UNIVERSITE SAAD DAHLAB BLIDA

Faculté des Sciences

Département d'Informatique

MEMOIRE DE MAGISTER

en Informatique

Spécialité: Système d'Information et de Connaissance

SYSTEME HYBRIDE D'INTEGRATION AUTOMATIQUE DES BASES DE DONNEES A BASE ONTOLOGIQUE

Par

Merièm ARKAM

Devant le jury composé de

N. BENBLIDIA	Maitre de conférences A USDB	Président
S. OUKID-KHOUAS	Maitre de conférences A USDB	Promotrice
F. SOUAMI	Maitre de conférences A USTHB	Co-promotrice
R. CHALAL	Maitre de conférences A ESI	Examinateur
F. NADER	Maitre de conférences A ESI	Examinatrice

Blida, Octobre 2012

RESUME

La diversité des sources de données distribuées et leur hétérogénéité est l'une des principales difficultés rencontrées par les utilisateurs aujourd'hui. Les solutions à l'intégration des sources de données tireront parti des recherches concernant les approches de médiation, les entrepôts de données ou encore les approches semi-matérialisée, cependant leur grande difficulté est l'interprétation automatique de la signification et la sémantique des données hétérogènes et autonomes. Le travail présenté dans cet article porte sur l'intégration des sources de données à base d'ontologie par une approche hybride ou semi-matérialisée, cette approche offre un compromis entre le temps de réponse et le temps de mise à jour des données.

Dans notre travail, nous avons nous avons modélisé notre architecture d'intégration hybride à base ontologique, et puis défini un algorithme d'intégration automatique des sources de données à base ontologiques (HybExtendOnto) permettant de créer l'entrepôt de données et le schéma virtuel (médiateur), en se basant sur le principe suivant : tout concept existant dans un nombre supérieur à ou égal à un nombre n de sources de données ou étant référencé par au moins n sources sera intégré dans la partie matérialisée, sinon il sera intégré dans la partie virtuelle, , les résultats obtenus jusqu'à maintenant sont jugé satisfaits.

Mots clés :

Intégration de données, ontologie, Approches d'intégration, Médiateur, Entrepôt de données, approche hybride, Bases de données à base d'ontologie, intégration DBDO,

ABSTRACT

The variety of distributed data sources and their heterogeneity is one of the main difficulties faced by users today. The solutions to the integration of data sources will benefit research on approaches to mediation, data warehouses or semi-materialized approaches, however, their greatest difficulty is the automatic interpretation of the meaning and semantics of heterogeneous data and autonomous. The work presented in this thesis focuses on the hybrid (or semi-materialized) integration of data sources based on ontology, this approach offers a compromise between response time and the time to update data.

In our work, we then we have modeled our hybrid integration architecture ontological basis the we defined an algorithm for automatic integration of data sources based ontological (HybExtendOnto) to create the data warehouse and virtual (mediator) schemas, based on the following principle: any existing concept in a number greater than or equal to n number of data sources or being referenced by at least two (02) sources will be included in the materialized, otherwise it will be integrated into the virtual part, the results so far are considered satisfied.

Keywords:

Data integration, ontology, integration approaches, Mediator, Data Warehouse, hybrid approach, DataBase-Based Ontology; Integration of DBBO

ملخص

تنوع مصادر البيانات الموزعة وعدم تجانسها هي واحدة من الصعوبات الرئيسية التي يواجهها المستخدمين اليوم. الحلول المستعملة لإدماج هذه المصادر تستفيد من البحوث الخاصة بالمناهج الافتراضية (الظاهرية)، ومستودعات البيانات أو النهج الهجين بين المنهجين الأوليين، ومع ذلك، فإن الصعوبة الأكبر هي في التفسير التلقائي لمعنى ودلالات البيانات المستقلة وغير المتجانسة. العمل المقدمة في هذه المذكرة تركز على تكامل مصادر البيانات استنادا الأنطولوجيا مع هجين أو شبه متحقق، هذا النهج يوفر حلا وسطا بين زمن الاستجابة والوقت لتحديث البيانات.

في عملنا هذا، قمنا بتحديد الهيكل الرئيسي لنظام دمج مصادر المعلومات على أساس الأنطولوجيا، ثم حددنا خوارزمية لتحقيق الإدماج الأوتوماتيكي (التلقائي) لمصادر البيانات المستندة على أنطولوجيا (HybExtendOnto) وذلك لإنشاء مستودع بيانات المخطط والنظام الإفتراضي (وسيط)، استنادا إلى المبدأ التالي: أي مفهوم موجود في أكثر من مصدر بيانات أو يتم الرجوع إليه من مصدرين على الأقل يندمج في مستودع البيانات، وإلا فسيتم دمجه في الجزء الافتراضي، ثم ونحن على غرار فن العمارة لدينا التكامل الهجين أساس وجودي، والنتائج حتى الآن تعتبر جيدة

كلمات البحث:

إدماج البيانات، نهج الإدماج، الأنطولوجيا، الإدماج، مستودع البيانات، نهج هجين، قواعد البيانات المستندة إلى الأنطولوجيا، إدماج البيانات على أساس الأنطولوجيا

REMERCIEMENTS

Louange à Dieu Tout Puissant qui m'a accordé force et volonté pour aboutir à terminer mon mémoire.

{وَمَا بِكُمْ مِنْ نِعْمَةٍ فَمِنَ اللّهِ} [النحل، 53]

Je tiens à adresser mes plus sincères remerciements à ma promotrice madame OUKID KHOUAS, et responsable du laboratoire LRDSI, pour m'avoir acceptée, encadrée et aidée tout au long de mes études en magister et surtout pour son soutien moral et la confiance qu'elle m'a accordée tout au long de cette année pour tous ses conseils avisés..

Je tiens également à remercier madame SOUAMI pour avoir contribuée à l'encadrement de mon travail.

Je souhaiterais remercier madame BENBLIDA, madame NADER et monsieur CHALLAL qui m'ont fait l'honneur d'accepter d'être examinateurs.

Je remercie tous mes enseignants depuis le primaire jusqu'à la post-graduation, sans exception, et toute personne ayant contribué à enrichir mon savoir.

A mes proches qui ont cru en moi et qui m'ont aidé et soutenu tout au long de cette épreuve.

SOMMAIRE

RESUME 2
REMERCIEMENTS 5
SOMMAIRE 6
LISTE DES FIGURES 11
LISTE DES TABLEAUX 13
INTRODUCTION GENERALE 14
1. Introduction 14
2. Intégration des données 16
3. Correspondance entre les Schémas 16
4. Domaine d'application et objectif 19
5. Contribution 21
6. Organisation du mémoire 21
PARTIE 1 ETAT DE L'ART 23
1. INTEGRATION DE DONNEES 25
1.1. Introduction 25
1.2. Problèmes d'intégration de données 26
1.2.1. Problème d'Autonomie 26
1.2.2. Problème d'Interdépendance 27
1.2.3. Problème d'Hétérogénéité 27
1.2.4. Discussion 31
1.3. Systèmes d'intégration de données 32
1.3.1. Définition 33
1.3.2. Processus d'intégration 34

1.3.3. Tache d'un système d'intégration....35
1.3.4. Classification des approches d'intégration de données 36
1.3.4.1. Localisation de données intégrées / Architecture d'intégration 37
1.3.4.2. Nature de correspondance entre schéma global et schéma local 46
1.3.4.3. Processus d'intégration / Automaticité du mapping 49
1.4. Quelques systèmes d'intégrations....51
1.4.1. Information Manifold....51
1.4.2. InfoMaster 51
1.4.3. L'approche de Florescu....52
1.4.4. GARLIC....52
1.4.5. MIX....53
1.4.6. PICSEL 53
1.4.7. OBSERVER 54
1.4.8. DISCO....54
1.4.9. TSIMMIS 54
1.4.10. MOMIS....55
1.4.11. XYLEME 55
1.4.12. PIAZZA 55
1.4.13. Discussion....56
1.5. Conclusion....57
2.LES SYSTEMES D'INTEGRATION HYBRIDES....58
2.1. Introduction....58
2.2. Système d'intégration de données Hybride....59
2.3. Modèle d'Architecture d'système hybride (semi-matérialisé) 60
2.4. Les principaux systèmes hybrides....61
2.4.1. Une approche hybride basée sur l'intégration des médiateurs Squirrel 62
2.4.2. Intégration de données semi-structurées avec le système de base de données Lore....63
2.4.3. Une architecture hybride pour l'entreposage du contenu Web....64
2.4.4. Une approche hybride pour l'intégration d'ontologie 65
2.4.5. IXIA, IndeX-based data Integration Approach....65
2.4.6. Synthèse....67
2.5. Conclusion....69

3. ONTOLOGIE ET BASE DE DONNEES A BASE D'ONTOLOGIE 70
3.1. Introduction 70
3.2. Ontologie 71
3.2.1. Description d'une ontologie 73
3.2.1.1. Les composants de l'ontologie 73
3.2.1.2. Types d'ontologies 74
3.2.2. Quelques domaines d'utilisation d'ontologies 77
3.2.2.1. Traitement du langage naturel 77
3.2.2.2. Web sémantique 77
3.2.2.3. Base de données 78
3.2.2.4. Intégration des sources de données 78
3.2.3. L'ontologie et les systèmes d'intégration 80
3.2.3.1. Architecture utilisant une ontologie unique 82
3.3. Bases de Données à Bas d'Ontologie (BDBO) 84
3.3.1. Schémas existants pour la représentation des ontologies et des données 84
3.3.1.1. Représentation des ontologies 84
3.3.1.2. Représentation des données 85
3.3.2. Quelques architectures des BDBO 86
3.3.2.1. Architecture Sesame 86
3.3.2.2. ICS-FORTH RDFSuite 87
3.3.2.3. Jena Architecture 87
3.3.2.4. Le système kwOWLer 88
3.3.2.5. Le Modèle OntoDB 88
3.3.3. Caractéristiques des SGBDBO : 90
3.3.4. Synthèse 91
3.4. Conclusion 92
PARTIE 2 NOTRE ARCHITECTURE PROPOSEE 93
4. APPROCHE D'INTEGRATION HYBRIDE DE BASE DE DONNEES A BASE D'ONTOLOGIE 94
4.1. Introduction 94
4.2. Présentation de l'approche 96
4.3. Objectifs à atteindre 96
4.4. Description des sources de données 97
4.5. Processus d'intégration 98

4.6. Représentation formelle de l'approche 99
4.6.1. Représentation de l'ontologie 99
4.6.2. Représentation des sources de données à Base d'Ontologie 99
4.6.3. Représentation de l'approche d'intégration à base ontologique ... 100
4.7. Architecture du système hybride d'intégration de données « HybOnto » 101
4.7.1. Architecture logicielle du système HybOnto: 101
4.7.2. Les principaux composants du système HybOnto 102
4.7.3. Représentation des connaissances dans HYBOnto (Mapping de données) 108
4.7.3.1. Langage de vues 108
4.7.3.2. Correspondances entre schémas – Global as View 109
4.7.4. Traitement de requêtes 110
4.8. Discussion 111
4.9. Amélioration de l'approche 112
4.10. Conclusion 114
5. SCENARII D'INTEGRATION 115
5.1. Introduction 115
5.2. Architecture d'intégration du système HYBOnto 117
5.2.1. Notions élémentaires 117
5.2.1.1. Concept ontologique 117
5.2.2. Présentation du système d'intégration HYBOnto 118
5.2.2.1. Infrastructure 118
5.2.2.2. Base de connaissances 118
5.2.2.3. Langage de manipulation des données 119
5.3. Principe d'engagement sur l'ontologie partagée (de référence) .. 119
5.4. Les scénarii d'intégration 120
5.4.1. Description formelle du contexte 120
5.4.2. Formalisation des scénarii d'intégration proposés 122
5.4.2.1. Le scénario basé sur la fragmentation HybFragOnto 122
5.4.2.2. Le scénario basé sur la projection HybProjOnto 126
5.4.2.3. Scénario d'extension de projection HybProjExtendOnto 128
5.4.2.4. Scénario d'intégration global « HybExtendOnto » 131
5.5. Conclusion 135
6. REALISATION ET VALIDATION 136

6.1. Introduction 136
6.2. Environnement de mise en œuvre 137
6.3. Les étapes d'implémentation 137
6.4. La construction globale de notre architecture 138
6.4.1. Construction des sources de données locales à base d'ontologie 138
6.4.1.1. Architecture conceptuelle d'une Base de Données à Base d'Ontologie 139
6.4.1.2. Création de la partie méta-schéma 140
6.4.1.3. Remplissage de l'ontologie 142
6.4.1.4. Création de la source de données 142
6.4.1.5. Remplissage des instances de la source de données locale 144
6.4.2. Construction de l'ontologie partagée (l'ontologie de domaine) 145
6.4.3. Construction du système d'intégration Hybride 147
6.4.3.1. Construction de l'entrepôt de données (EDBD) 147
6.4.3.2. Construction du médiateur (MEDBO) 148
6.4.4. Construction de la partie utilisateur 149
6.4.4.1. L'interface Utilisateur 149
6.4.4.2. Le moteur de requête 150
6.5. Conclusion 154
CONCLUSION ET PERSPECTIVE 155
1. Conclusion 155
2. Perspectives 156
APPENDICE 158
REFERENCES 175

LISTE DES FIGURES

Figure 1.1 Exemple de source hétérogène 31
Figure 1.2 Système d'intégration de données 34
Figure 1.3 Modèle d'architecture fédérée 39
Figure 1.4 Architecture d'un système médiateur 40
Figure 1.5 Architecture d'un entrepôt de données 43
Figure 1.6 Architecture d'un système Pair-à-Pair 44
Figure 2.1 Architecture d'un système d'intégration de données hybride (semi-matérialisé) 61
Figure 2.2 Architecture d'un médiateur en utilisant une ontologie partagée 62
Figure 2.3 Architecture du système d'intégration de données Lore 63
Figure 2.4 Architecture de l'approche hybride pour l'intégration d'ontologie 65
Figure 2.5 Architecture d'intégration de données IXIA 66
Figure 3.1 Combinaison des différentes ontologies 77
Figure 3.2 Différentes architectures d'intégration à base d'ontologie proposées par Wache et al. 83
Figure 3.3 Architecture d'une base de données à base ontologique selon le modèle OntoDB 89
Figure 4.1 Architecture Hybride d'Intégration de Bases de Données à Base d'Ontologie à priori d'Ontologie « HYBOnto » 102
Figure 4.2 Architecture améliorée du système HYBOnto 113
Figure 5.1 Exemple d'une intégration de BDBOs par *FragOnto* 125
Figure 5.2 Exemple d'une intégration de BDBO par ProjOnto 128
Figure 5.3 Exemple d'une intégration BDBO par HybProjExtendOnto 131

Figure 5.4 Exemple d'intégration de la partie Ontologie de des BDBO par HybExtendOnto.....134
Figure 6.1 Architecture d'une base de données à base ontologique selon le modèle OntoDB.....140
Figure 6.2 Interface d'insertion d'ontologie locale « exemple des classes ».....142
Figure 6.3 Exemple du choix conceptuel des tables de la source de données locale.....143
Figure 6.4 Schéma de l'ontologie CIHO.....146
Figure 6.5 Fragment de l'ontologie partagée « HOnto ».....147
Figure 6.6 Interface utilisateur (Moteur de Recherche Sémantique).....150
Figure 6.7 Exemple de résultats de recherche.....153

LISTE DES TABLEAUX

Tableau 1.1 Taxonomie de conflits – Parent et Spaccapietra – 29
Tableau 1.2 Taxonomie de conflits 30
Tableau 1.3 Comparaison entre les différentes approches d'intégration 49
Tableau 2.1 Avantages et inconvénients des différentes approches d'intégrations de données à base d'ontologie 83

INTRODUCTION GENERALE

1. Introduction

Le besoin d'interroger différentes sources d'informations disponibles dans différents systèmes d'information autonomes et hétérogènes rend indispensable l'utilisation des systèmes d'intégrations de données qui permettent à l'utilisateur une vue uniforme et une interrogation transparente des informations.

Ces systèmes s'avèrent très utiles dans le domaine de santé du fait que l'activité des hôpitaux (administrative et médicale) génère un grand nombre de données et d'informations de natures diverses disponibles de plus en plus sur le Web, ces données peuvent être stockées de façons totalement structurée ou pas du tout structurée (fichiers textes par exemple) ce qui nous emmène aux problèmes de redondances et d'incohérences d'informations d'où le problème d'intégration de sources de données hétérogènes et autonomes.

Plusieurs approches d'intégration de sources de données hétérogènes et autonomes ont été proposées, en l'occurrence les systèmes multi-bases [01], les systèmes fédérés [02], les médiateurs [03], les entrepôts de données [04], les systèmes pair-à-pair [05] et aussi les approches semi-matérialisées (hybrides) [06]. Toutes ces approches visent à offrir aux utilisateurs une vue unique sur les différentes sources qu'ils veulent interroger sans toucher toutefois à leurs autonomies.

Nous nous intéressons à l'approche hybride ou semi-matérialisée, cette approche se compose d'un entrepôt de données et d'un médiateur (d'où son

appellation), dans cette approche on a recours aux experts du domaine pour le choix des données mises dans la partie matérialisée et les données consultées directement à partir des sources locales ce qui laisse le concepteur décide de la structure de l'entrepôt de données ainsi que le médiateur, on parle dans ce cas d'une intégration manuelle des sources. Dans ce cas, une modification de la structure d'une source locale ou l'intégration d'une nouvelle source s'avère trop lourde car elle implique généralement l'intervention des experts et une mise à jour de l'architecture globale du système, afin de résoudre ce problème nous proposons une intégration automatiques des sources de données selon un ensemble de critères, ce qui rend le système plus souple et plus rapide à interagir lors des mise à jours des sources locales ainsi que l'ajout de nouvelles sources à la longue.

Le choix d'une architecture hybride est fondé sur les avantages qu'elle apporte, d'un coté elle offre une plus grande flexibilité de rafraîchissement des données qu'une approche entièrement matérialisée du moment que ce n'est qu'une partie des données qui ont besoin d'être tout le temps mis à jour au niveau de l'entrepôt donc moins de temps pris pour les mises à jour, de l'autre coté elle apporte une meilleure optimisation des requêtes et un taux de réponse plus rapide que les méthodes entièrement virtuelles concernant la partie des données regroupées dans au sein de l'entrepôt [06].

De nombreux systèmes d'intégration ont été proposés, reste que leur grande difficulté est l'interprétation automatique de la signification, la sémantique des données hétérogènes et autonomes ce qui représente un réel défi, pour cela, plusieurs travaux sur l'intégration de données basant sur les ontologies ont été reportés ([06], [07], [08]), et particulièrement dans le domaine biomédical ([09], [10]), les ontologies sont utilisées dans ces travaux pour représenter le schéma global et/ou les schémas locaux du système d'intégration.

Nous nous intéressons plus particulièrement aux sources de données à base ontologiques pour résoudre le problème d'interopérabilité sémantique entre les sources et le système, et entre le système en lui-même et l'utilisateur, nous proposons donc *une Intégration Automatiques des Données à Base d'Ontologie au sein d'une Architecture Hybride*. Cette approche permettra d'obtenir d'une part

une mise en place modulaire et totalement automatique de la plateforme hybride ainsi qu'une intégration automatique des sources de données locales au sein de cette architecture, d'autre part, une sélection plus pertinente des services à interroger, et des requêtes cohérentes (dont les résultats, issus des différentes sources, sont systématiquement regroupés, pour fournir une réponse globale à la requête).

2. Intégration des données

Dans un système d'intégration de données, les données de différentes sources sont intégrées dans un schéma intégré global qui répond aux besoins des utilisateurs et qui est contrôlé par un système de gestion de données (généralement un SGBD). Toutes les hétérogénéités sont cachées pour l'utilisateur, qui interroge le schéma global comme un simple schéma de base de données.

3. Correspondance entre les Schémas

Pour interroger les données de plusieurs sources locales par l'intermédiaire d'un schéma global, ce schéma doit être relié aux schémas des sources locales. Différentes méthodes ont été développées pour connecter plusieurs schémas locaux à un schéma global: Global-as-View (GAV), Local-as-View (LAV) et Both-as-View (BAV) sont les plus importants. Dans une méthode de correspondance GAV, le schéma global est défini comme vue des schémas des sources sous-jacentes. Dans une méthode LAV, le schéma de chaque source locale est défini comme une vue du schéma global, et dans une méthode BAV, ces deux méthodes sont combinées. Dans la méthode BAV, nous pouvons obtenir un schéma local des relations du schéma global et vice-versa.

3.1. Intégration Matérialisée de Données

Le schéma global d'un système d'intégration de données peut être entièrement matérialisé. Cela signifie qu'une nouvelle base est développée via un système de gestion des données et une copie de toutes les données qui correspondent au schéma global est enregistrée dans cette base. Des données des sources locales sont extraites, transformées et chargées dans cette base. L'évaluation de requêtes

dans une telle approche est semblable à celle d'une base de données simple et nous avons accès à un langage d'interrogation puissant et à l'optimisation de requêtes. Cependant, on ne peut accéder directement aux données des sources locales et la base de données doit être périodiquement mise à jour. Par conséquent, il y a toujours un délai pour accéder aux dernières données mises à jour.

3.2. Intégration Virtuelle de Données

Le schéma global d'un système d'intégration de données peut être virtuel. Dans ce cas, toutes les données demeurent dans les sources locales et on y accède par l'intermédiaire d'une infrastructure intermédiaire, généralement appelée un médiateur, contenant le schéma global. Ce médiateur décompose ou reformule une requête d'utilisateur sur un ensemble de sources locales. Il recompose les réponses partielles dans une réponse pour l'utilisateur. Les données et les langages d'interrogation des sources locales peuvent être différents de ceux du logiciel personnalisé. Les wrappers (adaptateurs) contiennent des correspondances entre les schémas, traduisent les données et les questions entre les sources et le médiateur. Avec cette approche, les utilisateurs ont un accès en ligne aux données des sources locales, mais le traitement de requête est complexe et coûteux.

3.3. Approche Hybride pour l'Intégration de Données

Une troisième solution possible pour l'intégration de données est le développement d'un schéma intégré partiellement matérialisé. Une approche d'intégration de données qui emploie un tel schéma global s'appelle une approche hybride. Les approches entièrement matérialisées et entièrement virtuelles d'intégration de données obéissent à des priorités différentes. Dans une approche entièrement matérialisée, la priorité principale est le temps de réponse aux requêtes, et dans l'intégration entièrement virtuelle de données, la fraîcheur de données est prioritaire.

Cependant, dans beaucoup de scénarios d'intégration de données, différentes priorités peuvent être associées aux différentes données, et un compromis entre

le temps de réponse aux requêtes et la fraîcheur de données peut être préféré à la satisfaction d'une seule de ces deux questions. Une approche souple qui permet à quelques données d'être matérialisées et à d'autres données d'être virtuelles permet de satisfaire les deux objectifs. Dans les approches hybrides existantes, la vue globale est divisée en une partie matérialisée et une partie virtuelle. Certains objets ou relations sont choisis pour être matérialisés tandis que d'autres restent dans les sources locales et seront extraits au moment de la requête.

3.4. Utilisation d'ontologie dans l'intégration de données

Un système d'intégration de données basée sur l'ontologie est un système qui offre une vue conceptuelle sur des sources d'information préexistantes. Dans ce cas, les ontologies sont utilisées pour: (i) la représentation de la sémantique des données qui vise à interpréter le sens de chaque source en l'associant `a une ontologie locale. (ii) l'intégration sémantique qui vise à intégrer les ontologies des sources. Elle consiste à établir les relations sémantiques (équivalence, subsomption) entre les concepts des ontologies. (iii) l'intégration de données qui vise à peupler les données dans un entrepôt pour les systèmes matérialisés ou à construire des interfaces de requêtes pour les systèmes virtuels qui fournissent une vue unique sur les données.

3.5. Traitement des Requêtes

Le traitement efficace des requêtes est l'un des défis les plus importants dans l'intégration de données. En raison de la nature dynamique du contexte d'intégration de données, de nouveaux défis surgissent pour optimiser le traitement des requêtes. Au niveau du médiateur, l'optimiseur peut ne pas avoir assez d'information pour décider d'un bon plan et, en outre, au moment de l'exécution, une source peut ne pas répondre exactement comme prévu au moment de l'optimisation [11].

L'optimisation sémantique des requêtes est l'une des manières les plus efficaces de réaliser l'optimisation de requête dans le contexte de l'intégration de données. Dans une base de données simple, l'optimisation de requête est dite

sémantique si la transformation de la requête se fonde sur la connaissance sémantique liée au schéma conceptuel de la base de données.

Dans un système d'intégration de données, deux niveaux d'optimisation de requête peuvent être considérés: l'optimisation globale pour le plan de requête et l'optimisation locale pour les sous-requêtes qui cherchent les données dans les différentes sources [12]. La restriction de l'espace de recherche pour une requête, en utilisant la connaissance sémantique au niveau du médiateur, est une solution pour fournir une optimisation globale pour le traitement de la requête. Un des objectifs principaux de cette thèse est de développer une approche d'intégration de données qui se concentre dans son architecture sur la performance du traitement des requêtes. Cette architecture étend un mécanisme sémantique d'optimisation de requête d'un système de bases de données à une plateforme d'intégration de données. Nous décrivons maintenant notre contribution.

4. Domaine d'application et objectif

Notre domaine d'application cible est les données médicales, il s'agit, d'une part, de pouvoir rechercher de façon entièrement automatique n'importe quelle information médicale concernant n'importe quel patient (antécédents, traitements, médecins traitants, hospitalisation…), de n'importe quel médecin (les jours de consultations, bureau, dates d'interventions…). Il s'agit d'autre part, de pouvoir intégrer automatiquement les données de différents patients, médecins, staff médico-administratif au sein d'une seule interface utilisatrice, en offrant, sans aucune programmation, des possibilités d'accès ergonomiques

De nombreux systèmes d'intégration ont été proposés, reste que leur grande difficulté est l'interprétation automatique de la signification, la sémantique des données hétérogènes et autonomes.

L'intégration des données matérialisée (Entrepôt de données) ou virtuelles (médiateur) sont des solutions proposées pour permettre la gestion du problème d'hétérogénéité structurelle et sémantique des données et avec l'avènement des ontologies, un progrès important a pu être réalisé dans l'automatisation du processus d'intégration de sources hétérogènes grâce à la représentation explicite de la signification des données.

Les approches entièrement matérialisées et entièrement virtuelles d'intégration de données obéissent à des priorités différentes. Dans une approche entièrement matérialisée, la priorité principale est le temps de réponse aux requêtes, et dans l'intégration entièrement virtuelle de données, la fraîcheur de données est prioritaire.

Cependant, dans beaucoup de scénarios d'intégration de données, comme dans le domaine hospitalier, différentes priorités peuvent être associées aux différentes données, et un compromis entre le temps de réponse aux requêtes et la fraîcheur de données peut être préféré à la satisfaction d'une seule de ces deux questions.

Une approche souple qui permet à quelques données d'être matérialisées et à d'autres données d'être virtuelles satisfait les deux objectifs, une telle approche est dite : approche hybride où la vue globale est divisée en une partie matérialisée et une partie virtuelle. Certains objets ou relations sont choisis pour être matérialisés et d'autres restent dans les sources locales et seront extraits au moment de la requête, et pour remédier au problème d'hétérogénéité de données et de sémantique, on a opté d'utiliser une ontologie conceptuelle médicale.

L'objectif de ce travail consiste à proposer une *approche hybride basée sur les ontologies pour l'intégration de sources de données hétérogènes* au sein des hôpitaux. Cette approche permettra d'obtenir une sélection plus pertinente des services (pair) à interroger, et des requêtes cohérentes (dont les résultats, issus des différentes sources, sont systématiquement regroupés, pour fournir une réponse globale à la requête), et d'intégrer les sources de données décisionnelles et les vues matérialisées au sein d'un *Entrepôt de Données à Base Ontologique*.

Ce travail s'inscrit dans le contexte d'une approche d'intégration sémantique a priori, et est associé à trois hypothèses :

- il existe une ontologie de domaine recouvrant la totalité des termes consensuels, et
- les administrateurs (fournisseurs de données) des sources de données veulent communiquer entre eux, mais

- les besoins sont différents et ils souhaitent une grande autonomie schématique.

Dans un tel contexte, l'approche d'intégration que nous proposons est basée sur les deux principes suivants :

1- chaque base de données contient outre ces données, son schéma et l'ontologie (locale) qui en définit le sens. Une telle source est appelée base de données à base ontologique: BDBO;

2- chaque ontologie locale s'articule a priori avec une (ou des) ontologie(s) partagée(s) de domaine. Elle référence « autant que cela est possible » cette ontologie partagée.

5. Contribution

Ce présent travail consiste à proposer un modèle d'architecture hybride pour l'intégration de sources de données hétérogènes à base d'ontologie. Notre modèle est basé sur un système modulaire qui offre un nombre d'avantages : des performances acceptables, une meilleure fiabilité, un asynchronisme, une sécurité des données et une meilleure flexibilité du modèle.

Un schéma a été adopté pour la définition de données. Ce schéma permet de prendre en compte les fils et le père d'un médiateur. Trois autres schémas ont été définis pour exporter, vers le médiateur, les informations relatives aux schémas gérés par les différentes sources, les capacités de traitement des requêtes par ces sources et les statistiques et les formules de coût d'évaluation des requêtes sur les adaptateurs. Ces informations exportées ainsi que le schéma de définitions de données sont stockés dans des métadonnées d'une façon à permettre d'interroger le système de n'importe quel point afin de s'approcher d'une architecture pair-à-pair.

6. Organisation du mémoire

Notre mémoire est partagé en deux grande parties, la partie bibliographique qui est composée en trois chapitres, le premier chapitre est consacré à l'intégration de données, où on étudiera l'histoire des approches d'intégration de

données, nous présentons brièvement quelques études dans ce domaine, le deuxième chapitre sera consacrés aux systèmes d'intégration hybrides qui sont le noyau de notre étude et où on présentera quelques connexes. Quant au troisième chapitre, il sera consacré aux ontologies, et aux bases données à base ontologiques, on essayera de synthétiser les plus grands travaux fait dans cette optique.

La deuxième partie sera divisée elle aussi en trois chapitres, dans le premier, nous présentons notre approche «Approche hybride d'intégration de données à base d'ontologie» ; et nous proposons cette architecture en développant un entrepôt de donnée à base ontologique intégrant quelques vues matérialisées des données des sources locales, le choix des données à matérialisé est effectué selon ce critère, chaque table qui existe dans plus d'une source de données sera matérialisé, le reste sera interrogé par le médiateur qui est inclus dans notre système hybride. Cette approche est développée sur la base d'un système hospitalier, dans le deuxième chapitre nous détaillerons tous les scénarii qu'on a proposés pour l'intégration des sources de données. Alors que le troisième chapitre sera consacré à l'implémentation et la validation de notre architecture.

PARTIE 1 ETAT DE L'ART

Il existe un corps d'information important et croissant, sous la forme de sources autonomes et hétérogènes telles que des documents, des sources des données et des systèmes de gestion de bases de données, qui peuvent modéliser les mêmes univers de discours.

Beaucoup d'organismes et d'individus doivent intégrer ces sources pour créer un système unifié et pour manipuler les données unifiées. Les entreprises possédant des sources multiples de données profitent de l'intégration de leurs sources de données. Selon [13], 79 % des organismes ont plus de deux sources et 25 % plus de quinze.

L'objectif d'un système d'intégration de données est de développer une interface homogène pour que les utilisateurs interrogent plusieurs sources hétérogènes et autonomes. Établir une interface homogène soulève beaucoup de défis parmi lesquels l'hétérogénéité des sources, la fragmentation des données, le traitement et l'optimisation des requêtes semblent être les plus importants.

La littérature courante offre diverses approches, chacune d'elles proposant une solution à tous ces problèmes. Il y a eu des progrès significatifs dans différents domaines, mais l'intégration de données reste difficile, à la fois en théorie et en pratique.

Dans le chapitre 1, nous discutons les approches existantes d'intégration de données et les principes d'un système d'intégration de données, en particulier la correspondance entre les schémas et l'évaluation des requêtes ainsi les

problèmes (conflits) soulevés par la livraison d'un système d'intégration de données. Le chapitre 2 est consacré à l’approche d’intégration hybride on détaillera quelques travaux menés dans cette optique. Dans le chapitre 3 nous parlerons des ontologies et de leurs rôles à résoudre les conflits sémantiques dans les systèmes d’intégration, puis nous étudierons les bases de données à base d’ontologie sur lesquelles repose notre architecture proposée.

CHAPITRE 1. INTEGRATION DE DONNEES

1.1. Introduction

La nécessité d'intégrer plusieurs sources d'information en vertu d'une interface d'interrogation unique a été estimé peu après l'adoption des bases de données dans les années soixante, mais il est particulièrement un domaine active de 1985 [14]. La première génération de bases de données et systèmes d'information a été développé pour une utilisation unique, mais avec la croissance importante des systèmes informatiques, le partage et la combinaison de ces sources est devenue inévitable. La plupart des entreprises ou organisations ont besoin de partager et de combiner leurs données. Cette nécessité a encore augmenté avec le World Wide Web. On peut trouver une grande quantité de sources de données qui contiennent des informations sur une même réalité et qui peuvent être très différents dans leurs structures et dans leur sémantique. On trouve des sources de données structurées, semi-structurées ou non structurées sur le Web, et les intégrant a conduit à de nouveaux défis.

L'objectif de l'intégration des informations est de permettre le développement rapide de nouvelles applications nécessitant des informations provenant de sources multiples » [13]. Cette intégration peut se faire à plusieurs niveaux tels que les données, middleware, applications, par conséquent, plusieurs technologies sont proposées pour intégrer ces sources. La nouvelle génération de systèmes d'intégration de données propose des solutions pour intégrer les données au niveau sémantique. Il fournit aux utilisateurs un langage de requête de haut niveau.

1.2. Problèmes d'intégration de données

L'intégration de sources de données autonomes soulève plusieurs défis. Création d'une interface appropriée pour accéder à plusieurs sources de données, mise en correspondances entre les vues intégrées et les schémas de sources, interrogation des différentes sources est l'unes des plus importants défis. Dans ce qui suit, nous décrivons trois problèmes fondamentaux soulevés face à l'intégration des sources de données.

1.2.1. Problème d'Autonomie

Les sources de données qui participent à un scénario d'intégration sont autonomes. Cela signifie que les parties de l'organisation qui gèrent et contrôlent ces sources sont indépendantes. Souvent, les administrateurs des sources de données ne permettent de partager les données d'autres systèmes que si le contrôle est conservé. En conséquence, pour construire un système d'intégration de données, l'aspect de l'autonomie doit être pris en considération [15].

Les différents aspects de l'autonomie sont [02]:

- Autonomie de conception: chaque source d'information locale a son propre modèle de données, langage de requête, schéma conceptuel, etc.

- Autonomie de communication: chaque source d'information locale décide du moment où elle communique avec le système d'intégration.

- Autonomie d'exécution : ce n'est pas la gestion du système d'intégration qui contrôle transactions, et les opérations locales ou externes des sources locales. En d'autres termes, une source de données dans un scénario d'intégration de données n'a pas besoin d'informer d'autres sources de ses transactions et de ses opérations.

- Autonomie d'association: les sources locales ont la possibilité d'associer ou de dissocier de l'intégration du système, c'est à dire décider lesquelles de ses données et comment leurs fonctionnalités participent au système d'intégration.

1.2.2. Problème d’Interdépendance

L'interdépendance implique que les données de différentes sources dépendent les unes des autres. Différentes sources de données peuvent se chevaucher et contiennent les mêmes objets du monde réel dans différents formats. Il peut aussi avoir des dépendances entre les fonctionnalités des différentes sources de données. La gestion des données interdépendantes applique l'utilisation de contraintes de cohérence dans un système d'intégration.

1.2.3. Problème d’Hétérogénéité

Un des problèmes les plus complexes lors de l'intégration de plusieurs sources de données autonomes est l'hétérogénéité de ces sources. Les systèmes d'information qui doivent être intégrés peuvent être développés dans des environnements différents et avec différents schémas de conception et différentes définitions de données. Cela conduit à une hétérogénéité à différents niveaux du système.

Selon [15] un système d'information est homogène si le même logiciel gère les données sur tous les sites, les données ont le même format et structure (même modèle de données) et appartiennent au même univers de discours. Au contraire, un système d'information est hétérogène s’il n'a pas accès à toutes les caractéristiques d'un système homogène

Richard Hull [16] classe l’hétérogénéité en deux perspectives: structurelle et sémantique. La perspective structurelle contient du matériel, modèle de données, SGBD et API qui sont pris en charge. Réseau de communication protocoles et les standards tels que ODBC, JDBC et CORBA peuvent aider à gérer ces types d'hétérogénéité. Quant à l’hétérogénéité sémantique, elle reporte à la différence dans la signification et l'utilisation des données [15]. Elle caractérise les différences de signification, l'interprétation ou de l'utilisation de la même donnée [02]. Pour développer un système d'intégration de données, l'hétérogénéité sémantique doit être gérée. En règle générale, l'hétérogénéité sémantique est classée dans l'hétérogénéité des schémas (ou de l'hétérogénéité schématique) et de l'hétérogénéité dans les données.

L'hétérogénéité entre les deux schémas différents est significative lorsque les données sont sémantiquement liées. Les conflits entre les schémas des sources de données apparaissent lorsque les concepts équivalents sont représentés différemment dans les sources. Ces conflits sont liés à des noms, des types de données, attributs, etc.

Plusieurs taxonomies des conflits ont été proposées [17]:

1. Conflits de classification : ils apparaissent lorsque les types en correspondance décrivent des ensembles différents, mais sémantiquement liés, du monde réel. Par exemple, deux sources médicales peuvent contenir chacune une entité nommée Médecin avec pour extension, pour la première, tous les médecins de l'hôpital, et pour la seconde uniquement les médecins spécialistes. La solution standard pour ce genre de conflit est d'inclure dans le schéma intègre la hiérarchie de généralisation/spécialisation appropriée.

2. Conflits descriptifs : ils surviennent des qu'il y a une différence entre les propriétés des types en correspondance (les types d'objets peuvent différer selon leurs : noms, clés, attributs, les attributs peuvent aussi différer selon leur noms, structures, etc.).

3. Conflits structurels : un conflit structurel survient lorsque les éléments en correspondance sont décrits par des concepts de niveaux de représentation différents ou soumis a des contraintes différentes. Par exemple, une classe d'objet, et un attribut, ou un type d'entité et un type d'association. Par exemple, adresse qui est un attribut d'une table relationnelle dans un schéma A peut correspondre a une table dans un schéma B.

4. Conflits d'hétérogénéité des modèles de données : nous avons déjà souligne le fait que la plupart des travaux sur l'intégration ne considèrent que des schémas exprimés sur le même modèle. Les schémas qui ne sont pas exprimés dans ce modèle doivent au préalable être traduits lors d'une phase de prétraitement.

5. Conflits données/métadonnées : surviennent lorsqu'une donnée dans une base est en correspondance avec une métadonnée (le nom d'un type) dans le schéma d'une autre base.

6. Conflits de données : surviennent au niveau des instances, lorsque des occurrences en correspondance ont des valeurs en conflit pour des attributs en correspondance. Les conflits de données sont détectes lors de l'exécution d'une requête de l'utilisateur. Ce type de conflits a fait l'objet d'une attention particulière de la part de la communauté des bases de données, avec les travaux sur la résolution d'entités [18].

Le tableau suivant résume les différentes taxonomies de conflits vues et qui était proposées par Sheth et Kashyap [19] et qui ont été reprises par Parent et Spaccapietra dans cette version plus simple détaillée précédemment [17]

Conflits	Description
Conflits Classification	Les populations du monde réel représentées par les deux types sont différentes
Conflits Description	Les types ont des ensembles différents de propriétés et/ou leurs propriétés sont représentées de manière différente
Conflits Structure	Les concepts utilisés pour décrire les types sont différents
Conflits Hétérogénéité	Les modèles de données utilisés sont différents
Conflits Méta données	La correspondance relie un type à un méta-type (un ensemble de noms de types) Données des instances en correspondance ont des valeurs différentes pour des propriétés en correspondance
Conflits Données	Des instances en correspondance ont des valeurs différentes pour des propriétés en correspondance.

Tableau 0.1 Taxonomie de conflits – Parent et Spaccapietra – [17]

Goh et ses collègues [20] proposent une autre classification des conflits pouvant apparaitre lors de la mise en correspondance entre schémas. Ce sont les

conflits de nommage (naming), les conflits de graduation (scaling), les conflits de confusion (confounding conflicts) et les conflits de représentation.

Conflits	Description
Conflits de nommage	Les différents schémas utilisent des noms différents pour représenter le même concept
Conflits de graduation	Les concepts semblent avoir la même signification dans deux schémas mais sont différents à cause de leur contexte
Conflits de confusion	Liés à la manière de coder la valeur d'un concept du monde réel dans un système de mesure.
Conflits de représentation	Deux schémas sources décrivent le même concept de manière différente

Tableau 0.2 Taxonomie de conflits [20]

1. Les conflits de nommage : se produisent lors de l'attribution des noms dans des schémas qui diffèrent de manière significative. Les cas les plus fréquents sont les cas de présence de synonymes et d'homonymes.

2. Les conflits de graduation : apparaissent lorsque différents systèmes de graduation sont utilises pour mesurer une valeur. On peut citer en exemple la mesure de la température exemples : - Degrés Celsius ou Fahrenheit.- Dollar $ ou l'Euro €.

3. Les conflits de confusion : se produisent lorsque les concepts paraissent avoir la même signification mais différent en réalité. Ce type de confusion peut être cause par des contextes temporels différents par exemple. On a, par exemple, le poids d'une personne dépend de la date ou elle se pèse.

4. Les conflits de représentation : se produisent quand deux schémas sources décrivent le même concept de manière différente. Par exemple, dans une source l'adresse peut être désignée par une chaine de caractères tandis que dans une autre, l'adresse est une structure composée du numéro et du nom de la rue, du code postal et de la ville.

Pour illustrer quelques conflits présentés ci-dessus, considérons l'exempe suivant : soient deux sources de données, source 1 et source 2, modélisant les patients (voir figure 1). La Source 1 utilise cinq attributs pour décrire un malade, tandis que la Source 2 en utilise seulement quatre (l'attribut Sexe n'est pas défini).

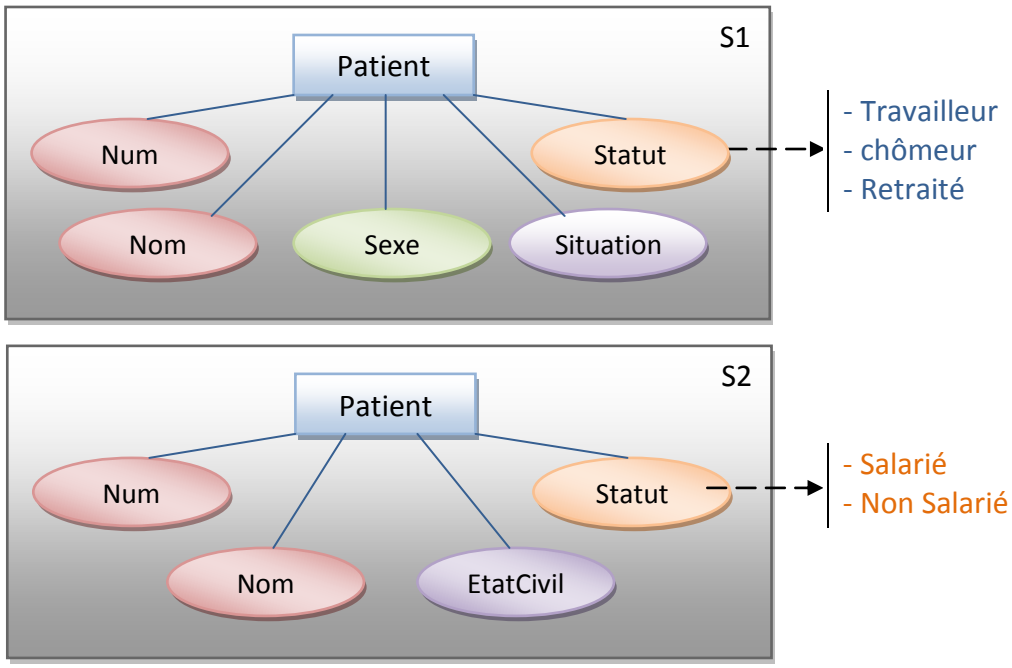

Figure 0.1. Exemple de source hétérogène

1.2.4. Discussion

Pour définir la situation familiale de chaque malade, la source 1 utilise la propriété *Situation* tandis que la deuxième utilise *EtatCivil* Conflit structurels.

Dans la source 1, le domaine de l'attribut *Statut* est un type énuméré d'entiers (1 : Travailleur, 2 : Chômeur, 3 : Retraité) tandis que le domaine de l'attribut Statut de la source 2 est un type énuméré de caractères (S : Salarié, N : Non salarié) Conflit de noms.

L'attribut Statut dans les deux sources a le même nom (Statut); mais avec deux sens différents. Source1 statut signifie la catégorie du patient, alors que dans la Source2, signifie la position du patient Conflit de noms.

La diversification des sources de données, a conduit tout naturellement à l'idée d'offrir à 1'utilisateur un système permettant d'avoir une vue centralisée uniforme des données. De cet état de fait, les chercheurs se sont investis dans l'intégration des différentes sources de données qui est devenue un axe de recherche important.

L'objectif de cette idée est de donner aux utilisateurs une interface uniforme pour les différentes sources de données, dite système d'intégration de données. Ainsi, les utilisateurs vont se focaliser pour spécifier ce qu'ils veulent et non pas perdre du temps en réfléchissant a comment obtenir la réponse. Ils n'ont pas à chercher les sources adéquates, à interagir avec chaque source en isolation en utilisant une interface particulière et combiner les différents résultats obtenus.

Pour une revue complète des différentes approches utilisées dans le monde des bases de données pour traiter le problème de l'hétérogénéité entre schémas, nous renvoyons le lecteur a l'état de l'art fait par Rahm et Bernstein [21] sur les différentes approches de correspondance automatique de schémas, et à celui effectué par Kalfoglou et ses collègues [22] sur la correspondance entre ontologies. Les problèmes qui apparaissent lors de la mise en correspondance de schémas peuvent se retrouver lors de la mise en correspondance d'ontologies. Kalfoglou et ses collègues présentent les caractéristiques de projets relatifs a la mise en correspondance entre ontologies. Pinto et ses collègues [23] fournissent aussi une classification des différentes approches pour l'intégration d'ontologies.

1.3. Systèmes d'intégration de données

Les données accessibles sur le réseau peuvent prendre différentes formes et ont chacune leurs spécificités propres, ce qui nous mène au problème de combiner des sources de données hétérogènes et de les interroger via une seule interface de requête qui ne date pas d'aujourd'hui. Depuis l'arrivée et l'adoption des bases de données dans les années soixante, leur partage et leur combinaison sont naturellement devenus indispensables. Cette combinaison peut être effectuée de différentes façons et à différents niveaux de l'architecture du système.

1.3.1. Définition

« *Un système d'intégration de données fournit une vue unifiée de données provenant de sources multiples et hétérogènes. Il permet d'accéder à ces données à travers d'une interface uniforme, sans se soucier de leur structure ni de leur localisation* » [24].

Formellement, un système d'intégration de données est un triplet I : <G, S, M>, où : G représente le schéma global (défini sur un alphabet AG) modélisant le schéma intégré, S est l'ensemble des schémas des sources (définis sur un alphabet AS) décrivant la structure des sources participantes au processus d'intégration, M est une correspondance entre G et S qui établit la connexion entre les éléments du schéma global et ceux des sources.

Pour interroger le système intégré, les requêtes sont exprimées en termes de constructions du schéma global G. Par raisonnement, tout système d'intégration de données doit considérer l'intégration à deux niveaux:

(1) le niveau schéma, qui consiste à consolider tous les schémas des sources en un seul schéma global, ou schéma de médiation, qui sera utilisé pour supporter les requêtes,

(2) le niveau des données (population du schéma intégré) [24].

Un système d'intégration se compose de deux parties [25] :

- Une partie (1) externe et correspond aux utilisateurs du système intègre (décideurs) ou autres systèmes.
- Une partie (2) interne et comprend des sources d'informations et une interface uniforme qui permet a la partie externe d'interroger d'une manière transparente les sources de données, comme s'il n'y avait qu'une source unique.

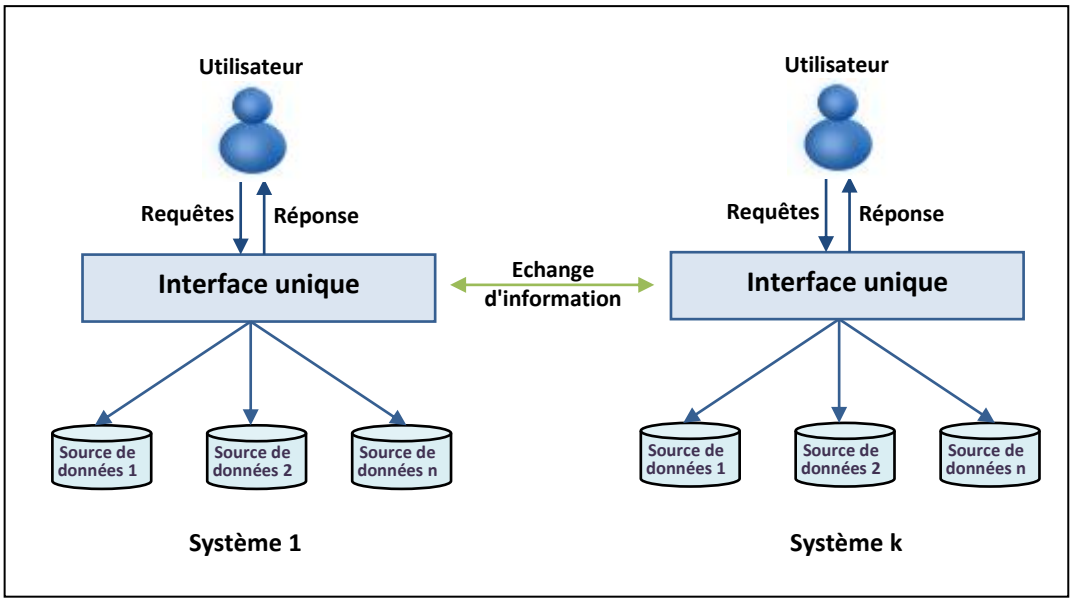

Figure 0.2 Système d'intégration de données [26]

1.3.2. Processus d'intégration

Etant donné un ensemble de sources de données hétérogènes $\{S_1, S_2, \ldots, S_n\}$, le problème d'intégration consiste à construire un schéma intégré (ou schéma global) qui sera utilisé comme interface d'accès aux sources de données. La construction du schéma global à partir des schémas locaux est une tâche difficile. Cette difficulté est liée au fait que les sources stockent différentes sortes de données, en différents formats, ayant différentes significations et associées aux différents noms

Il existe plusieurs méthodologies permettant l'intégration de bases de données classiques voir [25]. Nous présentons le processus de [17].

Le processus d'intégration est ainsi décomposé en trois phases distinctes :

- La pré-intégration : Cette phase vise à préparer l'intégration des schémas en les rendant plus homogènes. Elle consiste principalement à traduire les schémas initiaux dans un modèle de données commun (réduction de l'hétérogénéité syntaxique). Elle s'attache également à enrichir leur sémantique.

- L'identification des correspondances : Durant cette phase, les correspondances entre les éléments des schémas source sont détectées et formalisées de même que les différents conflits.

- L'intégration : Cette phase finale produit le schéma intégré et fournit les règles de traduction permettant de passer des schémas source au schéma intégré et inversement « mapping ».

1.3.3. Tache d'un système d'intégration

On peut distinguer quatre taches principales d'un système d'intégration. Les deux premières concernent la traduction de données provenant de sources différentes et résolvent le problème de l'hétérogénéité physique/logique des sources en fournissant une interface d'accès uniforme. Les deux dernières sont des taches d'intégration sémantique et résolvent le problème de l'hétérogénéité sémantique en reliant chaque source au schéma global. Ces quatre tâches sont décrites ci-après :

Transformation de données : Un exemple typique de cette tache est la transformation de données relationnelles en XML et inversement. Les problèmes importants qui doivent être résolus a ce niveau sont la perte d'information, la taille des données générées et la performance des traitements sur ces données. L'exploitation de la structure de données à transformer (types, schémas) joue un rôle crucial dans ce contexte.

Traduction de requêtes : La traduction de requêtes d'un langage (exemple : XQuery) en un autre langage (exemple : SQL) est liée au problème de transformation de données. Elle doit également prendre en compte la puissance d'expression du langage cible et nécessite souvent des extensions spécifiques afin d'obtenir la puissance d'expression du langage source.

Réécriture de requêtes : Cette tache est différente de la tâche précédente et généralement plus complexe car elle doit prendre en compte l'hétérogénéité structurelle et sémantique entre les schémas. Elle joue un rôle primordial dans l'intégration de données sur le Web.

Fusion de données : La fusion de données essaye de répondre au problème de la représentation multiple d'une même information dans différentes sources. Elle fait partie de la tâche de réécriture de requêtes, mais se pose également dans un environnement ou les données sont matérialisées.

Cette séparation fonctionnelle se traduit souvent par une séparation physique dans la forme d'un système d'intégration relie a un ensemble de traducteurs (adaptateurs). Néanmoins il est possible de déléguer des fonctions de traduction au système d'intégration ou, inversement, des fonctions d'intégration aux traducteurs. Par exemple, un traducteur plus intelligent peut préserver des informations structurelles sur les données sources afin de faciliter la réécriture et l'optimisation de requêtes. Inversement, le système d'intégration peut être conscient des langages de requêtes des sources et traduire directement les requêtes locales avant de les envoyer aux sources. Tous ces services font partie d'autres services plus généraux comme le traitement des requêtes et le stockage de (méta-) données.

1.3.4. Classification des approches d'intégration de données

Plusieurs approches et systèmes d'intégration ont été proposés dans la littérature, souvent classifiés, à travers des critères différents [27], une vue générale est présentée dans [28]. Bellatrache et al. [29] a proposé une classification des systèmes d'intégration de données en se basant sur trois critères orthogonaux :

1- Localisation de données intégrées.
2- Nature de correspondance (mapping).
3- L'automaticité du processus d'intégration.

Dans [30], la classification selon ses trois critères a été proposée pour faciliter la compréhension des caractéristiques, des avantages et insuffisances des systèmes existants. Nous allons suivre cette classification pour présenter les systèmes d'intégration existants.

1.3.4.1. Localisation de données intégrées / Architecture d'intégration

L'intégration de l'information de plusieurs sources de données est effectuée sur la base des besoins des utilisateurs pour une fin donnée. Ces exigences sont représentés par une vue appelée vue intégrée. Il existe plusieurs recherches et projets commerciaux dans l'intégration de données où chacun propose une approche d'intégration. Selon la vue intégrée, ces différentes approches peuvent être classées en deux domaines principaux: matérialisée et virtua1.

Il existe également des approches hybrides quand il y a une combinaison de facteurs matérialisée et virtuel. Nous allons discuter de cette troisième catégorie dans le chapitre 3.

1.3.4.1.1. Systèmes Multi-bases

Les systèmes Multi-bases sont des systèmes dits faiblement couplés [31]. On les caractérise de cette manière car ils n'offrent pas une vision unifiée des données. Il n'existe pas de schéma global permettant un accès transparent aux différentes sources de données.

La coopération est seulement assurée par l'intermédiaire d'un langage commun : le langage multi-base de type SQL notamment [01].

L'utilisateur peut poser une requête aux différents systèmes à l'aide de ce langage commun, sans se soucier de l'hétérogénéité des systèmes sources. Ceci ne signifie pas qu'une requête nécessitant l'accès à diverses sources est exécutée en une seule fois.

L'utilisateur doit envoyer autant de requêtes qu'il y a de sources impliquées. C'est donc à l'utilisateur de relier les différentes réponses aux requêtes formulées.

Ces systèmes sont donc faiblement intégrés et gardent une grande autonomie. Les sources peuvent évoluer de manière indépendante, sans conséquence sur leur accès.

Cependant, la cohérence entre ces sources n'est pas assurée. L'hétérogénéité n'est pas traitée en amont. L'intégration est dynamique : les correspondances entre les données ne sont pas prédéfinies.

Certains systèmes offrent la possibilité de rendre les correspondances persistantes entre les sources par la création de vues Multi-bases. Les requêtes Multi-bases sont exprimées sur ces vues multi bases. C'est le cas du système MSQL [01].

1.3.4.1.2. Systèmes Fédérés

Le concept de base de données fédérée a émergé du fait que pour pouvoir alimenter un entrepôt de données, on est mené à se connecter à plusieurs sources de données qui constituent une grande base de données hétérogène. [32] définit les bases de données fédérées comme suit :

« Une base de données fédérée est une base de données repartie hétérogène constituée de sources de données de natures variées : fichiers HTML, XML ».

L'élaboration d'une base de données fédérée nécessite donc une architecture qui permet la communication entre les différentes sources de données.

Cette architecture s'articule en trois niveaux [31][33] :

- un niveau présentation formé de composants qui permettent de formuler des requêtes dans le langage de la base de données fédérée ;
- un niveau médiation formé de médiateurs qui se chargent de collecter les demandes des utilisateurs déjà collectées par les composants de présentation et de les traduire dans le langage de chaque source de données ;
- un niveau adaptation formé de composants qui permettent la communication entre une source de données et les médiateurs

[32] a proposé une implémentation de base de données fédérée en utilisant des composants de connexion aux sources de données et de communication entre les utilisateurs et les sources de données.

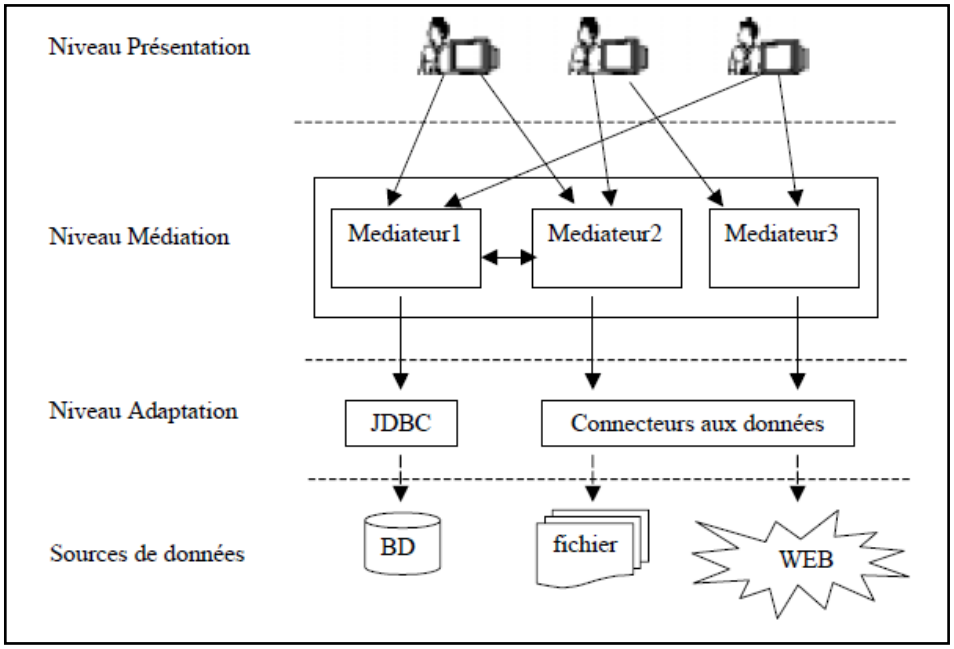

Figure 0.3 Modèle d'architecture fédérée [32, 34]

1.3.4.1.3. Système de médiation

Un *système de médiation* est un outil puissant permettant un accès simple aux différentes informations collectées de sources de données pouvant être très disparates. Il doit intégrer des données diverses afin de pouvoir offrir à l'utilisateur une vue centralisée et uniforme des données en masquant les caractéristiques spécifiques à leur localisation, méthode d'accès et formats [35].

Dans cette approche, l'intégration de données est fondée sur la définition d'un schéma global unifiant les schémas hétérogènes des sources à intégrer et sur la description homogène et abstraite, du contenu des sources par des vues. Une première proposition d'architecture médiateur a été proposée dans [03] ; celle-ci est représentée par trois couches (voir figure4) : médiateur, adaptateurs et sources. Au niveau du médiateur, le schéma global fournit un vocabulaire unique pour l'expression des requêtes des utilisateurs et pour la description des contenus des sources par un ensemble de vues abstraites sur les sources. Les adaptateurs ou wrappers (adaptateurs) traduisent les requêtes exprimées en termes du

vocabulaire du schéma global en des requêtes exprimées dans le langage des sources. De plus, les adaptateurs transforment les réponses aux requêtes en des réponses conformes au schéma global du médiateur. [03]

Globalement un médiateur offre un accès direct aux sources et se caractérise par [36]:

- approche "paresseuse", pas de matérialisation.
- migration de requêtes vers les sources.
- avantages: cohérence, données réelles.
- inconvénients: performances, traduction de requêtes, capacités des sources.

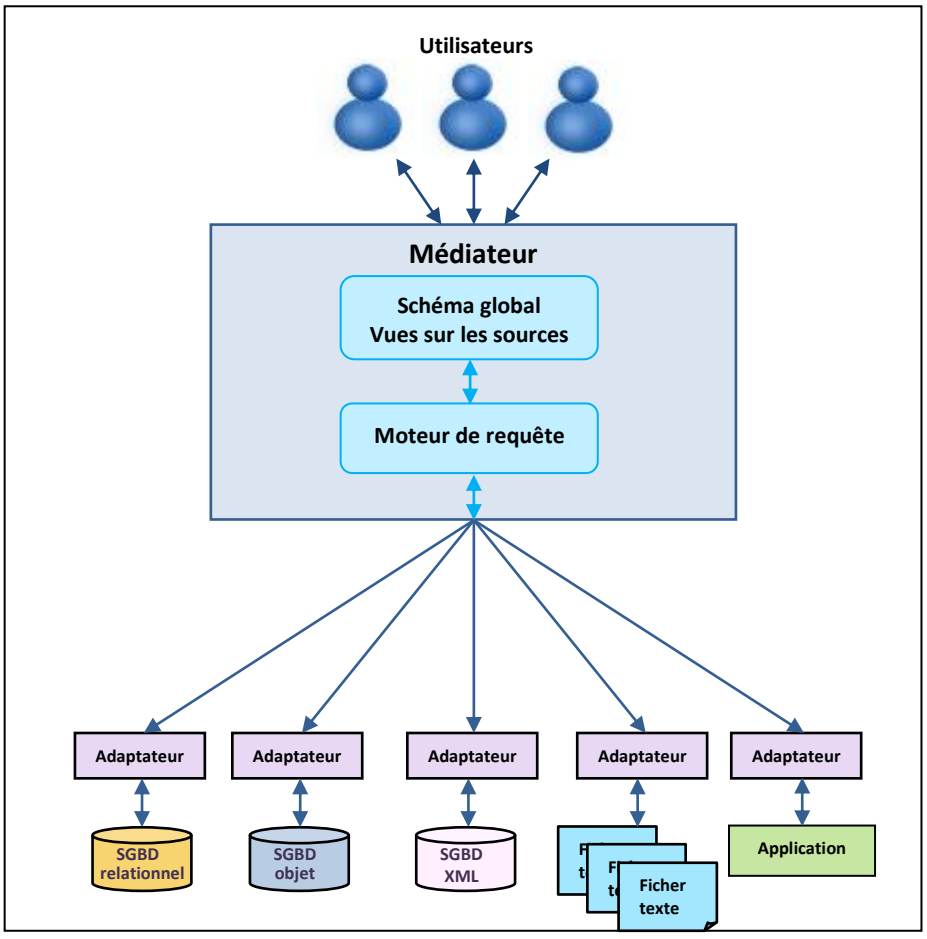

Figure 0.4 Architecture d'un système médiateur [38, 39]

1.3.4.1.4. Système Entrepôt de données

L'approche entrepôt de données consiste à voir l'intégration comme la construction d'une base de données réelle appelée entrepôt, stockant des données intégrées provenant de différentes sources. Dans le contexte d'une entreprise, les entrepôts de données sont mis en œuvre pour fournir un support pour le processus décisionnel. Un système d'intégration suivant une approche entrepôt de données est constitué de quatre niveaux, comme le montre la Figure1.6:

- ✓ Les sources d'informations : Les sources d'informations peuvent être multiples (systèmes de bases de données, systèmes de fichiers, documents HTML, bases de connaissances, etc.) et sont classées en fonction de la nature de leur participation dans le procédé du Data Warehousing.

- ✓ Extracteur/Adaptateur : Ces composants ont essentiellement deux rôles, à savoir la traduction et la détection des changements. La traduction consiste à rendre les sources d'informations adaptées au modèle de données utilisé par le système d'intégration (modèle du schéma globale). Par exemple, si la source d'information a une structure de fichiers mais que le modèle de l'entrepôt de données est relationnel, alors le wrapper / moniteur doit disposer d'une interface présentant les données de la source d'informations comme si elles étaient relationnelles. La détection (assurée par le wrapper appelé aussi traducteur) permet quant à elle le contrôle des sources afin de communiquer tout changement sur celles-ci à l'intégrateur. Une autre alternative consiste à remplacer la fonctionnalité de détection par la transmission des données des sources vers l'entrepôt de façon périodique.

- ✓ Intégrateur : La mise en place, par l'intégrateur, de l'information dans l'entrepôt de données peut inclure le filtrage de celle-ci, son résumé, ou sa fusion avec des informations issues d'autres sources. Parfois, des informations supplémentaires de la même source ou de sources différentes peuvent s'avérer nécessaires à l'intégrateur afin que la nouvelle information soit proprement intégrée dans l'entrepôt de données. La suite du travail de l'intégrateur, une fois celui ci chargé par son ensemble initial de données,

consiste à recevoir les notifications des changements des wrappers / moniteurs et à les répercuter au niveau de l'entrepôt de données. en d'autres termes, le rôle de l'intégrateur est d'assurer la maintenance de la vue matérialisée (data warehouse).

- ✓ L'entrepôt de données : est une base de données volumineuse contenant les vues matérialisées intégrées, l'entrepôt de données peut dans ce cas aider à la prise de décision, cela dépend des données intégrée.

Une approche entrepôt de données est fondée sur un schéma global de l'entrepôt fournissant une vue intégrée des sources. Une fois que le schéma de l'entrepôt est conçu, les données sont extraites à partir des sources, transformées au format de représentation des données de l'entrepôt par des extracteurs, elles sont éventuellement filtrées pour ne garder que les données pertinentes, et enfin stockées dans l'entrepôt. L'interrogation s'appuie sur des techniques classiques d'interrogation du domaine des bases de données. L'utilisateur interagit avec l'entrepôt pour une interrogation directe des données de l'entrepôt ou à travers les magasins de données soit pour effectuer de la fouille de données soit pour générer des rapports statistiques. [39]

Les principaux problèmes de cette approche sont l'extraction et l'intégration des données permettant d'assurer une bonne qualité des données, comme par exemple, l'absence de données redondantes et la fraîcheur des données. En effet, contrairement à une approche médiateur où les réponses aux requêtes proviennent directement des sources, dans le cadre d'un entrepôt les données matérialisées localement doivent être mises à jour pour assurer la fraîcheur des données. [40]

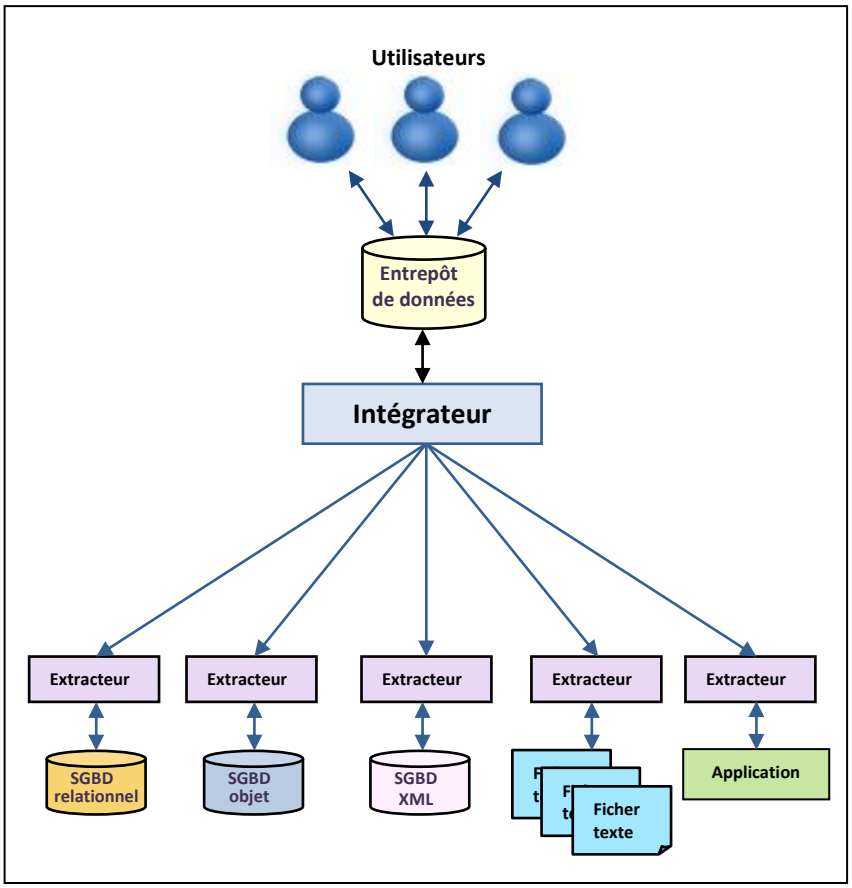

Figure 0.5 Architecture d'un entrepôt de données [38]

1.3.4.1.5. Systèmes d'intégration P2P

L'émergence de systèmes pair à pair (Peer-to-Peer) de partage de fichiers a conduit les chercheurs à considérer l'architecture P2P dans le contexte de l'intégration et le partage de données [41][42][43][44][45]. Ces systèmes d'intégration P2P suivent une approche décentralisée pour l'intégration de pairs autonomes et distribués contenant des données qui peuvent être partagées. L'objectif principal de tels systèmes est de fournir une interopérabilité sémantique entre plusieurs sources en l'absence de schéma global.

Chaque pair est interconnecté avec un certain nombre de pairs du réseau (appelés voisins) à l'aide de formules de coordination [46]. Edutella [43] [47], SomeWhere [42], PIAZZA [41] sont des exemples de travaux récents sur les systèmes P2P.

Selon Lenzerini [48], un système d'intégration de données pair-à-pair doit respecter les trois principes suivants :

- Modularité : i.e. l'autonomie des différents pairs en respectant la sémantique. En effet, puisque chaque pair est construit et géré séparément, il doit être clairement interprétable quand il est utilisé seul et dans le cas où il est en liaison avec d'autres pairs.
- Généralité : liberté de placement d'interconnexions entre les différents pairs (mappings P2P).
- Décidabilité : vérifier si les mécanismes disponibles fournissent des réponses significatives et complètes aux requêtes posées, sinon, il sera difficile d'assurer la qualité des réponses retournées par le système.

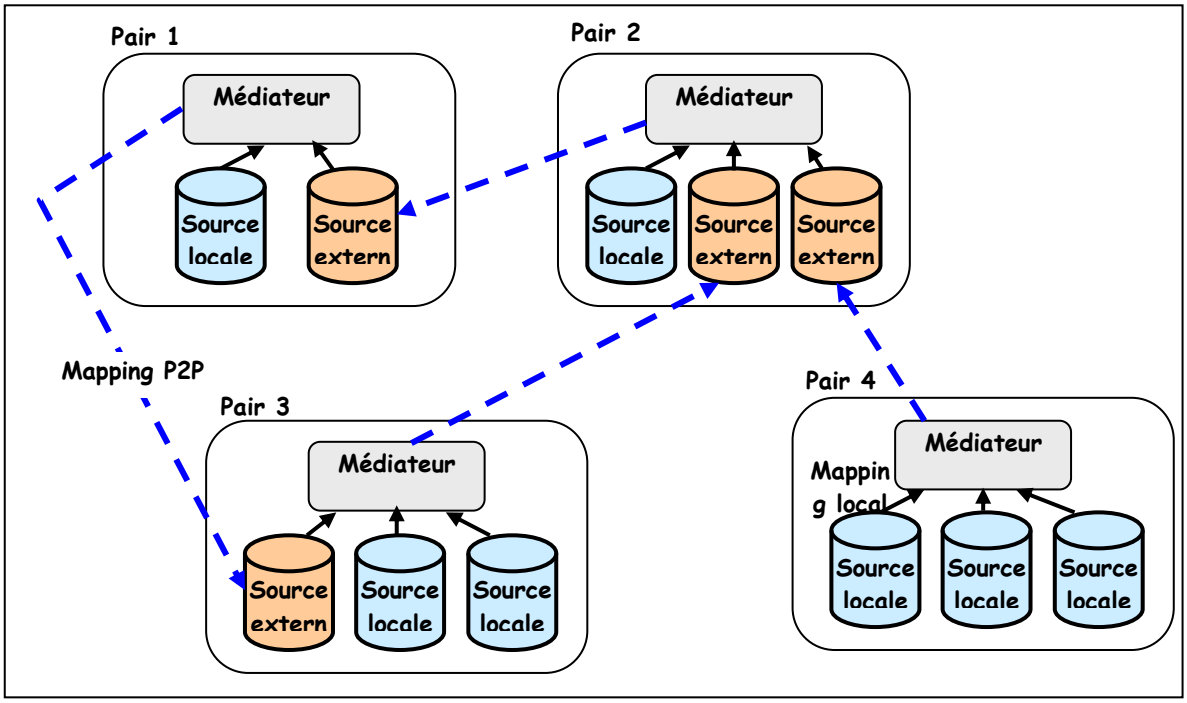

Figure 0.6 Architecture d'un système Pair-à-Pair [49]

1.3.4.1.6. Système hybride ou semi-matérialisé

Cette troisième solution d'intégration de données concerne le développement d'un schéma intégré partiellement matérialisé (combinaison entre

un entrepôt de données et un médiateur). Nous détaillerons cette approche dans le chapitre suivant.

1.3.4.1.7. Discussion

Nous venons d'exposer un bref aperçu sur les principales approches d'intégration des sources de données hétérogènes en l'occurrence les systèmes multi-bases, les bases de données fédérées, le médiateur, l'approche entrepôt de données, l'approche pair-à-pair et aussi l'approche semi-matérialisée (hybride). Toutes ces approches visent à offrir aux utilisateurs une vue unique sur les différentes sources qu'ils veulent interroger sans toucher toutefois à leurs autonomies.

Les systèmes multi-bases sont faiblement couplés et c'est à l'utilisateur d'envoyer les requêtes aux différentes sources de données et à filtrer et réorganiser les données obtenues, quant aux systèmes de médiation ou encore les systèmes de bases de données fédérées, ils apportent deux avantages : (1) une vue virtuelle sur les données et l'information est extraite directement des sources (2) l'absence d'incidence de mise à jour des données sur le système intégré. Cela permet d'éviter le problème de maintenance des données mais la construction des schémas médiateurs dans ces deux approches ainsi que les correspondances nécessite beaucoup d'intervention manuelle. L'utilisateur indique les ressemblances entre les différents schémas de sources, une fois les ressemblances trouvées, elles sont regroupées pour créer leur équivalent (relation ou attribut) dans le schéma médiateur. La deuxième limite est que la modification des schémas des sources de données engendre la régénération entière du schéma médiateur.

L'approche entrepôt de données améliore le temps d'accès aux données centralisées dans une source globale construite à partir des sources locales, les entrepôts de données ont été conçus principalement pour l'aide à la prise de décision. Ils intègrent les informations qui ont pour objectif de fournir une vue globale de l'information aux analystes et aux décideurs, mais présentent néanmoins l'inconvénient de la difficulté de l'ajout des sources et celui du coût de stockage élevé.

Les systèmes P2P est l'approche la plus attrayante en terme de flexibilité. Elle est aussi l'approche idéale pour traiter un grand nombre de sources d'information.

Cette approche présente, cependant, certains inconvénients qui sont :

- Difficulté d'effectuer des mappings de manière automatique d'un schéma à un autre, ce qui nécessite une intervention humaine.
- Les pairs qui ne sont pas directement reliés ne peuvent communiquer que par un intermédiaire qui sera par conséquent surchargé (goulot d'étranglement).

Quant à l'approche hybride, elle englobe les avantages de l'approche médiateur et celles de l'approche entrepôt de données elle offre une plus grande flexibilité de rafraîchissement des données qu'une approche entièrement matérialisée, et une meilleure optimisation des requêtes que les méthodes entièrement virtuelles.

1.3.4.2. Nature de correspondance entre schéma global et schéma local

Ce critère permet de définir les relations existant entre le schéma global et les schémas locaux. Deux principales approches ont été proposées: GAV (Global as View) et LAV (Local As View)

1.3.4.2.1. Global As View (GAV)

L'approche GAV consiste à définir de façon manuelle ou semi-automatique le schéma global en fonction des schémas des sources de données à intégrer puis à le connecter aux différentes sources. Pour cela, les prédicats du schéma global, aussi appelés relations globales, sont définis comme des vues sur les prédicats des schémas des sources à intégrer.

Comme les requêtes d'un utilisateur s'expriment en termes des prédicats du schéma global, on obtient facilement une requête en termes de schéma des sources de données intégrées, en remplaçant les prédicats du schéma global par leurs définitions.

Parmi les systèmes utilisant GAV, on peut citer TSIMMIS [50] et MOMIS [02]. En effet, dans TSIMMIS, les médiateurs sont conceptuellement représentés par des vues définies sur une ou plusieurs sources.

1.3.4.2.2. Local As View (LAV)

L'approche LAV [51] consiste à décrire le contenu des sources à intégrer en fonction du schéma médiateur. Le traitement des requêtes sur le schéma global implique leur réécriture en des requêtes utilisant les vues. La requête sur le schéma global doit être reformulée suivant les schémas des sources locales, ce qui la rend complexe.

Dans une telle approche l'ajout ou la suppression d'une relation est facile à traiter car elle n'affecte pas le schéma médiateur, vu que chaque source est définie indépendamment. Parmi les systèmes appartenant aux approches LAV, nous citons Information Manifold [41], PICSEL [06] et OBSERVER [07].

1.3.4.2.3. BAV (Both As View)

L'approche BaV (Both-As-View) [52], qui utilise des transformations incrémentales afin de lier deux schémas. L'approche BAV a été introduite dans le cadre du projet d'intégration AutoMed [53].

1.3.4.2.4. GLAV (Generalized Local As View)

Les avantages des approches LAV et GAV ont été combinés dans des approches mixtes. C'est ainsi qu'a été proposée l'approche GLAV (Generalized-Local-As-View) [54], qui utilise des règles de correspondance ayant plus d'un terme dans leur entête (l'entête peut être la combinaison d'un nombre quelconque de prédicats, contrairement à l'approche LAV ou GAV).

Dans GLAV (Generalized-Local-As-View), on dispose de vues au niveau global et local. Une correspondance entre les vues au niveau global et local est requise. Ces correspondances n'ont pas de direction et s'appliquent dans les deux sens puisque les concepts dans l'ontologie globale sont considérés comme des vues.

En fait, l'approche hybride permet d'avoir la correspondance entre les ontologies locales via le vocabulaire partagé. Donc la correspondance peut être calculée via l'ontologie globale. Dans le cas de l'approche hybride modélisée selon GLAV, nous gardons une indépendance entre les sources (permet l'ajout et la suppression de sources), et nous pouvons calculer indirectement les correspondances entre les sources. Cette caractéristique est impérative si nous voulons avoir des résultats cohérents.

1.3.4.2.5. BGLAV (BYU Global-Local-As-View)

Cette approche est venue juste après l'approche GLAV, L'approche BGLAV (Brigham Young University Global-Local-As-View) se présente comme un point de vue alternatif qui n'est ni GAV ni LAV. L'approche emploie des mapping de la source à la source cible basés sur un schéma conceptuel prédéfini de la source cible, qui est spécifié ontologiquement et indépendamment des sources les unes des autres.

Il est plus facile de maintenir le système proposé d'intégration de données avec BGLAV contrairement à GAV et LAV, la reformulation de requêtes est réduite au déploiement de règles. Comparée à d'autres approches d'intégration de données, BGLAV combine les avantages de GAV et LAV, réduit leurs inconvénients, et fournit une alternative pour l'intégration de données flexibles et extensibles.

1.3.4.2.6. Discussion

Dans les systèmes d'intégration basés sur les ontologies[*] et relativement à l'approche hybride, le schéma global et les schémas locaux sont des ontologies. Trois modèles d'intégration peuvent être appliqués à cette approche : GAV, LAV, GLAV.

Les systèmes XYLEME et PICSEL utilisant l'approche hybride. BGLAV et BAV sont aussi des approches hybrides.

[*] Les systèmes d'intégration basés sur les ontologies sont détaillés au niveau du chapitre 2.

Razor [54], Internet Softbot [55], InfoMaster [56], Information Manifold [57] sont des systèmes fondés sur un schéma global a base de règles, suivant tous l'approche LAV. Le système HERMES [58], suit une approche GAV.

Les systèmes fondes sur un schéma global à base de classes, tel que TSIMMIS [59] et MOMIS [60] suivent l'approche GAV, à l'inverse de SIMS et OBSERVER qui suivent LAV. Le système PICSEL [06], suit une approche LAV, Xyleme [62] et C-Web [62], relèvent à la fois des deux approches GAV et LAV et sont des systèmes fondés sur un schéma global a base d'arbres.

BGLAV, est une approche d'intégration combinant les avantages et évitant les inconvénients de GAV et de LAV. Elle augmente l'évolutivité et la rentabilité au-dessus des approches précédemment proposées.

Le tableau suivant présente une comparaison entre tous les scénarios de correspondance entre le schéma global et les schémas locaux qu'on vient de citer selon ces critères : Passage à l'échelle, Extension et Maintenance et Traitement de requêtes [26]

Nature de correspondance	Passage à l'échelle	Extension	Traitement
Gav	Etude	MAJ manuelle	Requêtes faciles
Lav	Oui	MAJ manuelle	Réécriture difficile (Vue)
GLaV	Oui	MAJ manuelle	Réécriture facile (Vue)
BGLaV	Oui	MAJ manuelle	Réécriture facile (Vue)
BaV	Oui	MAJ manuelle	A prouver

Tableau 0.3 Comparaison entre les différentes approches d'intégration

1.3.4.3. Processus d'intégration / Automaticité du mapping

Un troisième critère important permet de caractériser l'automaticité de génération du système intégré. La notion de passage à l'échelle, étant, de plus en plus, un aspect essentiel, on peut caractériser cette automaticité par l'automaticité d'intégration d'une nouvelle source (éventuellement convenablement préparée) au sein d'un système intégré.

1.3.4.3.1. Manuel

Dans les systèmes d'intégration qui n'utilisent pas d'ontologie, la synthèse des schémas locaux et les correspondances schéma global/schémas locaux sont faites manuellement. La signification des concepts utilisés, tant au niveau global qu'au niveau d'une nouvelle source, n'étant pas explicite, aucun traitement automatique ne peut être envisagé.

1.3.4.3.2. Semi automatique

Un premier niveau d'automatisation devient possible lorsque des ontologies linguistiques, (ensemble de termes associés à des relations terminologiques: synonymes, homonymes, etc.), sont utilisées. De telles relations ne sont pas de type booléen mais sont associées à des valeurs de vraisemblance. Lorsque les éléments du schéma global font explicitement référence à une ontologie linguistique, en général assez complète comme dans KRAFT [63], beaucoup de termes apparaissent dans la nouvelle source locale (dans son schéma, dans son ontologie locale Si elle existe) et peuvent être automatiquement retrouvés et classifiés dans l'ontologie globale.

Cette classification automatique, qui doit néanmoins être revue à la main compte tenu de l'imprécision des relations linguistiques, constitue une aide importante pour réaliser la modification finale du système intégré afin de prendre en compte la nouvelle source [64].

Une telle intégration est qualifiée de semi-automatique lorsque le domaine visé par l'intégration est suffisamment limité et formalisé, l'ontologie de domaine peut s'exprimer sous forme de prédicats de valeurs logiques. [29] qualifie ces ontologies d'ontologies logiques. C'est le cas par exemple, dans les approches orientées relations, des systèmes tels que Manifold [65] ou PICSEL [06].

1.3.4.3.3. Automatique

Dans les deux types de mapping précédents, il suffit que la nouvelle source soit explicitement exprimée en fonction de l'ontologie globale pour que l'intégration automatique soit possible.

1.3.4.3.4. Discussion

L'aspect modélisation des données, c'est-à-dire comment intégrer les différents schémas des sources, et leur interrogation, c'est-à-dire comment répondre efficacement aux requêtes posées sur le schéma global, ont été abondamment abordés dans la littérature [66].

Pour l'établissement des correspondances entre schémas, la plupart des systèmes sont basés sur une analyse manuelle du schéma effectuée par un expert, soit du domaine, soit d'intégration, même si des approches semi-automatiques de mise en correspondance sont parfois proposées, avec notamment l'utilisation de techniques issues de l'intelligence artificielle. Apporter une certaine automatisation dans le processus de prise en compte des sources permet une réduction et économie de temps de réponse et de coût.

1.4. Quelques systèmes d'intégrations

Dans cette section, nous passons en revue brièvement quelques grands projets d'intégration de données. Nous essayons de présenter un ou plusieurs exemples dans chaque catégorie.

1.4.1. Information Manifold

Information Manifold [51] [54] [67] est un système de décision logique pour intégrer des sources d'informations basées sur le Web. Il présente à l'utilisateur une vue globale unique, appelé World View, sur laquelle il formule ses requêtes. Cette vue est une collection de relations virtuelles et de classes qui décrivent le contenu des sources d'informations. Son modèle de données est relationnel, augmenté avec les hiérarchies de classes. Chaque fois qu'un utilisateur accède au système il identifie les sources pertinentes et exécute sur elles ses sous-requêtes. Une fois les requêtes exécutées l'utilisateur est responsable de nettoyer l'information redondante.

1.4.2. InfoMaster

InfoMaster [56] utilise une interface WEB ou un langage logique pour interroger une vue définie sous la forme d'un schéma relationnel. Ce schéma est

constitué de la partie commune des modèles fédérés. A chaque source de données est affectée une liste de règles associant les attributs de la vue à ceux des schémas locaux. Ces règles sont définies en utilisant le langage KIF [68]. Elles permettent aussi de convertir l'information afin de la mettre sous format adéquat et de l'adapter aux sources de données.

1.4.3. L'approche de Florescu

Le fonctionnement du système [69] [70] se base sur des interfaces d'accès présentées par chaque système, et un modèle commun représentant ces interfaces. Une interface d'accès a été définie en ODMG [71] (Object Database Management Group) pour fédérer les interfaces locales. Le système utilise un ensemble de règles afin de reformuler les requêtes tout en préservant la sémantique. Il existe deux types de règles: les règles de contraintes (mise en correspondance) qui ajoutent de la sémantique à la structure des informations et les règles de réécriture qui aident à reformuler des requêtes. Une fois que la requête est réécrite les wrappers prennent en charge les sous-requêtes pour les exécuter sur les systèmes locaux et renvoient les résultats.

1.4.4. GARLIC

L'objectif de GARLIC [72] [73] [74] est l'intégration de diverses sources multimédias en fournissant une vue intégrée des schémas des sources de données locales.

Ces schémas sont fusionnés en un schéma global exprimé en ODMG. L'accès aux objets de GARLIC peut se faire de deux façons, via une interface graphique ou en utilisant le langage de requêtes de GARLIC. Son langage de requêtes GQL est une extension de SQL supportant les expressions de chemins, les collections imbriquées ainsi que les méthodes. Les requêtes formulées par les utilisateurs sont envoyées au processeur de requêtes. Ce dernier développe des plans d'exécution de requêtes décomposées pour les multiples sources de données avant d'être envoyées aux wrappers. Un wrapper est associé à chaque source d'informations

1.4.5. MIX

MIX [75] [76] (Médiation of Information using XML) est un médiateur basé sur XML fournissant à l'utilisateur une vue intégrée de différentes sources développé dans le cadre d'un projet au sein de « University of California at San Diego Database Laboratory ». Cette vue est une interface définie en utilisant le langage XMAS (Xml Matching And Structuring language). Le système MIX est également composé d'une interface graphique appelée BBQ (Blended Browsing and Querying) et de wrappers. Les réponses aux requêtes sont sous forme de documents XML. Le médiateur de MIX récupère une requête formulée par l'utilisateur en utilisant l'interface BBQ, la traduit en utilisant les termes de la vue de médiation, et produit un ensemble de requêtes XML avant d'être envoyées aux wrappers. Ces requêtes peuvent être ensuite simplifiées en se basant sur les DTDs.

La difficulté dans MIX est que la modification de l'existant ou l'ajout d'une source réclame la modification de toute la vue de médiation.

1.4.6. PICSEL

PICSEL [77] [78] [79] est composé d'un médiateur jouant le rôle d'interface entre les utilisateurs et les sources de données à interroger. Son schéma global est exprimé dans le langage Carin [80]. L'architecture de PICSEL est constituée d'une ontologie du domaine, et de bases de connaissances (BCcomp) connectées au médiateur, décrivant le contenu des sources d'information. L'ontologie dans PICSEL fournit tout le vocabulaire utile aux utilisateurs pour formuler leurs requêtes. Elle est décrite en utilisant le langage de représentation CARIN-ALN [77].

Son inconvénient est qu'une fois l'ontologie partagée définie, chaque source doit utiliser le vocabulaire commun, ce qui limite l'autonomie des sources de données locales.

1.4.7. OBSERVER

OBSERVER [81] [82] [83] [84] associe des ontologies représentant chacune le vocabulaire d'un domaine d'application particulier, à des sources de données. À chaque terme du vocabulaire d'une ontologie sont associées les structures des différentes bases de données qui lui correspondent. Les ontologies donnent une description de la sémantique des informations indépendamment de la représentation syntaxique des données. Elles permettent une accessibilité plus large aux données car elles peuvent être utilisées pour capturer de nouvelles vues du monde.

L'inconvénient de OBSERVER est dû à l'exploitation des relations entre les concepts qui ne sont que purement terminologiques (synonymes). OBSERVER résout les conflits sémantiques, mais ne prévoir rien pour les conflits de valeurs. Un deuxième inconvénient est dû à la communication de différentes ontologies qui provoque une hétérogénéité d'ontologies. Cette hétérogénéité est produite si deux systèmes font des suppositions ontologiques différentes au sujet de leurs connaissances du domaine.

1.4.8. DISCO

Dans DISCO [85] [86] [87] les sources de données peuvent être des bases de données, des fichiers, des serveurs de données ou des pages HTML. Leurs données peuvent êtres structurées, semi-structurées ou non structurées. Le modèle de données de DISCO est une extension du modèle de données orienté objet ODMG et son langage de requêtes est OQL. DISCO est composé d'un médiateur qui présente à l'application une vue unifiée des sources, il reçoit une requête, l'optimise et la renvoie au wrapper. Une fois les requêtes exécutées, il récupère les résultats, les recompose avant de les renvoyer à l'application.

1.4.9. TSIMMIS

TSIMMIS [88] [89] [90] est un projet en collaboration entre Stanford et IBM Almaden Research Center dont le but est de développer un outil facilitant l'intégration rapide des sources de données hétérogènes qui peuvent inclure des données structurées et non structurées. Le modèle commun utilisé, est le modèle

OEM (Modèle d'Échange d'Objet) [90]. Un langage de règles MSL est utilisé comme langage de spécification des médiateurs et des wrappers. MSL utilise des règles de prologue et des fonctions pour traduire des objets. Le langage de requêtes utilisé est OEM-QL afin d'interroger les objets OEM.

L'intégration des données dans TSIMMIS exige la participation des humains. Son inconvénient est dans le choix du médiateur par l'utilisateur qui ne possède pas d'informations sémantiques pour identifier les médiateurs qui répondent à ses besoins. Le modèle de représentation des données de TSIMMIS est simple et n'est pas très adapté à la représentation des informations complexes.

1.4.10. MOMIS

Le projet MOMIS (Mediator envirOnment for Multiple Information Sources) [91] [92] [93] [94] concerne la création d'un médiateur basé sur les systèmes d'intégration de données structurées et semi-structurées. Il a été développé en collaboration entre l'Université de Modène et Reggio Emilie et l'Université de Milano et Brescia. Il a été conçu pour permettre l'accès à des sources d'informations hétérogènes stockées dans des bases de données classiques (relationnelles et orientées objet), des systèmes de fichiers et des sources de données semi-structurées.

1.4.11. XYLEME

Est un système d’intégration physique (approche entrepôt) avec XML comme modèle de données. Il adopte l’approche hybride dans son architecture. Les ontologies globale et locales sont exprimées à l’aide d’arbres, avec un mapping GaV / LaV. Le mapping est réalisé semi automatiquement par la génération de tables de correspondance entre les chemins de l’ontologie globale et les chemins des ontologies locales.

1.4.12. PIAZZA

Est un système qui intègre des sources qui sont, soit des ontologies, soit des données semi structurées (décrites par schéma XML ou DTD). Il s’appuie sur

le modèle de données XML et une adaptation de XQuery comme langage de requêtes. Ce système suit une architecture Peer-to-Peer avec des ontologies et/ou schémas multiples. Le mapping entre les différentes ontologies est bidirectionnel (une source est décrite en fonction d'une autre et vice versa). Pour une requête donnée, le schéma d'un nœud (source de données) peut être considéré comme étant le schéma global et ainsi, de proche en proche, les sources concernées seront interrogées.

1.4.13. Discussion

La plupart de ces systèmes utilisent une architecture à ontologie unique. Cette approche est intéressante dans le cas ou les sources auraient une vue similaire du domaine. Si ce n'est pas le cas, il est difficile de trouver une ontologie consensuelle minimale. D'un autre coté, une source ne peut être changée sans reconsidérer et l'ontologie globale et les autres sources pour s'assurer qu'il n'y a pas de conflits. Ceci ne privilégie pas l'autonomie des sources.

Pour les autres systèmes qui sont basés sur des ontologies multiples, la comparaison d'ontologies est difficile en l'absence d'un vocabulaire commun. De plus, ils ont besoin de définir le mapping entre plusieurs couples d'ontologies.

XYLEME et PICSEL sont les seuls à utiliser l'approche hybride. Mais le premier adopte l'approche entrepôt qui ne favorise pas la fraicheur des données. Quant au second, il utilise son propre formalisme de langage de description.

L'intégration de sources de données à base ontologique par l'approche matérialisée dite Datawarehouse a constitué le travail de thèse de Mr Dung NGUYEN XUAN, alors que par l'approche médiateur a été étudié par Melle AMEL BOUSSIS dans son travail de magister. Au cours de notre étude, nous allons tirer profit des résultats obtenus dans ces deux thèses dans la mesure de leur application dans le cadre d'une approche HYBRIDE.

1.5. Conclusion

L'intégration des différentes sources d'information est une nécessité cruciale aujourd'hui. Entreprises, e-commerce, la bioinformatique, le Web sémantique et de nombreux autres secteurs sont des environnements qui ne peuvent pas continuer à vivre sans l'intégration de leurs informations..

Différents défis existent et de rompre les technologies al sont proposées pour intégrer et gérer des sources hétérogènes de manière homogène.

Dans ce chapitre, nous avons décrit les défis fondamentaux qui apparaissent lors de l'élaboration d'un système d'intégration et nous avons brièvement passé en revue les principales technologies existantes. Il existe d'autres approches, parmi lesquels des approches hybrides d'intégration de données dont nous parlerons dans le deuxième chapitre en détail. Notre objectif est de mettre en évidence qu'une approche d'intégration de données basée sur un système hybride (semi-matérialisé) est une solution optimale pour de nombreux scénarios d'intégration.

Nous avons aussi, brièvement présenté les exemples de prototypes de recherche pour toutes les approches discutées.

Dans le chapitre suivant nous allons donner un état de l'art des approches hybrides et nous proposons une approche hybride à base indexé pour l'intégration des données et le positionnement de notre approche.

CHAPITRE 2 LES SYSTEMES D'INTEGRATION HYBRIDES

2.1. Introduction

Dans ce chapitre, nous présentons une architecture semi-matérialisée (hybride) d'un système d'intégration de données. Elle fournit une optimisation des requêtes au système d'intégration ainsi que d'une flexibilité de rafraîchissement des données pour les différentes sources de données, selon les besoins de l'application d'intégration.

Dans la première section, nous essayons d'expliquer pourquoi et dans quels cas une approche d'intégration de données hybride est préférable. Nous donnons un état de l'art pour de telles approches et nous passons en revue de quelques exemples. Dans la deuxième section, nous donnons une rapide présentation de notre architecture proposée. Nous discuterons également de l'efficacité du temps de réponse aux requêtes par rapport à une approche d'intégration entièrement virtuelle et de sa souplesse d'actualiser les données en comparaison avec une autre approche complètement matérialisée.

2.2. Système d’intégration de données Hybride

Dans les systèmes d'intégration de données, il ya un compromis entre le temps de réponse des requêtes et la fraîcheur des données. Les approches entièrement matérialisé et entièrement virtuelles favorisent l'un de ces objectifs. D'un autre point de vue, dans les applications d'entreprise, par exemple, le compromis se situe entre le coût de la construction d'un entrepôt et le coût d'une requête en direct.

Les besoins des différentes applications d'intégration sont très divers et une modèle unique pour toutes ces approche est souvent inadapté [93].

Les approches hybrides ont été développées pour créer des systèmes d'intégration optimales et plus souple.

Ils essaient de donner la possibilité de matérialiser certaines données sous-jacentes et l'interrogation d'autres d'une manière virtuelle.

Dans de nombreux cas dans un processus d'intégration, pour certaines données de sources sous-jacentes, les modifications sont fréquentes et les utilisateurs ont besoin de savoir immédiatement s'il ya des mises à jour. Dans ces cas, les données doivent rester dans les sources et être extrait au moment de la requête, ce qui conduit à un traitement de la requête lent. Dans le même processus d'intégration, certaines données peuvent changer moins souvent ou les utilisateurs peuvent appuyer un certain retard de mise à jour de ces données. Matérialiser cette seconde catégorie de données optimise le traitement des requêtes du système d'intégration. Une telle demande a été la principale motivation pour développer des approches d'intégration hybrides.

L'approche hybride à l'intégration de données a été introduite dans [94] avec le projet Squirrel. Le projet concerne le développement d’une approche médiateur générique et souple qui permet à une vue intégrée d'être entièrement matérialisée, entièrement virtuelle ou partiellement matérialisée.

D'autres efforts pour l'approche hybride sont faits dans [95], [96], [97], [98]. Dans la section suivante nous examinons brièvement ces approches.

2.3. Modèle d’Architecture d’système hybride (semi-matérialisé)

Cette troisième solution d'intégration de données concerne le développement d'un schéma intégré partiellement matérialisé. Les approches entièrement matérialisées et entièrement virtuelles d'intégration de données obéissent à des priorités différentes. Dans une approche entièrement matérialisée, la priorité principale est le temps de réponse aux requêtes, et dans l'intégration entièrement virtuelle de données, la fraîcheur de données est prioritaire. Nous détaillerons cette approche dans le chapitre suivant.

Cependant, dans beaucoup de scénarios d'intégration de données, différentes priorités peuvent être associées aux différentes données, et un compromis entre le temps de réponse aux requêtes et la fraîcheur de données peut être préféré à la satisfaction d'une seule de ces deux questions. Une approche souple qui permet à quelques données d'être matérialisées et à d'autres données d'être virtuelles permet de satisfaire les deux objectifs. Dans les approches hybrides existantes, la vue globale est divisée en une partie matérialisée et une partie virtuelle. Certains objets ou relations sont choisis pour être matérialisés tandis que d'autres restent dans les sources locales et seront extraits au moment de la requête [55].

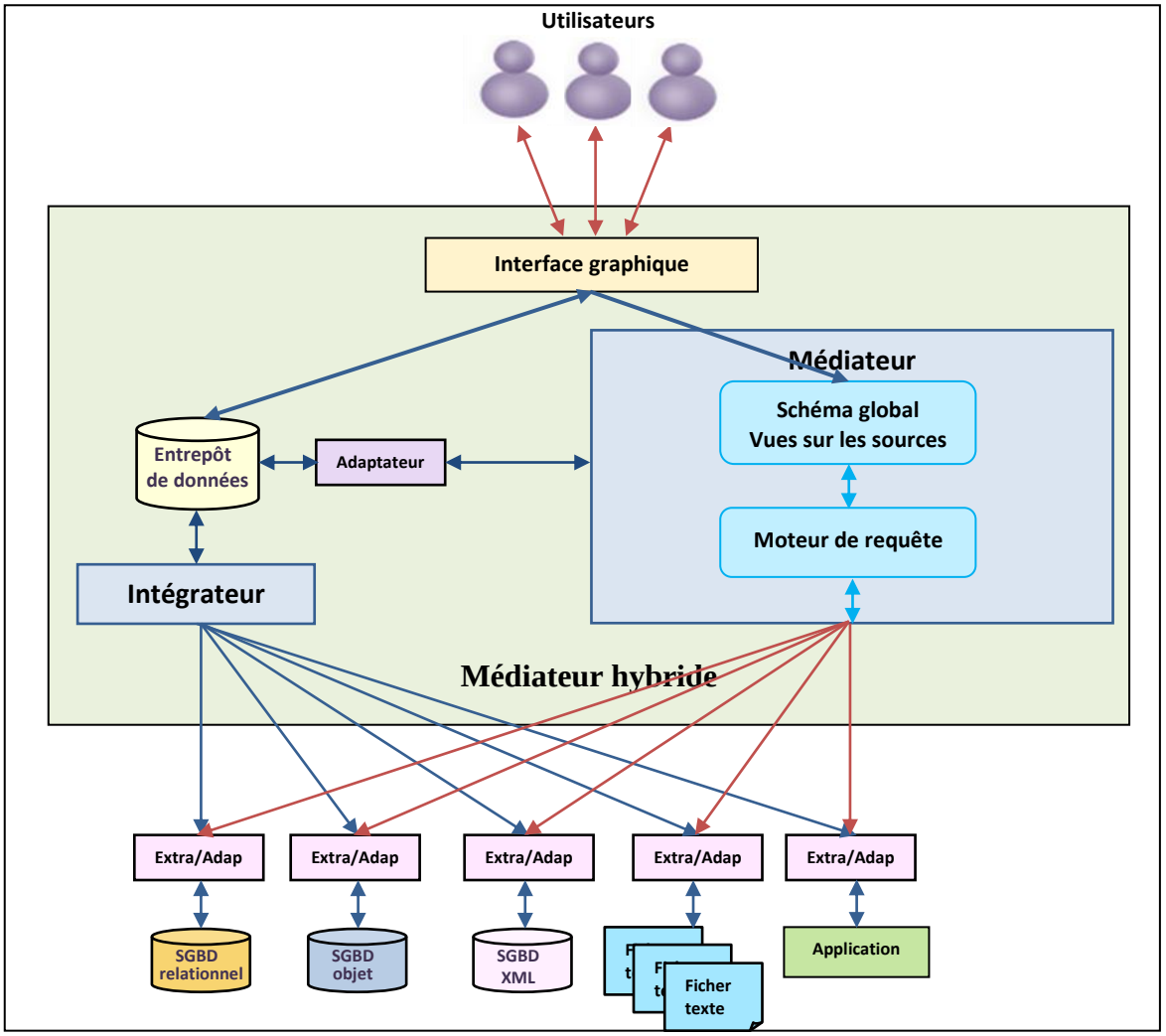

Figure 0.1 Architecture d'un système d'intégration de données hybride (semi-matérialisé)

Notre architecture d'intégration hybride qu'on a proposée est basée sur cette architecture modèle.

2.4. Les principaux systèmes hybrides

Dans cette section nous allons décrire quelques architectures hybrides existantes dans la littérature.

2.4.1. Une approche hybride basée sur l'intégration des médiateurs Squirrel

L'intégration des médiateurs Squirrel a été développée initialement pour soutenir l'intégration des vues complètement matérialisées [99]. Plus tard [94], il a été généralisé en élaborant une architecture hybride ainsi que des vues intégrées pour les modèles relationnels et orienté objets.

Dans d'intégrateur hybride de médiateur Squirrel, certaines relations peuvent être concrétisées, certaines peuvent être virtuelles et d'autres sont hybrides. Une relation hybride est une relation dans laquelle certains attributs sont matérialisées et d'autres sont virtuels, ils sont interrogés au moment de la requête. La figure 13 montre l'architecture du médiateur Squirrel intégration hybride.

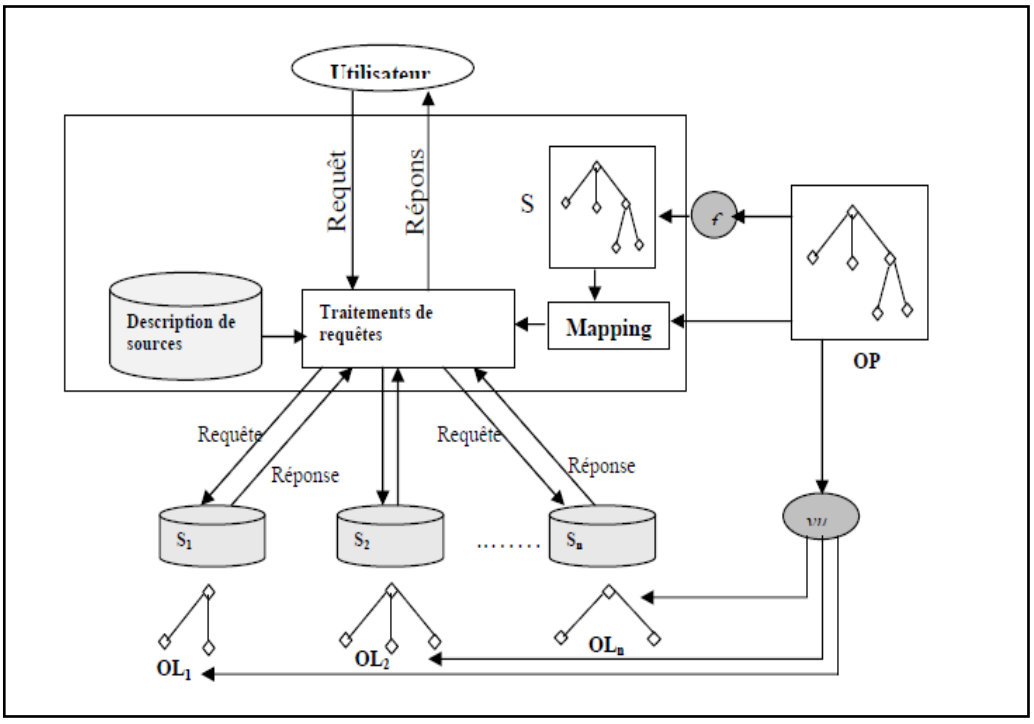

Figure 0.2 Architecture d'un médiateur en utilisant une ontologie partagée

2.4.2. Intégration de données semi-structurées avec le système de base de données Lore

Une autre approche hybride a été conçue pour intégrer les bases de données semi-structurées avec le système de Lore. Lore (Lightweight Object Repository) [96] est un système de gestion de base de données qui a été conçu spécifiquement pour la gestion des données semi-structurées avec les OEM (Object Exchange Model) modèle de données.

Le projet Lore intégration hybride utilise la plate-forme Lore de développer un entrepôt de données et il ajoute une partie virtuelle au système d'interroger les données externes nécessaires pour le système. Un DW dispose de certaines données qui n'ont jamais ou rarement mise à jour, par exemple des données géographiques ou à l'hôtel de données pour une agence de Voyage. D'autres données, comme les données météorologiques en revanche, changer très souvent et peut être important pour la prise de décision système basé sur DW ainsi que pour le client de l'agence de Voyage. Lore matérialise les données plus stables dans une base de données Lore et il fournit la Lore gestionnaire de données externes pour interroger des données externes telles que les informations météorologiques. Il fait la distinction des données locales et externes invisible pour l'utilisateur.

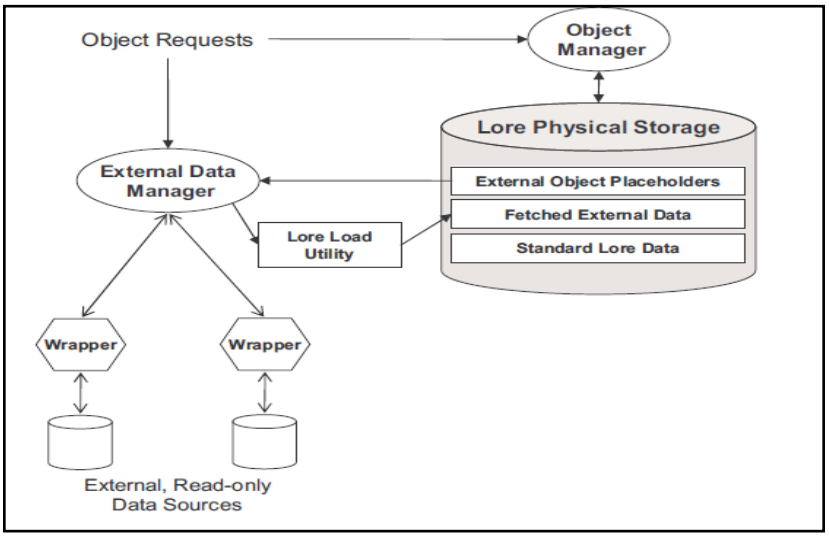

Figure 0.3 Architecture du système d'intégration de données Lore

La figure 2.3 montre l'architecture du système d'intégration de données Lore. Lorsqu'une requête arrive au système, un moteur d'exécution de requêtes envoie les requêtes pour les objets soit Object Manager ou gestionnaire de données externes doivent être gérés. L'objet des demandes Gestionnaire de données dans la base de données Lore. Le gestionnaire de données externes intègre les données de sources externes et fonctionne également comme l'intégrateur entre les données dans une source de données externe et les données des bases de données locales.

2.4.3. Une architecture hybride pour l'entreposage du contenu Web

Zhu [95] propose une approche hybride à l'entreposage du contenu Web. Les données sur le Web sont généralement mis à jour fréquemment. L'objectif de ce projet est de relever le défi de l'entretien des entrepôts de données au moment de leur élaboration sur le Web. Il sélectionne un ensemble de données stables ou fréquemment demandé à se matérialiser dans un DW contenant les données historiques. Les données externes, qui sont les données complémentaires du Web pour DW et sont fréquemment modifiées, sont interrogés par une méthode virtuelle. Une requête peut être appliquée sur le DW ou en utilisant l'approche virtuelle ainsi que par l'aide d'une combinaison de données d'entrepôt de données et Web.

Il utilise le modèle MIX (modèle de métadonnées pour les données en fonction d'intégration X-change) pour représenter des données ainsi qu'une description de leur contexte d'interprétation sous-jacente. Avec ce modèle, les données et les métadonnées sont interprétées en utilisant des ontologies de domaine spécifique. Les règles de correspondance sont définies par une ontologie du système spécifique pour semi-automatiquement des codes de transformation. Un moteur ontologie rend les règles de correspondance entre les données du Web et les attributs dans le système d'intégration de données. Ces règles de correspondance sont utilisées pour l'approche virtuelle ainsi que pour l'entretien DW.

2.4.4. Une approche hybride pour l'intégration d'ontologie

Une approche hybride pour l'intégration d'ontologie est également abordée dans [93]. Sa principale différence avec [95] est que les sources de données externes peuvent être des ontologies, des données sur le Web, ou toute autre forme de sources de données structurelles ou semi-structurées. Les ontologies sont utilisées comme formalisme de la vue intégrée. Pour la cartographie entre les concepts de la source ontologie et le schéma intégré, cette approche utilise une méthode LAV dans lequel le concept de l'ontologie de la source correspond à une requête de l'ontologie vision intégrée.

La figure suivante représente l'architecture logicielle de cette approche.

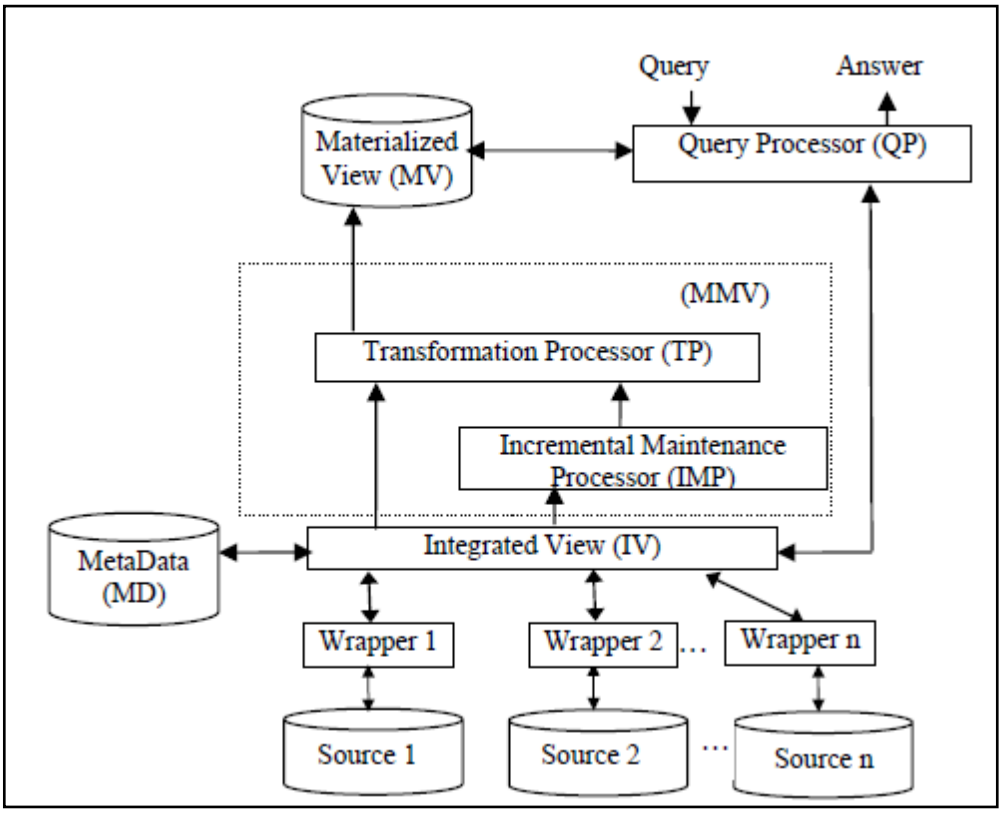

Figure 0.4 Architecture de l'approche hybride pour l'intégration d'ontologie [93]

2.4.5. IXIA, IndeX-based data Integration Approach

Comme d'autres approches hybrides, l'objectif principal d'IXIA [98] est de faire un compromis entre le temps de réponse des requêtes et la fraîcheur des données dans une application d'intégration de données.

Cette approche est basée sur l'indexation des objets au niveau du médiateur. La structure d'indexation d'objet et les données requises pour la mettre à jour sont matérialisées, alors que d'autres données demeurent dans les sources et sont interrogées au moment de la requête.

L'approche proposée, IXIA, est basée sur la plateforme Osiris qui est un système de base de base de données et de connaissances basé sur un modèle objet. Il met en application le modèle de données des P-types, qui est un modèle basé sur une hiérarchie de vues. IXIA a deux objectifs principaux:

1. Etendre l'optimisation sémantique des requêtes à toutes les requêtes du système d'intégration.

2. Fournir un mécanisme flexible de mise à jour des données qui rend cette approche capable de tolérer des sources de données avec différentes caractéristiques, et choisir un compromis optimal entre le temps de réponse des requêtes et la fraîcheur de données.

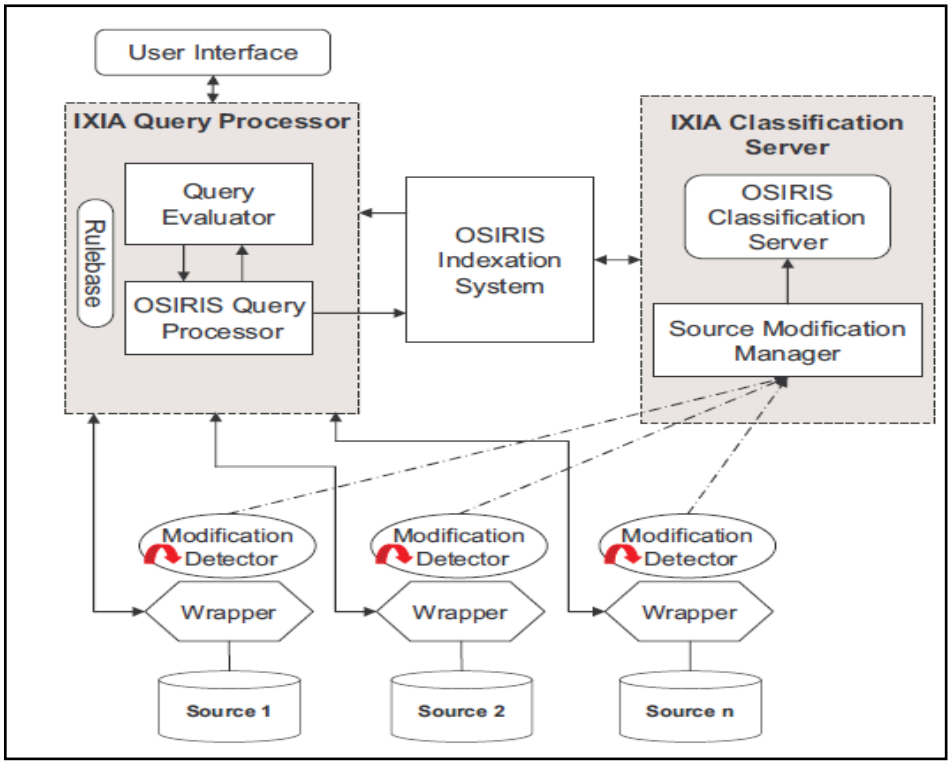

Figure 0.5 Architecture d'intégration de données IXIA

2.4.6. Synthèse

Comme nous venons de citer, plusieurs projets ont été développé en adoptant l'approche semi-matérialisée, on y trouve le projet Squirrel [28] qui utilise une approche hybride pour élaborer un cadre médiateur général et souple, mais se concentre uniquement sur les sources de données relationnelles ou orientées objet. Une architecture pour l'entreposage du contenu web a également été discutée dans [37], elle utilise une approche hybride dans le but d'intégrer les données de l'entrepôt de données avec le "nécessaire" d'information sur Internet. Elle utilise les ontologies pour exprimer des connaissances de domaine liées à des sources web et le modèle logique de l'entrepôt de données. En outre, un moteur d'ontologie est déployée comme une couche intermédiaire en définissant les règles de correspondance entre les données web et les attributs de l'entrepôt de données dans les ontologies à l'aide de la structure de l'entrepôt de données et les exigences de la réparation, aussi certaines données sur le Web sont sélectionnées pour la matérialisation, Néanmoins, certaines requêtes peuvent ne pas être répondues en utilisant uniquement des données matérialisée dans l'entrepôt de données.

Dans [06], une architecture pour l'intégration d'ontologie a été introduite sur la base de l'approche entièrement virtuelle. Ils construisent l'intégration « virtuelle » des vues selon les approches de mapping entre les ontologies locales et l'ontologie globale. Le sens de mapping ici est qu'un concept d'une seule ontologie correspond à une requête sur d'autres ontologies. Puis, une requête sera posée sur le l'ontologie « virtuelle » globale et basée sur le mapping entre les concepts de l'ontologie globale et des ontologies locales, la requête sera dépliée et les réponses récupérées à partir des sources.

La proposition [07] étend l'architecture d'intégration d'ontologie en utilisant DL-Lite pour exprimer le schéma global, et en utilisant l'approche de mapping LAV entre les bases de données « locales » et le schéma de l'ontologie globale. Il y a une différence remarquable entre notre travail et le leur.

[97] a proposé une architecture hybride pour l'intégration de données hétérogènes dans le but de réduire le temps de réponses, certaines données

fréquemment demandées sont matérialisées : certaines requêtes ne doivent être répondues qu'à partir des vues matérialisées, d'autres doivent être répondues directement des sources,

Notre architecture se base beaucoup plus sur celle proposée dans [97], à la différence qu'elle est fondée sur le principe suivant : dans la partie entrepôt de données de notre architecture, seuls les concepts ontologiques existants ou étant référencés qui seront mis dans l'entrepôt de données ainsi que les données dérivants de ces concepts, le reste des concepts seront intégrés dans la partie médiateur, cette intégration repose sur un algorithme entièrement automatique.

2.5. Conclusion

Dans ce chapitre, nous avons brièvement présenté l'approche hybride de l'intégration de données. Tout comme une approche de médiation qu'il a une architecture d'adaptateur, mais une partie de l'information est matérialisée au niveau du médiateur.

Tout comme les autres méthodes hybrides [94, 95, 96, 97,98], nous avons choisissons quelques données pour être matérialisé et laisser les autres virtue1les, En plus de cela, nous utiliseront des ontologies (locales, globales) pour la résolution des conflits de sémantique, et l'intégration que nous proposons est entièrement automatique, aussi, les sources de données locales sont munies chacune d'une ontologie locale, ces sources de données sont dites des sources de données à base d'ontologie.

Dans le chapitre suivant, nous présenterons les ontologies, et leurs domaines d'utilisation de façon générale, et dans l'intégration des données plus spécialement, puis nous allons aborder les bases de données à base ontologique, leurs structures et quelques exemples existants.

CHAPITRE 3. ONTOLOGIE ET BASE DE DONNEES A BASE D'ONTOLOGIE

3.1. Introduction

Plusieurs conflits de l'intégration de données ont été soulevé dans le premier chapitre. Afin de résoudre quelques conflits, l'utilisation d'une ontologie apparait comme une solution assurant l'automatisation du processus d'intégration sémantique de données.

Ce chapitre comporte deux sections, dans la première section, nous allons d'abord donner une définition de l'ontologie puis leurs domaines d'application de façon générale ensuite au niveau des systèmes d'intégration,

La deuxième section est consacrée aux Bases de Données à Base Ontologique, nous verrons par la suite l'apport apporté par cette architecture lors de l'intégration des données. Notre approche proposée repose sur des sources de données à base ontologique.

3.2. Ontologie

Le terme ontologie est un terme grec composé des mots « Ontos » et « logos » qui signifient l'essence de l'être. Ce terme hérite d'une branche de la philosophie qui s'intéresse à la Science de l'Etre initié par Aristot [100], il a été réutilisé en Intelligence Artificielle au début des années 90 grâce au projet ARPA Knowledge Sharing Effort [101].

La première définition des ontologies était donnée par Gruber : « Une ontologie est une spécification explicite d'une conceptualisation » [102].

Une ontologie est une modélisation de connaissances du monde, d'informations extralinguistiques, organisée en un réseau de concepts, elle consiste en un ensemble de définitions de catégories de base (objets, relations, propriétés) qui permettent de décrire les objets du domaine que l'on traite, leur propriétés et les relations qu'ils entretiennent les uns avec les autres [103].

Les ontologies ont pour but de fournir une représentation des connaissances dans un domaine toute ontologie doit être communément acceptée et pourra être réutilisée et partagée par divers applications et groupes [104].

Une ontologie est un : « Modèle conceptuel spécifique élaboré dans le domaine de la gestion du savoir, une ontologie peut représenter des relations complexes entre des objets et inclure, les règles et les axiomes manquants dans un réseau sémantique » [105]

Une ontologie permet de définir de façon *formelle*, *explicite*, *référençable* et *consensuelle* l'ensemble des concepts partagés d'un domaine [102].

- Par *formelle* nous entendons que l'ontologie est définie dans un langage traitable par machine.
- *Consensuelle* signifie que l'ontologie est acceptée par les membres d'une certaine communauté travaillant dans le domaine de discours. Tous les concepts partagés peuvent être modélisés à l'identique par tous les membres de cette communauté, qui peut être, avec les ontologies normalisées, l'ensemble d'un secteur professionnel.

- *Référençable* désigne la possibilité de référencer de manière unique les concepts ontologiques sans importation des définitions de ces derniers.

- Enfin *explicite* indique que le domaine modélisé est décrit indépendamment d'un point de vue ou d'un contexte implicite. Plusieurs points de vues, légèrement différents pourront donc partager les mêmes concepts.

Ces différentes caractéristiques offrent aux ontologies un haut niveau d'abstraction et la possibilité d'être partagées. Les ontologies sont définies au moyen de langages de définition comme RDF Schéma [106], DAML+OIL [107], OWL [108], PLIB [109], etc.

Formellement, une ontologie peut être définie par un quadruplet : O : < C, P, Sub,Applic > [109, 110] avec :

- C : ensemble des classes utilisées pour décrire les concepts d'un domaine donné. Chaque classe est associée à un identifiant universel globalement unique ;

- P : ensemble des propriétés utilisées pour décrire les instances de l'ensemble des classes C. Nous supposons que P définit toutes les propriétés consensuelles dans le domaine. Chaque propriété est associée à un identifiant globalement unique ;

- Sub est la relation de subsomption de signature Sub : $C \rightarrow 2^C$, qui à chaque classe ci de l'ontologie, associe ses classes subsumées directes. La relation de subsomption utilisée dans toutes les ontologies est la relation *OOSub* (*is-a*) qui est la relation de subsomption avec héritage. Elle est seulement utilisée entre classes d'une même ontologie et permet de définir des hiérarchies simples.

- Applic est une fonction de signature Applic : $C \rightarrow 2^P$, qui associe à chaque classe de l'ontologie les propriétés qui sont applicables pour chaque instance de cette classe. Seules les propriétés ayant pour domaine une classe ou l'une de ses superclasses peuvent être applicables pour décrire les instances de cette classe.

A partir de cette formalisation, chaque langage de définition d'ontologies propose de nouvelles constructions [111].

3.2.1. Description d'une ontologie

Une ontologie fournit un vocabulaire commun d'un domaine et définit le sens des concepts et les relations entre ces concepts.

Même si les ontologies peuvent avoir certaines divergences, elles sont toujours les mêmes : une ontologie est constituée de concepts et de relations ainsi que des propriétés et des axiomes.

3.2.1.1. Les composants de l'ontologie

Une ontologie débute par une taxinomie, un arrangement structuré d'informations en classes qui catégorisent un sujet et ses dépendants.

D'après Gruber [102], la formalisation d'une ontologie se met en place grâce aux composants suivants :

a. *Les concepts (ou classes) :* sont des notions ou objets permettant la description d'une tâche, d'une fonction, d'une action, d'une stratégie ou d'un processus de raisonnement, etc. Ils peuvent être abstraits ou concrets, élémentaires ou composés, réels ou fictifs. Habituellement, les concepts sont organisés en taxonomie. Une taxonomie est une hiérarchie de concepts (ou d'objets) reliés entre eux en fonction de critères sémantiques particuliers.

b. *Les relations :* sont des liens organisant les concepts de façon à représenter un type d'interaction entre les concepts d'un domaine. Elles sont formellement définis comme tout sous-ensemble d'un produit de n ensembles, c'est-à-dire R : C1 x C2 x …x Cn. Des exemples de relations binaires sont sous-classe de, connecté à certaines relations binaires entre des objets sont considérées comme des rôles [112].

c. *Les propriétés (ou attributs) :* sont des restrictions des concepts ou des relations.

d. *Les rôles :* « Un rôle caractérise une entité par quelque rôle qu'elle joue dans sa relation à une autre entité. Le type « humaine », par exemple, est un type de phénomène qui dépend de la forme interne de l'entité ; mais la même entité peut être caractérisée par des rôles de type mère, employé ou piéton ». [113].

e. *Les fonctions :* Sont des cas particuliers des relations dans lesquelles le $n^{ième}$ élément de la relation est unique pour les n-1 précédents. Formellement, les fonctions sont définies ainsi, F : C1 x C2 x … x Cn-1, Cn, comme exemple de fonction binaire, nous avons la fonction mère-de.

f. *Les axiomes :* permettent de définir la sémantique des termes (classes, relations, instances), leurs propriétés et toutes contraintes quant à leur interprétation. Ils sont définis à l'aide de formules bien formées de la logique du premier ordre en utilisant les prédicats de l'ontologie.

g. *Les instances :* sont utilisés pour représenter des éléments dans un domaine.

3.2.1.2. Types d'ontologies

Les types d'ontologies mises au point sont très diverses, on propose dans cette section deux classification distinctes :

a. *$1^{ère}$ classification :* Selon le type de la structure de conceptualisation, l'ontologie se présente de façon différente selon le degré de formalisation du langage utilisé pour définir la signification des termes [114]. Quatre sortes d'ontologies sont distinguées en fonction du type de langage utilisé :

   - Ontologies hautement informelles : exprimées en langage naturel sans aucune restriction.

   - Ontologies semi-informelles : exprimées dans un langage naturel qui est structuré avec un vocabulaire limité afin de restreindre les ambiguïtés.

   - Ontologies semi-formelles : Quand l'ontologie est représentée à l'aide d'un langage artificiel définie de façon formelle.

- Ontologies rigoureusement formelles : exprimées dans un langage contenant une sémantique formelle, des théorèmes et des preuves de propriétés telles que la robustesse et l'exhaustivité.

b. *2ème classification :* L'ontologie se définie selon le domaine étudié et le degré de généralité ou de précision des connaissances représentées, les types d'ontologie les plus utilisées :

- Les ontologies générales : Appelée aussi « top level onology » [115] ou « meta-ontology » [116] ou « ontologie générique ou noyau d'ontologie » [117], elles portent sur les concepts généraux indépendamment d'un domaine ou d'un problème particulier, tels que les concepts de temps, d'espace, de notion mathématiques « elles sont prévues d'être utilisées dans des situations diverses et variées et peuvent service une large communauté d'utilisateurs ainsi elles peuvent être utilisées dans l'organisation des parties substantielles des connaissances humaines, comme la compréhension du langage naturel » [118].

  Une ontologie générale ou appelée aussi de haut niveau est généralement conçue pour réduire les incohérences des termes (définis plus bas en hiérarchie).

  Parmi les ontologies générales, on trouve Cyc qui est développée avec le modèle logique, en utilisant le langage CycL.

  On trouve aussi KROntology qui, elle, utilise le modèle de treillis et le Formal Concept Analysis (FCA) pour représenter l'ontologie.

- Les ontologies du domaine : Expriment des conceptualisations spécifiques à des domaines particuliers tout en étant génériques pour un domaine précis. Ces conceptualisations mettent des contraintes sur la structure et les contenus des connaissances du domaine.

  Les ontologies de domaines sont réutilisables au sein d'un domaine donné, mais pas d'un domaine à un autre. Elles fournissent le vocabulaire des concepts du domaine. De nombreuses ontologies de domaine ont été

développées notamment en médecine, physique et aussi dans le domaine de la modélisation de l'entreprise.

- Les ontologies d'applications : Elles sont généralement spécifiques à une application ; elles contiennent suffisamment de connaissances pour structurer un domaine particulier.

  Ce type d'ontologie décrit des concepts qui dépendent en même temps d'un domaine particulier et d'une tâche particulière [115].

  Ces ontologies peuvent contenir des extensions spécifiques telles les méthodes et tâches. Elles contiennent toutes les définitions nécessaires pour décrire la connaissance requise pour une application particulière.

- Les ontologies de tâche : Selon [115], ce type d'ontologie décrit un vocabulaire en relation avec une tâche ou une activité générique comme le diagnostic ou la vente.

  Les ontologies de tâche fournissent un lexique systématisé de termes utilisés pour résoudre les problèmes associés à des tâches particulières (dépendantes ou non du domaine).

  Ces ontologies fournissent un ensemble de termes au moyen desquels on peut décrire au niveau générique comment résoudre un type de problème. Elles incluent des noms génériques (par exemple plan, objectif, contrainte), des verbes génériques (par exemple assigner, classer, sélectionner), des adjectifs génériques (par exemple assigné) et d'autres mots qui relèvent de l'établissement d'échéances.

Différemment des ontologies de tâches qui ont accès sur les connaissances visant à résoudre des problèmes, les autres types d'ontologies (ontologies générales, ontologies de domaine, et ontologies d'application) décrivent les connaissances statiques indépendamment de la façon dont on résout les problèmes.

Combiner toutes ces ontologies peut construire une nouvelle ontologie dite *Ontologie partagée*.

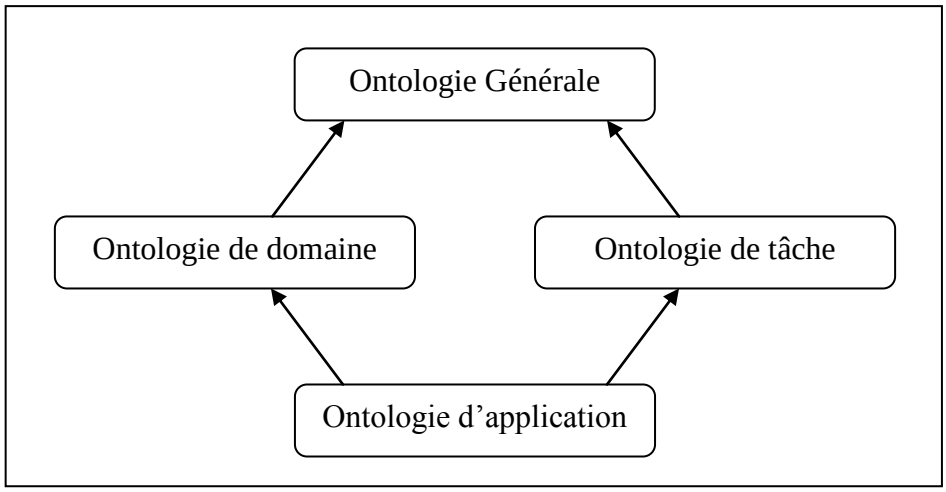

Figure 0.1 Combinaison des différentes ontologies [113]

3.2.2. Quelques domaines d'utilisation d'ontologies

L'émergence des ontologies a apporté beaucoup d'évolution dans le domaine informatique, dans cette section nous présenter quelques domaines d'utilisation des ontologies et l'apport qu'elles apportent.

3.2.2.1. Traitement du langage naturel

Retrouver et reconnaître des similitudes conceptuelles entre des mots permet d'améliorer la recherche documentaire, l'utilisation d'ontologies par des moteurs de recherche permet de retrouver une réponse contenant des documents pertinents par rapport à la sémantique des mots recherchés.

3.2.2.2. Web sémantique

Aujourd'hui, le web est accessible à tous pour une utilisation veste et variée, cette extraordinaire source d'informations souffre du problème de l'isolement complet entre les services offerts, cela induit qu'une personne doit, ou bien, avoir une connaissance approfondie du web, ou bien avoir passer un long moment à parcourir divers sites afin d'arriver au résultat escompté ; et vue l'évolution rapide et incontrôlée du contenu présent sur le web conduit généralement à la deuxième solution, tout cela illustre le besoin d'agents

intelligents capables de s'adapter à cette masse d'information présente sur le web, et d'être autonome et capables de savoir quels services appeler. Les Web Services fournissent le moyen de réaliser les appels aux différents services néanmoins l'agent doit avoir une bonne connaissance du domaine d'application sur lequel il rend des services, c'est l'idée du Web Sémantique [119] : Rendre les données compréhensibles à des agents automatiques des ontologies qui puissent être partagées et référencées sur le web.

3.2.2.3. Base de données

Afin de concevoir toute base de données, il faut commencer par définir le modèle conceptuel où l'on décrit toutes les données à décrire, puis passer à la modélisation logique des données qui consiste à traduire automatiquement le modèle conceptuel afin d'obtenir une représentation physique en machine, donc l'absence totale du modèle conceptuel au niveau physique, cela engendre deux problèmes :

- Dans le cas d'une évolution de la base de données, il faut mettre à jour son modèle conceptuel et son modèle logique respectivement.
- Tout utilisateur de la base de données doit connaître à priori la signification des données, ou il accède par un applicatif.

Pour résoudre ces problèmes, l'architecture OntoDB [120] a été conçue avec un prototype implémenté au sein de LISI, cette architecture permet de représenter l'ontologie et les données dans une base de données, ce qui permet à l'utilisateur d'accéder directement à la signification des données et des éléments du modèle conceptuel qui les structure grâce à l'ontologie stockée.

3.2.2.4. Intégration des sources de données

Intégrer des sources de données (bases de données, pages web, documents, etc.) au sein d'un seul référentiel pose d'énormes problèmes vu leurs diversités et leurs hétérogénéités.

3.2.2.4.1. Bases de données

La réalisation d'un modèle conceptuel d'une base de données se fait selon une approche perspective ce qui implique :

- Les données décrites sont les données pertinentes pour l'application.
- Toutes les données doivent respecter les contraintes et les définitions mises dans le modèle conceptuel.
- La conception est restreinte à l'application en utilisant les conventions des concepteurs, d'où un modèle conceptuel répondant qu'aux besoins de l'application.

Afin de résoudre les problèmes d'hétérogénéité engendrés par une telle approche, plusieurs solutions ont été apportées, parmi elles le composant logiciel « le médiateur », qui est souvent utilisé pour donner une vue homogène de l'ensemble des sources, il permet aussi d'accéder par une même requête à des données se trouvant dans des sources différentes, mais la conception d'un tel composant nécessite l'intervention humaine.

L'utilisation d'ontologies vise à permettre la conception d'un composant informatique générique qui est capable de comprendre l'hétérogénéité des sources de données, dans ce cas l'ontologie sera conçue selon une approche descriptive visant à définir l'ensemble des concepts présents dans un domaine particulier et non pour une application particulière comme le modèle conceptuel, dans ce cas, toute donnée non utilisée dans l'application sera considérée comme inconnue même si elle existe, bien sur dans l'ontologie, ce qui nous permet de passer du monde fermer de l'application vers le monde ouvert.

3.2.2.4.2. Sources WEB

Les données dans le Web sont généralement structurées par un langage de balises tel que XML associé à une grammaire tel que XML Schema qui permettent de structurer l'information mais ne suffisent pas à en définir toute la sémantique. Le problème majeur retrouvé est que contrairement aux bases de données, les données, ici, ne respecte pas une structure définie par un langage

de modélisation précis, alors que dans une base de données, toute données est structurée par un modèle conceptuel, ce qui rend la problématique de l'intégration de sources de données présentes sur le Web plus complexe. Les mêmes solutions apportées dans les bases de données ont été envisagées pour les sources de web.

3.2.3. L'ontologie et les systèmes d'intégration

Aujourd'hui, un des plus grands problèmes qui se pose est bien l'hétérogénéité des sources de données et cela quelque soit l'architecture du système d'intégration. L'hétérogénéité des sources de données est généralement classée en deux types : (i) hétérogénéité des schémas et (ii) hétérogénéité des données. L'hétérogénéité des schémas ou de structures apparaît lorsque différents modèles de données sont utilisés ou encore lorsque des schémas décrits dans un même modèle sont différents, en effet, les choix considérés pour concevoir le schéma de la source de données, tels que, par exemple, les noms des relations, des attributs, des types de données sont diffèrent d'une source à une autre. La présence de variations dans les structures est inéluctable puisque les humains pensent différemment et les sources de données sont conçues pour des besoins distincts. Alors que l'hétérogénéité de données apparaît lorsque différents vocabulaires et référentiels sont utilisés pour représenter les données. Ce type d'hétérogénéité est fréquent lorsqu'on souhaite intégrer des données provenant de différentes sources.

Pour traiter l'hétérogénéité des schémas et des données, la majorité des travaux menés se contentent de l'exploitation des informations présentes dans les schémas et dans les données.

Des techniques de comparaison syntaxique, telle que les mesures de similarité ont été utilisées.

Des heuristiques ont également été utilisées, par exemple, le fait que des noms des attributs comportent des informations sur leur sémantique. Pourtant ces derniers peuvent contenir des abréviations, des concaténations de plusieurs noms d'attributs et des homonymes. Par exemple, deux organisations qui souhaitent fusionner leur base de données ont dans leur schéma respectif des attributs

homonymes qui possèdent des sens différents. Dans l'une des deux bases de données, l'attribut « Dépenses » représente la somme dépensée alors que dans l'autre base il correspond au nombre de produits achetés. Les mêmes noms d'attributs peuvent ne pas représenter la même information et n'ont donc pas la même sémantique. Le problème de l'hétérogénéité des schémas et des données ne peut donc pas être résolu simplement avec ce type de méthodes. C'est pour cela que rendre explicite la sémantique précise des données intégrées est essentiel pour avoir une intégration sémantiquement correcte des données.

Une première idée pour rendre explicite la sémantique des données et des éléments du schéma des sources de données est de la spécifier de manière exhaustive pour toutes les données et les éléments des schémas considérés. Seulement il est impossible d'envisager de définir complètement la sémantique ou le sens d'une donnée ou d'un élément du schéma dans une base de données[121].

Les ontologies* peuvent contribuer à la résolution du problème de l'hétérogénéité sémantique. En effet, elles offrent une description formelle des concepts et de leurs relations existant dans certains domaines. Un utilisateur ou concepteur peut grouper et déclarer explicitement la sémantique des concepts et des données fournis par les sources.

Ainsi, en exploitant cette sémantique explicite et partagée, l'impact nuisible de l'hétérogénéité sémantique peut être réduit. Il s'agit dans ce cas d'une approche d'intégration de données fondée sur les ontologies. Une fois que les données des sources sont rendues conformes à l'ontologie, les requêtes peuvent être exprimées en termes de l'ontologie et les données peuvent être interrogées via une seule interface de requêtes. Dans le cas d'une intégration matérialisée des données, les requêtes exprimées en termes de l'ontologie sont évaluées directement sur l'entrepôt ; dans le cas d'une intégration virtuelle des données ces requêtes sont réécrites en fonction des vues sur les sources et les extensions de ces vues sont ensuite intégrées et combinées pour pouvoir répondre à l'utilisateur.

* Ontologie : Une spécification explicite et formelle d'une conceptualisation commune (d’après Thomas Gruber)

Les ontologies peuvent jouer plusieurs rôles dans le processus d'intégration. Elles peuvent être utilisées pour établir les liens sémantiques entre des éléments dans des sources différentes (exp. OBSERVER [07]). Elles peuvent aussi servir de modèle d'interrogation pour le système intégré lorsqu'elles sont utilisées pour décrire le schéma global (exp. SIMS [09], PICSEL [06]).

Trois architectures d'intégration à base d'ontologie de données sont distinguées en fonction de la façon dont les ontologies sont utilisées au sein d'une infrastructure d'intégration [122] :

3.2.3.1. Architecture utilisant une ontologie unique

Dans cette approche, chaque source à intégrer est liée à une seule ontologie de domaine globale. Ceci implique qu'une nouvelle source ne peut être décrite par sa propre ontologie. Tout ajout de source peut entraîner la modification de l'ontologie globale. Les systèmes d'intégration SIMS [09] et PICSEL [06] sont des exemples de cette approche.

3.2.3.1.1. Architecture utilisant plusieurs ontologies

Dans cette approche, chaque source à intégrer est décrite par sa propre ontologie, indépendamment des autres sources. Des correspondances entre les ontologies doivent être définies (ex. OBSERVER [07]). Ces correspondances peuvent se révéler parfois très complexes, notamment à cause de niveaux de granularité différents entre les ontologies.

3.2.3.1.2. Architecture utilisant une approche hybride

Dans cette approche, la sémantique de chaque source est décrite par sa propre ontologie. Cependant, les différentes ontologies sont connectées entre elles par un vocabulaire commun partagé (ex. le projet KRAFT [123]).

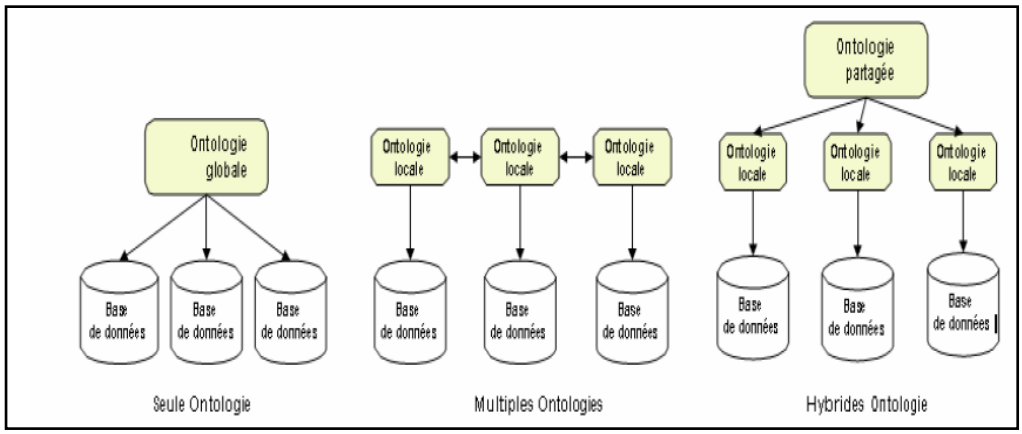

Figure 0.2 Différentes architectures d'intégration à base d'ontologie proposées par Wache et al. [122]

Le tableau suivant résume les avantages et les inconvénients de ces trois architectures :

	Architecture à une seule ontologie	Architecture à multiples ontologies	Architectures à ontologies hybrides
Effort d'implémentation	Facile	Couteux	Raisonnable
Hétérogénéité sémantique	Vues similaires au domaine	Supporte des vues hétérogènes	Supporte des vues hétérogènes
Ajout/suppression de sources de données	Besoin de quelques adaptations au niveau de l'ontologie globale	Il faut fournir l'ontologie de la nouvelle source, et la relier à au moins une des autres ontologies	Il faut fournir l'ontologie de la nouvelle source
Comparaison entre les multiples ontologies	—	Difficile en raison d'absence d'un vocabulaire commun	Simple puisque les ontologies utilisent un vocabulaire commun

Tableau 0.1 Avantages et inconvénients des différentes approches d'intégrations de données à base d'ontologie [122]

3.3. Bases de Données à Bas d'Ontologie (BDBO)

Une Base de Données à Base Ontologique est une source de données contenant :

- Une ontologie locale (les classes).
- Un ensemble de données (instances).
- Une relation entre chaque donnée et la notion ontologique qui en définit le sens.
- Des références (articulations) de l'ontologie locale à des ontologies externes (partagées).

Tout élément du schéma de la BDBO (classe ou propriété) doit appartenir à l'ontologie, donc le schéma, finalement, est un sous-ensemble de l'ontologie, chaque entité représentée correspond à une classe, et ses attributs correspondent aux propriétés applicables choisies.

Une seule Base de données contenant l'ontologie et les données à la fois d'où le même traitement est effectué partout.

Chaque donnée est liée à un élément ontologique qui définit son sens.

3.3.1. Schémas existants pour la représentation des ontologies et des données

Dans cette section nous allons présenter les représentations des ontologies ainsi que pour les données d'une base de données munie d'une ontologie, ces représentations nous permettent d'avoir une idée plus précise sur l'implémentation physique de ces deux partie au niveau de n'importe quel SGBD.

3.3.1.1. Représentation des ontologies

Il existe jusqu'à maintenant deux approches qui sont : (a) l'approche verticale où une seule table appelée « Triples » sous forme de <Sujet, Prédicat, Objet>. (b) Approche spécifique où on y trouve au moins les cinq tables suivantes :

- Range (Type, Property).
- Property (ID, Name).
- Domaine (Property, Class).
- Class (ID, Name).
- Subproperty (Super, Sub).
- Subclass (Super, Sub).

3.3.1.2. Représentation des données

Quand à la représentation des données on y trouve trois approches qui sont :

3.3.1.2.1. Approche verticale (générique)

Tout comme la représentation des ontologies, une seule table appelée « Triples » sous forme de <Sujet, Prédicat, Objet>.

3.3.1.2.2. Approche binaire

Séparation des propriétés et des instances :

✓ Représentation des identifiants des instances :

Trois façons existent :

- Table unique : est composé de : <ID, ClassID>
- ISA : Pour chaque classe ontologique, une table spécifique est associée pour stocker les identifiants des instances
- NOISA : Elle ressemble à la précédente mais sans la notion d'héritage, les tables des propriétés et des classes sont définies séparément sans aucune relation.

✓ Représentation des valeurs de propriétés : on y trouve deux approches :

- Approche triples : Une seule table à trois colonnes <Sujet, Prédicat, Objet>.
- Table par propriété : Une table spécifique pour chaque propriété ontologique, elle est constitué de deux colonnes <ID, Value>.

3.3.1.2.3. Approche hybride

Combinaison entre l'approche de représentation par triples et l'approche de représentation binaire. Une table triples pour chaque type de co-domaine (Entier, String, réel, etc.).

3.3.2. Quelques architectures des BDBO

Plusieurs bases de données à base d'ontologie ont été proposées depuis l'émergence des ontologies et leur utilisation dans le domaine des systèmes d'information, en ce qui suit nous allons résumer quelques une de ces bases de données.

3.3.2.1. Architecture Sesame

C'est une architecture générique indépendante de tout système de base de données, elle se base sur plusieurs modules :

- « RDFSAIL » : qui permet de rendre persistant le contenue de la BDBO, et qui offre une interface de programmation pour l'accès indépendant aux Bases de données.
- « Query Module » : Couche du moteur de requêtes du langage RQL.
- « Export Module » et « Admin Module » : permettent l'échange avec d'autres outils basés sur le RDF.

Sesame est se base sur un des deux schémas spécifiques suivants :

- PostgreSQLSAIL : qui comporte l'approche de représentation spécifique pour l'ontologie (les classes), l'approche ISA pour les données (instances) et l'approche binaire pour les valeurs des propriétés.
- MySQLSAIL : qui, lui, se compose de l'approche de représentation spécifique pour l'ontologie, l'approche hybride avec la variation ID pour les données

3.3.2.2. ICS-FORTH RDFSuite

Gère un nombre important de données, utilise plusieurs modules parmi eux :

- RSSDB : pour le stockage des données (méta-données) dans une base de données.
- RQL : Langage de requêtes, interrogation des données.
- VRP : Validation des données.

Deux schémas sont proposés par RDFSuite pour la représentation de l'ontologie et des données à base ontologiques, le premier conçu sous PostgreSQL qui adopte l'approche spécifique pour le stockage des ontologies et l'approche binaire pour les valeurs de propriétés et ISA avec héritage des tables pour les instances de l'ontologie, le deuxième utilise l'approche de représentation spécifique pour l'ontologie, l'approche de la table unique à deux colonnes pour les données, et l'approche binaire pour les valeurs de propriétés.

Trois contraintes sont exigées par RDFSuite :

- Toute propriété doit appartenir à un domaine ou un co-domaine.
- Le domaine ou le co-domaine d'une propriété doit être unique.
- La définition des classes et des propriétés doit être complète.

3.3.2.3. Jena Architecture

Elle est intégrée dans l'éditeur d'ontologie Protégé2000, elle est constituée des outils suivants :

- API de programmation pour la gestion des données du web sémantique.
- AQL : Langage de requêtes.
- Une structure relationnelle pour le stockage des données.
- Un parseur RDF/XML.

- Un moteur d'inférence.

- Outils de migration d'instances RDF d'un format à un autre.

La version Jena1 s'appuie sur l'approche verticale avec URI pour l'ontologie et les instances, tandis que Jena2 s'appuie sur l'approche verticale avec ID que se soit pour l'ontologie ou les instances.

3.3.2.4. Le système kwOWLer

Est un système de base de données à base ontologique, son architecture se compose de quatre modules :

- Kernel : permet la manipulation de la structure et de la sémantique définie par l'ontologie.

- Import/Export : pour l'importation dans la base de données et l'extraction des ontologies des différents modèles ontologiques.

- Storage : pour le stockage des ontologies et des données à base ontologique

- User : offre des API et des applications graphiques pour l'accès aux fonctionnalités du système.

Ce système opte pour une approche spécifique pour l'ontologie, les données sont stockées selon une approche binaire.

3.3.2.5. Le Modèle OntoDB

Ce modèle d'architecture a été conçu de manière à définir séparément, l'implémentation de l'ontologie de celle des données, tout en répondant à toutes les fonctionnalités et les caractéristiques des SGBDBO décrites précédemment.

Une base de données à base ontologique (BDBO), dans le modèle OntoDB est définie comme un quadruplet BDBO : < O, I, Pop, Sch >, avec :

– O représente son ontologie (O :< C, P, Sub, Applic >).

– I représente l'ensemble des instances de la base de données. La sémantique de ces instances est décrite par O en les associant à des classes et en les décrivant par les valeurs de propriétés applicables définies dans l'ontologie,

– Pop : C → 2^I, associe à chaque classe les instances qui lui appartiennent. Pop(c_i) constitue la population de c_i.

– Sch : C →2^P, associe à chaque classe c_i les propriétés qui sont applicables pour cette classe et qui sont effectivement utilisées pour décrire tout ou partie des instances de Pop(c_i).

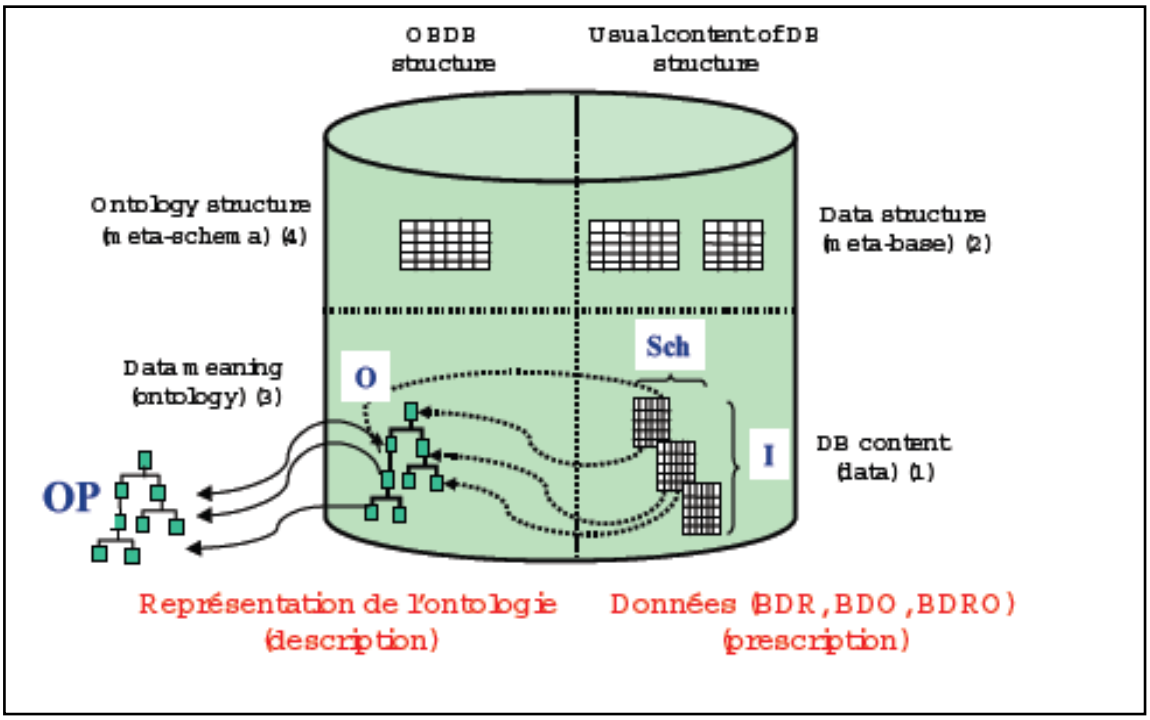

Figure 0.3 Architecture d'une base de données à base ontologique selon le modèle OntoDB [63]

La figure 2.8 représente l'architecture logique d'une BDBO. Ce modèle d'architecture, appelé *OntoDB*, permet de représenter simultanément des ontologies de domaines, décrites en termes de classes et de propriétés, et des objets du domaine

Le modèle comporte quatre parties :

1. La partie *ontologie* permet de représenter complètement les ontologies de domaines définissant la sémantique des différents domaines couvert par la BD, ainsi, éventuellement, que l'articulation (représentée sous forme de "*mapping*" [46] [124]) de ces ontologies avec des ontologie externes.

2. La partie *méta-schéma* permet de représenter, au sein d'un modèle réflexif, à la fois le modèle d'ontologie utilisé et le méta-schéma lui-même.

3. La partie *méta-base* permet de représenter le schéma de représentation des objets du domaine couvert par les ontologies.

4. La partie *données* représente les objets du domaine; ceux ci sont décrits en termes d'une classe d'appartenance et d'un ensemble de valeurs de propriétés applicables à cette classe.

3.3.3. Caractéristiques des SGBDBO :

Un système de gestion des bases de données à base ontologique comporte, généralement, les fonctionnalités et les caractéristiques suivantes :

- Respect d'un standard (suivant un des Langages de définition d'ontologies connus)
- Echange de données : importation et extraction de données et d'ontologies.
- Langage de requêtes : RQL, SPAQL permettant la création des concepts et d'instances, et l'interrogation de la base de données.
- API d'accès aux ontologies et aux données (interfaces, programmes…).

Toutefois, d'autres fonctionnalités très importantes sont souhaitable dans un SGBDBO, telles que :

- Représentation des données dans un schéma : afin de rendre le traitement sur une masse volumique de données plus rapide et plus aisée en offrant le moyen d'utiliser les mêmes mécanismes d'optimisation de requête offerts par les SGBD traditionnels.

- Le versionnement : afin de garder une trace de toutes les versions antécédentes d'une ontologie évolutive dans le temps, la gestion des versions s'avère nécessaire dans quelques applications.

- Cycle de vie des instances: comme les ontologies, les instances (données) évoluent, ou bien lors de l'évolution de l'ontologie, ou bien par rapport aux opérations sur les données, pour cela on doit garder une trace sur l'instance et son évolution, c'est ce qu'on appelle de cycle de vie d'une instance.

3.3.4. Synthèse

Les bases de données à base ontologique (BDBO) répondent aux besoins de gérer une quantité volumineuse de données décrites par référence à des ontologies (données à base ontologique). Elles permettent de stocker, dans une base de données, des ontologies et les instances qu'elles décrivent. Les instances peuvent être structurées par un schéma comme dans une base de données traditionnelle. Les éléments de ce schéma sont alors liés à une ontologie pour en définir la sémantique.

Le schéma des instances peut être construit à partir de l'ontologie. Il peut également avoir été conçu indépendamment d'une ontologie (indexation sémantique). La tendance actuelle des BDBOs est de gérer des données de plus en plus volumineuses, de séparer ontologies et données, et de permettre d'importer/exporter ontologies et données représentées avec différents modèles d'ontologies.

3.4. Conclusion

Dans ce chapitre nous avons vu brièvement la notion d'ontologie, et ses domaines d'utilisation, puis nous avons détaillé son utilité lors de l'intégration de données, nous avons constaté que l'utilisation d'ontologies conceptuelles apparaît comme la seule approche assurant l'automatisation du processus d'intégration sémantique de données.

L'utilisation d'ontologies conceptuelles pour développer des systèmes d'intégration devient populaire. Nous avons présenté comment les approches existantes utilisent des ontologies conceptuelles pour résoudre les conflits sémantiques de données.

Puis nous avons les bases de données à base d'ontologies, ces bases de données facilitent d'intégration automatique des données.

Dans la partie suivante, nous allons présenter notre approche proposées, les algorithmes d'intégration ainsi que la mise en œuvre de notre architecture pour validation.

PARTIE 2

NOTRE ARCHITECTURE PROPOSEE

En général, les approches d'intégration de données sont classées en approches matérialisées et virtuelles. Dans la première approche, une vue intégrée matérialisée est développée et maintenue, une base physique est créée et des données sont extraites dans cette base à partir des sources locales. Dans l'approche de médiation (virtuelle) [03], les données résident dans les sources locales et un ou plusieurs schémas virtuels sont définis qui constituent l'interface entre les utilisateurs et le système d'intégration. Les utilisateurs posent leurs requêtes au travers de ce schéma virtuel. Alors profiter des avantages des deux approches, les approches hybrides d'intégration de données ont été conçues [03].

Cette partie est constituée de trois chapitres : Le premier chapitre est consacré à la présentation de notre approche d'intégration Hybride à base d'ontologie ainsi que tous ces modules en détail, alors que le chapitre 5 est destiné à décrire tous les scénarii d'intégration propsées avec des exemples d'application pour chaque scénario. Afin de valider notre architecture ainsi que les scénarii d'intégration nous avons implémenté un prototype pour la validation au niveau du chapitre 5

CHAPITRE 4. APPROCHE D'INTEGRATION HYBRIDE DE BASE DE DONNEES A BASE D'ONTOLOGIE

4.1. Introduction

On assiste depuis quelques années à l'émergence de nouvelles applications qui ont besoin de partager des informations entre différents systèmes. C'est le cas du e-gouvernement, du e-learning, du e-commerce, de la bioinformatique ou encore des bibliothèques électroniques. Or, dans ce contexte, les systèmes d'informations, conçus et développés par des organisations différentes, constituent généralement des sources de données autonomes et hétérogènes.

De ce fait, l'interopérabilité entre ces systèmes d'information est complexe puisque les applications doivent être adaptées pour pouvoir déterminer, pour chaque requête, les sources de données pertinentes, la syntaxe requise pour l'interrogation, la terminologie (concept) propre à la source, et pour pouvoir combiner les fragments de résultats issus de chaque source en vue de construire le résultat final. Ce processus d'adaptation peut être plus ou moins complexe : exploitation des résultats d'une source pour interroger une autre, élimination des redondances, etc.

Les approches matérialisées et virtuelles sont deux extrêmes d'un large spectre possible pour un scénario d'intégration de données [94].

Dans certaines applications, nous avons besoin de profiter des possibilités de ces deux approches d'une façon flexible. Les approches hybrides d'intégration de données [03] ont été conçues pour satisfaire de telles applications.

Alors que l'intégration hybride des sources de données hétérogènes, autonomes et réparties est une solution pour qui nous permettra de profiter des possibilités de ces deux approches d'une façon flexible [03].

Nous nous intéressons particulièrement à l'intégration sémantique qui représente un défi aussi bien pour les chercheurs que pour les industriels. En effet, l'intégration sémantique suscite de plus en plus d'intérêt avec l'émergence du web sémantique. Grâce à ce dernier, les systèmes devraient pouvoir échanger des informations et des services dans un contexte sémantiquement riche.

Nous proposons une solution pour l'intégration de systèmes d'informations hétérogènes, fondée sur une architecture mixte (médiateur et entrepôt de données) et les ontologies pour résoudre les conflits sémantiques et syntaxiques.

Suite à l'état de l'art effectué et après recensement des différentes solutions proposées selon l'approche hybride, notre contribution à travers ce travail de recherche est de présenter notre solution d'intégration relative à l'environnement étudié à savoir une solution hybride selon les deux approches : médiateur et entrepôt appliquée au domaine hospitalier.

Cette vision est bâtie sur le fait de :

- La Nature spécifique des sources de données
- L'optimisation des résultats Solution hybride

Dans ce chapitre, nous proposons notre solution via l'approche hybride à l'intégration de sources de données à base ontologique combinant les avantages des deux approches médiateur et entrepôt dans une large échelle. A notre connaissance, il n'y a pas eu à ce jour une telle solution d'intégration pour un tel type de sources de données.

4.2. Présentation de l'approche

Ce travail s'inscrit dans le cadre du projet OntoDB, en utilisant l'approche hybride comme méthode d'intégration.

Cette approche est basée sur les hypothèses suivantes :

(a) Il existe une ontologie de domaine (ontologie partagée) recouvrant la totalité des termes consensuels, nommée CIHO [49].

(b) Chaque source locale peut se définir en terme d'une ontologie qui lui est propre (sources de données à base ontologique).

(c) Chaque ontologie locale référence a priori autant que cela est possible l'ontologie de domaine [110].

Le processus d'intégration intègre d'abord les ontologies ensuite les données. L'ontologie partagée peut jouer le rôle d'un schéma global et chaque source pourrait être vue comme un sous schéma de cette ontologie. Cette situation est quasi similaire aux bases de données réparties homogènes, où une base de données centralisée pourrait être décomposée en plusieurs fragments qui seront placés sur des sites répartis.

4.3. Objectifs à atteindre

Les objectifs majeurs de cette étude se décomposent en trois parties :

1. Processus d'intégration

   - ✓ Etude des scénarii d'intégration
   - ✓ Proposition d'une architecture hybride

2. Traitement de requêtes

3. Optimisation et raffinement de requêtes

4. Implémentation de l'architecture

4.4. Description des sources de données

Dans un processus d'intégration, les sources de données utilisées peuvent être de nature structurée, semi structurée ou bien non structurée. Dans notre approche les données sont de type structuré du fait qu'on s'intéresse aux bases de données. Les bases de données sont structurées par leur modèle conceptuel. Néanmoins, il ne s'agit pas de bases de données connues traditionnellement mais de bases de données à base ontologique, concept développé au sein du LISI, dans le cadre du projet OntoDB [63].

Rappel : Une Base de données à base ontologique consiste en une base de données pouvant stocker à la fois :

- Le contenu usuel d'une base de données
- L'ontologie qui décrit la signification de ces données
- Contrairement aux bases de données classiques, formées principalement de deux parties (données et méta base). Les bases de données à base ontologique sont formées de quatre parties, de ce fait elle tire son appellation d'architecture 4-quart :
    - Meta Base : ensemble de tables systèmes décrivant la structure des tables de données
    - Contenu : données ou instances du modèle conceptuel
    - Ontologie : dictionnaire contenant la signification (sémantique) des données
    - Meta Schéma : structure décrivant le modèle de l'ontologie utilisée. Il est pour l'ontologie ce que la méta-base est pour les données.

L'intégration de ce type de données (sources de données à base ontologique) a déjà fait l'objet des travaux de thèse de Mr Dung Nugyen [30] « l'intégration via l'approche matérialisée dite entrepôt », et l'approche médiateur était l'une de ses perspectifs qui a étudié dans travaux de mémoire de magister de Melle Amel Boussis [125].

Nous nous sommes intéressées à adapter ces travaux sur une approche hybride afin de répondre aux problèmes trouvés dans les deux approches précédentes.

4.5. Processus d'intégration

Dans la thèse de Dung Nugyen [30], une approche permettant la gestion des évolutions ontologiques était proposée, afin de bénéficier de cet apport, nous proposons une solution dite hybride combinant les avantages des deux approches : médiateur et entrepôt de données.

Chaque source de données possède son schéma local à travers son ontologie locale (source à base ontologique). Le schéma global est représenté par l'ontologie partagée (de domaine). Les ontologies locales doivent référencer *autant que cela est possible* l'ontologie partagée [110]. Les mises en correspondance entre les deux schémas peuvent être réalisées selon des scénarios d'intégration.

A travers différents scenarii, nous allons montrer comment le processus de fragmentation pourrait être utilise dans notre approche.

Pour cela, nous avons étudiés les scénarios d'intégration suivants :

1. *FragOnto :* Chaque source locale est définie comme un fragment (pouvant être horizontal vertical ou mixte) de l'ontologie partagée (un fragment horizontal résulte d'une opération de sélection appliquée sur l'ontologie partagée, un fragment vertical résulte d'une opération de projection appliquée sur l'ontologie partagée, un fragment mixte est une combinaison de la fragmentation horizontale et verticale).

2. *ProjOnto :* chaque source définit sa propre ontologie (elle n'instancie aucune classe de l'ontologie partagée). Par contre l'ontologie locale référence l'ontologie partagée en respectant la condition SSCR (Smallest Subsuming Class Reference) [110]. Dans ce scénario, on souhaite néanmoins intégrer les instances de chaque source comme des instances de l'ontologie partagée ;

3. *ExtendOnto :* chaque ontologie locale est définie comme dans le scénario ProjOnto, mais l'on souhaite enrichir automatiquement l'ontologie partagée. Ensuite toutes les instances de données sont intégrées, sans aucune modification, au sein du système intégré.

4.6. Représentation formelle de l'approche

Nous allons représenter formellement dans ce qui suit notre approche d'intégration hybride de données fondée sur les ontologies.

4.6.1. Représentation de l'ontologie

Une Ontologie O (Locale ou Partagée) est définie formellement comme un quadruplet : $O :< C, P, Sub, Applic >$, avec:

- C : l'ensemble des classes utilisées pour décrire les concepts d'un domaine donné.
- P : l'ensemble des propriétés utilisées pour décrire les instances de l'ensemble des classes C.
- Sub : est la relation de subsomption (is-a et is-case-of), qui, à chaque classe Ci de l'ontologie, associe ses classes subsumées directes.
- Applic : associe à chaque classe de l'ontologie les propriétés qui sont applicables pour chaque instance de cette classe.

4.6.2. Représentation des sources de données à Base d'Ontologie

Nos sources de données sont en effet des *Bases de Donnéess à Base Ontologique (BDBOs)* chaque BDBO est définie formellement comme un quadruplet: $BDBO :< O, I, Pop, Sch >$, avec :

- O : Représente son ontologie ($O :< C, P, Sub, Applic >$).
- I : représente l'ensemble des instances de données de la base de données. La sémantique de ces instances est décrite par O.

- Pop: associe à chaque classe les instances qui lui appartiennent (directement ou par l'intermédiaire des classes qu'elle subsume). $Pop(C_i)$ constitue donc la population de C_i.

- Sch : associe à chaque classe C_i les propriétés qui sont applicables pour cette classe et qui sont effectivement utilisées pour décrire tout ou partie des instances de $Pop(C_i)$.

4.6.3. Représentation de l'approche d'intégration à base ontologique

Soient $OP :< C, P, Sub, Applic >$ l'ontologie partagée dite de domaine, et S={S_1, ... , S_n} un ensemble de BDBOs participantes au processus d'intégration. Dans une intégration à priori, on intègre d'abord les ontologies ensuite les données.

L'intégration des ontologies est effectuée et exprimée à l'aide de relations de subsomption entre classes de différentes ontologies et d'importation de propriétés. Nous supposons dorénavant ce qui suit :

(i) Chaque source S_i contient sa propre ontologie locale : O_{Li} : < C_i, P_i, Sub_i, $Applic_i$ >, en référençant l'ontologie partagée O_P. Ces références sont stockées dans la source S_i et constituent une articulation M_i entre O_P et O_{Li}.

(ii) Chaque classe C_i a son schéma $Sch_i(C_i)$, et ses instances $Pop_i(C_i)$.Une source S_i articulée avec une ontologie partagée O_P peut être formulée comme suit: S_i :<O_{Li}, I_i, Sch_i, Pop_i, M_i >.

Concernant les classes, nous supposons ce qui suit :

(i) chaque classe de chaque S_i référence sa (ou ses) plus petites classes subsumante(s) dans l'ontologie partagée,

(ii) chaque classe de chaque S_i importe de ses classes subsumantes dans l'ontologie partagée les propriétés qu'elle souhaite utiliser, en préservant leurs identifications,

(iii) toutes les classes de S_i, ainsi que toutes les propriétés de S_i qui ne sont pas identifiées avec des propriétés importées, sont spécifiques de la source S_i.

4.7. Architecture du système hybride d'intégration de données « HybOnto »

Notre recherche se porte sur l'intégration de bases de données à base ontologies, (BDBO), afin de résoudre les problèmes d'hétérogénéité sémantique, on a mis au point une architecture semi-matérialisée d'intégration de données a base d'ontologie.

L'approche adoptée dans la création de l'ontologie est l'approche hybride, suivant la méthode définie par [26], Notre architecture est faite en trois étapes : construire les ontologies locales (chaque base de données doit être munie à préalable de son ontologie ou encore la construire), puis construire le vocabulaire commun et finalement la définition du mapping entre le schéma globale et les schémas locaux.

4.7.1. Architecture logicielle du système HybOnto:

En s'inspirant des travaux menés par Ahmed Alasoud et al. [115] (voir la section4.4 du deuxième chapitre), nous avons construit notre système d'intégration hybride à base d'ontologie illustré dans la figure suivant.

La plateforme est divisée en deux grandes parties, la partie virtuelle et la partie matérialisée mais qui ne sont visibles à l'utilisateur que par une et une seule interface nommé Interface Utilisateur (IU).

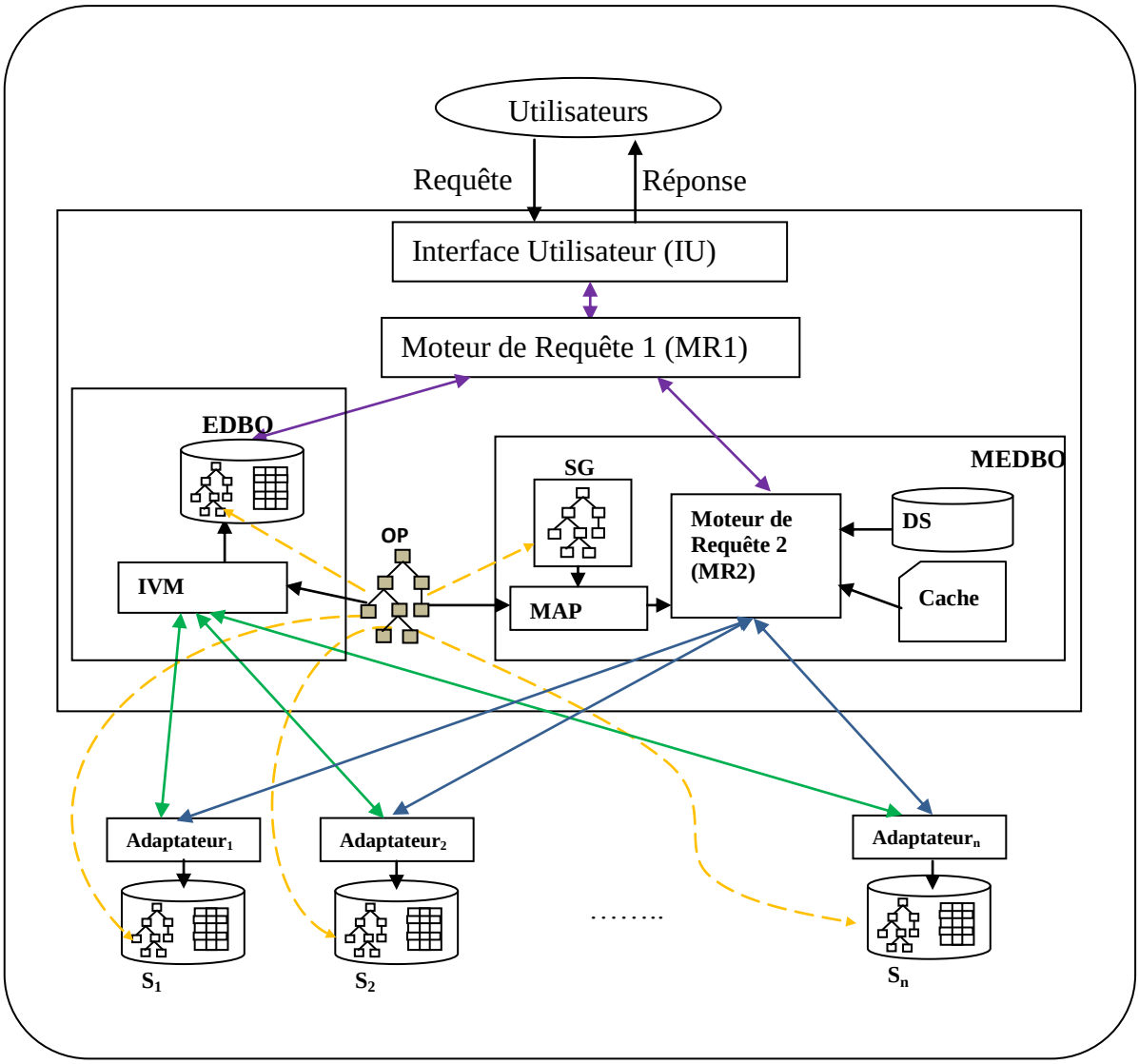

Figure 0.1 Architecture Hybride d'Intégration de Bases de Données à Base d'Ontologie à priori d'Ontologie « HYBOnto »

4.7.2. Les principaux composants du système HybOnto

Notre système d'intégration a une architecture à quatre niveaux qui sont :

1. Le premier niveau : concerne la partie utilisateur et comporte :
    a. Une interface utilisateur
    b. Deux moteurs de requêtes
2. Le deuxième niveau : qui se compose de deux sous parties :
    a. Le médiateur
    b. L'entrepôt de données

c. L'ontologie partagée

3. Le troisième niveau : se compose des adaptateurs
4. Le quatrième niveau : est l'ensemble des sources de données locales (S_i)

4.7.2.1.1. Le premier niveau (couche utilisateur)

Comme nous venons de citer, le premier niveau se compose d'une interface utilisateur et de deux moteurs de requêtes :

a- L'Interface Utilisateur (IU)

Elle permet de recevoir les requêtes des utilisateurs, de les envoyer vers le moteur de recherche, de récupérer les résultats et de les afficher, cette interface permet une sélection dans la recherche de telle sorte qu'on peut viser la recherche directe sur les instances (les données) ou encore la recherche d'ontologie ou de termes.

b- Les moteurs de requêtes

Nous avons utilisé deux moteurs de requêtes, le premier est pour rechercher l'emplacement des données demandées dans le médiateur et/ou l'entrepôt de données, le deuxième est pour localiser l'emplacement des données au niveau des sources de données locales dans le cas du médiateur où les données sont toujours dans leur emplacement original.

- *Le moteur de requêtes (MR1)*

Ce moteur de requêtes permet de filtrer la requête émise par l'utilisateur tout au début c'est-à-dire de voir si la requête sera traitée par l'entrepôt de données (résultat entièrement récupérable depuis les vues matérialisées) ou par le médiateur (résultat recueilli depuis les sources de données), ou encore par les deux modules à la fois i.e. que la requête (ou les sous requêtes) sera (seront) envoyée (envoyées) au deux modules, le principe du moteur de requête est simple, il recherche dans le schéma de l'entrepôt de données, si la requête est entièrement satisfaite à ce niveau c'est bon, sinon, il renvoi la requête au deuxième Moteur de Requête (MR2) qui interrogera le médiateur (qui récupérera

les résultats des sources de données locales), le résultat sera réécrit au niveau du MR1 avant affichage des résultats.

- *Le moteur de recherche (MR2) :*

De même que MR1, la tâche de MR2 est de décomposer les requêtes en sous requêtes, localiser les sources de données pertinentes, réécrire les requêtes dans le cas ou plusieurs sources sont ciblées, à la récupération des données, il reconstruit les réponses et l'envoi au MR1 qui s'en occupe de l'affichage des résultats.

Remarque :
Il faut noter que le langage d'interrogation utilisé ici est tout simplement le SQL. Au niveau de l'implémentation, et afin d'améliorer le temps de réponse, nous avons fusionné les deux moteurs de recherches dans un seul moteur de recherche, (voir 4§12).

4.7.2.1.2. Le deuxième niveau

Cette couche englobe les deux parties les plus importantes du système à voir le médiateur et l'entrepôt de données à base ontologique*.
Pour bien répondre aux besoins d'un système d'intégration, cette couche globale doit :

1- Accepter les requêtes des utilisateurs.
2- Réécrire (décomposer) et optimiser les requêtes (optimisation repartie).
3- Envoyer les plans d'exécution à faire exécuter par les adaptateurs des différentes sources.
4- Regrouper/combiner les résultats.
5- Assurer le passage à l'échelle lors de l'ajout ou la mise a jour d'une source.

* L'architecture physique de l'entrepôt de données (EDBO) est similaire à celle d'une base de données à base ontologique(BDBO), l'EDBO est vu plutôt comme une grande base de données comportant des vues matérialisées sur les sources de données.

a- La sous couche Médiateur (MEDBO)

Le médiateur à base d'ontologie permet d'interroger des sources de données à base ontologique de natures multiples, hétérogènes et éventuellement réparties dans le cas ou les données recherchées ne se trouvent pas au niveau de l'entrepôt de données[**].

Les sources sont relatives à un domaine d'application donné. Le médiateur comporte une base de connaissance spécifique au domaine du médiateur (qui est dans notre cas, le domaine hospitalier), la structure de cette base ressemble le plus à celle d'une base de données à base ontologique (OntoDB).

Cette base se compose d'une ontologie de domaine ou schéma global, et des descriptions du contenu des sources de données.

Le médiateur comporte un schéma global des sources (SG), une base de données relationnelle pour la description des sources (DS) et un cache dont le rôle est, d'une part, la sauvegarde momentanées des résultats intermédiaires d'une requête reçus par les sources de données (S_i) interrogées afin de permettre au moteur de requête de reconstruire le résultat final escompté par l'utilisateur, et d'autre part, il pourra servir de stocker les résultats des requêtes fréquemment posées.

- *Le schéma global (SG)* : Le schéma global (SG) Représente l'ontologie du médiateur appelée (O_{MED}) conçue par notre algorithme qui est la fusion de l'ontologie partagée (O_P) et les ontologies locales (O_{Li}) selon les critères sélectionnés dans notre algorithme[*], le schéma global permet d'avoir une vue globale sur les sources de données ou une partie des sources de données, en outre, il fournit un vocabulaire unique pour l'expression des requêtes des utilisateurs et pour la description des contenus des sources ce qui permet de présenter et d'extraire les données sources, l'utilisateur pose ses requêtes en terme des relations dans le schéma global.
- *Le Descripteur des Sources (DS) :* Le descripteur des sources comporte l'ensemble des vues abstraites des sources de données locales, dans notre cas, les sources de données sont toutes du même format c'est-à-dire des

[**] Un ensemble de données est stocké au niveau de l'entrepôt de données et cela selon des critères fixes que nous détailleront dans le chapitre suivant.
[*] Voir le chapitre suivant

bases de données à base ontologique, ceci ne veux pas dire qu'elles ne sont pas hétérogènes, autonomes ou encore réparties.

- *Le cache :* Le médiateur utilise le cache pour stocker les résultats intermédiaires des requêtes envoyés aux sources de données locales (S_i) afin de réinterroger tous les résultats obtenus et avoir une réponse plus précise à donner à l'utilisateur. On peut aussi utiliser le cache pour stocker les données fréquemment demandées, et cela nous permettra de mettre en place un système d'apprentissage afin d'améliorer par la suite, l'algorithme d'intégration des données.

b- L'Entrepôt de Données à Base d'Ontologie (EDBO)

L'entrepôt de données contient les données (instances des sources de données locales) intégrées par notre algorithme, l'entrepôt de données est muni aussi de sa propre ontologie dite O_{ED} construite à partir de l'ontologie partagée (O_P) lui fusionnant les ontologies locales (O_{Li}) selon nos scénarii d'intégration, l'ontologie O_{ED} permet la description des données de l'entrepôt (ces données qui ne sont en réalité que les vues matérialisées des données locales), l'apport principal de cet entrepôt de données dans notre architecture est le gain de temps de réponse pour les requêtes utilisateur.

La sous couche entrepôt de données comporte deux composants, l'entrepôt de données lui-même qui n'est qu'une grande base de données et l'intégrateur des vues matérialisées.

- *L'Entrepôt de données EDBO :* L'entrepôt de données a la même architecture d'base de données à base ontologique (architecture 4-quart) permettant de collecter, ordonner, journaliser et stocker des données locales destinées à être intégrées à son niveau.
- *L'Intégrateur des Vues Matérialisées (IVM) :* L'intégrateur des vues matérialisées permet l'intégration des données matérialisées au niveau de l'entrepôt de données ainsi que la mise à jour et la maintenance de l'entrepôt de données.

c- L'ontologie partagée

L'ontologie partagée est une ontologie de domaine, elle modélise le domaine d'application et représente un vocabulaire commun pour toutes les ontologies locales (y compris l'ontologie de l'entrepôt de données et le schéma global du médiateur), ce qui facilite l'intégration des sources de données et de résoudre le problème d'hétérogénéité sémantique. Dans notre approche, nous traitons des ontologies formelles canoniques.

L'ontologie de domaine doit comprendre un ensemble de classes d'objets du domaine, chaque classe étant caractérisée par des propriétés et pouvant être reliée par des relations spécifiques au domaine à d'autres classes. Pour une classe donnée, le modèle précise la classe qui la généralise (subsumée), éventuellement ses propriétés spécifiques ou bien l'ensemble des propriétés nécessaires et suffisantes d'un objet pour appartenir à cette classe. Un des problèmes à résoudre pour automatiser la construction des systèmes d'intégration concerne la construction automatique de l'ontologie. Dans notre cas et afin de mettre en place de notre architecture, nous nous somme basées sur le modèle CIHO [126] pour mettre en place une petite ontologie de domaine nommée « HOnto » juste pour le test et la validation de notre approche, quoique la construction d'une ontologie du domaine hospitalier reste l'une de nos perspectives.

4.7.2.1.3. Le troisième niveau : « Les adaptateurs (wrappers)

Classiquement, les adaptateurs traduisent les sous requêtes émises par le médiateur en langages des sources de données spécifiques, les résultats seront ensuite transmis au médiateur qui s'occupe de les présenter.

Dans notre cas, en plus du rôle principal des adaptateurs, ils jouent aussi le rôle de l'intégrateur des vues matérialisées (IVM), de sorte à ce que les données à mettre à jour sont récupérées par l'adaptateur et traduit au langage de l'entrepôt de données, idem pour la maintenance des sources de données, ou les adaptateurs seront chargés de tout échange entre les sources et l'IVM.

4.7.2.1.4. Le quatrième niveau « Les sources de données (S_i) »

Les sources de données sont représentées autant que bases de données à base d'ontologie, c'est-à-dire outre les données, on trouve le modèle de conception, l'ontologie et le méta-modèle de l'ontologie, les sources de données sont conçues d'après le modèle OntoDB.

4.7.3. Représentation des connaissances dans HYBOnto (Mapping de données)

HYBOnto est un système d'intégration hybride qui permet d'interroger des sources d'informations multiples, hétérogènes et éventuellement réparties. Les sources sont relatives à un domaine d'application donné (le domaine hospitalier). HYBOnto comporte deux moteurs de requêtes génériques utilisables quel que soit le domaine d'application et une partie base de connaissances spécifique au coté médiateur.

La partie base de connaissances se compose d'une description du domaine, dite « ontologie du domaine » ou schéma global, et des descriptions du contenu des sources d'informations. HYBOnto suit une approche GaV (Global asView).

La représentation des connaissances dans notre système est supposée faite dans un langage bien défini nommé : HOntoL. L'ontologie du domaine est représentée dans le formalisme HOntoL. Le système possède également un langage de vues et un langage de requêtes. Dans les sections suivantes nous présentons ces deux langages.

4.7.3.1. Langage de vues

Dans HYBOnto, la description du contenu d'une source est exprimée en termes de vues. Il s'agit de représenter chaque source S à partir du vocabulaire V_S constitué d'autant de noms de relations locales v_i que de relations du domaine dont on sait que la source S fournit des instances. Ces relations locales sont appelées des vues. La description du contenu d'une source S en terme de ses vues est constituée de :

i- Un ensemble d'implications logiques de la forme $v_i \rightarrow \mathrm{p}$ reliant chaque vue v_i à la relation du domaine p dans l'ontologie dont elle peut fournir des instances.

ii- Un ensemble de contraintes d'intégrité et de typage sur les vues, de la forme :

(a) $v \subseteq C$ ou C est une expression de concept, avec, pour chaque vue v, au plus une contrainte de cette forme.

(b) $l_1(\overline{X_1}), \ldots\ldots, l_n(\overline{X_n}) \longrightarrow \perp$ où chaque $l_i(\bar{X}_i)$ est une vue $V_1(X_i)$ et

(c) $l_1(\overline{X_1}), \neg l_2(\bar{X}) \longrightarrow \perp$ où l_1 et l_2 sont des vues et $X \in \bar{X}$

Exemple

Soit la source PathoSource décrivant les pathologies, leur code, et différentes propriétés. Cette source permet d'obtenir des instances du concept PathoSegment ainsi que des instances des rôles PathoNumberAssociate, CountryAssociate.

4.7.3.2. Correspondances entre schémas – Global as View

L'approche utilisée pour les correspondances entre les schémas des sources locales et le schéma global est l'approche GaV (Global as View). C'est une approche ascendante depuis les sources vers le schéma global. La réécriture des requêtes est simple, une requête sur le schéma global se traduit en termes de schémas de sources en remplaçant les vues par leurs définitions (dépliement des requêtes). La requête dépliée est évaluée sur les sources.

Généralement, l'inconvénient de cette approche est bien le fait qu'elle ne permet pas le passage à l'échelle, aussi l'ajout de nouvelles sources est difficile (nécessité de reconstruire le schéma global) donc évolution difficile.

Mais dans notre cas, ce problème est résolu du fait de l'hypothèse suivante : «Chaque ontologie locale référence a priori autant que cela est possible l'ontologie de domaine» [110], effectivement, quand l'ontologie locale référence autant que possible l'ontologie de domaine, l'ajout de nouvelles sources de données ne modifie pas le schéma global car il y a un référencement a priori.

4.7.4. Traitement de requêtes

A travers une interface de communication, l'utilisateur formule sa requête dans un langage de requêtes (SQL). HYBOnto possède un langage de requêtes permettant, d'exprimer, en termes de l'ontologie du domaine, respectivement, le contenu et les requêtes des utilisateurs.

Notre système permet à un utilisateur de poser une requête utilisant les termes du vocabulaire de l'ontologie ou encore des données tout simplement. L'entrepôt de données et/ou les sources seront ensuite interrogés (après traduction de la requête par des adaptateurs). Les requêtes sont des requêtes conjonctives de la forme :

$$Q(\bar{X}) \longleftarrow \mathrm{p}_1(\overline{X_1}, \overline{Y_1}, \overline{c_1}) \wedge \ldots \wedge \mathrm{p}_n(\overline{X_n}, \overline{Y_n}, \overline{c_n}) \qquad 0.1$$

Ou les p_i sont des atomes concepts, des atomes rôles ou des atomes ordinaires.

Leur conjonction constitue le corps de la requête et $Q(X)$ est appelée la tête de la requête.

Les variables du vecteur $X = X_1, \ldots, X_n$ sont appelées variables distinguées de la requête et représentent les variables dont l'utilisateur veut obtenir des instances lorsqu'il pose la requête.

Les variables du vecteur $Y = Y_1, \ldots, Y_n$ sont des variables existentielles, et les constantes c_i e servent à exprimer des contraintes sur les variables distinguées.

Les requêtes sont reçues par le Moteur de requêtes, ce dernier permet :

- La décomposition de la requête principale en sous requêtes
- La localisation des sources pertinentes
- La réécriture et reformulation de requêtes
- La construction de la réponse

Le moteur de requêtes comprend : Analyseur – Optimiseur , dans le chapitre 6 nous décrivons en détail notre algorithme de recherche.

Exemple :

La requête suivante exprime le dénombrement de la HTA (Hyper Tension Artérielle) au niveau d'Algérie et de Tunisie jusqu'au 01 janvier 2007

Q(X) ← (PathoSegment)(X)

(PathoNumberAssociate)(X,Y), (PathoNumber)(Y)

(CountryAssociate)(X, Algérie)

(CountryAssociate)(X, Tunisie)

(DateAssociate)(X,01/01/2007)

Dans cette requête, il existe :

- ✓ Une variable distinguée, X, signifiant que l'utilisateur veut obtenir des instances du concept PathoSegment.
- ✓ Une variable existentielle, Y, obligeant a la deuxième composante du rôle PathoNumberAssociate à être de type PathoNumber.
- ✓ Trois constantes : Algérie, Tunisie et 01/01/2007 qui contraignent respectivement les deux pays concernés par l'étude et la date.

4.8. Discussion

L'utilisation d'ontologies formelles dans notre approche permet une intégration automatique des sources à l'opposé des ontologies linguistiques où l'intégration est partielle (semi-automatique) ou même manuelle. Les ontologies formelles constituent des spécifications explicites de conceptualisations de domaines. Elles permettent de définir formellement les concepts ainsi que les relations inter concepts du domaine étudié.

L'utilisation d'ontologies locales au niveau des sources de données et d'une ontologie globale dite de domaine au niveau de la couche supérieure rend plus facile l'interopérabilité entre les données issues de sources distinctes. Plus précisément, les ontologies permettent l'intégration de sources de données (conflits sémantiques), l'utilisateur interroge le système en utilisant le vocabulaire de l'ontologie et ce qui permet une formulation exacte et raffinée des requêtes (conflits structurels).

Notre approche apporte les avantages suivants :

- Une intégration automatique via les ontologies formelles de sources de données spécifiques : des bases de données à base ontologique.

- Un compromis entre le temps de réponse, la pertinence des résultats, et la mise à jour des données est intéressant.

Toutefois, cette approche a quelques points faibles qui sont :

- L'utilisation de deux moteurs de requêtes s'avère très lourde, d'où la nécessité d'englober les deux moteurs de requêtes dans un seul et unique module.

- Le passage à l'échelle est difficile et peut même mener à une réinitialisation complète du système à cause de l'algorithme de l'intégration que nous avons proposé et qui est détaillé dans le chapitre suivant et aussi à cause de la centralisation du médiateur, la solution à ce problème reste une de nos perspectives.

4.9. Amélioration de l'approche

Afin de résoudre le premier inconvénient de notre approche à savoir l'utilisation d'un seul moteur de requête, nous avons proposé une nouvelle architecture améliorée qui est la suivante :

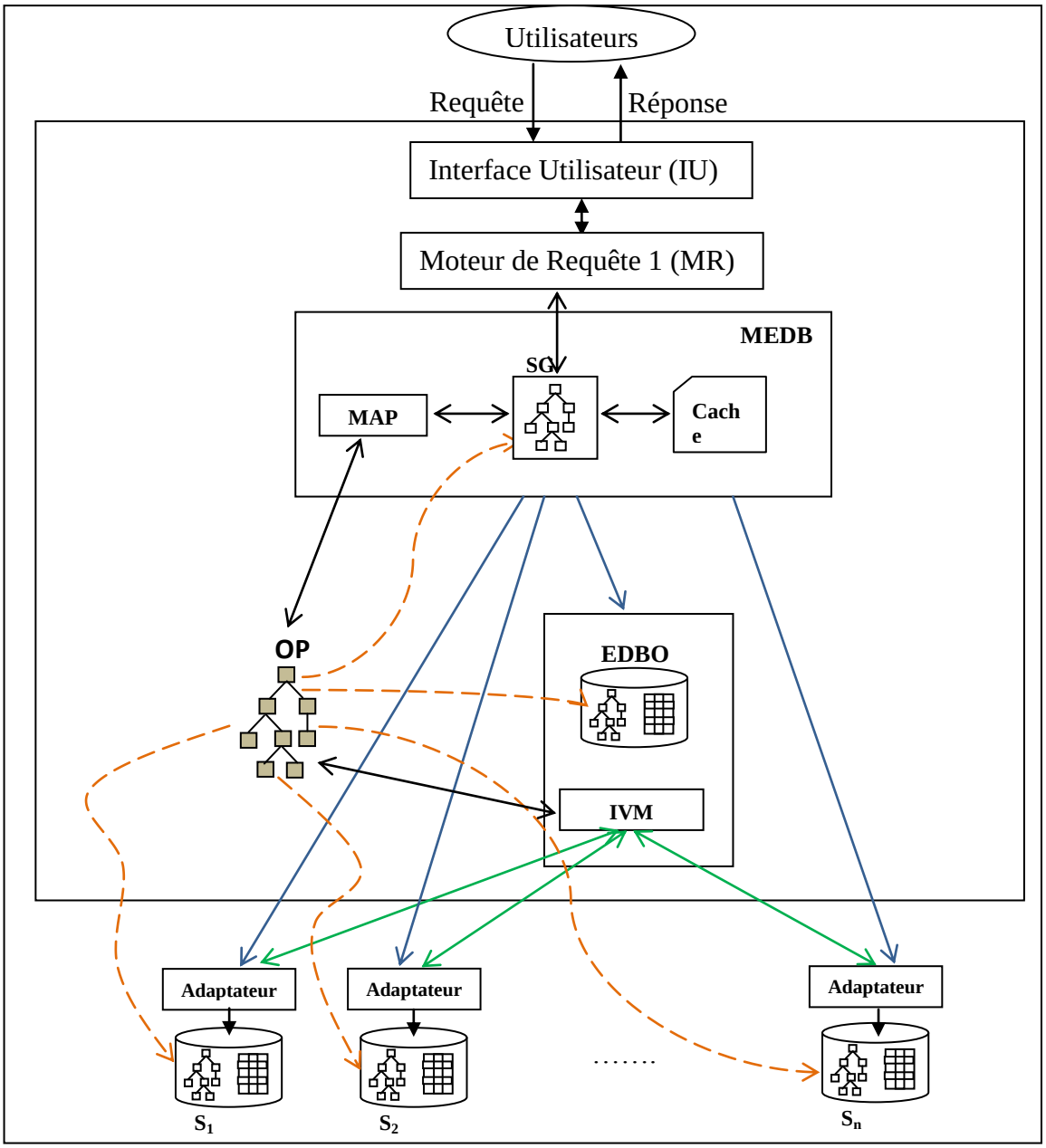

Figure 0.2 Architecture améliorée du système HYBOnto

Dans cette version de HYBOnto, nous avons proposé une nouvelle lecture des requêtes émies par un utilisateur, et pour cela, nous avons proposé ce scénario :

i- La première étape de l'intégration concerne la construction des ontologies de EDBO (O_{Ed}), et du médiateur qui n'est que le schéma global (SG) appelée aussi (O_{MED}),

ii- La deuxième étape concerne la mise à niveau du schéma global (SG), et cela en lui fusionnant aussi l'ontologie de l'entrepôt de données, de ce fait, l'entrepôt de données (EDBO) deviendra une source locale comme toutes les autres sources de données, et suivant le principe de notre algorithme qui n'est d'autres que chaque source de données ne sera interrogée que pour les données qui n'existent dans aucune autre sources de données, nous allons améliorer le temps de réponse des requêtes de façon très considérable.

4.10. Conclusion

Dans ce chapitre, nous avons proposé une approche d'intégration de sources de données structurées hétérogènes et autonomes. Cette approche, appelée intégration hybride par articulation a priori d'ontologies, suppose l'existence d'une (ou plusieurs) ontologie(s) de domaine, mais elle laisse chaque source autonome quand à la structure de sa propre ontologie. Le fait d'utiliser des ontologies (formelles) rend cette méthode complètement automatique.

Notre approche permet une intégration automatique des connaissances (ontologie) ainsi que des données (instances), nous avons expliqué chaque module et chaque couche de notre architecture en détail.

A la fin, nous avons proposé une amélioration de notre architecture plus légère souple pour le temps de réponse des requêtes utilisateur, mais aussi même pour le processus d'intégration et d'enrichissement des ontologies.

Dans le chapitre suivant nous commençons par présenter l'architecture logicielle de notre système d'intégration, puis, nous expliquerons tous les scénarii d'intégration des connaissances et des instances au sein de notre système. Nous présenterons pour chaque scénario un exemple concret.

CHAPITRE 5. SCENARII D'INTEGRATION

5.1. Introduction

De nombreux systèmes d'intégration ont été proposés mais leur principale difficulté est l'interprétation automatique de la signification sémantique des données hétérogènes et autonomes, ce qui donne lieu à différents conflits. Dans la première génération de systèmes d'intégration, la signification des données n'est pas représentée explicitement. Les correspondances entre le schéma global et les schémas locaux sont réalisées manuellement et encodées dans des définitions de vues. Avec l'avènement des ontologies, un progrès important a pu être réalisé dans l'automatisation du processus d'intégration de sources hétérogènes grâce à la représentation explicite de la signification des données.

Plusieurs types d'ontologies ont été utilisés dans les systèmes d'intégration d'ordre linguistique ou conceptuel. Les ontologies linguistiques permettent une automatisation partielle du processus d'intégration avec la supervision d'un expert humain quant aux ontologies conceptuelles, celles-ci permettent une automatisation effective pour autant que chaque source référence exactement la même ontologie appelée ontologie partagée, sans possibilité d'extension ou d'adaptation. La limite de ces systèmes réside dans le fait qu'une fois l'ontologie partagée définie, chaque source doit utiliser le vocabulaire commun. L'ontologie partagée est en fait un schéma global, et, en conséquence, chaque source locale ne peut pas avoir une autonomie schématique.

Ce chapitre est dédié à la présentation détaillée l'architecture d'intégration pour notre approche d'intégration qui est appelée système d'intégration hybride

des Bases de Données à Base d'Ontologie, l'automatisation de l'intégration n'élimine pas le besoin d'une réflexion humaine pour l'identification de conceptualisations différentes pour la même réalité, cette réflexion est néanmoins à priori, c'est-à-dire bien avant l'intégration des sources des sources de données, et non pendant ou après la phase de l'intégration.

Le chapitre est organisé comme suit. La section 2 comporte de façon globale l'architecture d'intégration de notre système. La section 3 vise à décrire les différents scénarii d'intégration proposés. Pour chaque scénario, nous présenterons tout d'abord son contexte, puis l'algorithme d'intégration qui lui correspond, et enfin son application dans le domaine hospitalier, quant à la section 4, elle présentera notre algorithme d'intégration proposé pour notre approche d'intégration. Et finalement une conclusion à la section 5.

5.2. Architecture d'intégration du système HYBOnto

Notre architecture d'intégration part, en entrée, d'un ensemble de BDBO référençant a priori une ontologie partagée, et vise à produire, en sortie, un entrepôt de données ayant la même structure d'une BDBO et aussi un schéma global pour le médiateur, cette architecture est illustrée dans la figure 15, et cela afin de fournir des réponses pertinentes aux questions des utilisateurs à travers un système hybride. Cette architecture définit plusieurs opérateurs d'intégration correspondant à différents scénarii possibles qui dotent l'ensemble les BDBOs d'une structure d'algèbre.

Nous allons d'abord présenter quelques notions élémentaires, ensuite on parlera de façon générale sur l'architecture d'intégration de notre système.

5.2.1. Notions élémentaires

5.2.1.1. Concept ontologique

Un concept ontologique o est ou bien un organisme fournisseur (la description d'une source), ou une classe, soit une propriété, soit un type de données.

5.2.1.1.1. Classe locale

Une classe locale c_i d'une source S_i référence directement la classe c_i de l'ontologie partagée.

5.2.1.1.2. Autonomie significative

L'autonomie significative signifie que chaque organisme fournisseur de données peut avoir des concepts propres dans sa base, et est en plus indépendant de ses utilisateurs.

5.2.1.1.3. Articulation Sémantique

Une articulation sémantique est une relation sémantique existant entre les concepts des ontologies locales et globale.

5.2.2. Présentation du système d'intégration HYBOnto

Le système d'intégration hybride que nous avons proposé permet une intégration automatique des sources de données, pour cela, plusieurs couches (abstraites et concrètes) interviennent qui sont :

5.2.2.1. Infrastructure

Les infrastructures de notre système sont :

- La localisation de données à intégrer : Hybride
- Nature de Mapping : GaV.
- Processus d'intégration : Automatique.
- Médiateur : MEDBO.
- Entrepôt de données : EDBO
- Architecture à base ontologique : Approche Hybride.
- Type ontologie : Conceptuelle (Formelle).

5.2.2.2. Base de connaissances

Notre base de connaissance comporte

- Sources de données à base ontologique : Architecture OntoDB.
- Ontologie de domaine (Domaine Hospitalier) : HOnto.
- Description des sources :Bases Relationnelles.

5.2.2.3. Langage de manipulation des données

On utilise deux langages d'interrogation

- Langage de requête : SQL
- Langage de l'ontologie : HOntoL

5.3. Principe d'engagement sur l'ontologie partagée (de référence)

Une ontologie est dite « partagée » entre plusieurs sources, lorsque les sources s'engagent sur le sens et sur le fait d'utiliser les définitions ontologiques de l'ontologie partagée qui ont été acceptées et éventuellement normalisées. Afin de garder l'autonomie d'une source, cette source peut définir sa propre hiérarchie de classes, et, si besoin est, rajouter les propriétés qui n'existent pas dans l'ontologie partagée.

Plus précisément, s'engager sur une ontologie partagée signifie respecter la double contrainte suivante (appelée SSCR : Smallest Subsuming Class Référence [110]):

- Toute classe locale doit référencer, par la relation OntoSub, la plus petite classe subsumante existante dans la hiérarchie de référence si ce n'est pas la même que celle de sa propre super classe ;
- Toute propriété nécessaire à l'ontologie locale et existante dans l'ontologie de référence doit être importée à travers la relation OntoSub.

Si une ontologie locale O_i est articulée avec l'ontologie partagée O_p en respectant le principe SSC R, nous disons qu'elle « référence autant que cela est possible » l'ontologie O_p. Nous formulons l'articulation entre une $BDBO_i$ et l'ontologie O_p comme un triplet $Ai,p :< BDBO_i, O_p, OntoSub_i, p >$, où :

- $BDBOi :< O_i, I_i, Pop_i, Sch_i >$ représente une base de données à base ontologique, dont $O_i :< C_i, P_i, Sub_i, Applic_i >$ est l'ontologie locale
- $O_p :< C_p, P_p, Sub_p, Applic_p >$ représente l'ontologie partagée portant sur le même univers du discours que $BDBO_i$,

- $OntoSub_i, p : C_p \rightarrow 2C_i$ représente les relations de subsomption entre O_p et O_i qui associent à chaque classe c_p de C_p l'ensemble des classes $c_i \in C_i$ qui sont subsumés directement par c_p.

5.4. Les scénarii d'intégration

Dans cette section, nous allons présenter la description formelle du contexte afin de faciliter la compréhension par la suite des scénarii d'intégration considérés que nous présenterons juste après.

5.4.1. Description formelle du contexte

Soit $S = \{S_1, S_2, \ldots, S_n\}$ l'ensemble des sources de données participant au processus d'intégration. Chaque source S_i est définie comme suit: $S_i :< O_i, I_i, Sch_i, Pop_i >$. Notons que dans une BDBO, tout élément représentédans le schéma, classe ou propriété doit appartenir à l'ontologie, de sorte que le schéma est un sous-ensemble de l'ontologie, chaque entité représentée correspondant à une classe et ses attributs correspondant aux propriétés applicables choisies. On fait abstraction ici du découpage éventuel résultant des opérations de normalisation, une vue étant, dans tous les cas, créée pour représenter la population de chaque classe.

Plus simplement, nous allons supposer que seules les classes feuilles sont choisies comme classes de base et sont directement instanciables. Les classes non feuilles sont supposées « abstraites », c'est-à-dire que leur population est l'union des populations de leur sous-classes.

Dans la méthode d'intégration par articulation a priori d'ontologie, nous supposons que l'ontologie partagée O_p pré-existe à la définition de la source $BDBO_i$. Notons que cette hypothèse est toujours faite lorsque l'on annote à l'aide de métadonnées. On suppose aussi que l'administrateur de la source (DBA) a défini sa propre ontologie, et que cette ontologie référence autant que cela est possible, l'ontologie partagée (Hypothèse de Base).

La construction de sources de données passe par six étapes, le DBA de chaque source effectue les opérations suivantes pour construire ses vues sur les sources $BDBO_i$:

1. le DBA choisit la hiérarchie de classes (C_i, Sub_i) de sa propre ontologie O_i.

2. le DBA articule cette hiérarchie de classes avec celle de l'ontologie partagée C_p en définissant les relations de subsomption $< OntoSub_i, p >$ entre C_i et C_p.

3. à travers les relations de subsomption $< OntoSub_i, p >$, le DBA importe dans $Applic_i\ (c_i)$ les propriétés de $Applic_p\ (OntoSub^{-1} i, p\ (c_i)) \subset P_p$ qu'il souhaite utiliser dans sa propre ontologie. Ces propriétés appartiennent alors à P_i.

4. le DBA complète éventuellement les propriétés importées par des propriétés supplémentaires, propres à son ontologie définissant ainsi l'ontologie locale: $O_i : < C_i, P_i, Sub_i, Applic_i >$.

5. le DBA de chaque source choisit pour chaque classe feuille les propriétés qui seront valuées en définissant $Sch_i : C_i \rightarrow 2P_i$, et

6. le DBA choisit une implémentation de chaque classe feuille (e.g., afin d'assurer la troisième forme normale), et il défini ensuite $Sch(c_i)$ comme une vue sur l'implémentation de c_i.

Dans ce cas, le schéma de chaque classe feuille est explicitement défini. Et celui d'une classe non feuille est calculé comme étant l'intersection entre les propriétés applicables de c_j et l'intersection des ensembles de propriétés associées à des valeurs dans toutes les sous classes $c_{i,j}$ de c_j :

$$Sch(c_j) = Applic(c_j) \cap (\cap i\ Sch(c_{i,j})) \qquad 0.1$$

Ou encore d'une autre façon, nous pouvons prendre l'union des propriétés existantes dans au moins une sous-classe, et en complétant par des valeurs nulle:

$$Sch'(c_j) = Applic(c_j) \cap (\cup iSch(c_{i,j})) \quad 0.2$$

Dans notre cas nous choisissons deuxième représentation, ce choix sera justifié par la suite.

Nous pouvons distinguer plusieurs scénarii d'intégration, correspondants à différentes articulations entre les ontologies locales et l'ontologie partagée du domaine. Dans ce mémoire, deux catégories de scénarios sont étudiées. La première catégorie de scénarii est basée sur la technique de fragmentation, habituellement, utilisée dans les bases de données classiques. Quant à la deuxième catégorie de scénarii, c'est sur l'opérateur algébrique de composition, dit projection. Dans la section suivante l'ensemble des scénarii sont exposés. Pour chaque scénario, nous décrivons son contexte appliqué et son algorithme d'intégration.

5.4.2. Formalisation des scénarii d'intégration proposés

Dans cette partie nous allons présenter l'ensemble des scénarii d'intégration* adaptés et appliqués à notre domaine de travail, et puis nous expliqueront notre scénario proposé.

5.4.2.1. Le scénario basé sur la fragmentation HybFragOnto

Le premier scénario d'intégration que nous présentons est nommé HybFragOnto. Ce scénario d'intégration suppose que l'ontologie partagée est suffisante pour couvrir toutes les sources locales. Une hypothèse de ce type a déjà été utilisée (par exemple le projet Picsel2, et COIN). Dans ce cas, l'autonomie des sources se limite à (i) sélectionner un sous ensemble pertinent de l'ontologie partagée (classes et propriétés) et (ii) concevoir le schéma local de la base de données.

* Les scénarii d'intégration sont issus des travaux menés dans la thèse de DUNG XUAN NGUYEN pour plus de détail, voir [30]

L'ontologie O_i de chaque source S_i étant un fragment horizontal de l'ontologie partagée O_p. Elle se définit comme le quadruplet $O_i :< C_i, P_i, Sub_i, Applic_i >$, avec :

- $C_i \subseteq C_p$;
- $P_i \subseteq P_p$;
- $\forall c \in C_i, Sub_i(c) \subseteq Sub_p(c)$;
- $\forall c \in C_i, Applici(c) \subseteq Applicp\,(c)$.

Pour intégrer ces sources au sein d'une BDBO il suffit de trouver l'ontologie, le schéma et la population du système intégré. Le système intégré est donc défini comme Int :

$$< O_{Int}, Sch_{Int}, Pop_{Int} > \quad 0.3$$

Ainsi, nous aurons deux ontologies d'intégration qui sont $\mathrm{O}_{\mathrm{Int}_{\mathrm{ED}}}$ et $\mathrm{O}_{\mathrm{Int}_{\mathrm{MED}}}$ pour les deux systèmes intégrés respectueusement.

Du moment que le médiateur n'intègre pas la population des sources de données (les instances des bases de données), les deux systèmes d'intégration Int_{ED} et Int_{MED} seront, respectueusement, comme suit :

$$<O_{int_{ED}}, Sch_{int_{ED}}, Pop_{int_{ED}} > \quad 0.4$$

et

$$<O_{int_{MED}}, Sch_{int_{MED}} > \quad 0.5$$

Initialement on a :

$$O_{Int_{ED}} = O_{Int_{MED}} = O_P \quad 0.6$$

$$Sch_{int_{ED}} = Sch_{int_{MED}} = \emptyset \quad 0.7$$

Pour chaque concept $C_p \in O_p$, on doit vérifier si il appartient à au moins une source S_i , si cela est vérifié deux cas se présentent :

1er cas :

$$C_p \in O_p \text{ et } \exists S_i \neq S_j / C_p \in (S_i \cap S_j) \quad 0.8$$

L'intégration se fait au niveau de la couche Entrepôt de données INT_{ED} et cela comme suit :

Dans le cas de l'intersection :

$$Sch_{int_{ED}} = Sch_{int_{ED}} \cap (\bigcap_{i=1..n} Sch_i(c)) \quad 0.9$$

$Pop_{Int_{ED}}$ représente la population de chaque concept du système intégré Int_{ED}, elle est définie comme suit :

$$Pop_{Int_{ED}}(C) = i * Pop_i(C). \quad 5.10$$

Dans le cas d'union

$$Sch_{int_{ED}} = Sch_{int_{ED}} \cup (\bigcup_{i=1..n} Sch_i(c)) \quad 0.11$$

$Pop_{Int_{ED}}$ représente la population de chaque concept du système intégré Int_{ED}, elle est définie comme suit :

$$Pop_{Int_{ED}}(C) = i * Pop_i(C) \quad 0.12$$

2ème cas :

$$C_p \in O_p, \forall S_i \neq S_j \ / \ C_p \in (S_i \cup S_j) \wedge C_p \notin (S_i \cap S_j) \quad 0.13$$

Dans ce cas, l'intégration concerne le médiateur INT_{MED} on aura :

$$Sch_{int_{MED}} = Sch_{int_{MED}} \cup Sch_i(C_p) \quad 0.14$$

Afin de mieux comprendre l'algorithme nous présentons un exemple d'application dans la figure suivant :

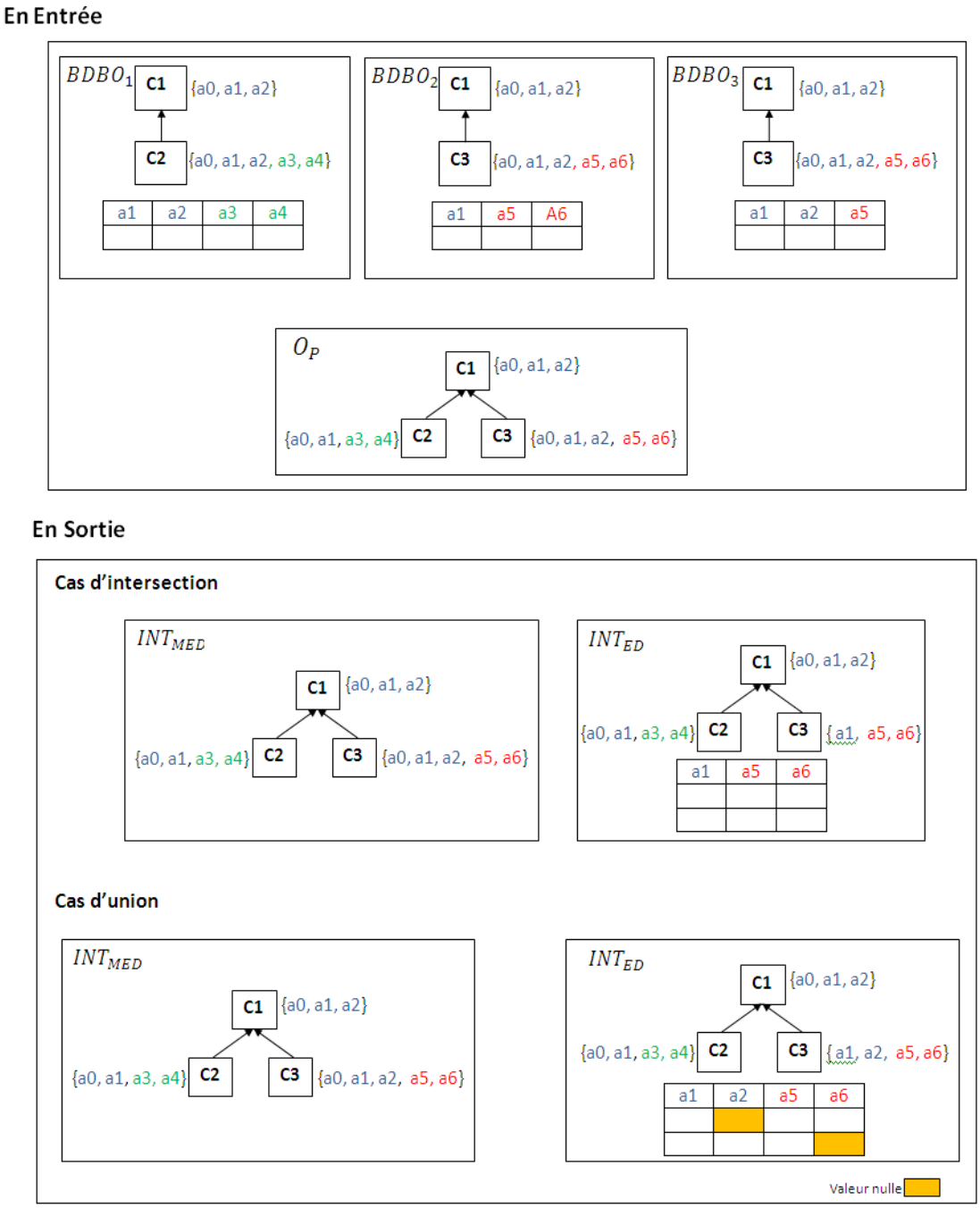

Figure 0.1 Exemple d'une intégration de BDBOs par *FragOnto*

5.4.2.2. Le scénario basé sur la projection HybProjOnto

Le deuxième scénario que nous étudions est nommé HybProjOnto. De nombreuses applications conçues autour de l'approche d'intégration à priori exigent plus d'autonomie :

- La classification de chaque source doit pouvoir être complètement différente de celle de l'ontologie partagée, et

- Certaines spécialisations de classe et certaines propriétés n'existant pas dans l'ontologie partagée doivent pouvoir être ajoutées dans les ontologies locales.

  Ce cas est très différent du précédent du fait que chaque source S_i a sa propre ontologie O_i et ses classes spécifique. Néanmoins, l'ontologie O_i référence autant que possible l'ontologie partagée O_p à travers l'articulation $A_{i,p} :< BDBO_i, O_p, Onto_{Sub_{i,p}} >$. Dans ce scénario, les sources ont des concepts propres qui n'existent pas dans l'ontologie partagée, mais ces concepts ne sont pas supposés intéresser les utilisateurs du système intégré. Ainsi, le système intégré vise à intégrer les instances de données de chaque source comme des instances de l'ontologie partagée.

Les instances de données de chaque S_i seront donc projetées sur l'ontologie partagée. Notons que, dans ce scénario, les sources ne peuplent que les classes qui lui sont propres, pas celles de l'ontologie partagée. L'algorithme qui suit déterminera la structure de chaque élément des deux systèmes intégrés $<O_{int_{ED}}, Sch_{int_{ED}}, Pop_{int_{ED}}>$ et $<O_{int_{MED}}, Sch_{int_{MED}}>$

Comme dans le scénario précédent (HybFragOnto), l'ontologie du système intégré est exactement l'ontologie partagée :

$$O_{Int_{ED}} = O_{Int_{MED}} = O_P \qquad 0.15$$

Cette définition montre qu'aucun concept défini localement n'est intégré dans le système intégré.

Pour ce scenario, chaque instance de données d'une source sera projetée sur les propriétés applicables de sa plus petite classe subsumante dans l'ontologie partagée. Cette plus petite classe subsumante devient donc la classe de base de l'instance intégrée.

Contrairement au cas précédent, la classe de base d'une instance dans l'un des deux systèmes intégrés n'est pas celle d'origine de cette instance dans sa source locale.

Soit $Pop*(c)$ la population des instances projetées sur la classe C de l'ontologie partagée, Pop∗(c) est calculée par l'union des populations de toutes les classes locales référençant directement $c.Pop*(c)$ est donnée par l'équation suivante :

$$Pop*(c) = \cup_{i\in[1..n]}\left(\cup_{c_j \in OntoSub_i(c)} Pop_i(c_i)\right) \quad 0.16$$

Le schéma des instances projetées de la classe c est déterminé comme :

$$Sch*(c) = Applic_p(c) \cap \left(\cap_{i\in[1..n]}\left(\cap_{\left(c_j \in OntoSub_i(c)\right)\wedge(Pop_i(c_j)\neq\emptyset)} Sch_i(c_j)\right)\right) \quad 0.17$$

OU

$$Sch*(c) = Applic_p(c) \cap \left(\cup_{i\in[1..n]}\left(\cup_{\left(c_j \in OntoSub_i(c)\right)\wedge(Pop_i(c_j)\neq\emptyset)} Sch_i(c_j)\right)\right) \quad 0.18$$

La population et le schéma de chaque classe feuille de l'ontologie partagée sont calculés par les équations précédentes, en revanche, pour une classe non feuille, les trois équations doivent être comme suit :

$$Sch_{Int_{ED}}(c) Applic_p(c) \cap \left(\left(\cap_{c_k \in OntoSub_{Int}(c)} Sch_{Int_{ED}}(c_k)\right) \cap_{Pop*(c)\neq\emptyset} Sch*(c)\right)$$

OU

$$Sch'_{Int_{ED}}(c) = Applic_p(c) \cap \left(\left(\cup_{c_k \in Sub_{Int}(c)} Sch'_{Int_{ED}}(c_k)\right) \cup \quad Sch*(c)\right) \quad 0.19$$

Et dans les deux cas :

$$Pop_{Int_{ED}}(c) = \left(\cup_{c_k \in Sub_{Int}(c)} Pop_{Int_{ED}}(c_k)\right) \cup Pop*(c) \quad 0.20$$

En ce qui suit un exemple schématisé de l'intégration par projection :

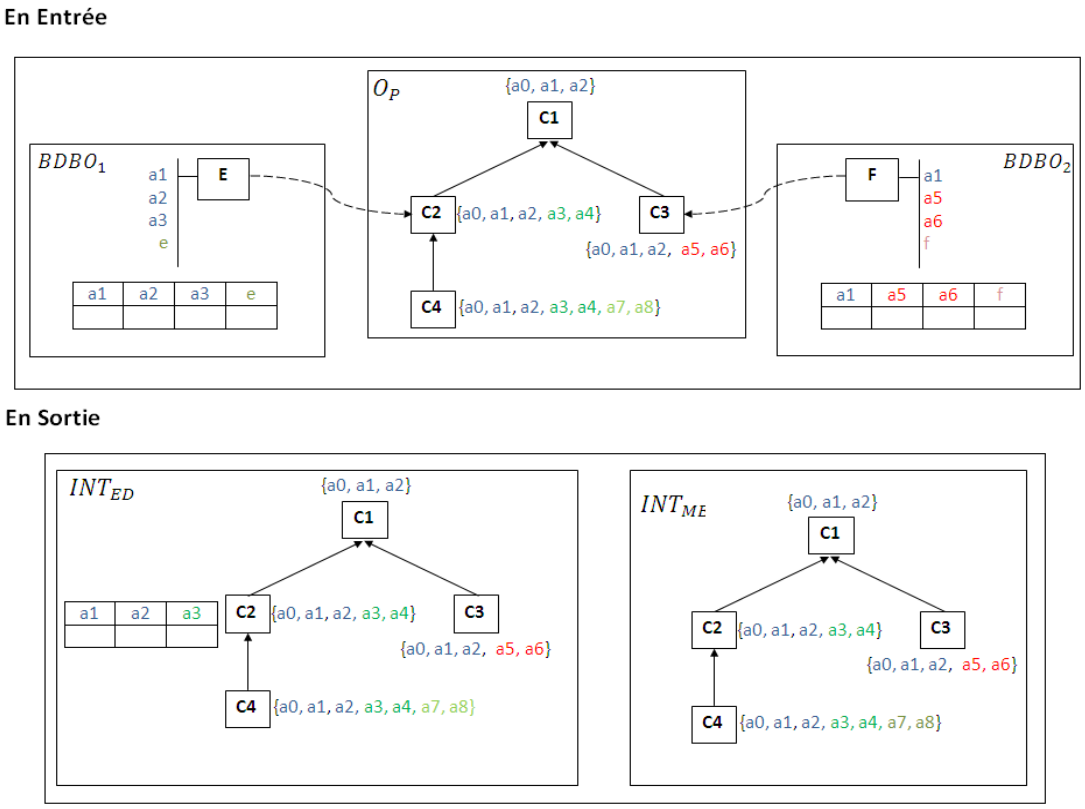

Figure 0.2 Exemple d'une intégration de BDBO par ProjOnto

5.4.2.3. Scénario d'extension de projection HybProjExtendOnto

Ce scénario permet un enrichissement du système d'intégration, ce qui permettra, contrairement au scénario de projection, une autonomie plus large aux sources de données.

L'entrée du système intégré du scénario *HybProjExtendOnto* est identique à celle de *ProjOnto* où chaque source a sa propre ontologie et ses classes spécifiques. En revanche, ce cas est différent du précédent du fait que le système intégré permet d'intégrer même les extensions de chaque S_i. L'ontologie partagée sera donc étendue en intégrant les ontologies locales.

Pour un tel contexte, l'intégration de BDBOs consiste à intégrer d'abord les ontologies, puis les données.

Dans ce cas également, une automatisation du processus d'intégration est possible.

Pour ce faire, nous devons trouver la structure finale du système intégré Int_{ED} et Int_{MED} : $<O_{int_{ED}}, Sch_{int_{ED}}, Pop_{int_{ED}}>$ et $<O_{int_{MED}}, Sch_{int_{MED}}>$.

Pour chaque concept $C_p \in O_p$ ne figurant dans aucune source mais qui est référencé, par articulation [17], par au moins une classe $C_i \in S_j$ telque $C_i \notin O_p$,

1er cas : il existe au moins deux classes C_i et C_j appartenant à deux sources de données différentes mais qui référencent toutes les deux la meme classe C_p de l'ontologie partagée O_p par articulation:

$$C_p \in O_p \text{ et } \exists\, C_i \in S_r et\ C_j \in \frac{S_s}{A_{Ci,Cp}} < S_r, O_p, OntoSub_{i,p} > \wedge\ A_{Cj,Cp} < S_s, O_p, OntoSub_{j,p} > \quad 5.21$$

Alors, on doit d'abord enrichir l'ontologie de Int_{ED} ($O_{Int_{ED}}$) comme suit

- $\mathrm{C}_{\mathrm{Int}_{\mathrm{ED}}} = \mathrm{C}_{\mathrm{Int'}_{\mathrm{ED}}} \cup \mathrm{C_i} \cup \mathrm{C_j}$
- $\mathrm{P}_{\mathrm{Int}_{\mathrm{ED}}} = \mathrm{P}_{\mathrm{Int'}_{\mathrm{ED}}} \cup \mathrm{P_i} \cup \mathrm{P_j}$
- $\mathrm{Applic}_{\mathrm{Int}_{\mathrm{ED}}}(\mathrm{C}) = \mathrm{Applic_i}(\mathrm{C})$
- $\mathrm{Sub}_{\mathrm{Int}_{\mathrm{ED}}}(\mathrm{C}) = \mathrm{Sub_i}(\mathrm{C})$

Pour la population de Int_{ED} de chaque classe C n'appartenant pas à O_p, elle est donnée par l'équation suivante :

$$Pop_{Int_{ED}}(C) = Pop_i(C) \quad 0.22$$

Finalement, le schéma de chaque classe du système intégré INT_{ED} est calculé en utilisant le même principe que la population en considérant les classes appartenant à C_i et les classes appartenant à C_p. Les schémas des classes appartenant à l'une des C_i sont explicitement définies :

$$(Sch_{Int_{Ed}}(c) = Sch_i(c)) \quad 0.23$$

Concernant les classes de C_p, le schéma de la classe C peut être calculé en appliquant la formule précédente sur l'ontologie du système intégré $O_{Int_{ED}}$.

Cela montre qu'il est possible d'offrir aux sources locales une large autonomie et tout en permettant également une construction automatique du système intégré d'une manière déterministe et exacte.

2ème cas : il n'existe qu'une seule et unique classe C_i parmis toutes les classes de toutes les sources de données S_i référançant la classe C_p de l'ontologie partagée O_p par articulation :

$C_p \in O_p$ et $\exists!\ C_i / A_{Ci,Cp} < S_r, O_p, OntoSub_{i,p} >$ 0.24

Alors, on doit commencer par enrichir l'ontologie de Int_{MED} ($O_{Int_{MED}}$) comme pour le cas précédent :

- $\mathrm{C}_{\mathrm{Int_{MED}}} = \mathrm{C}_{\mathrm{Int'_{MED}}} \cup \mathrm{C_i}$
- $\mathrm{P}_{\mathrm{Int_{MED}}} = \mathrm{P}_{\mathrm{Int'_{MED}}} \cup \mathrm{P_i}$
- $\mathrm{Applic}_{\mathrm{Int_{MED}}}(\mathrm{C}) = \mathrm{Applic_i}(\mathrm{C})$
- $\mathrm{Sub}_{\mathrm{Int_{MED}}}(\mathrm{C}) = \mathrm{Sub_i}(\mathrm{C})$

Le schéma de chaque classe du système intégré Int_{MED} est donné comme suit :

$Sch_{Int_{MED}}(C) = Sch_i(C)$ 5.25

Voici un exemple explicatif de ce scénario :

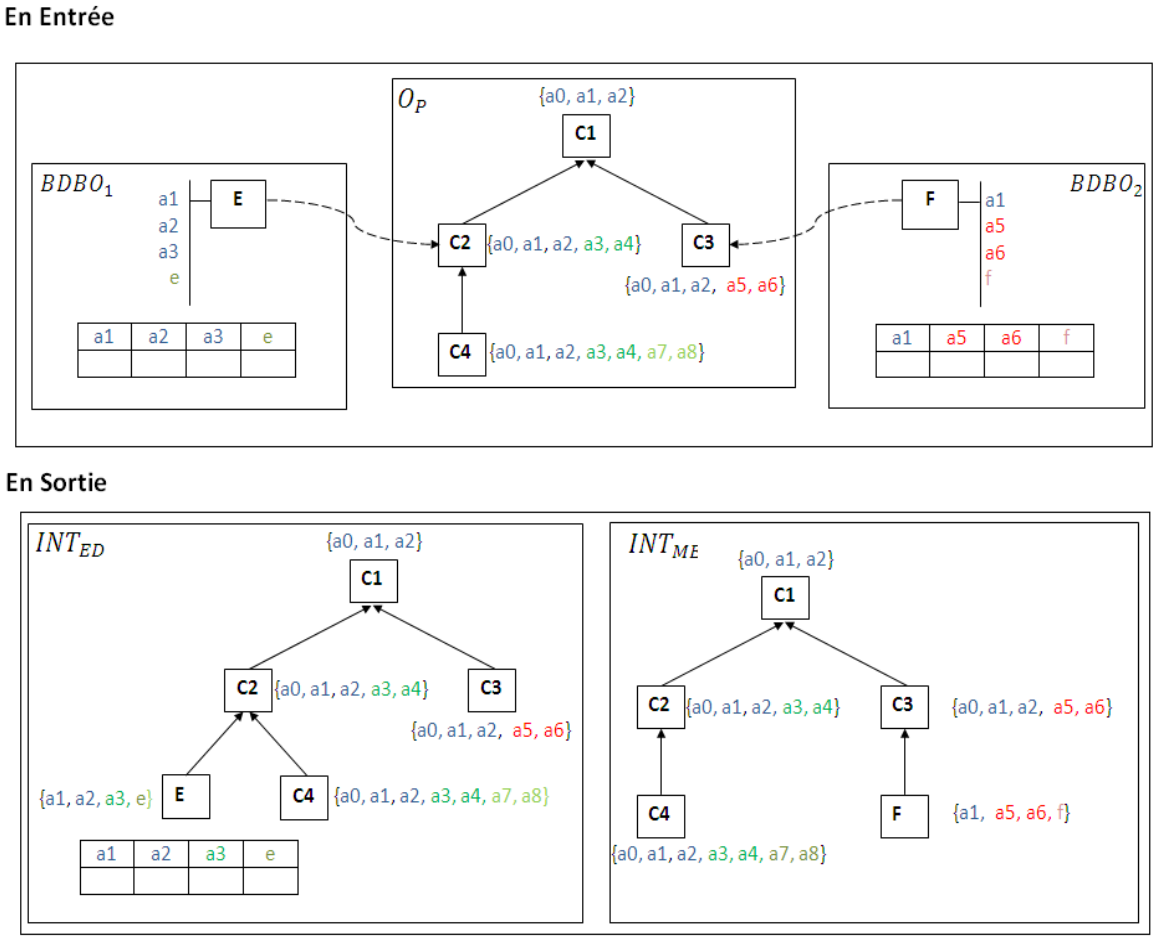

Figure 0.3 Exemple d'une intégration BDBO par HybProjExtendOnto

5.4.2.4. Scénario d'intégration global « HybExtendOnto »

Dans notre approche, nous avons fusionné les scénarios proposés précédemment de sorte que l'Intégration Hybride par Intersection sera adoptée pour la partie matérialisée et l'Intégration Hybride par Union moins les Intersections pour la partie virtuelle du système intégré.

En ce qui suit, l'algorithme récapitulatif des trois algorithmes cités précédemment :

1. Pour chaque concept $C_p \in O_p$, on doit vérifier si il appartient à au moins une source S_i , si cela est vérifié deux cas se présentent :

1er cas :

$C_p \in O_p$ et $\exists S_i \neq S_j / C_p \in (S_i \cap S_j)$ 0.26

alors :

$Sch_{int_{ED}} = Sch_{int_{ED}} \cup \left(\cup_{i=1..n} Sch_i(C_p)\right)$ 0.27

$Pop_{Int_{ED}}$ représente la population de chaque concept du système intégré Int_{ED}, elle est définie comme suit :

$Pop_{Int_{ED}}(C) = i * Pop_i(C).$ 0.28

2ème cas :

$C_p \in O_p, \forall S_i \neq S_j / C_p \in (S_i \cup S_j) \wedge C_p \notin (S_i \cap S_j)$ 0.29

alors :

$Sch_{int_{MED}} = Sch_{int_{MED}} \cup Sch_i(C_p)$ 0.30

2. Pour chaque concept $C_p \in O_p$ ne figurant dans aucune source mais qui est référencé, par articulation [17], par au moins une classe $C_i \in S_j$ telque $C_i \notin O_p$

1er cas :

Il existe au moins deux classes C_i et C_j appartenant à deux sources de données différentes mais qui référencent toutes les deux la meme classe C_p de l'ontologie partagée O_p par articulation:

$C_p \in O_p$ et $\exists C_i \in S_r et C_j \in S_s / A_{Ci,Cp} < S_r, O_p, OntoSub_{i,p} > \wedge A_{Cj,Cp} < S_s, O_p, OntoSub_{j,p} >$ 0.31

Alors, on doit d'abord enrichir l'ontologie de Int_{ED} ($O_{Int_{ED}}$) comme suit

- $C_{Int_{ED}} = C_{Int'_{ED}} \cup C_i \cup C_j$
- $P_{Int_{ED}} = P_{Int'_{ED}} \cup P_i \cup P_j$

- $\mathrm{Applic}_{\mathrm{Int}_{\mathrm{ED}}}(\mathrm{C}) = \mathrm{Applic}_{\mathrm{i}}(\mathrm{C})$

- $\mathrm{Sub}_{\mathrm{Int}_{\mathrm{ED}}}(\mathrm{C}) = \mathrm{Sub}_{\mathrm{i}}(\mathrm{C})$

Pour la population de Int_{ED} de chaque classe C n'appartenant pas à O_p, elle est donnée par l'équation suivante :

$$Pop_{Int_{ED}}(C) = Pop_i(C) \quad 0.32$$

Finalement le schéma de chaque classe du système intégré Int_{ED} est calculé comme suit :

$$Sch_{Int_{ED}}(C) = Sch_i(C) \quad 0.33$$

2ème cas :

Il n'existe qu'une seule et unique classe C_i parmis toutes les classes de toutes les sources de données S_i référançant la classe C_p de l'ontologie partagée O_p par articulation :

$$C_p \in O_p \text{ et } \exists!\ C_i/A_{Ci,Cp} < S_r, O_p, OntoSub_{i,p} > \quad 0.34$$

Alors, on doit commencer par enrichir l'ontologie de Int_{MED} ($O_{Int_{MED}}$) comme pour le cas précédent :

- $\mathrm{C}_{\mathrm{Int}_{\mathrm{MED}}} = \mathrm{C}_{\mathrm{Int'}_{\mathrm{MED}}} \cup \mathrm{C}_{\mathrm{i}}$

- $\mathrm{P}_{\mathrm{Int}_{\mathrm{MED}}} = \mathrm{P}_{\mathrm{Int'}_{\mathrm{MED}}} \cup \mathrm{P}_{\mathrm{i}}$

- $\mathrm{Applic}_{\mathrm{Int}_{\mathrm{MED}}}(\mathrm{C}) = \mathrm{Applic}_{\mathrm{i}}(\mathrm{C})$

- $\mathrm{Sub}_{\mathrm{Int}_{\mathrm{MED}}}(\mathrm{C}) = \mathrm{Sub}_{\mathrm{i}}(\mathrm{C})$

Le schéma de chaque classe du système intégré Int_{MED} est donné comme suit :

$$Sch_{Int_{MED}}(C) = Sch_i(C) 05.35$$

La figure suivante est un exemple d'intégration selon ce scénario, nous présentons ici juste la partie ontologie du fait que l'intégration de la partie donnée ne change est la même que dans les scénarii précédents.

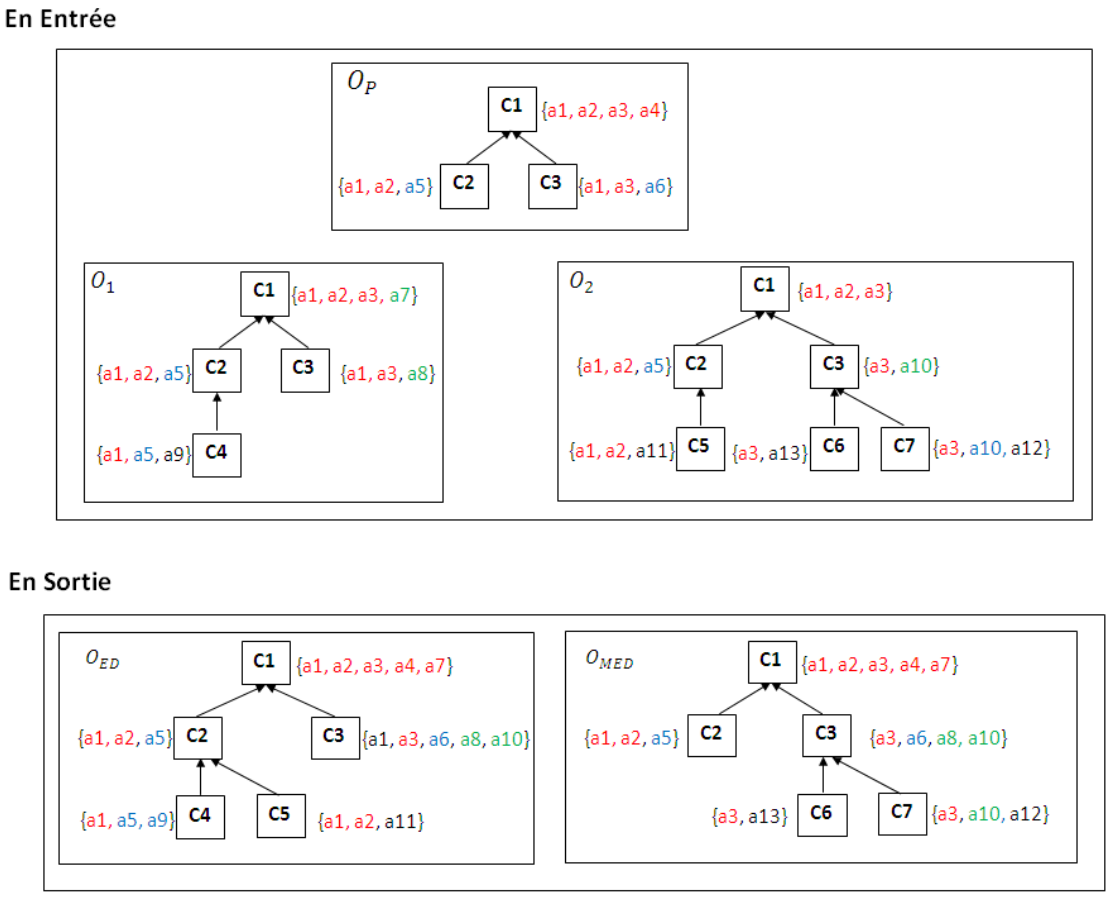

Figure 0.4 Exemple d'intégration de la partie Ontologie de des BDBO par HybExtendOnto

5.5. Conclusion

Dans ce chapitre, nous avons présenté l'architecture logicielle de notre système d'intégration, et les différents scénarii d'intégration. Ces scénarii nous permettent une intégration automatique à priori d'ontologie, permettant d'une part l'autonomie des sources, et d'une autre part l'enrichissement de l'ontologie globale en exécutant les algorithmes d'intégration avec extension.

Reste que le grand inconvénient de cette approche est bien la mise à jour et la maintenance de l'entrepôt de données notamment le cas ou les données locales sont mises à jour fréquemment et représentent des vues matérialisées pour notre système.

Dans le chapitre suivant, nous allons valider notre approche d'intégration après implémentation de notre système d'intégration.

CHAPITRE 6. REALISATION ET VALIDATION

6.1. Introduction

Nous avons présenté dans le chapitre précédent les scénarios d'intégration pour l'approche d'intégration des BDBOs par articulation a priori d'ontologies dans un système Hybride. Dans ce chapitre, nous allons présenter une implémentation qui consiste à réaliser effectivement l'intégration au sein d'une base de données à base ontologique sous ORACLE, puis à interfacer le résultat avec le langage php. Cette implémentation permet de valider les quatre scénarios d'intégration proposés ci-dessus.

Nous décrirons tout d'abord l'environnement dans lequel nous avons effectué notre implémentation. Le scénario et l'implémentation de cette mise en œuvre seront ensuite détaillés ainsi que l'interface utilisateur et le moteur de requête que nous avons déployés et finalement nous analyserons les résultats obtenus.

6.2. Environnement de mise en œuvre

Notre implémentation est réalisée dans l'environnement de base de données ORACLE où les ontologies et les sources de données sont représentées sous la forme de bases de données physique conformément au modèle OntoDB.

Afin de manipuler les bases de données sous oracle, nous avons utilisé un outil de développement appelé « SQL Manager for Oracle », l'interface ce cet outil facilite la conception des différentes sources de données, ainsi que le système d'intégration Hybride.

Chaque étape d'implémentation ainsi que les scénarii d'intégration ont été programmé en PLSQL.

Toute l'implémentation est réalisée localement, afin de pourvoir simuler la décentralisation nous avons utilisé le langage web PHP pour la mise en place des interfaces utilisateurs et administrateurs, ainsi que notre moteur de requête.

6.3. Les étapes d'implémentation

Afin de valider notre algorithme d'intégration au sein d'un système hybride, nous avons tout d'abord mis en œuvre des sources de données locales se référant au modèle déjà cité OntoDB ainsi que l'ontologie partagée sous Oracle, nous avons par la suite défini la structure initiale de l'entrepôt de données et le médiateur.

La création des sources de données est semi-automatique, l'administrateur de la source doit introduire l'ontologie qui sera automatiquement mise dans la partie Ontologie sera effectuée la liste des classes feuilles ainsi que ses attributs sera affichée et l'administrateur choisie la structure de sa base de données tout en ayant le choix de changer les noms des tables ainsi que les attributs et les relations, après validation la base de données sera automatiquement créée.

Par la suite l'administrateur du système d'intégration doit introduire l'ontologie partagée au biais d'une interface très simple et elle sera

automatiquement insérée au niveau du système, et puis il doit remplir les informations nécessaires sur les sources locales, après validation l'algorithme s'en charge de faire automatiquement de faire la correspondance entre l'ontologie partagées et les ontologies locales de mettre à jour les différentes parties du système en suivant ces étapes :

1. Création automatique de la structure du système d'intégration.
2. Réécriture de l'ontologie partagée dans l'entrepôt de données (O_{ED}) et le médiateur (O_{MED}).
3. Enrichissement des deux ontologies O_{ED} et O_{MED} en appliquant l'algorithme d'intégration selon les critères déjà défini au paravent.
4. Récupération des données depuis les sources locales vers l'entrepôt de données et définition du schéma global du médiateur
5. Mettre à jour la description des sources (DS) au niveau du médiateur ainsi que le cache.

Puis nous avons mis en place un moteur de requête qui:

1. Comporte un analyseur de requête SQL qui permet de vérifier si la requête est bien écrite, et puis si elle sera traitée par l'entrepôt de données, ou le médiateur ou encore les deux à la fois,
2. réécrit la requête de l'utilisateur et la décompose en sous requêtes selon les sources de données concernées,
3. récupère les différents résultats envoyés par les sources dans le cache
4. ré-exécute la requête sur le cache avant d'envoyer le résultat à l'utilisateur.

6.4. La construction globale de notre architecture

6.4.1. Construction des sources de données locales à base d'ontologie

Suivant les étapes citées dans (chap1§….), nous avons construit nos sources de données locales suivant le model OntoDB, dans cette section, nous allons faire un petit rappel sur la structure de ces sources de données, puis nous détaillerons toutes les étapes suivies dans la construction d'une source de données à base d'ontologie

6.4.1.1. Architecture conceptuelle d'une Base de Données à Base d'Ontologie

La figure 20 représente l'architecture logique d'une BDBO. Ce modèle d'architecture, appelé *OntoDB*, permet de représenter simultanément des ontologies de domaines, décrites en termes de classes et de propriétés, et des objets du domaine, définis en termes de ces ontologies et respectant les deux hypothèses R1 et R2.

Le modèle comporte quatre parties :

1. La partie *ontologie* permet de représenter complètement les ontologies de domaines définissant la sémantique des différents domaines couverts par la BDD, ainsi, éventuellement, que l'articulation (représentée sous forme de "*mapping*" [123]) de ces ontologies avec des ontologie externes.

2. La partie *méta-schéma* permet de représenter, au sein d'un modèle réflexif, à la fois le modèle d'ontologie utilisé et le méta-schéma lui-même.

3. La partie *méta-base* permet de représenter le schéma de représentation des objets du domaine couvert par les ontologies.

4. La partie *données* représente les objets du domaine; ceux ci sont décrits en termes d'une classe d'appartenance et d'un ensemble de valeurs de propriétés applicables à cette classe.

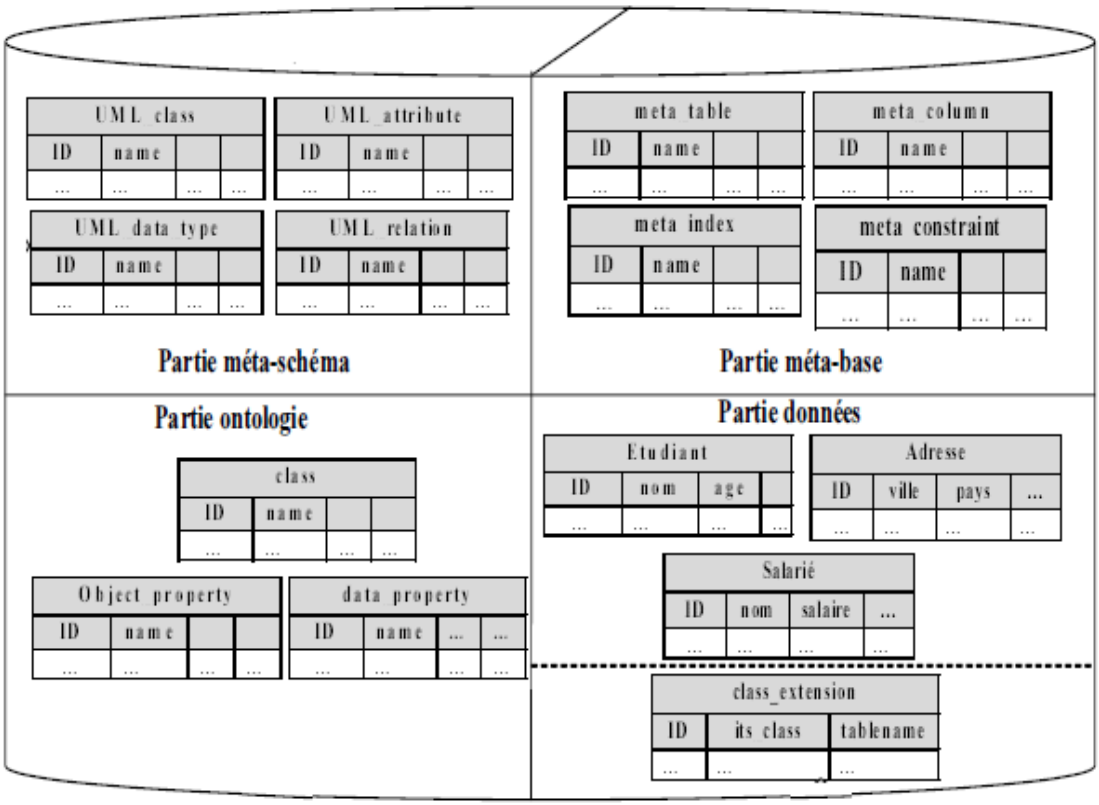

Figure 0.1 Architecture d'une base de données à base ontologique selon le modèle OntoDB [63]

6.4.1.2. Création de la partie méta-schéma

Le méta-schéma représente la structure où l'ontologie sera stockée, il comporte quatre classes : UML_CLASS, UML_ATTRIBUTE, UML_DATA_TYPE et UML_RELATION, la création de cette partie est très simple, il suffit d'exécuter les requêtes SQL de création, (voir annexe 2).

En ce qui suit la description des champs des quatre classes du méta-schéma :

UML_CLASS

Attribut	Description	Type
IDO	Identifiant de la classe	Entier (séquentiel)
Nom	Nom de la classe	Chaine de caractère
Superclasse	L'identifiant de la superclasse de cette classe (la classe mère)	Entier

UML_ATTRIBUT

Attribut	Description	Type
IDO	Identifiant de l'attribut	Entier (séquentiel)
Nom	Nom de l'attribut	Chaine de caractère
EST_DE_TYE	Le type de l'attribut	Entier

UML_RELATION

IDO	Identifiant de la relation	Entier (séquentiel)
Nom	Nom de la relation	Chaine de caractère
Debut	IDO de la classe source de la relation	Entier
Fin	IDO de la classe terminale de la relation	Entier
S_MIN	La cardinalité minimale pour la classe source	Entier
S_MAX	La cardinalité maximale pour la classe source	Entier
T_MIN	La cardinalité minimale pour la classe terminale	Entier
T_MAX	La cardinalité maximale pour la classe terminale	Entier

UML_DATA_TYE

IDO	Identifiant du type de la donnée	Entier (séquentiel)
NOM	Nom du type des données possibles	Chaine de caractère

6.4.1.3. Remplissage de l'ontologie

Une interface web faisant appel aux scripts PL/SQL d'insertion des propriétés de l'ontologie permet au DBA de la source de données de remplir son ontologie locale de façon simple, cette interface est illustrée dans la figure suivante :

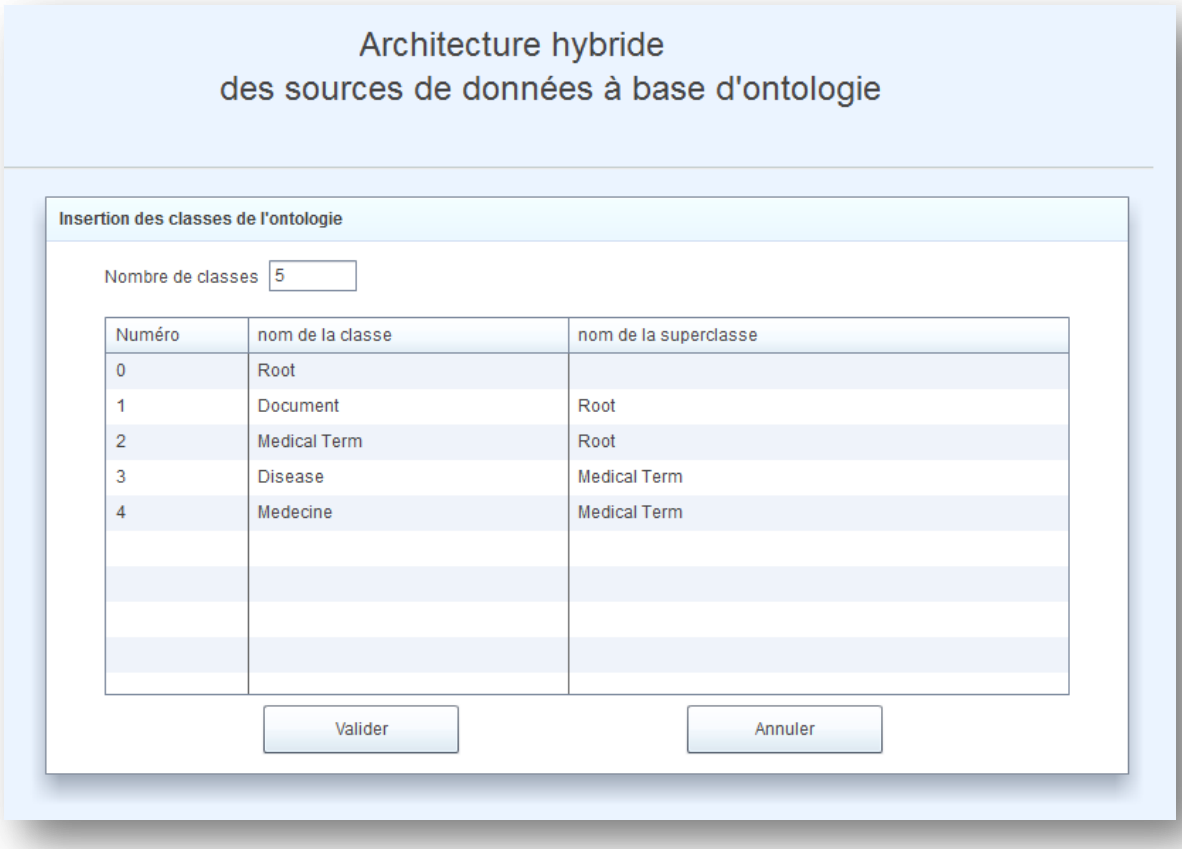

Figure 0.2 Interface d'insertion d'ontologie locale « exemple des classes »

6.4.1.4. Création de la source de données

Selon l'hypothèse qu'on avait fixé tout au début, seules les classes feuilles peuvent être instanciables c'est-à-dire que toutes les tables de la source de données sont des feuilles au niveau de l'ontologie locale.

L'administrateur va choisir les classes, les relations ainsi que les attribues constituant le schéma de sa base de données, pour cela, une interface web lui

permet de sélectionner les classes feuilles, leurs attributs ainsi que les relations cela minimisera le taux d'erreur.

La création du schéma de la base de données induit directement à la mise à jour des tables d'extensions qui garantie la liaison entre les deux parties de notre source de données à savoir l'ontologie et les instances.

Afin de permettre cette mise à jour, nous exécuterons le script PL/SQL « INSERT_EXTENSION_CLASS » qui permet l'insertion des enregistrements au niveau des tables « EXTENSION_CLASS » et « EXTENSION_ATTRIBUT » et le script « INSERT_EXTENSION_RELATION » pour l'insertion des tuples au niveau de la table « EXTENSION_RELATION » (voir annexe 2).

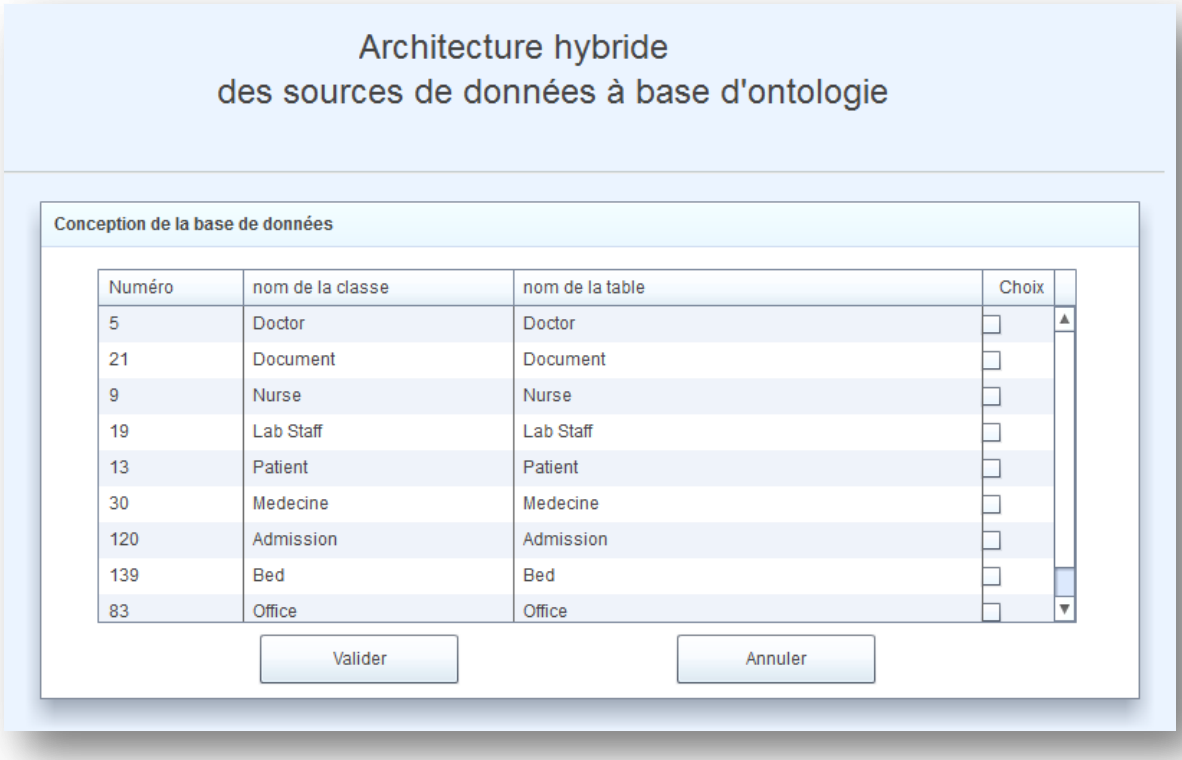

Figure 0.3 Exemple du choix conceptuel des tables de la source de données locale

6.4.1.5. Remplissage des instances de la source de données locale

Selon le schéma de la base de données choisi par le DBA, une interface web peut être générée afin de permettre la mise à jour des enregistrements des tables.

Dans notre cas, et du moment qu'on est juste en simulation, nous avons rempli directement nos sources de données à partir de données collectées depuis une source d'une clinique médicale, et aussi de la base de données de la sécurité sociale.

Le tableau suivant décrit les champs des tables d'extension des sources de données :

UML_EXTENSION_CLASS

Attribut	Description	Type
IDO	Identifiant	Entier (séquentiel)
Nom_classe	Nom de la classe feuilles faisant partie du schéma de la base de données	Chaine de caractère
Nom_table	Nom donnée à la table par le DBA	Chaine de caractère

UML_EXTENSION_ATTRIBUT

Attribut	Description	Type
IDO	Identifiant de l'attribut	Entier (séquentiel)
Nom_Classe	Nom de la classe où appartient l'attribut	Chaine de caractère
Nom_Attribut	Nom de l'attribut choisi	
Type_attribut	Le type de l'attribut	Chaine de caractère

UML_EXTENTION_RELATION

IDO	Identifiant de la relation	Entier (séquentiel)
Nom_Onto	Nom de la relation au niveau de l'ontologie	Chaine de caractère
Nom	Nom de la relation au niveau de la base de données	Chaine de caractère
Debut	IDO de la classe source de la relation	Entier
Fin	IDO de la classe terminale de la relation	Entier

6.4.2. Construction de l'ontologie partagée (l'ontologie de domaine)

Après la création des sources de données, nous avons mis en place l'ontologie partagée, l'architecture conceptuelle de cette dernière est la même que celle du méta-schéma dans les sources de données.

Notre ontologie de domaine qu'on avait choisi est conçu selon le modèle d'ontologie CIHO [49].

La figure suivante est un exemple de l'ontologie partagée qu'on a utilisé dans notre travail (l'ontologie CIHO) :

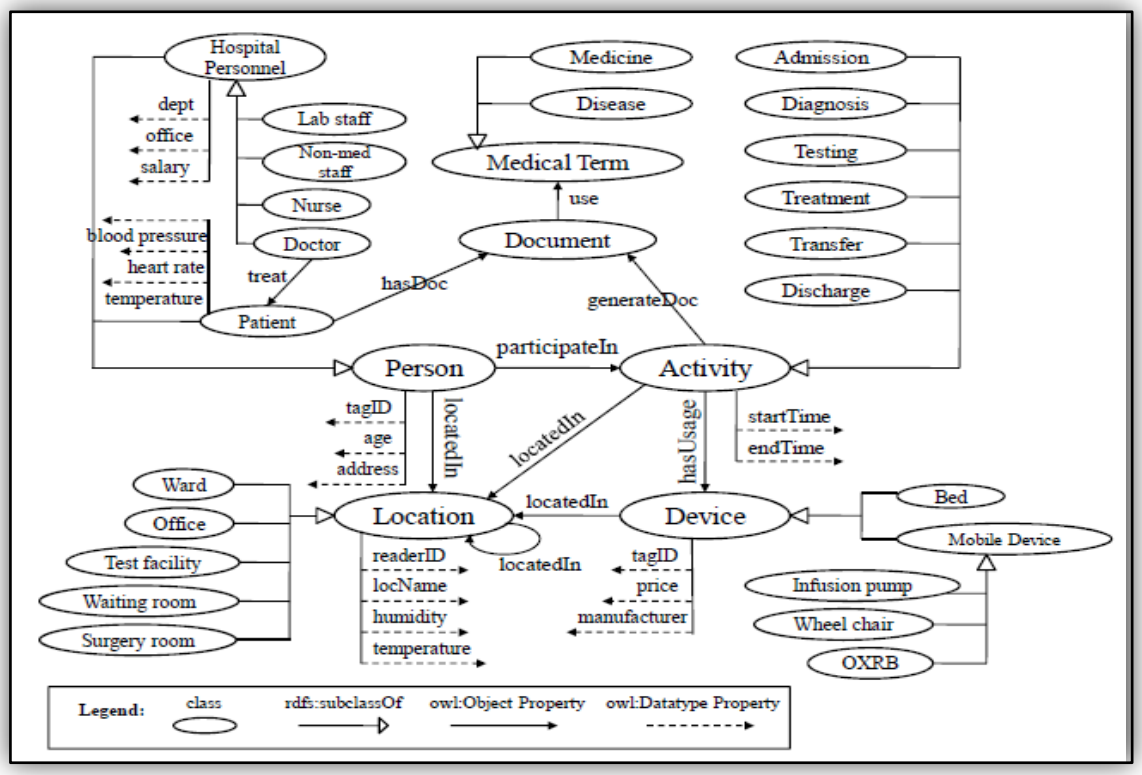

Figure 0.4 Schéma de l'ontologie CIHO [24]

A partir du schéma conceptuel de cette ontologie hospitalière, nous avons pu construire notre ontologie de domaine en XML qu'on avait nommé « HOnto », dans la figure suivante, nous présentons un fragment de notre ontologie.

Figure 0.5 Fragment de l'ontologie partagée « HOnto »

6.4.3. Construction du système d'intégration Hybride

Afin de mettre on place notre système d'intégration hybride à base d'ontologie, nous devons construire les deux couches principales à savoir l'entrepôt de données et le médiateur, cela est décrit dans les deux sous sections suivantes :

6.4.3.1. Construction de l'entrepôt de données (EDBD)

L'architecture conceptuelle de L'entrepôt de données à base d'ontologie (EDBO) est similaire à celle d'une base de données à base ontologique, avec en plus, deux autres tables qui sont : « SOURCE » contenant toutes les sources de données locales et leurs emplacements et « OBJET_SOURCE » qui nous permet de retrouver la (les) source(s) locale(s) pour chaque objet (classe, attribut, ou

relation) de l'ontologie faisant référence à une instance dans la partie « donnée » de l'entrepôt de données, ces tables sont nécessaire par la suite lors des mises à jours des vues matérialisées.

La construction de cet entrepôt de données est entièrement automatique, le programme passe par les étapes suivantes:

1. La création avec SQL des classes du « méta-schéma ».
2. la récupération de l'ontologie partagée et cela en exécutant le PL/SQL « RECUPERATION_ONTO ».
3. L'enrichissement de l'ontologie O_{ED} et cela déroulant l'algorithme « HybExtendOnto » sur la partie EDBO, cet algorithme est traduit en SQL dans le script PL/SQL qu'on a nommé : « ENRICHIR_ONTOED »,
4. La mise à jour des tables d'extensions,
5. Le remplissage des tables « SOURCE » et « OBJET_SOURCE »
6. La création de la méta-base
7. la récupération des instances sous formes de vues matérialisées existantes dans les sources de données et qui devront être intégrées dans l'entrepôt de données

La mise à jour des vues matérialisée n'a pas été prise en charge dans notre application, elle reste une perspective.

6.4.3.2. Construction du médiateur (MEDBO)

Notre médiateur a la même architecture conceptuelle que l'entrepôt de données (EDBO).

Le choix de laisser la partie données « méta base » est justifié du fait, d'une part, qu'on a besoin d'une partie cache pour la mise des données temporairement lors de la récupération des résultats des sources de données quand plusieurs sous requêtes sont envoyées à plusieurs sources de données locales, cela permet la réécriture des résultats finaux avant l'affichage, d'autre part, que la partie méta-

base pourra permettre par la suite à stocker les données fréquemment demandées, ce qui permettra un temps de réponse plus optimal.

On suivra les étapes de 1 à 6 de la construction de l'entrepôt de données pour la construction du médiateur (le code source est dans annexe 2).

6.4.4. Construction de la partie utilisateur

La partie utilisateur se compose de deux modules essentiels, le premier module est l'interface utilisateur, le deuxième est le moteur de requête

6.4.4.1. L'interface Utilisateur

L'interface utilisateur est une page web de recherche que nous avons programmé en langage PHP, l'utilisateur a le choix de rechercher un terme, des instances des sources de données, ou encore d'une ontologie.

Dans notre cas, nous avons programmé les deux premier choix, jusqu'à maintenant on devra écrire les requêtes SQL de sélection directement par exemple dans le cas d'une recherche d'instances l'utilisateur introduit la requête suivante :

```
Select * from "hospital personnel"
```

Et il aura par la suite toutes les instances existantes concernant tous les personnels des hôpitaux existants dans les sources de données locales (ou encore l'entrepôt de données) à savoir les instances des classes « Doctor », « Nurse », « Lab Staff », « Non-med staff »

Ces classes sont les classes filles de la classe «hospital Personal » et aussi se sont des classes feuilles donc qui peuvent exister réellement dans la partie « méta-base » des sources de données comme nous l'avons déjà expliqué dans les précédemment.

Les résultats seront récupérés et seront affichés à l'utilisateur en spécifiant même la table de chaque instance pour plus de clarté.

Nous avons traité des requêtes de sélection simples dont la structure s'approche des formats suivants :

$$Select\ *\ from\ table1\ where\ clause;$$

$$Select\ a, b\ from\ table1\ where\ clause;$$

$$Select\ table1.a, table2.b\ from\ table1, table2\ where\ clause;$$

La figure suivante présente Voici l'interface utilisateur où il peut saisir ses requêtes en choisissant le mode de recherche entre « ontologie », « terme » et « données »

Figure 6.6 Interface utilisateur (Moteur de Recherche Sémantique)

6.4.4.2. Le moteur de requête

Le moteur de requête permet la réécriture de la requête, en ce qui suit le pseudo-algorithme de notre moteur de requête :

Algorithme de recherche

Début

1. Récupération de la requête.
2. Décomposition de la requête entre attributs, tables et conditions.
3. vérification du nombre des tables

a.si il y a une seule table

a.1.Vérifier si la table est feuille

si oui

1. rechercher son emplacement dans la table "OBJECT_SOURCE"
2. vérifier si la table est instanciable (qui existe dans la partie donnée d'au moins une source de données)

Si oui

réécrire la requête en incluant le nom de source de données

Sinon

1. rechercher toutes les sous tables feuilles de notre table
2. Sélectionner les tables instanciables

a. Si ∃ une table instanciable

1. rechercher son emplacement de des tables "OBJET_SOURCE"
2. réécrire la requête en changeant le nom de la tables mère par la table feuille, et en incluant le nom de source de données
3. Récupérer les résultats

b. si nombre > 1

pour chaque tables

1.rechercher son emplacement dans la table "OBJET_SOURCE"
2. réécrire la requête en changeant le nom de la tables mère par la table feuille, et en incluant le nom de source de données
3. récupérer les résultats
4. Mettre les résultats dans le cache jusqu'à récupération de tous les autres résultats

b. sinon (il y a plusieurs tables)

b.1. réécrire la requête en plusieurs sous requête autant de sous requêtes que de tables
b.2. décomposer chaque sous-requête en attributs, tables et conditions.
b.3. refaire toutes les étapes de a.1 pour chaque sous requête table

b.4. Mettre des les résultats de chaque sous requête dans le cache.
b.5. Ré-exécuter la requête principale sur le cache du médiateur/

4. Afficher les résultats

Fin

Exemple de déroulement de l'algorithme

Prenons la requête suivante

$$\text{Select} * \text{from "hospital personnel"}$$

Nous allons dérouler notre algorithme de recherche

1. Après la décomposition de la requête, nous aurons une seule table appelée « hospital personnel », avec tous ses attributs

2. La table « hospital personnel » n'étant pas une classe feuille, l'algorithme va récupérer toutes ses sous tables feuilles qui sont « Doctor », « Nurse », « Lab Staff », « Non-med staff »

3. L'algorithme sélectionne d'abord les tables existantes dans au moins une source de données locales autant qu'une table de la base, ou encore dans l'entrepôt de données, La table « Non-med staff » sera alors éliminé du fait qu'elle n'est pas instantiable

4. Pour chaque table sélectionnée, l'algorithme recherche dans la table « OBJECT_SOURCE » l'emplacement de chaque table, et va réécrire les sous-requêtes comme suit :

$$\text{Select} * \text{from " Doctor"@"EDBO"}$$

$$\text{Select} * \text{from " Nurse"@"EDBO"}$$

$$\text{Select} * \text{from " Lab Staff "@"S1"}$$

Explication:

Les tables "Doctor" et "Nurse" existent en effet dans au moins deux sources de données locales, en appliquant précédemment le scénario d'intégration, les tables sont devenues des vues matérialisées au niveau de l'entrepôt de données, donc la source de données est maintenant « EDBO ».

Quant à la table « Lab Staff » elle n'existe que dans une seule source de données locale qui est « S1 » donc elle sera interrogée directement.

5. Les requêtes seront alors envoyé à leurs sources de données respectives pour exécution.

6. A la réception des résultats, l'algorithme s'en charge de les afficher/

Les résultats obtenus après exécution de cette requête sont illustrés dans la figure suivante :

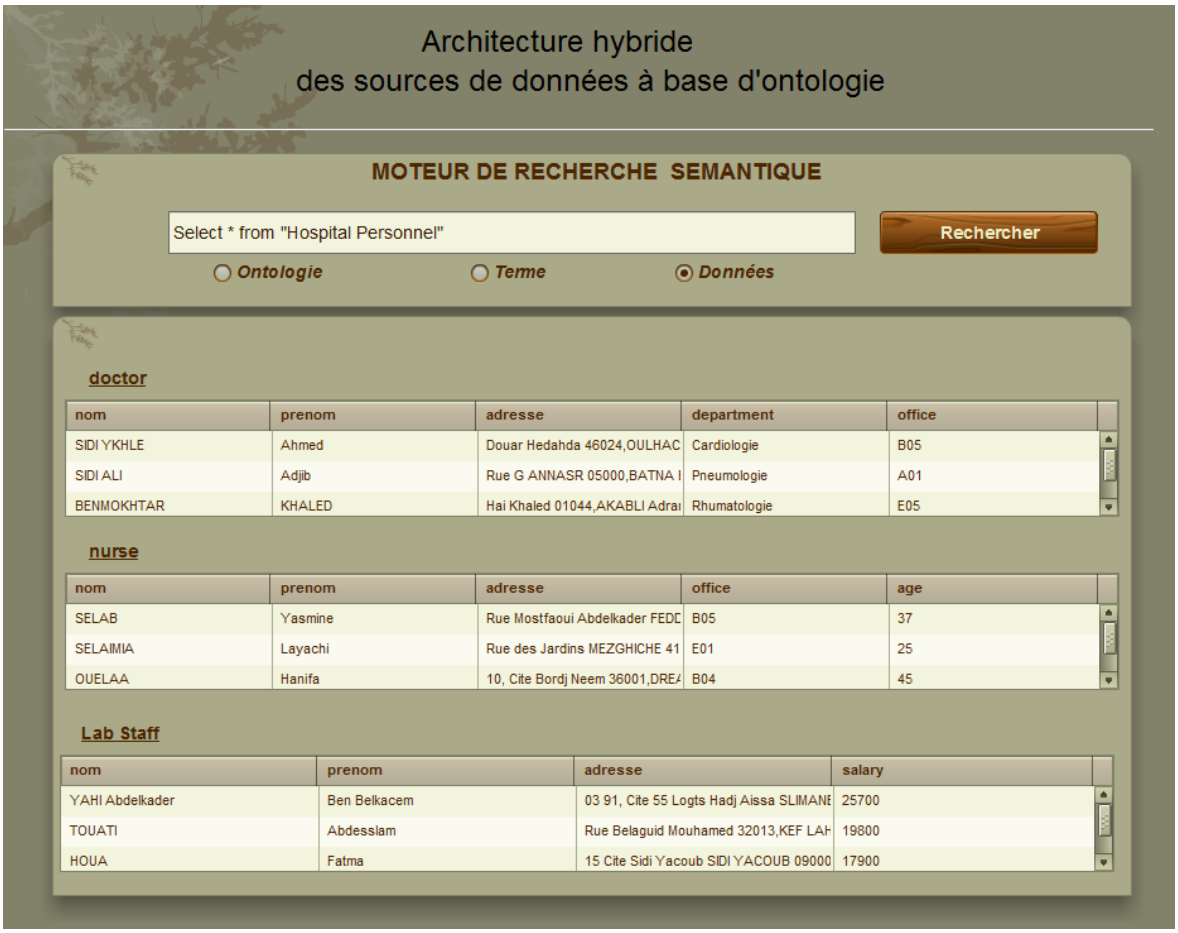

Figure 0.7 Exemple de résultats de recherche

Remarque :

L'adaptateur est implicitement inclus dans l'algorithme du fait que toutes les sources de données ont été créées sous SGBD ORALE.

6.5. Conclusion

Dans ce chapitre, nous avons présenté les étapes d'implémentation de notre prototype détaillé dans le chapitre 4, puis nous l'avons construit, chaque étape de construction est un ensemble de scripts PL/SQL.

Après la construction de notre architecture, nous sommes passées à l'intégration des sources de données, au niveau du médiateur et de l'entrepôt de donnée.

Finalement, nous avons mis en place l'interface utilisateur, et nous avons construit l'algorithme de recherche, cet algorithme est partagé en trois étapes :

1. La décomposition.
2. La localisation des données
3. La réécriture des résultats

Nous avons déroulé notre algorithme pour une requête, et nous avons affiché les résultats obtenus ainsi que les étapes d'exécution de notre algorithme de recherche.

CONCLUSION ET PERSPECTIVE

1. Conclusion

Les investigations scientifiques et les préoccupations abordées dans le cadre de ce mémoire de fin d'études relèvent globalement du domaine de l'intégration des données ;

Nous avons proposé une méthode complètement automatique d'intégration de sources de données structurées hétérogènes et autonomes. Cette approche, appelée système hybride d'intégration automatique des BDBO, suppose l'existence d'une (ou plusieurs) ontologie(s) de domaine mais, elle laisse chaque source autonome quand à la structure de sa propre ontologie.

Notre approche exige de l'administrateur de chaque source à intégrer (1) que sa base de données contienne une ontologie et (2) qu'il ajoute a priori à cette ontologie les relations (articulations) existantes entre celle-ci et l'ontologie de domaine. Cette hypothèse est réaliste dans tous les secteurs où des ontologies de domaines existent ou apparaissent, et où chaque administrateur qui publie sa base de données souhaite à la fois lui conserver sa structure propre, et la rendre accessible à des usagers de façon homogène à travers une ontologie de domaine. Elle exige également que chaque source publie non seulement ses données, mais également son ontologie. Nous avons proposé une mise en œuvre de cette approche pour les données structurées à travers le modèle de base de données à base ontologique (OntoDB).

Nous avons mis en place juste un prototype de cette approche dans une perspective d'intégration automatique des données au sein d'un système hybride c'est-à-dire qu'une partie des sources de données est intégrée dans un système

complètement matérialisé (l'entrepôt de données) et l'autre partie a subie une intégration virtuelle (juste le schéma qui est représenté au niveau du médiateur). Nous avons utilisé une petite ontologie hospitalière qu'on a adaptée pour qu'elle puisse représenter les entités du domaine. L'intégration automatique du contenu de toute nouvelle source autonome au sein du système et déjà constituée à partir de l'ontologie de domaine.

Beaucoup de questions, apparaissent actuellement ouvertes et nécessitent d'être explorées : (1) faisabilité de la mise en place réelle de cette même approche avec des modèles d'ontologie plus complexes, par exemple UMLS(2) enrichir notre architecture dans le coté mise à jour et maintenance des données intégrées au seins de l'entrepôt de données, (3) représenter explicitement les dépendances fonctionnelles pouvant exister entre propriétés, et les expressions de classes, voire le lien existant entre ontologie locale et l'ontologie partagée (synonymes, acronymes…).

Pour la mise à jour et la maintenance de notre système d'intégration que ça soit coté mise à jour des ontologies locales ou des données locales, de l'ontologie partagée, changement de la structure d'une des sources, nous envisageons la mise en place d'agent mobiles spécialisés selon chaque évènement et qui s'occupent d'alerter le système d'une éventuelle modification et de permettre le mapping entre différentes sources locales et le système d'intégration.

2. Perspectives

Ce travail ne peut être complètement et définitivement achevé, et reste ouvert à plusieurs perspectives, dont on cite :

- *Utilisation du système dans un cadre professionnel*

Les bases de données utilisées dans le cadre de notre thèse ont été créées dans le but de valider notre prototype. Nous les avons créées de telle manière qu'elles soient les plus proches possible de celles qui peuvent contenir des dossiers médicaux. Afin de rendre plus général notre système, nous allons l'utiliser aussi bien sur des bases de données hospitalières que dans d'autres domaines telles que les banques, les universités etc.

En effet notre système hybride n'est pas conçu que pour le domaine médical, il peut également être utilisé dans tout autre domaine nécessitant l'intégration de bases de données et la protection des données

- *Enrichir l'ontologie*

Notre ontologie ne représente qu'une petite portion d'une ontologie médicale dédiée au domaine hospitalier, elle ne contient pour le moment que quelques termes qui peuvent décrire les schémas des bases de données à intégrer. Il est donc impératif de présenter à l'utilisateur un outil qui lui permette d'enrichir l'ontologie.

- *Un système d'apprentissage*

Durant le processus de description de schéma et de création de la vue unifiée, l'utilisateur concepteur intervient dans la validation des résultats et le choix des concepts décrivant les bases de données. Il serait judicieux d'exploiter ces validations pour les utiliser dans les cas similaires.

Nous devrons donc définir un modèle d'apprentissage, qui dans les cas semblables, présente à l'utilisateur en premier les informations déjà validées.

- *Interface d'interrogation*

Les requêtes envoyées à notre médiateur sont écrites en SQL. Nous devrons créer une interface d'interrogation qui permette à l'utilisateur de créer sa requête en mode graphique. Il n'aura plus à apprendre un langage de requêtes mais simplement à sélectionner les relations et attributs à interroger.

APPENDICE

LES ONTOLOGIES

Introduction

Dans cette annexe nous allons compléter la première section du chapitre 3, nous allons synthétiser en premier lieu les étapes de construction d'une ontologie, nous présentons plus tard quelques exemples d'ontologies conceptuelles existantes pour le domaine médical.

1. Construction d'une ontologie

1.1. But de la construction de l'ontologie

Le développement des ontologies a quitté les laboratoires d'Intelligence Artificielle et s'est approprié des postes informatiques des experts de domaines, mais pour quelles raison développer une ontologie ? Plusieurs points de motivation sont offerts par les ontologies, les principaux sont :

a. Partager la compréhension commune d'un domaine entre les personnes ou des agents de logiciels [102], par exemple si un nombre de sites web contiennent de l'information médicale ou fournissent des servies de e-commerce, dans le cas ou ces sites partagent et publient tous la même ontologie cela permettra d'extraire et d'agréger l'information qui pourra être utilisée par des agents informatiques, pour répondre aux interrogations des utilisateurs, par exemple, ou comme des données d'entrée pour d'autres applications.

b. Permettre la réutilisation du savoir sur un domaine, par exemple, dans le cadre de la représentation d'évènements vidéos, plusieurs approches permettent de

représenter la notion de temps. Cette représentation comprend les notions d'intervalles de temps, de moments précis de temps, de mesures relatives de temps, etc. Si un groupe de chercheurs développe une telle ontologie, d'autres peuvent tout simplement la réutiliser pour d'autres applications.

c. Rendre explicites les postulats d'un domaine ce qui rend possible la modification de ces spécifications en cas d'évolution du savoir sur le domaine. Les spécifications explicites du savoir sur un domaine sont utiles pour les nouveaux utilisateurs qui doivent apprendre la signification des termes du domaine.

d. Distinguer le savoir sur un domaine du savoir opérationnel est une des finalités courantes des ontologies, ainsi on peut décrire la tâche de configuration d'un produit à partir de ses constituants, tout en respectant les spécifications requises et implémenter un programme qui réalisera cette configuration indépendamment des produits et de leurs composants [126].

e. Analyser le savoir sur un domaine, ce qui est possible dès que la spécification des termes du domaine est faite. L'analyse formelle des termes est extrêmement précieuse aussi bien quand on veut réutiliser les ontologies existantes, que quand on veut les étendre [127].

f. Finalement l'analyse ontologique clarifie la structure de la connaissance, plus qu'un vocabulaire spécialisé, pour un domaine, les ontologies fournissent les conceptualisations des termes du domaine, de plus, l'analyse formelle des termes est extrêmement importante pour la réutilisation ou l'extension d'ontologies existantes.

3. Les principes de constructions d'une ontologie

Le processus de construction d'une ontologie, quelque soit la méthode choisie, doit respecter quelques principes de bases, d'après Gruber [102], les principaux critères à garder et à respecter sont :

- La clarté et l'objectivité : toute ontologie doit offrir des définitions objectives ainsi qu'une documentation en langage naturel des termes.
- L'exhaustivité : de préférence toute définition doit être exprimée par une condition nécessaire et suffisante.

- La cohérence : afin de formuler des inférences cohérentes avec les définitions.
- L'extensibilité monotone maximale : tout nouveau terme doit être inclus dans l'ontologie sans entraîner de modifications dans les définitions existantes.
- L'intervention ontologique minimale : cela signifie que l'ontologie devra spécifier le moins possible le sens de ses termes de façon de laisser aux parties impliquées la possibilité de spécialiser et d'instancier l'ontologie à leur guise.

4. Les ontologies en médecine

La médecine est un domaine complexe qui utilise de très nombreux termes et concepts. La communication de plus en plus importante d'informations médicales entre les différents professionnels de santé rend le recours aux ontologies tout à fait indispensables.

En ce qui suit, quelques ontologies médicales :

a. ON9

ON9 est une librairie d'ontologies décrite en Ontolingua [102] et LOOM. Elle inclut des milliers de concepts. Ce travail a mené à une intégration de cinq systèmes de terminologie médicale. Pour ce faire la méthodologie ONIONS (ONtology Integration On Naïves Sources) a été utilisée pour construire cette ontologie. ON9 inclut des milliers de concepts médicaux. Parmi les ontologies qui la constituent, SNOMED, ICD (identifient les maladies), UMLSSN (orienté vers les connaissances scientifiques).

b. GALEN

Le Système GAL EN est un Projet Européen (Anglais, Hollandais, Suisse) dont l'objectif est la modélisation des concepts médicaux, et la production d'un serveur de terminologie générique voulant représenter toute la terminologie médicale au sein d'une ontologie de concepts. Il est composé d'un ensemble de concepts et de relations (rôles). La relation IS-A est l'opération fondamentale. Elle donne aux concepts une structure de graphe dirigé acyclique.

c. UMLS

UMLS (Unified Medical Language System), est un vaste projet conçu par la NLM (National Library of Medicine) aux Etats-Unis. Son objectif est d'unifier les

classifications médicales et de présenter un thésaurus pour la recherche d'information. Il est composé d'environ 40 vocabulaires qui contiennent plus de 700 milles concepts biomédicaux et plus d'un million de termes pour les décrire. Il intègre actuellement la quasi-totalité des nomenclatures standards utilisées pour coder ou indexer l'information biomédicale. La base de connaissances de UMLS contient, entre autres, une ontologie et un méta thesaurus constituant une base unifiée des concepts médicaux. La version de 2000 contient 730 000 concepts, 1 330 000 Termes et 1 718 000 Chaînes. Le méta-thesaurus recense tous les concepts inventoriés dans les nomenclatures intégrées à UMLS. Il est constitué de 135 types sémantiques (Exemple Types anatomiques) reliés par des relations. Les 135 types sémantiques constituent une hiérarchie (un arbre) structurée par la relation IS-A. Chaque concept du méta-thésaurus possède un ou plusieurs types sémantiques.

d. Nautilus

Nautilus est une ontologie médicale sous forme de base de données contenant une nomenclature où les termes sont reliés entre eux par des liens de type: "est un", "se situe sur", "se mesure en". Cette ontologie a été étendue par l'équipe ACACIA en y intégrant des concepts qui concerne les réseaux de soin. Une fois ces concepts intégrés, leurs relations avec les autres concepts existant déjà dans l'ontologie Nautilus ont été définis. Nautilus est représentée dans un formalisme standard de représentation des connaissances, RDF(S) (Res ource Description Framework Schema). RDF(S) est un langage recommandé par le W3C pour la description des ressources du Web, il est intéressant pour décrire des ressources accessibles via un réseau local, un intranet ou un extranet.

e. MED

MED (Medical Entities Dictionary) est un vocabulaire contrôlé servant de passerelle normalisée entre les vocabulaires de différents systèmes cliniques du « Columbia Presbyterian Medical Center ». L'ensemble des concepts est hiérarchisé par la relation IS-A, et forme un graphe dirigé acyclique : certains concepts peuvent posséder plusieurs pères. Les concepts du haut du graphe ont été repris du métathésaurus UMLS. MED décrit chaque concept sous forme d'un objet possédant des propriétés et des relations sémantiques, qui sont héritées par les

objets fils. Cette description structurée permet l'accès aux concepts selon leurs propriétés.

Les ontologies générales de la médecine représentent un très grand nombre de concepts. Ceci a poussé à chercher à développer des ontologies partielles relatives à une pathologie spécifique par exemple cancer du sein, maladies coronariennes.

f. Ontologie du cancer du sein

Une ontologie du cancer du sein a été conçue dans le cadre du projet européen In face. Elle a été développée à partir d'une ontologie générale de L&C, qui comporte 5000 concepts liés au cancer du sein. En plus de l'ontologie L&C de départ, conçue par la société belge « Language and Computing », des corpus de textes anglais et français du domaine ont été utilisés. Des termes provenant de la nomenclature médicale MEDCIN ont été également pris en compte. Le corpus français contenait 670 fichiers, dont 68 articles scientifiques, 20 comptes-rendus médicaux et des pages web contenant des textes provenant de diverses nomenclatures et classifications. Le corpus anglais était plus centré sur des textes scientifiques.

g. MENELAS

MENELAS est une ontologie construite pour les pathologies coronariennes, elle est constituée de 1800 types de concepts atomiques et 300 relations, structurés en arbre contenant des relations de type IS-A. MENELAS comprend aussi des lexiques sémantiques et morpho - syntaxiques de mots simples et composés.

5. Méthodes et cycles de développement d’une ontologie

5.1. Méthode de développement d’une ontologie :

Une ontologie peut être construite selon trois méthodes distinctes :

a. Méthode manuelle :

Les experts réalisent une nouvelle ontologie d’un domaine ou étendent une ontologie existante en s’appuyant sur des techniques classiques de collecte et d ‘analyse des connaissances.

b. Méthode automatique :

Se base sur des méthodes formelles et des techniques d'extraction des connaissances en employant des outils linguistiques et statistique. Parmi les méthodes qui peuvent être utilisées, l'utilisation des méthodes de classification automatiques issues de la théorie de l'information ou encore les méthodes de regroupement conceptuel (Conceptual Clustering) qui segmentent automatiquement les textes en unités thématiquement homogènes, et regroupent ces unités en fonction d'une mesure de similarité fondée sur la fréquence des mots.

c. La méthode mixte :

Les ontologies sont construites par des techniques automatiques tout en intégrant des méthodes permettant d'étendre des ontologies ayant été construites manuellement.

5.2. Cycle de développement d'une ontologie :

Le cycle de développement général d'une ontologie se déroule en trois principales étapes qui sont : la conceptualisation, la formalisation (appelée aussi ontologisation) er l'implémentation. Ces étapes sont généralement précédées par une étape d'évaluation des besoins et de délimitation du domaine de connaissances à modéliser

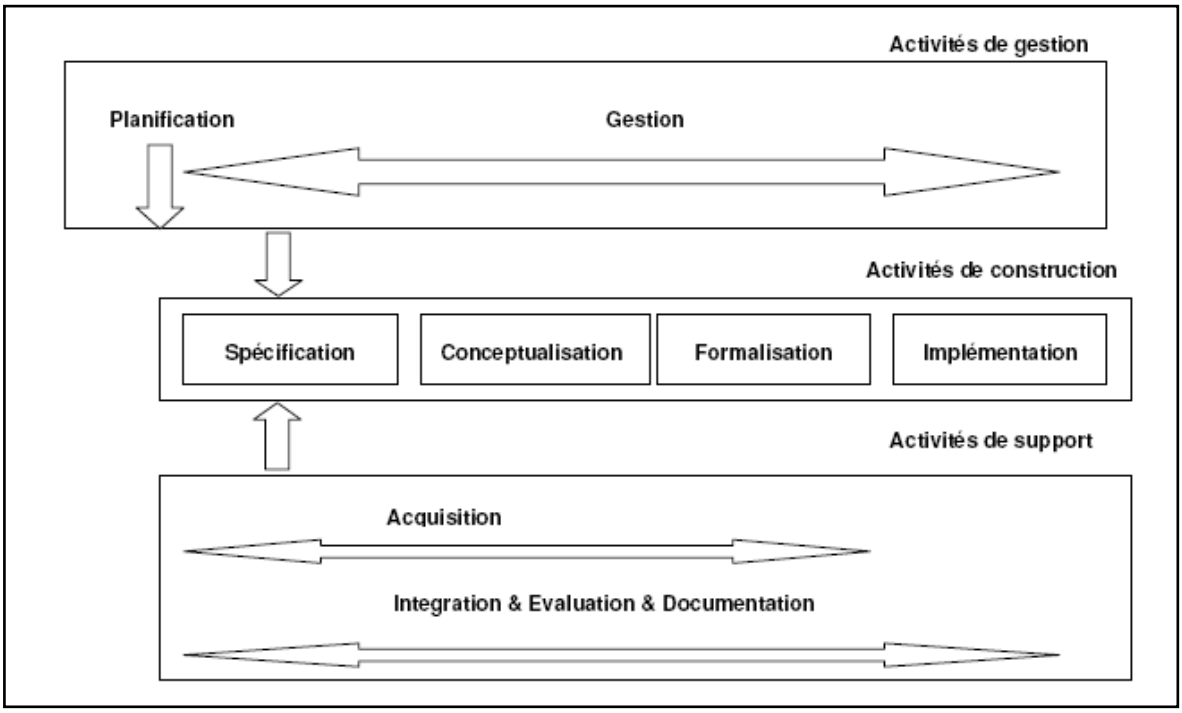

Figure 1. Cycle de vie d'une ontologie

a. L'évaluation des besoins :

L'évaluation des besoins doit cerner le but visé par la construction de l'ontologie. Cette évaluation des besoins, se décline en trois aspects qui sont l'objectif opérationnel de l'ontologie, le domaine de connaissances à modéliser et les utilisateurs potentiels. En effet, il est indispensable de bien préciser l'objectif opérationnel de l'ontologie, en particulier à travers des scénarios d'usage. Aussi, le domaine de connaissances doit être délimité aussi précisément que possible, et découpé si besoin en termes de connaissances du domaine, connaissances de raisonnement et connaissances de haut niveau. La délimitation du domaine de connaissance repose sur l'utilisation de ressources textuelles et/ou multimédia, constituant le corpus du domaine et au travers desquelles peuvent être appréhendées la terminologie du domaine et les significations des concepts.

Un domaine n'est pas seulement défini par le champ de connaissance qu'il couvre, mais aussi par le point de vue sous lequel les utilisateurs de l'ontologie considèrent ce champ de connaissances. Ces utilisateurs, doivent être identifiés en accord avec l'objectif opérationnel ainsi que le degré de formalisation de l'ontologie.

b. La conceptualisation

La conceptualisation est un processus d'abstraction qui consiste à identifier les concepts essentiels du domaine de connaissances et d'établir les relations entre ces concepts, au sein d'un corpus représentatif du domaine. Il s'agit donc de décrire le domaine de connaissances grâce à des concepts plus ou moins précis et aux relations qui peuvent exister entre ces concepts. L'identification des concepts et relations peut se faire selon l'analyse des textes (documents, notes, comptes rendus d'interviews, etc.) Cette analyse est généralement « une analyse informelle des textes qui peut être doublée par une analyse automatique permettant de détecter les termes et structures sémantiques (définitions, règles) présentes dans le corpus de documents ».

Mais cette analyse ne peut suffire, toute seule, à spécifier la sémantique du domaine, car certaines connaissances ne prennent sens que lorsqu'elles sont lues par un expert ou un spécialiste du domaine. La sémantique doit donc être précisée ou validée par les experts du domaine.

Après l'identification des concepts et termes importants, il faut leur attribuer des définitions. Le but de la définition du sens des concepts dans une ontologie est différent de celui de la définition du dictionnaire. Les définitions d'ontologie ont un rôle normatif et indiquent comment un ensemble réduit de termes peut être utilisé par rapport à un autre. Ainsi, chaque définition exige une bonne compréhension, notamment dans sa relation avec les autres définitions dans l'ontologie. L'usage de définitions explicites en langage naturel favorise le partage sémantique.

Néanmoins, la principale difficulté rencontrée durant le processus de la conceptualisation réside dans l'émergence de différentes significations ou des contradictions au niveau du sens prêté à certains concepts ou relations. De fait, une normalisation sémantique est nécessaire. Cette normalisation doit être le fruit d'un dialogue entre les experts. De manière générale, l'échange entre experts est le meilleur moyen de faire émerger une sémantique claire et non ambiguë.

« L'engagement ontologique est une garantie de cohérence entre une ontologie et un domaine» [102], car il spécifie les objets du domaine qui

peuvent être associés au concept, conformément à sa signification formelle.

Un concept est ainsi doté d'une sémantique référentielle (celle imposée par son extension) et d'une sémantique différentielle (celle imposée par son intension).

Au cours de ce processus, les connaissances sont structurées grâce notamment à l'utilisation des relations et aux définitions spécifiant le sens associé à chaque concept. « La conceptualisation structure les connaissances identifiées, c'est à dire, les concepts, instances et relations en représentations intermédiaires semi-formelles, comme le dictionnaire des données, les arbres de classification de concepts, et bien d'autres » [118]. Ainsi, le processus de la conceptualisation mène à l'élaboration d'un modèle conceptuel qui décrit, au travers des éléments terminologiques et sémantiques les connaissances du domaine. Ce modèle conceptuel peut être complètement informel (exprimé en langage naturel et présenté sous forme de tableau par exemple) ou semi-formel, combinant langage naturel et propriétés formelles telle que les diagrammes de classes UML. La structuration des connaissances, le consensus ontologique et la formalisation de l'ontologie font partie d'un processus appelé l'ontologisation.

c. La formalisation

La formalisation concerne le modèle conceptuel obtenu à l'étape précédente. Elle peut être vue comme une représentation explicite et formelle de la conceptualisation. Elle est réalisée par le biais d'un langage formel ou formalisme qui est un ensemble de composants sémantiques (contenu), de règles structurelles (mode d'emploi) et d'une notation formelle particulière (forme) destinée à organiser les relations entre les éléments constituant l'ontologie. L'objectif de l'utilisation d'un langage de formalisation d'ontologies, est de permettre d'une part de réduire les ambiguïtés du langage naturel en offrant une plus grande expressivité ; et d'une autre part de rendre l'ontologie compréhensible par les machines.

«Les formalismes offrent un support formel à la composition des concepts et à leur comparaison».

Différents formalismes tels que les logiques de description, les réseaux sémantiques et les frames (schémas) peuvent être employés pour représenter formellement une ontologie. Les logiques de description représentent la connaissance sous forme de propositions ou affirmations sur le domaine. Les réseaux sémantiques, tout en gardant cette approche propositionnelle, tiennent compte de la structure et les relations entre ces propositions. Les frames représentent le domaine en termes de ses objets et leurs propriétés et relations. Ces paradigmes se différencient les uns des autres par leur formalisme de représentation des connaissances et leurs mécanismes d'inférence permettant de raisonner sur les représentations. Compte tenu de la diversité des formalismes de représentation des connaissances, les concepteurs d'ontologies sont amenés à prendre en considération certains critères avant de choisir le langage le plus adéquat.

A la fin de l'étape de formalisation, on obtient une ontologie formelle. Cependant, certaines connaissances du domaine peuvent être abandonnées, du fait de l'impossibilité de lever certaines ambiguïtés, ou du fait des limitations de l'expressivité du langage utilisé.

d. L'implémentation

L'implémentation consiste à outiller une ontologie pour permettre à une machine de manipuler des connaissances du domaine via cette ontologie. La machine doit donc pouvoir utiliser des mécanismes opérant sur les représentations de l'ontologie.

6. Langages d'ontologies

Pour implémenter des ontologies, plusieurs langages ont été utilisés. La plupart sont basés sur XML et les logiques de description, les plus populaires sont : XML, XMLSchema, RDF, RDFSchema et OWL :

a. XML (eXtensible Markup Langyage)

Recommandé par le W3C, est une spécification destinée à rendre des documents lisibles par une machine.

```
<employee>
  <name="Son">
   <phone>7634</phone>
  </ name>
 </employee>
```

Figure 2 Exemple de représentation XML

XML fournit une structure syntaxique pour des documents, et ne permet pas une interprétation sémantique des données.

b. XML Schema

Permet de définir les balises ainsi que l'agencement de ces balises autorisé pour définir la validité d'un document XML.

c. RDF et RDFS : (Ressource Description Framework) :

Le *Resource Description Framework* (RDF) est un langage pour la représentation de l'information concernant les ressources disponibles sur le Web. Les informations associées à une ressource couvre une large palette d'attributs simples tels que le titre, l'auteur, la date de modification, jusqu'aux données plus complexes telles que le contenu ou bien les droits d'auteurs. Le langage RDF a été développé par le consortium W3C. Depuis 2004, ce langage est considéré comme un standard par ce même consortium dans le cadre de ses activités organisées autour du Web sémantique. RDF rend possible l'interopérabilité sémantique entre applications qui échangent des données dans un langage machine. En effet, il permet de donner du sens aux ressources et a été conçu afin d'être facilement lu et compris par les machines.

➢ Présentation générale

RDF repose sur un modèle logique qui permet d'énoncer des propositions qui décrivent des ressources (*i.e.* des composants documentaires) en utilisant des triplets {*ressource*, *propriété*, *valeur*}.

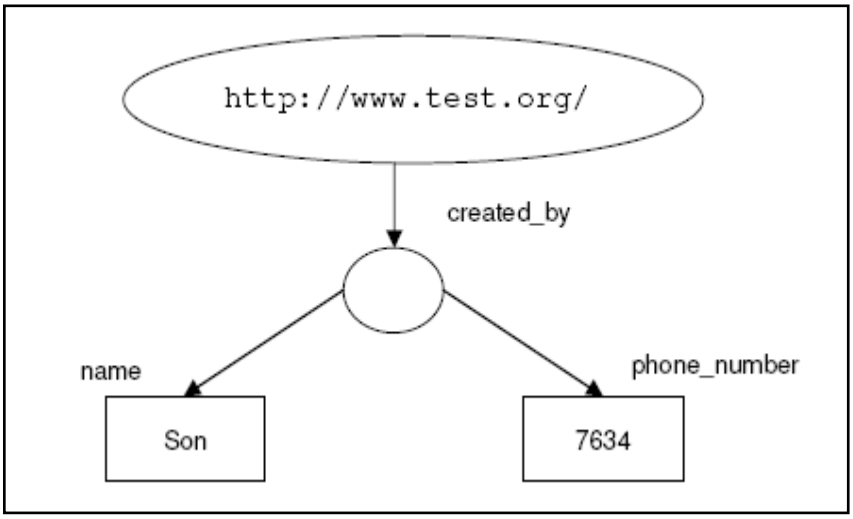

Figure 3 Représentation d'un graphe de triplet

Les triplets peuvent être également représentés par un graphe étiqueté, orienté, aux propriétés mathématiques bien définies. Ces triplets constituent la construction de base en RDF :

- la *ressource* est toute entité d'information pouvant être référencée en un bloc, par un nom symbolique (littéral) ou un identificateur (URI15). Les URI utilisés ne pointent pas forcément vers des documents consultables sur le Web et peuvent référencer n'importe quel objet identifiable (tel qu'une page Web, un livre, un morceau de musique). En plus, les propriétés RDF elles-mêmes peuvent être associées à des URI afin de pouvoir définir de manière précise les relations entre les items liés par la propriété en question ;
- la *propriété* est un aspect spécifique, une caractéristique, un attribut ou une relation utilisée pour décrire une ressource. Chaque propriété a une signification spécifique, des valeurs autorisées, des types de ressources qu'elle peut décrire et des relations avec d'autres propriétés ;
- la *valeur* de la propriété.

Une ressource spécifique associée à une propriété et une valeur constituent ensemble une expression ou proposition RDF. Ces trois parties (ressource, propriété, valeur) sont appelées respectivement sujet, prédicat et objet.

RDF offre également une syntaxe de type XML (nommée *RDF/XML*) pour l'enregistrement de ces graphes. Tout comme HTML, RDF/XML est facilement lu et interprété par un ordinateur.

Un des grands avantages de RDF est son extensibilité, à travers l'utilisation de schémas RDF (*RDF Schema*) qui peuvent s'intégrer et ne s'excluent pas mutuellement grâce à l'utilisation du concept d'espace de noms. Les déclarations RDF définissent des relations entre des objets (noeud d'un graphe) qui appartiennent à un univers sémantique. À chacun de ces univers sémantiques correspond un domaine nominal, identifié par un préfixe particulier. Un tel domaine, pour lequel sont définies des propriétés spécifiques et des catégories conceptuelles, est appelé un schéma. La notion d'espace de noms est une notion fondamentale puisque c'est grâce elle que l'on peut étendre les schémas de descriptions existants.

- Extensibilité du langage RDF

Tout utilisateur a la possibilité de créer un nouveau schéma, de modifier, et notamment d'enrichir et d'affiner un schéma existant. Ceci permet de définir un nouveau schéma personnalisé qui répond mieux aux besoins de caractérisation des propriétés d'objets existant dans un nouveau contexte.

RDFS (pour RDF Schema) offre les moyens de définir un schéma de métadonnées qui permet de :

- donner du sens aux propriétés associées à une ressource en utilisant la construction rdfs:class qui permet de déclarer une nouvelle classe de ressources ;
- formuler des contraintes sur les valeurs associées à une propriété :
  - ✓ en utilisant la construction rdfs:range qui permet d'introduire une restriction du domaine de valeurs d'une propriété – par exemple, pour une propriété qui caractérise l'âge d'une personne on peut imposer que les valeurs soient des entiers;

- ✓ ou bien en utilisant la construction rdfs:domain pour restreindre le domaine d'application de propriétés – par exemple, pour une propriété qui caractérise un auteur par sa date de naissance, on peut exiger que les valeurs de cette propriété soient une référence à une personne.

Une propriété spécifiée dans un schéma RDFS est vue comme une ressource dont le type rdf:type est rdf:property.

a- OWL (Ontology Web Language) :

OWL a été conçu afin de maîtriser l'hétérogénéité inhérente au processus de descriptions.

Il permet d'exploiter les informations contenues dans les documents directement par les applications sans intervention humaine. OWL est un vocabulaire XML basé sur RDF. Ce langage est aussi un moyen standard de décrire les relations entre informations ce qui peut amener par la suite à la construction d'un réseau de confiance à propos de ces informations.

OWL représente de manière explicite le sens des termes et les relations entre eux, aux moyens de vocabulaires interconnectés. OWL est beaucoup plus expressif en termes de sémantique que les normes XML, *XML Schema*, RDF ou *RDF Schema*.

XML offre uniquement une syntaxe pour structurer les documents. *XML Schema* ajoute des restrictions sur la structure des documents XML et permet la création de nouveaux types de données. Mais ces deux langages ne s'intéressent qu'à la syntaxe descriptive, aucun des deux n'impose de contraintes sémantiques sur le sens des documents. RDF ajoute une sémantique simple au modèle de données permettant de décrire des ressources et les relations entre-elles.

RDF Schema est un vocabulaire RDF qui permet de décrire et organiser les propriétés et les classes de ressources au sein d'une hiérarchie. OWL va plus loin que ces langages car il est capable de représenter de l'information interprétable par les agents logiciels sur le Web en considérant, entre autres, les relations existantes entre classes et/ou instances, leurs cardinalités, les caractéristiques des propriétés (symétrie, transitivité), etc.

OWL s'impose également en permettant de déclarer des équivalences entre classes d'information, de construire de nouvelles hiérarchies et de nouvelles

classes en utilisant des opérations ensembliste telles que l'union, la disjonction, la différence, etc.

OWL est une révision du langage des ontologies Web DAML+OIL suite aux expériences de conception et d'utilisation de ce langage dans des applications sémantiques du langage. OWL est défini en trois profils de plus en plus expressifs, chacun étant une extension du précédent : *OWL Lite*, *OWL DL*, *OWL Full*.

- Le profil OWL Lite

OWL Lite est un ensemble minimal de construction OWL permettant aux utilisateurs d'organiser les informations, les propriétés et les classes de ressources dans une hiérarchie et d'ajouter des contraintes simples sur la cardinalité des rôles (un ensemble est limité à 0 ou 1 élément) dans une classification hiérarchique. *OWL Lite* permet de réaliser des annotations sur les classes, les propriétés, les individus ainsi que les définitions des ontologies. *OWL Lite* reprend toutes les fonctionnalités de *RDF Schema* liées à la définition et la construction des hiérarchies de classes et propriétés, aux annotations et la définition de nouveaux types de données.

OWL Lite ajoute des constructions permettant de définir l'équivalence entre les extensions de classes (owl:equivalentClass) et de propriétés (owl:equivalentProperty) ou de se prononcer sur les différences ou les similitudes entre les instances (owl:sameAs, owl:differentFrom, owl:AllDifferent, owl:distinctMembers). Ces outils facilitent la mise en correspondance d'ontologies car des relations (synonymies, antinomies, etc.) peuvent être définies entre concepts issus de plusieurs ontologies.

OWL Lite fait la distinction entre les propriétés caractérisant les instances d'une classe au moyen d'autres instances (owl:ObjectProperty) et celles liant les instances aux données simples (owl:DataProperty).

OWL Lite introduit des restrictions (owl:Restriction, owl:onProperty) sur les valeurs d'une propriété. La portée de la restriction peut être restreinte à un sous-ensemble (owl:someValuesFrom) ou bien elle s'applique à toutes les valeurs (owl:allValuesFrom).

OWL Lite permet également de construire de nouvelles classes en considérant les instances communes de deux classes existantes. Ce premier niveau de langage OWL se propose comme un langage permettant une migration rapide des thésaurus et autres taxonomies vers OWL. *OWL Lite* est le point de départ pour l'introduction d'outils plus expressifs.

- Le profil OWL DL

Par rapport à *OWL Lite*, de nouvelles constructions de *OWL DL* permettent de définir des classes énumérées.

Les classes sont définies par l'ensemble de leurs instances (owl:one of – la classe *JourDeLaSemaine* est définie par ses instances *Lundi*, *Mardi*, …, *Dimanche*). La définition de nouvelles classes en considérant les opérations ensemblistes ne se restreint plus à l'intersection (owl:intersectionOf) sur les instances issues de deux classes existantes, mais s'étend également à l'union (owl:unionOf), au complément (owl:complementOf) et à la disjonction (owl:disjointWith).

Dans le même souci d'enrichissement de l'expressivité du langage, il est désormais possible de définir l'équivalence, ou la relation d'héritage, entre des classes anonymes construites comme indiqué précédemment.

OWL DL se base sur les Logiques de Description (*Description Logic* – DL) et s'adresse aux utilisateurs qui veulent un maximum d'expressivité en garantissant la complétude (toute conclusion est calculable) et la décidabilité (toute conclusion est calculée dans un temps fini) de tous les raisonnements.

OWL DL inclut toutes les constructions du langage OWL, mais leur utilisation est soumise à certaines restrictions. Par exemple, une classe ne peut pas être une instance ou une propriété d'une autre classe, une propriété doit être définie explicitement en tant que propriété soit de type individu (owl:objectProperty) soit de type donnée (owl:dataProperty).

- Présentation du profil OWL Full

À la différence d'*OWL DL*, la version *Full* du langage ne garantit ni la calculabilité ni la décidabilité des conclusions. Par exemple, une classe peut être considérée simultanément comme une collection d'individus et un individu en soi.

Ceci est dû au fait qu'une ressource peut être tant une classe qu'une instance d'une autre classe. Il est peut probable qu'un raisonneur puisse prendre en compte tous les aspects d'*OWL Full* à cause de la liberté d'expression que le langage offre.

La puissance d'expressions de solutions mariant OWL, RDF et *RDF Schema* sont intéressantes par le fait qu'elles peuvent couvrir l'ensemble des propriétés sémantiques que l'on peut associer à une ressource Web (par exemple, données multimédia ou données 3D).

Cependant, il n'existe pas dans ces langages, de structures prédéfinies capables d'accueillir les caractéristiques des données multimédia et, en conséquence, il faut donc les définir explicitement. Dans la sous-section suivante, nous présentons MPEG-7, un standard universel des descriptions qui contient de constructions spécifiques aux données multimédia.

REFERENCES

1 Litwin W., Abdellatif A., Zeroual A., Nicolas B., Vigier Ph., "MYSQL : A Multibase Language", (1989).

2 Sheth A-P, Larson J-A. “Federated database systems for managing distributed, heterogeneous, and autonomous databases”. ACM Computing Surveys, (1990), 1(3) : 173-236.

3 Wiederhold G. “Mediators in the architecture of future information systems”. Computer, 25(3) : 38-49, (1992).

4 Widom J.. Research problems in data warehousing, in Conference on Information and Knowledge Management, (1995).

5 Nejal W., Wolf B., Qu C., Decker S., Sintek M., Naever A., Nilsson M., Palmer M., Risch T. ”Edutella : A P2P networking infrastructure based on rdf”, in Proceeding of the 11th International Conference on World Wide Web, (2002).

6 Rousset M-C., Reymaud C. “PICSET and HYLEME : Two illustrative information integration agents”, in Intelligent Information Agents – The AgentLink Perspective, Springer, (2003).

7 Mena E., Illarrmedi A., Kashyap V. et Sheth A-P. “OBSERVER : an approach for query processing”. In global information systems based on interoperation across pre-existing, (2000).

8 Arens Y., Hsu C-N. et Knoblock C-A, “Query processing inf the SIMS information mediator”, in HUHLS M-N et SIN§GH M-P, éditeurs : Reading in agents, Morgan Kaufmann, San Francisco CA, USA, (1997).

9 Perez-Rey D., Maojo V., Garcia-Remesal M., Alonso-Calvo R, Billhardt H;

Martin-Sanchez F., “Sousa A : ONTOFUSION : Ontology-based integration of genomic ans clinical databases”. Computers in Biology and Medecine, (2006).

10 Stevens R., Baker P-G., Bechhofer S., NG G., Jacoby A., Faton N-W., Goble C-A et Brass A., “TAMBIS : Transparent Access to Multiple Bioinformatics Information Sources”. Bioinformatics, (2000).

11 Levy H.A.., Rajaraman A., Ordille J., “Data integration: the teenage years”, In VLDB '06: Proceedings of the 32nd international conference on Very large data bases, (2006), pages 9-16. VLDB Endowment.

12 Hsu C.-N., Knoblock C. A., “Semantic query optimization for query plans of heterogeneous multidatabase systems”, IEEE Trans. on Knowl. and Data Eng., (2000), 12(6):959-978,.

13 Haas L. M., Beauty and the beast: The theory and practice of information integration, In ICDT, (2007), pages 28-43.

14 Heimbigner D., Mcleod, D., “A federated architecture for information management”, ACM Transactions on Office Information Systems, (1985), 3:253-278.

15 Elmagarmid A., Rusinkiewicz M., Sheth A., “Management of Heterogeneous and Autonomous Database Systems”, Morgan Kaufmann Publishers, 3rd édition, (1999).

16 Hull R., "Managing semantic heterogeneity in databases: a theoretical prospective”, In PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pages 51-61, New York, NY, USA. ACM, (1997).

17 Parent C, Stefano S. "Intégration de bases de données : Panorama des problèmes et des approches". Ingénierie des systèmes d’information, École Polytechnique Fédérale de Lausanne, Volume 4, N°3, (1996).

18 Benjelloun O., Garcia-Molina H., Jonas J., Su Q., Widom J., Swoosh. “A generic approach to entity resolution”, Rapport technique, Stanford University, (2005).

19 Sheth, A. P. et Kashyap, V. So far (schematically) yet so near (semantically). In DS-5, (1992), pages 283-312.

20 Goh, C. H., Bressan, S., Madnick, S. et Siegel, M. "Context interchange: New features and formalisms for the intelligent integration of information". ACM Trans. Inf. Syst., (1999), 17(3):270-293.

21 Rahm E., Bernstein P., "A survey of approaches to automatic schema matching", VLDB Journal, 10(4), (2001) p.334350.

22 Kalfoglou, Y. et Schorlemmer, M. Sheth,A., Staab, S.et Uschold,M. "Ontology mapping : The state of theart. In Semantic Interoperability and Integration", numéro 04391 de Dagstuhl Seminar Proceedings. Germany, (2005).

23 Pinto, H., Prez, A. et Martins, J. "Some issues on ontology integration". Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods, (1999).

24 Baril. X. "Un modèle de vues pour l'intégration de sources de données XML : VIMIX", Thèse de Doctorat en Informatique, de l'Université des sciences et techniques du Languedoc, (Décembre 2003).

25 Rahni Amidha. A. "Une approche médiateur d'intégration de sources de données hétérogènes et autonomes", Mémoire de stage effectué à l'ENSMA, Université de Poitiers, Juillet 2005.

26 Buccella A, Cechich A. "An Ontology Approach to Data Integration", Journal of computer science and Technology, - Citeseer (2003).

27 Pitoura. E. "Extending an object-oriented programming language to support the integration of heterogeneous database systems". Proceedings of the 28th annual Hawaii International Conference on System Sciences, (1995), 2:707-716,.

28 Cherat L, Ezziyyani M, Essaadi M, Bennouna M. "Hybrid integration Approach for Heterogenous Information Sources : Applied to Medical Resources and Patients". Medical Documents, in New Technologies, Mobility and Security, (2008).

29 Bellatreche L., Pierra, G., Nguyen Xuan, D., Hondjack, D. "An Automated Information Integration Technique using an Ontology-based database approach". To appear in: Proceedings of CE'2003, Special track on Data Integration in Engineering, Madeira, Portugal, (Juillet, 2003).

30 Xuan D.N. "Intégration de bases de données hétérogènes par articulation à priori d'ontologies: application aux catalogues de composants industriels". Thèse de Doctorat. Université de Poitiers, (Décembre 2006).

31 Busse S., Kutsche R.-D., Leser U., Weber H, "Federated Information Systems: Concepts, Terminology and Architectures", Technical Report n°99-9, Technical University of Berlin, (1999), p 38.

32 Gardarin G., Sha, F., Ngoc, T.D. "XML-based components for federating multiple heterogeneous data sources", LNCS 1728, (1999), 506–519.

33 Bertini M., Del Bimbo A., Pala P. "Content-Based Indexing and Retrieval of TV News", Pattern Recognition Letters 22, (2001), pp. 503-516.

34 Miniaoui S., "Intégration de données en provenance du web dans un entrepôt de données", Mémoire de DEA, Ecole doctoriale d'Informatique et Information pour l'entreprise, Laboratoire Eric, Groupe de Recherche Base de Données, (Juillet 2001).

35 Gardarin G., Sha. F., Tang Z.-H. "Calibrating the Query Optimizer Cost Model of IRO-DB, an Object-Oriented Federated Database System". In Proceeding of the 22nd. International Conference on Very Large Data Bases (VLDB), pages 378-389, Mumbai (Mombay), Inde, (1996).

36 Vodislav. D . "Intégration de données et XML", Groupe de recherche Bases de Données", DEA SIR (2003), CNAM PARIS.

37 Souka. A, "Génération automatique des requêtes de médiation dans un environnement hétérogène", Thèse de Doctorat en Informatique, de l'université de Versailles Saint-Quentin en Yvelynes, (Septembre 2008).

38 Sais. F. "Integration sémantique de données guidée par une ontologie", Thèse de Doctorat en Informatique, de l'université Paris Sud, (2007).

39 Hammer, J., Mcleod, D. And Si, A. An intelligent system for identifying and integrating non-local objects in federated database systems. In HICSS (3), (1994), pages 398-407.

40 Hull, R. And Zhou, G. "A framework for supporting data inte¬gration using the materialized and virtual approaches", (1996), pages 481-492.

41 Ives Z, Florescu D, Friedman M,Levy A, Weld D. An adaptative query execution engine for data integration. In proceedings of the ACM SIGMOD International Conference on Management of Data, Philadelphia, (June2004), p.299-310.

42 Rousset, M.-C., Adjiman, P., Chatalic, P., Goasdoué, F. et Simon, L. Somewhere in the semantic web. In Wiedermann, J., Tel, G., Pokorný, J., Bieliková,M.et Stuller,J.,éditeurs: SOFSEM 2006 - Proceedings, (2006). pages 84-99. Springer.

43 Nejdl,W.,Wolf,B.,Qu,C.,Decker,S.,Sintek,M.,Naeve,A.,Nilsson,M.,Palmér,M.Risch,Tx). “Edutella: a p2p networking infrastructure based on rdf”. (2003), pages 604-615.

44 Yan, L, L. Miller, R.J, Haas, L.M, Fagin, R. “Data driven understanding and refinement of schema mappings”. In proceedings of the 2001 ACM SIGMOD International conference on Management of data. Santa Barabara, Californiana, United States, (2001), p. 485-496.

45 Milo, T., Abiteboul, S., Amann, B., Benjelloun, O. et Ngoc, F. D. “Exchanging intensional XML data”. Proc. of SIGMOD, (2003), pages 289-300.

46 Bernstein,P.,Giunchiglia,F.,Kementsietsidis,,Mylopoulos,J.,Serafini,L.etZaihrayeu,I. “Data management for peer-to-peer computing:A vision. Workshop on the Web and Databases”, WebDB (2002).

47 Nejdl,W.,Wolf,B.,Staab,S.etTane,J. “Edutella : Searching and annotating resources within an rdf-based p2p network”, (2002).

48 Lenzerini M. “Principles of p2p data integration”, In Zohra Bellahsene and Peter McBrien, editors, DIWeb, (2004).

49 Lenzerini Maurizio, Diego Calvanese , Giuseppe De Giacomo , Riccardo Rosati, “Logical foundations of peer-to-peer data integration”, Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, (June 14-16, 2004), Paris, France.

50 Yao W., Chu C-H., Kumar A., Li Z., “Using ontology to support context awareness in healthcare”, In 19th workshop on Information Technologies and Systems, Phoenix, AZ, USA, (2009).

51 Levy A. Y., A. Rajaraman, et J.J. Ordille, "Querying Heterogeneous Information Sources Using Source Description", In Proceeding of International Conference on Ver, LarCe Data Base, Bombay, (1996).

52 McBrien P., A, Poulovassilis. “Data Integration by Bi-Directional Schema Transformation Rules”. 19th International Conference on Data Engineering, Bangalore, India, (March 2003).

53 Boyd,M., McBrien,P. et Tong,N. "Theautomed schema integration repository”. In Proceedings of BNCOD02, Springer Verlag LNCS, (2002), 2405:42-45

54 Friedman, M., Levy, A. et Millstein, T.”Navigational plans for data integration”. Proc. of the 16th National Conference on Articial Intelligence AAAI, (1999), pages 67-73.

55 Etzioni O., Weld D. “A Softbot-Based Interface to the Internet. Communications of the ACM”, (1994), Vol. 37(7). p. 72-76.

56 Genesereth, M. R., Keller, A. M., Duschka O. “Infomaster: An Information Integration System”. In Peckham J. Eds., Proceedings ACM SIGMOD International Conference on Management of Data . ucson, Arizona, USA, ACM Press, (May 1997), p. 539-542

57 Kirk, T., Levy, A. Y., Sagiv, Y. et Srivastava, D. “The Information Manifold. Information Gathering from Heterogeneous, Distributed Environments” (1995).

58 Subrahmanian, V.S,; Adali, S.; Brink, A.; Emery, R.; Lu, J.; Rajput, A.; Rogers, T.; Ross, R.; and Ward, C. ”HERMES: A heterogeneous reasoning and mediator sys”. Tem. Technical report, Univserity of Maryland (1995).

59 Chawathe S.S., H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, D. Ullman, J. Widom. “The TSIMMIS Project: Integration of Heterogeneous Information Sources”. Proccessings of the 10th Meeting of the Information Processing Society of Japon, (1994), pages : 7-18.

60 Beneventano D. & Bergamaschi S. & Castano S. & Corni A. & Guidetti R. & Malzevezzi G. & Melchiori M. & Vincini M. “Information integration: The MOMIS project demonstration”. In VLDB proceedings of 26th International Conference on Very large Data Bases ». September 10-14. Cairo – Egypte. (2000), p. 611-614.

61 Delobel. C, Rousset. M.C, ”A uniform approach for quering large tree-structured data through a mediated schema”. (2001).

62 Amann, B., Michard. A. “C-Web functional specification (v1.0)”, (2001). http://cweb.inria.fr/Resources/CWebtm.

63 Hondjack Dehainsala H, "Explicitation de la sémantique dans les bases de données : Base de données à base ontologique et le modèle OntoDB", Thèse de Doctorat (PhD) à l’université de Poitiers, France, (2004).

64 Visser , P. R. S., Beer, M. D., Bench-Capon, T. J. M., Diaz, B. M. et Shave, M. J. R. “Resolving ontological heterogeneity in the KRAFT project”. In Database and Expert Systems Applications, (1999), pages 668-677.

65 Levy A. Y., A. Rajaraman, and J. J. Ordille. “The world wide web as a collection of views: Query processing in the information manifold”. Proceedings of the International Workshop on Materialized Views: Techniques and Applications (VIEW’1996), (June 1996), pages 43–55.

66 Alon Y. Halevy, R., Pottinger. M. “A Scalable Algorithm for Answering Queries Using Views”. VLDB Journal (2001).

67 Levy A. & Srivastava D., Kirk T. “Data Model and Query Evaluation in Global Information Systems”. Journal of Intelligent information Systems, (1995). Vol.5. p.121-143

68 Genesereth, M., Fikes, R. “Knowledge Interchange Format (Version 3.0) - Reference Manual”, In Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

69 Florescu D., Raschid L., Valduriez P. “Using Heterogeneous Equivalences for Query Rewriting in Multidatabase Systems”. In Laufmann S., Spaccapietra S., Yokoi T. Eds., Proceedings of the Third International Conference on Cooperative Information Systems (CoopIS- 95), Vienna, Austria, (May 1995). p. 158-169.

70 Florescu D., Raschid L., Valduriez P. “A Methodology for Query Reformulation in CIS Using Semantic Knowledge”. International Journal of Cooperative Information Systems (IJCIS), (1996), Vol. 5, Num. 4, p. 431-468.

71 Cattell, R.G.G. The Object Database Standard : ODMG-93, San Francisco, CA

: Morgan kaufman, (1996), 169p (ISBN 1-558-60302-6)

72 Tork Roth M., Schwarz. P. “Don't Scrap it, Wrap it! A Wrapper Architecture for Legacy Data Sources”. In Proceedings of the 23rd International Conference on Very Large Data Bases, Athens, (1997). p. 266- 275.

73 Carey, M., Haas, L. Towards Heterogeneous Multimedia Information Systems: The Garlic Approach. In Proceedings of the International Workshop on Research Issues in Data Engineering: Distributed Object Management, Taipei, (1995), p. 124-131.

74 Cody, W., Haas, L., Niblack, W., Arya, M., Fagin, R., Flickner, M., Lee, D., Petkovic, D., Schwarz, P., Thomas, J., Tork Roth, M., Williams, J., Wimmers E. “Querying Multimedia Data From Multiple Repositories By Content: The Garlic Project”. In Proceedings of the IFIP Working Conference on Visual Database Systems, Lausanne, Switzerland, (March 1995), p.17-35.

75 Baru, C., Gupta, A., Ludaescher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P., Chu V. “XML-Based Information Mediation with MIX”. In Proceedings of ACM SIGMOD Conference on Management of Data, Philadelphia, Pennsylvania, (June 1999), p. 540-543.

76 Ludäscher E., Papakonstantinou Y., Velikhov P. “A Framework for Navigation-Driven Lazy Mediators“. In Proceedings of the ACM SIGMOD Workshop on The Web and Databases Philadelphia, Pennsylvania, USA. (June 1999), p. 85-90.

77 Rousset, M.C., Safar, B. "Construction de Médiateurs pour Intégrer des Sources d'Information Multiples et Hétérogènes : le Projet PICSEL". In Journal I3 : Information - Interaction - Intelligence, France, (2002), Vol. 2, Num. 1, p. 9-59.

78 Goasdoué, F. "Assistance à la conception de bases de connaissances dédiées au médiateur PICSEL", Mémoire de D.E.A. d’informatique, Université Paris 11, (September 1998), 77 pages.

79 Goasdoue, F., Reynaud, C. Modeling information sources for information integration. In Fensel, D., Studer, R. Editors, “Knowledge Acquisition, Modeling and Management”, Lecture Notes in Artificial Intelligence, Berlin. Springer,

(1999), p. 121-138.

80 Levy, A., Rousset, M.C. “Combining Horn Rules and Description Logics in CARIN”, In Artificial Intelligence Journal, (September 1998), Vol. 14, p. 165-209.

81 Mena, E., Kashyap, V., Sheth, A. Illarramendi, A. “OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies”. In International journal Distributed and Parallel Databases, (Avril 2000), Vol. 8, Num. 2, p. 223-271

82 Mena, E., Kashyap, V., Sheth, A., Illarramendi, A. “OBSERVER : An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies”. In Proceedings of the First IFCIS International Conference on Cooperative Information Systems (CoopIS'96), Brussels, Belgium, (1996). p. 14-25.

83 Mena, E., Kashyap, V., Sheth, A. Illarramendi, A. “Managing Multiple Information Sources through Ontologies : Relationship between Vocabulary Heterogeneity and Loss of Information”. In Proceedings of Knowledge Representation Meets Databases (KRDB'96), ECAI'96 conference, Budapest, Hungary, August 1996, p. 50-52.

84 Goni, A., Mena, E., Illarramendi A. “Querying Heterogeneous and Distributed Data Repositories using Ontologies”. [en ligne] In proceedings of the 7th European-Japanese Conference on Information Modelling and Knowledge Bases(IMKB’97), Toulouse (France), (May 1997). <http://sid.cps.unizar.es/PUBLICATIONS/POSTSCRIPTS/imkb97.ps.gz>

85 Naacke. H, "Modèles de coût pour médiateurs de bases de données" , Thèse de Doctorat (PhD), Université de Versailles-Saint Quentin , Septembre, (1999).

86 Bugnion, E., Devine, S., Govil, K., and Rosenblum, M. Disco: Running Commodity Operating Systems on Scalable Multiprocessors. ACM Transactions on Computer Systems, (November 1997), Vol. 15, Num. 4, p. 412-447.

87 Tomasic, A., Raschid, L., Valduriez, P. “Scaling heterogeneous databases and the design of disco”. In Proceedings of the International Conference on

Distributed Computer Systems, Hong Kong, (1996).

88 Li, C., Yerneni, R., Vassalos, V., Garcia-Molina, H., Papakonstantinou, Y., Ullman, J. D., Valiveti, M. “Capability Based Mediation in TSIMMIS”. In Haas L. M., Tiwary A. Eds, In Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, (June 1998), p. 564-566. (ISBN 0-89791-955-5).

89 Garcia-Molina H., Papakonstantinou Y., Quass D., Rajaraman A., Sagiv Y., Ullman J. D., Vassalos V., Widom J. The TSIMMIS Approach to Mediation: Data Models and Languages . In Journal of Intelligent Information System. (1997), Vol. 8, Num. 2, p. 117-132.

90 Papakonstatniou, Y., Garcia-Molina, H., Widom, J. ” Object exchange across heterogeneous information sources”. In Proceedings of the 11th International Conference on Data Engineering, Taipei, 1995, p. 251-260.

91 Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M. “The MOMIS approach to Information Integration”. IEEE and AAAI International Conference on Enterprise Information Systems, Setubal, Portugal, (Juillet 2001), pages 194-198.

92 Bergamaschi, S., Castano, S., Beneventano, D., Vincini, M. “Semanitc integration of heterogenous information sources”. Journal of Data and Knowledge Engineering, (2001), Vol. 36, Num. 3. p. 215-249

93 Bergamaschi, S., Castano, S., Vincini, M., Beneventano D. “Intelligent Techniques for the Extraction and Integration of Heterogeneous Information”. IJCAI-99 Workshop on Intelligent Information Integration, (31 July 1999), Stockholm.

94 Bergamaschi, S., Castano, S., Vimeracati S.D.C.D., Vincini, M. “An Intelligent Approach to Information Integration”. In Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS-98), Trento, Italy, (1998), p. 253-267.

95 Halevy, A. Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A. And Sikka, V. “Enterprise information integration: suc¬cesses, challenges and controversies. In SIGMOD'05” : Proceedings of the 2005 ACM

SIGMOD international conference on Management of data, (2005), pages 778.

96 Hull, R. And Zhou, G.”A framework for supporting data integration using the materialized and virtual approaches”, (1996), pages 481.

97 Zhu, Y. A framework for warehousing the web contents. In ICSC '99: Proceedings of the 5th International Computer Science Conference on Internet Applications, (1999), pages 83.

98 Mchugh, J., Abiteboul., S., Goldman, R., Quass, D. And Widon, J. Lore: a database management system for semistructured data. SIGMOD Rec., (1997), 26(3):54-66.

99 Zhou, G., Hull, R. And King, R. “Generating data integration mediators that use materialization”. K. Intell. Inf. Syst., (1996), 6(2-3):199-221.

100 Tamine L., Boughanem M., "Un algorithme génétique spécifique à une reformulation muti-requêtes dans un système de recherche d’information", Revue I3 en Sciences et Traitement de l’Information, 1(1), (2001), p. 49-76.

101 Gruber. T. R. '”The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases”. In J. A. Allen, R. Fikes, & E. SAndewall (Eds), Principales of Knowledge Representation and Reasoning: Proceedings of the Second International Conference, Cambridge, MA. San Mateo, Calif: Morgan Kaufmann, (1991), pages 601-602

102 T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5, (1993), 199-220.

103 Bouillon P., Vandooren F., Da Sylva L., Jacqmin L., Lehmann S., Russel G., Viegas E. "Traitement automatique des langues naturelles", Duculot, (1998).

104 Benjamin R., Gomez-Perez, A. (1999): Overview of Knowledge Sharing and Reuse Components: Ontologies and Problem-Solving Methods. Proc. IJCAI-99 Workshop on Ontologies and Problem-Solving Methods (KRR5), Morgan-Kaufmann.

105 Gail, H. “Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files”. CLIR Pub91. (April 2000) http://nkos.slis.kent.edu/KOS_taxonomy.htm

106 Lassila O., Swick R., “Resource Description Framework (RDF)Model and

Syntax Specification", World Wide Web Consortium, (1999).

107 Connolly D., Horrocks I., Mcguinness D., Patel-Schneider F., Stein A., "DAML+OIL Reference Description", World Wide Web Consortium, (2001).

108 Dean M., Schreiber, "WLWeb Ontology Language Reference", *W3C Recommendation* (2004).

109 Pierra G., Potier J. C., Battier G., Sardet E., Derouet J. C., Willmann N., Mahir A., « EXCHANGE OF COMPONENT DATA : THE PLIB (ISO 13584) MODEL, STANDARD AND TOOLS », p. 160-176, (September 98)

110 Bellatreche L., Pierra G., Xuan D., Hondjack D., « Intégration de sources de données autonomes par articulation a priori d'ontologies », Proc. du 23ème congrèsInforsid, p. 283-298, (May 2004).

111 FANKAM C., AIT-AMEUR Y., PIERRA G., "Exploitation of ontology languages for both persistence and reasoning purposes : Mapping PLIB, OWL and Flight ontology models", in Third International Conference on Web Information Systems and Technologies(WEBIST), INSTICC Press, (2007).

112 Borgida. A. "On the relative expressiveness of descripton logics and predicate logics". Artificial Intelligence, volume 82, number 1-2, pages 353-367, (1996)

113 Gaëlle. L. "État de l'art ontologies et intégration/fusion d'ontologies". (2002)

114 Uschold, M., King, M., Moralle, S., Zorgios, Y. "The Entreprise Ontology". Technical Report AIAI-TR-191. Edinburgh: Artificial Intelligence Applications Institute, University of Edinbrgh, (1997).

115 Guarino, N. "Formal ontology and information systems". National Research Council, (1998).

116 Mizoguchi, R.,Vanwelkenhuysen J., Ikeda M., "Task Ontology for Reuse of Proble Solving Knowledge in Towards Very Large Knowledge Bases", N, Mars (editor), pages 46-59, IOS Press.

117 Heikst, G., Schreiber, A., Wielinga B.J. "Using explicit ontologys for KBS developpement". Internationa Journal of Human-Computer Studies (1997).

118 Mihoubi, H. "Une approche déclarative de traduction d'ontologies". Thèse de Doctorat. Grenoble : Université Stendhal – Grenoble1, (2000).

119 Berners-Lee, T., Hendler, J., Lassila. O. “The semantic web. Scientific American”, pages 36–43, (Mai 2001).

120 Pierra, G., Dehainsala, H., Ameur, Y. A., Bellatreche, L., Chochon, J., Mimoune. M. "Base de Données à Base Ontologique : le modèle OntoDB". In Proceeding of Base de Données Avancées 20èmes Journées (bda’04), pages 263–286, (Octobre 2004).

121 Sheth Amit, P., Gala Sunit, K., Navathe Shamkant, B. ”On automatic reasoning for schema integration”. International Journal of Intelligent and Cooperative Information Systems, 2(1) :23-50, (1993).

122 Wache, H., Vogele, T., Visser, U., Stuckenschmidt H., Ontology based integration – A survey of existing approaches. (2001)

123 Visser, U., “Resolving ontological heterogeneity in the KRAFT project”. In Database and Expert Systems Applications, pages 668-677, (1999).

124 Pierra. G. “Context-explication in conceptual ontologies : The PLIB approach”. In Proc. of Concurrent Engineering (CE’2003), pages 243–254, (July 2003).

125 Boussis, A. Intégration de sources de données à base ontologique dans un environnement p2p, Mémoire pour l’obttention de diplôme de magister, Ecole Supérieur en Informatique (ex INI), Algérie, (2007).

126 McGuinness, D.L. and Wright, J.. Conceptual Modeling for Configuration: A Description Logic-based Approach. Artificial Intelligence for Engineering Design, Analysis, and Manufacturing - special issue on Configuration, (1998).

127 McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S.. “An Environment for Merging and Testing Large Ontologies. Principles of Knowledge Representation and Reasoning”. Proceedings of the Seventh International Conference (KR2000). A. G. Cohn, F. Giunchiglia and B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers, (2000).