F.S.D : ..... N° D'ordre : .....

UNIVERSITE SAAD DAHLAB BLIDA

Faculté des Sciences

Département d'Informatique

Mémoire Présenté par :

ALILICHE Hafsa

AMMAM Nawel

En vue d'obtenir le diplôme de MASTER

Domaine : Mathématique et Informatique

Filière : Mathématique et Informatique

Spécialité : Informatique

Option : Ingénierie de Logiciel

Sujet :

Médiation Sémantique des données hétérogènes : Relationnel, Objet-Relationnel & XML

Soutenu publiquement le : 01/07/2012 devant le jury composé de :

M^me **FAREH. M**	Promotrice
M^r **HAMMOUDA.M**	Présidente
M^me **BOUMAHDI .F**	Examinatrice
M^lle **MEZZI.M**	Examinatrice

2011 / 2012

Dédicaces

Qu'il me soit permis de dédier ce travail :

À mes très chers parents, c'est l'occasion pour moi de vous dire à quel point je vous aime.

Jamais à un seul instant votre amour et votre affection ne m'ont fait défaut.

Je vous dédie ce travail pour vous remercier de m'avoir toujours soutenu durant toute ma vie.

A ma sœur Maria, mes frères Anes, Mohamed et Sohaib.

À mes chères grands-mères pour ses encouragements et ses prières.

À ma grande famille.

À mes très chères copines Nabila et Fatima pour votre aide et amour inconditionnel.

À tous mes amis et mes collègues de la promotion, pour tous ce qu'on a partagés ensemble.

À mon amie Nawel et toute sa famille.

À tous qui m'ont soutenu.

Hafsa

Dédicaces

C'est avec un immense plaisir que je dédie ce travail

A mes très chers parents qui sont toute ma vie,

et tout ce que j'ai de plus cher au monde,

en témoignage de ma reconnaissance infinie pour les nombreux sacrifices

qu'ils n'ont cessé de déployer pour moi et dont je serais à jamais redevable.

A ma chère sœur Yasmine, mes frères bien aimés A/Aziz, A/Fettah et Zakaria.

A mon mari Réda qui m'a tellement soutenu et aidé et m'a supporté durant toute cette année d'études.

A mon amie Hafsa et toute sa famille.

En témoignage de ma grande affection pour eux.

Que dieu les garde et leur procure la santé et le bonheur.

A mon regretté mon chère grand-père Ahmed que j'ai souhaité infiniment qu'il soit présent le jour de ma soutenance.

Nawel

Remerciements

En préambule à ce mémoire, nous souhaitons adresser nos remerciements les plus sincères aux personnes qui nous ont apporté leur aide et qui ont contribué à l'élaboration de ce modeste travail.

Nous tenons à remercier sincèrement Madame Fareh qui, en tant que promotrice, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de ce mémoire, ainsi que pour l'inspiration, l'aide et le temps qu'elle a bien voulu nous consacrer et sans qui ce mémoire n'aurait jamais vu le jour.

Nos remerciements s'adressent également à tous nos enseignants du département d'informatique.

Nous n'oublions pas nos parents pour leur contribution, leur soutien et leur patience. Nous tenons à exprimer nos reconnaissances envers Mlle à MEZZI Melyara, Mlle KHELLADI Fatmazora et Mr MOKEDDEM Allal qui nous ont aidés énormément dans ce travail, et à monsieur BAKHTOUCHE Abdelghani qui a corrigé notre mémoire.

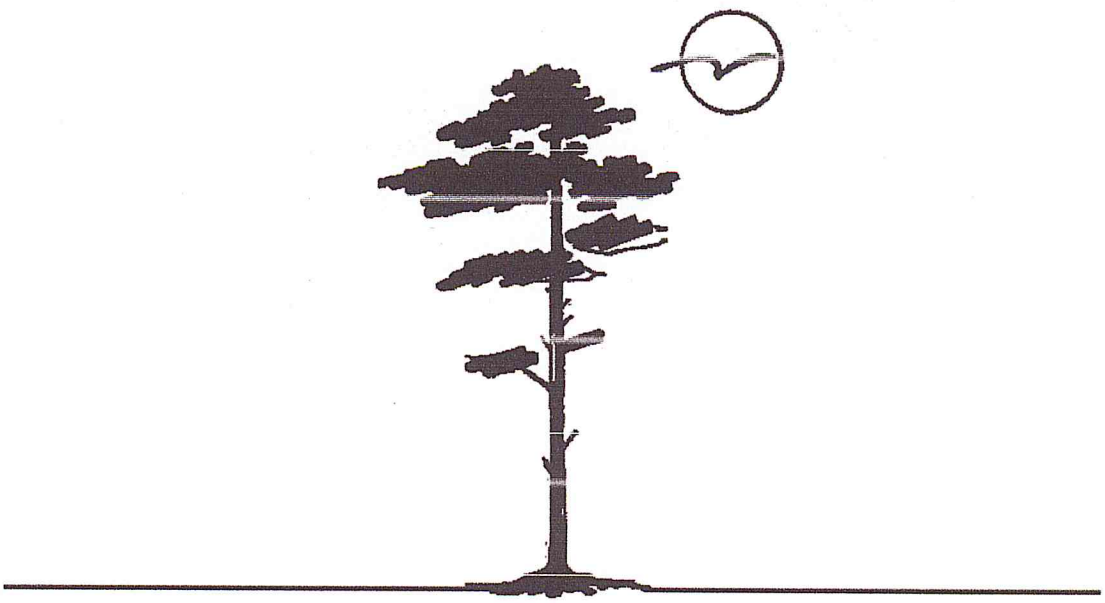
Enfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, et plus particulièrement TOUBAL SEGHEIR Smail qui nous a aidé au cours de la réalisation de ce mémoire.

Merci à tous

« Plus s'étend et s'approfondie le champ de ma connaissance,

Plus s'aiguise la conscience de l'étendu de mon ignorance. »

Michel Beaud



ملخص

يضم نظام المعلومات في المنظمة مجموعة من المعلومات ذات طبيعة متخلفة و منتشرة على مصادر متعددة ، هذا الأخير زاد من صعوبة العثور على معلومات ذات قيمة و التي يحتاجها الفرد في المنظمة بغية اتخاذ القرارات الملائمة، و من أجل تجاوز كل هذا فلابد من توفر نظام يدمج مختلف مصادر المعلومات غير المتجانسة، بغية ضمان الشفافية و التبادل السلس لهذه المعلومات.

ترتكز دراستنا حول معالجة إشكالية تعدد نماذج المعلومات و التي أضفت عليها صفة عدم التجانس، لقد انصب كامل اهتمامنا حول ثلاثة نماذج و هي النموذج العلائقي، النموذج الكائن العلائقي، و نموذج XML.

قمنا بمعالجة هذه الإشكالية باقتراح نظام وسيط قائم على الأنطولوجيا، سيعالج إشكالية عدم التجانس الهيكلي و الدلالي.

استندنا عند تصميم هذا النظام الوسيط على استخدام الانطولوجيا ألهجينة و تسمى بالهجينة نظرا لتعدد نماذج و مصادر هذه المعلومات أين قمنا ببناء انطولوجيا خاصة لكل مصدر من مصادر هذه المعلومات و ذلك بالاعتماد على OWL، و من هنا يتم دمج كل الأنطولوجيات في انطولوجيا عامة معتمدين في ذلك على منهج GLAV.

الهدف من هذه الدراسة هو إعطاء الفرد في المنظمة القدرة على استجواب النظام الوسيط و منه الحصول نتائج متجانسة بطريقة شفافة، مما سيعطي الانطباع بوجود مصدر وحيد للمعلومات و ذلك عن طريق إخفاء عدم التجانس اللغوي و الدلالي لهذه المعلومات.

الكلمات الرئيسية : دمج المعلومات، الانطولوجيا، الوساطة، عدم التجانس الدلالي، OWL, GLAV.

Résumé

Les systèmes d'information des organisations contiennent des informations de diverses natures, dispersées dans une grande variété de sources. Donc, il est devenu difficile de retrouver l'information pertinente désirée par un utilisateur et de l'exploiter de façon transparente. Il faut donc offrir un système de gestion, intégrant des sources de données hétérogènes, tout en assurant la transparence, à la distribution et à l'hétérogénéité.

Nous nous intéressons à la médiation des sources de données hétérogènes représentées par les modèles de données relationnel, objet-relationnel et XML. Le système de médiation que nous proposons est un système à base d'ontologie, qui permet de gérer le problème de l'hétérogénéité structurelle et sémantique, notre architecture se base sur l'utilisation d'une l'ontologie hybride. Cela consiste a créer pour chaque source de données, une ontologie locale représentée en langage OWL, puis de les fusionner dans une ontologie globale en utilisant l'approche GLAV (Generalized Local As View).

Notre objectif est d'offrir à l'utilisateur la possibilité d'effectuer des requêtes, et de recevoir des résultats homogènes de façon transparente, afin de lui donner l'impression d'interroger une seule source en lui cachant l'hétérogénéité syntaxique et sémantique. -

Mots clés : Intégration de données, Ontologie, Médiation, Hétérogénéité sémantique, OWL, GLAV.

Abstract

The organization's information systems contain information of various nature dispersed in a large variety of sources, therefore, it became difficult to find relevant information needed by user and to exploit them in a transparent way. It is thus necessary to offer an integral management system of the heterogeneity data sources while ensuring the transparency in distribution and heterogeneity. We are interested by the mediation of the heterogeneous data sources represented by relational, relational-object and XML models.

The system of mediation that we propose is based on ontology. To manage the problem of structural and semantic heterogeneity, our architecture is based on the use of the hybrid ontology which consists in creating a local ontology for each data source represented in OWL language, then to amalgamate them in a global ontology by using GLAV approach. Our objective is to allow the user to interrogate the system and receive homogeneous and clear results which gives him the impression to ask only one source by hiding syntactic and semantic heterogeneity.

Keywords: Data integration, Ontology, Semantic Heterogeneity, OWL, Mediation, GLAV.

Sommaire

Chapitre II: Les modèles des sources de données

Chapitre III: Conception du système de médiation

Chapitre IV: Implémentation et test

Liste des figures

Liste des tableaux

Liste des acronymes

XML	:	eXtensible Markup Language.
SGBD	:	Système de Gestion des Bases de Données
SI	:	Système d'Information
BD/BDD	:	Base de Données
ODBC	:	Open Data Base Connectivity
JDBC	:	Java Data Base Connectivity
OQL	:	Object Query Language
HTML	:	Hyper Text Markup Language
GAV	:	Global As View
GLAV	:	Generalized Local As View
SQL	:	Structured Query Language
XSL	:	eXtended Style Sheet
DTD	:	Document Type Définition
OWL	:	Ontology Web Language
OID	:	Object Identifier

Introduction générale

Introduction générale

Contexte général

De nos jours, les sources d'informations sont réparties, de plus en plus d'informations sont crées partout dans le monde et publiées sur le Web, de nombreuses entreprises ont des ramifications dans plusieurs pays et les états décentralisent leurs administrations. Elles sont autonomes, car les sources de données sont conçues par différentes personnes, à différents moments et pour répondre à différents besoins applicatifs. Enfin, les sources d'informations sont hétérogènes : des logiciels différents sont utilisés pour créer et gérer les données (Oracle, MySQL, SQL Server, ...), les données sont publiées dans divers formats (HTML, etc.) et des modèles de données différents sont utilises pour leur représentation (Relationnel, Objet, Semi structuré...).

De ce fait, l'accès « transparent » aux ressources et de manière plus générale à l'information constitue un des challenges actuels majeurs de l'informatique. L'hétérogénéité, la quantité, la dispersion des ressources constituent autant de verrous que les systèmes d'intégration doivent lever, ce qui a induit que les systèmes multi-source deviennent de plus en plus développés. Parmi ces systèmes d'informations, nous distinguons les entrepôts de données, les systèmes d'informations basés sur le web, les systèmes de bases de données fédérées, ou encore les systèmes de médiation.

Notre étude s'inscrit principalement dans le contexte des systèmes de médiation. Un système de médiation est un système qui permet d'agir sur un ensemble de sources hétérogènes et distribuées. Ses composants essentiels sont : le schéma global appelé schéma de médiation, les mappings du schéma global avec les schémas des sources, les fonctions de réécriture de requêtes et les fonctions de composition des résultats.

Les systèmes de médiation d'information, doivent être capables de gérer l'hétérogénéité pour fournir des vues intégrées homogènes des informations dispersées. L'hétérogénéité des informations couplée à la distribution des systèmes d'information pose d'importants problèmes de localisation et d'intégration de données. En effet, un utilisateur souhaitant poser une requête sur Internet ne sera pas à même de savoir à quel système l'adresser ni même dans quel langage la formuler. Les systèmes de médiation pallient à ces besoins. Ils collectent les données et uniformisent les langages de requêtes permettant alors à

l'utilisateur de n'interroger que le système de médiation pour obtenir une réponse. Cette dernière est générée à partir de sources d'information rattachées à ce que l'on appelle un médiateur de données via un module de traduction de requêtes et de données appelé adaptateur de source.

Problématique

Nous nous intéressons particulièrement dans notre travail aux systèmes de médiation sémantique qui représentent un défi aussi bien pour les chercheurs que pour les industriels. En effet, des millions de dollars sont dépensés pour des projets d'intégration. Par ailleurs, la médiation 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. L'hétérogénéité sémantique est due aux multiples interprétations que des systèmes autonomes peuvent avoir de la même donnée. Ainsi, la médiation des sources de données hétérogènes, autonomes et réparties passe par la résolution de ces conflits sémantiques et différences syntaxiques.

Pour régler ces conflits sémantiques et syntaxiques, nous repérons dans ce domaine des spécifications formelles des différents concepts que nous appelons les ontologies.

Il existe différentes utilisations possibles des ontologies dans un système d'intégration. Si nous supposons que chaque source contient sa propre ontologie. Le problème d'intégration sémantique devient alors un problème d'intégration d'ontologies.

Objectif

L'objectif de ce travail est de réaliser un système de médiation en intégrant l'aspect sémantique des données. Ce système permet d'accéder à des données hétérogènes à travers une interface uniforme, sans se soucier de leur structure ni de leur localisation.

Le travail vise à concevoir et à réaliser un système de médiation sémantique de données hétérogènes, présentées par les modèles relationnel, objet-Relationnel, et XML Pour permettre à un simple utilisateur de trouver des réponses à ses requêtes en lui cachant l'hétérogénéité syntaxique et surtout sémantique des sources.

Le système de médiation que nous allons mettre en œuvre doit satisfaire les besoins suivants :

(1) Fournir une vue globale intégrée des données représentées à travers différents modèles ;

(2) Identifier et spécifier les correspondances entre des donnés sémantiquement liées ;

(3) Offrir à l'utilisateur une vue uniforme et une interrogation transparente des informations sans qu'il n'ait le souci de la provenance des informations ni de leur format d'origine ;

(4) Cacher l'hétérogénéité sémantique entre les données.

Organisation du mémoire

Notre mémoire est organisé autour de quatre chapitres :

Le 1er Chapitre présente des généralités sur les systèmes d'intégration de données, où nous citerons les définitions, les tâches, les composants et les approches des systèmes d'intégrations, puis nous abordons le sujet des systèmes de médiation que nous détaillerons. Par la suite nous présenterons les principaux travaux proposés sur l'intégration des ontologies.

Dans le 2ème Chapitre nous donnerons un bref aperçu sur les concepts de base des modèles des sources de données avec lesquels nous avons construit notre système de médiation à savoir: Le Relationnel, L'Objet Relationnel et XML.

Le noyau de notre mémoire se trouve dans le 3ème Chapitre qui sera consacré à la description de la solution proposée, après nous présentons la démarche utilisée dans laquelle nous trouverons la spécification des besoins, ensuite l'analyse du système et nous terminons par présenter l'architecture du système.

Dans Le 4ème chapitre, nous citons les différents outils utilisés pour notre application, ensuite nous présenterons une démonstration globale du système par illustration des différentes interfaces, ainsi nous avons consacré une partie de ce chapitre pour tester la validité de notre application.

Enfin, dons la conclusion générale qui conclura notre mémoire. D'une part, nous dressons le bilan de nos contributions et d'autre part, nous présentons nos perspectives de recherche.

Etat de l'art sur les systèmes de médiation

Introduction

Les données sont de plus en plus disséminées sur les réseaux. Le type de ces dernières peut être variés (données textuelles, relationnelles, multimédia, semi–structurées) et leurs systèmes de stockage est très différents (système de fichiers, SGBD, applications). Il faut offrir un système de gestion intégrant des sources de données hétérogènes tout en assurant la transparence à la distribution et à l'hétérogénéité. Dans ce chapitre nous allons nous intéresser à l'approche médiateur des systèmes d'intégration, pour cela différentes approches de médiation ont été mises en œuvre pour assurer la cohérence des données et homogénéiser les différents formats trouvés.

1. Les systèmes d'intégration

Un des problèmes les plus complexes lors de l'intégration de plusieurs sources de données est l'hétérogénéité de ces sources. Les systèmes d'information qui doivent être intégrés peuvent être développés dans des environnements différents, avec des schémas conceptuels et définitions de données différents. Cela conduit à une hétérogénéité à différents niveaux du système comme l'hétérogénéité des données, c'est-à-dire que les données relatives à un même sujet sont représentées différemment sur des systèmes d'information distincts, et elles se différencient selon plusieurs axes de distinction. L'hétérogénéité est indépendante de la distribution physique des données.

1.1. Diversité des sources de données[BAK, 06]

Nous pouvons décrire une source de données par sa localisation, le type de données qu'elle gère, ses possibilités d'interrogation et le format des résultats renvoyés.

1. La localisation d'une source de données englobe tout aussi bien le référencement du site sur lequel se situe la source (URL, adresse IP + port), que le protocole de communication utilisé (TCP/IP: *Transmission Control Protocol /Internet Protocol*), les moyens d'accès à la base (ODBC: *Open Data Base Connectivity*, JDBC: *Java Data Base Connectivity*) ainsi que le support (pages Web, SGBD: *Système de Gestion des Bases de Données*).

2. Le type de données gérées par une source peut être structuré (base de données relationnelles), semi-structuré (sources XML) ou non–structuré (images, multimédia, texte libre).

3. Les possibilités d'interrogation sont aussi nombreuses, et vont des langages de requêtes évolués et standardisés (SQL, OQL) ou propriétaires à de simples interfaces de programmation ou encore des recherches par mots clés (moteur de recherche Web).

4. Les formats des résultats renvoyés complètent les écarts qui peuvent exister entre les différentes sources de données. Celles-ci peuvent être formatées suivant divers standards (XML, HTML).

Les sources de données auxquelles nous avons accès dans un contexte interconnecté via le Web se sont diversifiées selon les directions que nous avons citées, de sorte que nous pouvons à présent parler véritablement de sources de données hétérogènes

1.2. Définition des systèmes d'intégration

Selon le dictionnaire, l'intégration désigne l'action d'intégrer, de son étymologie latine **"integrare"**, intégrer signifie "rendre entier"; faire entrer dans un ensemble, dans un groupe plus vaste.

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 une interface uniforme, sans se soucier de leur structure ni de leur localisation **[MEH et JAO, 10].**

1.3. Composants des systèmes d'intégration

Un système d'intégration se compose de deux parties:

- **Une partie externe** : elle correspond aux utilisateurs du système intégré et autres systèmes.
- **Une partie interne :** elle comprend des sources d'informations et une interface uniforme qui permet à la partie externe d'interroger d'une manière transparente les sources de données, comme s'il n'y avait qu'une source unique **[BOU, 07].**

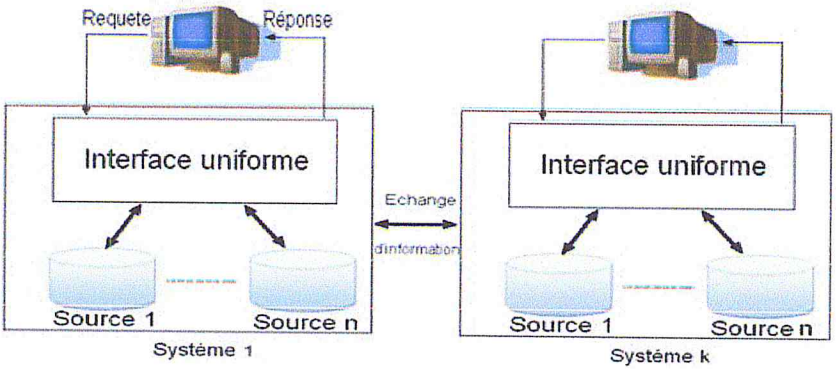


Figure 01: Systèmes d'intégration d'information **[BOU, 07]**

1.4. Différentes classes de conflits :

Comme nous avons cités (section 1.1), les données sont diverses sur plusieurs axes et avec cette diversité nous rencontrons plusieurs conflits lors de la manipulation de ces dernières et dans ce paragraphe nous citons ces différents conflits :

1.4.1. Conflits de modèles de données : ces derniers sont dus à la diversité des modèles utilisés dans la conception des différentes sources (relationnel, objet, relationnel objet, ….).

1.4.2. Conflits de schémas : ces types de conflits regroupent :

a) – Conflits structurels : ils apparaissent lorsque le même élément du monde réel est perçu avec des structures différentes d'une source à une autre.

b) – Conflits de types de données : ils apparaissent lorsque deux sources de données utilisent deux types différents pour le même concept du monde réel.

c) – Conflits d'organisation et de description des données : ils apparaissent sous forme de classes et de relations.

d) – Conflits de généralisation/spécialisation : ces conflits sont basés sur la notion d'une relation généralisée ou spécialisée.

1.4.3. Conflits sémantiques : Les conflits sémantiques sont dus au fait que les données présentés sur plusieurs systèmes admettent des interprétations différentes en fonction du contexte local d'utilisation, ils regroupent :

a) – Conflits de nommage : ils touchent essentiellement des entités et des attributs, et consistent à affecter des noms différents pour le même concept (synonymes), ou affecter le même nom à des concepts différents (homonymes).

b) – Conflits de codification de valeurs : ils se présentent dans la différence d'échelles utilisées, d'unités de mesure, d'intervalles de valeurs affectées aux différents attributs, … etc **[KAD, 06].**

1.5. Tâches d'un système d'intégration

Nous pouvons distinguer quatre tâches 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 tâches 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 :

1.5.1. Transformation de données

Un exemple typique de cette tâche est la transformation de données relationnelles en XML et inversement. Les problèmes importants qui doivent être résolus à 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.

1.5.2. 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.

1.5.3. Réécriture de requêtes:

Cette tâche 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.

1.5.4. 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 **[BOU, 07] .**

1.6. Panorama des approches d'intégration de données

L'hétérogénéité et la distribution constituent les deux problèmes majeurs à résoudre dans l'intégration de données. Bien que les problèmes liés à la distribution des sources soient traités par l'intégration, c'est l'aspect hétérogène des sources de données qui est le plus investi. Garcia-Molina présente les principaux conflits à résoudre dans le cadre de l'intégration de données, à savoir les conflits de noms, de type de données et les conflits sémantiques.

Les solutions et les approches d'intégration sont principalement liées aux différents types d'hétérogénéités existants, à savoir l'hétérogénéité structurelle et l'hétérogénéité sémantique.

1.6.1. L'intégration structurelle

Dans cette approche d'intégration, les différents systèmes possèdent des structures et des modèles différents pour le stockage et la gestion de leurs schémas et données. Afin d'intégrer les sources d'informations, les conflits structurels qui existent entre les descriptions des sources d'information doivent être éliminés **[DJA, 03].**

1.6.2. L'intégration sémantique

Dans cette approche, on prend en considération et le contenu et le sens des données à intégrer. Même si les différentes sources utilisent la même structure, l'interprétation sémantique de chaque élément peut être différente. L'intérêt principal de cette approche est de résoudre les conflits sémantiques et établir un mapping entre les éléments hétérogènes **[DJA, 03].**

1.7. Evolution des systèmes d'intégration

Il existe plusieurs systèmes d'intégration tel que chaque type de ces systèmes a ses propres caractéristiques, si nous reviendrons à l'historique des systèmes d'intégration nous trouvons **les systèmes multi-bases** qui sont des systèmes dits faiblement couplés. 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 multibase de type SQL notamment **les systèmes fédérées,** ces derniers sont dits fortement couplés. Ils se caractérisent par l'existence d'un schéma unifié appelé schéma fédéré qui constitue l'interface d'accès au système intégré. L'intégration se situe au niveau des schémas, **les entrepôts de données** dans cette apporche, les données provenant des sources à intégrer sont stockées sur un support spécifique et aussi **les systèmes de médiation** qui constitue, aujourd'hui sans doute, la solution la plus courante pour relier différentes sources qui ne correspondent pas nécessairement à des bases de données. Dans les sections qui suivent nous nous intéresserons aux systèmes de médiation.

2. Les systèmes de médiation

L'intégration virtuelle des sources de données hétérogènes, autonomes et reparties est une solution pour interagir entre différents systèmes d'information, puisqu'elle simplifie l'accès aux données. Elle est fondée sur la médiation et plus particulièrement sur le couple médiateur-adaptateur. Le médiateur a pour rôle de masquer l'hétérogénéité et la répartition des sources de données. Quant a l'adaptateur, il a pour fonction d'adapter les requêtes aux formats des sources de données.

L'approche médiateur consiste à définir une interface entre l'agent (humain ou logiciel) qui pose une requête et l'ensemble des sources accessibles via le Web potentiellement pertinentes pour répondre.

L'objectif est de donner l'impression d'interroger un système centralisé et homogène alors que les sources interrogées sont réparties, autonomes et hétérogènes.

2.1. Définition des systèmes de médiation

Un système de médiation est un système qui permet d'agir sur un ensemble de sources hétérogènes et distribuées. Ses composants essentiels sont : le schéma global appelé schéma de médiation, les mappings du schéma global avec les requêtes et les fonctions de composition des résultats. Les mappings du schéma global avec les sources sont des requêtes, appelées requêtes de médiation.

Plusieurs problèmes de conception émergent lors de l'utilisation de ces médiateurs.
L'une des principales difficultés rencontrées dans un système de médiation est la définition du schéma global et la définition des mappings (requêtes de médiation) qui relient le schéma global aux sources de données **[MEH et JAO, 10].**

2.2. Architecture de médiation

Une architecture de médiation se compose de trois niveaux (sources, médiateur et clients).

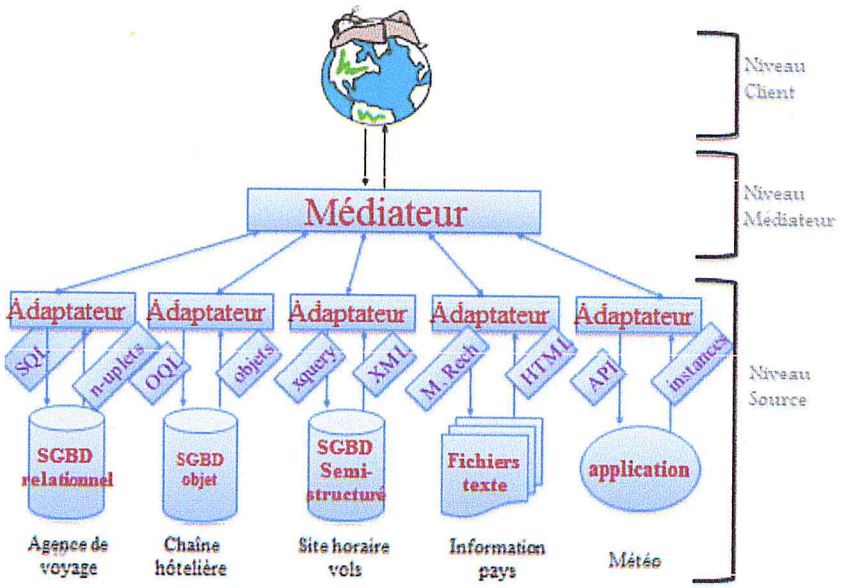


Figure 2: Architecture de médiation.

1. **Le niveau sources:** comporte les différentes sources de données, à l'aide d'un adaptateur, il est capable de communiquer avec les médiateurs.

2. **Le niveau médiateur:** comporte des médiateurs permettant d'intégrer les sources .

3. **Le niveau clients:** comporte les applications clientes (navigateurs, programmes d'application, interfaces graphique).

2.3. Un system de médiation est composé de :

2.3.1. Médiateur (Mediator)

Le médiateur est une solution qui offre une interface unique et transparente à plusieurs sources de données, il résout avec efficacité le problème de distribution des sources.

Rôle et fonctions :

_ Localisation des sources de données ;

_ Décomposition des requêtes pour les adapter à chacune des sources de données ;

_ Envoi des requêtes aux différentes sources ;

_ Recomposition des différents résultats provenant de chacune des sources ;

_ Interprétation des correspondances et des conflits.

2.3.2. Adaptateur (Wrapper)

L'adaptateur est un ensemble de règles de transformation qui convertit les données et les requêtes d'un modèle vers un autre modèle, il s'occupe donc de l'hétérogénéité des sources en offrant une interface homogène locale d'accès aux données.

Rôle et fonctions :

_ Traduction des requêtes du langage commun en langage natif propre à la source ;

_ Traduction des résultats du langage natif en langage commun.

Communication médiateur/adaptateur :

Pour faciliter le travail d'intégration, il existe **[KAD, 06]** :

– Un langage commun dans lequel le médiateur interrogera les adaptateurs;

– Un format de résultat commun dans lequel les adaptateurs répondront au médiateur

2.4. Mapping de données / Nature du mapping

La méthode la plus ancienne pour définir un schéma intégré est la correspondance schéma global/schémas locaux, elle consiste à utiliser le concept classique de "vue SQL" existant dans les bases de données. GaV (Global-as-View), LaV (Local-as-View), GLaV (Generalized -Local-as-View) sont les méthodes de mapping connues, que nous allons présenter dans les paragraphes suivants. GaV et LaV en sont les principales approches, GLaV est une approche mixte **[BOU, 07].**

2.4.1. Approche globale (GAV : Global As View)

Cette approche suppose que les sources à intégrer soient connues à l'avance. Le point faible de cette approche est qu'elle ne permet pas l'ajout de nouvelles sources de données. Cet ajout peut entraîner une modification du schéma du médiateur car les vues qui le définissent seront modifiées **[BOU et deb, 07].**

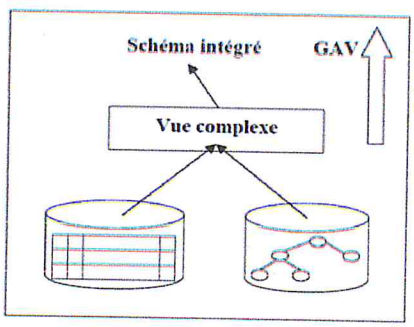


Figure 3 : L'architecture GAV **[KAD, 06]**

Les avantages :

1– La construction des réponses à des requêtes dans un système adoptant l'approche GAV est simple, il suffit de remplacer les prédicats du schéma global de la requête par leur définitions ;

2– Lorsqu'une source de données se change, ou une nouvelle source est ajoutée et doit participer au système, il suffit de mettre à jour le schéma GAV et la phase de réécriture dans ce cas est très simple.

L'inconvénient : L'ajout d'une nouvelle source, entraîne la modification du schéma global.

2.4.2 – Approche locale LAV (Local As View)

A l'inverse de l'approche GAV, l'approche LAV suppose que le schéma global est défini indépendamment des sources. Son principe consiste à définir les sources à intégrer comme des vues sur le schéma du médiateur.

L'approche LAV est flexible par rapport à l'ajout (ou la suppression) de sources de données à intégrer, cela n'a aucune incidence sur le schéma global, il suffit pour cela de spécifier les vues qui permettent d'intégrer (supprimer) la nouvelle source **[BOU et deb, 07].**

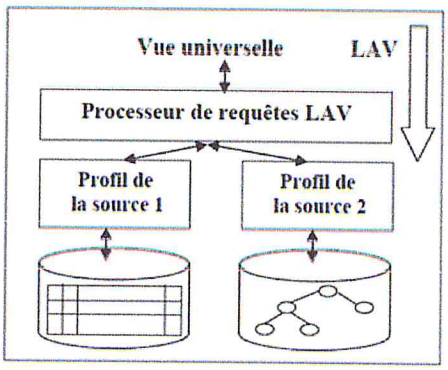


Figure 4 : L'architecture LAV **[KAD, 06]**

Les avantages :

1-L'ajout d'une source de données est facile, cela n'a aucun effet sur le schéma global ;

2-Il est aisé d'attacher des contraintes assez riches relatives au contenu des différentes sources (pouvoir d'expression).

L'inconvénient :

1-Difficulté de réconcilier source et vue locale ;

2-La construction des réponses à des requêtes (réécriture des requêtes) est complexe.

2.4.3 - L'approche mixte (GLAV : Generalized Local as View)

Les avantages des approches LaV et GaV ont été combinés dans une approche mixte. C'est ainsi qu'a été proposée l'approche GLaV (Generalized-Local-as-View), 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) **[BOU, 07].**

2.5. Le Traitement des requêtes

Le traitement des requêtes suppose le passage du schéma global aux schémas de sources locales, autrement dit :

- La transformation des requêtes exprimées à partir du schéma global vers des requêtes destinées aux sources locales,
- La transformation et le regroupement des résultats retournés par les sources locales en vue de les présenter à l'aide du schéma global.

Le traitement des requêtes est organisé en cinq étapes :

1. L'analyse sémantique et structurelle de la requête.
2. La réécriture de la requête : cette étape consiste à décomposer la requête en une ou plusieurs sous-requêtes sur les schémas des sources locales et une requête de recomposition. La réécriture est réalisée grâce aux assertions établies dans le schéma global pour définir les correspondances entre les schémas des sources locales et le schéma global.
3. L'optimisation : cette étape a pour but d'optimiser le traitement de la requête par l'affectation des sous-requêtes aux différentes sources locales en tenant compte de leurs capacités d'exécution.
4. La planification : l'objectif de cette étape est de définir l'ordre d'exécution des sous-requêtes afin d'obtenir les résultats attendus par le client, certaines sous-requêtes pouvant dépendre du résultat d'autres sous-requêtes.
5. L'exécution : les sous-requêtes sont envoyées aux sources locales suivant le plan d'exécution déterminé dans l'étape précédente. Le médiateur recompose les résultats à l'aide de la requête de recomposition déterminée dans la seconde étape, avant de retourner au client un résultat global **[BAR, 05].**

2.6. Discussion / Travaux réalisés

Les différents systèmes d'intégration d'informations à base de médiateurs se distinguent par : d'une part, la façon dont est établie la correspondance entre le schéma global et les schémas des sources de données à intégrer, d'autre part, les langages utilisés pour modéliser le schéma global, les schémas des sources de données à intégrer et les requêtes des utilisateurs.

Concernant le premier point, on distingue l'approche Global As Views (GAV) de l'approche Local As Views (LAV). L'approche GAV comme définit dans la section 2.4.1 a été utilisée par les systèmes suivants : HERMES, TSIMMIS, MOMIS.

L'approche LAV elle est adoptée dans les systèmes suivants : Razor, Internet Softbot, Infomaster, Information Manifold, OBSERVER, PICSEL. Les avantages et inconvénients de ces deux approches sont inverses.

Les systèmes existants se différencient également par le langage qu'ils utilisent pour exprimer le schéma global. On distingue les systèmes fondés sur un schéma global à base de règles (Razor, Internet Softbot,Infomaster, Information Manifold, HERMES), des systèmes fondés sur un schéma à base de classes (langage orienté objet (TSIMMIS)), logique de description (SIMS, OBSERVER, MOMIS), ou encore des systèmes combinant le pouvoir d'expression d'un formalisme à base de règles et d'un formalisme à base de classes (PICSEL). Enfin, plus récemment, ont apparus des médiateurs au-dessus de données semi-structurées ayant le format des documents XML .Ces systèmes sont fondés sur un schéma global à base d'arbres. Ils relèvent à la fois de l'approche GAV et LAV, la correspondance entre le vocabulaire du médiateur et celui des sources étant exprimée par de simples mappings de chemins **[HAC et rey, 02] .**

2.7. Classification des solutions de médiation

La plupart des auteurs distinguent deux catégories fondamentales de médiation; la médiation de schémas qui est une extension directe de l'approche fédérée, et la médiation de contextes reposant sur la distance sémantique et l'unification de contextes **[S12]** :

2.7. 1. La médiation de schémas

La médiation de schémas est essentiellement une évolution de l'architecture fédérée fortement couplée. Le rapprochement à effectuer entre un traducteur et un adaptateur et entre un intégrateur et un médiateur est que les premiers sont directement les ancêtres des seconds. En fait, on peut considérer que les chercheurs travaillant sur l'approche fédérée fortement couplée ont décidé d'adapter leur vocabulaire à celui de Wiederhold. Il existe dans cette approche, comme dans l'approche fédérée fortement couplée, un schéma conceptuel global (contexte global ou schéma de médiation), auquel doivent s'apparier les différents schémas locaux (contextes locaux) : il y a donc intégration des schémas locaux au schéma global. De plus, comme dans l'approche de fédération fortement couplée, la construction du schéma global repose en général sur l'analyse préalable des schémas locaux et ceux-ci sont intégrés d'une manière statique au schéma global. Nous remarquerons également que les différentes phases de pré-intégration, recherche des correspondances, intégration et restructuration doivent être réalisées comme dans l'approche fédérée.

Ces systèmes se distinguent par contre d'une approche par fédération fortement couplée en résolvant certains problèmes comme l'indisponibilité des sources et la combinaison des informations d'une manière plus flexible; en offrant un ensemble d'outils de haut niveau permettant de combiner et de restructurer les informations dans le schéma de la médiation. De plus, comme pour tout système de médiation.

2.7. 2. La médiation de contextes

Ce type de médiation est adapté à des environnements ouverts où les sources d'information sont susceptibles d'évoluer, d'apparaître et de disparaître. Cette approche est également caractérisée par une prise en charge automatisée de la sémantique grâce aux mécanismes d'unification ou de réconciliation des contextes. La distinction fondamentale entre une médiation de schémas et une médiation de contextes est que, dans la médiation de contextes, aucune information d'intégration statique n'est nécessaire, les liens entre adaptateurs et médiateurs sont établis dynamiquement lors de la résolution d'une requête.

2.8. Avantages des systèmes de médiation

L'approche médiateur présente les avantages suivant **[DAN, 08]** :

• **Accès intégré** : Aider à accéder ces sources en évitant à l'utilisateur d'interroger lui-même chacune d'elles selon leurs propres modalités et leurs propres vocabulaires.

• **Transparence** à la localisation des données pour les applications : Découvrir les sources pertinentes hétérogènes et transparentes à la localisation de leurs données pour l'intérêt des applications.

• **Disponibilité** accrue des données en cas de pannes des serveurs.

• Support de l'**hétérogénéité** des sources.

•Offrir à l'utilisateur une vue globale uniforme et centralisée des données .

3. Les ontologies

Nous introduisons dans cette partie le concept d'**ontologie** et plus précisément le rôle de cette dernière dans l'intégration sémantique des données. Nous commençons par décrire les raisons pour lesquelles les ontologies ont été introduites dans les systèmes de médiation puis nous présenterons diverses définitions proposées à leur sujet. Aussi nous découvrirons les éléments qui constituent une ontologie. Enfin, nous mettons l'accent sur les trois approches d'Ontologies dans l'intégration.

3.1. Définition d'un ontologie

Le terme ontologie est d'origine grecque, il a vu le jour dans le domaine de la philosophie, où il signifie : **explication systématique de l'existence**, l'étude de ce qui existe dans le monde .Par la suite le terme a été utilisé en informatique et en science de l'information où une ontologie est une représentation des connaissances du monde au niveau conceptuel qui par la suite sera employée pour permettre le raisonnement automatique sur les objets du domaine concerné **[DAO et CHI, 09]**.

Une ontologie est une représentation des connaissances d'un domaine au niveau conceptuel, elle regroupe les concepts représentatifs d'un domaine donné, leurs relations et aussi la sémantique des concepts et des relations. C'est une représentation non ambiguë et formelle, manipulable par des systèmes informatiques et indépendante d'une application particulière afin de permettre sa portabilité d'une application à une autre dans le contexte du domaine modélisé : elle est réutilisable et partageable **[DAO et CHI, 09].**

3.2. Utilisation des ontologies pour l'intégration

L'ontologie comme représentation consensuelle et explicite d'une conceptualisation permet de fournir un vocabulaire partagé pour les différentes sources. Elle permet et facilite la communication de connaissances. D'un point de vue général, Ushold et Gruninger divisent l'espace d'utilisation des ontologies en trois parties :

1. La communication entre personnes ayant des points de vue et des besoins différents ;

2. L'interopérabilité entre utilisateurs qui ont besoin de s'échanger des données et qui emploient des outils différents ;

3. L'ingénierie des systèmes, où la capacité des ontologies à faire partager et réutiliser des connaissances est exploitée dans la construction et l'utilisation des systèmes à base de connaissances.

L'utilité des ontologies dans le domaine de l'intégration de données peut se retrouver dans les deux premières catégories identifiées par Ushold et Gruninger. 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. Elles peuvent aussi servir de modèle d'interrogation pour le système intégré lorsqu'elles sont utilisées pour décrire le schéma global **[DIA, 06].**

Classiquement, nous distinguons trois architectures dont les ontologies sont utilisées au sein d'une infrastructure d'intégration :

3.2.1. 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 **[DIA, 06] .**

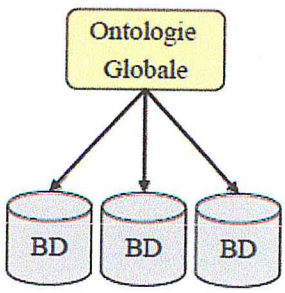


Figure 5 : Architecture utilisant une ontologie unique **[IZZ, 06]**

3.2.2. 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. Ces correspondances peuvent se révéler parfois très complexes, notamment à cause de niveaux de granularité différents entre les ontologies **[DIA, 06].**

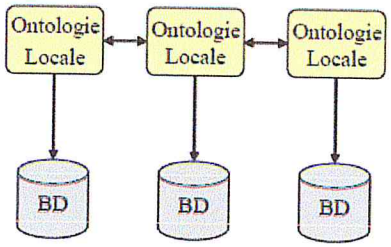


Figure 6 : Architecture utilisant plusieurs ontologies **[IZZ, 06]**

3.2.3. 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 **[DIA, 06].**

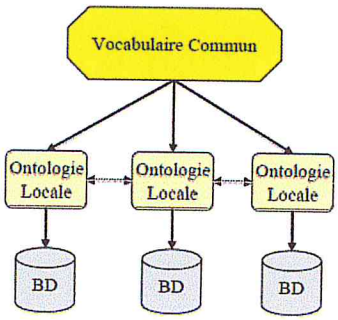


Figure 7 : Architecture utilisant une approche hybride **[IZZ, 06]**

3.2.4. Discussion

Nous avons brièvement reproduit dans le tableau suivant les principales caractéristiques de chacune de ces approches. Il ressort de l'étude de ce tableau que l'approche hybride peut constituer une approche pertinente pour l'intégration des systèmes d'information industriels dans la mesure où elle peut fournir à la fois de la flexibilité et une simplicité de mise en œuvre **[IZZ, 06].**

	Approche mono-ontologie	Approche multi-ontologies	Approche hybride
Effort d'implémentation	Simple	Elevé	Raisonnable
Hétérogénéité sémantique	Sémantique unifiée d'un domaine	Hétérogénéité sémantique (plusieurs domaines)	Hétérogénéité sémantique (plusieurs domaines)
Impact lors de L'ajout/mise à jour d'applications	Besoin d'adaptation de l'ontologie globale	Ajouter une autre ontologie locale, mettre à jour l'ontologie locale associée à l'application mise à jour. Liaison avec les autres Ontologies locales.	Ajouter une autre ontologie locale, mettre à jour l'ontologie locale associée à l'application mise à jour.
Comparaison d'ontologies	Pas de comparaison d'ontologies (car il n'existe qu'une seule ontologie)	Difficile du fait qu'il n'existe pas d'ontologie partagée	Simple du fait qu'il existe une ontologie partagée

Tableau 1 : Comparaison des approches ontologiques **[IZZ, 06]**

Du fait qu'elles comportent plus d'une ontologie, les approches à ontologies multiples et les approches hybrides peuvent être confrontées au problème d'hétérogénéité. Dans ce cas, il devient nécessaire d'intégrer et de faire inter opérer des ontologies.

3.3. Les systèmes d'intégration réalisée à base d'ontologie

De nombreux travaux de recherche ont été menés sur les ontologies, dans le domaine de l'ingénierie des connaissances d'une part, et sur l'intégration de l'information, dans le domaine des bases de données d'autre part. Dans ce dernier contexte, ce sont essentiellement les problèmes liés à l'hétérogénéité syntaxique qui ont été résolus. C'est seulement depuis quelques années que l'hétérogénéité sémantique est traitée dans les systèmes d'intégration par l'utilisation d'ontologies. Ceci a donné lieu à divers systèmes tels que SIMS, MOMIS, KRAFT, XYLEME, PICSEL, OBSERVER et PIAZZA. Pour montrer la diversité des systèmes d'intégration, nous en présentons quelques uns en mettant l'accent sur le modèle de données utilisé et le formalisme sous-jacent ainsi que sur l'architecture d'ontologie adoptée et le type de mapping (GAV, LAV).

SIMS est un système réalisé par Arens et Knoblock en 1996 qui vise l'intégration de sources de données hétérogènes et de bases de connaissances qu'il représente en utilisant le langage LOOM, basé sur la logique de description. SIMS adopte l'architecture à ontologie unique et utilise le mapping GAV. Le médiateur dans SIMS est spécialisé dans un seul domaine d'application.

MOMIS est un système d'intégration réalisé par Beneventano et Bergamashi en 2004 qui se base sur son propre langage orienté objet dénommé ODL. Il utilise une ontologie unique globale appelée GVV (Global Virtual View) qui est générée semi-automatiquement. MOMIS adopte l'approche GAV pour le mapping entre l'ontologie globale et les sources locales.

OBSERVER est un système qui permet l'interopérabilité entre différentes sources, il a été réalisé Mena et Illarramendi en 1996, en utilisant pour cela de multiples ontologies pour décrire les sources de données. Il se base sur CLASSIC, un langage de logique de description. Il n'y a pas d'ontologie globale dans OBSERVER ; le mapping entre les multiples ontologies est réalisé à l'aide de tables de correspondance. Cependant, les relations entre ontologies sont limitées à des relations lexicales basiques telles que les synonymes, hyponymes et hyperonymes.

PICSEL est un système qui a connu plusieurs versions, il a été réalisé par Reynaud et Giraldo en 2003. Il vise à construire un médiateur à base de connaissances. Le langage utilisé est ALN-CARIN, un formalisme à base de règles et de logique de description. Dans sa dernière version, PICSEL adopte l'approche hybride comme architecture d'ontologie et utilise l'approche LAV pour le mapping entre l'ontologie globale et les ontologies locales. Ce mapping est généré semi automatiquement.

Quant à XYLEME, c'est un système d'intégration physique (approche entrepôt) avec XML comme modèle de données et il est réalisé par Delobel et Reynaud en 2003, Il adopte l'approche hybride dans son architecture. Les ontologies globales et locales sont exprimées à l'aide d'arbres, avec un mapping GAV/LAV. Le mapping est réalisé semi automatiquement par la génération des tables de correspondance entre le chemin de l'ontologie globale et les chemins des ontologies locales.

Enfin, 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 a été réalisé par Halevy et Ives en 2003. 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.

La plupart de ces systèmes utilisent une architecture à ontologie unique. Cette approche est intéressante dans le cas où 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 côté, une source ne peut être changée sans reconsidérer 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 **[S03].**

Conclusion

Nous avons étudié dans ce chapitre les systèmes d'intégration ainsi que les systèmes de médiation, comme nous avons introduit la notion des ontologies dans ces derniers, afin de concevoir un système suivant cette logique. Dans le chapitre suivant nous allons présenter les modèles de données, avec lesquels nous testerons notre système.

Les modèles des sources de données :

- Relationnel
- Objet-Relationnel
- XML

Introduction

La diversité des sources de données distribuées et leur hétérogénéité sont une des principales difficultés rencontrées par les utilisateurs aujourd'hui. Cette hétérogénéité peut provenir du format ou de la structure des sources (sources structurées : bases de données relationnelles, sources semi-structurées : documents XML, ou non structurées : textes), du mode d'accès et de requêtes ou de l'hétérogénéité sémantique.

Dans ce chapitre nous allons tout d'abord présenter les concepts fondamentaux des sources de données commençant par un petit aperçu sur l'historique de ces dernières. Nous exposerons ensuite les modèles de données que nous allons détailler à savoir : le Relationnel, l'Objet-Relationnel et enfin le XML.

1. Généralités sur les sources de données

En informatique, il existe des sources de données de tailles importante, et qui offrent un accès multiutilisateurs, et pour pouvoir manipuler ces sources, il existe un langage de bases de données qui est un langage de haut niveau, il est constitué par le langage de requêtes qui est utilisé pour spécifier les données recherchées.

Historique des sources des données

L'évolution des sources de données a suivi le cycle de développement suivant :

- Jusqu'aux années 60 : organisation classique en fichiers ;
- Fin des années 60 : apparition des premiers SGBD (*Systèmes de Gestion de Bases de Données*), les systèmes réseaux et hiérarchiques ;
- À partir de 1970 : deuxième génération de SGBD, les systèmes relationnels ;
- Début des années 80 : troisième génération de SGBD, les systèmes orientés objet ;
- Actuellement : 4^{e} génération, les données du web XML**[GAR, 02]**.

2. Les sources de données traitées

2.1. Le modèle relationnel

Le modèle relationnel a été introduit par E.F.Codd.70 , qui travaillait au centre de recherche d'IBM San-José et s'opposait au développement du modèle DIAM, un modèle utilisant des entités liées par de multiples pointeurs. La première volonté du modèle relationnel fut d'être un modèle ensembliste simple, supportant des ensembles d'enregistrement aussi bien au niveau de la description que de la manipulation. Les premières idées d'un modèle ensembliste avaient été proposées un peu avant, notamment dans (Childs68). Le modèle relationnel est aujourd'hui la base de nombreux systèmes, et les architectures permettent d'accéder depuis une station de travail à des serveurs de données qui s'appuient en général sur lui. Le relationnel a donc atteint ses objectifs au-delà de toute espérance **[GAR, 02].**

2.1.1. Définition du modèle relationnel

Dans ce modèle, les données sont représentées par des tables, sans préjuger de la façon dont les informations sont stockées dans la machine. Les tables constituent donc la structure logique du modèle relationnel, au niveau physique, le système est libre d'utiliser n'importe quelle technique de stockage (fichiers séquentiels, indexage, adressage dispersé, séries de pointeurs, compression, ...) dès lors qu'il est possible de relier ces structures à des tables au niveau logique. Les tables ne représentent donc qu'une abstraction de l'enregistrement physique des données en mémoire.

Les succès du modèle relationnel auprès des chercheurs, concepteurs et utilisateurs est dû à la puissance et à la simplicité de ses concepts. En outre, contrairement à certains autres modèles, il repose sur des bases théoriques solides, notamment la théorie des ensembles et la logique des prédicats du premier ordre **[GAR, 02].**

2.1.2. Objectifs du modèle relationnel

Les premiers objectifs du modèle ont été formulés par E.F.Codd comme suit :

1. Permettre un haut degré d'indépendance des programmes d'applications et des activités interactives à la représentation interne des données, en particulier aux choix des ordres d'implantation des données dans les fichiers, des index et plus généralement des chemins d'accès ;
2. Fournir une base solide pour traiter les problèmes de cohérence et de redondance des données ;
3. Permettre le développement des langages de manipulation de données non procéduraux basés sur des théories solides ;
4. Etre un modèle extensible permettant de modéliser et de manipuler simplement des données tabulaires, mais pouvant être étendu pour modéliser et manipuler des données complexes ;
5. Devenir un standard pour la description et la manipulation des bases de données **[GAR, 02].**

2.1.3. Les structures de données de base

Le modèle relationnel est fondé sur la théorie mathématique bien connue des relations. Cette théorie se construit à partir de la théorie des ensembles. Trois notions sont importantes pour introduire les bases de données relationnelles **[GAR, 02].**

A. *Domaine* : Ensemble de valeurs caractérisé par un nom ;

B. *Relation* : Sous-ensemble du produit cartésien d'une liste de domaines caractérisé par un nom ;

C. *Attribut* : Colonne d'une relation caractérisée par un nom. Ces trois concepts de base du modèle relationnel représentent un n-uplet qui est une ligne d'une relation correspondant à un enregistrement.

2.1.4. Extension et intention

Comme tous les modèles de données, le modèle relationnel permet de décrire des données dont les valeurs varient en fonction du temps. Les relations varient en fonction du temps en ce sens que des tuples sont ajoutés, supprimés et modifiés dans une relation au cours de sa vie. Cependant, la structure d'une relation caractérisée par les trois concepts de domaine, relation et attribut est un invariant pour la relation (elle ne change pas en fonction du temps). Cette structure est capturée dans le schéma de la relation **[GAR, 02].**

2.1.5. Le langage de requête SQL

SQL (Structured Query Language) est le langage des bases de données relationnelles répondant à la fois à la problématique de création des objets de bases de données (modèle), de manipulation des données (algèbre relationnelle), de gestion de la sécurité (« droits d'accès »),et de traitements locaux de données (procédures). De plus, il est désormais doté d'extensions objet. Mais, avant tout, SQL est une norme.

2.2. Le modèle objet-relationnel

Le modèle objet-relationnel (OR) reprend le modèle relationnel on ajoutant au relationnel un certain nombre des concepts de l'approche objet.

2.2.1. Pourquoi intégrer l'objet au relationnel ?

L'intérêt du modèle Objet-Relationnel est de conserver les avantages du modèle relationnel et de limiter ses faiblesses.

1. Les Forces du modèle Relationnel

Le relationnel s'est imposé dans l'industrie au cours des années 1980 à cause de ses points forts **[S07]:**

- Repose sur un modèle théorique et des principes simples ;
- SGBD fiables et performants très présents (applications de gestion) ;
- Langage d'interrogation de type d'assertion (SQL) ;
- Adaptation aux architectures client-serveur ;
- Théorie pour la conception des bases (normalisation) ;
- Optimisation de requêtes (algèbre, réécriture, modèle de coûts) ;
- Gestion de transactions (concurrence, fiabilité).

2. Les Faiblesses du modèle Relationnel

Le relationnel présente cependant des faiblesses qui justifient son extension vers l'objet, on cite les inconvénients suivants **[S07].**

- Règles de modélisation trop restrictives pour structurer les données ;
- Fragmentation d'un objet en plusieurs n-uplets relationnels ;
- Non support de domaines composés : Insuffisance d'objets longs et non structurés ;

- Non intégration d'opérations : Impossible de modéliser le comportement des objets et de spécifier les attributs privés accessibles seulement à travers des opérations.

3. Difficultés de passage du relationnel à l'objet

L'idée de passage du modèle relationnel à l'objet rencontre les difficultés suivantes :

- Identité des objets ;
- Traduction des associations ;
- Traduction de l'héritage ;
- Navigation entre les objets ;
- Les objets persistants doivent être enregistrés dans la base relationnelle ;
- Un objet a une structure complexe : représentée par un graphe **[S04].**

2.2.2. Définition du modèle objet-relationnel

Le modèle objet-relationnel est fondé sur l'idée d'extension du modèle relationnel avec les concepts essentiels de l'objet (Figure 08) .Le cœur du modèle reste donc conforme au relationnel, mais on l'ajoute les concepts clés de l'objet sous une forme particulière pour faciliter l'intégration des deux modèles **[GAR, 02].**

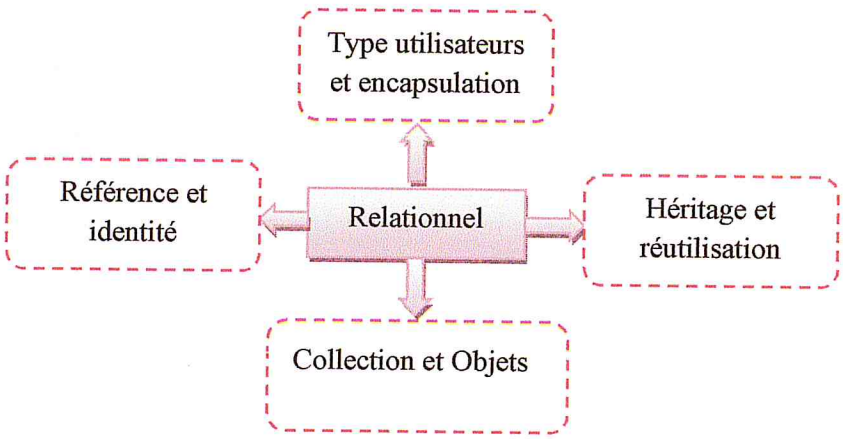


Figure 8 : Les extensions apportées au relationnel **[GAR, 02].**

Le modèle objet-relationnel est une tentative de réunion des concepts présents dans les modèles relationnel et objet. Cette réunion est réalisée en étendant le modèle relationnel pour lui conférer un certain nombre de qualités reconnues du modèle objet. La totalité des fonctions d'un SGBDR classique est préservée et les concepts qui font le succès de l'approche objet ainsi que de nouveaux types de données y sont intégrés. Elle est apparue en 1992 avec les SGBD UNISQL et Open ODB d'Hewlett-Packard (appelé par la suite Odapter) **[S05].**

Le succès de ce modèle provient de **[BRO et bru, 10]**:

• L'encapsulation des données des tables. Les méthodes définies sur les types composant les tables permettent de programmer explicitement l'encapsulation (il faudra, simultanément, que le programmeur interdise les accès directes aux objets par SQL).

• La préservation des acquis des systèmes relationnels (indépendance données/traitements), fiabilité et performances, compatibilité ascendante : l'utilisation de tables relationnelles est possible à travers des vues d'objet, et leur mise à jour à travers des procédures stockées.

• L'enrichissement du langage SQL par des extensions qui sont désormais normalisées (SQL:1999).

• La mise en œuvre des concepts objets (classes, héritage, méthodes) qui ont indéniablement démontré leur intérêt dans la maintenance des applications (modularité extensibilité et réutilisabilité).

2.2.3. Les Concepts additionnels essentiels / Notions de base

1. Type de données utilisateur (User Data Type)

C'est un type de données définissable par l'utilisateur composé d'une structure de données et d'opérations encapsulant cette structure.

Le système de type du SGBD devient donc extensible, et n'est plus limité aux types alphanumériques de base, comme avec le relationnel pur et SQL2.On pourra par exemple définir des types texte, image, point, ligne, etc. Les types de données définissables par l'utilisateur sont appelés **types abstraits (ADT)** en SQL3.La notion de type abstrait fait référence au fait que l'implémentation est spécifique de l'environnement ou seule l'interface d'un type utilisateur est visible **[GAR, 02].**

2. Patron de Collection (Collection Template)

C'est un type de données générique permettant de supporter des attributs multi values et de les organiser selon un type de collection.

Les SGBD objet relationnel offrent différents types de collections, tels que le tableau dynamique, la liste, l'ensemble, la table, etc. Le patron LISTE(X) pourra par exemple être instancié pour définir des lignes sous forme de listes de points : LIGNE LISTE(POINT) **[GAR, 02].**

3. Référence d'objet (Object Reference)

C'est un type de données particulier permettant de mémoriser l'adresse invariante d'un objet ou d'un n-uplet.

Les références sont les identifiants d'objets (OID) dans le modèle objet-relationnel.

Elles sont en théorie des adresses logiques invariantes qui permettent de chainer directement les objets entre eux, sans passer par des valeurs nécessitant des jointures **[GAR, 02].**

4. Héritage de type (Type inheritance)

C'est une forme d'héritage impliquant la possibilité de définir un sous-type d'un type existant, celui-ci héritant alors de la structure et des opérations du type de base.

L'héritage de type permet donc de définir un sous-type d'un type SQL ou d'un type utilisateur. Une table correspondant à un type sans opération, l'objet-relationnel permet aussi l'héritage de table **[GAR, 02].**

5. Héritage de table (Table inheritance)

C'est une forme d'héritage impliquant la possibilité de définir une sous-table d'une table existante, celle-ci héritant alors de la structure et des éventuelles opérations de la table de base **[GAR, 02].**

2.2.4. Le Langage de requête SQL3

SQL3 est la dernière version de SQL, qui y est étendu avec des fonctions orientées objets, les déclencheurs, le multimédia, le spatial, les séries temporelles, le support automatique de l'héritage, le parcours de référence, la constitution de tables imbriquées en résultat, etc..., tout en restant compatible avec le SQL classique relationnel: toute instruction de SQL classique est toujours valable en SQL3.

Les nouvelles possibilités offertes aux utilisateurs sont les suivantes :

- Définir de nouveaux domaines à structure complexe, appelés types ;
- Associer à chaque type des méthodes ;
- Créer des hiérarchies de types ;
- Créer des objets qui sont composés d'une valeur structurée et d'un OID ;
- Etablir des liens de composition par des attributs référence qui contiennent l'OID de l'objet composant ;
- Créer des tables contenant soit des tuples normaux (en première forme normale), soit des tuples en non première forme normale (des valeurs structurées), soit des objets **[S06]**.

2.2.5. Synthèse :

En résumé le modèle Objet Relationnel apporte cinq nouveautés au relationnel qui sont :

- Définir de nouveaux types complexes avec des fonctions pour les manipuler ;
- Une colonne peut contenir une collection (ensemble, sac, liste) ;
- Ligne considérée comme un objet, avec un identificateur (*Object Identifier* OID) ;
- Utilisation de références aux objets ;
- Extensions du langage SQL (SQL3 ou SQL99) pour la recherche et la modification des données.

En parallèle ce modèle présente des problèmes qui sont:

- Ne s'appuient pas sur une théorie solide comme le modèle relationnel ;
- Manquent de standard de fait d'implantations différentes, et encore partielles dans les divers SGBD **[GAR, 02].**

2.3. Le modèle semi-structuré (XML)

Les données semi structurées sont les données qui possède une structure flexible et qui n'ont pas un schéma à priori mais plutôt dont le schéma peut être extrait à partir de la donnée. La plupart du temps, un ensemble de données semi structurées est représenté sous la forme d'un graphe dont les feuilles contiennent les données et dont les nœuds et les liens représentent la structure de l'ensemble.

2.3.1. Le langage de description XML

XML (eXtended Markup Landuage) est un format textuel extensible de description de document défini par le W3C. De la famille des langages de marquage SGML, il permet de s'adapter à quasiment tous les domaines où l'on a besoin de structurer de l'information de façon portable.

XML permet de faire le lien entre un langage conçu plus spécialement pour le formatage de documents(SGML) et un modèle de données en émergence permettant une vision plus réaliste mais plus complexe des données qu'est le modèle **semi-structuré**. Ce langage permet ainsi de définir une structure de données et son contenu.

XML est conçu de façon à faciliter l'intégration et l'échange de données entre applications. Il isole le formatage et le rendu des documents par rapport à sa structure. C'est à des langages de style spécifiques tels que XSL (eXtended Style Sheet) qu'on laisse le soin de s'occuper du rendu de la page XML lors de la publication **[DAN, 03].**

2.3.2. Structure d'un fichier XML

XML est un langage à base d'éléments, d'étiquettes, d'attributs et de valeurs. Les balises (tag) ouvrantes (resp. fermantes) sont constituées d'*étiquettes* (label) représentées entre le symbole <resp. </) et le symbole >. Le composant logique constitué de la balise ouvrante, de la valeur et de la balise fermante est appelé élément (element). La valeur peut être vide, contenir du texte, d'autres éléments ou contenir un mélange des deux (mixed element content). Les balises définissent la structure du document. L'élément de plus haut niveau englobant tous les autres et n'ayant pas de parents est appelé élément racine. Un élément peut contenir des informations additionnelles appelées *attributs* (attributes). Un attribut est un couple formé d'un nom et d'une valeur et est représenté à l'intérieur de la balise ouvrante sous la forme nom = " valeur " Un document XML est un ensemble d'éléments ainsi imbriquées **[DAN, 03].**

Une source de données est un document XML si elle est « bien formée », c'est à dire si elle correspond parfaitement à la spécification de XML. Un document XML dispose alors de deux structures : une structure logique et une structure physique, les deux correspondant parfaitement.

Un document XML est représenté physiquement sous la forme d'un fichier texte structuré en éléments, à l'aide de balises éventuellement imbriquées **[LAC, 05]**.

En-tête: En en-tête du document doit figurer un « prologue », une déclaration qui identifie le document comme un document XML. Ce prologue indique la version de XML employée, le codage de caractères, et si le document est associé à une DTD ou s'il est autonome. Comme l'illustre l'exemple suivant :

< ?xml version= ''1.0'' encouding=''ISO-8859-1'' standalone =yes ?>

Cette expression indique au processus qui va traiter le document : La version du langage XML utilisé dans le document, le codage des caractères utilisé dans le document, si le document XML fait référence à une DTD ou à un schéma XML définit dans un autre fichier, il faut mettre '' no'' Sinon, on utilise ''yes''.

Racine : Il existe un élément particulier : l'élément « racine », encore appelé « élément document ». Cette racine doit contenir tous les autres éléments du document et ne peut apparaître qu'une fois dans un document XML. En conséquence, aucun autre élément ne contient la racine.

2.3.3. Les qualifications d'un document XML

Un document XML peut avoir deux qualifications, il peut être :

Bien formé : quand il respect la syntaxe du langage XML définie par le W3C.

Valide : quand il est associé à une définition de type de document et qu'il la respecte (nom des éléments, type, répétition et ordre d'apparition dans le document).

2.3.4. Exemple d'un document XML simple

L'exemple suivant de document XML permet de décrire une personne :

```
< ?xml version= "1.0'' encoding=''UTF-8'' ?>
   <EtatCivil>
       <nom>ALILICHE </nom>
       <prénom>Hafsa</prénom>
       <nationalité>Algérienne</nationalité >
       <adresse>
          <voie>248 rue 11 décembre ain beniane</ voie >
          <codePostal>16042</ codePostal >
          <ville>Alger</ville>
       </adresse >
  </EtatCivil>
```

2.3.5. Les langages de définition d'un document XML

Généralement, il existe deux langages pour définir un modèle XML **[KETT, 11]**

. DTD, Document Type Définition simplifié pour XML ;

. XML schéma mieux typé et plus puissant.

3.5.1. Les DTD [KETT, 11] : Le W3C propose une définition de document type appelée **DTD** (**D**ocument **T**ype **D**éfinition), c'est-à-dire une grammaire permettant de vérifier la validité du document XML. Une DTD est un fichier permettant de vérifier qu'un document XML est conforme à une structure donnée. La norme XML n'impose pas l'utilisation d'une DTD pour un document XML mais elle impose par contre le respect exact des règles de base de la norme XML. Une DTD peut être définie de deux façons :

1. Sous **forme interne**, en incluant la grammaire dans le même document XML, c'est à dire déclarer la DTD au début du document XML par <!DOCTYPEnom_racine [description de la DTD] > ;

2. Sous **forme externe**, soit en appelant un fichier contenant la grammaire à partir d'un fichier local ou bien en y accédant par son URL (adresse web).

3.5.2. Le schéma XML (XMLS) : Le W3C a proposé un nouveau langage de définition de schéma de documents qu'est XML Schéma pour traiter les déficiences des DTD. XML Schéma propose, en plus des fonctionnalités fournies par les DTD, plusieurs nouveautés à savoir **[KETT, 11]** :

1. Un grand nombre de types de données intégrés comme les booléens, les entiers, les intervalles de temps, etc. De plus, il est possible de créer de nouveaux types par ajout de contraintes sur un type existant ;

2. Des types de données utilisateurs qui permettent de crée notre propre type de données nommé ;

3. La notion d'héritage : les éléments peuvent hériter du contenu et des attributs d'un autre élément. C'est sans aucun doute l'innovation la plus intéressante de XML Schéma, car elle permet la réutilisation ;

4. Les indicateurs d'occurrences des éléments peuvent être tout nombre non négatif ;

5. Une grande facilité de conception modulaire des schémas ;

2.3.6. Interrogation des documents XML

L'interrogation de documents XML demande un langage spécifique pour traiter les particularités de XML. Ce langage doit être structuré pour permettre de naviguer dans un document, de poser des questions et d'utiliser la réponse. Pour cela plusieurs langages ont été mis en œuvre : XPath, XSLT, XQuery.

XPath est un langage pour extraire des noeuds dans un arbre XML:

- On navigue dans l'arbre grâce à des axes de navigation ;
- Un chemin de navigation est une séquence d'étapes ;
- Dans chaque étape on choisi un axe, un filtre et éventuellement des prédicats ;
- Le résultat d'une étape (d'une séquence d'étapes) est un séquence de nœuds.

XSLT est un langage pour transformer des arbres XML:

- On définit des règles de transformation qui transforme un fragment d'arbre en un autre fragment (copie) ;
- Les fragments à transformer sont choisis par des expressions XPath ;
- Les règles de transformations à appliquer à un fragment sont choisies par une expression XPath **[AMA, 03] ;**
- Le résultat d'une transformation est un document XML.

XQuery est un langage de requêtes pour permettre d'accéder à des composants structurels d'un document XML (p. ex. accéder au titre et aux auteurs d'un article scientifique).

- Les premières propositions de langage de requêtes à la SQL pour SGML ou XML datent de 1994**[KETT, 11].**

Conclusion

Dans ce chapitre nous avons défini les trois modèles de données, avec leurs concepts de base, aussi le langage de manipulation de chacun d'eux. En accordant ces modèles avec notre problématique, nous remarquons qu'il y'a une hétérogénéité totale entre ces derniers, et pour éliminer ce problème nous allons commencer à concevoir notre système de médiation dans le chapitre suivant.

Conception du système

Introduction

L'objectif de notre travail, est de concevoir et réaliser un système de médiation des sources de données hétérogènes. Pour ce faire et afin de mettre en place de nouvelles idées, nous allons présenter une étude comparative des différents systèmes de médiations, citées précédemment et aussi nous présentons, la solution proposée de notre système d'une manière globale. Ensuite, nous définissons le processus général. Et aussi le processus unifié (Unified Process) que nous avons suivis pour la réalisation de ce projet. Par la suite et dans le but d'élucider les besoins, que doit vérifier notre système nous décrivons d'une façon détaillée la solution de notre système, en se basant sur le langage UML. Et nous présentons enfin l'architecture du système.

1. Solution proposée

1.1. Etude comparative des systèmes de médiation

Nous avons introduit vers la fin du premier chapitre (section 3.4) les différents travaux réalisés à base d'ontologies et après avoir fait une étude approfondie de chacun de ces derniers nous sommes arrivés à faire -en guise de synthèse- le tableau comparatif :

	SIMS	*MOMIS*	*OBSERVER*	*PICSEL*	*WASSIT*
Sources à intégrer	Données/Connaissances	Données	Données	Connaissances	Données
Langage utilisé	LOOM	ODL	CLASSIC	ALN-CARIN	XML
Architecture	Ontologie Unique	Ontologie Global « GW »	Ontologies Multiples	Approche Hybride	Approche Hybride
Mapping utilisé	GAV	GAV	Table de Correspondances	LAV	GAV
Inconvénients	Spécialisé dans un seul domaine d'application	Génération de l'ontologie globale semi-automatique	Relations entre les ontologies sont limitées à des relations lexicales.	Mapping généré semi-automatique	Utilisation de l'algèbre XML

Tableau 2 : Tableau comparative des systèmes de médiation

1.2. Description de la solution proposée

Après avoir fait une comparaison dans le Tableau 2, nous avons remarqué que la plupart des systèmes utilisent une architecture à ontologie unique tel que, les systèmes SIMS et MOMIS. Cette approche peut être intéressante dans le seul cas où les sources de données appartiennent au même domaine d'application, pour pouvoir trouver une ontologie globale, mais si l'administrateur veut mettre à jour une des sources de données, il doit prendre en considération les autres sources et l'ontologie globale, ce qui peut poser un problème d'autonomie des sources.

D'autres part, pour les systèmes qui sont basés sur des ontologies multiples, tel que le système OBSERVER, la comparaison des ontologies pourrait être difficile si elles n'ont pas une ontologie globale qui les regroupe, et en plus le mapping entre chaque couple d'ontologies doit être définis PICSEL et WASSIT sont les systèmes qui utilisent l'approche hybride tel que le premier utilise son propre formalisme de langage de description et le second est fondé sur le modèle de données XML et son langage de requête XQuery.

Nous avons vu dans l'état de l'art qu'une des solutions des problématiques des systèmes de médiation se basait sur les ontologies est l'architecture. L'architecture hybride est la mieux adaptée pour un système de médiation basée sur plusieurs ontologies. L'architecture hybride consiste à associer à chacune des sources une ontologie locale, et utilise une ontologie globale commune pour les lier. Les méthodologies existantes proposent une démarche dscendante, de l'ontologie globale vers les ontologies locales. Cette méthode ne facilite pas le travail de réconciliation sémantique. La démarche qui prend en charge la construction de telles ontologies doit être facilitée par la résolution de l'hétérogénéité sémantique entre ces ontologies. Le processus de développement des ontologies doit être particulièrement fiabilisé pour la construction de l'ontologie globale, car cette ontologie assure la liaison entre les différentes ontologies locales.

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.

Pour faire face à ces problèmes rencontrés, nous proposons un système de médiation basé sur l'approche d'ontologie hybride, pour assurer l'autonomie des sources tout en bénéficiant d'un vocabulaire commun. Nous avons utilisé OWL comme langage de représentation globale, pour représenter les ontologies grâce à sa puissance d'expression des concepts, et de leurs relations. Et pour la fusion des ontologies, nous avons utilisé un outil de fusion FOnES [2] (Fusion d'Ontologies par Enrichissement Sémantique), qui est un système de construction d'ontologies, par fusion des ontologies existantes, en traitant le problème de l'hétérogénéité sémantique d'une manière automatique. La fusion des ontologies permette de créer une ontologie unique et cohérente, les différentes ontologies à propos du même sujet sont fusionnées en une seule qui les unifie toutes. Nous avons choisis l'approche GLAV pour décrire le mapping du système global parce qu'elle règle le problème d'autonomie des sources de données .Nous avons réalisés la transformation des schémas des sources (XML schéma, relationnel, objet, etc.) vers une ontologie OWL.

Quant au traitement de requêtes, la requête initiale sera enrichie par des concepts de liaison, il y'a aussi la création d'un catalogue de manière automatique, qui sera utilisé par tous les modules du système de médiation. Il comporte toutes les correspondances entre les différentes couches du système, et contient toutes les données et leur emplacement nécessaires pour le traitement d'une requête, à savoir les ontologies, les informations sur les sources, le mapping entre l'ontologie globale et les ontologies locales.

[2] FOnES : a été réalisé dans le cadre de projet de master en informatique à l'USDB con ce dernier n'est pas outil open source ou commercialisé et donc il n'est pas connu .

2 .Présentation de la démarche utilisée

UML est un langage qui permet de représenter des modèles, mais il ne définit pas le processus d'élaboration des modèles. Cependant, dans le cadre de la modélisation d'une application informatique, les auteurs d'UML préconisent d'utiliser une démarche. Pour standardiser les démarches, plusieurs modèles de démarches ont été décrits et parfois formalisés, parmi ces derniers, UP.

2.1. Le processus unifié UP (Unified Process)

Le processus unifié est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques **[DAO et CHI, 09].**

Les activités de l'UP

1. Expression des besoins : Cette phase permet de définir les différents besoins :

- Recenser les besoins fonctionnels (du point de vue de l'utilisateur) qui conduisent à l'élaboration des modèles de cas d'utilisation ;
- Inventorier les besoins principaux et fournir une liste de leurs fonctions. Apprendre les besoins non fonctionnels (technique) ;
- Le modèle de cas d'utilisation présente le système du point de vue de l'utilisateur et le représente sous forme de cas d'utilisation et d'acteurs et les besoins du client.

2. Analyse: L'objectif de l'analyse est d'accéder à une compréhension des besoins et des exigences du client .Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisation, et aussi les structures sous une forme qui facilite la compréhension (scénarios), la préparation, la modification et la maintenance du futur système.

3. Conception : La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle constitue un point de départ à l'implémentation :

- Elle décompose le travail d'implémentation en sous-systèmes ; et
- Elle crée une abstraction transparente de l'implémentation.

4. Implémentation : L'implémentation est le résultat de la conception pour implémenter le système sous formes de composants, c'est-à-dire, de code source, de scripts, de binaires, d'exécutables et d'autres éléments

5. Test: Les tests permettent de vérifier des résultats de l'implémentation en testant la construction. Pour mener à bien ces tests, il faut les planifier pour chaque intégration, les implémenter en créant des cas de tests, effectuer ces tests et prendre en compte le résultat de chacun **[DAO et CHI, 09]**.

2.2. UML :

La plupart des différents cycles de vie d'un projet font ressortir les notions de spécification, conception et analyse. Depuis les années 80, ils ont été mises en œuvre suivant différents procédés, parmi eux, il se trouve le langage UML.

2.2.1. Définition d'UML

UML (Unified Modeling Language) est un langage de modélisation qui a pour but de faciliter les transitions, lors du développement d'un projet, du besoin originel à la phase d'implémentation.

Ce langage, est une fusion de 3 méthodes orientées objet très utilisées (OMT, Booch et OOSE) dont le développement a débuté en 1994. et qui a été standardisé en 1997 par l'OMG dans sa version 1.1, il s'appuie essentiellement sur la technologie objet et les concepts qu'elle véhicule. Néanmoins, l'utilisation d'UML n'est pas restreinte au développement de systèmes informatisés, mais peut servir au développement de systèmes de gestion, tout comme à la résolution de problèmes d'organisation de tout style. De plus. UML. de par sa représentation essentiellement graphique, se veut extrêmement vulgarisé et intuitif, et est utilisable par les êtres humains ainsi que les machines.

2.2.2. Objectifs du l'UML

Au final, le langage UML est une synthèse de tous les concepts et formalismes méthodologiques les plus utilisés, pouvant être utilisé, grâce à sa simplicité et à son universalité, comme langage de modélisation pour la plupart des systèmes devant être développés.

Le langage UML permet ainsi d'apporter des solutions lors du développement de systèmes informatisés :

- Décomposer le processus de développement en distinguant la phase d'analyse (aspects fonctionnels) de la phase de réalisation (aspects technologiques et architecturaux) ;
- Décomposer le système en sous-systèmes plus facilement abordables : réduction de la complexité, répartition du travail, réutilisation des sous-systémiers ;
- Utiliser une technologie de haut niveau proche de la réalité pour aborder le développement. **[S11]**

3. Expression des besoins

3.1. Identification des acteurs

La première étape de cette phase est d'énumérer les acteurs susceptibles d'interagir avec le système. Un acteur représente l'abstraction d'un rôle joué par des entités (utilisateur, dispositif matériel ou autre système), qui interagissent directement avec le système étudié. Le tableau ci-dessous identifie les acteurs de notre système et décrit la mission de chacun d'eux :

Acteur	Désignation
Utilisateur	Les utilisateurs tiers sont des utilisateurs non privilégiés. L'utilisateur s'occupe du lancement de la requêté.
Médiateur	Le médiateur est chargé du traitement de la requête et de produire le résultat.
Adaptateur	L'adaptateur est l'acteur qui s'occupe de la traduction de la requête d'un langage à un autre.

Tableau 3 : Les acteurs du système.

3.2. Identification des cas d'utilisation

L'identification des cas d'utilisation, donne un aperçu des fonctionnalités futures que doit implémenter le système. Cependant, il nous faut plusieurs itérations pour arriver à constituer des cas d'utilisation complets. D'autres cas d'utilisation vont apparaître au fur à mesure de la description de ceux là, et l'avancement dans le « recueil des besoins fonctionnels ».

Dans notre cas d'étude nous avons défini les diagrammes de cas d'utilisation suivant:

- Création du schéma global
- Création du catalogue
- Lancement d'une requête
- Traitement d'une requête
- La traduction des sous requêtes

Nous détaillerons chaque cas d'utilisation qui doit faire l'objet d'une définition a priori qui décrit l'intention de l'acteur lorsqu'il utilise le système, et les séquences d'actions principales qu'il est susceptible d'effectuer. Ces définitions servent à fixer les idées et n'ont pas pour but de spécifier un fonctionnement complet et définitif.

3.2.1. Diagramme de cas d'utilisation global

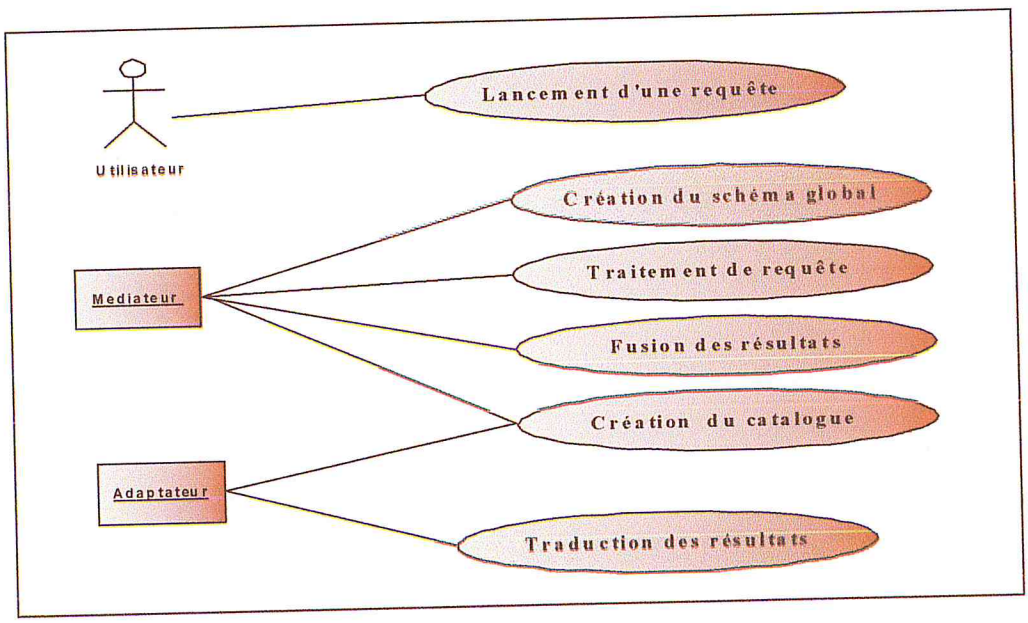


Figure 9 : Diagramme de cas d'utilisation global du système

3.2.2. Diagramme de cas d'utilisation « Lancement d'une requête »

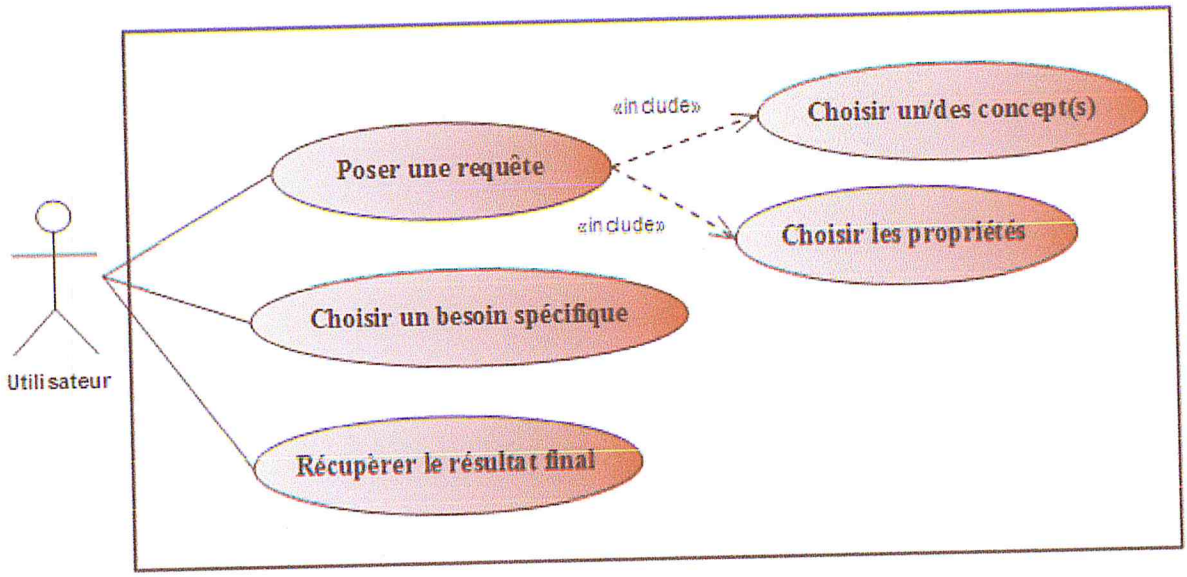


Figure 10 : Diagramme de cas d'utilisation « Lancement d'une requête »

Description de l'enchainement :

- **Titre :** lancement d'une requête
- **Acteur :** Utilisateur
- **But :** Consulter un ensemble de données
- **Description :** L'utilisateur choisit un ou plusieurs concepts avec leurs propriétés dans la liste affiché sur l'interface ou bien il choisit l'un des besoins déjà préparé selon les critères qu'il veut avoir.
- **Pré condition :** Nécessité d'avoir des informations dans un domaine donné.
- **Post condition :** Création d'une requête OWL sur le schéma global.

3.2.3. Diagramme de cas d'utilisation « Création du schéma global »

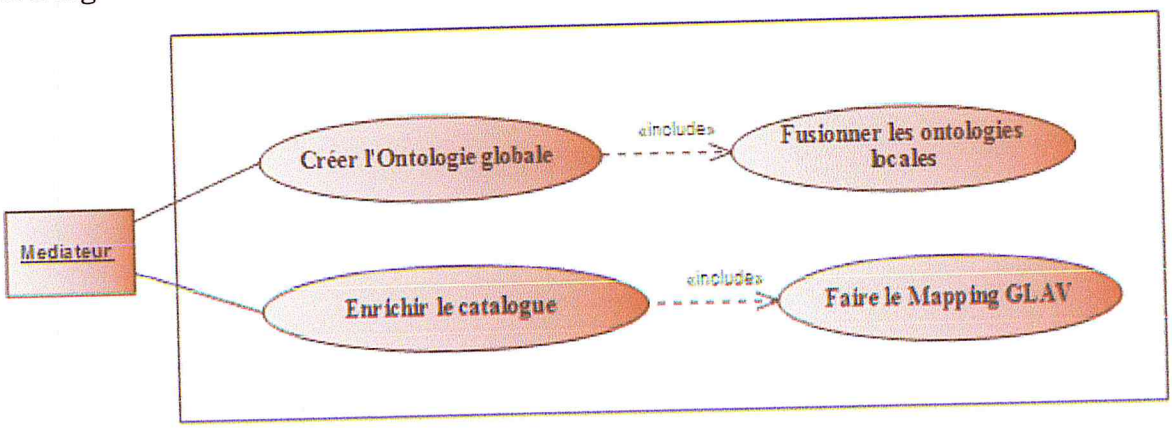


Figure 11 : Diagramme de cas d'utilisation création du schéma global

3.2.3.1. Le cas d'utilisation « Enrichissement du catalogue »

Description de l'enchainement :

- **Titre :** Enrichissement du catalogue
- **Acteur :** Médiateur
- **But :** pour optimiser le traitement de requête en évitant le mapping entre l'ontologie globale et les ontologies locales.
- **Pré condition :** Existence du catalogue
- **Post condition :** Catalogue
- **Exception :** Aucune.

3.2.4. Diagramme de cas d'utilisation « Traitement d'une requête »

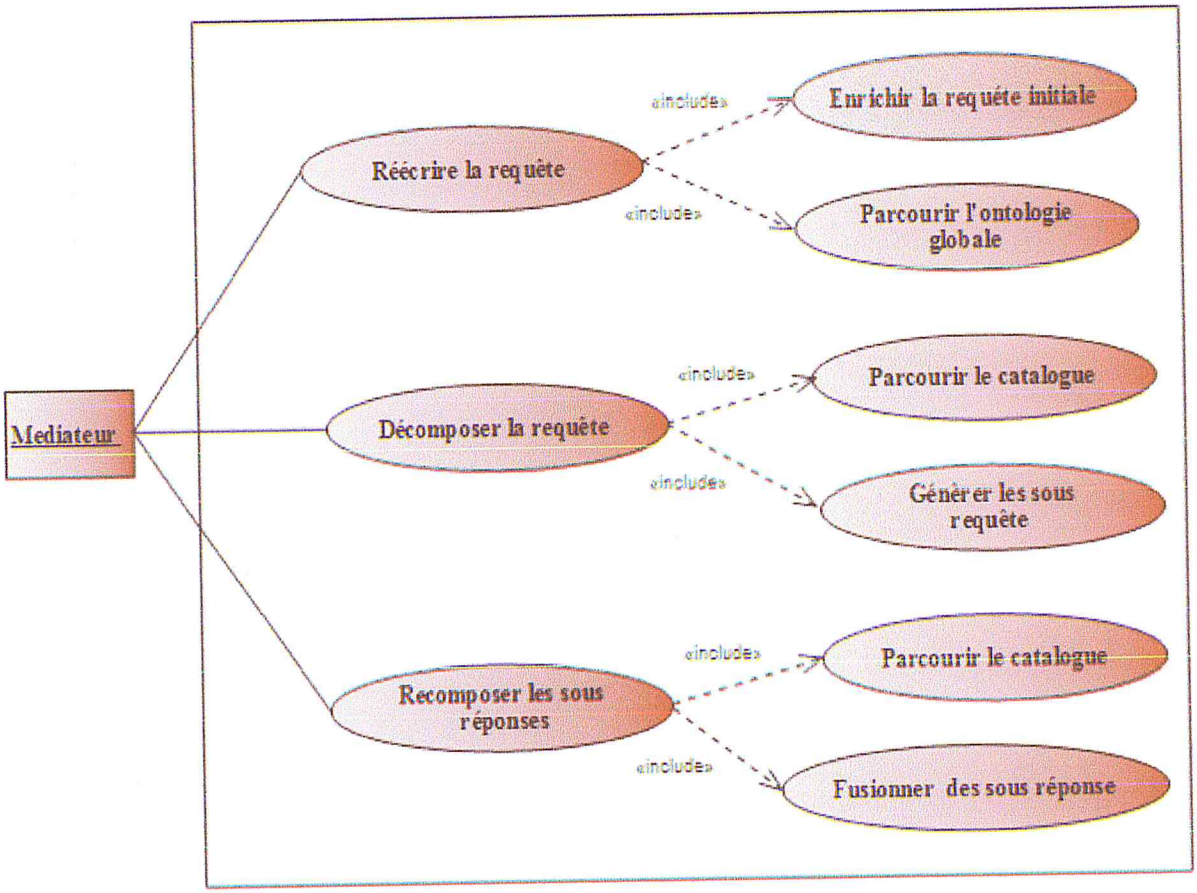


Figure 12 : Diagramme de cas d'utilisation « Traitement d'une requête »

3.2.4.1. Le cas d'utilisation « Réécriture de la requête »

Description de l'enchainement :

- **Titre :** la réécriture de la requête.
- **Acteur :** Médiateur.
- **But :** Transformer selon les concepts du schéma global.
- **Pré condition :** Requête initiale.
- **Post condition :** Requête enrichie.
- **Exception :** Retourner la requête initiale.

3.2.4.1. Le cas d'utilisation « Décomposer la requête »

Description de l'enchainement :

- **Titre :** la décomposition de la requête.
- **Acteur :** Médiateur.
- **But :** La décomposition de la requête initiale en sous requêtes selon les ontologies locales.
- **Pré condition :** Une requête.
- **Post condition :** Des sous requêtes.

3.2.4.2. Le cas d'utilisation « Recomposer les sous réponses »

Description de l'enchainement :

- **Titre :** Recomposer les sous réponses.
- **Acteur :** Médiateur.
- **But :** Fusionner des sous réponse selon la requête initiale.
- **Pré condition :** Des sous réponses.
- **Post condition :** Réponse fusionnée.

3.3.5. Diagramme de cas d'utilisation « Traduction des sous requêtes/réponses »

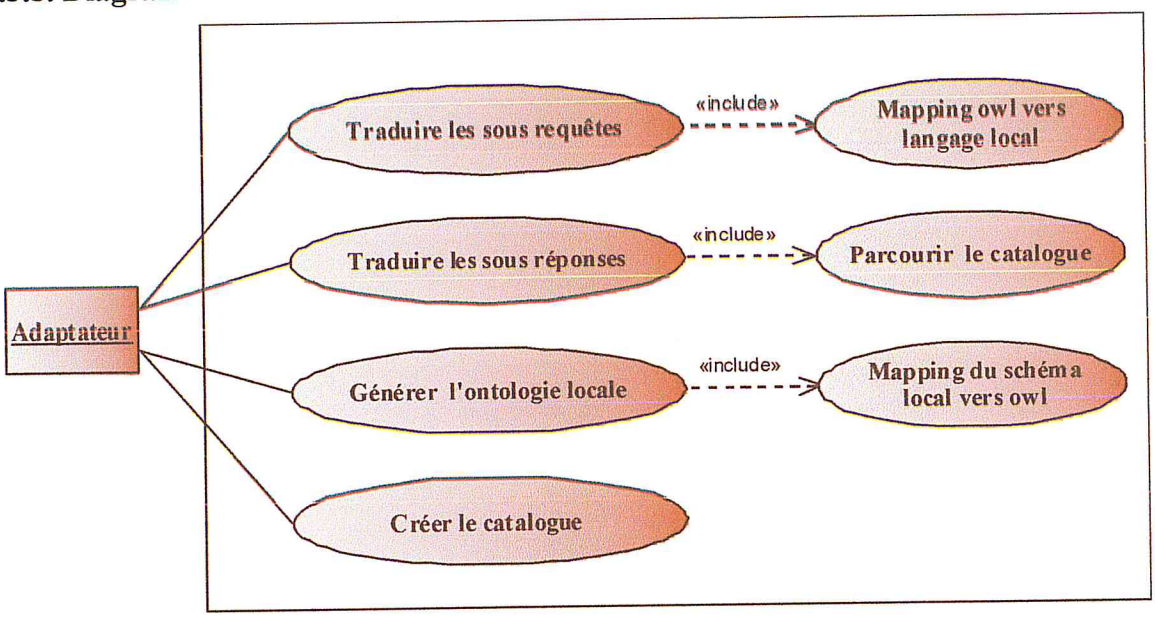


Figure 13 : Diagramme de cas d'utilisation de la traduction

3.3.5.1. Le cas d'utilisation « Traduction des sous requêtes »

Description de l'enchainement :

- **Titre :** La traduction des sous requêtes.
- **Acteur :** Adaptateur
- **But :** La transformation d'une sous requête en OWL vers les langages des sources de données
- **Pré condition :** Sous requête en OWL.
- **Post condition :** Sous requête en langage des sources de données

3.3.5.3. Le cas d'utilisation « Génération de l'ontologie locale »

Description de l'enchainement :

- **Titre :** La génération de l'ontologie locale
- **Acteur :** Adaptateur
- **But :** La création des ontologies locales
- **Pré condition :** Le schéma des bases de données
- **Post condition :** L'ontologie locale

3.3.5.4. Le cas d'utilisation « Création du catalogue »

Description de l'enchainement

- **Titre :** la création du catalogue
- **Acteur :** Adaptateur
- **But :** Avoir les mappings entre les sources et les ontologies locales
- **Pré condition :** Aucune
- **Post condition :** Catalogue créé.

4. Analyse du système

4.1. Diagramme de séquence

4.1.1. Qu'est qu'un scénario ?

Un scenario représente un ensemble ordonné de messages échangés par des objets. On parle ici d'objet au sens large : instance de classe d'analyse ou instance d'acteur. Les échangés de messages entre objets peuvent être représentés en UML dans une sorte de diagramme complémentaire appelé diagramme de séquence.

4.1.2. Qu'est qu'un diagramme de séquence ?

Un diagramme de séquence est une représentation séquentielle du déroulement des traitements et des interactions entre les éléments du système et /ou de ses acteurs. Les diagrammes de séquences permettent de représenter des collaborations entre objets selon un point de vue temporel on y met l'accent sur la chronologie des *envoi*-de messages.

Dans ce qui suit nous allons présenter les diagrammes de séquence afin de formaliser les scénarios des cas d'utilisation vus précédemment. Nous allons voir le système comme un ensemble d'objets en interaction.

4.1.3. Diagramme de séquences « Traitement de requête »

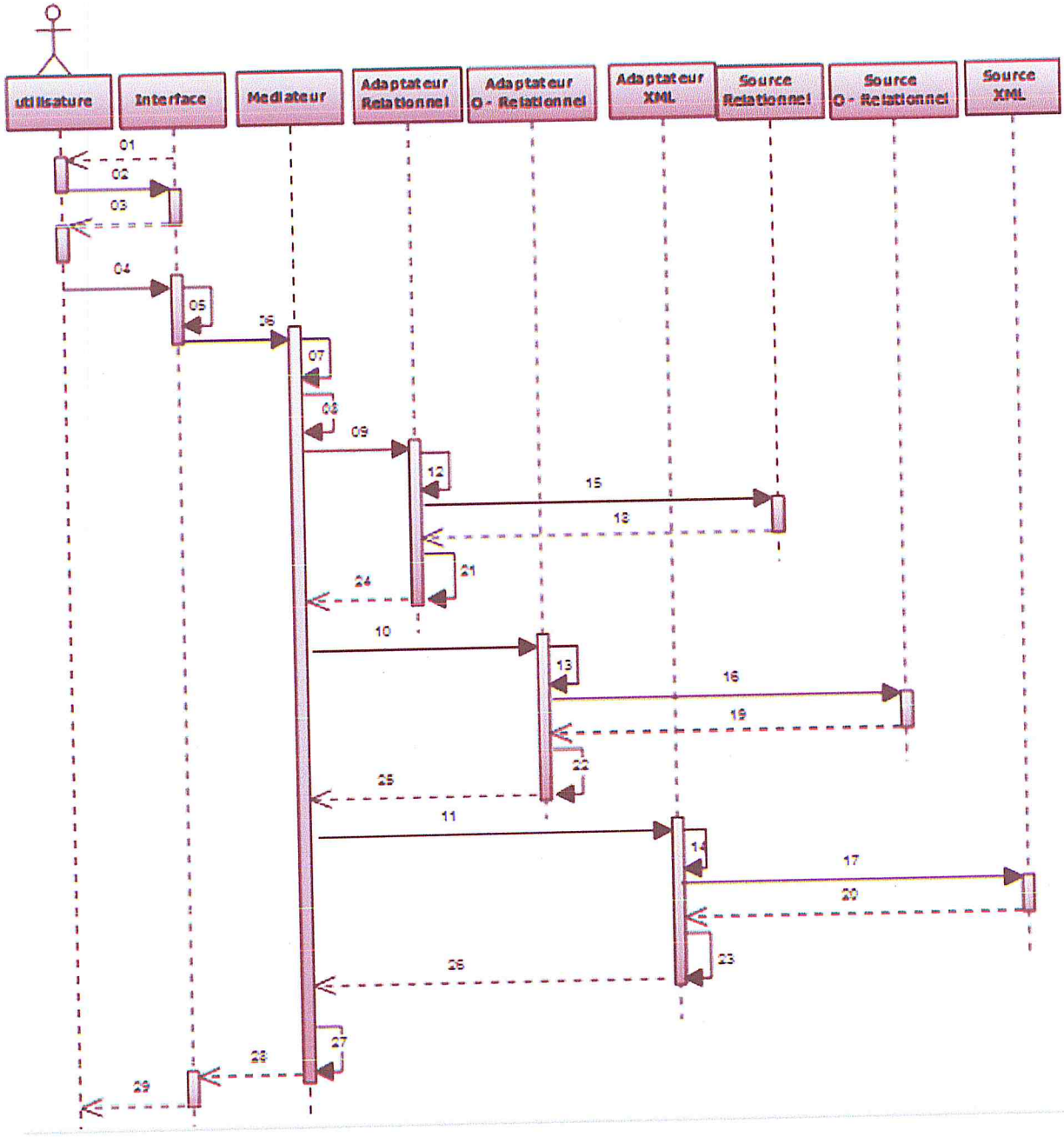


Figure 14 : Diagramme de séquences « Traitement d'une requête »

4.1.4. Scénario du diagramme de séquences « Traitement de requête »

1 : Afficher la page d'accueil.

2 : Choisir la page « accéder aux données ».

3 : Renvoyer la page.

4 : Saisir la requête.

5 : Vérifier la validité de la requête.

6 : Envoyer la requête au médiateur.

7 : Réécriture de la requête.

8 : Décomposer la requête en sous requête

9 : Envoyer une sous requête à l'adaptateur relationnel.

10 : Envoyer une sous requête à l'adaptateur objet-relationnel.

11 : Envoyer une sous requête à l'adaptateur XML.

12 : Traduire la sous requête du langage local en SQL.

13 : Traduire la sous requête du langage local en SQL3.

14 : Traduire la sous requête du langage local en XML.

15 : Interroger la source relationnelle.

16 : Interroger la source objet-relationnel.

17 : Interroger le document XML.

18 : Renvoyer la réponse de la requête envoyée par l'adaptateur relationnel.

19 : Renvoyer la réponse de la requête envoyée par l'adaptateur objet-relationnel

20 : Renvoyer la réponse de la requête envoyée par l'adaptateur XML

21 : Traduire la réponse relationnelle en langage local.

22 : Traduire la réponse objet-relationnelle en langage local

23 : Traduire la réponse XML en langage local.

24, 25 et 26 : Envoie la sous réponse au médiateur.

27 : Recomposer les résultats.

28 : Renvoyer le résultat final

29 : Affichage du résultat.

4.1.5. Diagramme de séquences «Création du schéma global »

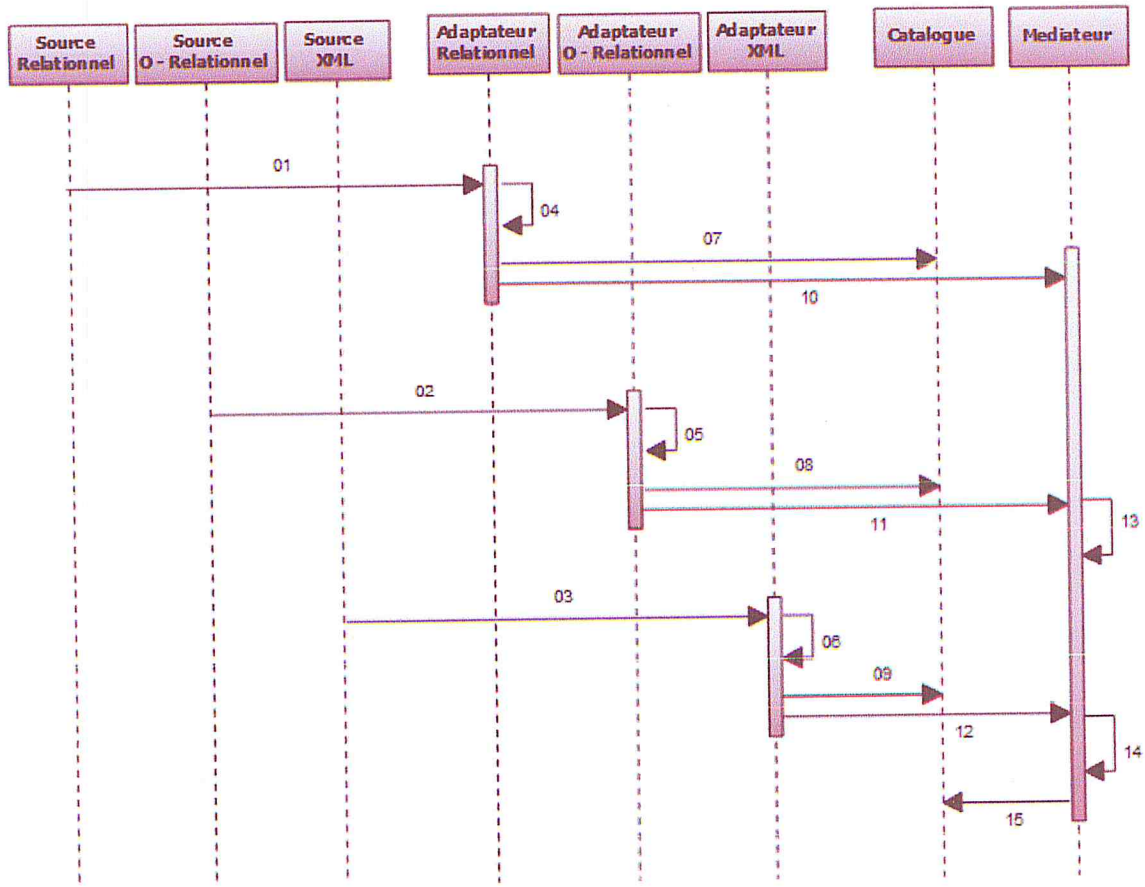


Figure 15 : Diagramme de séquences «Création du schéma global»

4.1.6. Scénario du diagramme de séquences « Création du schéma global »

01 : Envoyer le schéma de la source relationnel.

02 : Envoyer le schéma de la source objet-relationnel.

03 : Envoyer le schéma de la source XML.

04, 05 et 06 : Transformer les schémas en OWL.

07, 08 et 09 : Alimenter le catalogue par les concepts locaux

10 ,11 et 12 : Envoyer les ontologies locales pour la fusion

13 : Lancer la fusion des deux ontologies locales (A= ontoLocal XML+ ontoLocal O-R)

14 : Lancer la fusion pour le reste des ontologies (A+ontoLocal R)

15 : Alimenter le catalogue par les concepts globaux.

4.1.7. Diagramme d'activité

Le diagramme d'activité n'est rien d'autre que la transcription de la représentation du processus dans UML telle qu'elle a été élaborée lors du travail qui a préparé la modélisation : il montre l'enchaînement des activités qui concourent au processus.

4.1.7.1. Diagramme d'activité « création du schéma global »

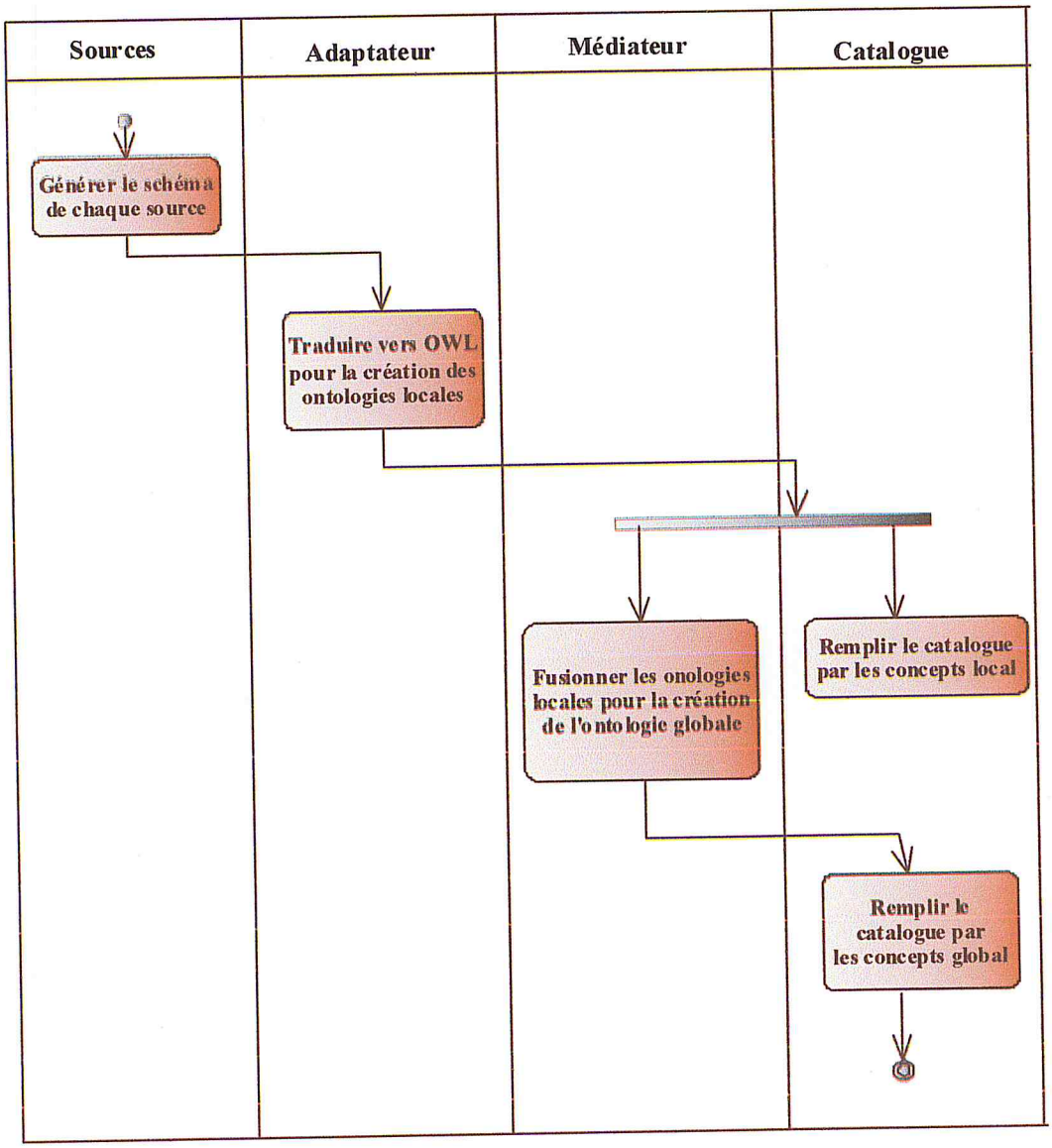


Figure 16 : Diagramme d'activité du cas d'utilisation « création du schéma global»

4.1.7.2. Diagramme d'activité «Traitement de requête »

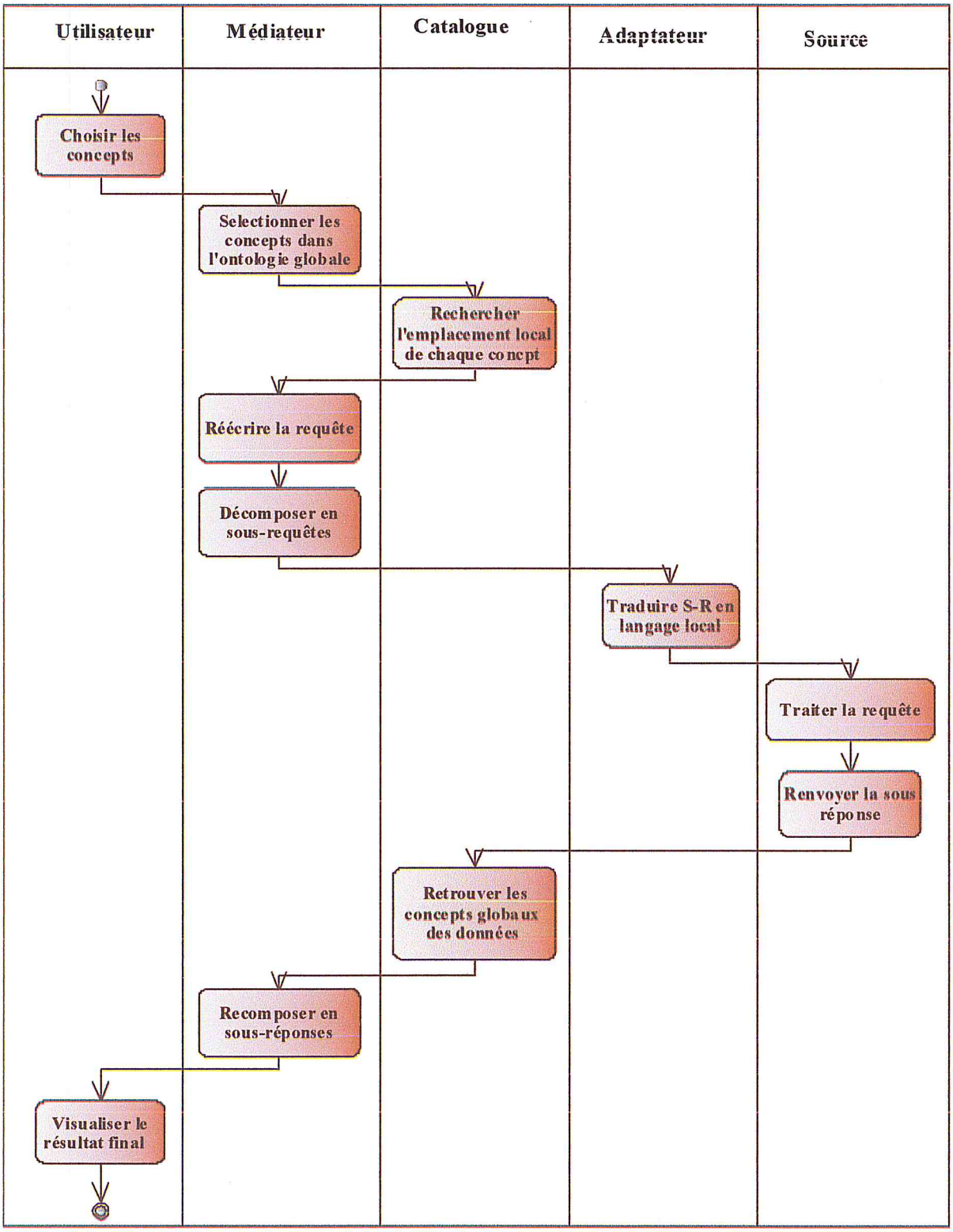


Figure 17 : Diagramme d'activité « Traitement d'un requête »

5. Conception du système

5.1. Diagramme de classes

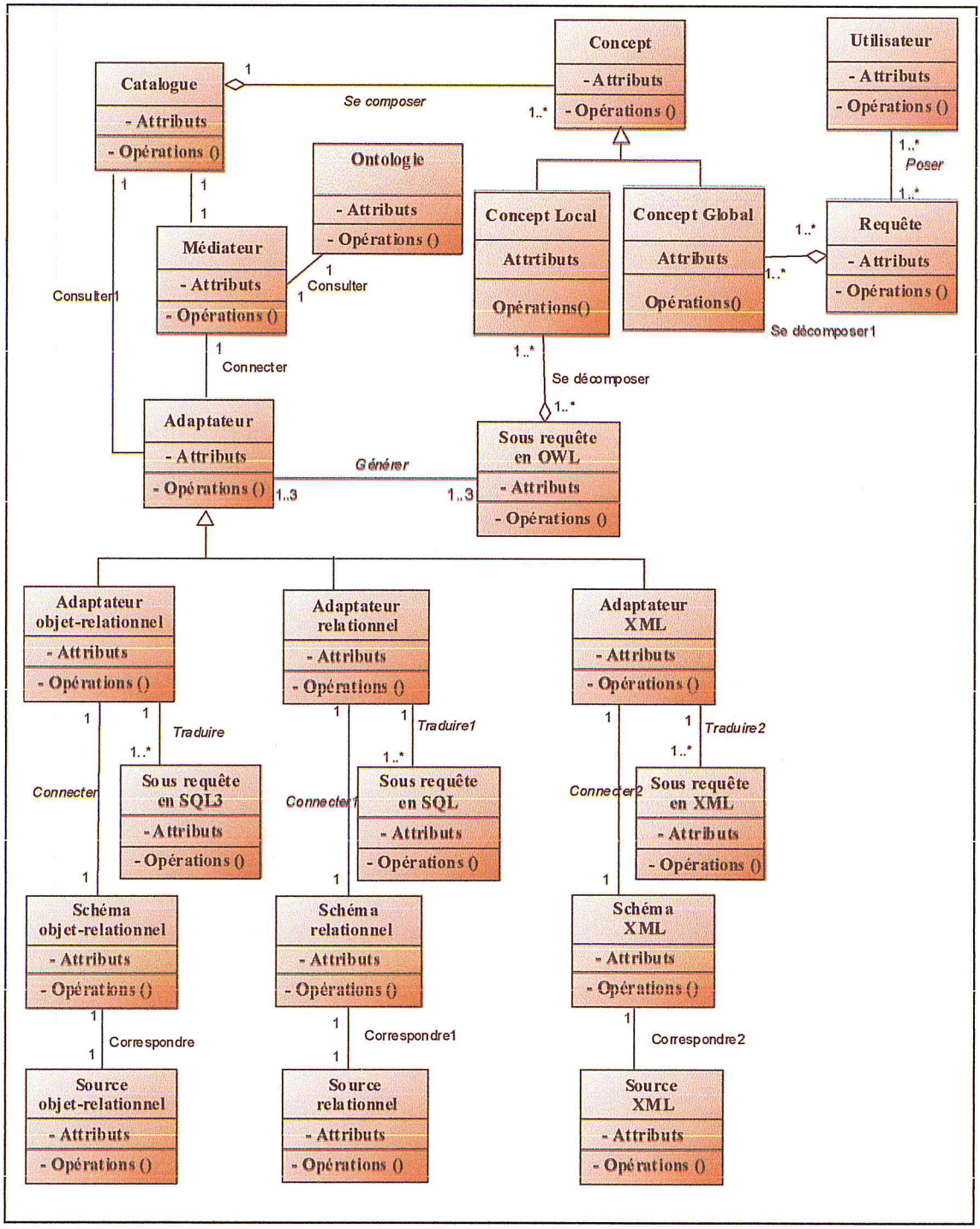


Figure 18 : Diagramme de classes du système

5.1. Liste des attributs :

Nom Classe	*Attribut*	*Désignation*
Catalogue	Nom_Fichier	Nom du catalogue
	Emplacement	
Concept	Nom_concept	Nom du concept
	Propriétés	Propriétés du concept
Utilisateur	Identifiant	Le nom de l'utilisateur
	Mot_de_passe	Le mot de passe
Requête	Select	Les attributs globaux
	From	Les concepts globaux
	Where	Les conditions sur la sélection des attributs
Médiateur	Nom_ontoG	Le nom de l'ontologie globale
	Nom_catalogue	Le nom du catalogue
Ontologie globale	Nom_OntoG	Nom de l'ontologie
	Emplacement	Le chemin de l'ontologie
Adaptateur	Nom-adat	Le nom de l'adaptateur
	Emplacement	Emplacement de l'adaptateur
Adaptateur objet-relationnel	AdatOR	Le nom de l'adaptateur
	Emplacement	Emplacement de l'adaptateur
Adaptateur relationnel	AdatR	Le nom de l'adaptateur
	Emplacement	Emplacement de l'adaptateur
Adaptateur xml	AdatXML	Le nom de l'adaptateur
	Emplacement	Emplacement de l'adaptateur
Sous requête en SQL3	Select	Les attributs locaux
	From	Les concepts locaux
	Where	Les conditions sur la sélection des attributs
Sous requête en SQL	Select	Les attributs de locaux
	From	Les concepts locaux
	Where	Les conditions sur la sélection

		des attributs
Sous requête en XML	attribut	Les attributs locaux
	concept	Les concepts locaux
Schéma objet-relationnel	Sch_OR	Le nom du fichier
	Emplacement	Emplacement du fichier
Schéma relationnel	Sch_R	Le nom du fichier
	Emplacement	Emplacement du fichier
Schéma XML	Sch_XML	Le nom du fichier
	Emplacement	Emplacement du fichier
Source relationnel	Nom_Src1	Le nom du fichier
	Emplacement	Emplacement du fichier
Source objet-relationnel	Nom_Src2	Le nom du fichier
	Emplacement	Emplacement du fichier
Source XML	Nom_Src3	Le nom du fichier
	Emplacement	Emplacement du fichier
Concept_Global	Nom_concept	Nom du concept
	Propriétés	Propriétés du concept
Concept_Local	Nom_concept	Nom du concept
	Propriétés	Propriétés du concept

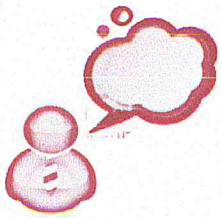
Tableau 4 : Définition des classes et des attributs

5.2. Liste des opérations :

Nom Classe	*Méthode*	*Désignation*
Catalogue	Créer()	Création automatique du catalogue
Concept	Consulter_Cpt()	Consulter le concept
Utilisateur	Se conneter (mot_passe,identifiant)	L'utilisateur se connecte au système.
	Poser_Requ (ConceptG,Propriétés)	L'utilisateur pose sa requête.
Requête	Créer_requ()	Créer la requête intiale.
Médiateur	Créer_schèmaGlobal ()	La création du shèma global
Sous requête en OWL	Créer_ssReq() Décomposer_ssReq()	Créer la sous requête Décomposer la sous requête
Adaptateur	Traduire (sous_requête)	Traduire les sous requêtes
Adaptateur objet-relationnel	Traduire (sous_requêteOR)	Traduire la sous requête en Objet-Relationnel
Adaptateur relationnel	Traduire (sous_requêteR)	Traduire la sous requête en Relationnel
Adaptateur XML	Traduire (sous_requêteXML)	Traduire la sous requête en XML
Sous requête en SQL3	Créer_ssReq()	Créer la sous requête
	Décomposer_ssReq()	Décomposer la sous requête
Sous requête en SQL	Créer_ssReq() Décomposer_ssReq()	Créer la sous requête Décomposer la sous requête
Sous requête en XML	Créer_ssReq() Décomposer_ssReq()	Créer la sous requête Décomposer la sous requête

Tableau 5: Définition des opérations

5.2. Architecture du système



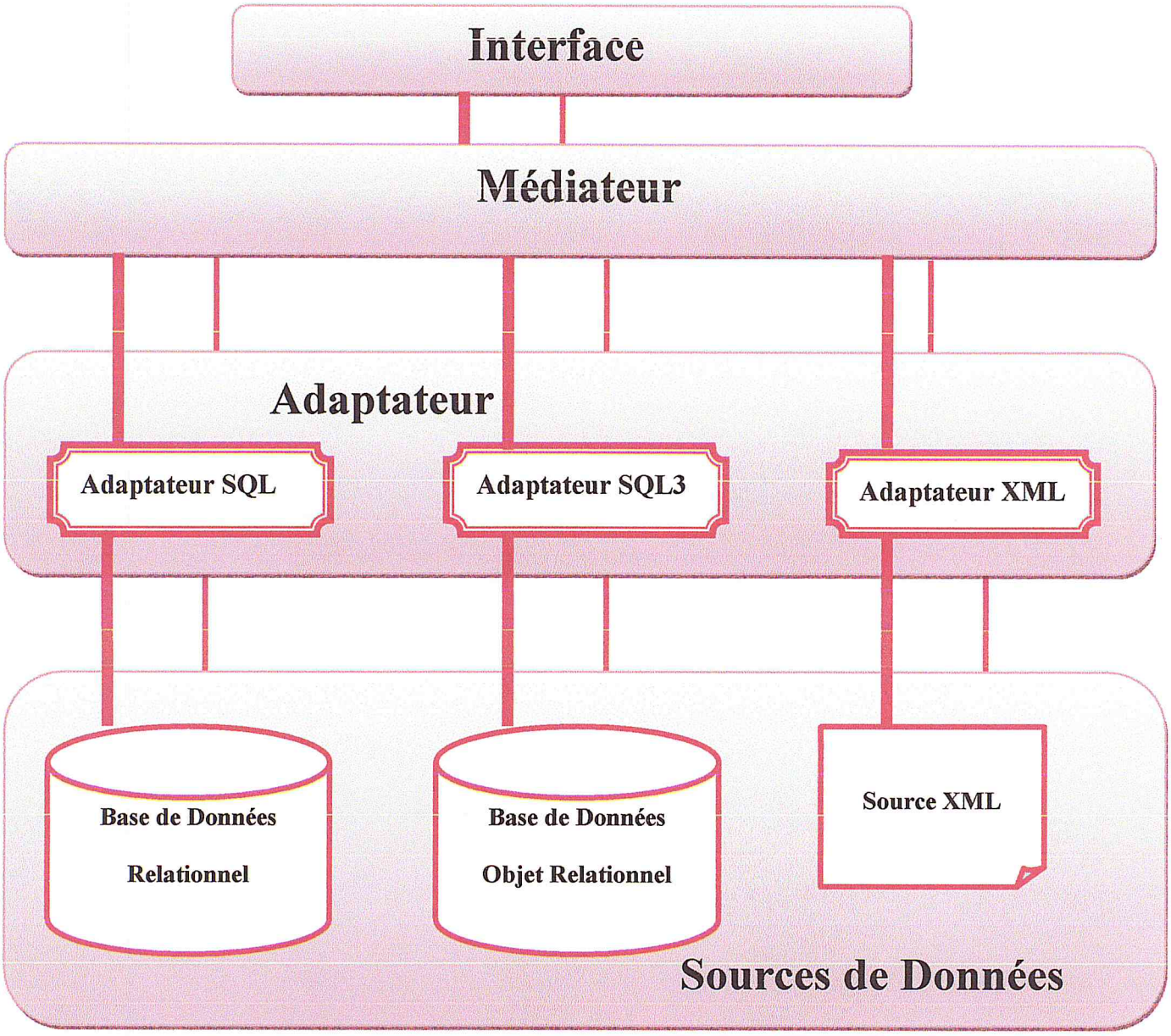


Figure 19: Architecture du système

5.4. Description des couches du système

Couche01 « **Interface**» : C'est la couche la plus simple du système, elle contient une interface graphique pour les utilisateurs du système afin qu'ils puissent poser leur requêtes ou choisir leur besoins en sélectionnant les concepts et recevoir des réponses.

Couche02 « **Médiateur** » **:** Au niveau du médiateur, nous retrouvons trois modules qui collaborent entre eux pour garantir le bon fonctionnement du système.

Elle contient le module catalogue, ou nous avons spécifié le mapping général du schéma global. Aussi le module de création du schéma global en utilisant le mapping GLAV, c'est la partie la plus importante dans cette couche, vu qu'elle regroupe tous les concepts et leurs propriétés sous le langage OWL. Le lien entre ces deux modules est que lors de la création de l'ontologie globale, le catalogue sera alimenté de façon automatique avec les concepts globaux.

Enfin le module de traitement de requête qui est le noyau du système, chargé de faire la liaison avec les autres modules de cette couche lors de la réécriture de la requête, nous devons passer par l'ontologie globale pour sélectionner les concepts désirés, et aussi lors de la décomposition nous passons par le catalogue afin d'optimiser le temps de réponse, c'est à dire nous éviterons de faire le mapping entre l'ontologie globale et les ontologies locale.

Couche03 « **Couche adaptateur** » **:** L'adaptateur cache l'hétérogénéité au médiateur. Il est associé à une seule source et joue le rôle d'intermédiaire entre cette source et le médiateur.

C'est un « Traducteur » qui fait :

- La Traduction de la source de langage natif propre à la source en langage pivot ;
- L'extraction des résultats de la sous requête envoyés par le médiateur ;
- L'envoie lu résultat au médiateur.

Couche04 « **Sources de Données** » **:** Cette couche est la plus basse dans le système, elle contient les sources de données et leur schémas, dans notre cas nous étudierons modèles de données suivants : relationnel, objet-relationnel et XML, nous détaillerons ces sources de données avec des exemples dans le prochain chapitre consacré à l'implémentation.

5.3. Description des modules

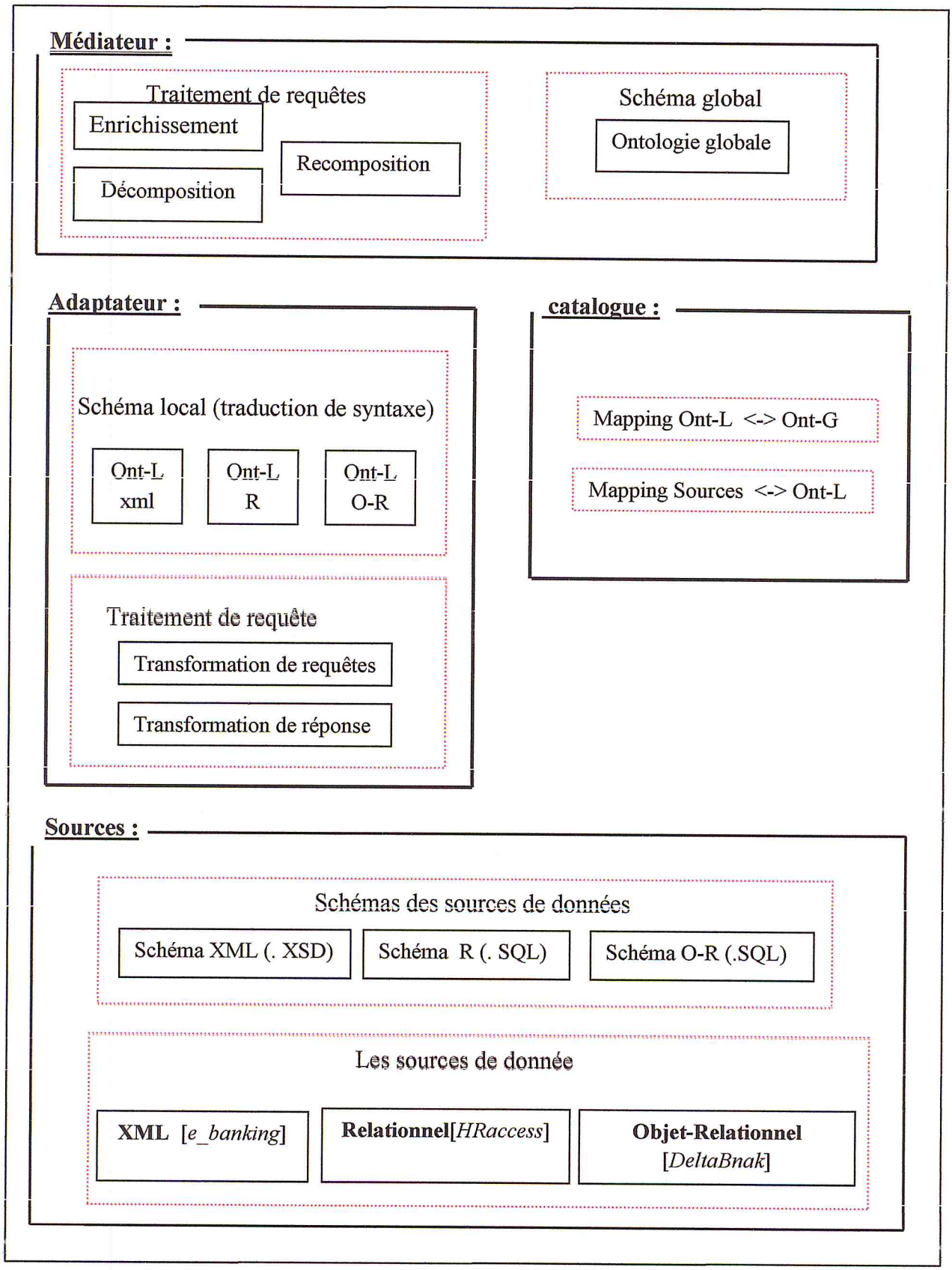
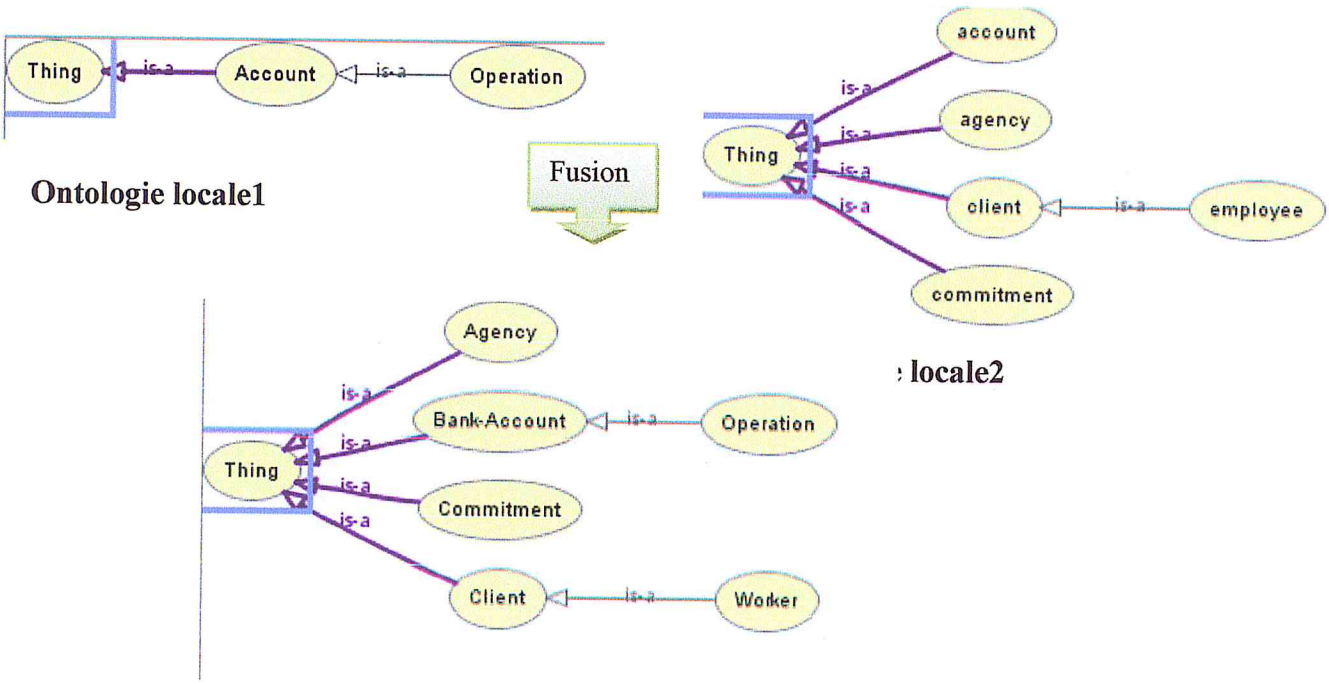


Figure 20 : Architecture détaillée du système

Module01 : « Création du schéma global »

Tout d'abord nous avons trois ontologies locales, cela est une condition nécessaire pour la construction de notre schéma global, et comme nous avons cités précédemment dans la description de notre solution proposée, nous travaillerons avec l'ontologie hybride, c'est-à-dire nous allons fusionner nos ontologies locales en utilisant la méthode de fusion FonEs pour avoir une ontologie globale qui regroupe tous les concepts plus les relations des autres ontologies deux à deux , l'hétérogénéité sémantique sera réglée lors de la fusion, telle que cette méthode est basée sur la combinaison des similarités terminologique et structurelles, et à la fin de cette étape, nous aurons une matrice des mesures de similarité « matrice de correspondances » que nous allons utiliser pour notre fusion .

Exemple : Fusion des deux ontologies



Ontologie Globale

Module02 : « Catalogue »

Le catalogue est un document XML qui est crée lors de la transformation des schémas des Sources de données vers les ontologies locales, il est basée sur le mapping GLAV, tel que pour chaque attribut de la base de données nous faisons correspondre son concept dans l'ontologie locale et aussi un autre lien de ce dernier avec le concept de l'ontologie globale.

Source I (attribut x) <--> OntologieLocale J (concept y)

OntologieLocale J (concept y) <--> OntologieGlobale (concept z)

Exemple :

```
<Concept>
  <BDD>hr</BDD>
  <OntoLocale>salaried</OntoLocale>
  <Properties>
    <Propertie>Login</Propertie>
    <Propertie>name_s</Propertie>
    <Propertie>surname_s</Propertie>
    <Propertie>birth_date</Propertie>
    <Propertie>birth_place</Propertie>
    <Propertie>blood_group</Propertie>
    <Propertie>sex</Propertie>
    <Propertie>date_entered</Propertie>
    <Propertie>exit_date</Propertie>
    <Propertie>status</Propertie>
    <Propertie>adress_s</Propertie>
    <Propertie>Employment_code</Propertie>
  </Properties>
  <OntoGlobale>Worker</OntoGlobale>
</Concept>
<Concept>
  <BDD>delta</BDD>
  <OntoLocale>account</OntoLocale>
  <Properties>
    <Propertie>account_num</Propertie>
    <Propertie>account_type</Propertie>
    <Propertie>ammount</Propertie>
```

Algorithme : Création du schéma global et du catalogue

Entrée : Ontologies locales, Sources de Données
Sortie : Ontologie Globale, Catalogue
Début
Se connecter aux sources de données
Si connexion établite {
    Pour chaque Source de Données
        Lancer la transformation vers OWL
        Remplissage du catalogue
    Si ontologie écrite en OWL alors
        Lancer la Fusion de deux Ontologies
        Relancer la Fusion entre le résultat de la 1ère Fusion et la 3ème ontologie
        Remplissage finale du catalogue
}

Module03 : « Sources de données »

Ce module contient les trois sources de données qui sont exprimées selon trois modèles de données différents (relationnel, objet-relationnel, XML) et comme nous l'avons déjà détaillé dans le 2ème chapitre, nous avons remarqué que toutes les sources sont hétérogènes. Aussi dans ce module, nous avons les schémas de chaque source de données qui sont prêts pour la création des ontologies locales.

Module04 : « Adaptateur »

L'adaptateur cache l'hétérogénéité au médiateur. Il traduit le schéma des sources en termes du schéma global et les requêtes du médiateur en termes compréhensibles par les sources. De plus, l'adaptateur transforme les réponses aux requêtes en des réponses conformes au schéma global du médiateur. Il est assez complexe et ardu à écrire. Il est généralement associé aux sources mais peut se trouver dans le médiateur. L'adaptateur peut être intelligent et donc effectuer des optimisations spécifiques aux sources. Dans notre approche, les adaptateurs représentent l'interface de communication entre les sources et le médiateur.

Sous-Requête en langage local → Requête SQL (modèle relationnel)

Sous-Requête en langage local → Requête SQL3 (modèle objet-relationnel)

Sous-Requête en langage local → Requête XML (modèle XML).

Exemple :

Requête locale : {<**cpt1**> <pp1, pp2>}, {<**cpt2**><pt1, pt2, pt3>}, {<**cpt3**><pp>}

En SQL : Select **t1**. pp1 , **t1**.pp2 ,**t2**.pt1 ,**t2**.pt3

from table1 **t1** , table2 **t2** , table3 **t3**

where **t1**.ID=**t3**.pp and **t3**.pp=**t2**.ID

Tel que : **t1, t2, t3** sont les noms des tables qui correspondent aux noms des concepts des ontologies locales (**cpt1, cpt2, cpt3**), et **pp1, pp2, pp3** sont les attributs des tables qui correspondent aux propriétés des concepts des ontologies locales.

Module05 : « **Traitement de requête** »

Le processus traitement de requête passe par trois phases principales, telle que la première étape est l'enrichissement puis la décomposition et après la recomposition des réponses:

1. Enrichissement d'une requête

Algorithme : Enrichissement d'une requête

Entrée : Requête I

Sortie : Requête E

Début

création d'une liste de requête enrichie

Initilisation : liste_enrichie = requête_intiale

Pour chaque Concept Ci dans la requête I

Trouver l'ensemble des relations Ri de ce concept

Pour chaque relation Ri

Trouver le Rang r

Rechercher dans l'ontologie globale les concepts qui a domain D=Rang r

Pour chaque Concept C

Rechercher dans les concepts de la requête E

Si (le Concept C n'existe pas)

Ajouter une nouvelle clé cpt qui est égale à Domain D à la requête E ;

Fin

2. Décomposition de requête et localisation des sources

Ce module est chargé de la décomposition de la requête globale (de l'utilisateur) en sous requêtes locales.

Formellement, considérons une requête **Q** une conjonction de sous requêtes dont la syntaxe est la suivante :

Requête utilisateur Q : conjonction de sous requêtes Qi => $Q = \wedge_{i=1 \ldots n} Qi$

Exemple :

Un employé qui souhaite savoir les opérations qu'il a effectué sur son compte bancaire et il ne possède rien que son code, alors la requête émise sera comme suit :

Select e-banking .libele, e-banking .date, e-banking .hour
from e-banking.account
where hraccess. Employment_code = '14569'

Comme nous avons remarqué que cette requête sera décomposée ou projetée sur deux sources de données tels que :

Employment_code : se trouve dans la source de donnée relationnelle hraccess.

Libele, date, hour : se trouvent dans le document XML.

Pour décomposer la requête initiale en deux sous requêtes, nous devons parcourir le catalogue qui contient les mapping entre schéma global/Schéma Local

Sous Requête 1: **Select e-banking .libele, e-banking .date, e-banking .hour**
from e-banking.account

Sous Requête 2: **Select employment_code**
from hracces
where hraccess.Employment_code=e-banking.code

3. Recomposition des résultats

Ce module construit la réponse d'une requête globale en utilisant les résultats des requêtes locales envoyés par les adaptateurs. Nous présentons l'algorithme de reconstruction de la réponse d'une sous requête Qi.

Algorithme : Recomposition des sous résultats

Entrée : Un ensemble de sous réponses
Sortie : réponse globale

Début

Pour chaque sous réponse SR **Faire**
Insérer dans la réponse Globale RG, SR

Fait

Fin

Conclusion

Dans ce chapitre nous avons présenté une conception détaillée de notre système de médiation des sources de données. Cette étude conceptuelle nous a permis de mettre en évidence les étapes nécessaires pour la création d'un système d'intégration des sources de données.

La prochaine étape consiste donc en la concrétisation de ce que nous avons proposé. En d'autres termes, la réalisation d'un système de médiation des sources de données tout on présentant les outils que nous avons utilisé pour l'implémenter.

Implémentation et Test

Introduction

A ce niveau nous sommes arrivés à la partie pratique du système, après avoir présenté des différentes étapes pour la conception de notre système de médiation, maintenant il est temps de passer au coté techniques de application1. C'est dans cette partie du mémoire que nous allons présenter notre système en définissant les différents outils dont nous avons eu besoin pour la mise en œuvre de l'application et les différentes interfaces de cette dernière, ainsi vu que nous avons choisis un domaine d'application spécifique nous détaillerons nos trois sources de données avec lesquels nous avons construit et testé notre système.

1. Environnement de travail

1.1. Langage utilisé

Pour la réalisation de notre système nous avons utilisé le langage de programmation Java, sous l'environnement Eclipse.

JAVA

JAVA est le nom d'une technologie mise au point par Sun Microsystems qui permet de produire des logiciels indépendants de toute architecture matérielle. Java est à la fois un langage de programmation et une plateforme d'exécution. Le langage Java a la particularité principale d'être portable sur plusieurs systèmes d'exploitation tels que Windows, Mac OS ou Linux. C'est la plateforme qui garantit la portabilité des applications développées en Java.

Il permet de créer des applications autonomes et de doter les documents html de nouvelles fonctionnalités : animations interactives, applications intégrées, modèles 3D, etc. Ce langage est orienté objet et comprend des éléments spécialement conçus pour la création d'applications multimédia **[S09].**

Nous avons choisit le langage Java pour les raisons suivantes :

- ✓ L'application peut s'exécuter sur n'importe quel système d'exploitation à condition d'avoir la machine virtuelle java installée sur la machine (portabilité);
- ✓ La disponibilité de la documentation et de l'assistance (forums) ;
- ✓ Il existe des API développement en Java dont on a en besoin pour la manipulation de nos ontologies comme Jena.

1.2. Les outils utilisés

Protégé

C'est un logiciel graphique pour la création d'ontologies. Il a été créé par le SMI (Stanford Medical Informatics) à l'université Stanford. Il est très populaire dans le domaine du Web Sémantique et au niveau de la recherche en informatique. Protégé est un éditeur d'Ontologies distribué en open source développé en Java. C'est un éditeur hautement extensible, capable de manipuler des formats très divers, tels que: RDF, RDFS, OWL, etc grâce aux plugins dédiés. Dans le modèle des connaissances de Protégé les ontologies consistent en une hiérarchie de classes ou de concepts représentés sous forme générique qui ont des attributs, qui peuvent eux-mêmes avoir certaines propriétés. L'édition des listes de ces trois types d'objets se fait par l'intermédiaire de l'interface graphique, sans avoir besoin d'exprimer ce que l'on a à spécifier dans un langage formel : il suffit juste de remplir les différents formulaires correspondant à ce que l'on veut spécifier. Aujourd'hui, il regroupe une large communauté d'utilisateurs et bénéficie des toutes dernières avancées en matière de recherche ontologique : compatibilité OWL de référence, services inférentiels, gestion de bases de connaissances, visualisation d'ontologies, etc**[MEZ et nad, 11] .**

FOnES (Fusion d'Ontologies par Enrichissement Sémantique)

C'est un système de construction d'ontologies par fusion des ontologies existantes, il permet de créer une ontologie unique et cohérente en traitant le problème de l'hétérogénéité sémantique d'une manière automatique, le fonctionnement du système est basé sur le calcul d'une similarité sémantique entre les concepts des Ontologies. Il s'appuie sur une combinaison pondérée des méthodes de calcul de similarité existantes et ne prend en charge que les ontologies au format OWL **[MEZ et nad, 11].**

Stylus Studio 2011 XML

Stylus Studio 2011 XML est un environnement de développement avancée de XML (IDE XML) constitué de centaines de puissants outils XML et des plus hautes performances des composants de JAVA et de .NET pour concevoir des applications d'intégration de données. Stylus Studio ajoute de nouvelles fonctionnalités puissantes ce qu'il lui a permis d'être le premier ministre de XML. Les meilleures caractéristiques de Stylus Studio est qu'il travaille avec XML, XQuery, XMLPipeline, XSLT, XSL: FO, EDI, XML Schema/DTD, XPath, XMLet bases de données,XHTML, mappage XML, XMLédition.

1.3. Les SGBD utilisés

MySQL

MYSQL est un serveur de bases de données (Open Source). Il stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble. Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête. Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de données **[S09]**. Nous avons utilisés MYSQL pour la gestion de notre base de données en modèle Relationnel

PostgerSQL

PostgreSQL est un système de gestion de base de données relationnelle et objet (SGBDRO), c'est un descendant Open Source du code original de Berkeley. Il supporte une grande partie du standard SQL tout en offrant de nombreuses fonctionnalités modernes : requêtes complexes, clés étrangères, triggers, vues, intégrité transactionnelle, contrôle des versions concurrentes (MVCC ou Multi Version Concurrency Control) **[S08]**. Nous avons utilisés PostgerSQL pour la gestion de notre base de données en modèle objet relationnel

1.4. Les APIS utilisées

Jena

C'est un ensemble d'outils (une API) permettant de lire et de manipuler des ontologies décrites en OWL et d'y appliquer certains mécanismes d'inférences. Au cours de notre développement, on a utilisé la version Jena 2.6 qui était la version la plus récente. Et qui permettait entre autres : la création et l'extraction de concepts, de propriétés (relations et attributs) sur les concepts ainsi que les restrictions sur les propriétés mais aussi des instances et de leurs insertions dans notre ontologie qui n'est autre qu'un fichier OWL. Et ce à travers la classe OntModel. **[MEZ et nad , 11]**

JDBC

JDBC est une API fournie par JAVA, elle est constituée d'un ensemble d'interfaces de programmation qui permettent au langage JAVA d'accéder à des bases de données par l'intermédiaire du langage SQL. Comme des interpréteurs Java sont disponibles sur la plupart des postes de travail, cela permet d'écrire des applications indépendantes des bases de données.

Le parseur JDOM

La manipulation d'un document XML se fait par le biais d'un parseur. Un parseur est une application dont le rôle est de convertir un flux de balisage en une sortie accessible par un programme. Le parseur vérifie la conformité d'un document à la norme XML, à savoir qu'un document XML doit être bien formé.

Comme parseur nous avons choisis JDOM qui tire ses forces des deux parseurs sax et DOM, JDOM utilise des collections SAX pour analyser les fichiers XML, et utilise DOM pour manipuler les éléments créés par SAX.

JDOM permet donc de construire des documents, de naviguer dans leurs structures et d'ajouter, modifier ou supprimer leurs contenus.

L'utilisation de l'API JDOM dans le traitement de données XML avec java est simple malgré que cette API soit toute jeune et en voie d'être améliorée **[KETT, 11].**

2. Les sources de données

Comme nous avons déjà cité auparavant, quand il s'agit du développement d'un système de médiation, le but principal est de permettre une meilleure productivité au sein des entreprises par le déploiement rapide des systèmes d'information en intégrant plusieurs sources de données sans devoir changer ou reformater les données qui sont stockées. Pour notre cas nous aurons à traiter trois modèles de sources de données dans le domaine financier et plus précisément le secteur banquier afin de faciliter le travail et gagner du temps, notre système permet de capitaliser sur les ressources existantes en terme de machine, données et compétences pour intégrer toutes les sources de données de l'entreprise dans un environnement flexible et facilement maintenable.

Sachant que nous allons traiter trois sources de données financières qui sont :

2.1. La Source des données structurée « HR Access »

HR Access un progiciel de gestion des ressources humaines et de gestion de la paie produit par HR Access Solutions. Nous allons travailler sur une partie de sa base de données qui gère la paie des employés y compris les données individuelles et les absences. Cette base de données est conçu selon le modèle relationnel implémente sous MySQL. Elle regroupe les tables suivantes :

Absences (Num_abs,date_debut,date_fin,motif,*folder_number*)

Degree (Num_deg,Degree,Domain_D,Date_Degree,Institution,*Login*)

Employment (employment_code, Libele,Category,Class,Contract_type)

Official_Documents(Folder_number,Docs_type,Docs_number,Delivrance,End_validity, *Login*)

Pay (Pay_folder, Pay_period, Renumeration, *Folder_number*)

Person_to_prevent (Number, Name, Surname, *Login*)

Phone (Phone_number, Type_phone, *Login*)

Salaried (Login,Surname,Birth_Date,Birth_Place , Blood_Group, Sex, Date_entered, Exit_Date, Status, Adress, *Employment_code*)

2.2. La source de donnée structurée « DeltaBank »

C'est un progiciel de gestion financière créé par l'éditeur français de progiciels bancaires Delta Informatique. Nous allons traiter le coté de gestion des comptes bancaires des employés et des clients externes ainsi les opérations effectués sur ces derniers et aussi la gestion des clients. Cette base de données est conçue selon le modèle objet-relationnel implémente par PostgrSQL , et elle regroupe les tables suivantes :

```
Create table Account
{ Account_num
 Account_type
 Ammount
 Agency_code
}
```

```
Create table Agency
{ Agency_code
 Libele
Adress
 Phone_number
}
```

```
Create table Commiment
{ Commiment_num
 Commiment_type
Start_date
End_date
Intial_commiment
Mortgage
Tva
Interest_rate
account_num
}
```

```
Create table Client
{ Code
 Name
 Surname
 Address,
Phone_num,
Account_num
}
```

```
Create table Employment
{
[Client]
Status
}
```

```
Create type address
{ street
 City
 country
}
```

2.3. La source de données semi structurée « E_banking »

C'est un service fourni par plusieurs banques, il permet aux clients de mener des transactions bancaires à travers l'Internet. Parmi les services qu'il offre :

- Gérer l'ensemble des comptes depuis un ordinateur, 7 jours/7, 24h/24, et aussi souvent qu'on le souhaite.
- Consulter toutes les opérations, créer des fusions de soldes si l'utilisateur n'est pas un professionnel.

Nous nous intéresserons à quelques services de E_banking pour concevoir un document XML qui sera notre 3iéme modèle de données. Voici un fragment du document :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="e_banking">
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="Account"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Account">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="Amount"/>
        <xs:element ref="Account_num"/>
        <xs:element ref="Agency"/>
        <xs:element ref="Code"/>
        <xs:element ref="Currency"/>
        <xs:element ref="Operation"/>
        <xs:element ref="Type"/>
        <xs:element ref="Word_pass"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
  <xs:element name="Account_num" type="xs:NCName"/>
  <xs:element name="Agency" type="xs:NCName"/>
  <xs:element name="Code" type="xs:NCName"/>
  <xs:element name="Currency" type="xs:NCName"/>
  <xs:element name="Operation">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Code_op"/>
        <xs:element ref="Libelle"/>
        <xs:element ref="Date"/>
        <xs:element ref="Amount"/>
        <xs:element ref="Account_t"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Code_op" type="xs:integer"/>
  <xs:element name="Libelle" type="xs:NCName"/>
  <xs:element name="Date" type="xs:NMTOKEN"/>
  <xs:element name="Account_t" type="xs:string"/>
  <xs:element name="Type" type="xs:NCName"/>
  <xs:element name="Word_pass" type="xs:NCName"/>
  <xs:element name="Amount" type="xs:string"/>
</xs:schema>
```

3. Architecture de déploiement

Pour la construction de notre système nous avons utilisé un type d'architecture dit MVC (Modèle Vue Contrôleur) (Figure 21). Une architecture MVC propose le découpage d'une application en architecture 3-tiers selon le schéma suivant :

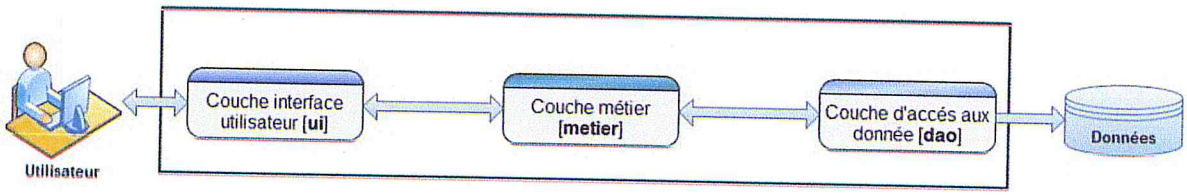


Figure 21 : Architecture MVC.

La couche DAO : S'occupe de l'accès aux données, le plus souvent des données persistantes au sein d'un SGBD. Mais cela peut être aussi des données qui proviennent de capteurs, du réseau, ... etc

La couche Métier : implémente les algorithmes " métier " de l'application. Cette couche est indépendante de toute forme d'interface avec l'utilisateur. Ainsi elle doit être utilisable aussi bien avec une interface console, une interface Web ou une interface de client riche. Elle doit ainsi pouvoir être testée en-dehors de l'interface graphique et notamment avec une interface console. C'est généralement la couche la plus stable de l'architecture. Elle ne change pas si on change l'interface utilisateur ou la façon d'accéder aux données nécessaires au fonctionnement de l'application.

La couche Interface Utilisateur : qui est l'interface (graphique souvent) qui permet à l'utilisateur de piloter l'application et d'en recevoir des informations.

La communication va de la gauche vers la droite :

Les couches métier et DAO sont normalement utilisées via des interfaces Java. Ainsi la couche métier ne connaît de la couche DAO que son ou ses interfaces et ne connaît pas les classes les implémentant. C'est ce qui assure l'indépendance des couches entre-elles : changer l'implémentation de la couche DAO n'a aucune incidence sur la couche métier tant qu'on ne touche pas à la définition de l'interface de la couche dao. Il en est de même entre les couches interface utilisateur et métier.

4. Présentation de l'application

Nous allons à présent présenter les interfaces de notre système. Pour accéder à la création du système, l'utilisateur doit s'authentifier avec un mot d'utilisateur et un mot de passe, puis la fenêtre de création (figure 22) s'affiche, l'utilisateur pourra lancer la création du système, pour ce faire il doit importer les sources de données (figure 23).

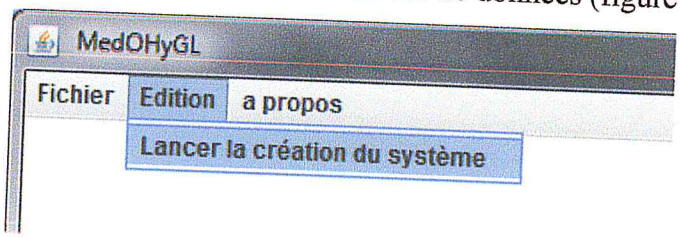


Figure 22 : Fenêtre « Lancement de la création du système »

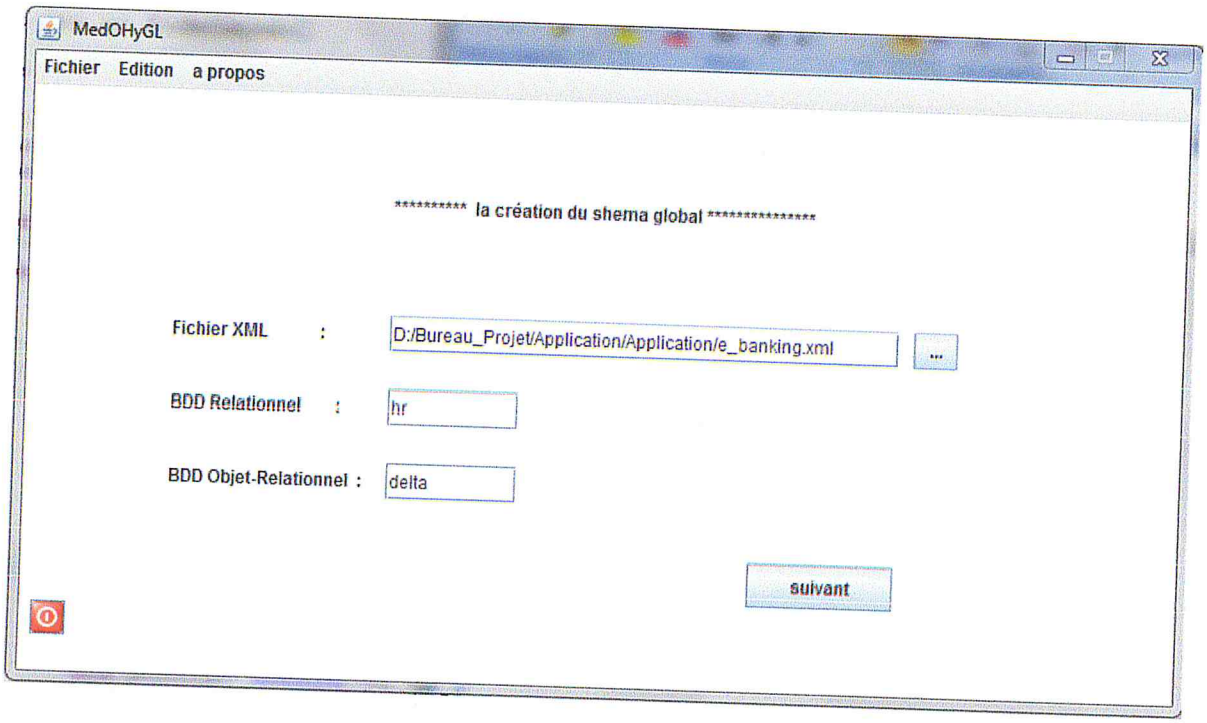


Figure 23 : «Importation des sources »

Après avoir lancé la création, le système vérifie la connexion des bases de données, puis il arrive à l'étape de la création des ontologies locales après il les fusionne et crée le catalogue (figure 24).

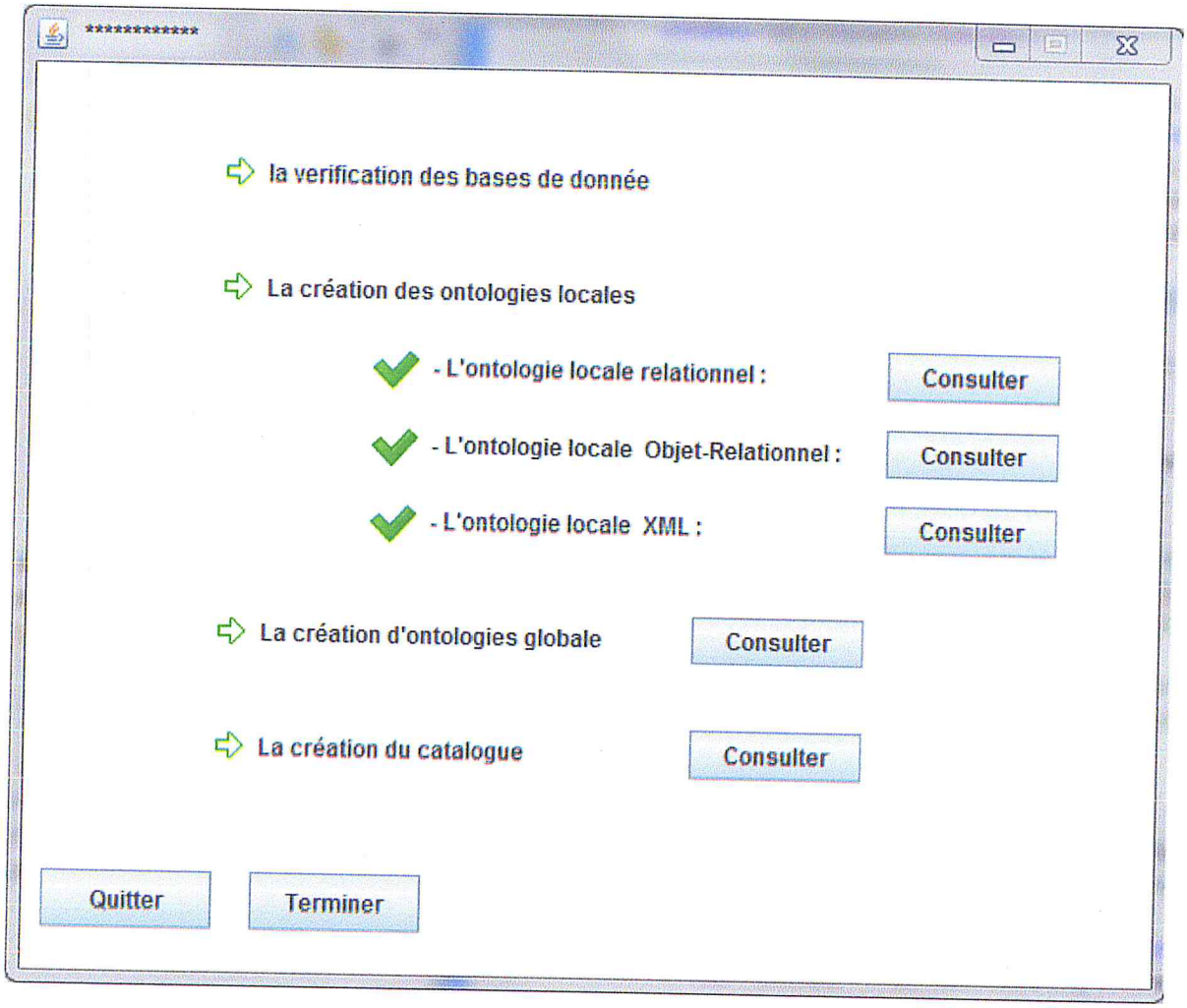


Figure 24 : « Processus de la création du système »

Comme il peut visualiser et consulter les ontologies (figure 25)

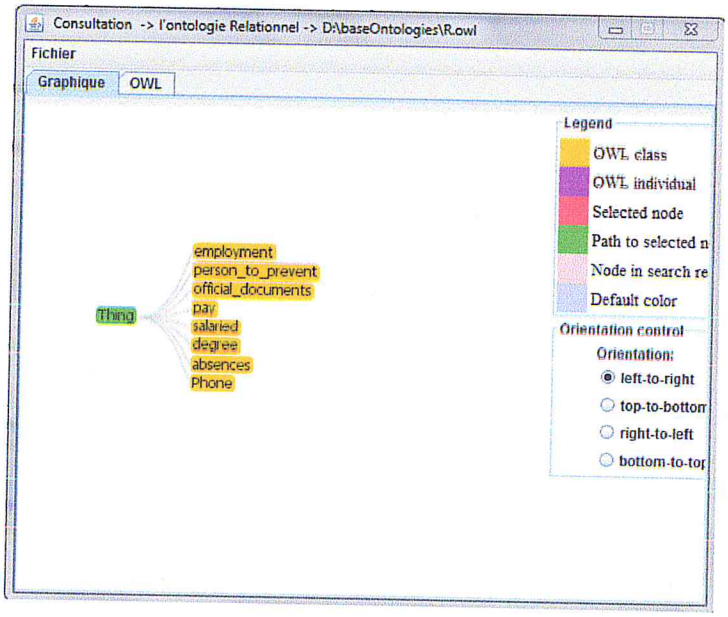


Figure 25 : « La consultation des ontologies »

Maintenant pour utiliser le système de médiation, un simple utilisateur dés qu'il exécute l'application, il aura le choix entre choisir une des requêtes déjà préparés, ou poser sa propre requête (figure 26).

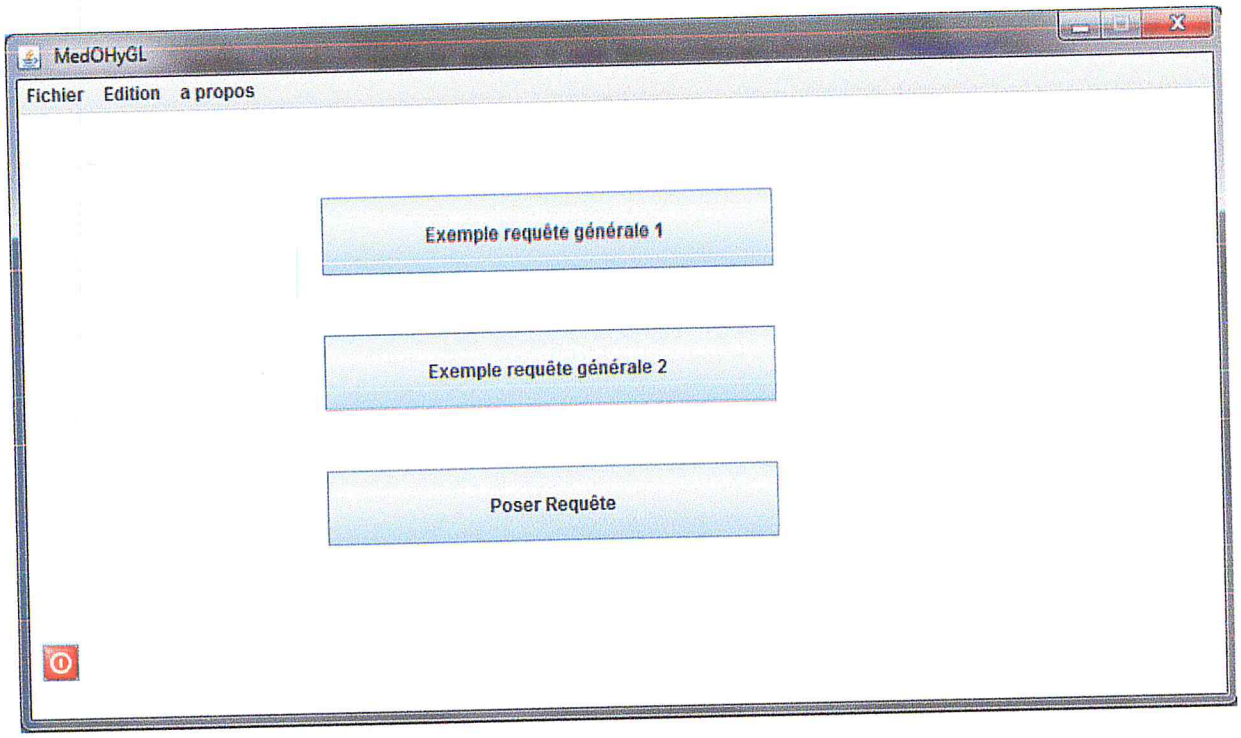


Figure 26 : « Poser requête »

Quand il pose sa requête, il choisit un ou plusieurs concepts avec leurs propriétés comme illustré dans les figures 28 et 27 :

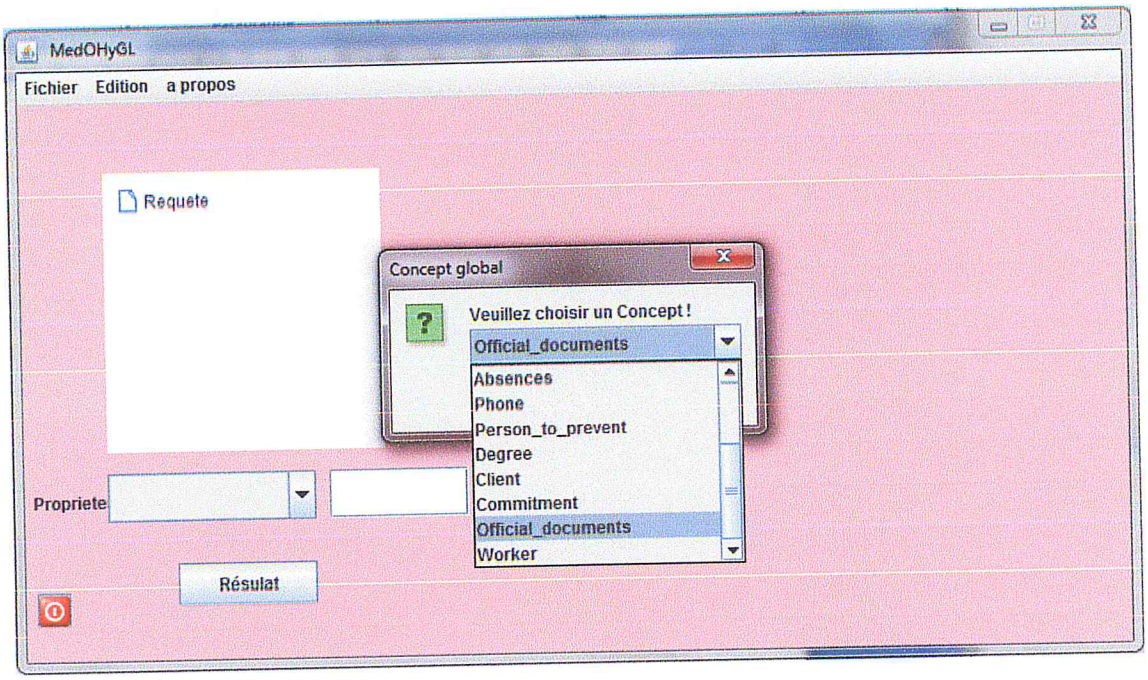


Figure 27 : « Liste des concepts »

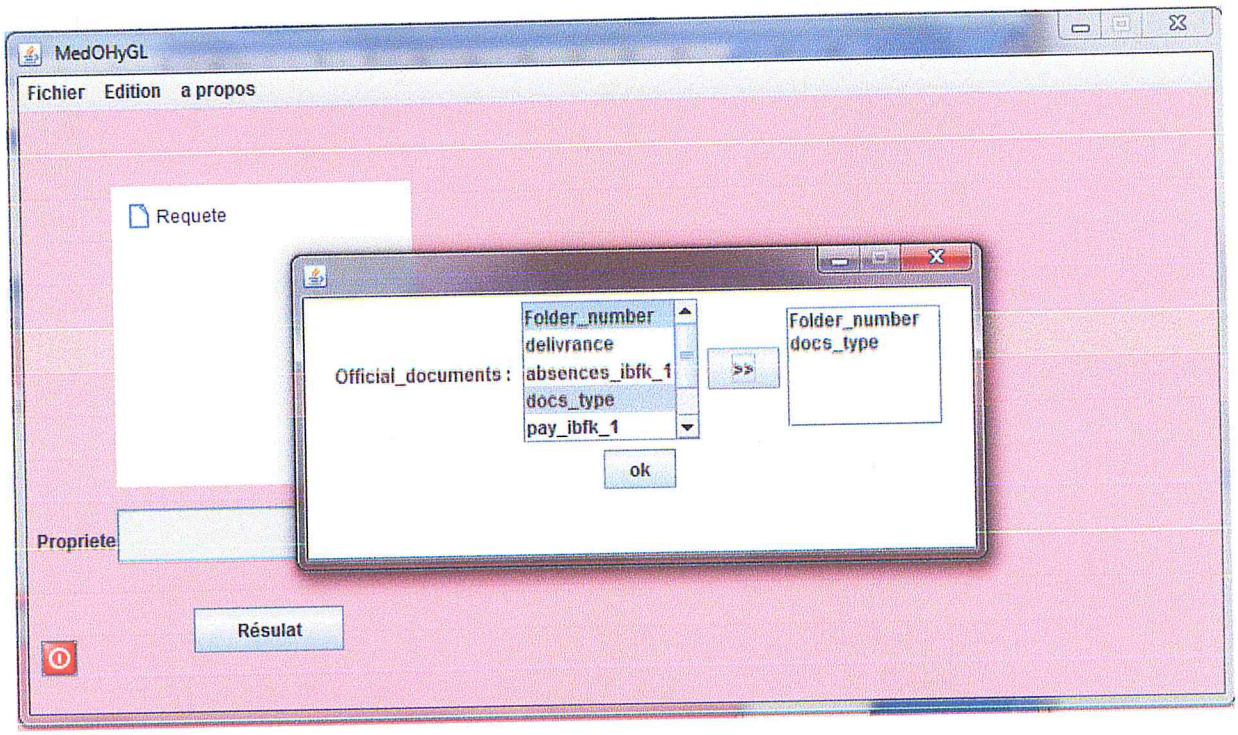


Figure 28 : « Liste des propriétés d'un concept »

A ce niveau, l'utilisateur verra sa requête, s'il veut ajouter d'autres concepts, aussi s'il veut mettre une condition comme dans l'exemple suivant il a précisé son numéro de dossier, puis il clique sur le résultat figure 29 :

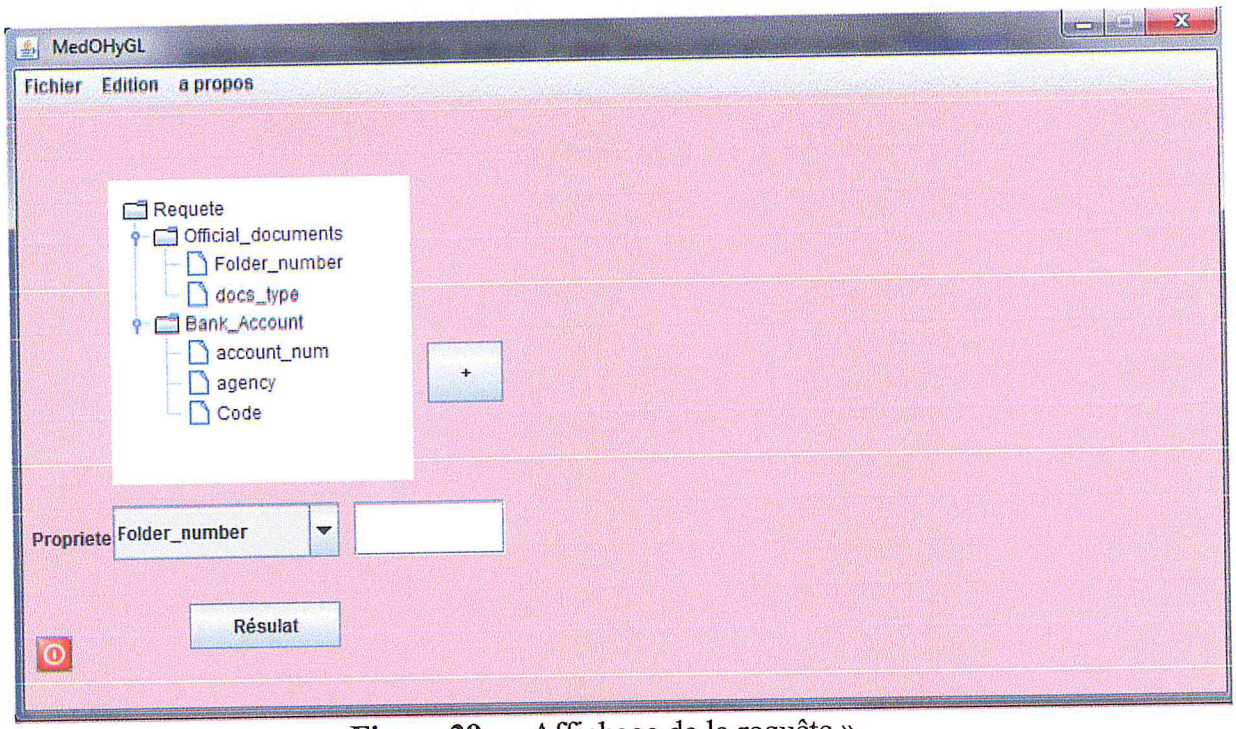


Figure 29 : « Affichage de la requête »

Sur cette interface, nous trouverons le résultat de la requête et le détail des différentes étapes de la requêtes depuis l'enrichissement puis la décomposition , figure 30 :

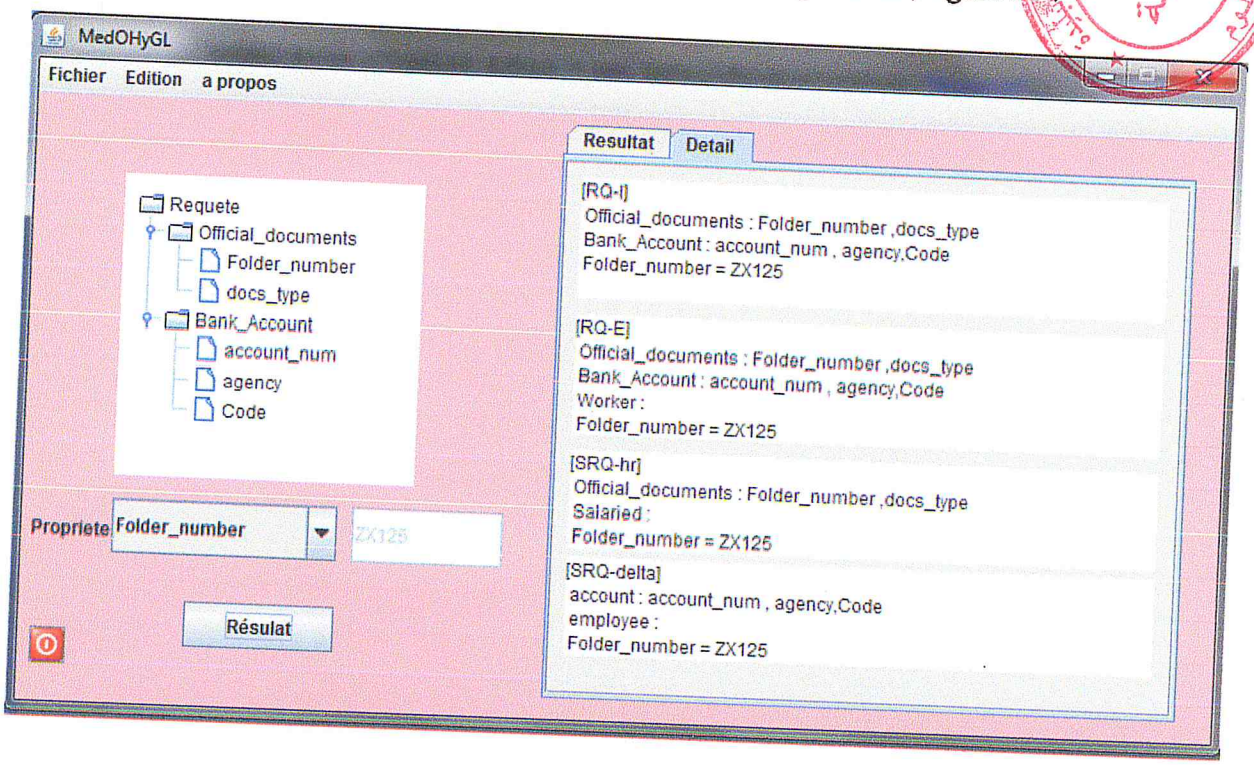
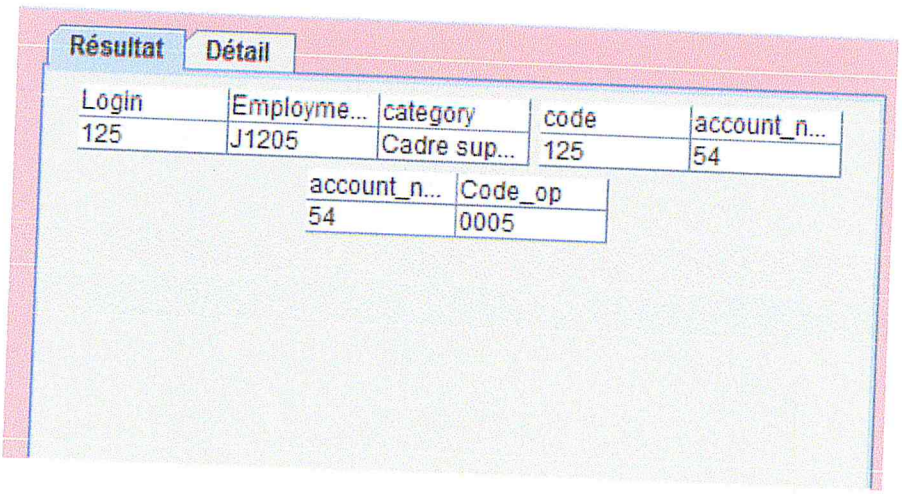


Figure 30 : « Le détail du traitement de requête »



5. Test du système :

Dans cette partie du chapitre, nous montrons les résultats de la création du système, nous allons prendre trois ontologies, nous commençons d'abord par fusionner ontologies ontologieOR et ontologieXML), puis à fusionner le résultat de la première fusion avec l'ontologieR, et nous montrons aussi le remplissage du catalogue tel qu'il se remplisse automatiquement après chaque transformation et après la fusion. Aprés nous montrons le traitement de requête.

OntologieR (HRaccess) :

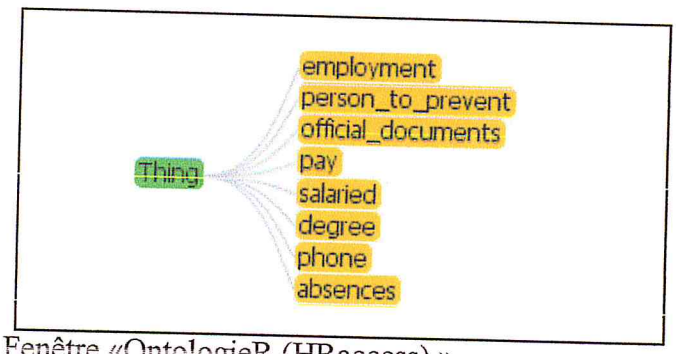


Figure 31: Fenêtre «OntologieR (HRaccess) »

OntologieOR (DeltaBank) :

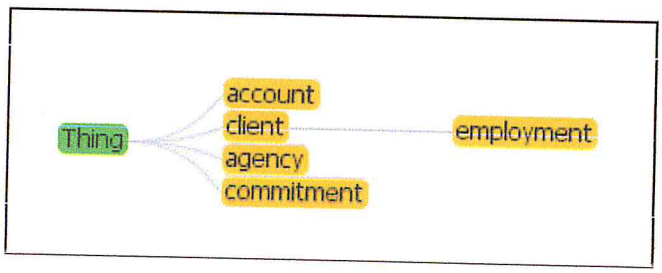


Figure 32: Fenêtre «OntologieOR (DeltaBank) »

OntologieXML (e_banking) :

Figure 33: Fenêtre «OntologieXML (E_banking) »

OntologieGlobale :

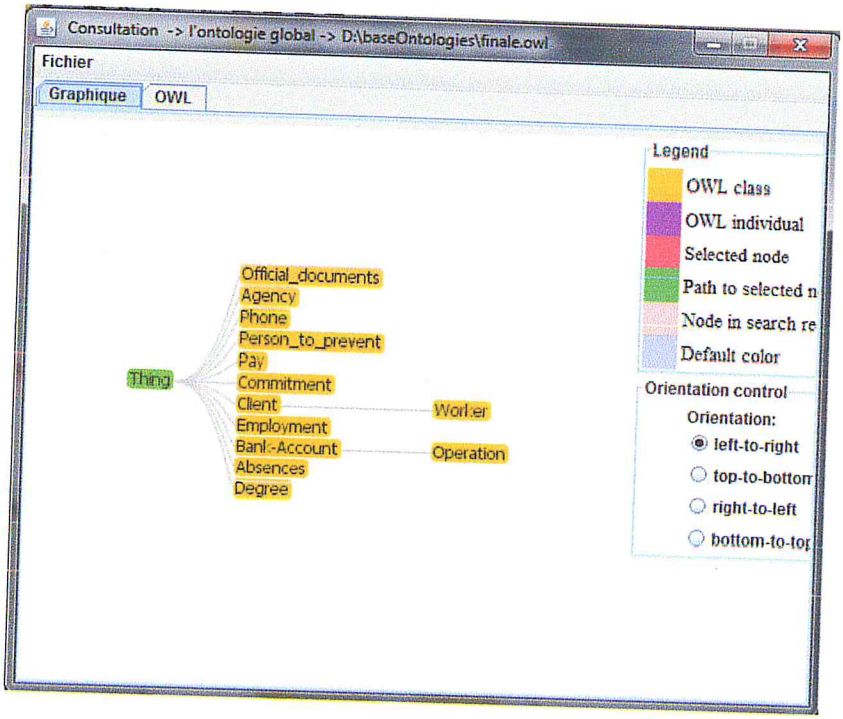


Figure 34: Fenêtre «Ontologie globale»

Catalogue :

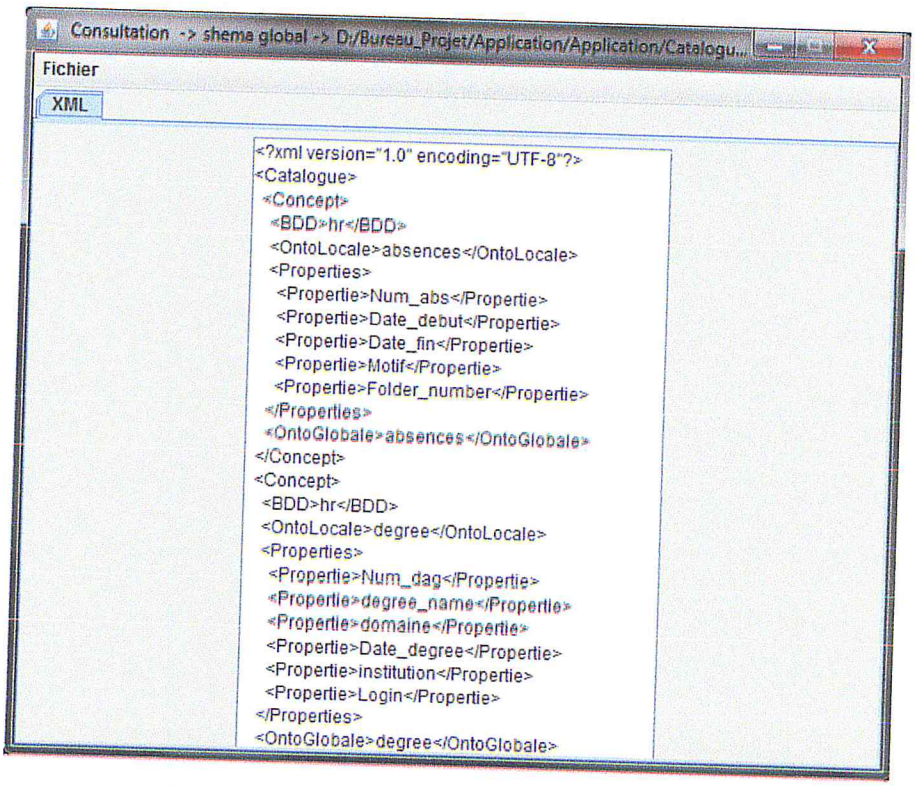


Figure 35: Fenêtre «Catalogue »

Nous présentons à ce niveau ,le détail du traitement de requête :

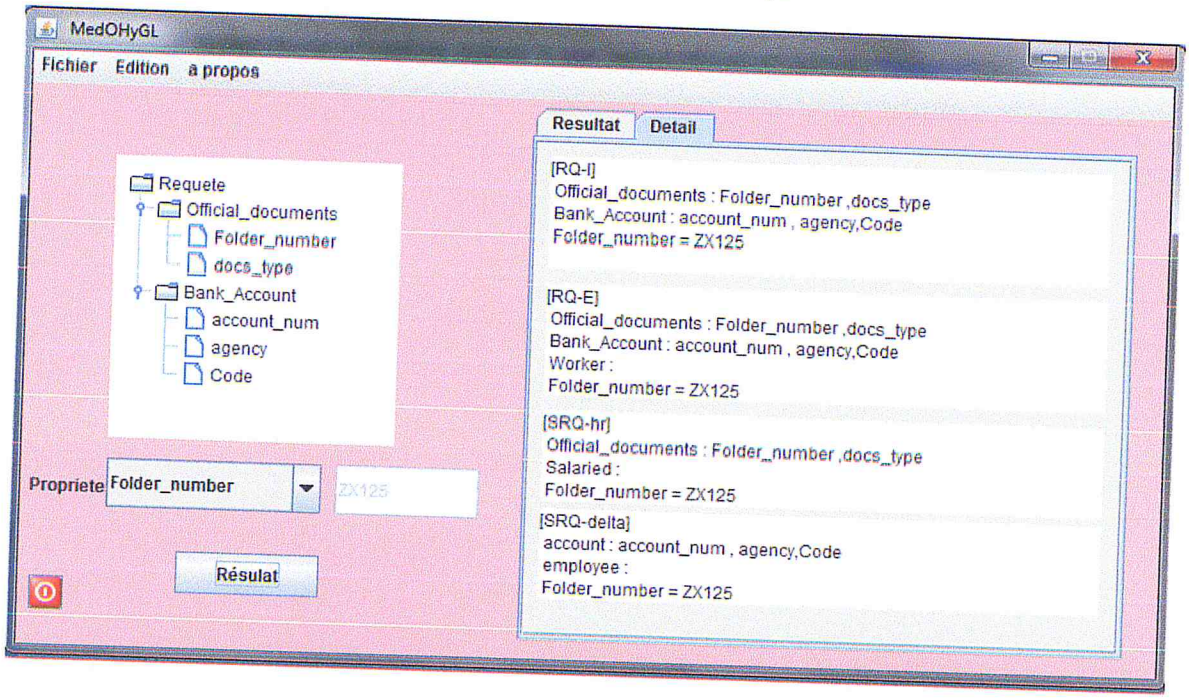


Figure 36: Fenêtre «traitement de requête »

Conclusion

Dans ce chapitre nous avons présenté l'implémentation de notre système, telle que nous avons définis les différents outils et SGBD utilisés et nous avons pu montré de manière plus concrète que travailler avec des sources de données spécifiques permet de prendre conscience de l'hétérogénéité sémantique, et à la fin nous avons présenté les interfaces de notre application.

Conclusion générale

Conclusion générale

La masse énorme d'information disponibles sur des sources de données distribuées et hétérogènes, nécessite une politique ou un plan de recherche, de sélection et d'analyse, de plus en plus performants, pour permettre à l'utilisateur de localiser et extraire précisément l'information désirée d'une manière simple et efficace.

Dans ce mémoire, nous avons réalisé un système de médiation basé sur les ontologies. Ce système repose sur l'architecture hybride, utilisant une ontologie globale pour le médiateur et des ontologies locales au niveau des sources. La plupart des approches existantes de construction des ontologies ne prennent pas en charge le contexte de médiation. Le peu d'approches qui prennent en compte le contexte de l'intégration prônent une démarche descendante, c'est-à-dire la construction de l'ontologie globale puis celles des ontologies locales. L'approche que nous préconisons consiste à créer l'ontologie globale en appliquant la fusion et ce, à partir des ontologies locales, ce qui facilite et améliore la résolution de l'hétérogénéité sémantique entre les ontologies de source.

L'architecture hybride est très générique car elle est sujette à plusieurs modélisations structurelles. Les modèles GAV (Global As View), LAV (Local As View) et GLAV (Generalized Local As View) peuvent y être appliqués. Le Modèle GLAV, qui est bien adapté pour la prise en charge des sources très hétérogènes, est simple à mettre en œuvre.

Notre travail se résume en l'utilisation d'un langage qui garantit le traitement de la requête qui est le langage OWL, en permettant l'expression de requêtes en termes des ontologies globales et locales. La seconde étape après la définition du langage a été d'utiliser un outil de fusion FOnES afin d'avoir une ontologie globale. La mesure de similarité qu'adopte FOnES étant composite (implique à la fois : similarités terminologiques aussi bien lexicales que syntaxiques, similarités structurelles et bien sûr sémantiques), elle lui confère un haut niveau de pertinence dans le calcul des correspondances entre les Ontologies initiales. Ce qui permet de produire une fusion riche et très fidèle (sémantiquement) au domaine auquel elle doit se rapporter. Nous avons présenté un langage permettant le traitement de requêtes dans GLAV, et un algorithme de réécriture garantissant la cohérence des résultats obtenus.

A l'issue de ce travail, nous croyons pouvoir affirmer que l'intégration sémantique d'applications d'entreprise constitue un problème dur et complexe qui nécessite à notre sens des services d'intégration basés sur la découverte et la médiation sémantique

Perspectives :

La recherche actuelle concernant la médiation de données s'oriente vers les systèmes de médiation de données, langages et optimisation de requêtes, cohérence (transactions, synchronisation, réplication), sécurité et confidentialité, intégration physique (entrepôts de donnée s et bases de données multidimensionnelles), grilles de données, interopérabilité et fédération de sources de données, systèmes coopératifs, performance (indexation, opérateurs de recherche, parallélisme, bancs d'essai, etc).

Dans le cadre d'améliorer notre système de médiation, nous proposons les perspectives suivants :

- Valider notre approche sur un nombre plus important de sources ;
- Prendre en considération la diversité des types de sources ;
- Considérer des requêtes plus complexes ;
- Optimisation de requêtes selon le temps de réponse ;
- Distribution des sources de données dans un système de médiation .

Bibliographie

[BAK, 06] : BAKHTOUCHI A, Etude et proposition d'une architecture de médiation entre sources de données hétérogènes, (2006). 1

[IZZ, 06] : IZZA S, Intégration des systèmes d'information industriels (Une approche flexible basée sur les services sémantiques), (2006).

[MEH et JAO, 10] : JAOUAF M et MEHENNI N, Génération automatique des requêtes de médiation dans un contexte relationnel, (2006).

[BOU, 07] : BOUSSIS A, Intégration des sources de données à base ontologique dans un environnement P2P, (2007).

[KAD, 06] : KADRI S, Interopérabilité des bases de données hétérogènes et réparties, (2007).

[DAN, 08] : DANG NGOC T, Intégration de données hétérogènes distribuées, (2008)

[HAC et rey, 02] : HACID M et REYNAUD C, L'intégration de sources de données, (2002)

[BOU et deb, 07] : BOUMGHAR F.O et DEBIANE S, L'imagerie Médicale Dans une Base De Données Distribuée Multimédia Sous Oracle 9i, (2007).

[DJA, 03] : DJAGHLOUL Y, Intégration des ressources Web dans un environnement P2P, basée sur les ontologies et la gestion de la confiance, (2003).

[LUM, 08]: LUMINEAU N, Ontology mapping in *P2P* networks, (2008)

[DIA, 06] : DIALLO G, Une Architecture à Base d'Ontologies pour la Gestion Unifiée des Données Structurées et non Structurées,(2006).

[BAR, 05] : BARDIN Y, Service de Médiation : Une proposition pour l'accès aux services en milieu hétérogène, (2005).

[DAO et CHI, 09] : DAOUADJI H et CHIKHI I, Génération d'une ontologie OWL à partir du méta-modèle SPEM, (2009).

[GAR, 02] : GARDARIN G, Base de données, (2002)

[BRO et bru, 10] : BROUARD F et BRUCHEZ R, Les bases de données et SQL, (2010).

[GRI, 06] : GRIN R, Modèle objet – relationnel SQL99, (2006)

[LAC, 05] : LACOT X, Introduction à OWL, un langage XML d'ontologies Web, (2005).

[DAN, 03] : DANG NGOC T, Fédération de données semi- structurées avec XML, (2003).

[TEL, 07] : TELGHMATI S, Validation des requêtes de mise à jour dans les bases de données XML natives, (2007)

[AMA, 03] : AMANN B, Bases de données avancées, (2003)

[MEZ et nad, 11] : MEZZI M et NADJI K, Construction d'un système de Fusion des ontolgies, (2011)

Webgraphie

[S01] http://www-poleia.lip6.fr/~doucet/CoursBDIA/ 6

[S02] http://www2.lirmm.fr/~libourel/MIFPRU/M2/ Mediation 7

[S03] http://revue-eti.net/document.php?id=1166#tocto6

[S04] http://odile.papini.perso.esil.univmed.fr/sources/BD/cours-BD-7

[S05] http://www.isnetne.ch/isnet15/

[S06] http://www.hec.unil.ch/gcampono/teaching/BDA/Poly/POLY%20CH10%20

[S07] http://www.cours.polymtl.ca/inf6701/

[S08] http://fr.wikipedia.org/wiki/PostgreSQL

[S09] http://dev.mysql.com/doc/refman/5.0/fr/what-is.html

[S10] http://www.6ma.fr/lexique/informatique/java

[S11] http://info.arqendra.net/UML_cours

[S12] http://fr.wikipedia.org/wiki/Architecture_de_m%C3%A9diation