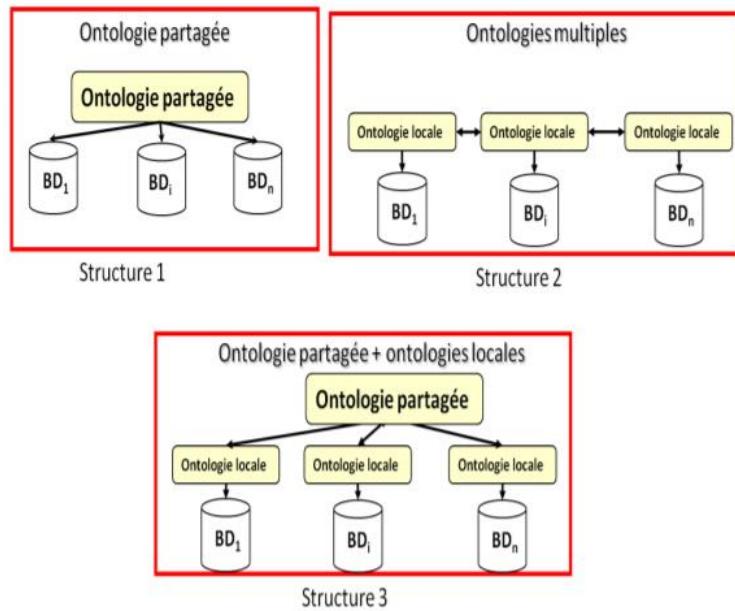


Figure 25 : Approches d'intégration à base d'ontologie conceptuelle. Source : [S. Khouri, 2013]

Conclusion

Dans la partie de l'état de l'art nous avons eu l'occasion de faire des recherches approfondies à propos des différents concepts clés qui nous permettent de bien comprendre le contexte de notre projet dont le principal but est l'exploitation des métadonnées du LOD dans les systèmes d'ED ainsi que des Entrepôts de Données Sémantiques.

Nous avons donc, présenté dans le 1^{er} chapitre l'exploitation d'un entrepôt de données, les objectifs de celui-ci et ses différents composants. Tandis que dans le 2^{ème} chapitre nous avons parlé du web sémantique, de son évolution, nous avons notamment pris connaissance de ses différentes technologies. Nous avons ensuite mis l'accent sur les Linked Open Data qui se basent sur les différentes technologies du web sémantique.

En fin, dans le 3^{ème} chapitre nous avons présenté les ED à base ontologique, et le rôle de ce système dans l'intégration des données hétérogènes à base d'ontologies.

Les notions clés présentées précédemment, nous ont servi pour proposer une solution allant de la définition des besoins opérationnels à la conception du processus d'intégration jusqu'à l'exploitation de celui-ci pour calculer des métriques et des indicateurs de qualité qui aident à la prise de décision. Le principal challenge de notre travail est de concevoir un système d'intégration conjointe des données sémantiques internes et externes (du LOD) ainsi que de leurs métadonnées. Comme nous l'avons décrit, les différents niveaux d'intégration (sémantique et formalismes) doivent être pris en compte lors de cette intégration. Le challenge de notre travail est d'appliquer l'approche à base d'ontologies partagées, afin de faire l'intégration des données sémantiques (de sources internes ou externes). Lors de cette opération, les deux niveaux d'intégration (sémantique et formalismes) doivent être pris en compte c'est le principal défi de notre travail.

Partie 2 : Conception de la solution

Avant d'aboutir aux résultats finaux de notre travail, nous avons consacré un effort très important à la phase de conception et de modélisation de la solution envisagée. Cette phase est sans doute la phase la plus importante dans tout projet informatique existant.

Nous allons traiter dans cette partie une étude complète qui va nous permettre de comprendre, en premier lieu, le contexte global du projet, la problématique et les différents objectifs que nous souhaitant obtenir.

En second lieu, nous allons présenter l'approche orientée besoins. Nous allons suivre cette approche dans le cadre de notre projet pour la conception d'un EDS. Un premier prototype a été développé à cet effet. Celui-ci par contre montre de nombreuses lacunes et il a fallu faire une refonte complète du prototype pour remédier aux différentes lacunes rencontrées.

En troisième lieu, nous allons présenter la conception de notre solution. De plus que l'approche de conception orientée besoins, notre solution va comporter un volet très important qui consiste à proposer un système d'intégration complet qui unifie les données venant de différentes sources, ainsi que leurs métadonnées aux deux niveaux formalisme et sémantique. Dans ce dernier chapitre, nous commencerons par présenter notre démarche de conception, ensuite nous allons passer à l'expression et l'analyse des besoins opérationnels. Nous détaillerons ensuite l'architecture de notre solution sous 4 couches (Couche client, Couche traitement, Couche service, Couche système).

Chapitre 1 : Contexte et présentation du projet

Introduction

Avant de passer à la phase de conception, il y a une nécessité de présenter d'abord le contexte du projet et les problématiques que nous visant à résoudre. Le but de ce chapitre est de faire comprendre au lecteur le positionnement de notre projet pour faciliter la compréhension de la solution proposée. Ce chapitre est organisé en trois parties. Nous allons commencer par présenter le contexte de notre projet. Nous discuterons ensuite la problématique du projet et les différents objectifs que nous voulons atteindre à la fin de ce projet.

1 Contexte du projet

Le projet consiste à mettre en place un EDS suivant une approche orientée besoins et assurant l'intégration conjointe des données et métadonnées (issues de sources internes et externes LOD). Un outil automatisant l'intégration et la construction d'un entrepôt de données sémantiques sera réalisé pour des fins BI et décisionnels. Cette solution est destinée principalement aux concepteurs des entrepôts de données. Il représente d'une part un EDS, qui permet en premier lieu d'enrichir de façon optimale les données et métadonnées du cube interne en utilisant un processus d'intégration des données et métadonnées extraites de différentes sources. En deuxième lieu, il permet de gérer le cube de données interne en utilisant les indicateurs techniques et organisationnels et les métriques de qualité. Notons cependant que cette solution est en plus d'être destinés aux concepteurs des entrepôts de données, elle est aussi destinée à servir pour les démonstrations du maître de stage lors de différents événements autour des entrepôts de données sémantiques mais représente aussi un accompagnement technique des articles de recherche.

2 Problématiques

La mission principale de notre projet s'intègre dans un projet initié au laboratoire LCSI (ESI)¹². Celui-ci implémente la démarche de conception d'ED orientée besoins [S. Khouri et al., 2018]. Un premier prototype a été réalisé à cet effet [L. Parkin, 2018]. L'approche en question est l'aboutissement de plusieurs années de recherche issues de [S. Khouri, 2013] qui traite divers sujets autour de la modélisation à base ontologique, l'intégration de sources de données hétérogènes mais aussi la conception orientée besoins des entrepôts de données.

L'objectif initial de cette approche est de permettre à partir d'un ensemble de besoins décisionnels de construire un ED répondant efficacement à ces besoins. Cette efficacité revient à faire une sélection par besoins des sources internes et des sources externes. En effet, cette sélection permet de trier dès le début les sources de données et de sélectionner celles qui sont pertinentes et qui répondent aux besoins des utilisateurs. La démarche permet aussi d'assurer une intégration incrémentale qui prend en compte de nouveaux besoins lorsqu'ils se présentent. La démarche procède ensuite à l'intégration des sources via un processus ETL sémantique défini dans le cadre du projet.

Contrairement aux sources internes dont la qualité est assurée et dont l'entreprise ou l'organisation maîtrise la provenance, les sources externes sont construites en dehors des frontières de l'entreprise, et doivent passer par un processus de vérification de leur qualité avant leur intégration au sein de l'ED.

¹² <http://lcsi.esi.dz/>

Une première étape à assurer pour vérifier la qualité de ces sources externes revient à analyser les métadonnées accompagnant ces sources qui fournissent des informations pertinentes sur leur provenance, leur création, leur date de validité, etc. Le challenge ici revient au fait que les métadonnées se présentent également de façon hétérogène aux deux niveaux : sémantique et formalisme (Voir Partie 1 – Chapitre 2 – Section 3.3.2). Une démarche d'intégration conjointe des données et des métadonnées issues de l'ensemble des sources (internes et externes) doit être assurée. Les deux couches données et métadonnées doivent être intégrées respectivement au niveau du modèle et méta-modèle de l'ED cible. Le principal objectif de notre projet de fin d'études est d'assurer cette intégration conjointe, en complétant le processus ETL existant défini dans le cadre de la démarche orientée besoins proposée.

Notre solution va donc traiter deux principales problématiques, la première revient à concevoir et à implémenter un processus d'intégration des données et des métadonnées conjointement, le second revient à concevoir ce processus selon la vision orientée besoins, permettant une construction incrémentale de l'ED.

3 Objectifs

Maintenant que la problématique du projet est bien positionnée, nous allons énumérer les différents objectifs de notre projet. L'objectif principal étant de compléter l'approche par besoins avec un système d'intégration complet traitant les données et leurs métadonnées et d'automatiser tout cela en proposant un outil case destiné aux concepteurs des ED.

Dans ce qui suit nous pouvons résumer notre projet dans les objectifs principaux suivants :

- Compléter l'approche de conception d'ED orientée besoins par une démarche d'intégration des données et leurs métadonnées.
- Offrir une démarche conjointe assurant l'intégration des données et de leurs métadonnées et leurs chargements respectifs dans le modèle et méta-modèle de l'EDS cible. Pour cela, le l'EDS cible se compose du :
 - Schéma de métadonnées cible (TargetMetaSchema) qui permet de stocker les métapropriétés suivant les différentes classes du méta-modèle (SM4AM) qu'on détaillera dans le chapitre suivant.
 - Schéma de données cible (TargetDataSchema) qui permet de stocker les données et les métadonnées après la phase d'extraction, de transformation et de chargement (Processus ETL).
- Refonte du prototype existant pour automatiser l'approche grâce à un outil destiné aux concepteurs d'ED et l'adapter à leurs besoins métiers et applicatifs.
- Construction de l'EDS cible unifiant les sources de données choisies.

Chapitre 2 : Etude de l'existant

Dans ce chapitre nous allons donner un aperçu de l'approche orientée besoins et nous allons présenter l'outil existant qui implémente cette approche. Nous allons ensuite lister les différentes lacunes et problèmes que pose cet outil et dont on devra remédier avec notre solution.

1 Présentation de l'approche orientée besoins

L'approche orientée besoins est l'une des approches les plus pertinentes pour la conception d'un entrepôt de données sémantique (EDS). C'est une démarche conduite par la définition des besoins qui représente l'une des phases les plus importantes de tout projet business intelligence [S. Khouri et al., 2018].

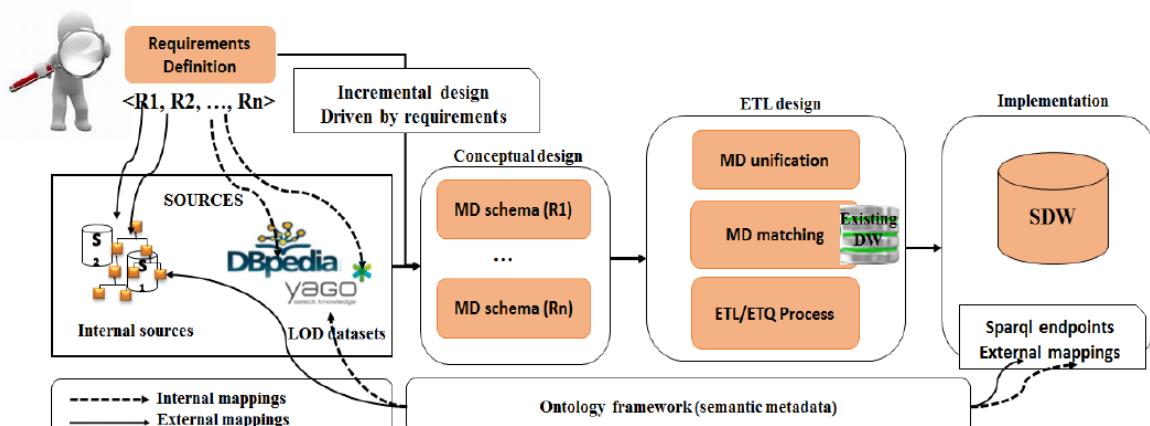


Figure 26 : Le processus de fonctionnement de l'approche orienté besoins. Source : [S. Khouri et al., 2018]

Cette approche fournit une formalisation du cycle de conception d'un EDS, y compris ses principales phases : définition des besoins, conception et phases de mise en œuvre. Elle propose l'utilisation d'une ontologie qui prend en charge le cycle de conception et la conservation des traces des besoins et des décisions de conception.

Dans un environnement LOD, garder trace des différentes étapes de conception est important spécialement dans le contexte où les données ne sont pas toujours matérialisées dans l'EDS et où de nombreuses décisions de conception et tâches fastidieuses peuvent être perdues (telles que l'intégration de données externes avec des données internes, la définition de mappings ou des annotations multidimensionnelles).

L'approche prend en compte les besoins progressivement. Différentes expériences sont menées pour illustrer l'impact de l'intégration des besoins exploratoires dans le processus de conception, à différents niveaux de conception (schéma et instances). La (figure 26) ci-dessus illustre le fonctionnement de l'approche proposée.

L'approche orientée besoins possède des entrées, un processus de traitement et des sorties. Nous allons détailler cela comme suit :

1.1 Les entrées de l'approche

L'approche orienté besoins possède un ensemble d'entrées comme suit :

- Ensemble de besoins suivant un modèle de buts. Un but est défini par des critère et un résultat. On prend par exemple le besoin ultime d'une entreprise est d'augmenter son chiffre d'affaire, pour cela il y'a un ensemble de buts intermédiaires à réaliser. L'ensemble des buts forment une hiérarchie appelée arbre des buts.
- Une ontologie interne (OI) couvrant un ensemble de sources internes. Elle représente l'architecture d'ontologie partagée, décrite dans la partie 1, chapitre 3 et point (2.2.3).
- Ensemble de fragments LOD identifiés par le concepteur selon les besoins.

1.2 Le processus de traitement

Le processus de traitement suit le cycle de conception d'un EDS. L'approche présente un système répondant aux besoins pour gérer le cycle de vie d'un EDS. L'approche proposée comprend quatre étapes principales comme suit : [S. Khouri et al., 2018].

- **Définition des besoins** : La première phase de conception commence par l'élaboration et l'analyse des besoins. Dans les systèmes d'intégration basés sur des ontologies, suivant l'architecture partagées. L'approche prend en compte les besoins de façon incrémentale. Chaque besoin possède la structure suivante *besoin* : <Id, Résultat, Critères>. Par exemple, dans un entrepôt analysant les transactions d'un système de e-commerce, si on considère le besoin suivant « Les produits les plus populaires spécifiques pour chaque pays ». Le besoin est composé d'un ensemble de classes critères (Ex : Region) et d'une classe résultat (Ex : Produit). L'ensemble de ces classes critères et résultats sont identifiées à partir de l'ontologie interne et/ou à partir des sources externes LOD.
- **Modélisation conceptuelle** : La phase de conception identifie le schéma requis pour satisfaire l'ensemble des exigences. La structure multidimensionnelle de ce schéma est également identifiée. Pour répondre aux besoins des utilisateurs, l'identification des concepts multidimensionnels est basée sur l'analyse des besoins. Compte tenu de la structure proposée des besoins <Id, Résultat, Critères>, nous considérons les concepts de "résultat" comme des concepts centraux à analyser (les faits) et les concepts de "critères" comme dimensions du fait. Des dimensions supplémentaires sont identifiées en fonction de leurs relations avec les concepts de fait, en prenant en compte les relations de cardinalités (1-N).
- **Conception du processus ETL** : La conception ETL est formalisée par le triplet <G, S, M>, où G est le schéma cible (c'est le schéma multidimensionnel identifié lors de la phase précédente), S est l'ensemble des sources. Les schémas des sources sont représentés par l'ontologie interne (OI) ainsi que les fragments LOD identifiés. Les schémas des sources ainsi que le schéma cible sont formalisés en format RDF. M c'est l'ensemble des mappings entre ces schémas. Le processus ETL est appliqué afin de peupler le schéma G avec les instances disponibles extraites des sources en exécutant les mappings M. Les mappings sont définis manuellement par le concepteur en utilisant un ensemble d'opérateurs ETL. [D. Skoutas et al., 2007] a défini dix opérateurs ETL conceptuels génériques généralement rencontrés dans un processus ETL, à savoir :

- **EXTRACT (S, C):** extrait des jeux d'enregistrements entrants la partie appropriée.
- **RETRIEVE (S, C):** récupère les instances associées à la classe C depuis la source S.
- **MERGE (S, I):** fusionne des instances appartenant à la même source
- **UNION (C, C '):** unifie les instances dont les classes correspondantes C et C' appartiennent respectivement à des sources différentes S et S'.
- **JOIN (C, C '):** joint les instances dont les classes correspondantes C et C' sont liées par une propriété
- **STORE (S, C, I):** charge les instances I correspondant à la classe C dans le magasin de données cible S
- **DD (I):** détecte les valeurs en double sur les jeux d'enregistrements entrants
- **FILTER (S, C, C '):** filtre les jeux d'enregistrements entrants, ne permettant que les enregistrements avec les valeurs de l'élément spécifié par C'
- **CONVERT (C, C '):** convertit les ensembles d'enregistrements entrants du format de l'élément C au format de l'élément C'
- **AGREGAT (F, C, C '):** agrège le jeu d'enregistrements entrant en appliquant la fonction d'agrégation F (COUNT, SUM, AVG, MAX) définie dans le magasin de données cible.
- **Modélisation physique :** cette phase nécessite l'implémentation du processus ETL en flux physique. Cette implémentation consiste à traduire des opérateurs ETL conceptuels en requêtes SPARQL. Chaque résultat de la requête est représenté par un ensemble de données RDF.

1.3 Le résultat de l'approche

En sortie de l'approche on aura un EDS avec les sources internes et externes (LOD) intégrées par un processus ETL. Des indicateurs sont calculés pour mesurer l'impact de cette intégration.

2 Outil existant

L'objectif initial de cet outil est d'implémenter l'approche orientée besoins, afin de permettre à un concepteur de créer son entrepôt de données à partir d'un ensemble de besoins. L'outil facilitera l'intégration des données venant de sources multiples internes et/ou externes, et présentera à l'utilisateur des indicateurs de performance lors de l'utilisation de l'entrepôt de données.

2.1 Fonctionnalités et interface de l'outil existant

L'outil implémente quatre principales fonctionnalités dont un module d'analyse et d'expression des besoins, un module de gestion des opérations ETL, un module de gestion des mappings et enfin un tableau de bord sommaire pour afficher certains indicateurs.

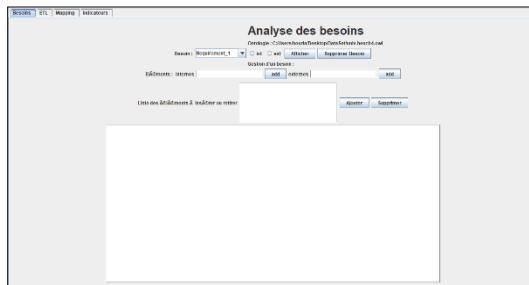


Figure : Interface d'expression et analyse des besoins

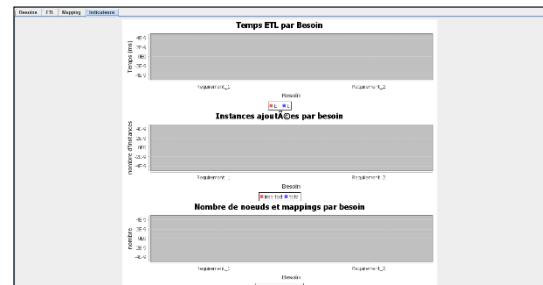


Figure : Interface du tableau de bord

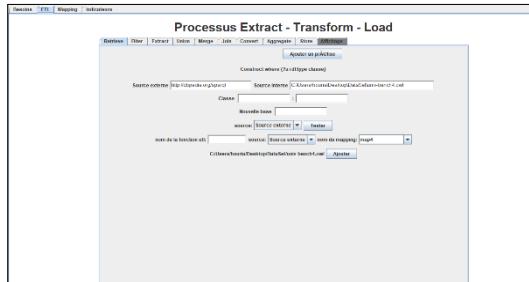


Figure : Interface de la manipulation ETL

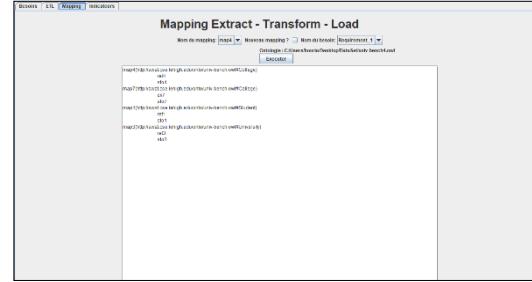


Figure : Interface des mappings

2.2 Les problèmes rencontrés :

Plusieurs problèmes techniques et fonctionnels ont été constatés sur l'outil. C'est un outil qui reste très basique et qui manque d'intuitivité d'utilisation et de manipulation. Nous avons donc soulevé certains problèmes et dysfonctionnements, qui nous ont poussés à proposer une refonte intégrale de cet outil.

- L'outil manque de flexibilité et d'évolutivité :** l'application en elle-même expose des difficultés pour appliquer dessus des évolutions fonctionnelles. L'évolution de l'outil est quasiment faible, et une refonte intégrale de celui-ci est nécessaire. Il faut donc une régression des fonctionnalités déjà présentes. De plus, l'outil ne fonctionne qu'avec une seule ontologie, celle du cas d'étude choisi, alors que le but du travail c'est de créer un modèle universel pour toutes ontologies existantes.

- **L'outil manque de performance et d'efficacité :** on entend par performance d'un logiciel sa rapidité de traitement et on entend par l'efficacité d'un logiciel l'économie des ressources matérielles requises pour son fonctionnement. Par ailleurs le temps de réponse de l'outil existant est très long et consomme beaucoup de ressources.

- **L'outil manque de souplesse, d'adaptabilité, de simplicité, de confort, voire l'ergonomie et l'esthétique d'utilisation :** on entend par ergonomie d'un logiciel l'économie des gestes de l'utilisateur pour réaliser les opérations (sélection des fonctions, des données, modifications, choix de paramètres, d'options). En effet, l'outil dans son ensemble manque d'intuitivité, et les interfaces ne sont pas toutes compréhensibles par l'utilisateur. Ce dernier peut prendre beaucoup de temps et essayer plusieurs clics pour arriver à la manipulation voulue de l'outil.

- **L'outil ne répond pas aux besoins correctement :** En effet, l'outil ne remplit pas une des exigences les plus importantes qui sont l'universalité et la compatibilité avec n'importe quelle ontologie. L'outil existant présente donc un problème de conception, il ne reconnaît que l'ontologie de base sur laquelle l'EDS a été construit, et ne supporte aucune autre ontologie. Par ailleurs, on remarque que l'outil ne respecte pas le besoin d'historisation exigé dans toute conception d'entrepôt de données. En effet, l'outil donne la main pour la suppression des besoins ce qui cause la perte de l'historique estimé essentiel pour les analyses et le reporting. On ajoute aussi, que l'outil reste très abstrait, il ne nous permet pas de consulter le contenu d'un besoin ou de le paramétriser. De plus, il utilise que des champs de texte pour la saisie de l'utilisateur alors qu'aucun contrôle de saisie n'a été fait. Par conséquent, l'outil génère constamment des erreurs.

2.3 *Pistes à compléter et vision de la solution*

Dans cette partie nous allons aborder en deux points notre contribution dans le projet, les pistes à améliorer et à compléter ainsi que la valeur ajoutée que en proposant pour la refonte de l'outil et l'ajout d'un processus d'intégration complet.

2.3.1 *Ajout de la couche intégration des données et des métadonnées*

Tout projet de construction d'Entrepôt de données, nécessite de passer par un processus ETL. Ce dernier se charge de l'intégration des données venant de différentes sources.

La contribution qu'on estime apporter à la conception de l'approche est de la revisiter en y ajoutant une couche permettant de gérer l'intégration des métadonnées simultanément avec l'intégration des données.

Le but de cette conception est l'exploitation de données externes venant de sources hétérogènes pour enrichir l'ED et répondre efficacement aux besoins des utilisateurs. L'exploitation de ces données n'est pas suffisante, il y a une nécessité d'unifier les différentes formes de leurs métadonnées. En effet, l'unification des métadonnées assure la cohérence, l'exactitude et la véracité des données et permet une exploitation efficace. Les métadonnées assurent au concepteur une vision meilleure de la qualité des données (comme leur provenance, leur créateur, leur date de création, leur validité, etc.), que l'entreprise n'a pas générée et dont elle ne possède aucune maîtrise. Le concepteur pourra ainsi savoir quelles sont les données externes à considérer et celles à vérifier. Les données résultantes du processus d'intégration sont chargées dans le schéma de données cible, et les métadonnées dans le méta schéma cible de l'EDS. Un tableau de bord sera également fourni contenant des indicateurs de traçabilité estimant l'effort de conception réalisé.

2.3.2 *Refonte de l'outil existant et implémentation de l'approche orientée besoins*

Après les différents problèmes et dysfonctionnements conceptuels, techniques et métiers constatés dans l'application existante, nous serons dans l'obligation de faire une refonte intégrale de l'outil en implémentant une conception flexible et évolutive et adéquate aux exigences des utilisateurs. Nous allons donc implémenter l'approche par besoin en prenant en compte les critères de qualité logiciels ainsi que les exigences des utilisateurs.

Conclusion

Nous avons vu dans ce présent chapitre le principe de l'approche par besoin, ainsi que le fonctionnement et l'intérêt de celle-ci pour la construction d'un EDS efficace. Par la suite, nous avons présenté l'outil existant qui a été proposé dans les précédentes recherches, afin d'implémenter l'approche par besoin. Nous avons recensé certaines lacunes et problèmes que possède cet outil et que nous allons corriger grâce à notre solution.

Chapitre 3 : Conception

Introduction

Dans le chapitre précédent nous avons identifié le contexte et les différentes problématiques de notre projet. Nous avons aussi présenté l'étude de l'existant où nous avons décrit l'approche par besoins et l'outil existant. Dans ce chapitre nous passons à l'étape de conception de notre solution. Cette partie comprend les étapes les plus importantes de tout projet informatique. Dans le 1^{er} point, nous allons présenter l'approche et la démarche de conception suivie tout au long de notre travail. Dans le second point, nous allons effectuer l'analyse et l'expression des besoins fonctionnels et non fonctionnels. Nous allons ensuite, dans le 3^{ème} point présenter le fonctionnement général de notre solution. Enfin, dans le 4^{ème} point nous verrons en détail l'architecture de la solution et ses quatre couches (Client, Traitement, Service et Persistance).

1 Approche et démarche de conception

Tout au long de la réalisation de notre système, les maitres de stage ont mis l'accent sur la validation régulière, souvent hebdomadaire, de l'évolution du travail. Plusieurs versions du système ont été remises pour la validation. Les phases de validation répétitives, nous ont permis de collecter les besoins des maitres de stage et d'adapter le système à leurs besoins. Notre système est composé de deux parties complémentaires. L'implémentation de l'approche par besoins et la création du système d'intégration des données et des métadonnées. Chacune de ses parties a dû faire l'objet de plusieurs prototypes.

Nous avons commencé notre projet de la façon suivante : tout d'abord nous nous sommes documentés et nous avons testé le logiciel existant. Nous avons ensuite soulevé l'ensemble des problèmes liés à la conception du modèle de données initial et nous avons détecté les dysfonctionnements liés à la réalisation des différentes fonctionnalités. Cette étape nous a aussi permis d'extraire les fonctionnalités pertinentes au projet et qui nécessitent une réutilisation.

Une fois que l'outil existant a bien été cerné, et les parties réutilisables sont détectées, nous avons créé un premier prototype de l'outil sur lequel on s'est basé pour construire itérativement le logiciel final. Des validations régulières et concrètes de l'avancement ont été faites avec le maître de stage. Plusieurs versions évolutives du produit ont été fournis au maître de stage pendant des périodes déterminées afin de collecter ses retours et adapter le système à ses exigences.

Nous avons donc suivi une démarche de prototypage. Cette démarche désigne l'ensemble des activités de création itérative de prototypes pour un produit informatique. Autrement dit c'est la création consécutive de versions incomplètes d'un programme en cours de développement. Chacune de ses versions est soumise à des tests et des évaluations par le client. Des améliorations sont ensuite apportées à chaque itération. On part donc du principe que la production intégrale d'un logiciel c'est le résultat d'un processus cyclique d'évolution, fournissant un prototype de plus en plus élaboré et de plus en plus proche de la version finale souhaitée. La figure ci-dessous représente un récapitulatif de la démarche du prototypage évolutif suivie durant la réalisation de notre projet.

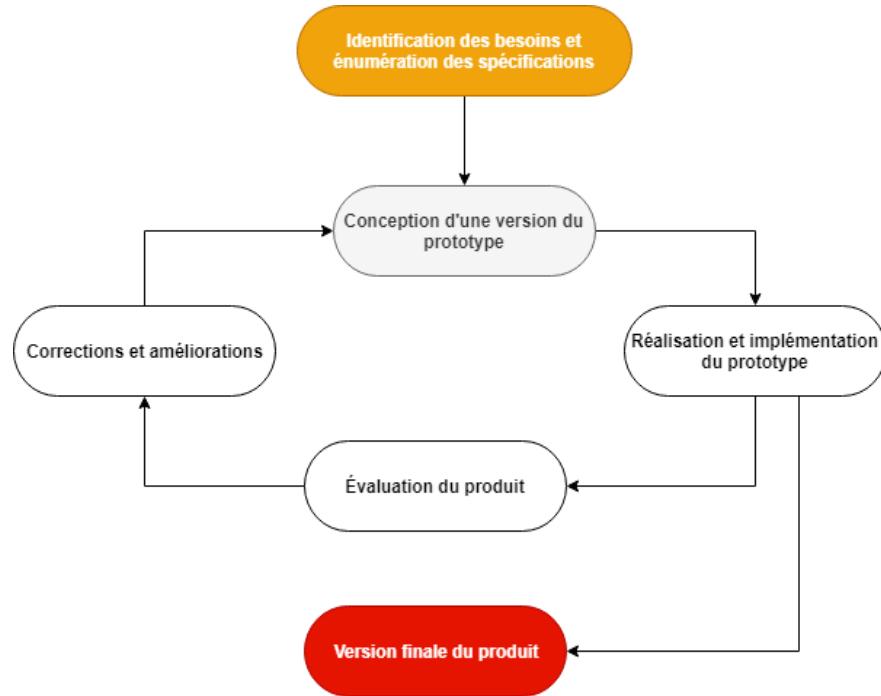


Figure 27 : Démarche de conception (Prototypage évolutif)

La démarche de conception suivie démontre une forte cohésion entre la conception, la réalisation et l'évaluation du système. En effet, depuis le lancement du projet nous avons proposé 6 prototypes et versions évolutives du produit de la version initiale à la version finale. Tout passage d'une version à une autre, ou d'un prototype à un autre, est justifié par les retours du maître de stage et par l'évolution des besoins et des exigences relatives aux métiers.

2 Expression et analyse des besoins

Dans cette partie nous allons identifier les différents acteurs de notre système et analyser les besoins exprimés par les concepteurs. Nous allons donc recenser les différentes spécifications fonctionnelles et non fonctionnelles extraites grâce à un processus évolutif. D'abord, à partir du cahier des charges de l'ancienne, ensuite grâce aux différents retours eut pendant les réunions hebdomadaires de travail avec le maître de stage.

2.1 Identification des acteurs

Afin de mener à bien l'analyse des besoins, nous devons avant tout identifier les acteurs du système que nous visons à réaliser.

Partant du fait que notre système représente un entrepôt de données sémantique. Nous distinguant deux acteurs principaux qui sont au même niveau hiérarchique d'utilisation du système avec des cas d'utilisation différents.

Les acteurs suivants peuvent utiliser le système dans deux cas :

- **Chercheurs** : ils utilisent le système pour traiter et enrichir des données sémantiques pour principalement des fins de recherches (Académiques, Scientifiques, Médicales ...). Le but principal c'est de créer un entrepôt de données sémantique qui répond à leurs besoins en complétant les ontologies internes par des données externes présentes dans les Linked Open Data en ligne (DBPEDIA, Yago ...etc).

- **Concepteurs d'entrepôts de données sémantiques** : Ils utilisent le système principalement pour des fins économiques. Le système répond donc aux problématiques de collecte d'intégration et de persistance des données tout en répondant à leurs besoins. Le système est notamment utilisé pour des fins BI en exploitant différents indicateurs de performance pour la réalisation d'un tableau de bord qui assiste à la prise de décision.

Néanmoins, nous considérerons dans notre système un seul acteur, "Concepteur" représentant soit un « chercheur » ou un « Concepteur d'EDS ». Ils auront très exactement les mêmes fonctionnalités, cependant, ils l'utilisent pour des fins différentes avec des cas d'études différents.

Dans le chapitre suivant, nous étudierons les fonctionnalités (Spécifications fonctionnelles), en rapport avec le métier que l'utilisateur aura à accomplir, ainsi que les besoins techniques (Spécifications non fonctionnelles) auxquels le système répondra afin d'assurer une performance optimale.

2.2 Identification des spécifications fonctionnelles

Nous commencerons par le recensement des besoins fonctionnels, relatifs au métier. Ces besoins seront organisés selon les phases des différents processus et modules fonctionnels.

2.2.1 Spécifications relatives à la gestion des ontologies internes et externes

- Le système doit permettre au concepteur de charger le schéma de l'ontologie globale et d'ajouter de nouveaux préfixes.
- Le système doit permettre au concepteur de consulter le contenu de l'ontologie interne (Classes, propriétés, individus et annotations).
- Le système doit permettre au concepteur de consulter les classes externes ainsi que le format de leurs données et métadonnées.

2.2.2 Spécifications relatives à la gestion des besoins

- Le système doit permettre au concepteur de créer ou mettre à jour un besoin (Ex : Les 10 catégories de produits les plus discutées d'un pays spécifique). Le concepteur devra ensuite déterminer les classes critères qui sont dans ce cas (DWCountry, DWProducer, DWProduct) et la classe résultat (Ex : DWProductCategory).
- Le système doit permettre au concepteur de consulter les détails des besoins existants déjà sauvegardés dans l'ontologie interne. L'utilisateur aura donc une visibilité sur l'ensemble des critères et résultats attendus du besoin, sa description et sa requête.
- Le système doit permettre au concepteur pour un besoin donné d'affecter l'ensemble des mappings d'extractions et de transformations qu'il aura défini précédemment. Ces mappings permettent de peupler les classes cibles de l'ED.

2.2.3 Spécifications relatives à la gestion des mappings

- Le système doit permettre au concepteur de créer des mappings entre les schémas sources et le schéma cible. D'abord il permet de définir l'ensemble des opérations ETL correspondant au mapping ainsi que la position de chacune (Ex : Retrieve, Union, Join, Aggregate ...). De plus, il permet de définir la classe cible où les données extraites/transformées seront chargées.

- Le système doit permettre au concepteur de consulter les détails des mappings existants déjà sauvegardés dans l'ontologie interne. L'utilisateur aura donc une visibilité sur l'ensemble des opérations ETL, la ressource et la position de chacun ainsi que la classe cible dans l'EDS où charger le résultat du mapping.
- Le système doit permettre de visualiser pour un mapping les instances de sa classe source, d'exécuter ensuite le mapping, et afficher le résultat de la classe cible. Le concepteur dans ce cas, choisira s'il veut charger le résultat des transformations dans l'EDS ou pas.

2.2.4 Spécifications relative à la gestion du processus d'intégration

▪ Extraction :

- Le système doit permettre au concepteur de faire l'extraction des données et des métadonnées associées aux instances (internes/externes) transformées et unifiées (grâce aux mappings définis précédemment).
 - Le système doit permettre d'extraire les métadonnées tout en respectant leurs formats au niveau schéma du méta-modèle cible (cpprop, naryrel, ngraphs, sgprop, stdreif, RDR) (Voir : partie 1 – chapitre 2 – section 3.2.2). Lors de l'extraction des métadonnées à partir des sources externes, le système permet d'extraire et d'unifier au même temps les différents formats de métadonnées présents.
 - Le système doit permettre au concepteur de visualiser les classes sources (internes et externes) utilisées. Ensuite, il permet de charger les instances unifiées dans une zone de stockage de données (DSA - Data Staging Area) et visualiser le résultat final de l'extraction.

▪ Transformation :

- Le système doit permettre au concepteur de détecter les instances dupliquées avec l'opérateur (DD - Detect Duplication). Ensuite créer une seule instance qui unifie toutes celles qui sont dupliquées.
 - Le système doit permettre ensuite au concepteur de visualiser les données et les métadonnées unifiées des instances dupliquées.
 - Le système doit ensuite détecter le chevauchement des métadonnées. Autrement dit le système doit détecter s'il existe deux valeurs différentes de la même métadonnée (Ex : creator = A | creator = B).
 - Le système doit ensuite donner la main au concepteur afin de sélectionner la bonne métadonnée à garder dans le DSA. Dans le cas où le concepteur n'est pas sûr de la bonne information, il peut choisir de garder toutes les valeurs.
 - Une fois l'unification des instances et de leurs métadonnées est effectuée, le système doit permettre au concepteur de supprimer les instances dupliquées du DSA et les remplacer par l'unique instance unifiée.
 - Le système doit permettre au concepteur de faire des transformations sur les instances et les données (Retrieve, Union, Join, Merge, Filter, Extract, Aggregate) (Chapitre 2 – section 1.2).
 - Le système doit permettre au concepteur de faire des transformations sur les métadonnées

- Transformation au niveau format : Unification des formats de métadonnées (Format Unification)
 - Transformation au niveau sémantique : Nous proposons un ensemble d'opérateurs pour l'unification des métadonnées au niveau sémantique dont (Metadata Enrichement, Overlapping Detection). Le premier permet d'unifier l'ensemble des métadonnées de différentes sources. Le deuxième nous permet de détecter le chevauchement de sens entre ces métadonnées.
- **Chargement :**
- Le système doit permettre au concepteur de visualiser les méta-propriétés extraites.
 - Le système doit permettre au concepteur de charger dans le schéma de métadonnées cible (TargetMetaSchema) les méta-propriétés extraites à partir des différentes sources externes.
 - Le système doit permettre au concepteur de charger dans le schéma de données cible (TargetDataSchema) le DSA qui contient toutes les instances accompagnées de leurs données et métadonnées.

2.2.5 Spécifications relatives à la gestion du tableau de bord

- Le système doit permettre au concepteur de calculer des mesures estimant l'effort de conception pour le processus ETL (Données et métadonnées).
- Le système doit permettre au concepteur d'évaluer l'impact de l'intégration des besoins en indiquant le nombre de mappings impliqués et de concepts concernés.
- Le système doit permettre au concepteur de visualiser la distribution des ressources internes et externes de chaque besoin.
- Le système doit permettre au concepteur de visualiser la distribution des instances insérées et déduites par le système d'inférences.
- Le système doit permettre de calculer certaines métriques de qualité lors de l'exécution du processus d'intégration des métadonnées, dont :
 - M1: le pourcentage des métadonnées relatives à la date
 - M2: pourcentage des métadonnées relatives à la provenance
 - M3: pourcentage des métadonnées relatives à la version
 - M4: pourcentage des métadonnées relatives à la description
 - Cohérence des données: Cohérence des données: la mesure dans laquelle chaque utilisateur voit de manière cohérente les données et l'intégrité des données est maintenue tout au long des transactions et des sources de données.
 - M5: Pourcentage de doublons.
 - M6: Pourcentage de contradiction (La contradiction représente le cas où l'instance de données possède deux métadonnées similaires mais avec des valeurs différentes).

- Performance : la performance du processus ETL tel qu'il est mis en œuvre sur un système, par rapport à la quantité de ressources utilisée et à la rapidité du service fourni.
 - M7: Chronologie maximale de la mémoire consommée par le processus ETL.
 - M8: Temps d'exécution du processus ETL.
- Audibilité : capacité du processus ETL à fournir une transparence des données, cela inclut: la traçabilité: possibilité de suivre l'historique des étapes d'exécution du processus ETL et la qualité des informations documentées sur l'exécution.
 - M9: Pourcentage des nodes impliquées pour chaque besoin.

2.3 Identification des spécifications non fonctionnels

Maintenant que les besoins relatifs au métier sont spécifiés, nous allons nous intéresser aux besoins non fonctionnels, ou bien techniques, en relation avec la performance du système.

Pour une meilleure performance et une plus grande fiabilité, le système doit assurer quelques spécifications que sont :

- Le système doit supporter, lire et manipuler des fichiers de données sémantiques sous formats OWL, TTL, RDF.
- Le système doit faire appel au Framework « Jena API » un ensemble de bibliothèques contenues qui facilite la manipulation des ontologies sous format RDF et OWL.
- La configuration du système doit être faite en suivant l'outil automatisation de la gestion de projets Java « Apache Maven ».
- Les interfaces du système doivent être développées en JavaFx en utilisant l'outil Scene Builder.
- Les fonctionnalités du système doivent être développées en Java.

3 Fonctionnement générale de la solution

Dans cette section nous nous intéresserons à la conception du système de façon détaillée. Nous allons décrire la solution globale que nous proposons.

Dans le schéma ci-dessous, nous présentons l'architecture de la solution générale que nous détaillerons dans les sections qui vont suivre.

Il faut noter que dans notre système on utilise l'approche orientée besoins que nous avons présentées dans la partie Etude de l'Existant.

En général, on se base sur les besoins de l'organisme pour créer, peupler et exploiter un EDS qui regroupe les différentes sources de données pour répondre au mieux aux besoins exprimés. La solution se compose de quatre couches de traitement représentées en rectangles dans le schéma ci-dessous :

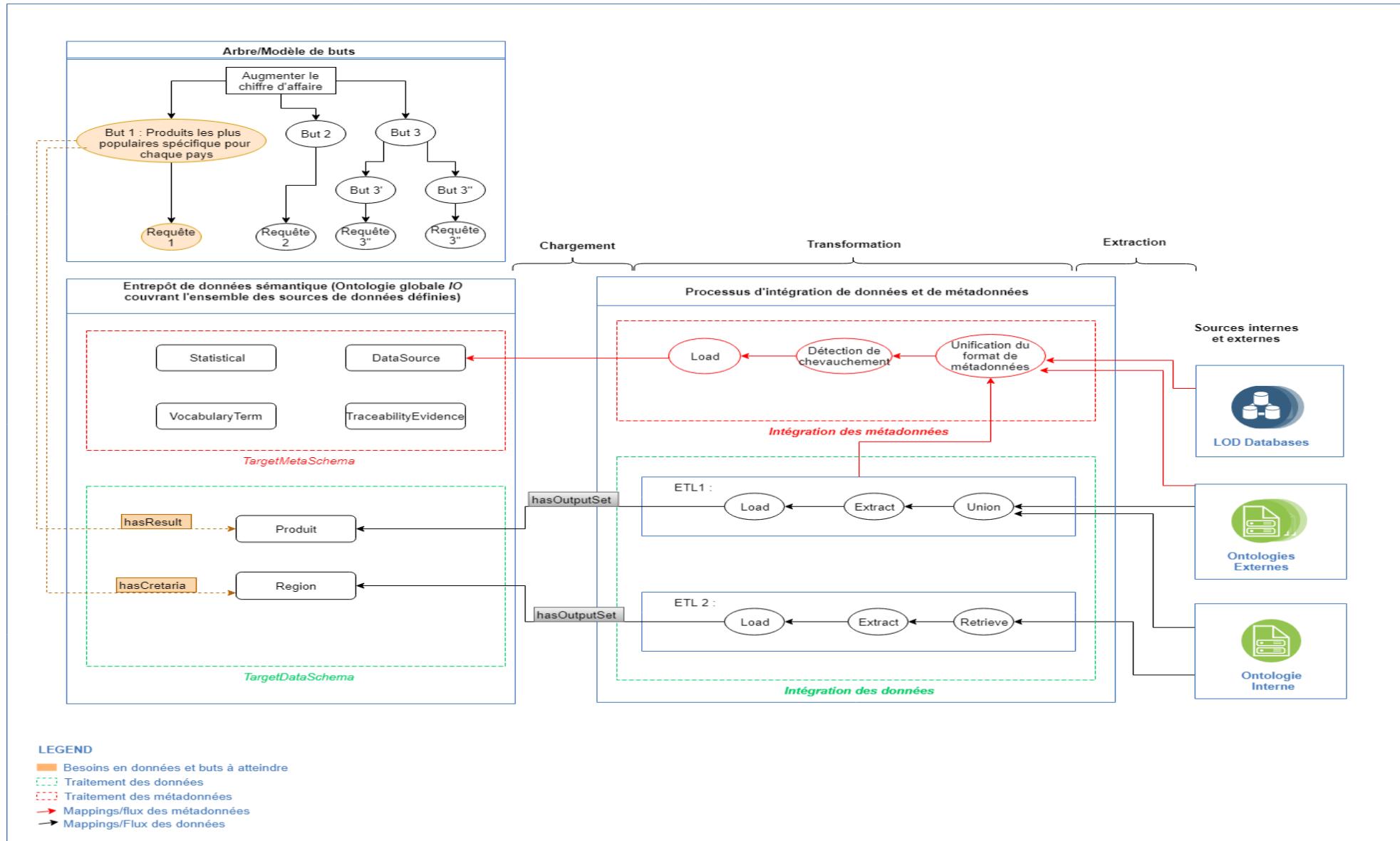


Figure 28 : Schéma de la solution générale

- **Arbre/modèle de but :** Le modèle de buts comporte un ensemble de besoins de l'organisme, il contient un but principal dont découle un ensemble de besoins intermédiaires. On note que le but principal n'est atteint que si l'ensemble ou la majorité des besoins intermédiaires sont satisfaits. Dans le schéma ci-dessus, nous avons pris comme but principal de l'organisme « l'augmentation du chiffre d'affaires ». Afin d'atteindre ce but, plusieurs besoins décisionnels se manifestent. Les relations entre les besoins sont du type soit « ET » ou « OU » (un but se réalise par la réalisation des buts B1 ET/OU B2) ou des relations de contribution lorsqu'un but influence positivement ou négativement un autre but.
- **Sources internes et externes :** Cette couche représente le modèle source de notre système. Nous avons principalement trois sources de données dans ce modèle : les fragments des données LOD, les ontologies externes extraites à partir des LODs, l'ontologie interne ou partagée.
- **L'entrepôt de données sémantique (le schéma cible) :** Cette couche représente le schéma cible de données. On note que l'arbre de but s'alimente à partir des données de ce schéma cible. Cependant, il doit être peuplé par un ensemble de données venant de sources différentes (Internes et externes) et possédant des formats hétérogènes. Le but du schéma cible est d'avoir une seule source de données commune qui répond aux besoins exprimées et qui contient une unification et une harmonisation des différents formats de données extraites à partir de l'ensemble des sources précédemment définis.
- **Processus d'intégration de données et de métadonnées :** Il représente le processus ETL (Extraction – Transformation – Chargement). C'est la couche intermédiaire qui assure l'extraction et l'unification des données venant du schéma source allant vers le schéma cible en suivant un ensemble de mappings ETL approprié pour peupler chaque classe de l'entrepôt de données.

4 Architecture globale du système

L'architecture de notre solution comporte quatre couches essentielles. Nous verrons en premier lieu l'architecture logicielle que nous avons choisie. Juste après, nous allons présenter la conception de la couche client, ensuite nous allons détailler la couche traitement qui contiendra les différents modules, nous verrons ensuite la conception de la couche service. Enfin nous allons, nous détaillerons après la conception de la couche persistance qui représente la couche de données.

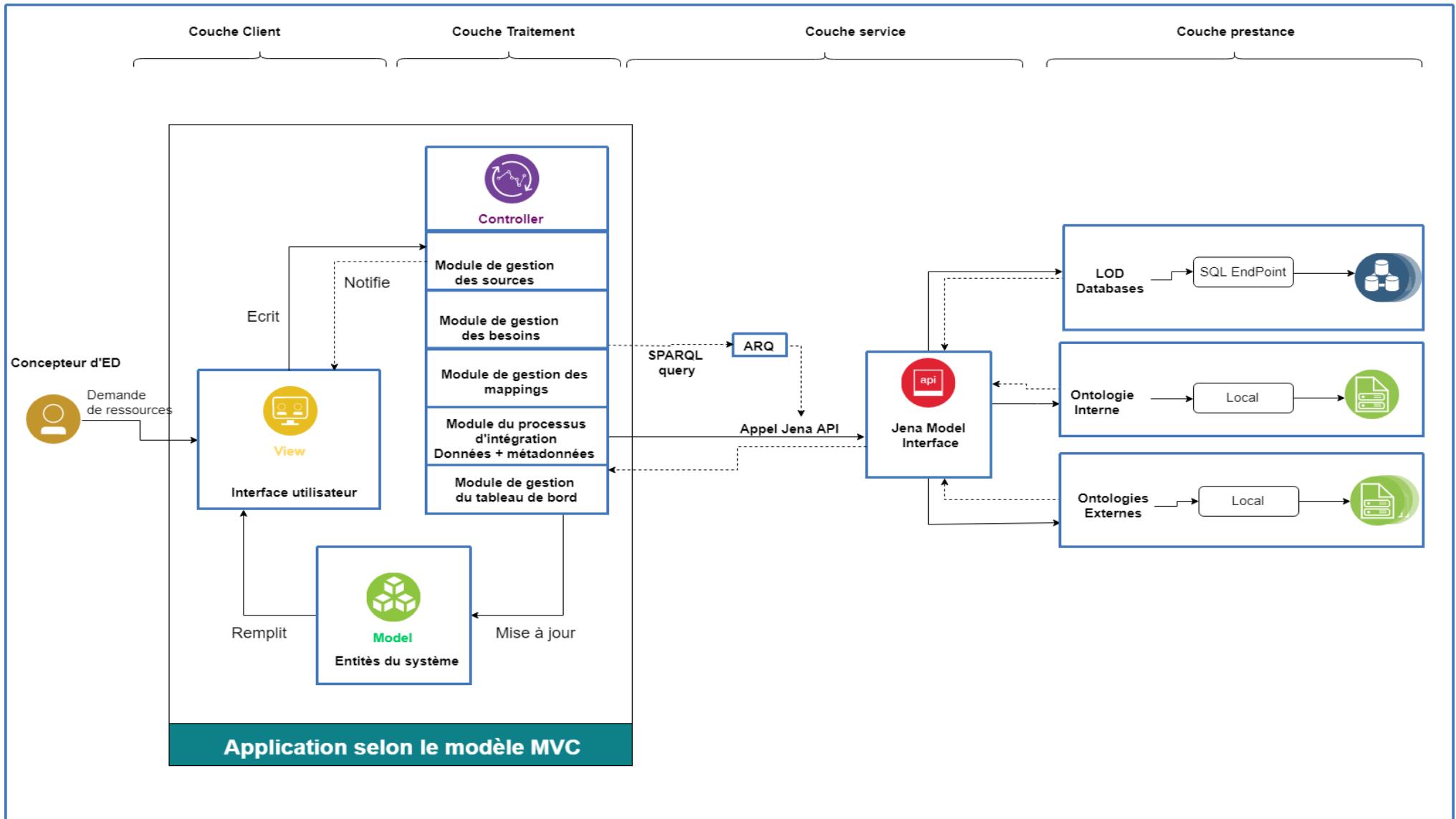


Figure 29 : L'architecture globale de la solution

Dans la figure 29, nous distinguant quatre couches principales dans l'architecture de la solution : la couche client, la couche traitement, la couche service et la couche persistance.

Les interactions entre les différentes couches se font dans l'application en s'appuient sur le modèle MVC (Modèle-Vue-Contrôleur) qui définit l'architecture logicielle de notre système. L'utilisateur demande des ressources à l'application à partir des vues, le contrôleur reçoit la demande, et lance des requêtes et interroge le service Jena pour récupérer ou écrire les données dans la couche persistance. Le contrôleur reçoit la réponse et met à jour le modèle qui remplit les vues. L'architecture MVC permet de bien organiser le code source et de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts comme suit :

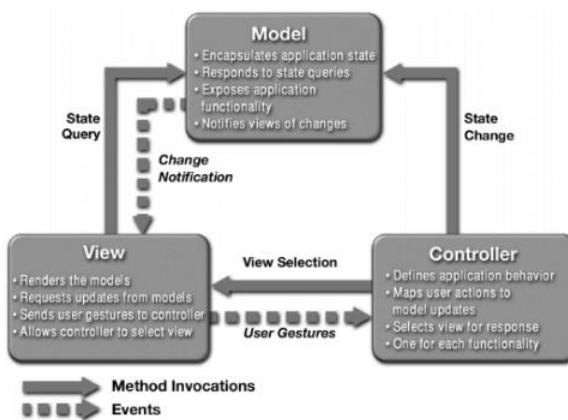


Figure 30 : Le modèle "MVC". Source : <http://java.sun.com/blueprints/patterns/MVC-detailed.html>

- **Modèle :** Celui-ci comprend les données et un certain nombre de méthodes pour les lire et les modifier. Le modèle n'a aucune connaissance de la façon dont les données sont présentées à l'utilisateur (la Vue). Le modèle peut toutefois enregistrer une ou plusieurs vues (une liste de dépendants) qui sont notifiées quand les données du modèle ont subi une modification. En Java, le modèle comprend une ou plusieurs classes (ou d'une interface ou classe équivalente).
- **Vue :** Une vue fournit une représentation graphique de tout ou partie des données du modèle. Chaque vue obtient les données à afficher en faisant la demande au modèle. Quand un utilisateur manipule une vue d'un modèle, la vue informe le contrôleur du changement désiré. En Java, les vues sont construites à base de composants JavaFX.
- **Contrôleur :** Les vues sont associées à des contrôleurs qui mettent à jour le modèle. Chaque vue est associée à un unique contrôleur. Le contrôleur interprète les actions de l'utilisateur (par exemple augmenter la température) et appelle les méthodes requises du modèle pour le mettre à jour. Le modèle informe alors toutes les vues qu'un changement est intervenu. Ces dernières se mettent à jour. Il peut aussi vérifier les actions de l'utilisateur et modifier la vue en conséquence si besoin.

Dans notre système, nous avons séparé les contrôleurs, les vues, et le modèle dans des packages différents afin de mieux distinguer les composants entre eux. Cette architecture logicielle nous permet de bien organiser nos classes. Nous l'avons également choisi pour faciliter l'évolution du système par

les perspectives citées en fin du rapport notamment concernant la prise en compte de nouvelles sources de données.

Le modèle MVC met donc en relation les trois couches Client, Traitement et Persistance (Voir figure 29), que nous allons détailler dans ce qui suit.

4.1 Couche client

Dans cette partie nous allons détailler la conception de la couche utilisateur. Celle-ci est constituée de deux points principaux. Le premier consiste à présenter la modélisation des différentes fonctionnalités du système déclenché par les utilisateurs. Le deuxième point consiste à présenter la modélisation de l'interface graphique de notre outil.

À la fin de cette section nous serons en mesure comprendre le système du point de vue de l'utilisateur. C'est principalement le côté qui exprime le plus, la valeur ajoutée de notre travail. Nous avons donc fourni un effort considérable dans la conception de la couche utilisateur, en matière d'ergonomie, d'intuitivité et de fonctionnalités disponibles.

4.1.1 Modélisation des fonctionnalités du système

La modélisation des fonctionnalités du système se fait grâce aux diagrammes de cas d'utilisation. C'est un diagramme qui peut résumer les détails des utilisateurs ainsi que leurs interactions avec le système. La réalisation d'un diagramme de cas d'utilisation nécessite un ensemble de symboles et de connecteurs spécifiques. Un diagramme de cas d'utilisation « use-case » peut aider à comprendre les :

- Scénarios dans lesquels le système ou application interagit avec des personnes, des organisations ou des systèmes externes.
- Les objectifs que le système ou application aide ces entités (appelées acteurs) à atteindre.
- La portée du système.

Dans ce qui suit nous allons représenter sous forme de cas d'utilisation les différentes spécifications fonctionnelles discuté dans la partie d'expression des besoins.

a) Gestion des ontologies internes et externes

L'utilisateur sera en mesure d'ouvrir les ontologies internes et externes afin de les consulter et les exploiter en tant que sources durant le processus ETL.

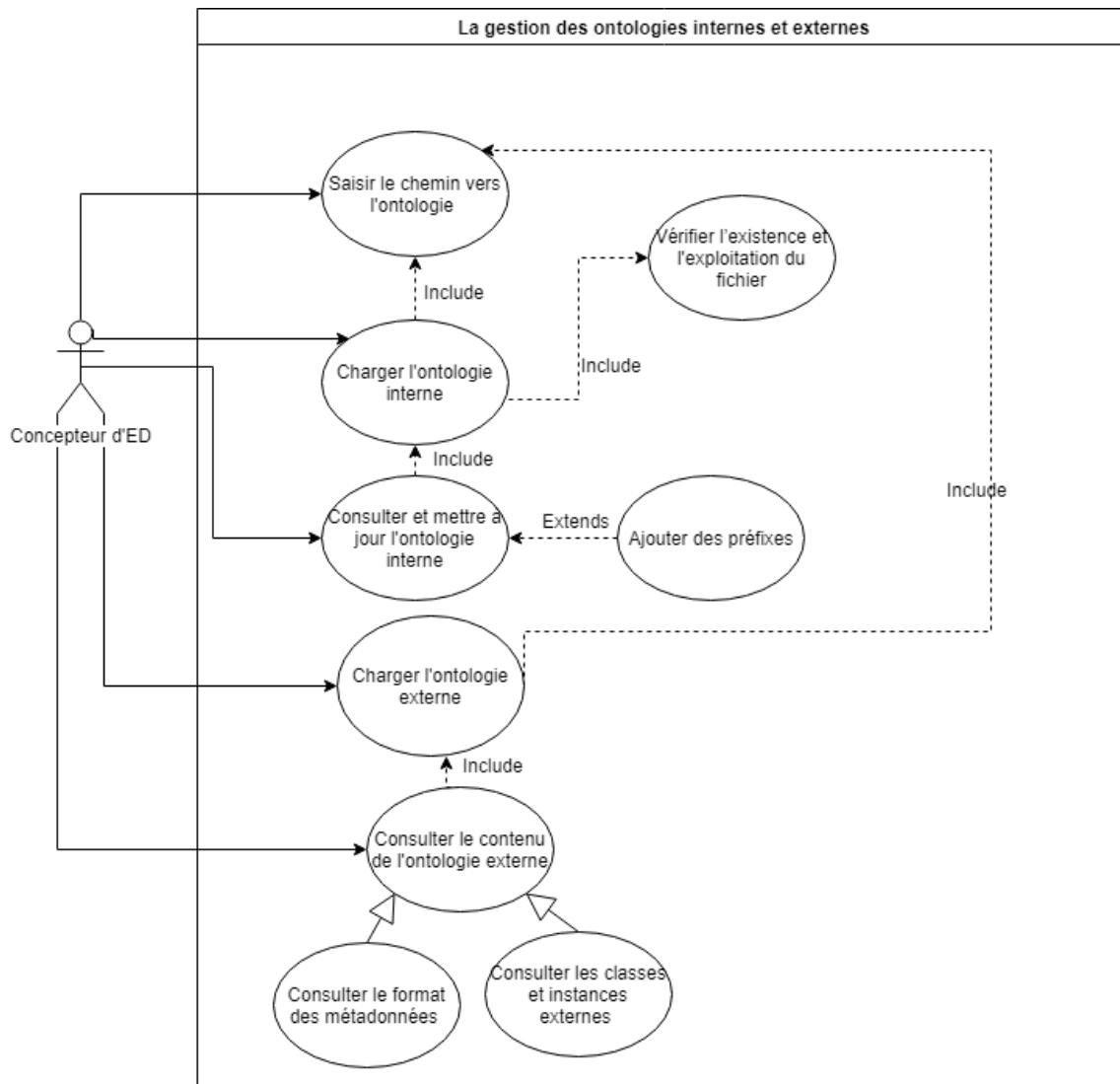


Figure 31 : Diagramme de cas d'utilisations de la gestion des ontologies sources

Nous allons ensuite documenter les cas d'utilisation, et décrire l'enchaînement de chacun.

Documentation du cas d'utilisation : Gestion des ontologies sources

CU : Gestion des ontologies sources
ID : 01
Description brève : Gérer les sources de données (Ontologies internes et externes)
Acteur primaire : Le concepteur d'EDS
Acteurs secondaires : /
Précondition : Application démarrée
Enchaînement principal : Ce cas démarre lorsque l'utilisateur souhaite démarrer l'application.
1. L'utilisateur démarre l'application. 2. L'utilisateur va dans l'onglet « Sources internes » ou « Sources Externes ». 3. L'utilisateur saisi le chemin vers l'ontologie. 4. L'utilisateur charge l'ontologie « interne » ou « externe » en cliquant sur le bouton « Open ». 5. L'utilisateur peut ajouter un nouveau préfixe à l'ontologie interne. 6. L'utilisateur clic sur « View classes » pour consulter les classes internes. 7. L'utilisateur clic sur « View instances » pour afficher les instances de la classe externe. 8. L'utilisateur clic sur « View format » pour afficher le format des métadonnées de la classe externe.
Post-Condition : Ontologies sources chargées
Enchaînement alternatif : Ontologies non reconnues

b) Gestion des besoins

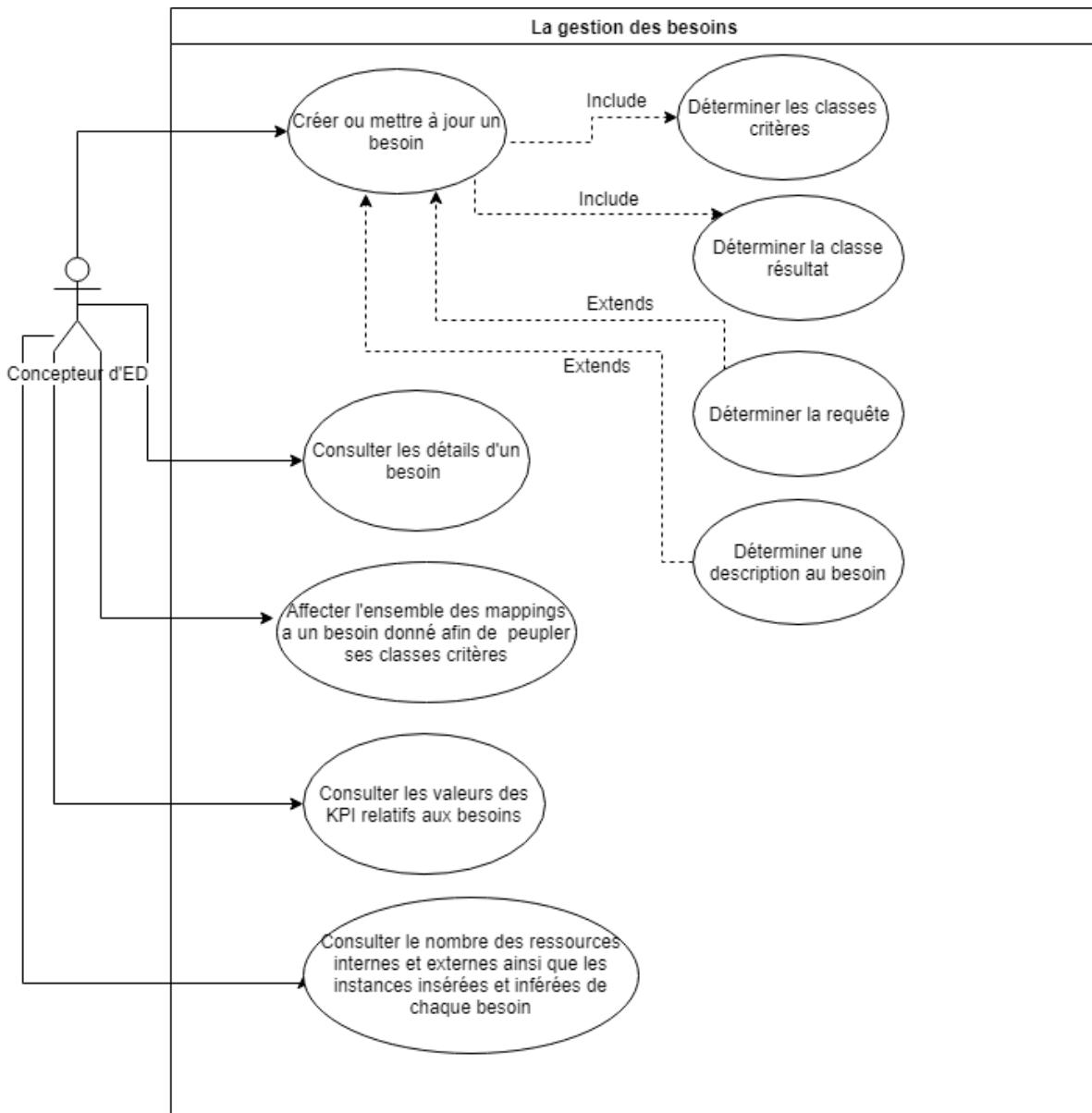


Figure 32 : Cas d'utilisations de la gestion des besoins

Documentation du cas d'utilisation : Gestion des besoins

CU : Gestion des besoins
ID : 02
Description brève : Gérer les besoins
Acteur primaire : Le concepteur d'EDS
Acteurs secondaires : /
Précondition :
1. Application démarrée 2. Ontologie interne chargée
Enchainement principal : Ce cas démarre lorsque l'utilisateur souhaite gérer les besoins. 1. L'utilisateur clique sur « Requirements » dans le menu. 2. L'utilisateur va dans l'onglet « Requirements ».

3. L'utilisateur choisit le besoin à consulter dans la liste déroulante.
4. L'utilisateur clique sur l'onglets « Add » pour ajouter un nouveau besoin.
5. L'utilisateur clique sur l'onglets « Update » pour modifier un besoin existant
6. L'utilisateur clique sur l'onglet « Delete » pour supprimer un besoin.

Post-Condition : Besoins cernés

Enchainement alternatif :

c) Gestion des mappings

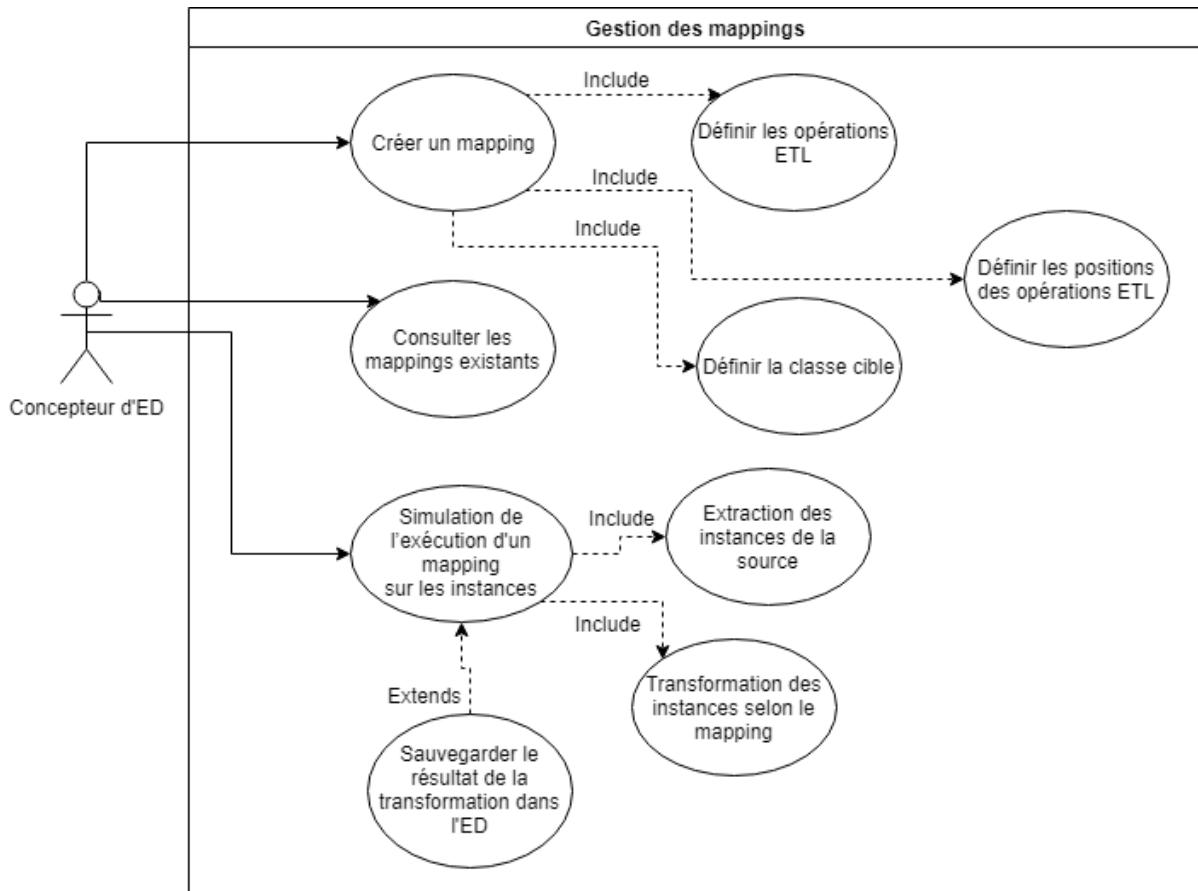


Figure 33 : Diagramme de cas d'utilisations de la gestion des mappings

Documentation du cas d'utilisation : Gestion des mappings

CU : Gestion des mappings
ID : 03
Description brève : Gérer les mappings du processus ETL
Acteur primaire : Le concepteur d'EDS
Acteurs secondaires : /
Précondition :
1. Application démarrée 2. Ontologie interne chargée
Enchainement principal : Ce cas démarre lorsque l'utilisateur souhaite gérer les mappings avant d'entamer le processus ETL.
1. L'utilisateur clic sur « Mappings » dans le menu. 2. L'utilisateur va dans l'onglet « Saved mappings ». 3. L'utilisateur sélectionne un mapping existant, l'ensemble de ses opérations ainsi que sa classe cible sont affichés. 4. L'utilisateur va dans l'onglet « Create a mapping ».

5. L'utilisateur saisi le nom, la classe cible et les différentes opérations constituant un mapping puis clic sur « Create ».
6. L'utilisateur va dans l'onglet « Simulation ».
7. L'utilisateur sélectionne un mapping et clic sur « Simulate » pour démarrer l'exécution du mapping.

Post-Condition : Mappings définis et testés

Enchainement alternatif :

d) Gestion du processus d'intégration des métadonnées

- Extraction

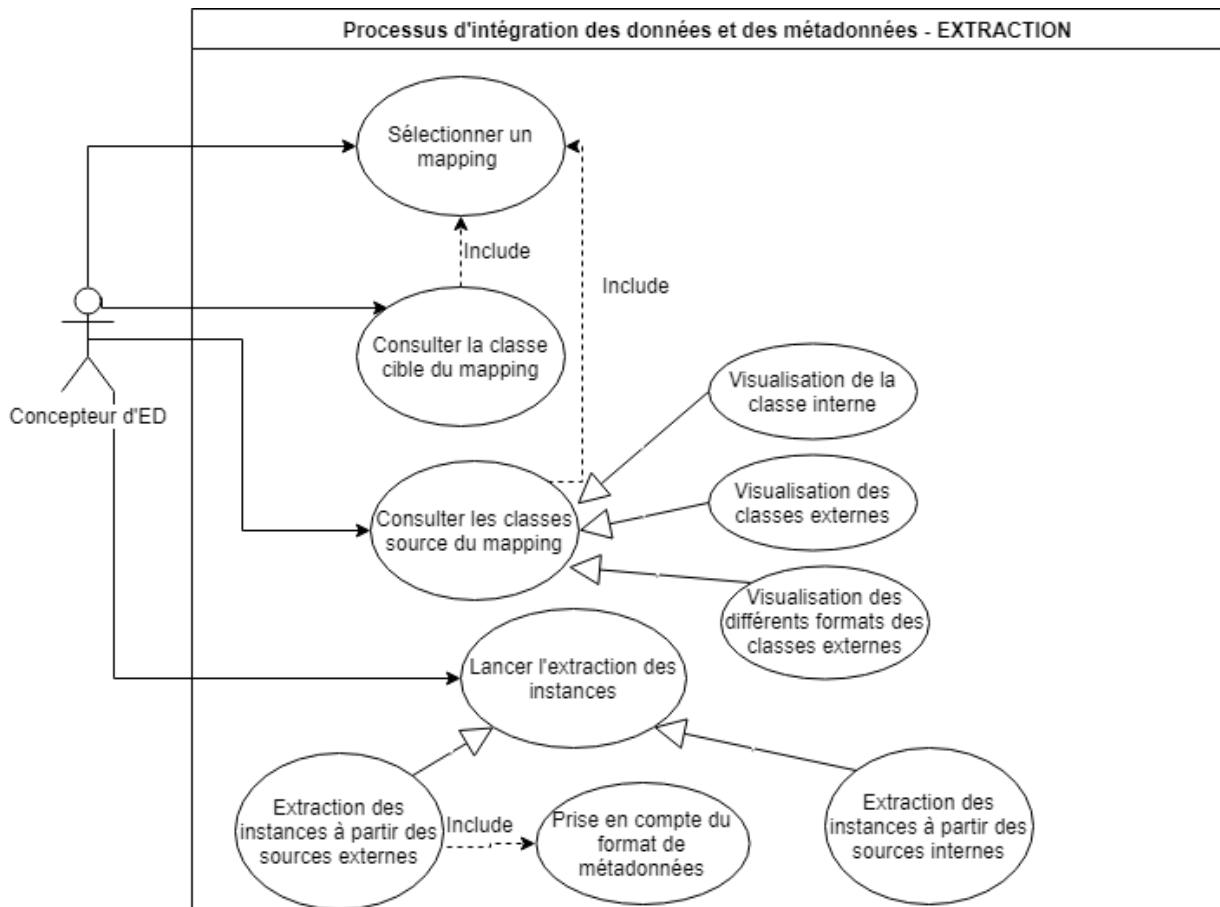


Figure 34 : Cas d'utilisation de la gestion de l'extraction des instances à partir des sources

Documentation du cas d'utilisation : Processus d'extraction

CU : Processus d'extraction
ID : 04
Description brève : Extraction des instances à partir des sources
Acteur primaire : Le concepteur d'EDS
Acteurs secondaires : /
Précondition :
1. Application démarlée 2. Ontologie interne chargée 3. Ontologies externes chargées
Enchainement principal : Ce cas démarre lorsque l'utilisateur souhaite gérer les mappings avant d'entamer le processus ETL. 1. L'utilisateur clic sur « ETL process » dans le menu.

- | |
|--|
| 2. L'utilisateur va dans l'onglet « Extraction ». |
| 3. L'utilisateur sélectionne un mapping existant, ses classes sources ainsi que sa classe cible s'affichent. |
| 4. L'utilisateur clic sur « Extract » pour que les instances des sources (Internes/externes) s'affichent. |

Post-Condition : Instances sources extraites

Enchainement alternatif :

- Transformation

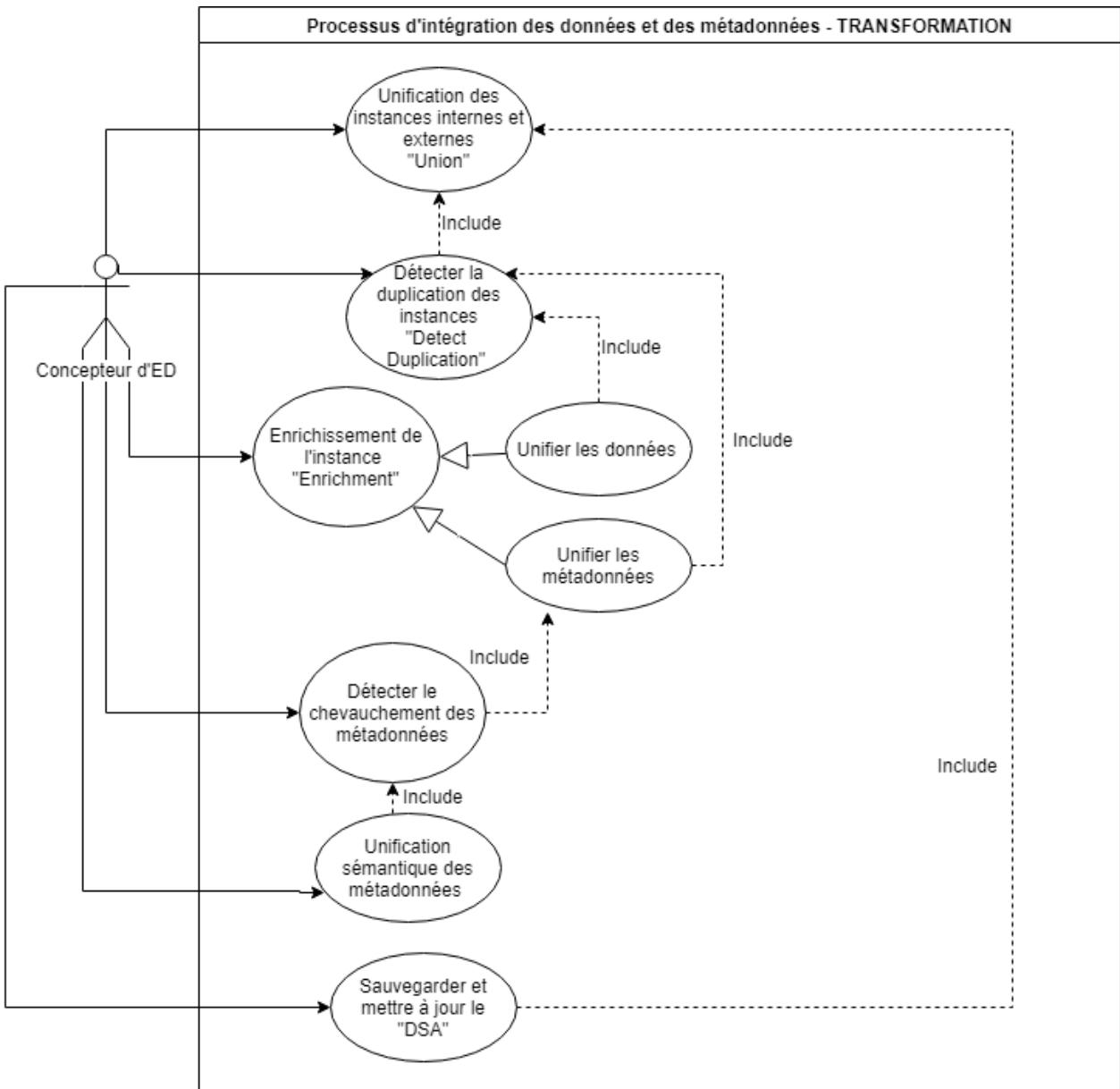


Figure 35 : Cas d'utilisation de la phase de transformation

Documentation du cas d'utilisation : Processus de transformation

CU : Processus de transformation
ID : 05
Description brève : Transformation des données et des métadonnées
Acteur primaire : Le concepteur d'EDS
Acteurs secondaires : /

Précondition :

1. Application démarrée
2. Ontologie interne chargée
3. Ontologies externes chargées
4. Mappings prédefinis
5. Extraction faites

Enchainement principal :

Ce cas démarre lorsque l'utilisateur souhaite gérer les mappings avant d'entamer le processus ETL.

1. L'utilisateur clic sur « ETL process » dans le menu.
2. L'utilisateur va dans l'onglet « Transformation ».
3. L'utilisateur clic sur « Union » pour unifier les instances internes et externes extraites précédemment dans le Data Staging Area « DSA ».
4. L'utilisateur clic sur « Detect duplciton » pour détecter et remplir la liste des instances dupliquées.
5. L'utilisateur sélectionne une instance dupliquée ensuite clic sur « Enrichement » pour effectuer l'union des données et des métadonnées des instances dupliquées pour en faire qu'une seul.
6. L'utilisateur clic sur « Detect overlapng » pour détecter le chevauchement entre les métadonnées unifiées.
7. L'utilisateur choisi une méta propriétés parmi la liste déroulante pour afficher les métadonnées en chevauchement (en conflit).
8. L'utilisateur choisit de sélectionner la bonne métadonné parmi celles proposées ou bien de toutes les garder ou les supprimer dans le cas où ça représente une mauvaise information.
9. L'utilisateur clic sur « Save changes » pour supprimer les instances dupliquées dans le DSA et les remplacer par l'instance avec les données et métadonnées unifiées et nettoyées.

Post-Condition : Transformation**Enchainement alternatif :**

- **Chargement**

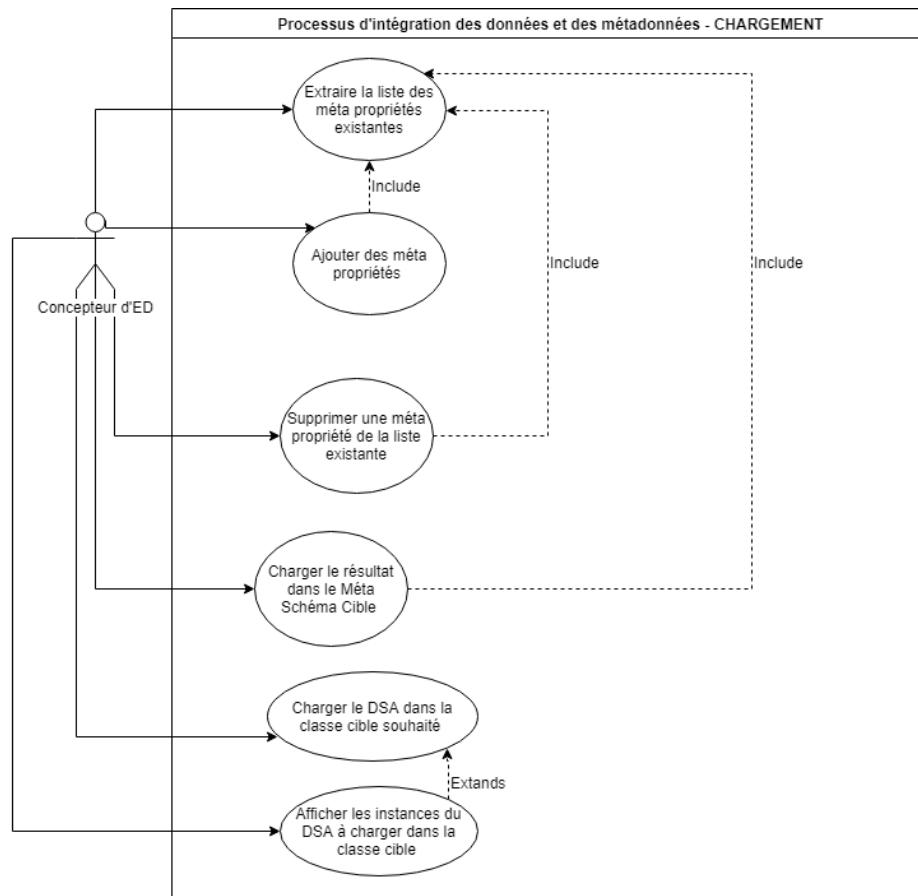


Figure 36 : Cas d'utilisation de la phase de chargement dans l'EDS

- Aperçus de l'entrepôt de données sémantique

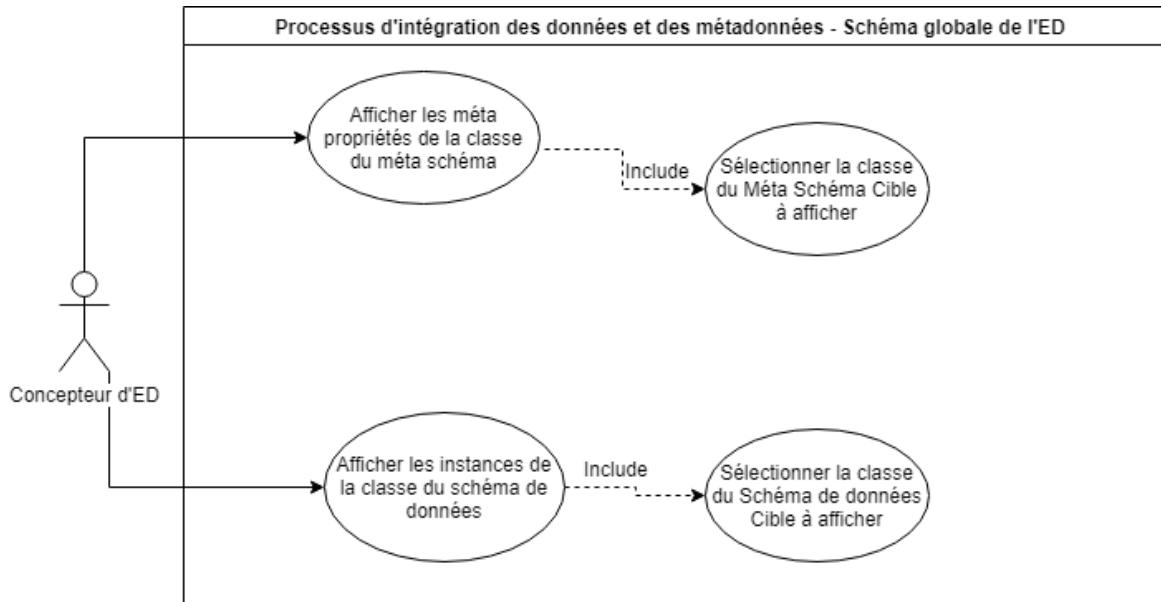


Figure 37 : Cas d'utilisation de la phase de consultation de l'EDS

4.1.2 Modélisation des interfaces graphiques

La conception de l'interface graphique représente une étape primordiale lors de tout projet de construction d'application logicielle. Elle représente le pont ultime avec l'utilisateur final, il se peut que l'application puisse répondre parfaitement aux besoins des utilisateurs, mais dans le cas où l'IHM (Interface Homme Machine) est non intuitive et difficile à manier, l'utilisateur ne trouvera aucun intérêt à l'utiliser.

Pour cela nous avons fourni un effort considérable concernant la conception des IHM de notre application pour qu'elles soient simples et intuitives pour l'utilisateur. Nous présenterons dans cette partie les différentes maquettes que nous allons ensuite mettre en œuvre pour que les utilisateurs finaux puissent interagir avec notre système.

Il faut noter que la 1^{ère} version de notre produit, les maquettes ainsi que les interfaces de l'application seront écrites en anglais. Le choix de la langue anglaise pour représenter les maquettes ainsi que les différentes interfaces de notre application revient au fait que l'application servira en tant que démonstration et support technique des articles de recherche en anglais émis par mon maître de stage dans ce sujet.

Nous verrons dans ce qui suit les maquettes les plus importantes :

- **Maquettes relatives à la gestion des sources :**

La maquette ci-dessous va représenter la 1^{ère} page qui s'affichera dans l'application. Elle permet de charger les ontologies internes et externes de l'organisme.

La maquette de la page de chargement des sources internes est une interface utilisateur pour gérer l'ontologie interne d'un Semantic Data Warehouse. Elle se compose d'un menu latéral sur la gauche avec les options Home, Requirements, Mappings, ETL et Dashboard. En haut à droite, il y a deux onglets : Internal sources et External sources. Le contenu principal affiche un formulaire pour ouvrir l'ontologie interne : "Open the internal ontology :" avec un champ "Internal ontology path :" et un bouton "Open Ontology". En dessous, il y a un champ "Enter prefix to add :" et un bouton "Add prefix".

Figure 38 : Maquette de la page de chargement des sources internes

- **Maquettes relatives à la gestion des besoins :**

La maquette dans la (figure 39) va permettre de gérer les besoins présents dans le schéma de l'ontologie interne. Plusieurs opérations peuvent être réalisées sur un besoin dont la visualisation des besoins existants, ajouter, supprimer et mettre à jour les besoins.

La maquette de la page de gestion des besoins est une interface pour gérer les besoins (Requirements) dans le Semantic Data Warehouse. Le menu latéral (Requirements) est actif. Le tableau de bord contient des boutons pour View, Add, Update et Delete. La partie centrale affiche les détails d'un besoin sélectionné ("Requirement 1") et fournit des champs pour la Description et l'Execution result. Il existe également un champ Query pour exécuter des requêtes.

Figure 39 : Maquette de la page de gestion des besoins

- **Maquettes relatives à la gestion des mappings :**

La maquette ci-dessous va représenter l'ensemble des opérations qu'un concepteur peut exécuter sur les mappings. Il existe trois principales opérations : la consultation, la création et la simulation du résultat d'un mapping.

La maquette montre une interface web pour la gestion des mappings dans un Semantic Data Warehouse. Le menu latéral gauche inclut les options Home, Requirements, Mappings (sélectionnée), ETL et Dashboard. La partie principale affiche les détails d'un mapping nommé "Mapping 1". Les champs "Mapping name" et "Target DWH class" sont vides. La section "ETL Operations" contient un champ pour l'ensemble des opérations. En bas, un diagramme de processus ETL est illustré par des cercles reliés par des flèches, nommé "Retrieve S1 -> Retrieve S2 -> Union S1+S2 -> Load".

Figure 40 : Maquette de la page de consultation des mappings existants

- **Maquettes relatives au processus ETL :**

La maquette dans la figure suivante va représenter la phase d'extraction des instances à partir de différentes sources.

La maquette montre une interface web pour l'extraction de données. Le menu latéral gauche inclut les options Home, Requirements, Mappings (sélectionnée), ETL (surligné en bleu) et Dashboard. La partie principale affiche les instances extraites à partir de classes définies. Un champ "Choose a mapping" indique "ETL 1". Les sections "Source Classes" et "Target class" sont vides. La section "Extract instances" contient deux groupes de champs : "Internal instances" et "External instances", chacun avec un bouton "Extract" en bas.

Figure 41 : Maquette de la page d'extraction des instances

La maquette dans la figure suivante va représenter la phase de transformation et d'unification des données et des métadonnées.

Semantic Data Warehouse				
Home	Extraction	Transformation	Load	Data Warehouse schema
Requirements	Transformation operations : 1. Instances unification : <input type="button" value="Union"/> <input type="text"/> 2. Detect instances duplication : <input type="button" value="Detect Duplication"/> <input type="button" value="Duplicated instance 1"/>			
Mappings	3. Data and Metadata unification for the selected instance: <input type="button" value="Enrichement"/> <input type="text"/> List of unified metadata : <input type="text"/> <input type="text"/> List of unified data :			
ETL	4. Detect metadata overlapping : <input type="button" value="Detect overlapping"/> <input type="button" value="Metadata 1"/> List of overlaped metadata : <input type="text"/> <input type="text"/> List of metadata to keep			
Dashboard	5. Save operations and update the DSA : <input type="button" value="Save changes"/>			

Figure 42 : Maquette de la page de transformation et unification des données et des métadonnées

La maquette dans la figure suivante va représenter dans l’application l’interface qui permet de gérer les opérations de chargement des méta propriétés dans le *TargetMetaSchema* d’abord ensuite des différentes instances du DSA (Données + métadonnées) dans le *TargetDataSchema*.

Semantic Data Warehouse				
Home	Extraction	Transformation	Load	Data Warehouse schema
Requirements	Loading data and metadata in the Data Warehouse : 1. Load the extracted meta properties in the MetaSchema : Operations : <input type="button" value="List extracted properties"/> List of extracted metaproPERTIES : <input type="button" value="MetaproPERTY 1"/> <input type="button" value="Delete"/> <input type="button" value="Load in Meta Schema"/> <input type="button" value="Add"/>			
Mappings	2. Load the Data Staging Area (DSA) in the Data Schema : Data Schema Classes : <input type="button" value="Class 1"/> Instances of DSA to load : <input type="button" value="List DSA content"/> <input type="button" value="Load DSA in selected Data Schema class"/>			
ETL				
Dashboard				

Figure 43 : Maquette de la page de chargement des données et métadonnées transformés

4.2 Couche persistance

La couche persistance correspond aux différents modèles de données de notre système. Nous verrons dans cette partie les différents modèles et schémas de données essentiels pour la solution. Nous énumérons en tout un modèle de données général et quatre schémas de données. L’ensemble de ces modèles nous permet de faire fonctionner notre système et répondre au mieux aux spécifications ce celui-ci.

Nous verrons donc en premier lieu le modèle de données générale, ensuite les différents schémas des données sources internes et externes, nous verrons par ailleurs les schémas de la cible (Données et métadonnées). En quatrième lieu nous allons détailler le schéma des besoins, celui-ci nous permettent de garder trace des différents besoins exprimés par les utilisateurs. Nous verrons enfin le schéma qui permet de garder trace des différents mapping ETL.

4.2.1 Modèle de données général

Dans ce qui suit nous décrirons le modèle de données générale grâce au diagramme des classes illustrées dans la (Figure 44).

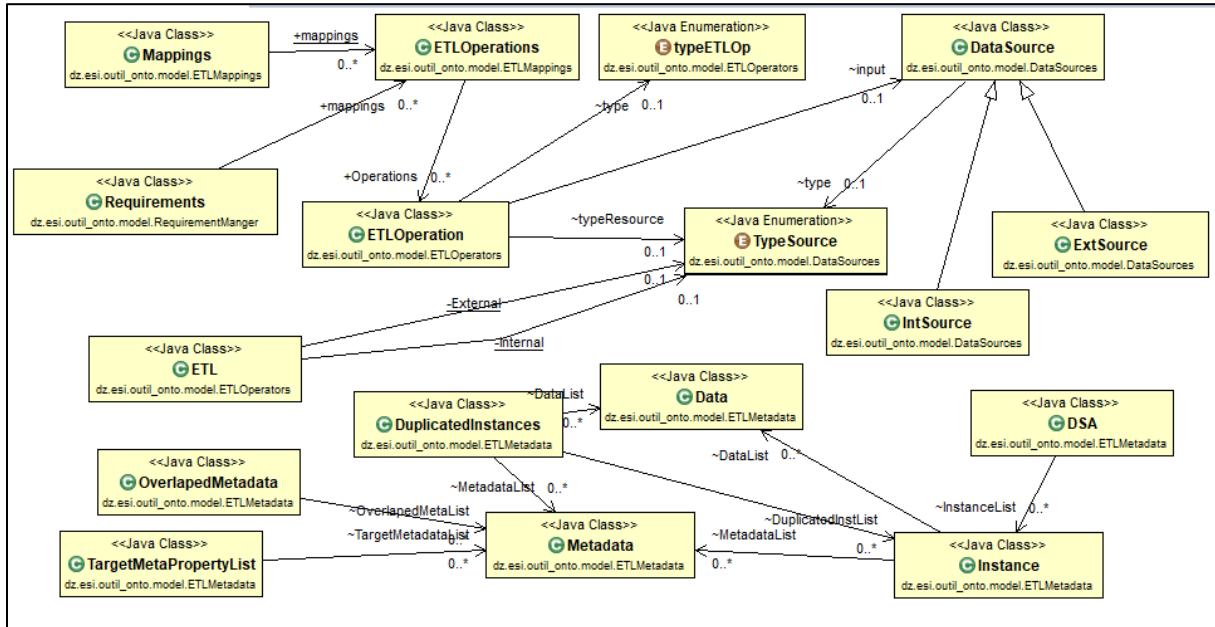


Figure 44 : Diagramme de classes du système

Pour concevoir notre système, et afin de manipuler les données lors du traitement de celles-ci, il est primordial de les modéliser en un ensemble de classes, comportant des attributs et des méthodes.

Le tableau suivant va décrire l'ensemble des classes et de leurs attributs.

Classe	attribut	Description
DataSource	typeSource	Le type de la source (Interne ou externe)
	dataset	Le dataset de la source
ExtSource	ressource	L'IRI complet de la ressource externe
	ressourcePath	Le chemin vers l'ontologie externe
	Format	Le format de métadonnées utilisé
IntSource	ontModel	Contient le graphe de données interne
ETLOperation	opName	Le nom de l'opération donné par le concepteur
	Ressource	Ressource (IRI) qui nous permet d'identifier la source de données
	Position	Position de l'opération dans le mapping
	opType	Le type de l'opération (Retrieve, Union, Merge ...)
	sourceType	Le type de la source (Interne, Externe)
	format	Format de la métadonnée (Nary, Reif, Ngraphs, ...).
	path	Le chemin vers l'ontologie local.
ETLOperations	ETLName	Le nom du mapping ETL.
	operationList	La liste des opérations (Liste de ETLOperation).
	target	La classe cible de l'ETL.
	internalClasses	La liste des ressources internes.
	externalClasses	La liste des ressources externes.
	formats	La liste des formats présents dans le mapping ETL.
Mappings	ETLOperationsListe	Il contient la liste des différents mapping ETL.
Requirements	RequirementName	Le nom du besoin.

	criterias	La liste des critères du besoin.
	results	Les liste des résultats du besoin.
	spaqlQuery	La requête SPARQL du besoin.
	description	Description du besoin.
	sourceExt	La liste des sources externes du besoin.
	sourcesInt	La liste des sources internes du besoin.
	kpis	La liste des KPI du besoin.
	mappings	La liste des mapping du besoin (Liste de ETLOperations).
ETL	/	Classe pour la gestion de l'exécution des opérations ETL.
Data	dataProperty	La propriété de la donnée (Ex : Prix, Caractéristiques ...).
	dataValue	La valeur de la donnée (Ex : 2000\$, 20MPixel ...).
Metadata	metaProperty	La propriété de la métadonnée (Ex : Creator, Publisher ...).
	metaValue	La valeur de la données (Ex : Tim-berners-Lee, ...)
	format	Le format de la métadonnée.
Instance	instance	L'IRI complet de l'instance.
	dataList	La liste de données de l'instance en question.
	metadataList	La liste de métadonnées de l'instance en question.
DSA	instancesList	La liste des instances.
DuplicatedInstances	duplicatedInstList	La liste des instances dupliquées.
	dataListUnification	La liste des données unifiées.
	metadataList	La liste des métadonnées unifiées.
OverlapedMetadata	metadataProperty	La propriété de la métadonnées dupliquées.
	overlapedMetaList	La liste des métadonnées avec chevauchement.
TargetMetaProperties	metadataList	Contient la liste des métadonnées à charger dans le méta-schéma cible.

Figure 45 : Tabelau représentant la description de sdifferentes classes de la solution globale.

4.2.2 Schémas des sources

Pour pouvoir faire la conception d'un entrepôt de données sémantique nous devons d'abord définir nos sources de données. Comme définit précédemment nous avons deux types de sources de données, les sources internes et les sources externes

Les schémas des sources sont donc divisés comme suit :

a) Source interne :

Comme expliqué, les sources internes référencent une ontologie partagée entre, que nous utilisons comme schéma interne. Notons que diverses approches existent dans la littérature permettant de générer une ontologie partagée à partir d'un ensemble de sources [S. Khouri, 2013], dans notre travail nous supposons donc cette ontologie interne existante. Ce schéma sera étendu au fur et à mesure par des fragments de schémas externes selon les besoins exprimés. Pour illustrer nos propos, nous considérons comme étude de cas le benchmark¹³ qui décrit un entrepôt analysant les transactions d'un système de e-

¹³<http://wifo5-03.informatik.unimannheim.de/bizer/berlinsparqlbenchmark/spec/BusinessIntelligenceUseCase/index.html>

commerce. Vous trouverez ci-dessous le diagramme de classes de l'ontologie E-commerce utilisé avec les détails correspondants :



Figure 46 : Le diagramme de classe de l'ontologie interne. Source : [F. Irini, 2016]

Comme indiqué dans le diagramme, nous possédant en tout huit classes qui représentent de façon générale la conception du cas d'étude d'une entreprise d'E-commerce :

- Product
- Review
- Person
- Offer
- Vendor
- ProductType
- Product feature
- Producer

Afin de visualiser le schéma de l'ontologie nous utilisant l'outil « protège », c'est un outil qui nous permet de visualiser, gérer et mettre à jour une ontologie portant différents formats (OWL, RDF, TTL ...) (Voir la figure ci-dessous).

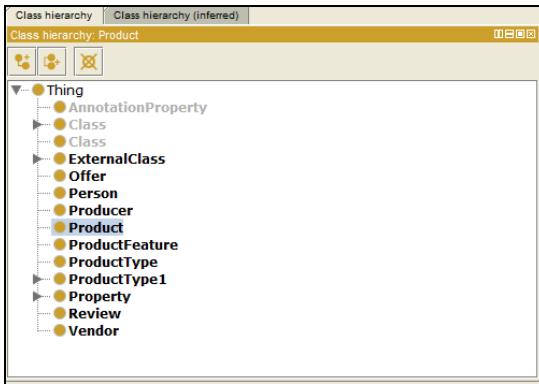


Figure 47 : L'ensemble des classes de l'ontologie interne E-commerce

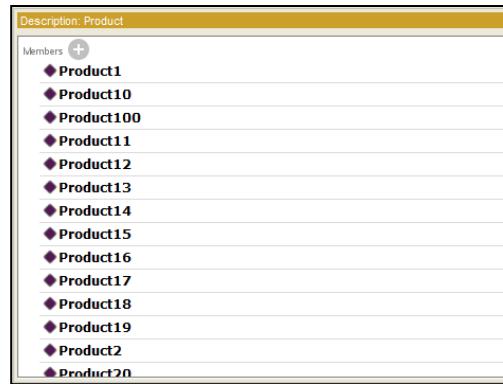


Figure 48 : Les différentes instances de la classe Product

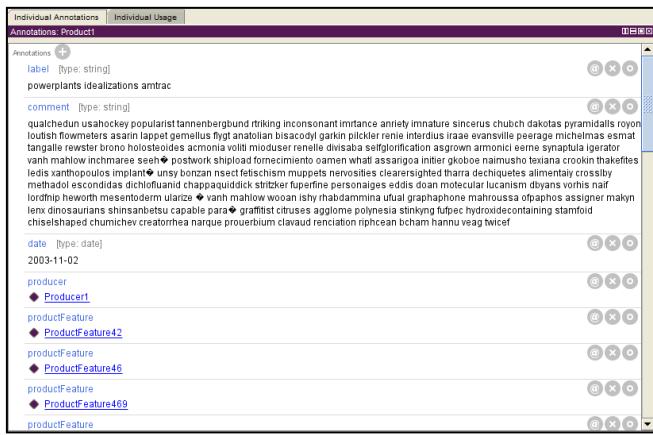


Figure 49 : L'ensemble des annotations (Données + Méta-Données) de 'instance « Product1 »'

b) Sources externes :

Représenté soit par les données du Linked Open Data comme (DBpedia, Yago, Dublin Core ...). Soit par des ontologies externes à notre ontologie partagée. Ces sources externes possèdent des données du même contexte que nos sources internes. Ces données peuvent être exploitées pour compléter et enrichir les données internes comme les classes suivantes : « Company » qui correspond à la classe interne « Product ». On distingue aussi « Supplier » pour la classe « Producer » dans l'ontologie interne.

Les ontologies externes peuvent avoir différents formats de métadonnées pour cela nous devons garder trace en local des différentes ontologies externes et de leurs formats pour pouvoir les exploiter dans le processus d'intégration ETL. Le schéma des ontologies externes est défini comme suit :

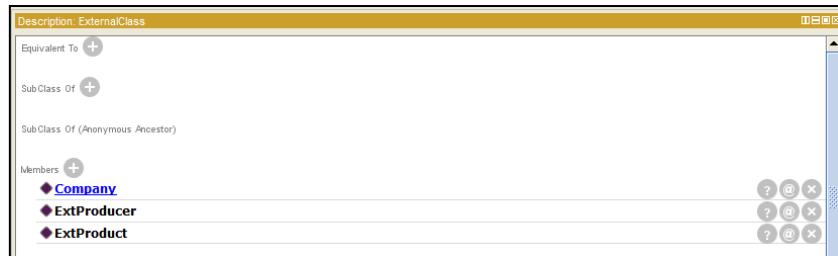


Figure 50 : Les instances des classes externes "ExternalClass"

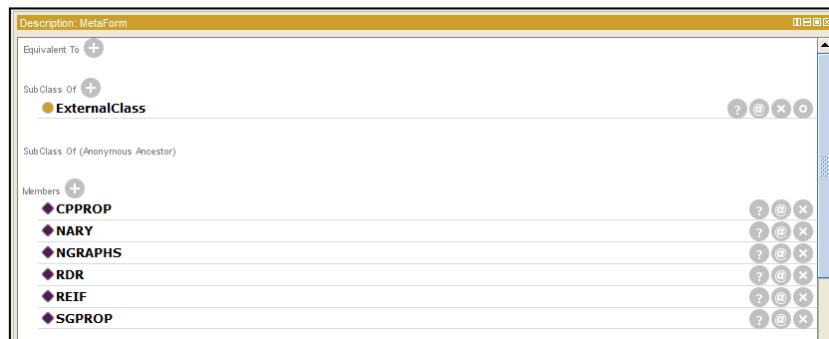


Figure 51 : Les instances des différents formats de données

Dans la conception nous avons considéré deux formats, STDREIF et NARY. Cependant nous allons définir les expressions des formats détaillé dans l'état de l'art en RDF/XL sont :

- **STDREIF :**

[Statement-ID1, sujet, Instance-Name]

[Statement-ID2, prédicat, Meta-Property]

[Statement-ID3, objet, Meta-Value]

- **NARY :**

[Instance-Name, Meta-Property1, Statement-ID]

[Statement ID, Meta-Property1-nary-value, Meta-Value1]

- **NAMED GRAPH :**

[Statement-ID1, type, Instance-Name-C]

[Instance-Name, data-property, data-value]

[Statement-ID, meta-property, meta-value]

- **Singleton Property :**

[Instance-Name, Statement-ID, data-value]

[Statement-ID, singleton-property-of, data-property]

[Statement-ID, Meta-property, meta-value]

- **Companion Properties :**

[Instance-Name, data-property.1, data-value;

data-property.1-SID, Statement-ID]

[data-property.1-SID, ID-property-of, data-property.1]

[data-property.1, companion-property-of, data-property]

[Statement-SD, leta-property, meta-value]

Au niveau de l'ontologie générale, nous distinguons les sources internes des sources externes ainsi que les liens avec les besoins. Toutes les traces sont sauvegardées.

4.2.3 Schémas de la cible

Le schéma cible dans notre système représente l'ontologie cible de l'EDS, elle décrit l'entrepôt de données sémantique qui va contenir l'intégration des différentes sources de données. Il contient deux modèles l'un des données et l'autre des métadonnées.

Le modèle de métadonnées que nous avons suivi, reviens à un modèle déjà existant

a) Description du méta-modèle cible : Semantic Metamodel for Analytical Metadata (SM4AM)

[J. Varga et al., 2018] proposent un méta-modèle sémantique pour les métadonnées analytiques nommé (SM4AM). L'objectif global de SM4AM est de classifier les métadonnées selon leur rôle. Il est formalisé en utilisant le langage RDF, ce qui correspond à notre contexte d'étude. De plus, l'utilisation de RDF permet à l'approche proposé d'effectuer une méta modélisation ontologique de façon à ce que les types de méta-modèles ontologiques ne nécessitent pas une conformité littérale au niveau du modèle.

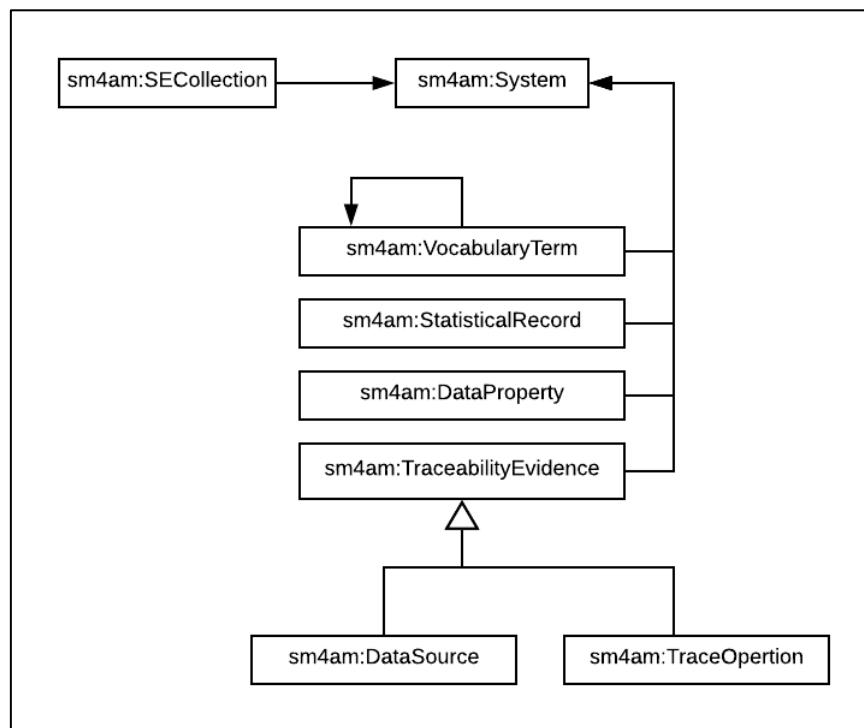


Figure 3.3. Le schéma global du méta-modèle sémantique SM4AM pour les métadonnées analytiques. Source : [J. Varga et al., 2018]

Le métamodèle (SM4AM) est conçu autour d'éléments concernant le système, l'utilisateur ou les deux. Les éléments concernant l'utilisateur concernent la gestion des métadonnées des requêtes posées par ce dernier, nous ne nous intéressons pas à cet aspect dans notre travail. Dans notre travail nous allons donc nous concentrer uniquement sur les éléments liés au système car il représente les types de métadonnées dans le system, et c'est l'un des majeurs objectifs de notre travail.

Au niveau système, nous trouvons les éléments suivants [J. Varga et al., 2018] :

- **Éléments liés à la traçabilité** : La traçabilité des MA est modélisé avec deux sous-classes de la métaclass *sm4am:TraceabilityEvidence* qui elle représente les éléments du métamodèle atomique. La première est *sm4am:DataSource* qui capture la source d'où proviennent les données. La seconde est *sm4am:TraceOperation* et représente une opération qui peut être effectuée sur des données ou métadonnées avant d'atteindre le référentiel de données/métadonnées. Notez que l'utilisation de la propriété *sm4am:attribute* au niveau du modèle est utile pour indiquer si la source de données est interne/externe et fiable/non fiable. De plus, il peut être utilisé pour établir un lien avec les valeurs de données particulières du schéma d'intégration. Un exemple d'éléments liés à la traçabilité peut être celui de *sm4am:DataSource* est instancié avec une classe de sources Open Data liées qui, à leur tour, a DBpedia comme instance.
- **Éléments liés au profilage** : Les métadonnées analytique de profilage sont modélisé avec la métaclass *sm4am:DataProperty* représentant la qualité technique des caractéristiques (ex. valeurs de cardinalité). Ces métadonnées sont généralement obtenues à partir du traitement du profilage des données afin d'améliorer la compréhension des utilisateurs de l'ensemble de données. Des propriétés de données spécifiques sont ensuite définies au niveau du modèle en fonction du système particulier. Par exemple, la cardinalité des composants de schéma peut être définie en tant qu'instance de propriété de données.
- **Élément lié au vocabulaire** : Le vocabulaire des MA est modélisé avec la métaclass *sm4am:VocabularyTerm* qui représente une entrée de vocabulaire comme élément constitutif de la construction du vocabulaire. De plus, la propriété *sm4am:mapsTo* relie deux termes de vocabulaire et définit la correspondance entre eux (par exemple, une relation synonyme). La métaclass *sm4am:VocabularyTerm* est instanciée au niveau du modèle avec les types d'entrées de vocabulaire concrets (par exemple, des termes commerciaux) et leurs liens qui, à leur tour, ont leurs instances.
- **Élément lié aux statistiques** : L'artefact statistique des MA est modélisé avec *sm4am:StatisticalRecord* comme élément atomique pour construire des statistiques. Comme précédemment, *sm4am:attribute* doit être instancié pour lier les enregistrements statistiques avec leurs valeurs. Ensuite, le niveau du modèle doit être utilisé pour définir la classe représentant le type d'indicateurs statistiques qui sont liés au type de données numériques spécifique comme leur valeur (par exemple, décimale), tandis que le niveau d'instance garde la trace des instances de l'indicateur et leur valeur. Il faut noter que les valeurs pour les métadonnées liées à *sm4am: StatisticalRecord* doivent provenir de la surveillance du système.
- **Éléments complexes liés au système** : Après avoir expliqué les éléments liés au système atomique, nous fournissons plus de détails sur la métaclass *sm4am:SEList* (i.e., liste de preuves système) pour les composer en listes ordonnées qui représentent différents concepts. Les éléments atomiques se composent d'une structure complexe via la propriété *sm4am:containsSE*. Les attributs (i.e., *sm4am:attribut*) de l'attribut devrait être utilisé pour déterminer l'organisation structurelle. Par exemple, nous pouvons avoir une trace complexe composée de sources de données et d'opérations de traçabilité alignées dans une structure de trace ordonnée. De même, nous pouvons également avoir un vocabulaire composé de termes de vocabulaire, des statistiques composées d'enregistrements statistiques et un profil de données composé de propriétés de données.

Pour construire le méta modèle cible nous nous sommes appuyés sur les éléments liés à la partie système du méta modèle existant (SM4AM) [J. Varga et al., 2018] répondant à nos besoins qui se basent

principalement sur l'utilisation des métadonnées analytiques en utilisant le format RDF. Nous allons donc commencer à partir d'une base existante et on sera amené à enrichir, améliorer et rapporter notre contribution à une partie du modèle existant afin de répondre à nos besoins.

b) Conception du schéma cible de l'EDS

Notre schéma cible représente l'EDS. Il est composé de deux sous schémas complémentaires. Le méta-schéma cible et le schéma de données cible.

En s'inspirant de la partie *système* du méta-modèle existant SM4AM, nous avons déterminé les composants de notre méta-schéma cible qui est représenté par la classe « MetaModel » (Voir figure 52). Il contient les cinq classes décrites ci-dessus. Nous trouvons donc les classes de métadonnées suivantes : DataProperty, SEList, StatisticalRecord, TraceabilityEvidence, VocabularyTerm. Notons que pour faire cela, nous avons affiché le méta-modèle de l'ontologie sous Protégé ayant comme principale méta-classe (OWL-Class), et nous l'avons étendu par les nouvelles métaclasses.

Nous avons aussi représenté le schéma de données cible, qui lui est représenté par la classe « DWClass ». Elle contient toutes les classes de données après intégration des sources internes et externes. Ces classes de données vont servir à alimenter et répondre aux besoins des entreprises définis selon le modèle orienté besoins vue précédemment.

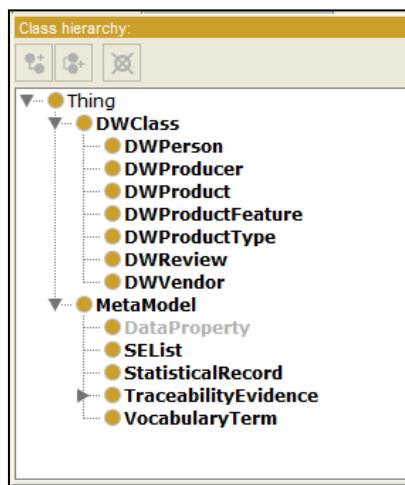


Figure 52 : Schéma des classes de données et de métadonnées de l'ontologie cible de l'EDS.

La classe « DWClass » contient l'ensemble des classes de données, c'est le *TargetDataSchema*. Tandis que « MetaModel » représente la classe contenant les classes du méta schéma cible « TargetMetaSchema ».

4.2.4 Schéma des mappings

Il existe deux types de mappings dans notre système. Il y a les mappings de données et il y a aussi les mappings de métadonnées ou méta-mappings. Nous détaillons chacune comme suit :

- **Mappings de données (Mapping ETL) :**

Il y a nécessité de garder trace des mappings d'intégration et de l'ensemble des opérations ETL correspondantes.

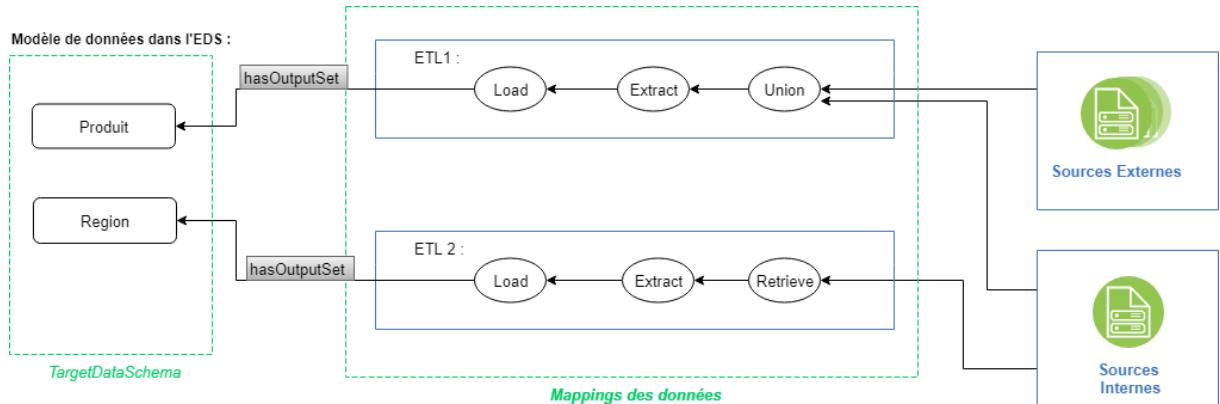


Figure 53 : Schéma représentant les mappings de données

Chaque mapping ETL est lié à deux types de propriétés « hasoutputSet » qui représente la classe cible de l'ontologie globale où charger le résultat de la transformation. Il possède aussi la propriété « hasinputSet » qui représente l'opération ETL à exécuter.

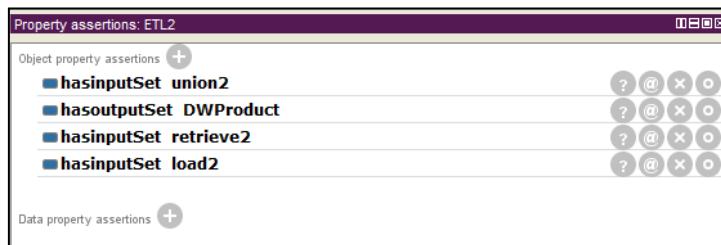


Figure 54: L'ensemble des propriétés de l'instance de mapping "ETL2"

Dans la (figure 53) nous illustrons un exemple de mapping sauvegardé dans l'ontologie globale. Nous avons un mapping nommé ETL2. Il possède comme classe cible « DWProduct » et comme opérateurs « retrieve2 », « union2 » et « load2 ». Cela est formalisé de la façon suivante.

- *ETL2.hasOutputSet(DWProduct)*
- *ETL2. HasInputSet (Extract(...), Union (...), Filter(...), Load (DWProduct))*

- **Mappings de métadonnées (Méta-mappings):**

Les métadonnées existant dans les sources de données internes et externes, passent aussi par un ensemble de mappings qui permettent de faire une correspondance entre les méta-propriétés LOD extraites et transformées à partir des sources et les lier avec leurs méta-classes cible selon le méta-modèle SM4AM (Voir section 4.3.3 – Point a).

La (figure 55) représente un exemple de méta-mapping, commençant par l'extraction des métadonnées (méta-propriétés et méta-valeurs), ensuite la transformation de celles-ci et en fin du chargement des méta-propriétés dans les classes du méta-modèle SM4AM.

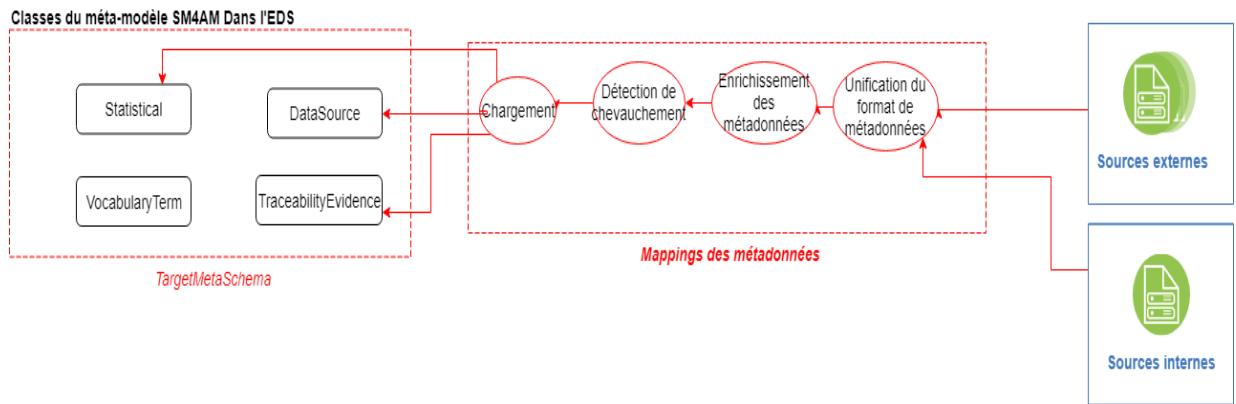


Figure 55 : Schéma représentant les méta-mappings

Les mappings qui correspondent au chargement des métapropriétés dans le métamodèle cible (*TargetMetaSchema*) sera illustré sur un exemple concret dans la réalisation plus loin dans le rapport. (Voir Partie 3 – Chapitre 1 – Section 2.1.3).

4.2.5 Schéma des besoins

Pour garder trace des besoins nous avons étendu le métamodèle ontologique cible pour permettre de modéliser les besoins.

La figure ci-dessous nous montre le schéma d'un besoin « Reuqirement_5 » qui contient deux propriétés objet « hasCriteria » qui définit les classes critères du besoin et « hasResult » qui définit la classe résultat du besoin. On trouve aussi deux propriétés de données « Description » un prédictat qui possède comme littérale un champ texte qui contient la description du besoin.

Les besoins sont sauvegardés pour pouvoir estimer l'effort de conception de notre démarche « orientée besoins », l'effort est estimé à travers les métriques (nombre ressources internes/externes, nombre mappings, nombre de métadonnées)) qui sont utiles au concepteur et qui lui seront affichées à travers un petit tableau de bord.

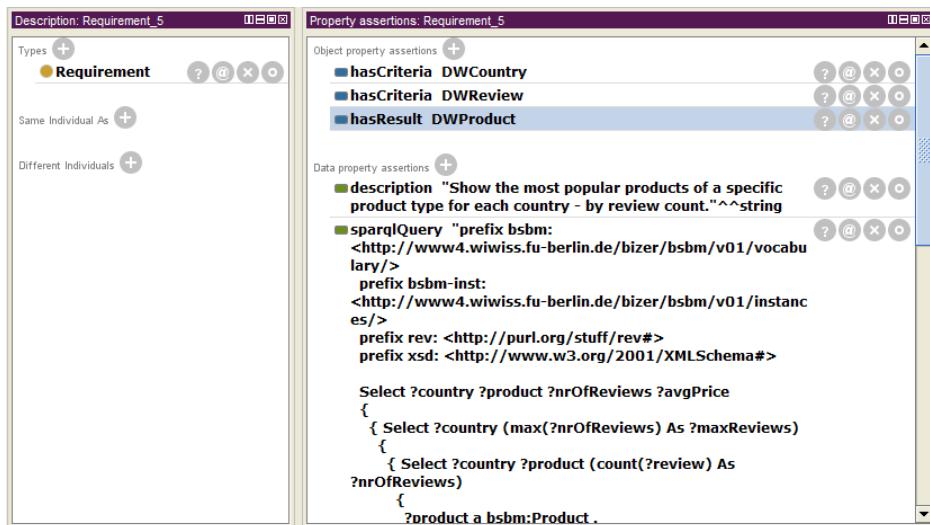


Figure 56 : L'ensemble des propriétés de l'instance "Requirement_5"

4.3 Couche traitement

Maintenant que nous avons présenté les schémas et les modèles de données, nous allons entamer la conception des différents modules de notre système. Dans ce qui suit, nous exposerons chaque module du système, où nous nous étaleront sur son fonctionnement, grâce aux diagrammes que nous jugeons les plus expressifs.

4.3.1 Module de la gestion des sources

Ce module va assurer la gestion de l'ontologie interne et des différentes sources de données externes. La figure ci-dessous représente le diagramme de séquence qui illustre l'interaction entre les quatre composants du système lors de la gestion de l'ontologie Interne.

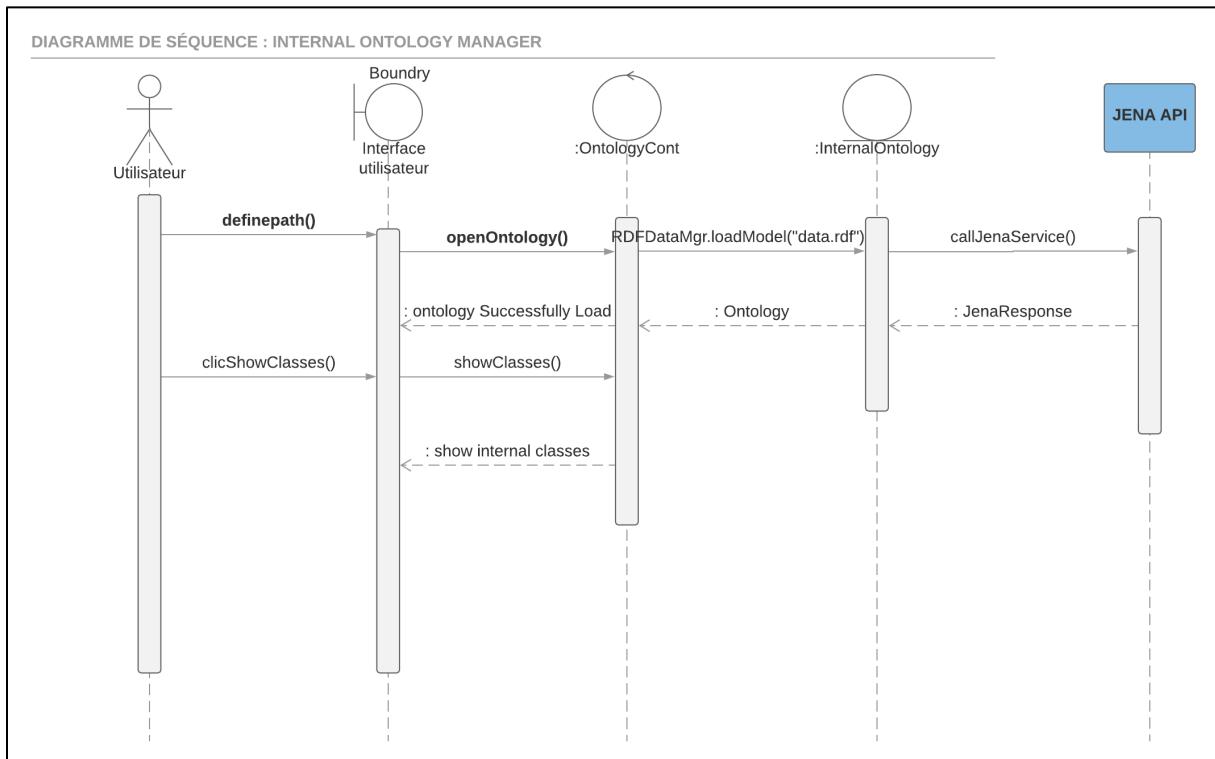


Figure 57 : Diagramme de séquence de la gestion des sources internes.

Au lancement du système il y a nécessité d'ouvrir et gérer l'ontologie interne et définir les différentes sources de données externes pour cela il existe une classe contrôleur qui regroupe les fonctionnalités de gestion de l'ontologie interne.

L'utilisateur définit d'abord le chemin vers l'ontologie interne ou externe, il clic sur « Open », il y'aura ensuite un appel au service « Jena API » à partir du contrôleur. Le service lance l'opération de chargement de l'ontologie et le renvoi au contrôleur. Pour l'afficher en cas de besoin de l'utilisateur.

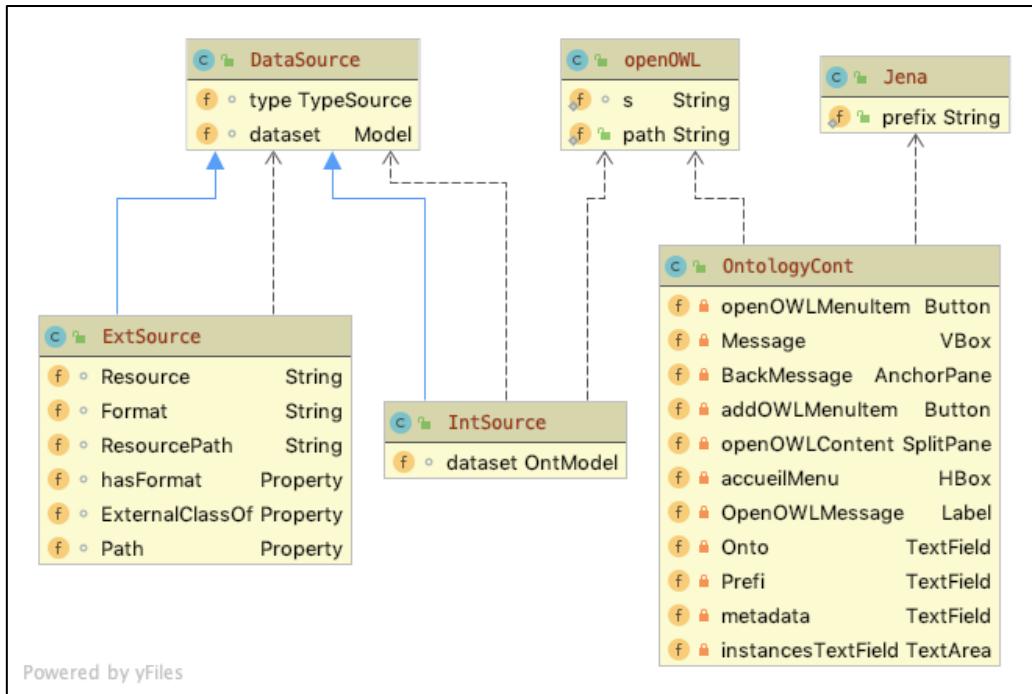


Figure 58 : Diagramme de classes montrons les interactions entre le contrôleur « OntologyCont » et les composants des ontologies sources.

4.3.2 Module de gestion des mappings

Un mapping est constitué d'un ensemble de classes sources, d'un séquencement d'opérations ETL (Retrieve, Union, Merge ...) et d'une classe cible.

Le module de gestion des mappings nous permet de créer un mapping en définissons l'ensemble des opérations ETL qui correspondent, définir pour chaque opération sa position dans le mapping et enfin renseigner la classe cible où se fera le chargement. Le module nous offre aussi la possibilité de consulter les différents composants d'un mapping déjà existant. Une fois que les mappings sont renseignés, le module offre une possibilité de simuler l'exécution d'un mapping. Dans ce processus l'ensemble des opérations ETL constituant le mappings sont exécutées une par une, le résultat final de la transformation des instances est affiché. Le concepteur choisit de charger ou non le résultat dans l'EDS.

Nous avons choisi de réaliser le diagramme de séquence de la phase de simulation de l'exécution d'un mapping.

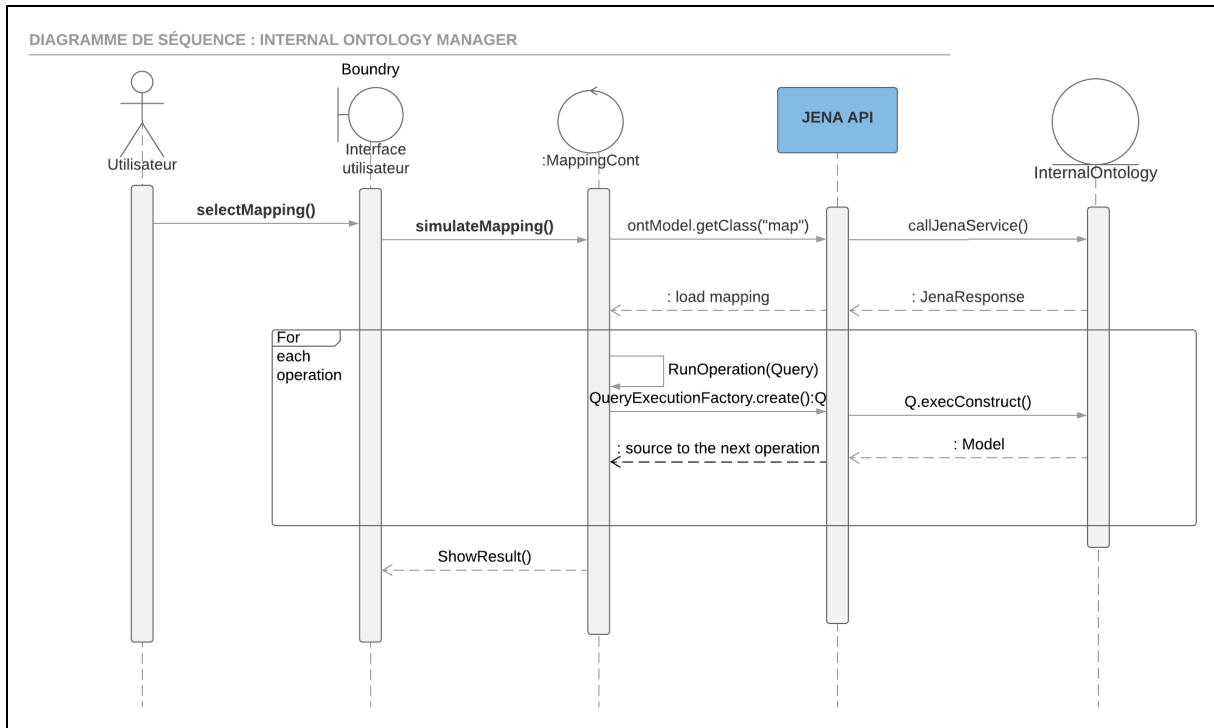


Figure 59 : Diagramme de séquence de la simulation de l'exécution d'un mapping sur les instances.

Dans la figure ci-dessous, nous avons le diagramme de classes qui représente les différentes interactions entre le contrôleur « MappingCont » et les différentes entités en relation avec les mappings et les différentes opérations ETL.

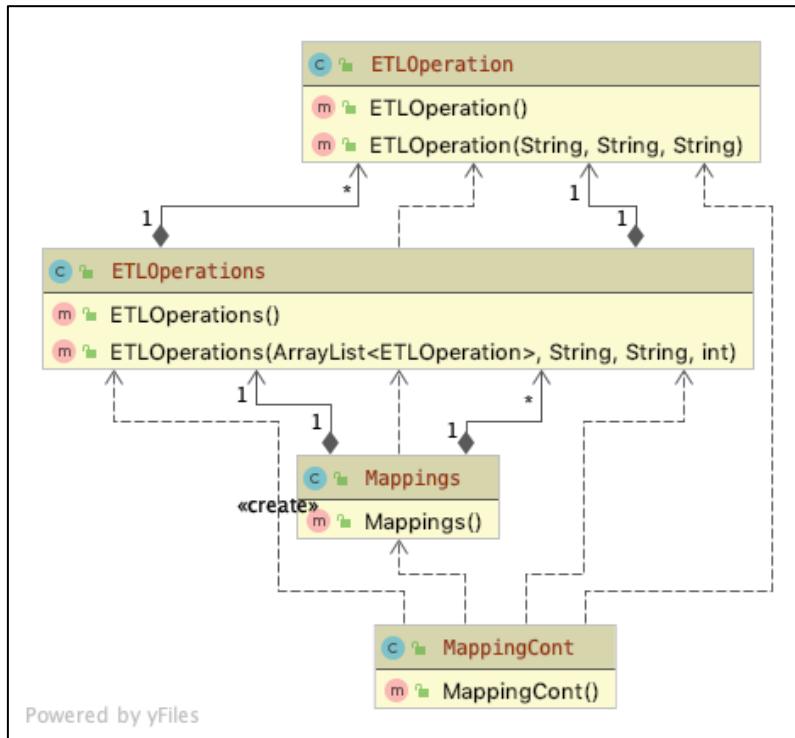


Figure 60 : Diagramme de classes montrant l'interactions entre les entités et le contrôleur du module de gestion des mappings.

4.3.3 Module de gestion des besoins

Ce module va assurer la gestion des besoins dont l'ajout et la consultation des besoins enregistrés. Il nous permet aussi d'affecter un nombre de mappings aux différents besoins pour indiquer le processus d'intégration à suivre pour peupler les différentes classes résultat et critères de chaque besoin. Le module nous permet aussi d'afficher les sources internes et externes ainsi que les instances ajoutées et inférées pour chaque besoin.

Nous allons présenter le diagramme de séquence de l'activité qu'on juge la plus importante dans le processus d'intégration. C'est celui en relation avec l'affectation des mappings aux différents besoins. Le diagramme de séquence schématisé le séquencement d'activités lors de l'exécution des contrôleurs comme suit :

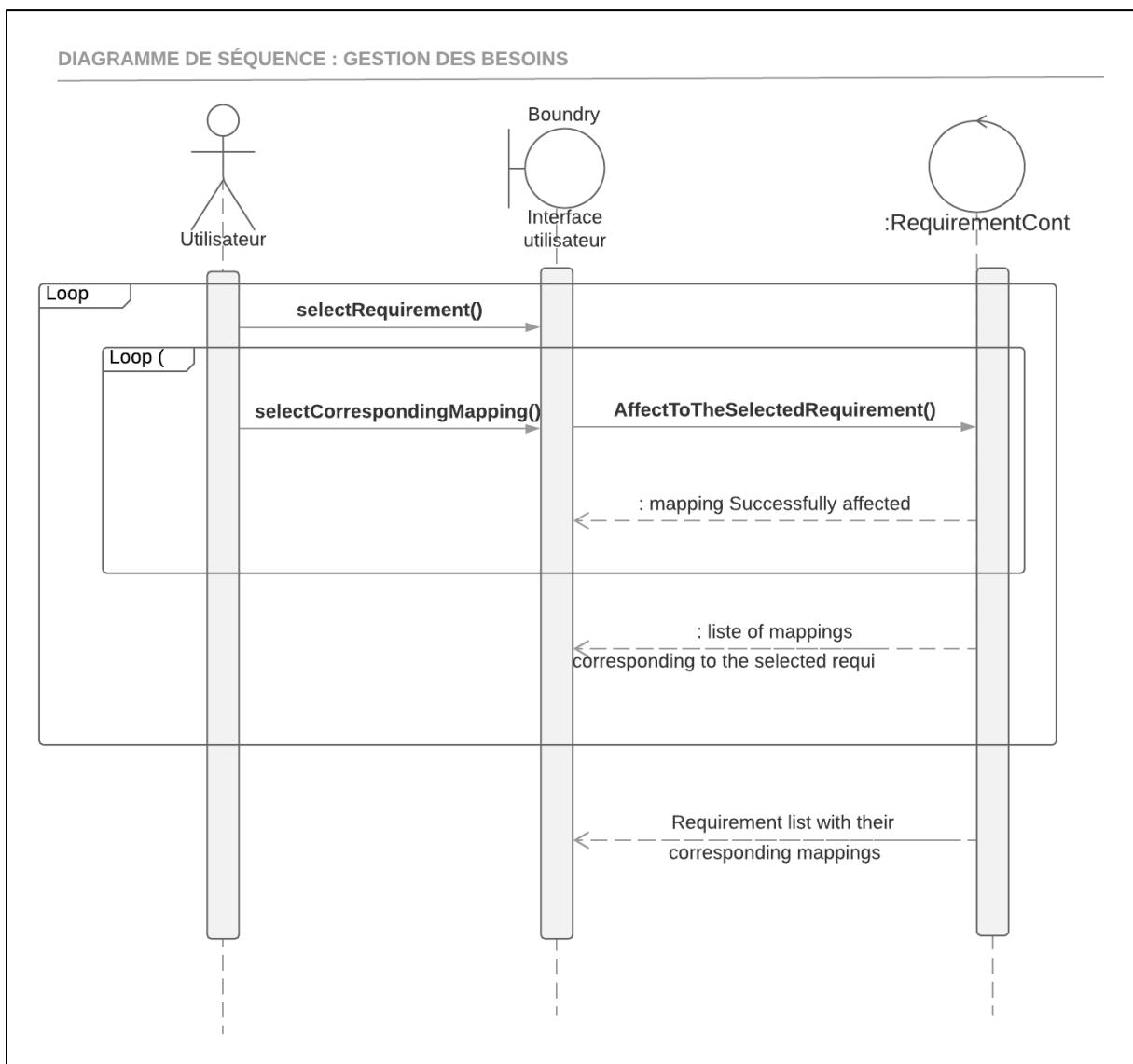


Figure 61 : Diagramme de séquence de l'affectation des mappings aux besoins.

Le modèle des besoins est lié fortement avec les mappings, pour cela nous avons conçus dans un seul package où on trouve quatre classes du modèle dont : « Requirements » qui représente le modèle des besoins définit précédemment, « ETLOperations » et « Mappings » qui représente les différents mappings spécifique à chaque besoin. Nous aurons ensuite la classe « RequirementCont » qui définit les fonctionnalités exécutées sur les besoins.

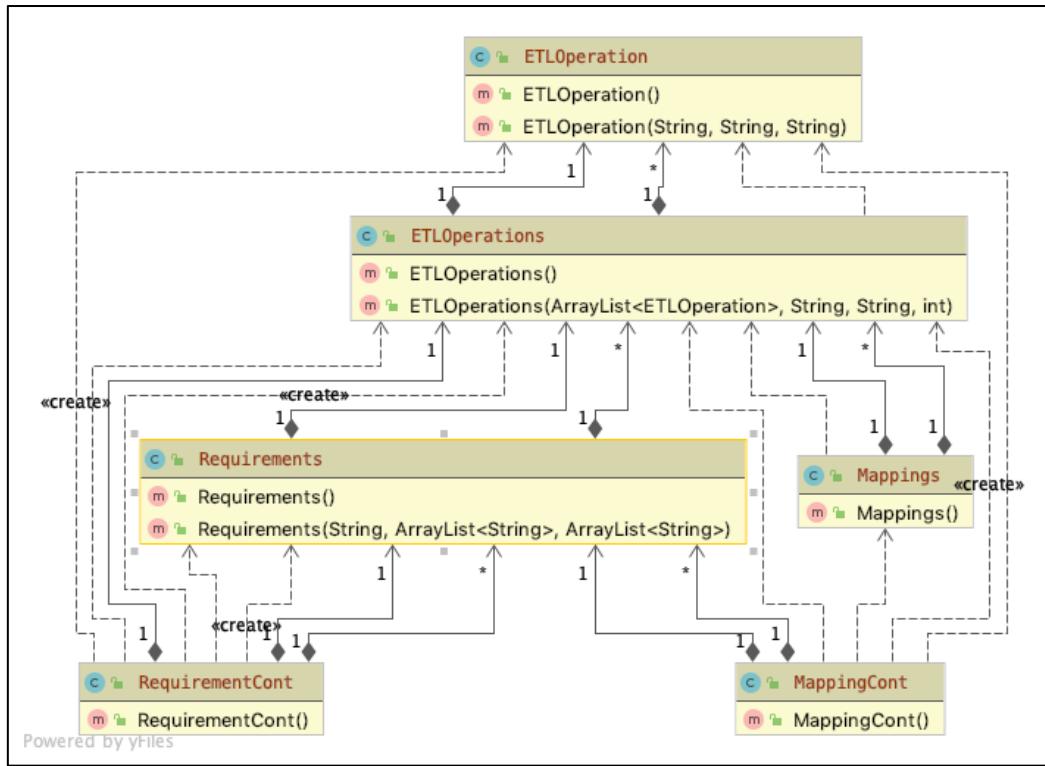


Figure 62 : Diagramme de classes représentant les interactions entre le module des besoins et celui des mappings

4.3.4 Module du processus d'intégration de données et de métadonnées

Ce module va assurer la gestion du processus d'intégration des données et des métadonnées. En se basant sur les mappings déjà définis par le concepteur, ce module permet d'effectuer l'extraction des données et des métadonnées à partir des sources, ensuite les unifier en supprimant les doublons au niveau des instances, et en détectant les chevauchements au niveau des métadonnées. Une fois toutes les transformations faites il permet de charger le résultat de l'intégration dans l'ontologie cible.

Pour mieux comprendre le fonctionnement du module d'intégration nous allons présenter en premier lieu le diagramme d'activité du processus, nous verrons ensuite le diagramme de classes qui schématisera la conception globale du modèle du processus d'intégration.

Partie 2 : Conception de la solution

Chapitre 3 : Conception

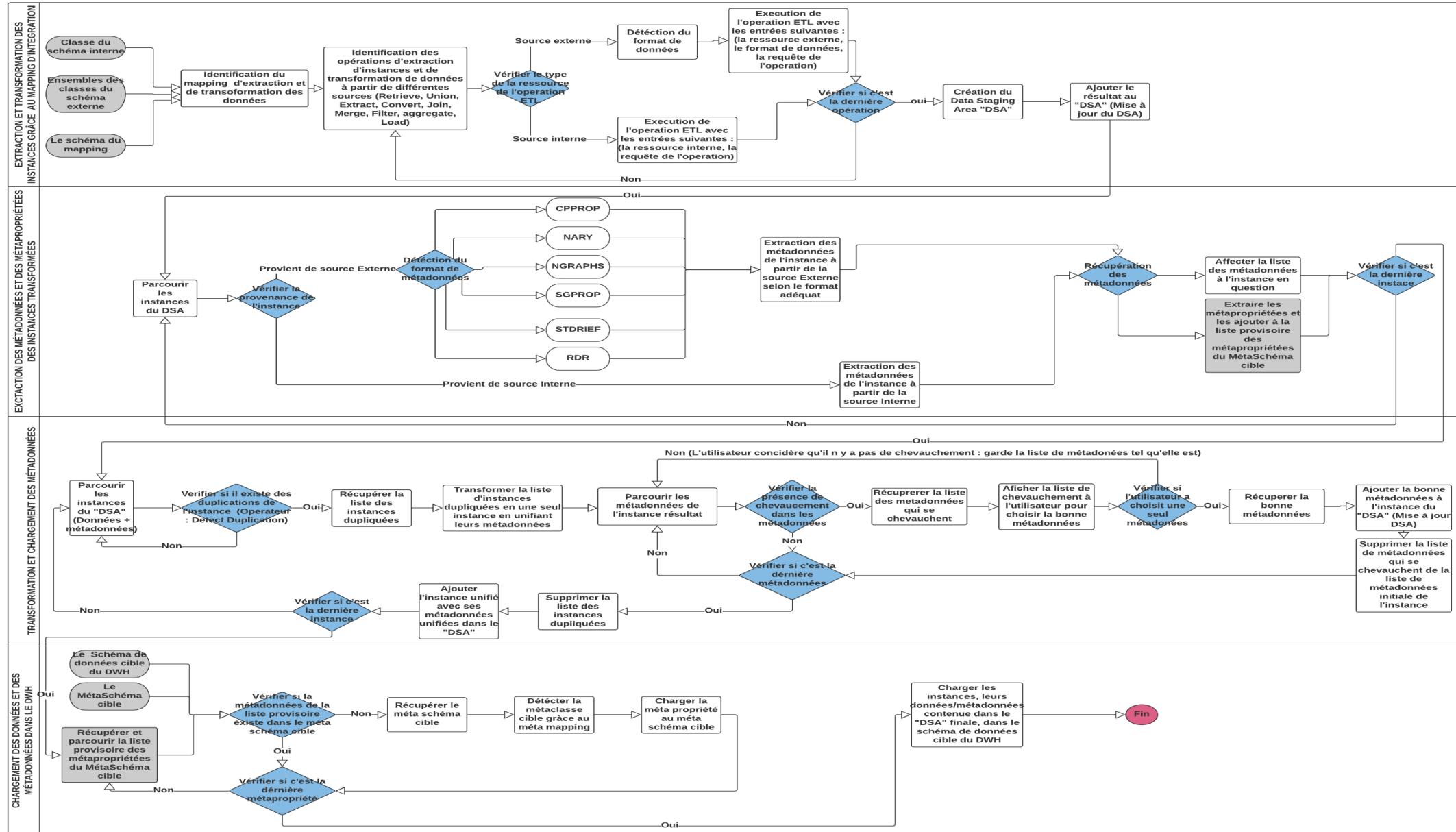


Figure 63 : Diagramme d'activité du processus d'intégration de données et de métadonnée

Le diagramme d'activité illustré dans la (Figure 63) représente les différentes phases du processus d'intégration des données et des métadonnées.

Dans le diagramme de classes suivant nous allons montrer les différentes classes du module du processus d'intégration.

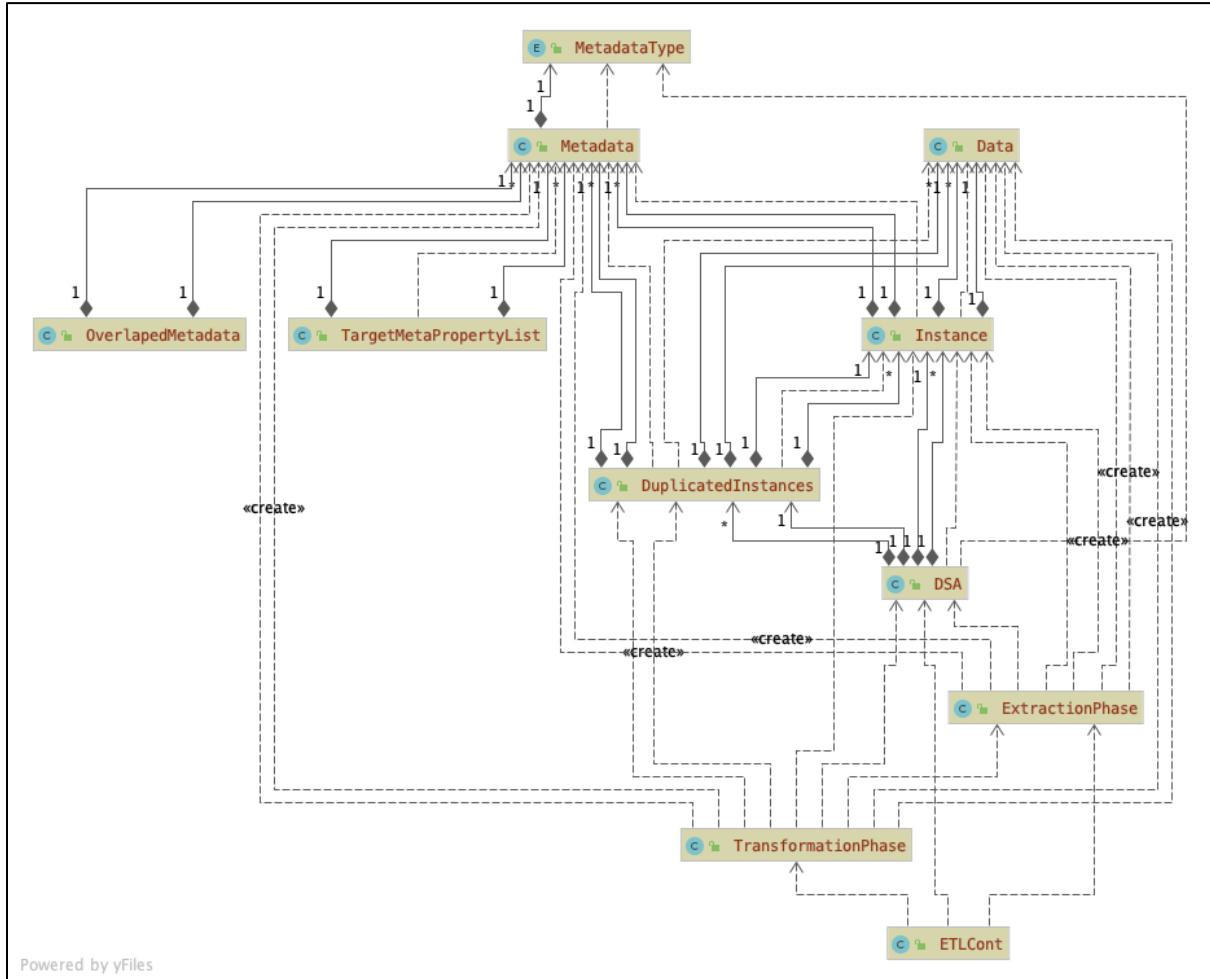


Figure 64 : Diagramme de classes représentant les différentes interactions entre les entités et le contrôleur du module de gestion du processus d'intégration.

- **La première phase :**

(Extraction et la transformation des instances) elle fait l'extraction des instances (données) à partir des sources internes et externes en lançant l'exécution d'un mapping et charger ensuite le résultat dans un Data Staging Area (DSA).

- **La deuxième phase :**

(Extraction des données et des métadonnées des instances transformées) durant cette phase nous parcourons l'ensemble des instances et pour chaque instance on extrait l'ensemble des métadonnées qui lui correspondent. Dans le cas où l'instance est externe, l'opération d'extraction va être différente selon le formalisme de la source externe. Comme c'est détaillé dans la (partie 1, chapitre 2, section 3.3.2) de notre rapport, le format de représentation des données et des métadonnées est différent d'un format à l'autre.

- La troisième phase :

(Transformation des métadonnées) elle représente la phase de transformation et d'unification des instances de leurs données et métadonnées. En premier lieu, il y a la détection des instances dupliquées. Ensuite les données et les métadonnées des instances dupliquées sont unifiées et affectées à une seule instance. Enfin dans les métadonnées unifiées, il y a le lancement de la détection de chevauchement, dans le cas où il y a présence d'un chevauchement, le concepteur choisit de garder ou de supprimer les données portant une contradiction. Les changements sont ensuite enregistrés dans le DSA.

Pour faire l'extraction des instances, leurs données et leurs métadonnées nous avons exploité la forme des différents Statements RDF pour les six formats. Le but principal c'est d'identifier les règles de traduction pour transformer les différents formats en format générique.

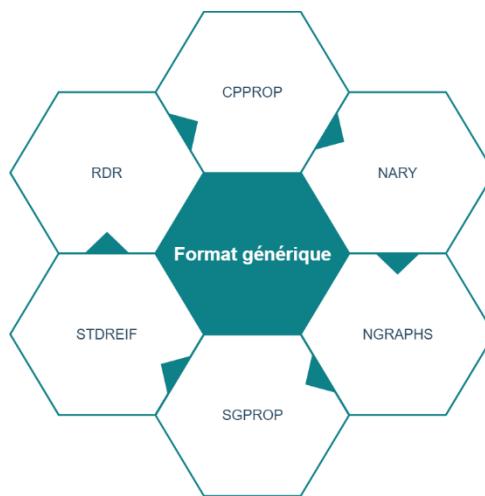


Figure 65 : Schéma représentant les six formalismes de métadonnées et le format générique qui devra les unifier

Pour l'ensemble des formats, il y'a un processus commun entre eux, qui est l'exploitation des *Statements RDF* (*Sujet, prédicat, objet*) pour l'identification des informations utiles. La démarche générale du processus d'extraction pour l'ensemble des formats est présentée comme suit :

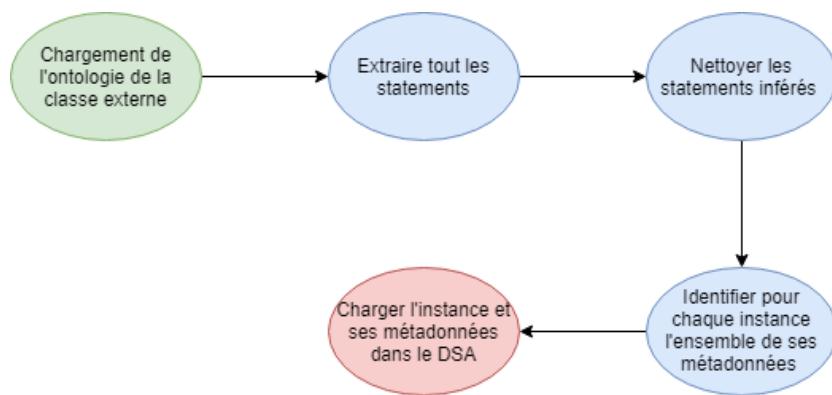


Figure 66 : Processus d'extraction générique des données et des métadonnées à partir des sources externes possédant des formats différents.

1. La première étape consiste à charger dans le système le contenu du fichier externe. Ce fichier peut être formalisé de différentes façons selon les six formats vus précédemment.
2. La deuxième étape consiste à parcourir tous les Statements RDF du fichier externe.

3. La troisième étape, consiste à supprimer les Statements RDF inférées qui ne contiennent pas des informations utiles ou à valeur ajoutée pour l'EDS.
4. Ensuite, selon la définition détaillée de chaque format présenté dans la (partie 1, chapitre 2, section 3.3.2) nous procérons à l'identification des instances, de leurs données et leurs métadonnées respectives.
5. La dernière étape consiste à charger les instances, avec leurs données et métadonnées dans le DSA.

Il faut noter qu'une fois toute instance avec ses données et métadonnées respectives est identifiées, extraites et chargées dans le DSA, nous considérons que l'unification des formats est faite. En effet, toutes les données sont représentées dans le DSA sous le même format qu'on nomme le *format générique* comme suit : *[DSA : [Liste d'instances : [Liste des données], [Listes des Métadonnées]]]* que nous allons stocker ensuite dans l'ontologie cible. Le diagramme de classes suivant représente les relations détaillées entre le DSA, les instances, les données et les métadonnées.

Vu les contraintes de temps et le compromis entre la qualité et la quantité, nous avons décidé de nous concentrer sur les deux formats externes les plus répondues qui sont : STD Reification et Nary.

Nous avons donc détaillé le processus d'intégration complet pour les deux formats sélectionnés grâce au diagramme d'activités suivant :

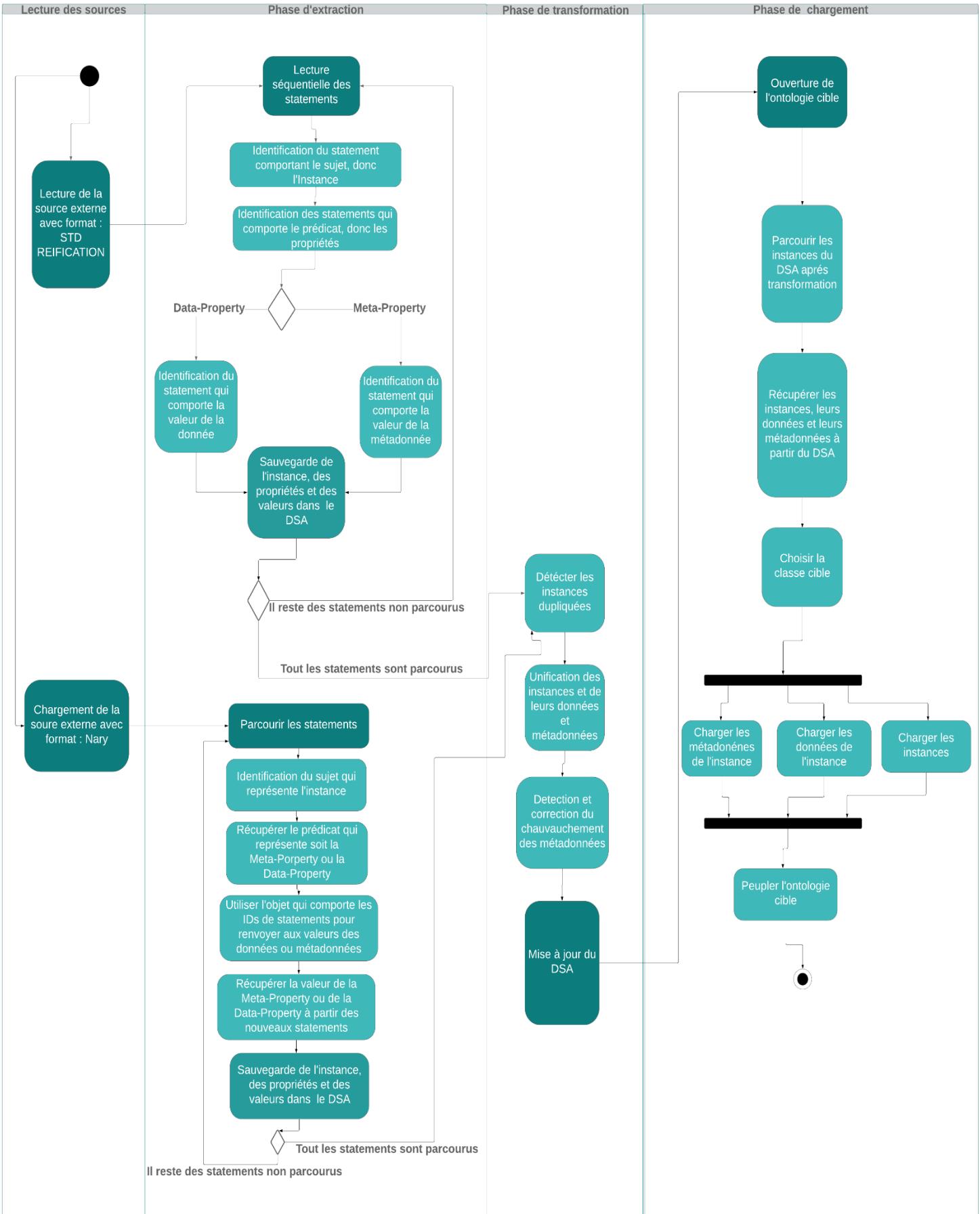


Figure 67 : Diagramme d'activités de l'unification des formats STD Reification et Nary

Le diagramme d'activité suivant est composé de quatre phases, la phase de lecture et identification des sources, la phase d'extraction, la phase de transformation et la phase de chargement. Le processus commence quand il y a appel pour une source externe. Dans ce cas de figure, la source externe peut être soit en format STD Reification ou Nary. Pour tous les formats existants la seule phase qui diffère c'est celle de l'extraction des instances, données et métadonnées, puis leur chargement dans le DSA. Tandis que la phase de transformation et chargement dans l'EDS reste la même. Sachant qu'une fois les instances sont détectées dans les deux fichiers, et une fois qu'elles sont chargées dans le DSA, on considère que leur format est unifié.

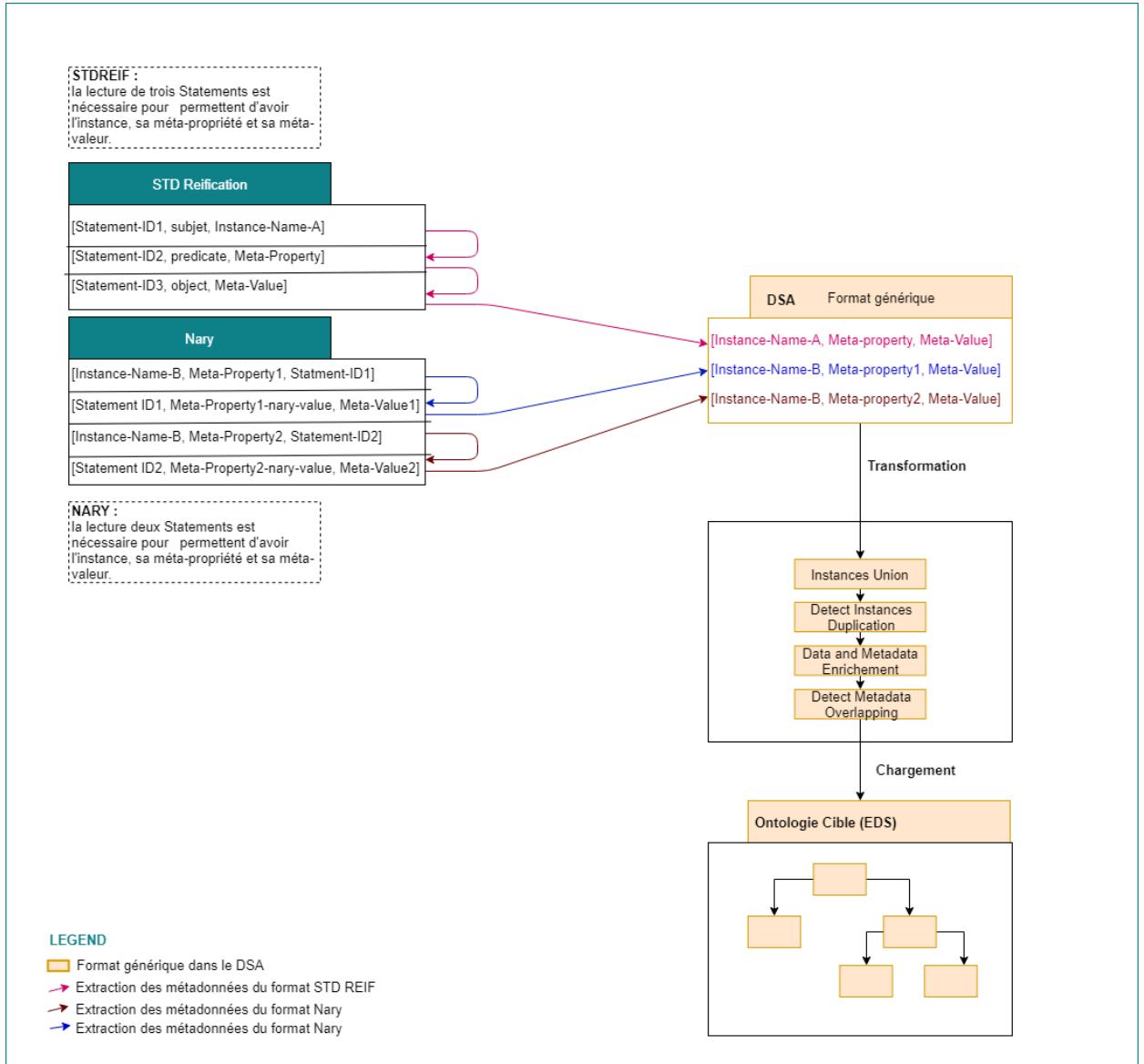


Figure 68 : Unification du format STDREIF et Nary sous le format générique

Concernant le format STDREIF comme représenté dans la (figure 68), il y a nécessité de lire trois Statements à la fois, afin d'avoir en premier lieu, le 1er Statement, qui comporte le sujet représente toujours le nom de l'instance. Ensuite, le 2^{ème} Statement, qui lui comporte le prédicat représente toujours le nom de la méta-propriété. Enfin, le 3^{ème} Statement qui comporte l'objet représente la valeur de la métadonnée.

Dans le diagramme d'activité le séquencement des étapes d'extraction est défini comme suit :

- 1- Lecture séquentielle des Statements (Trois à la fois).
- 2- Identifier le Statement qui comporte le sujet, donc l'instance.
- 3- Identifier le Statement qui comporte l'objet, donc les propriétés.
- 4- Une condition : la propriété elle est soit une Meta-Property ou une Data-Property.
- 5- Une fois le type de la propriété est détecté, le Statement suivant, va comporter l'objet: qui lui contient la valeur de la propriété (Meta ou Data).
- 6- Le processus sauvegarde l'instance, ses données et métadonnées dans le DSA.
- 7- Dans le cas où les Statements ne sont pas tous parcourus, on passe aux Statements suivants et on refait la détection.
- 8- Dans le cas où tous les Statements du fichier sont parcourus, donc toutes les instances sont chargées dans le DSA, on procède à la phase de transformation connue.

Concernant le format Nary, nous nous basons aussi sur les Statements. Cependant, les règles de traduction pour l'identification des instances, de leurs données et métadonnées diffèrent d'un format à un autre.

Comme représenté dans la (figure 68) la lecture des Statements pour le format Nary se fait différemment. Les étapes d'extraction sont représentées comme suit :

- 1- Lecture séquentielle des Statements.
 - 2- Dans le Statement sélectionné, identifier le nom de l'instance.
 - 3- Pour chaque Statement comportant le même nom d'instance comme sujet, on récupère le prédicat qui représente la Data-property ou la Meta-property et on récupère l'objet qui comporte l'ID du Statement qui renvoie vers la valeur de la propriété.
 - 4- Utiliser les IDs de Statement, pour pouvoir récupérer les valeurs des propriétés.
 - 5- Une fois le nom de l'instance, ses données et métadonnées sont détectées, elles sont toutes sauvegardées dans le DSA sous le format générique.
- 6- Dans le cas où tous les Statements sont parcourus, le processus passe à la phase de transformation, sinon il continue l'extraction.

La phase de transformation et chargement est la même pour tous les formats donc au final c'est la façon d'identifier les instances, leurs données et métadonnées respectives qui diffèrent d'un format à un autre. La représentation des formats en TTL est décrite dans la partie état de l'art dans la (partie 1, chapitre 2, section 3.3.2).

- **La quatrième phase :** Chargement des données et des métadonnées dans l'EDS

Dans cette étape le DSA est chargé dans la classe cible de l'EDS, et les méta-propriétés sont chargées dans le méta-schéma cible dans leur classe d'appartenance. On note que, tout comme pour les mappings entre les schémas de données, les mappings (sémantiquement parlant) entre les schémas de métadonnées sources et cible reviennent à l'effort du concepteur car cela demande une compréhension humaine et sémantique pour faire les méta-mappings.

4.3.5 Module de gestion du tableau de bord

Ce module permet de capturer et de visualiser sous forme de tableau de bord les valeurs des différentes distributions des instances, des ressources et des mappings du système.

Au fur et à mesure que le concepteur utilise l'outil, le tableau de bord enregistre l'évolution des indicateurs techniques comme le temps d'exécution et les indicateurs métiers liés à la gestion des besoins, des mappings et des données appropriées aux besoins du concepteur.

Une partie importante revient aux métriques d'intégration des métadonnées qui ont été ajoutées. Ces métriques sont décrites dans la (Partie 2 – Section 2.2.5).

4.4 Couche service

La couche service est constitué principalement de Jena API. Jena est un framework Java open source. Sa fonction principale comprend l'API RDF, le langage de requête RDQL, le sous-système de raisonnement, la mémoire de stockage persistant, le sous-système d'ontologie et fournit l'interface appropriée. Jena fournit une interface de base de données relationnelle dans les données RDF.

Jena enregistre le modèle d'ontologie à l'aide de deux types de tables. Le premier type correspond aux instructions affirmées et le second correspond aux instructions réifiées.

Jena prend en charge les modèles de mémoire et de base de données avec le package com.hp.hpl.jena.db. La persistance de la base de données suit les étapes suivantes :

- Chargez la classe de pilote JDBC de la base de données. Pour Nom (strDriver);
- Créer la connexion à la base de données IDBConnection conn = new DBConnection (strURL, strUser, strPassWord, strDB);
- Créer un ModelMaker pour la base de données ModelMaker maker = ModelFactory.createModelRDBMaker (conn);
- Créer un modèle par défaut «MyOntology» pour une ontologie;
- Effectuez la conversion des données d'ontologie stockées dans la base de données defModel.read (in, null).

Le schéma suivant représente l'architecture de Jena API.

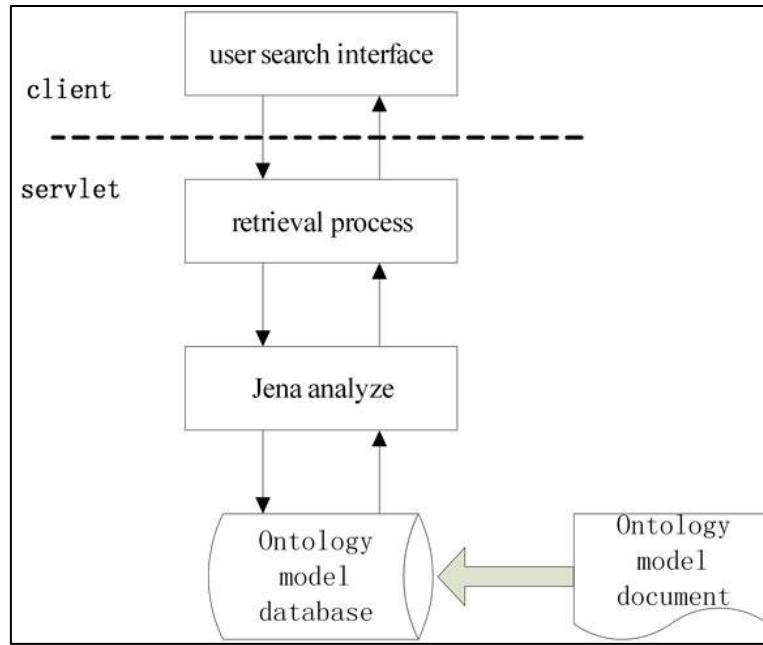


Figure 69 : Architecture du service Jena API. Source : http://file.scirp.org/Html/9-30305_21412.html

4.5 Diagramme de classes général

Les différentes couches du système sont en forte liaison suivant les principales techniques de l'architecture MVC. Dans le diagramme de classes suivant nous indiquons la liaison entre les contrôleurs « RequirementCont », « MappingCont » et « ETLCont » avec le modèle et les vues. Comme suit :

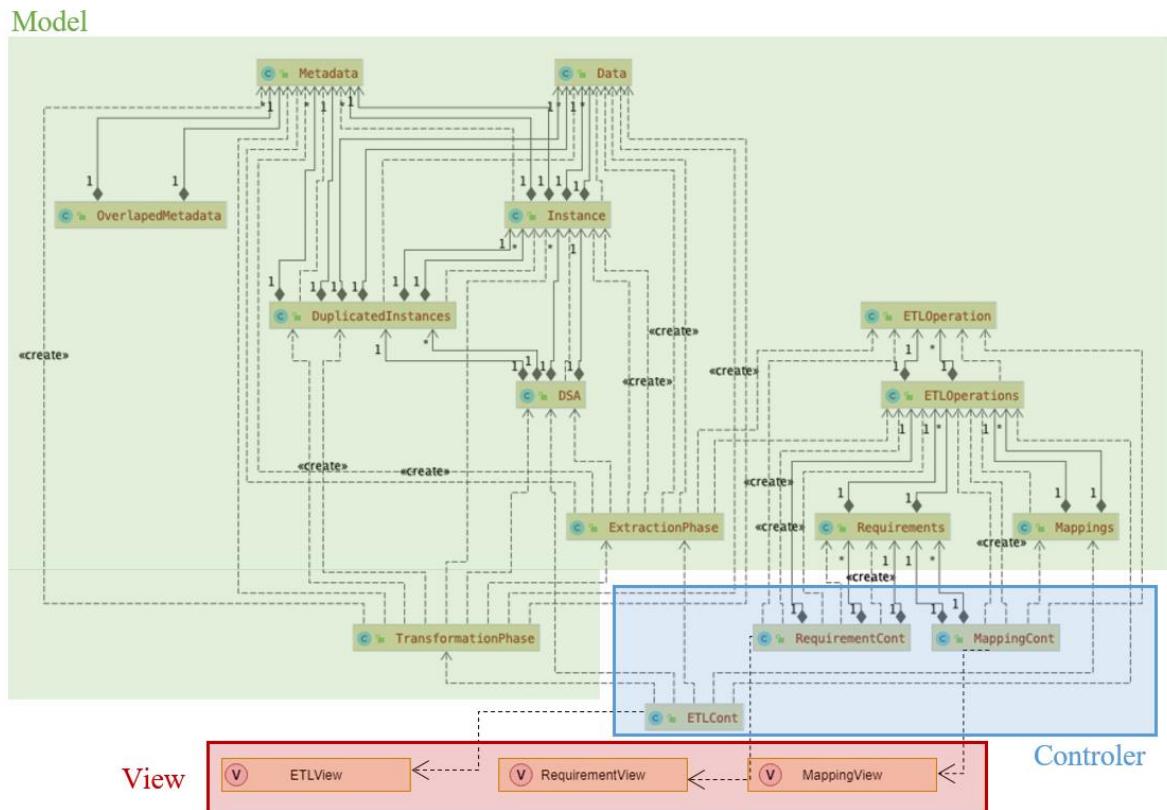


Figure 70 : Diagramme de classes représentant les différentes interactions entre les différentes couches du modèle MVC

Conclusion

Dans la partie de conception de la solution nous avons vu trois grands chapitres, le contexte et problématiques du projet, l'étude de l'existant et la conception de la solution. Dans le dernier chapitre nous avons fait une analyse des besoins et une conception générale des différentes parties de notre projet a été faite, en commençant par l'identification et l'analyse des besoins fonctionnels et non fonctionnels. Nous avons ensuite, présenté le fonctionnement général de notre solution, qui explique la liaison et la complémentarité entre les deux volets de notre projet, l'approche de conception orienté besoins et le processus d'intégration des données et des métadonnées. Ensuite pour aller plus dans le détail, nous avons présenté les quatre différentes couches de l'architecture de la solution, la couche client, la couche traitement, la couche service et la couche persistance. La phase de traitement est constituée des différents modules du système, l'un des modules les plus important est celui du processus d'intégration, un diagramme d'activité détaillé a été réalisé pour décrire le fonctionnement de l'algorithme en question. Comme cela, une étude détaillée d'une solution de création d'entrepôt de données sémantique orienté besoins a été réalisée grâce aux différentes étapes de la conception conçus à cet effet.

Une fois la conception faite, nous tenons une version prête à être implémentée, grâce à un ensemble d'outils que nous verrons dans la partie qui suit. Dans la partie suivante, qu'est la réalisation, nous verrons les outils qui nous permettront de réaliser notre système, ainsi que les tests de la solution afin d'assurer le bon fonctionnement de cette dernière.

Partie 3 : Réalisation

Dans la partie précédente, nous avons pris connaissance de l'architecture et de la conception du système d'entreposage de données sémantiques et de ses différents modules, services et données. Dans cette partie nous allons présenter dans le chapitre 1, l'environnement dans lequel notre système a été construit, en prenant connaissance des différents outils, langages et technologies utilisés. Ensuite nous verrons un aperçu détaillé de l'implémentation des principales fonctionnalités du système selon la conception vue précédemment. Nous verrons, enfin un aperçu des interfaces de l'application. Par la suite, dans le chapitre 2, nous verrons l'évaluation du système à l'aide des métriques de qualité décrit précédemment, et l'interprétation des résultats.

Chapitre 1 : Réalisation du système

Introduction

Ce chapitre va comporter les points les plus importants qui se rapportent à l'implémentation de notre projet. Pour pouvoir comprendre la réalisation que nous avons proposée nous allons tout d'abord énumérer les différents langages, outils et technologies utilisés dans ce projet. Ensuite nous allons mettre en évidence l'effort de réalisation fait pour chaque couche du système, commençant par la couche de persistance, puis la couche de traitement et enfin la couche client.

1 Langages, outils et autres technologies utilisées

Dans la partie des spécifications non fonctionnelles vue précédemment, nous avons cité quelques technologies utilisées pour l'implémentation du système. Dans ce chapitre nous allons aborder les différentes technologies et leurs utilités.

1.1 Langages et formats de données

RDF/XML : RDF / XML est une syntaxe, définie par le W3C, pour exprimer un graphe RDF sous la forme d'un document XML. RDF / XML est parfois appelé de manière trompeuse simplement RDF car il a été introduit dans les autres spécifications du W3C définissant RDF et était historiquement le premier format de sérialisation RDF standard du W3C. RDF/XML est donc le langage XML utilisé par RDF. Nous avons utilisé RDF/XML comme langage de format de données pour représenter les différentes ontologies internes et externes.



Turtle : En informatique, Turtle (Terse RDF Triple Language) est une syntaxe d'un langage qui permet une sérialisation non-XML des modèles RDF. C'est un sous-ensemble de la syntaxe Notation3. La spécification de Turtle est une Recommandation du W3C publiée le 25 février 2014.

Java : Java est un langage de programmation orienté objet. La particularité et l'objectif central de Java est que les logiciels écrits dans ce langage doivent être très facilement portables sur plusieurs systèmes d'exploitation tels que Unix, Windows, Mac OS ou GNU/Linux, avec peu ou pas de modifications, mais qui ont l'inconvénient d'être plus lourd à l'exécution (en mémoire et en temps processeur) à cause de sa machine virtuelle. Pour cela, divers plateformes et frameworks associés visent à guider, sinon garantir, cette portabilité des applications développées en Java.



FXML : FXML est un langage de balises XML pour la construction d'interfaces graphiques. Il offre une alternative pratique à la construction de telles interfaces en code de procédure, et est idéalement adapté à la définition de l'interface utilisateur d'une application JavaFX.



1.2 Outils

Eclipse : Eclipse IDE est un environnement de développement intégré libre (le terme Eclipse désigne également le projet correspondant, lancé par IBM) extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.



Visual Studio Code : Visual Studio Code est un éditeur de code open-source, gratuit et multi-plateforme (Windows, Mac et Linux), développé par Microsoft, à ne pas confondre avec Visual Studio, l'IDE propriétaire de Microsoft. VSC est développé avec Electron et exploite des fonctionnalités d'édition avancées du projet Monaco Editor. Principalement conçu pour le développement d'application avec JavaScript, TypeScript et Node.js, l'éditeur peut s'adapter à d'autres types de langages grâce à un système d'extension bien fourni.



Protégé : Protégé est un système auteur pour la création d'ontologies. Il a été créé à l'université Stanford et est très populaire dans le domaine du Web sémantique et au niveau de la recherche en informatique.



Glogg : C'est une application graphique multi-plateforme permettant de parcourir et de rechercher des fichiers journaux longs ou complexes. Il est conçu pour les programmeurs et les administrateurs système. glogg peut être vu comme une combinaison graphique et interactive de grep et de moins.

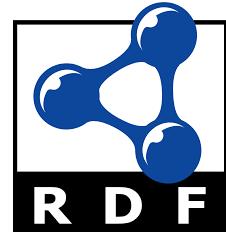


1.3 Framework et API

Jena API : Jena est un Framework Java que nous avons utilisée pour créer et manipuler des graphes RDF. Jena possède des classes pour représenter des graphes, des ressources, des propriétés et des littéraux. Les interfaces représentant les ressources, les propriétés et les littéraux.



RDF : Resource Description Framework (RDF) est un modèle de graphe destiné à décrire de façon formelle les ressources Web et leurs métadonnées, de façon à permettre le traitement automatique de telles descriptions. Développé par le W3C, RDF est le langage de base du Web sémantique. L'une des syntaxes (ou sérialisations) de ce langage est RDF/XML. D'autres syntaxes de RDF sont apparues ensuite, cherchant à rendre la lecture plus compréhensible.



JavaFX : JavaFX est une API de création d'interfaces graphique. Elle a été développée par Sun Microsystems, racheté par Oracle. JavaFX est qualifié comme le remplaçant de Swing. Cet article présente l'historique des versions puis les fonctionnalités et les particularités de JavaFX.



Apache Maven : Nous avons configuré l'environnement de développement sous « Apache Maven ». C'est un outil de gestion et d'automatisation de production des projets logiciels Java en général. Il est utilisé pour automatiser l'intégration continue lors d'un développement de logiciel. Il offre entre autres les fonctionnalités suivantes :

- Compilation et déploiement des applications Java (JAR, WAR)
- Gestion des librairies requises par l'application
- Exécution des tests unitaires
- Génération des documentations du projet (site web, pdf, Latex)
- Intégration dans différents IDE (Eclipse, JBulder)

EasyRDF : EasyRdf est une bibliothèque conçue pour faciliter la consommation et la production de RDF. Il a été conçu pour être utilisé par des équipes mixtes de développeurs RDF expérimentés et inexpérimentés. Il est écrit en PHP orienté objet.



Lors de l'analyse, EasyRdf crée un graphe d'objets PHP que l'on peut ensuite parcourir pour obtenir les données à placer sur la page. Les données sont généralement chargées dans un objet EasyRdf_Graph à partir de documents RDF sources. Il représente aussi un outil en ligne qui permet la conversion des fichiers RDF d'un format à un autre.

2 Implémentation du système

Dans cette partie nous allons voir les différentes procédures d'implémentation de notre système.

2.1 Couche persistance

Un grand travail d'implémentation a été fait au niveau de la couche persistance, nous avons construit nos sources de données et crée l'EDS comme suit :

2.1.1 Manipulation des sources

a. Source interne :

Notre source interne représente une ontologie analysant un système de e-commerce, elle contient les différentes classes en relation avec le thème, dont « Produit », « Producer », « Vendor » ...etc. L'ontologie interne est intégrée dans un modèle ontologique global dont on a étendu le méta-modèle pour représenter les besoins, les mappings et les différents concepts externes. Ces trois schémas nous permettent de garder traces des différentes données de traitement relatives à la gestion des besoins, des mappings et du processus d'intégration.

L'ontologie est à la base un fichier texte portant l'extension (.OWL), ce fichier est lisible avec l'éditeur d'ontologie « Portégé ». Nous avons exploité et fait plusieurs mises à jour sur le contenu de l'ontologie interne grâce à Protégé.

b. Sources externes :

Un grand effort a été fournis par rapport aux sources externes. Dans notre travail nous nous sommes basés sur un fragment de DBPEDIA.

Le but est de trouver les bonnes sources externes qui sont représentées sous différents formats. Le défi était de trouver une source LOD qui correspond à notre ontologie interne mais qui seraient aussi riche en métadonnées, et possède des formats de métadonnées hétérogènes.

Plusieurs recherches ont été faites à ce propos, mais c'était difficile de trouver une source qui remplit tous les critères. Mon maître de stage est intervenu en contactant l'auteur de l'article [J. Frey, 2017] sur lequel nous nous sommes basés pour déterminer les différents formats de représentation des métadonnées, afin de nous renseigner à propos des sources de données construites à l'aide de leur contribution. L'auteur de l'article [J. Frey, 2017] été réactif avec nous. Nous nous sommes donc reposés sur les sources de données décrites dans le tableau ci-dessous. Ces sources de données correspondent parfaitement à notre cas d'étude car elles comprennent des instances qui se rapportent aux produits et aux entreprises elles peuvent donc être intégrées dans notre benchmark e-commerce.

Tableau 2 : Tableau représentant les détails des sources externes

le source	Description	URL
Le site web de l'expérimentation	Devrait en réalité héberger toutes les données nécessaires.	http://vmpedia.informatik.uni-leipzig.de:8088/frey/meta-evaluation/
Fichiers : (sdw_***. Gz)	Les différents fichiers de modèle de réification des modèles de représentation de métadonnées - Metadata Representation Model (MRM)	http://vmpedia.informatik.uni-leipzig.de:8088/frey/sdw-dbpedia/paper/final-rdrshared/

Fichiers : revision-metadata	Les métadonnées qui sont partagées entre tous les MRM différents est dans le lien suivant.	http://vmdbpedia.informatik.uni-leipzig.de:8088/frey/sdw-dbpedia/paper/final-rdrshared/revision-metadata/
------------------------------	--	---

Nous nous sommes donc basés sur les sources de LOD décrites précédemment pour faire notre travail. Le site web¹⁴ de Johannes Frey, a décrit avec les six modèles de représentation de métadonnées voir (Partie 1, Chapitre 2, Section 3.3.2) les deux LOD les plus populaires, qui sont DBPEDIA et WIKIDATA.

Dans notre travail nous nous sommes penchés sur DBPEDIA comme source de données sémantiques. Pour un premier lieu, et pour un premier prototype dans l'intégration des différentes formes ou modèles de représentation de données, nous nous sommes contentés dans l'implémentation de traiter deux formats qui sont : Nary et Stdreif. Vous trouverez ci-dessous les caractéristiques de nos sources de données :

MRM	Nom du fichier	Taille
STDREIF	sdw_en_compact-stdreif.nq.gz	982615748 triplets
NARY	sdw_en_compact-nary.nq.gz	914385102 triplets

- **Difficultés rencontrées dans l'extraction des sources externes :**

Après avoir eu des difficultés de trouver le bon dataset convenable à notre étude, nous avons été confronté à deux autres problèmes :

1. ***Le volume énorme des fichiers sources :***

La taille des fichiers est très grande, et atteint les 982615748 triplets pour le formalisme STDREIF. L'ouverture et l'exploitation de ces fichiers volumineux ne peut pas se faire ni en *Protégé* ni avec un éditeur de texte quelconque, dans notre cas c'est *Visual Studio Code*. Pour remédier à ce problème nous avons utilisé l'outil *Glogg*. A la base c'est un outil qui permet de parcourir d'importants fichiers logs afin d'y effectuer des recherches. Nous allons l'utiliser beaucoup plus pour sa capacité de supporter la lecture de fichier de volume très important, et effectuer des recherches dessus.

Pour l'extraction des fragments de données qui nous intéressent, nous avons effectué des recherches des mots clés dont « Company », « Product » et « Supplier ». Ces mots clés se rapportent à notre ontologie interne autour de l'E-commerce. Nous avons donc sélectionné des fragments de données correspondant à chaque classe.

2. ***Le format des fichiers :***

Le deuxième problème rencontré c'est le format des fichiers téléchargés. Le format des deux sources est en NQUAD, l'inconvénient c'est que ce format rend difficile la lecture des triplets et complique la compréhension des différents formalismes de métadonnées qui est très important pour notre travail.

Pour cela, nous avons dû utiliser un convertisseur de formats RDF qui est *EasyRDF* pour pouvoir transformer le format du fichier premièrement en *TTL*, pour pouvoir comprendre les différents formalise de métadonnées et ensuite en *RDF/XML* pour pouvoir l'exploiter dans notre outil, grâce au service *JENA API*.

¹⁴ <http://vmdbpedia.informatik.uni-leipzig.de:8088/frey/meta-evaluation/>

- **Les caractéristiques finales de nos fichiers de source externe après transformation et adaptation :**

Après avoir remédié aux différentes difficultés rencontrées nous avons eu en résultats les deux fichiers avec les caractéristiques suivantes :

Formalisme (MRM)	Format du fichier	Poids
STDREIF	TTL	1519 Ko
NARY	TTL	1026 Ko

2.1.2 Création du méta-schéma cible

Notre méta-schéma cible est conçu en suivant la partie *système* du modèle (SM4AM) décrit dans la (Partie 2, Chapitre 3, Section 4.2.3).

Afin de créer le méta-schéma cible nous devons faire des correspondances entre chaque méta-propriété présente dans les sources internes ou externes avec la méta-classe qui lui correspond dans le modèle système de (SM4AM).

Un effort considérable a été fournis dans ce côté. En effet, nous avons parcouru les méta-propriétés disponibles dans les deux sources internes et externes et nous avons effectué une classification des méta-propriétés selon leur appartenance. Pour cela nous avons défini des méta-mappings entre la méta-propriété, sa provenance (Interne, Externe) et sa méta-classe cible.

Dans les tableaux suivant nous allons résumer les affectations :

- **Méta-propriétés de la source interne :**

Tableau 3 : Méta-propriétés de la source interne avec les méta-classes cibles correspondantes

Méta-propriétés	Préfixes	Normes	Méta-classe cible
dc:date	dc :http://purl.org/dc/elements/1.1/	Dublin Core	DataSource
dc:publisher			VocabularyTerm
dc:title			
Owl:priorVersion	owl:		Statistical
Owl:versionInfo	http://www.w3.org/2002/07/owl#		
Rev:reviewer	rev : http://purl.org/stuff/rev#		VocabularyTerm

- **Méta-propriétés des sources externes :**

Tableau 4 : Méta-propriétés des sources externes et leurs méta-classes cibles correspondantes

Méta-propriétés	Préfixes	Normes	Méta-classe cible
dc:created dc:modified dc11:description dc11:subject	dc: http://purl.org/dc/element/1.1/ dc11:http://purl.org/dc/elements/1.1/	Dublin Core	DataSource
prov:wasderivedfrom	prov: http://www.w3.org/ns/prov#	Prov-O	DataSource
ns0:uniquecontributornb ns0:revperyear2016	ns0: http://ns.inria.fr/dbpediafr/voc#		Statistical

ns0:revperlastmonth1			
ns0:hasmainrevision			
ns0:hasoldrevision			
ns1:isversion	ns1:http://www.w3.org/2004/03/trix/swp-2/		Statistical
ns2:publisher	ns2: http://dbpedia.org/ontology/		VocabularyTerm
Quel que soit la métapropriété			AceabilityEvidence (DataSource et Operation)

Pour effectuer le chargement dans l'EDS de l'ensemble des méta-propriétés détectés lors du processus d'intégration, nous faisons passer la liste des méta-propriétés par une méthode qui se charge de faire le mapping entre une méta-propriété et sa méta-classe cible comme représenté dans les tableaux ci-dessus. La définition sémantique des méta-mappings revient à l'effort du concepteur.

2.2 Couche traitement

Pour la réalisation de la partie traitement nous avons réalisé une application desktop en développant des fonctionnalités java en suivant le modèle MVC.

Nous avons créé un contrôleur pour chaque module, ce contrôleur assure les interactions entre le modèle que nous avons décrit dans la partie conception voir (Partie 2, Chapitre 3, Section 4.2.1) et l'interface graphique des utilisateurs. Chaque module de traitement est renseigné par son contrôleur, ils sont décrits comme suit :

2.2.1 Module de gestion des sources

Dans la partie du chargement des sources dans l'application, nous avons utilisé principalement des appels de service JENA. La méthode « *RDFDataMgr.LoadModel()* » pour le chargement de l'ontologie interne. En ce qui concerne les classes externes nous avons utilisé « *ModelFactory.createDefaultModel()* ».

2.2.2 Module de gestion des besoins

Ce module contient plusieurs volets. Le 1^{er} volet permet de charger et consulter les besoins existants, l'utilité de celui-ci est de permettre au concepteur de garder la traçabilité de son travail et le consulter selon le besoin. Pour réaliser ce volet nous avons fait des appels JENA pour le chargement des détails des besoins. Le 2^{eme} volet permet de faire une affectation des mappings correspondant à chaque besoin, pour garder trace des différentes opérations ETL à exécuter ensuite dans le module d'intégration. Les volets restants concernent le calcul et l'affichage de certaines statistiques calculées lors des différents traitements opérés sur la gestion des besoins.

2.2.3 Module de gestion des mappings

Ce module contient principalement trois volets, la consultation, la mise à jour ou l'ajout des mappings et enfin, la simulation des transformations. Les mappings sont enregistrés dans le schéma des mappings qui se trouve dans le modèle interne comme vue précédemment. Lors de la simulation, les différentes opérations ETL sont ordonnées selon leur position, puis chaque opération est exécutée sur

sa ressource de données. Nous avons pour chaque opérateur ETL une fonction Run() spécifique à celui-ci. Les opérateurs ETL sont traduits en requêtes Sparql, donc avant d'exécuter un opérateur sa requête est générée selon les entrées de celui-ci. Par exemple pour l'opérateur *Retrieve* nous avons la requête suivante :

```
Select ?Instances# Where {?Instances# rdf:type Namespace:Class}
```

2.2.4 Module du processus d'intégration

Ce module comporte trois phases principales, l'extraction, la transformation et enfin le chargement.

- a) **Extraction** : Lors de la phase d'extraction, les sources internes et externes sont extraites selon les mappings définis précédemment. Il y'a donc une extraction des instances, de leurs données et métadonnées selon les mappings prédefinis. Ses derniers, aident à détecter quelles sources utiliser et le chemin d'accès pour les récupérer. Les instances d'une même source peuvent être unifiées, jointes ou agrégées. Une fois les sources détectées, les instances des sources internes et externes sont extraites séparément. Une visualisation du résultat de l'extraction est faite afin de permettre au concepteur de vérifier la cohérence des données lors de la phase d'extraction.
- b) **Transformation** : Lors de la phase de transformation, la première étape c'est d'unifier les sources internes avec l'ensemble des sources externes dans une zone temporelle unique de stockage de données « DSA ». La deuxième étape consiste à identifier dans le DSA les instances dupliquées. La troisième étape consiste à créer une instance unifiée de l'ensemble des instances dupliquées, il s'agit d'enrichir l'instance grâce à l'unification de leurs données et de leurs métadonnées selon trois scénarios possibles. Ces scénarios concernent l'unification des instances dupliquées qui possèdent des données et des métadonnées soit complémentaires, contradictoires ou répétée. Une fois les données et les métadonnées unifiées pour chaque instance, la quatrième étape permet de détecter le chevauchement entre les métadonnées. Il existe deux scénarios possibles, existence ou absence de chevauchement. La cinquième étape consiste à sauvegarder les opérations d'unification des instances ainsi que leurs données et métadonnées dans le « DSA ».
- c) **Changement** : Lors de la phase de chargement, il y a deux étapes principales. La première consiste à charger le modèle de données « DSA » dans une des classes cible de l'EDS. La deuxième consiste à charger les méta-propriétés dans le méta-modèle cible de l'EDS. Le chargement des méta-propriétés dans la bonne méta-classes cibles revient à l'effort du concepteur dans la définition d'un certain nombre de méta-mappings (voir section 2.1.3) entre la méta-propriété et la méta-classe qui lui correspond.

2.2.5 Module de gestion du tableau de bord

Ce module permet de capturer les valeurs de l'ensemble des métriques et de les visualiser grâce à des graphes statistiques sous forme d'un tableau de bord. Les différentes métriques sont calculées lors de l'exécution des fonctionnalités du système. On cite principalement, les métriques évaluant le système de gestion des besoins et l'affectation des mappings et les métriques en relation avec l'exécution du processus ETL. Pour cet effet, nous avons ajouté des variables qui contiennent les valeurs des différentes métriques dans les classes « Requirements », « Mappings », « DSA », « ExtractionPhase »,

« TransformationPhase » et « LoadingPhase ». Les valeurs des métriques sont capturées à la demande du concepteur, puis affichées dans des graphes dans le volet « Dashbord » de l’application.

2.3 Couche Client

Nous allons présenter dans cette partie la version finale de l’implémentation de notre système. Les résultats seront présentés à travers des captures d’écran des différentes interfaces de l’outil, ainsi que les résultats d’utilisation de celui-ci.

Nous avons proposé une conception de l’outil respectant les différents critères ergonomiques dont l’utilisabilité, l’adaptabilité, la flexibilité, l’utilité et enfin l’intuitivité et la facilitation de l’utilisation.

Afin de nous concentrer dans la réalisation du processus d’intégration au complet, nous avons choisi de réaliser et d’implémenter les fonctionnalités les plus pertinentes pour le bon fonctionnement de notre système. Nous n’avons pas réalisé les différentes fonctionnalités optionnelles comme les mises à jour des besoins, et des mappings.

2.3.1 Interface de la gestion des sources

La figure ci-dessous représente l’interface de gestion de gestion des sources. Elle permet au concepteur de charger dans le système l’ontologie globale et les différentes sources internes et externes à intégrer.

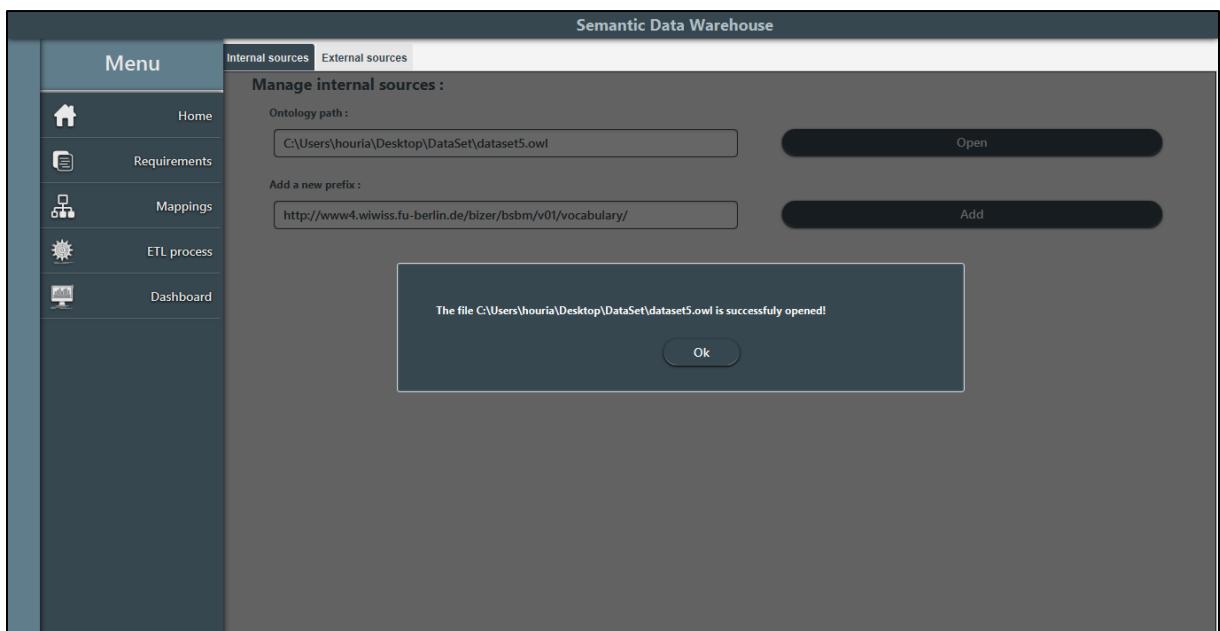


Figure 71 : Interface d’importation de l’ontologie globale

2.3.2 Interface de la gestion des mappings

La figure ci-dessous représente l’interface de gestion de smappings. Elle permet principalement de consulter et d’analyser les mappings existants, et de simuler le résultat des transformations des instances, pour les valider avant de les affecter aux besoins qui leurs correspondent.

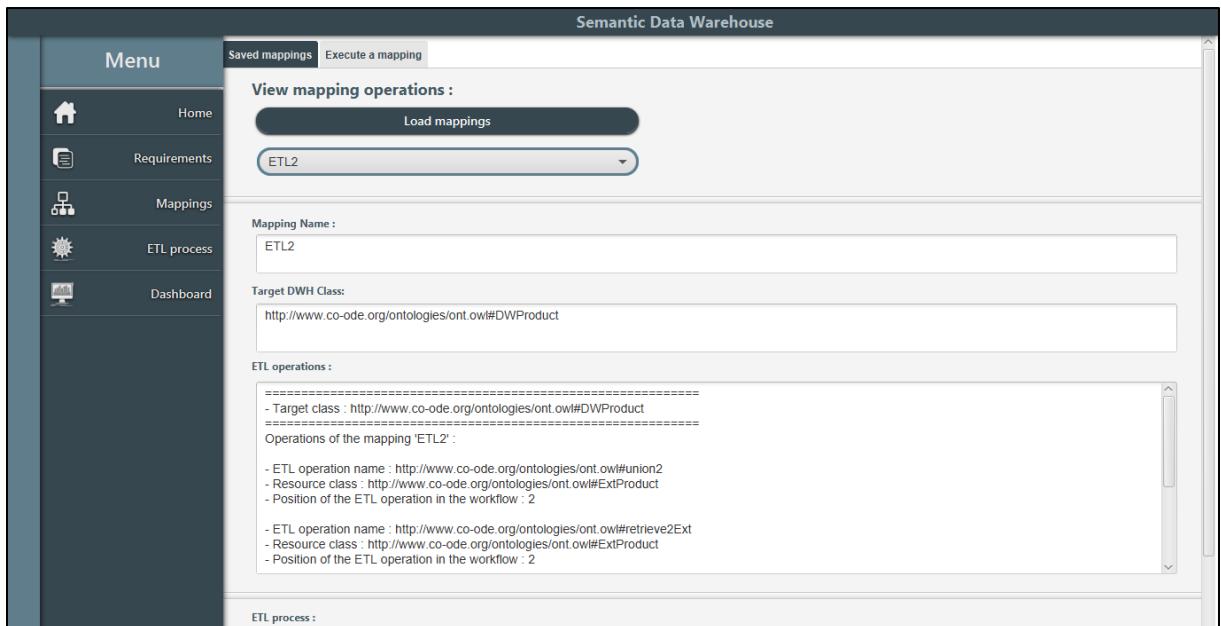


Figure 72 : Interface de gestion des mappings

2.3.3 Interface de la gestion des besoins

La figure ci-dessous représente l'interface de gestion des besoins. Elle permet de visualiser les différents besoins existants dans l'ontologie globale. Mais elle permet aussi de faire l'affectation des différents mappings aux besoins, afin de peupler l'EDS. De plus, cette interface permet de visualiser les différentes ressources internes et externes mais aussi les instances inférées et insérées utilisées pour chaque besoin.

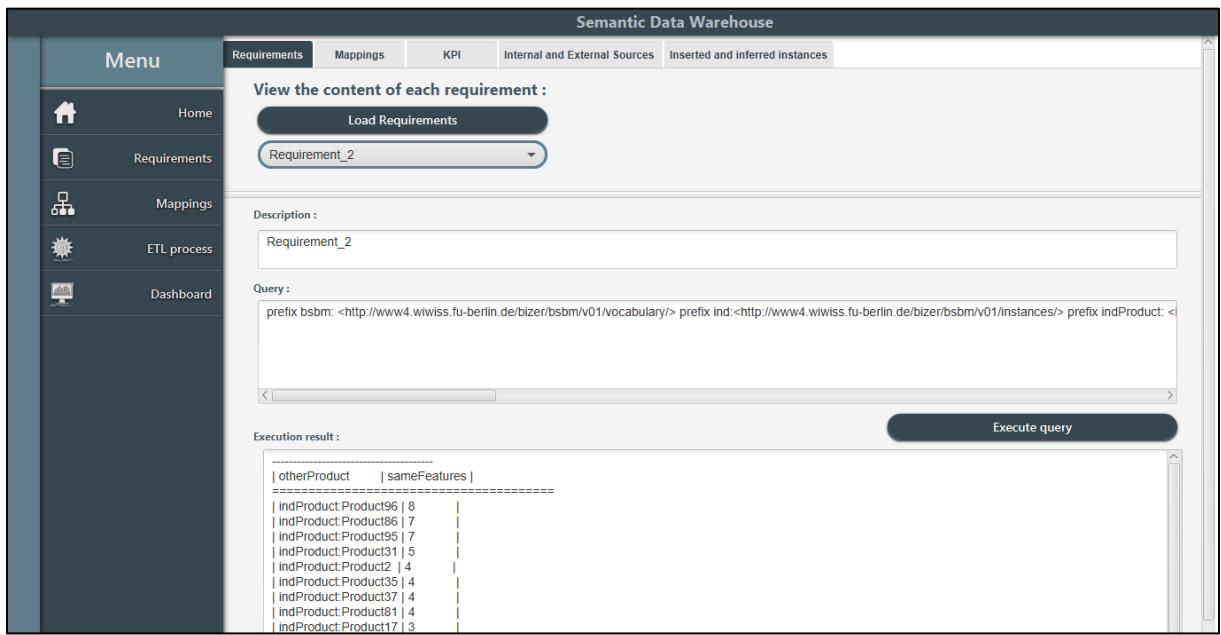


Figure 73 : interface de gestion des besoins

2.3.4 Interface de la gestion du processus ETL

Les figures ci-dessous représentent les différentes interfaces qui permettent l'exploitation des différentes fonctionnalités du processus d'intégration des données et des métadonnées. Trois phases chronologiques et principales se manifestent qui sont l'extraction, la transformation et le chargement.

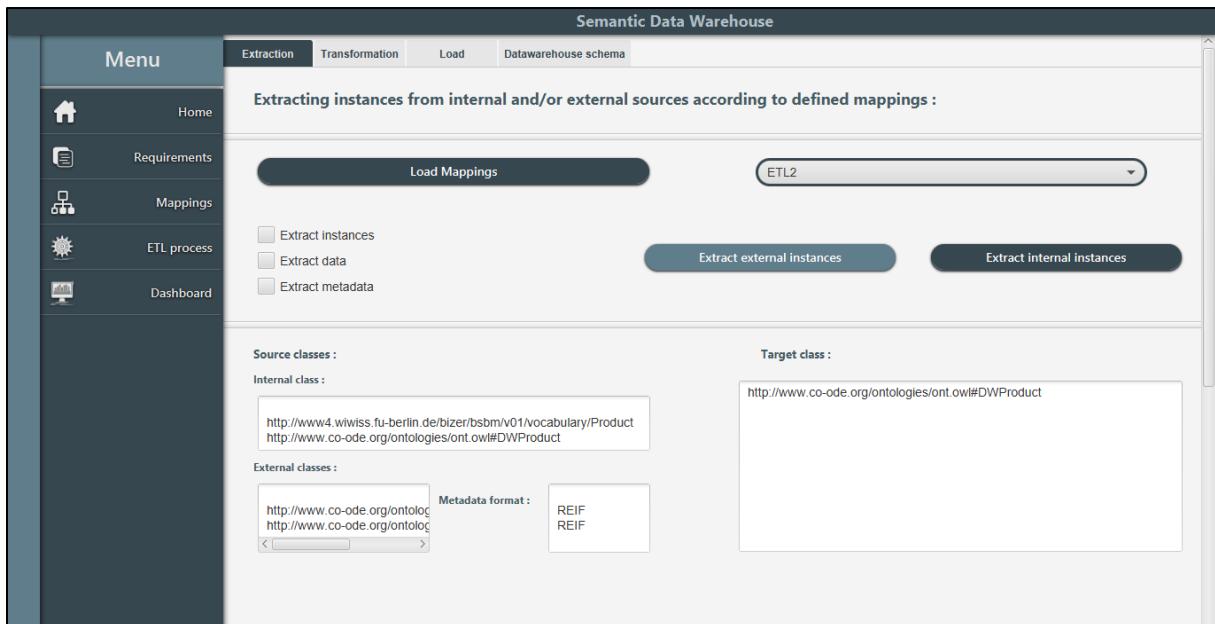


Figure 74 : Interface d'extraction des sources

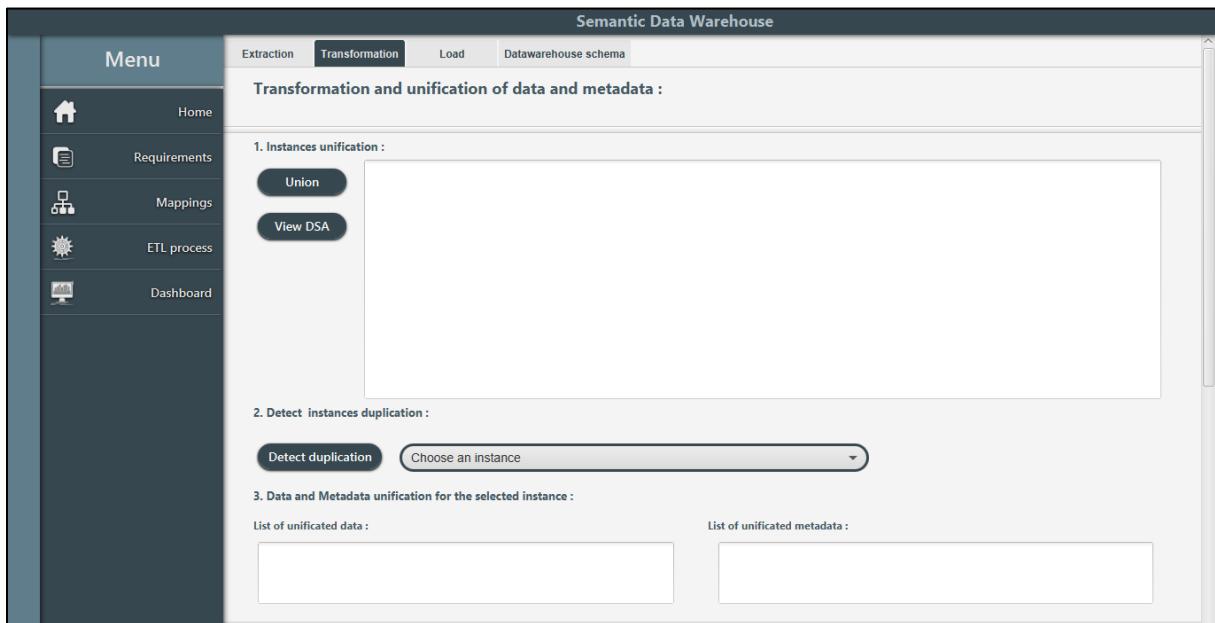


Figure 75 : Interface de transformation des données et des métadonnées

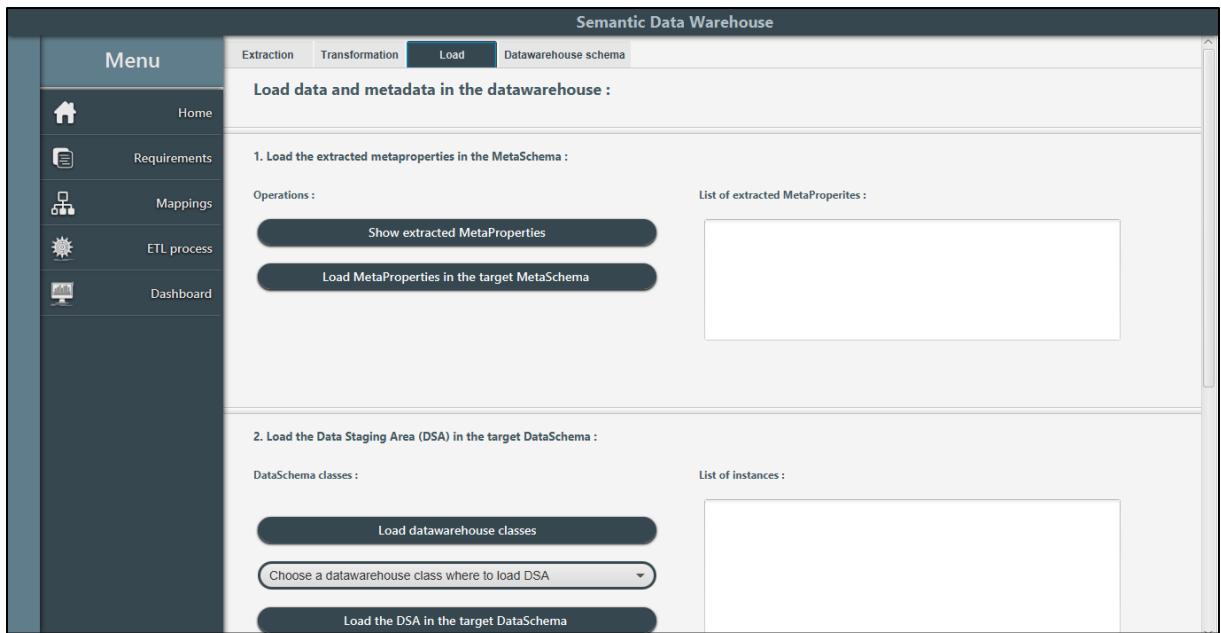


Figure 76 : Interface de chargement dans le schéma cible de l'EDS

2.3.5 Interface du tableau de bord

La figure ci-dessous représente les différents graphes statistiques qui donnent au concepteur une vision globale des métriques techniques concernant les besoins, les mappings et le processus d'intégration.



Figure 77 : Interface du tableau de bord

Conclusion

Nous avons abordé dans ce présent chapitre les grandes lignes en rapport avec la réalisation de notre projet. En premier lieu, nous avons abordé les différents outils, langages et technologies qui nous ont permis de réaliser le système. En deuxième lieu, nous avons présenté de façon détaillée l'implémentation de chaque couche de notre système, en commençant par l'implémentation de la couche persistance qui s'occupe de la gestion des données sources et des données cibles. Ensuite, la couche de traitement, qui elle décrit les différents modules réalisés. Enfin, la couche client, qui elle se concentre sur la présentation de l'outil réalisé et les résultats de son fonctionnement.

Chapitre 2 : Expérimentation et évaluation du système

Introduction

Afin de fournir des informations sur la qualité de notre système, nous avons réalisé une série de tests qui comporte l'évaluation du processus d'exécution de la solution proposée. Nous vous exposerons dans ce qui suit le scénario de test dans en premier lieu et ensuite la description des statistiques et métriques de qualité relatives aux besoins et mappings conjointement et aussi du processus d'intégration. Les résultats de ces métriques obtenues lors de l'exécution de la solution sont aussi présentés dans ce même point.

1 Démarche et scénario de test

Le test d'un système représente une étape importante dans un projet informatique. Elle nous permet dans notre cas d'évaluer les fonctionnalités mises à disposition du concepteur. Tout processus de test suit une certaine démarche qui permet de mener à bien l'étape d'évaluation du système.

Le scénario de test se passe comme suit :

- Entrer le chemin vers l'ontologie globale et cliquer sur ouvrir. L'ontologie globale va s'importer dans le système. Celle-ci contient les différentes sauvegardes des sources, des besoins, des mappings et du processus d'intégration.
- Définir un ensemble de mappings ETL et simuler l'exécution de chacun d'eux sur les instances afin de consulter la pertinence du résultat par rapport au système.
- Consulter les besoins sauvegardés et affecter les mappings ETL qui correspondent à chaque besoin afin de peupler l'EDS de façon stratégique.
- Une fois l'affectation et la vérification des mappings ETL sont faites, le concepteur passe à l'étape d'intégration des données et des métadonnées.
- Choisir le mapping à exécuter, vérifier les caractéristiques du mapping et lancer l'extraction des sources internes et externes.
- Passer à la phase de transformation, en unifiant en premier lieu les instances internes et externes. Ensuite, procéder à la détection des instances dupliquées et à l'enrichissement des instances. Cet enrichissement se fait grâce à l'unification des différentes données et métadonnées des instances dupliquées.
- Pour une instance donnée, procéder à la détection des chevauchements de ses métadonnées. Le concepteur choisit ensuite, soit de sélectionner la bonne métadonnée et supprimer les contradictions, soit de tout garder.
- Une fois les opérations de transformations sont faites, cliquer sur sauvegarder pour enregistrer les modifications faites sur le DSA.
- Refaire l'opération autant de fois qu'il y a d'instances dupliquées.
- Passer à la phase de chargement, en générant en premier lieu la liste des métapropriétés et procéder à leur chargement dans le métamodèle cible de l'EDS. Le chargement des métapropriétés dans le métamodèle cible suit un ensemble de métamappings qui permettent de faire la correspondance entre la métapropriété et la métaclass qui lui correspond.
- Passer ensuite au chargement de l'ensemble des résultats dans le l'EDS. Pour ce faire, le résultat de transformation des instances, des données et des métadonnées qui se trouvent dans le « DSA » sont chargées dans la classe cible qui lui correspond.
- Exécuter le processus d'intégration pour tous les mappings restants, selon les besoins des utilisateurs.

- Une fois le processus exécuté, le concepteur consulte le tableau de bord pour vérifier les différentes variations techniques enregistrées lors de l'exécution. Ces variations aident le concepteur à prendre connaissance du comportement du processus d'intégration, et détecter les anomalies techniques afin de mieux paramétriser le processus d'exécution, les sources, les mappinsng et les besoins.

Le respect de ce scénario permet d'exploiter le système de façon efficace et réussit.

2 Statistiques et métriques d'évaluation

Afin d'évaluer les performances de l'outil en général et du processus d'intégration en particulier, nous avons défini un ensemble de métriques, qui permettent de bien évaluer les différents critères de notre système et de son environnement. Nous avons deux types de métriques : les métriques de description et les métriques de qualité.

2.1 Volume global des sources internes et externes

Nous allons présenter dans cette section le volume global des sources de données que nous avons utilisées durant l'exécution du processus d'intégration.

Tableau 5 : Tableau représentant le volume des sources internes et externes en instances, données et métadonnées

Type des sources	Nombre d'instances	Nombre de données	Nombre de métadonnées
Internes	100	4085	500
Externes	1969	578	1394
Total	2069	4663	1894

2.2 Métriques de description

Les métriques de description permettent de décrire le contenu et le comportement de l'ontologie globale et des sources de données (Internes et Externes). On distingue trois différentes métriques de description :

2.2.1 Ressources internes et externes

Cette métrique permet de calculer le nombre de ressources internes et externes utilisées par tous les mappings ETL qui correspondent à un chaque besoin. Elle permet d'avoir une vision interne du type de ressources existantes, extraites des portails du LOD (Ex : dc:creator, foaf:person ...etc) ou internes au benchmark (Ex : ont:reviewer, ont:product ... etc).

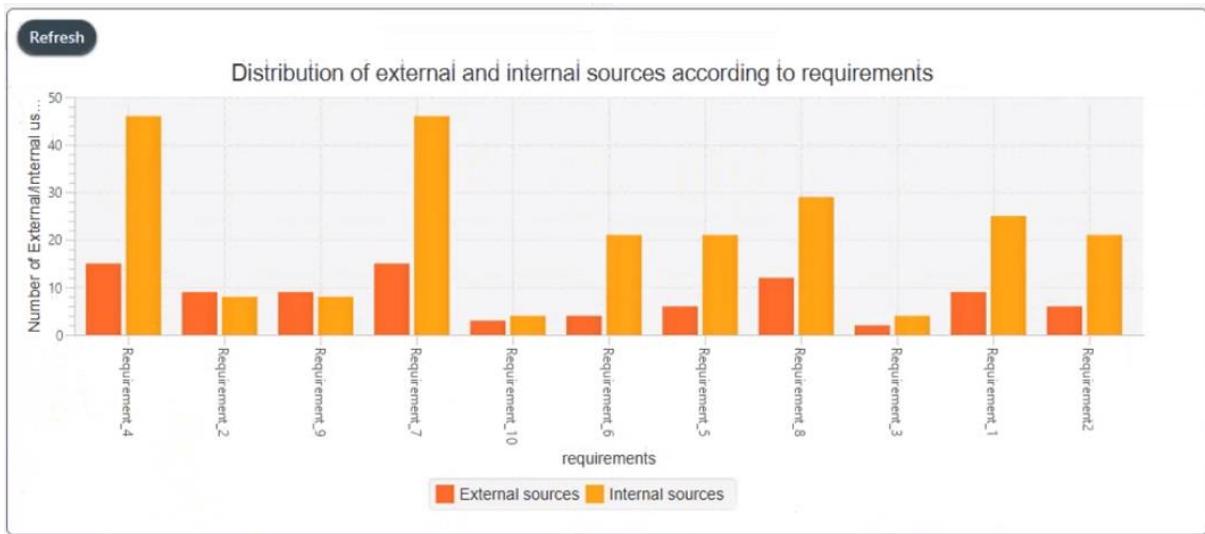


Figure 78 : La distribution des ressources internes et externes par rapport aux besoins

2.2.2 Instances inférées et insérées

Cette métrique permet de calculer le nombre d'instances insérées et inférées par le raisonnableur de l'ontologie. Elle permet d'avoir une vision globale du type d'instances dans l'ontologie globale.



Figure 79 : La distribution des données insérées et inférées par rapport aux besoins

2.2.3 Description des Métadonnées

Il y a principalement 4 métriques pour mesurer le taux de présence des différents types de métadonnées présentes dans les sources.

- M1: le pourcentage des métadonnées relatives à la date
- M2: pourcentage des métadonnées relatives à la provenance
- M3: pourcentage des métadonnées relatives à la version
- M4: pourcentage des métadonnées relatives à la description
- M5: pourcentage des métadonnées relatives aux statistiques

Tableau 6 : Tableau représentant les résultats des métriques de description des métadonnées

Métriques	Nombre	Pourcentage	Phase
Nombre totale des métadonnées	1894	100%	Extraction
M1	244	12.90 %	Extraction
M2	305	16.10%	Extraction
M3	139	7.40%	Extraction
M4	381	20.20%	Extraction
M5	678	35.80%	Extraction

2.3 Métriques de qualité

Le système doit permettre de calculer certaines métriques de qualité lors de l'exécution du processus d'intégration des métadonnées, dont :

2.3.1 Cohérence des données:

La mesure dans laquelle chaque utilisateur voit de manière cohérente les données et où l'intégrité des données est maintenue tout au long des transactions et des sources de données.

- M6: Pourcentage des instances dupliquées.
- M7: Pourcentage des métadonnées avec chevauchement
- M8: Pourcentage de contradiction en métadonnées (il y a chevauchement, soit une similitude soit une contradiction des valeurs).

Tableau 7 : Tableau représentant les résultats des métriques de qualité

Métriques	Nombre	Pourcentage	Phase
Nombre total d'instances	2069	100%	Extraction
Nombre total de métadonnées	1894	100%	Extraction
M6	713	34.50%	Transformation
M7	345	18.23%	Transformation
M8	25	1.32%	Transformation

Les résultats des différentes métriques lors du test de notre système prouvent une amélioration dans la fiabilité et la cohérence des données du schéma source dans le schéma cible. Cela revient à l'efficacité du processus d'intégration des métadonnées que nous avons proposé. L'épuration des données se fait grâce aux différentes opérations d'unification aux deux niveaux : sémantique et schéma.

2.3.2 Performance :

La performance du processus ETL tel qu'il est mis en œuvre sur un système, par rapport à la quantité de ressources utilisée et à la rapidité du service fourni.

- M9: La mémoire consommée par le processus ETL.
- M10: Temps d'exécution du processus ETL.

Tableau 8 : Métriques représentant les résultats des métriques de performances

Métriques	Phase d'extraction	Phase de transformation	Phase de chargement	Totale des phases
M9 (Mégabyte)	316,12	35689,26	454,89	36 460,28
M10 (Millisecondes)	121778	161664	109857	393299
M10 (Secondes)	121,77	161,66	109,85	393,29
M10 (Minutes)	2,02	2,69	1,83	6,55

Pour la masse de données utilisée nous estimons que les résultats de performances du processus d'intégration sont bons. Nous remarquant que la phase de transformation prend plus de temps d'exécution, nous avons utilisé des algorithmes de tries optimisés pour effectuer les différentes recherches et unifications sur les données et les métadonnées. Cependant, une optimisation de ce temps est nécessaire car le but ultime d'un ETL est le traitement d'une plus grande masse de données tout en diminuant le temps d'exécution.

La première préoccupation de notre projet était d'assurer la gestion et le fonctionnement de l'ETL mais comme perspectives, il faudrait réduire le temps d'exécution grâce à un ensemble de techniques comme: le traitement par lots où nous utilisons des opérations basées sur des ensembles de données, évitez les boucles imbriquées, réduire les données en exécutant les opérateurs « Union », « Join », « Merge » le plus tôt possible dans le processus d'intégration, utiliser des techniques de traitement parallèle, comme les threads et les outils de traitement des grandes masses de données comme l'opération « *MapReduce* ».

2.3.3 Audibilité :

C'est la capacité du processus ETL à fournir une transparence des données, cela inclut: la traçabilité: possibilité de suivre l'historique des étapes d'exécution du processus ETL et la qualité des informations documentées sur l'exécution.

- M11: Nombre des nodes impliqués pour chaque besoin.

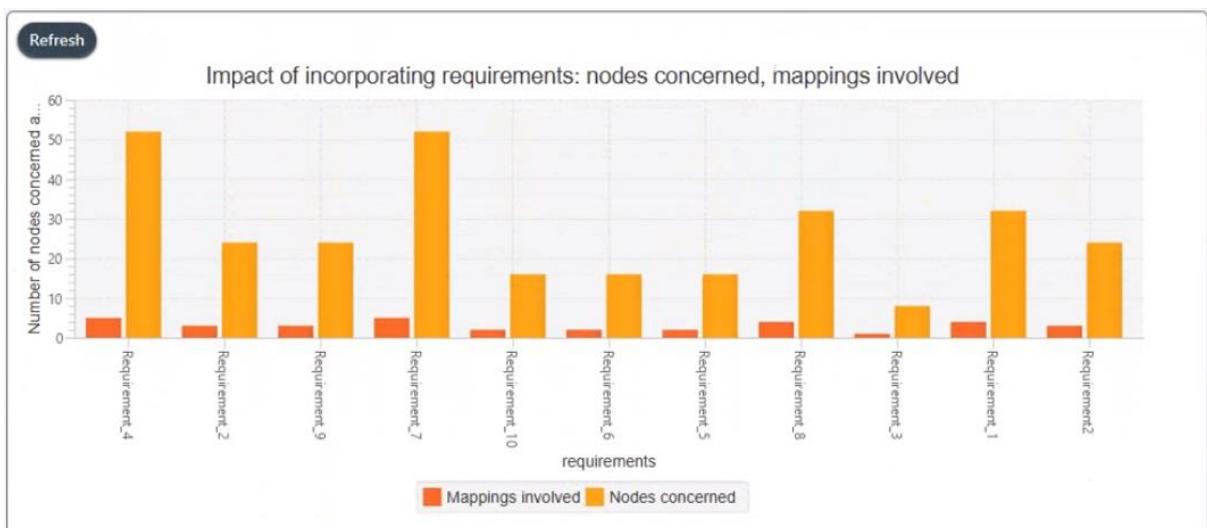


Figure 80 : Nombre de concepts et propriétés impliquées dans les mappings de chaque besoin

Conclusion

Nous avons vu dans ce chapitre deux points essentiels dans l'évaluation et test du système. D'abord nous avons décrit le scénario de test en détails afin de tester le fonctionnement de la solution proposée. Ensuite nous avons présenté certaines métriques que nous avons classifiés en deux types : les métriques de description et les métriques de qualité.

Conclusion

Dans la partie Réalisation, nous avons pris connaissance de la solution réalisée plus en détails, avec notamment une vue réelle sur le produit via la démonstration de ce dernier. Nous avons vu deux grands chapitres, l'implémentation de la solution et l'évaluation de celle-ci. Dans le 1^{er} chapitre nous avons donné pour les trois couches du système, les problèmes rencontrés et les solutions proposées pour y remédier. Nous avons également découvert les outils et technologies qui nous ont servi afin de réaliser ce projet. Dans le 2^{ème} chapitre, nous avons présenté les différentes métriques d'évaluations de la qualité du système d'intégration des métadonnées. Notons que la réalisation n'est en aucun cas la dernière partie du projet. En effet conformément aux bonnes pratiques, nous nous sommes penchés sur les tests du produit. Le résultat de ses dernières était positif ce qui est très satisfaisant au regard des attentes et des objectifs soulignées.

Partie 4 : Gestion de projet

Afin d'atteindre les objectifs ultimes de notre projet et pour réaliser le système dans de bonnes conditions tout en respectant les délais et les ressources, nous avons appliqué une démarche de gestion de projets rigoureuse et bien étudiée.

Dans cette dernière partie de notre mémoire nous allons présenter les différents aspects de gestion de projets auxquels nous avons eu recours. Tout d'abord nous allons commencer par présenter l'entité principale qui nous a orienté dans notre travail. Nous allons ensuite présenter l'évolution de notre projet du début de celui-ci jusqu'à sa fin, grâce à l'exposition des différents prototypes réalisés. En dernier lieu nous allons présenter le diagramme de gant de notre projet.

1 Encadrement du projet

Pour mener à bien tout projet informatique, il est nécessaire d'avoir une entité pour orienter le travail afin de mener à bien la conception et la réalisation du projet. Dans le cadre de tout projet de fin d'études au sein de l'Ecole Supérieur d'Informatique, il y a une nécessité d'être encadré par un enseignant de l'école pour s'assurer que le projet est conforme en consistance et en contenu et qu'il soit conforme aux qualités d'un travail d'un ingénieur de l'école. Dans notre cas, nous avons été principalement encadré par Dr Selma Khouri. Cet encadrement a donné résultat à de nombreuses réunions régulières, le plus souvent des entrevues hebdomadaires et parfois journalières afin de discuter de notre état d'avancement, des problèmes rencontrés et des différents points futurs à entamer. Mais aussi le rôle de Mme Khouri est de valider les principales lignes directrices du projet, et de s'assurer que le travail est en accord avec les conditions d'un bon projet de fin d'études, mais aussi valider le mémoire et les prototypes à plusieurs itérations.

2 Présentation des prototypes

Depuis le début du travail nous avons réalisé plusieurs prototypes que nous avons amélioré à chaque itération pour enfin arriver à la version finale de notre projet. Dans ce qui suit nous allons présenter l'historique des prototypes réalisés :

- **Prototype 1 :** Installation et reconfiguration de l'ancien outil existant, et restauration des différentes fonctionnalités réutilisable sous forme d'un nouveau projet Maven. Intégration et paramétrage des différents fichiers de configuration pour faire fonctionner le nouvel outil correctement et insertion de Jena API.
- **Prototype 2 :** Mise en place d'une architecture MVC qui va contenir toutes les vues, contrôleurs et modèles de bases comme squelette de l'application. Ce prototype contient des interfaces minimalistes, chacune connectée à un contrôleur qui utilise les différentes classes de base (Besoins, Mappings, Operations ETL).
- **Prototype 3 :** Réalisation de l'importation et la gestion des sources internes et de l'ontologie globale et implémentation des fonctionnalités relatives à la gestion des mappings et des besoins.
- **Prototype 4 :** Intégration des métriques relatives à la gestion des besoins et des mappings et réalisation des différents schémas statistiques de distribution des données dans le module de tableau de bord.
- **Prototype 5 :** Amélioration des interfaces graphiques et ajout d'une charte de couleurs. Amélioration de l'aspect visuel de l'application.
- **Prototype 6 :** Réalisation du processus ETL, de l'extraction, à la transformation et au chargement. Le 6^{ème} prototype représente le dernier prototype de notre système il a été suivie par des tests et des vérifications et corrections des erreurs.

3 Planning du projet

Notre projet a été réalisé dans le cadre d'un stage de fin d'études à l'Ecole Supérieur d'Informatique au sein du laboratoire LCSI du 01 octobre 2018 au 01 juillet 2019.

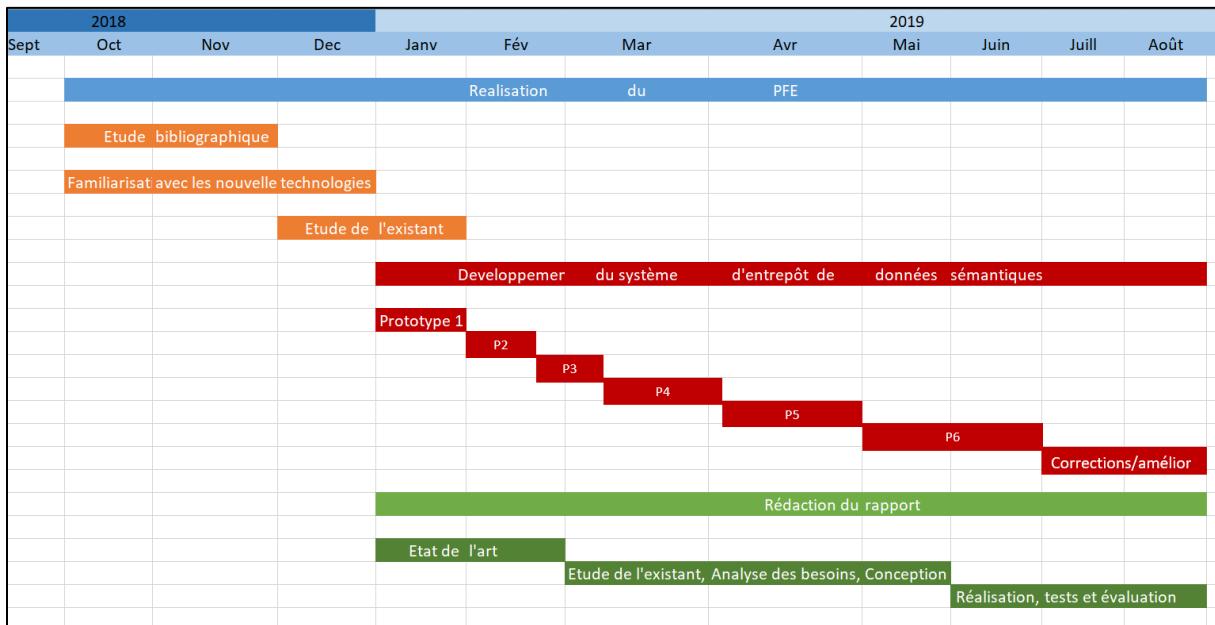


Figure 81 : Diagramme de Gantt du projet

4 Principales phases du projet

La réalisation de notre projet est passée par plusieurs phases complémentaires. Nous résumant ces phases comme suit :

- **Etude bibliographique :** La toute première étape que nous avons entamé dans notre projet c'est d'effectuer des recherches bibliographiques à propos des différentes notions clés de notre sujet. Nous nous sommes donc documenté en détail à propos des EDs, du Web Sémantique, des LOD et enfin des EDS ou ED ontologiques.
- **Familiarisation avec les nouvelles technologies :** L'étude bibliographique qui donne un aspect uniquement théorique des notions clés de notre projet s'avère insuffisant afin de comprendre les notions nouvelles comme les différents composants du Web Sémantique et les LOD. Par conséquent nous avons eu recours à d'autres techniques d'apprentissage qui se sont orientées beaucoup plus vers la pratique. Ces formes d'apprentissage, se résument en Tutoriels, Cours audiovisuels, Exploitation technique des LOD, Lancement de requêtes SPARQL, etc. Nous avons donc consacré cette phase pour l'apprentissage des outils et des technologies prévues pour la réalisation de notre système comme : Jena API, OWL, RDF, RDFS, SPARQL, Protege, etc.
- **Etude de l'existant :** Elle représente une phase très importante. C'est la phase avec laquelle nous avons lancé le développement du projet. Ça consiste à étudier et comprendre l'approche orientée besoins, et faire la rétro conception du prototype existant à partir d'un

outil qui présente beaucoup de problèmes et de lacunes relatives à l'utilisation, aux performances, aux fonctionnalités disponibles et à l'ergonomie de celui-ci.

- **Développement du système :** Représente la phase pendant laquelle nous avons développé les différents prototypes du système. Nous avons travaillé sur cette phase jusqu'à la fin du projet, suite aux corrections et améliorations proposées.
- **Tests et évaluation :** C'est la phase qui nous a permis de tester les performances de notre système et d'évaluer son fonctionnement grâce à un certain nombre de tests et métriques.
- **Rédaction du mémoire :** La rédaction du mémoire a été divisé en trois grandes parties : l'état de l'art, la conception et la réalisation.

Conclusion générale

Grâce à l'évolution du web, le phénomène d'ouverture des données vers l'extérieur et leur grande disponibilité, a pris une grande ampleur de nos jours. C'est l'une des raisons qui ont poussé les décideurs à recenser le besoin de considérer d'autres sources de données qui viennent enrichir et compléter les données internes à l'entreprise. Cette initiative, permet à l'entreprise d'avoir une meilleure visibilité et de créer une valeur ajoutée aux systèmes d'aide à la décision.

Vu l'évolution étonnante de la masse de données sur le web, certaines initiatives ont été prises pour les automatiser et les partager en licence libre afin de créer de la valeur. Deux de ses initiatives en particulier : le Web Sémantique et les Données Ouvertes et Liées, ont fortement suscité l'attention des concepteurs des entrepôts de données, grâce à leur structure sémantique et à leur richesse en données et métadonnées qui représentent potentiellement une valeur ajoutée au processus de décision.

C'est dans ce contexte, que notre projet de fin d'études a débuté. Le but de celui-ci, est de concevoir un entrepôt de données sémantique en prenant en considération les besoins des décideurs. Cette démarche permet de sélectionner uniquement les sources de données pertinentes parmi une grande masse de données externes afin d'atteindre les buts de l'entreprise. C'est donc une approche orientée buts, elle est conçue pour éviter le surmenage de l'EDS avec des données parasites qui ne représentent aucune valeur ajoutée à l'entreprise. Le principal objectif de notre travail, est de proposer et de concevoir un système d'intégration complet, qui permet d'intégrer les données et les métadonnées provenant de sources différentes.

Durant ce projet, nous distinguant deux grandes phases. En premier lieu, la phase de recherche documentaire et collecte d'informations à propos des Entrepôts de données, du Web Sémantique et du croisement entre les deux. La deuxième phase, représente la phase la plus longue et la plus pertinente. Elle présente notre contribution par la proposition d'une solution globale répondant aux problématiques de notre sujet de recherche. Nous avons apporté des améliorations à travers deux majeurs contributions. La première contribution est de proposer une amélioration à l'ancien système qui implémente la démarche par besoins. Nous avons recensé l'ensemble des problèmes et lacunes rencontrées lors de son utilisation. À partir de ces problèmes nous avons proposé une nouvelle conception du système existant qui le rend plus intuitif et efficace. La seconde contribution sur laquelle repose la majeure partie de notre travail, c'est de proposer une nouvelle conception, d'un système d'intégration d'un EDS qui intègre à la fois les données et les métadonnées des sources. Cette contribution touche deux volets principaux, le premier représente l'intégration des données grâce à des mappings ETL. Le but principal de celui-ci est d'unifier les sources internes et externes, et transforme les instances sources en instances cibles. Le deuxième volet c'est l'intégration des métadonnées au deux niveaux : sémantique et format. Nous avons vu dans ce travail, l'existence de plusieurs formats de métadonnées dans les sources externes. L'unification de ces sources nécessite aussi l'unification de ses différents formats de métadonnées. Le processus d'intégration des métadonnées permet grâce à des méta-mappings sémantiques de charger et classer les méta-propriétés dans le méta-modèle cible SM4AM que nous avons détaillé dans la partie conception de notre rapport.

Une fois que nous avons implémenté notre système, nous avons effectué un ensemble de tests afin d'évaluer le fonctionnement du processus d'intégration. Nous l'avons testé sur des données internes et externes de deux formats « STDREIF » et « Nary ». Les résultats du scénario de test confirment l'efficacité de notre système.

▪ Perspectives de recherches :

Dans la suite de la solution proposée et de l'étude que nous avons effectuée, de nombreuses perspectives résultent :

- Élargir les sources de données aux portails LOD: Dans notre solution nous avons pris en considération deux types de sources de données, les sources de données internes à l'entreprise et les données externes qui représentent des données extraites sous forme d'ontologies que nous avons exploitées. Pour éliminer les barrières du volume de données à traiter, nous comptant faire une extraction directement à partir des portails LOD en utilisant les différents SPARQLEndpoints.
- Unifier les formats de métadonnées restantes : Nous avons traité dans notre système l'unification de deux formats de données : « STDREIF » et « NARY ». Nous estimons dans les recherches à venir de proposer des algorithmes d'extraction et de transformation adéquats pour l'unification des formats de métadonnées restants.
- Optimiser le système d'intégration : L'efficacité d'un système d'intégration revient à diminuer continuellement le temps d'exécution tout en augmentant la masse de données traitée. Dans ce premier prototype nous nous sommes concentrés sur la solution et nous avons appliqué des méthodes de tri d'une faible complexité pour augmenter les performances lors de l'étape de transformation. Cependant, nous comptant à l'avenir utiliser des méthodes de traitement parallèles qui permettent de diminuer le temps d'exécution.
- Améliorer l'outil en ajoutant les fonctionnalités d'évolution des besoins et des mappings.
- Automatiser certaines fonctionnalités, qui sont manuelles dans l'outils comme la correction automatique des chevauchements.
- Prévoir un aspect sécurité des données dans le système d'EDS.

Bibliographie

[Goglin, 2001] Goglin J.-F. (2001, 1998). La construction du datawarehouse : du datamart au dataweb. Hermès, 2ème édition.

[B. Inmon, 2002] B. Inmon W.-H. (2002, 1990). Building the data warehouse. Wiley Publishing, 3rd edition.

[R. Kimbal et al., 2003] Kimball, R., & Ross, M. (2003). Entrepôts de données : guide pratique de modélisation dimensionnelle. Vuibert informatique.

[R. Kimball et al., 2008] Kimball R., Ross M. (2008, 2002). The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. Wiley Publishing, second edition.

[A. Vaisman et al., 2014] A. Vaisman, & E. Zimányi. (2014). Data warehouse systems. Data-Centric Systems and Applications.

[W. Eckerson, 2003] W. Eckerson. (2003). Four ways to build a data warehouse. What works, 15.

[Sansu George, 2012] George, S. (2012). Inmon vs. Kimball: Which approach is suitable for your data warehouse?. Data Warehous., 1-12. <<https://www.computerweekly.com/tip/Inmon-or-Kimball-Which-approach-is-suitable-for-your-data-warehouse>>

[S. Chafki, C. Desrosiers, 2011] S. Chafki, C. Desrosiers. (2011). MTI820-Entrepôts de données et intelligence d'affaires : Architecture des entrepôts de données. <https://cours.etsmtl.ca/mti820/public_docs/acetates/mti820-acetates-architecturedw_1pp.pdf>

[M.Volle, 2001] Michel VOLLE. (2001). <<http://www.volle.com/rapports/siad.htm>>

[J. Serra, 2013] James Serra. (2013). Building an Effective Data Warehouse Architecture. <<https://www.jamesserra.com/archive/2013/07/building-an-effective-data-warehouse-architecture-presentation-2/>>

[R. Kimball, 2000] R. Kimball. (2000). Concevoir et détailler un Data Warehouse. Editions Eyrolles. <<https://www.eyrolles.com/Chapitres/9782212116007/chap02.pdf>>

[S. Adelman et al., 2000] S. Adelman, & L. Terpeluk Moss. (2000). Data warehouse project management. Addison-Wesley Longman Publishing Co., Inc. <<http://www.eiminstitute.org/resource-portals/data-warehousing>> <<http://www.eiminstitute.org/resource-portals/data-warehousing/data-warehouse-goals-and-objectives-part-1/>>

[M. Mahdi Malik, 2002] Malik, M. M. (2002). Le rôle de la logique floue dans le web sémantique. Secrétariat du DEA d'Informatique—Université de la Méditerranée Faculté des Sciences de Luminy—Case 901 163, Avenue de Luminy—13288 MARSEILLE Cedex 9 Tel: 0 491 829 316—Télécopie : 0 491 829 275 secrétariat : secretariat-dea@ dil. univ-mrs. fr, 263.

[C. Rigaud-Alfano et al., 2001] Carine Rigaud-Alfano, Olivia Schnarch. Définir les standards du Web d'aujourd'hui et de demain. Dossier de presse W3C, Florence Gillier Communication, Automne 2001.

[J. Charlet et al., 2003] Charlet, J., Laublet, P., & Reynaud, C. (2003). Le web sémantique. Cépaduès-Ed.

[C. Fuchs et al., 2010] Fuchs, C., Hofkirchner, W., Schafranek, M., Raffl, C., Sandoval, M., & Bichler, R. (2010). Theoretical foundations of the web: cognition, communication, and co-operation. Towards an understanding of Web 1.0, 2.0, 3.0. Future Internet, 2(1), 41-59.

[S. Aghaei, et al., 2012] Aghaei, S., Nematbakhsh, M. A., & Farsani, H. K. (2012). Evolution of the world wide web: From WEB 1.0 TO WEB 4.0. International Journal of Web & Semantic Technology, 3(1), 1.

[B. Getting, 2007] Getting, B. (2007). Basic Definitions: Web 1.0, Web. 2.0, Web 3.0. Practical eCommerce : Insights for Online Merchants. (2007).<<http://www.practicalecommerce.com/articles/464-Basic-Definitions-Web-1-0-Web-2-0-Web-3-0>>.

[T. Berners-Lee, 1998] Berners-Lee, T. (1998). The World Wide Web : A very short personal history. Tim Berners-Lee, 7.

[E. Heindl, 2008] Heindl, E., & Suphakorntanakit, N. (2008). Web 3.0. E-Business Technology, Hochschule Furtwangen University, Furtwangen, Germany, <http://webuser.hsfurtwangen.de/~heindl/ebte-08ss-web-20-Suphakorntanakit.pdf> adresinden, 15, 2012.

[O. Nykänen, 2003] Ossi, Nykänen (2003). Semantic Web : Definition, <<http://www.w3c.tut.fi/talks/2003/0331umediaon/slide6-0.html>>.

[T. Berners-Lee et al., 2006] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. (2006). Tabulator: Exploring and Analyzing linked data on the Semantic Web.

[A. Zuiderwijk et al., 2012] Zuiderwijk, A., Jeffery, K., & Janssen, M. (2012). The potential of metadata for linked open data and its value for users and publishers. JeDEM-eJournal of eDemocracy and Open Government, 4(2), 222-244.

[N. Y. Chen et al., 2015] Chen, Y. N. (2015). A RDF-based approach to metadata crosswalk for semantic interoperability at the data element level. Library hi tech, 33(2), 175-194.

[C. Chiaramonti, 2001] Claude Chiaramonti. Semantic Web : contenu avec références. <<http://xmlfr.org/actualites/decid/010402-0001>, mai 2001>.

[J. Greenberg et al., 2003] Jane, Greenberg & Stuart, Sutton & D. Grant, Campbell (2003), Metadata: A Fundamental Component of the Semantic Web. Bulletin of the American Society for Information Science and Technology Volume 29, Issue 4, pages 16–18.

[H. Melhem, 2017] Melhem, H. (2017). Usages et applications du web sémantique en bibliothèques numériques (Doctoral dissertation, UNIVERSITE GRENOBLE ALPES/Laboratoire GRESEC (Groupe de recherche sur les enjeux de la communication)).

[J. Jaffe, 2016] J. Jaffe (W3C CEO). (2016). From a World-Wide Web of Pages to a World-Wide Web of Things Interoperability for Connected Devices.

[Gandon et al., 2012] GANDON, F., CORBY, O., et Faron-ZUCKER, C. (2012). Le web : Comment lier les données et les schémas sur le web?. Dunod, Paris.

[Bermès et al., 2013] BERMES, E., ISAAC, A. et POUPEAU, G. (2013) Le Web sémantique en bibliothèque. Editions du cercle de la Librairie, Paris.

[Brunet et Xuan, 2010] Brunet, N., Truong Vu, Xuan, (2010). Le web sémantique : en quoi le web sémantique permet-il d'aborder le sens ? <http://vuxuantr.free.fr/res/rapport_web_semantique-Nicolas_Brunet-Xuan_Truong_Vu.pdf>

[Z. Djilani, 2017] Djilani, Z. (2017). Donner une autre vie à vos besoins fonctionnels : une approche dirigée par l'entreposage et l'analyse en ligne (Doctoral dissertation, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers).

[S. Khouri, 2013] Khouri, S. (2013). Cycle de vie sémantique de conception de systèmes de stockage et manipulation de données (Doctoral dissertation, ISAE-ENSMA Ecole Nationale Supérieure de Mécanique et d'Aérotechique-Poitiers).

[L. Bellatreche et al., 2006] L. Bellatreche, N. X. Dung, G. Pierra, and D. Hondjack. Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry*, 57(8) :711– 724, 2006.

[Gruber, 1993] GRUBER, T. R. (1995). Toward principles for the design of ontologies used for knowledge sharing?. *International journal of human-computer studies*, 43(5-6):907-928.

[Borst, 1997] BORST, W.N. (1997). Construction of engineering ontologies for knowledge sharing and reuse. Universiteit Twente, Enschede.

[H. Wache et al., 2001] Wache, H., Voegele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., & Hübner, S. (2001, August). Ontology-Based Integration of Information-A Survey of Existing Approaches. In OIS@ IJCAI.

[G. Pierra, 2003] G. Pierra. (2003). Context-explication in conceptual ontologies : the plib approach. In Proceedings of 10th ISPE International Conference on Concurrent Engineering : Research and Applications (ce'03) : Special Track on Data Integration in Engineering, pages 243–253.

[Gandon et al., 2012] GANDON, F., CORBY, O., et Faron-ZUCKER, C. (2012). Le web sémantique : Comment lier les données et les schémas sur le web?. Dunod, Paris.

[H. BENHMIDI et al., 2011] BENHMIDI, H., & MAZOUZI, A. Construction et manipulation d'une ontologie médicale.

[A. Fernandez, 2017] A. Fernandez. (2017). Collecte des données : ETL Extract Transform load. <<https://www.piloter.org/business-intelligence/ETL.htm>>

[S. Auer, 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., & Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *The semantic web* (pp. 722-735). Springer, Berlin, Heidelberg.

[C. Bizer et al., 2011] Bizer, C., Heath, T., & Berners-Lee, T. (2011). Linked data: The story so far. In *Semantic services, interoperability and web applications: emerging concepts* (pp. 205-227). IGI Global.

[G. Pierra, 2003] Guy Pierra. (2003). Context-explication in conceptual ontologies : the plib approach. In Proceedings of 10th ISPE International Conference on Concurrent Engineering : Research and Applications (ce'03) : Special Track on Data Integration in Engineering, pages 243–253.

[R. Mizoguchi et al., 2000] Riichiro Mizoguchi, Kouji Kozaki, Toshinobu Sano, and Yoshinobu Kitamura. Construction and Deployment of a Plant Ontology, pages 113–128. Springer Berlin Heidelberg, Berlin, Heidelberg.

[C. Fankam, 2009] C. Fankam. (Décembre 2009). OntoDB2 : un système flexible et efficient de Base de Données à Base Ontologique pour le Web sémantique et les données techniques. PhD thesis, ENSMA.

[M. Hacid, 2004] Hacid, M. S., & Reynaud, C. (2004). L'intégration de sources de données. *Revue Information-Interaction-Intelligence*, 3, 4.

[G. Pierra et al., 2005] Pierra, G., Dehainsala, H., Ameur, Y. A., & Bellatreche, L. (2005). Bases de données à base ontologique. Principe et mise en oeuvre. *Ingénierie des systemes d'information*, 10(2), 91-115.

[J. Lu et al., 2007] J. Lu, L. Ma, L. Zhang, J.-S. Brunner, C. Wang, Y. Pan, and Y. Yu. Sor. (2007). A practical system for ontology storage, reasoning and search. In Proceedings of the 33rd international conference on Very large data bases, VLDB '07, pages 1402–1405. VLDB Endowment.

[Z. Wu et al., 2008] Z. Wu, G. Eadon, S. Das, E. Chong, V. Kolovski, M. Annamalai, and J. Srinivasan. (2008) Implementing an inference engine for rdfs/owl constructs and user-defined rules in oracle. In ICDE, pages 1239–1248, 2008.

[S. Alexaki et al., 2001] S. Alexaki, V. Christophides, G. Karvounarakis D. Plexousakis, and K. Tolle. (2001). The icsforth rdfsuite : Managing voluminous rdf description bases. In Proceedings of the 2nd International Workshop on the Semantic Web (SemWeb 2001).

[C. Rolland, 2003] C. Rolland. (2003). Ingénierie des Besoins : L'Approche L'Ecritoire. *Journal Techniques de l'Ingénieur*.

[A. Aybuke et al., 2005] A. Aybuke and W. Claes. (2005). Engineering and Managing Software Requirements.

[C. Calero et al., 2006] C. Calero, F. Ruiz, and M. Piattini. (2006). *Ontologies for Software Engineering and Software Technology*. Springer Verlag, 2006.

[D Brickley et al., 2002] Brickley D., Guha R. (2002). RDF Vocabulary Description Language 1.0 : RDF Schema. W3C. <<http://www.w3.org/TR/rdf-schema/>>.

[D. Connolly et al., 2001] Connolly D., van Harmelen F., Horrocks I., McGuinness D. L., Patel-Schneider P. F., Stein L. A. (Mars 2001). « DAML+OIL Reference Description », W3C, <<http://www.w3.org/TR/daml+oilreference>>.

[S. Bechhofer et al., 2004] Bechhofer S., van Harmelen F., Hendler J., Horrocks I., McGuinness D., Patel-Schneider P., Stein L. OWL Web Ontology Language Reference. W3C. <<http://www.w3.org/TR/owlref/>>.

[ISO-13584-42, 1998] ISO-13584-42. (1998). Industrial Automation Systems and Integration Parts LIBRARY Part 42 : Description methodology : Methodology for Structuring Parts families, Technical report. ISO.

[R. van Solingen et al., 2002] R. van Solingen, V. Basili, G. Caldiera, and H. D. Rombach. (2002). Goal question metric (gqm) approach. Encyclopedia of Software Engineering.

[P. Vassiliadis et al., 2000] P. Vassiliadis, M. Bouzeghoub, and C. Quix. (2000). Towards quality-oriented data warehouse usage and evolution. *Inf. Syst.*, 25(2) :89– 115.

[T. BERNERS-LEE et al., 2001] BERNERS-LEE, Tim, HENDLER, James et LASSILA, Ora. (Mai 2001). The semantic web. *Scientific American*. p. 29-37. <<http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html>>

[T. Berners-Lee, 2006] Tim Berners-Lee. (2006). "Linked Data". W3C. <<https://www.w3.org/DesignIssues/LinkedData.html>>

[V. Russo, 2015] Vanessa Russo. (2015). Semantic Web: Metadata Linked Data Open Data, Science & Philosophy, 3(2)/2015, 37- 46.

[E. Duval, 2001] Duval, E. (2001), Metadata Standards, What, Who & Why, Journal of Universal Computer Science, 7(7), 591-601, Springer.

[I. Sommerville et al., 1998] I. Sommerville and G. Kotonya. Requirements Engineering : Processes and Techniques. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[D.C. Moura, 1998] De Carvalho Moura, A.M., Machado Campos, M.L., Barreto, C.M. (1998), A survey on metadata for describing and retrieving Internet resources, World Wide Web, 1, (4), 221-240, Kluwer Academic Publishers.

[D. Skoutas et al., 2007] D. Skoutas and A. Simitsis, "Ontology-based conceptual design of etl processes for both structured and semi-structured data," Int. J. Semantic Web Inf. Syst., vol. 3, no. 4, pp. 1–24, 2007.

[Z. Marcia, 2004] Zeng, Marcia (2004). "Metadata Types and Functions". NISO. Archived from the original on 7 October 2016. <<http://marciazeng.slis.kent.edu/metadatabasics/types.htm>>

[OECD Statistics, 2018] Directorate, OECD Statistics. (2018). OECD Glossary of Statistical Terms - Reference metadata Definition. stats.oecd.org.

[W. Y. Arms, 1997] Arms, W. Y., Blanchi, C., & Overly, E. A. (1997). An architecture for information in digital libraries. D-lib Magazine, 3(2). <<http://www.dlib.org/dlib/february97/cnri/02arms1.html>>

[NISO, 2001] National Information Standards Organization (NISO) (2001). Understanding Metadata. NISO Press. p. 1. ISBN 978-1-880124-62-8. Archived (PDF) from the original on 7 November 2014.

[C. Dippo, 2000] Dippo, C. (2000). The role of metadata in statistics. In Proceedings of the Second International Conference on Establishment Surveys, 2000 (pp. 909-918). <<https://www.bls.gov/ore/pdf/st000040.pdf>>

[T. Stöhr et al., 1999] Stöhr, T., Müller, R., & Rahm, E. (1999, June). An integrative and uniform model for metadata management in data warehousing environments. In Proceedings of the International Workshop on Design and Management of Data Warehouses, Heidelberg, Germany (Vol. 189). <https://www.researchgate.net/profile/Erhard_Rahm/publication/220841950_An_Integrative_and_Uniform_Model_for_Metadata_Management_in_Data_Warehousing_Environments/links/568d692608aea d3f42ed9f4c.pdf>

[P. Y. Vandenbussche et al., 2017] Vandenbussche, P. Y., Atemezing, G. A., Poveda-Villalón, M., & Vatant, B. (2017). Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. Semantic Web, 8(3), 437-452.

[L. Parkin, 2018] L.Parkin. (2018). La Contribution des Données Ouvertes et Liées dans les Administrations.

[J. Frey, 2017] Frey, J., Müller, K., Hellmann, S., Rahm, E., & Vidal, M. E. (2017). Evaluation of Metadata Representations in RDF stores. Semantic Web, (Preprint), 1-25.

[J. Varga et al., 2018] Varga, J., Romero, O., Pedersen, T. B., & Thomsen, C. (2018). Analytical metadata modeling for next generation BI systems. Journal of Systems and Software, 144, 240-254.

[M. Golfarelli et al., 2009] Golfarelli, M., & Rizzi, S. (2009, November). A comprehensive approach to data warehouse testing. In Proceedings of the ACM twelfth international workshop on Data warehousing and OLAP (pp. 17-24). ACM.

[L. Bellatreche et al., 2006] L. Bellatreche, N. X. Dung, G. Pierra, and D. Hondjach. (2006). Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry*, 57(8) :711–724.

[M. Lenzerini, 2002] M. Lenzerini. (2002). Data integration: A theoretical perspective. In PODS, pages 233–246.

[F. Irini, 2016] Fundulaki Irini. (2016). Assessing the performance of RDF Engines: Discussing RDF Benchmarks Irini Fundulaki Institute of Computer Science–FORTH, Greece Anastasios Kementsietsidis. Google Research, USA 6/15/16 ESWC 2016: Assessing the performance of RDF Engines – Discussing RDF Benchmarks.

[S. Khouri et al., 2018] Khouri, S., & Bellatreche, L. (2018, May). Consolidation of BI efforts in the LOD era for african context. In Proceedings of the 2018 International Conference on Software Engineering in Africa (pp. 1-10). ACM.

[H. Wache et al., 2001] Holger Wache, Thomas Voegle, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian H'ubner. 2001. Ontology-based integration of information-a survey of existing approaches. In IJCAI-01 workshop: ontologies and information sharing, Vol. 2001. Citeseer, 108–117.