F.S.D..... N°D'ordre.....

Université Saâd DAHLAB, Blida

Faculté des sciences.

Département informatique.

Présenté par :

ECHIKR Fatima-Zohra

NADIR Imene

Promotrice :

M.FAREH

En vue d'obtenir le diplôme de master

Domaine : Mathématique et informatique

Filière : Informatique

Spécialité : Informatique

Option : ingénierie de logiciel

Sujet : INTEGRATION DES SOURCES DE CONNAISSANCE HETEROGENES: OWL et GRAPHE CONCEPTUEL, PAR L'APPROCHE MEDIATEUR

Soutenu le :

M.^lle	N.Bousta	**Président**
M.^lle	Amena Khadidja.	**Rapporteur**
M.^lle	Cheutir Affaf.	**Examinateur**

Promotion

2010/2011

Sommaire

Résume
Liste des figures
Liste des abréviations
Introduction Générale

Chapitre I :l'intégration des données hétérogènes par l'approche médiateur
1Introduction……..1
2L'intégration des Données……1
2.1Problème de l'intégration de données……..1
2.2Taxonomie de conflits d'intégration……5
2.3Systèmes d'intégration de données…… 11
2.3.1Définition et Composante…… 11
2.3.2Processus d'intégration……..12
2.3.3Tâche d'un système d'intégration ……13
2.4Panorama des approches d'intégration de données ……..14
2.4.1L'intégration structurelle ……14
2.4.2L'intégration sémantique ……18
3L'approche médiateur……..19
3.1Présentation générale……19
3.2Type de médiation……20
3.3Les composants d'un système de médiation…… 20
3.4Architecture d'un système de médiation……..21
3.5Les couches de l'architecture d'un système médiateur……23
3.6Les approches d'intégration dans la médiation……23
3.6.1L'approche GAV……24
3.6.2L'approche LAV……24
3.6.3L'approche BAV……25
3.7L'aide apporté par les systèmes de médiation……..26
4Conclusion……27

Chapitre II Modèles de représentation de connaissance OWL, Graphe Conceptuel
1Introduction……29
2L'ontologie……29
2.1Les composants d'une ontologie……31
2.2Rôles de l'ontologie……..31
2.3Classification des ontologies……32
3L'OWL……34
3.1Généralité……34
3.2Les espèces de l'OWL……..34
3.3Comparaison de OWL avec RDF et RDFS……35
3.4La structure des ontologies OWL……35
3.4.1Espaces de nommage……36

3.4.2En-têtes d'une ontologie.....37
3.4.3Eléments du langage.....38
4Les graphes conceptuels.....45
4.1Généralités.....45
4.2Graphe conceptuel.....47
4.3Concepts.....49
4.4Relations conceptuelles.....52
4.4.1Notation.....52
4.5Référents.....55
5Conclusion.....55

Chapitre III expression des besoins et analyse
1Introduction.....57
2Cycle de vie suivi.....57
3Les fonctionnalités générales du système.....58
4Expression des besoins.....59
4.1Diagrammes des cas d'utilisations.....59
4.2Diagrammes de séquence.....67
4.3Diagrammes d'activités.....71
5Conclusion.....76

Chapitre IV conception
1Introduction.....78
2Diagramme de classes.....78
2.1Description des classes existantes.....79
3L'architecture générale du système.....81
3.1Description des modules de l'architecture.....82
4Conclusion.....88
Chapitre V Implémentation et test

1Introduction.....91
2Le langage de programmation choisi.....91
3Les outils utilisés.....91
3.1Protégé.....91
3.2CoGui.....92
3.3SPARQL.....93
3.4Jena.....95
4Présentation du système MedGav.....96
5Conclusion.....100
Conclusion Générale
Annexe A
Annexe B
Annexe C

REMERCIEMENTS

Nous ne saurions remercier assez notre promotrice Madame M.FAREH, pour nous avoir proposé ce sujet et avoir dirigé nos travaux, pour sa constante disponibilité, pour les conseils qu'elle n'a cessé de nous prodiguer et surtout pour ses qualités humaines et scientifiques, son encadrement privilégié, sa patience et ses encouragements.
Merci de tout cœur.
Que Monsieur M.BALA, trouve ici l'expression de notre profonde reconnaissance d'avoir accepter de nous aidé et de sa disponibilité.
Tous nos remerciements accompagnés de nos gratitudes vont aux membres du jury.

Dédicace

Je dédie ce mémoire à mes parents, mon frère et ma sœur.

A ma tante badra, mon amie et mon binôme imene, halla, ibtissem et hadjer.

A mon professeur d'anglais et mon amie Hassini Hadjer.

ECHIKR Fatima_Zohra

DEDICACES

Le parcours d'une vie est jalonné d'opportunités qui dépendent de nous, mais également des personnes qu'il nous a été donné de rencontrer : des personnes qui nous guident, qui nous conseillent et qui nous font confiance, j'ai eu de la chance de rencontrer quelques uns d'entre eux et à qui je tiens à dédié ce mémoire :

Mes chers et adorables Parents : Pour votre amour, Votre affection et Votre sacrifice consentis, Pour votre soutien indéfectible, bienveillance et Vos encouragements, Que dieu bienveillant vous garde à mon coté

Mes chers frères : Riad et Sofiane pour votre aide et amour inconditionnel

A ma belle-sœur Fatima-Zohra.

A ma sœur de cœur: Dalila

A mes Amies Fatima-Zohra, Ibtissem, Halla ,hadjer, , Sabrina.

A tout mes professeurs pour l'effort et le dévouement pendant tout mon parcourt d'études.

A tous ceux que j'aime

NADIR Imene

Résumé

Via le développement exponentiel des masses des sources de données utilisées il est devenu difficile de retrouver l'information pertinente désirée par un utilisateur et de les exploiter de façon transparente, ces données sont représentées et stockées dans une multitude de sources de données et cela de façon très hétérogène, d'ou la nécessité d'offrir un système d'intégrant des sources de données hétérogènes.

Nous nous intéressons par l'intégration de source de connaissances hétérogènes représentées par le langage OWL et les Graphe Conceptuel par l'approche médiateur. Nous réalisons un système d'intégration de source de connaissances hétérogènes. Ce système permettra l'intégration de ces sources en se basant sur l'approche GAV. Cette dernière est basée sur la notion de schéma global crée à partir des schémas locaux des sources. Le schéma global est représenté par un modèle pivot pour notre système, le modèle pivot que nous avons choisi est le langage OWL ce dernier est connu par sa puissance d'expression. Notre système a pour but de créer un schéma global qui permettra l'intégration des sources hétérogènes et qui offre a l'utilisateur le droit d'effectue des requêtes et de recevoir des résultats homogènes et transparents qui lui donne l'impression d'interroger une seule source en lui cachant l'hétérogénéité syntaxique et sémantique.

Mots clés: médiation; intégration; ontologie; OWL; Graphes conceptuels ; GAV.

Abstract

Via the exponential development of the masses of data sources it has become difficult to find relevant information desired by a user and exploits

them in a transparent manner, these data are represented and stored in multiple data sources and do so very heterogeneous, hence the need for a system of integrating heterogeneous data sources.

We are interested in integrating heterogeneous source of knowledge represented in OWL and the conceptual graph approach for the mediator. We create a system for integrating heterogeneous sources of knowledge. This system will allow the integration of these sources based on the GAV approach. The latter is based on the notion of global schema created from patterns of local sources. The overall pattern is represented by a model for our system pivot, pivot the model we have chosen is the OWL it is known by his power of expression. Our system is designed to create an overall pattern that will allow the integration of heterogeneous sources and offer USER the right to make requests and receive consistent results and transparent which gives the impression of a single query source by hiding the syntactic and semantic heterogeneity.

Keywords: mediation, integration, ontology, OWL, Conceptual Graphs,GAV.

ملخص

الهدف من هذه المذكرة هو انجاز برنامج دمج للمعلومات الغير متجانسة و ذلك باعتماد منهجية الوساطة المبنية على المقاربة GAV المرتكزة على مبدأ المخطط الكلي انطلاقا من المخططات الأصلية لمصادر المعرفة. هذا البرنامج يسمح للمستخدم بالبحث في مصادر للمعرفة غير متجانسة بالحصول على أجوبة موحدة ومتناسقة

الكلمات الرئيسية: الوساطة, الدمج, الأنتولوجيا,GAV

Liste des figures

Figure 1	Exemple de sources hétérogènes	09
Figure 2	Système d'intégration d'information	12
Figure 3	La correspondance entre le schéma global et les schémas locaux	17
Figure 4	Architecture d'un système médiateur	22
Figure 5	Typologie et classification des ontologies	33
Figure 6	declaration de l'espace de nommage de l'ontologie OWL	37
Figure 7	En-têtes d'une ontologie OWL	38
Figure 8	description directe de la classe	38
Figure 9	énumération des individus composant une classe	39
Figure10	descriptions par intersection, union ou complémentaire d'une classe	40
Figure11	déclaration d'une sous classe	40
Figure12	déclaration d'un fait	41
Figure13	déclaration d'un fait anonyme	42
Figure14	déclaration d'un synonyme d'un fait	42
Figure15	définition d'une propriété d'objet	43
Figure16	définition d'une propriété entre deux classes	43
Figure17	définition d'une propriété type de donnée.	44
Figure18	définition d'une hiérarchie de propriétés	44
Figure19	caractériqtiques d'une propriété	45
Figure20	exemple de graphe conceptuel	47
Figure21	un exemple de relation entre concepts concepts	53
Figure22	exemple de relation n-aire	54
Figure23	Modèle du cycle de vie en cascade	58
Figure24	diagramme de cas d'utilisation choix d'ontologies	60

Figure25	diagramme de cas d'utilisation annotation de concepts	61
Figure26	diagramme de cas d'utilisation annotation de relations	62
Figure27	diagramme de cas d'utilisation création du schéma global	63
Figure28	diagramme de cas d'utilisation lancement d'une requête	65
Figure29	diagramme de cas d'utilisation fermeture du système	67
Figure30	diagramme de séquence annotation des concepts	68
Figure31	diagramme de séquence annotation des relations	68
Figure32	diagramme de séquence de lancement d'une requête	69
Figure33	diagramme de séquence récupération des sous résultats	70
Figure34	diagramme de séquence fusionner les sous résultats	70
Figure35	diagramme d'activité annotation des concepts	71
Figure36	diagramme d'activité annotation des relations	72
Figure37	diagramme d'activité création du schéma global	73
Figure38	diagramme d'activité lancement d'une requête	74
Figure39	diagramme d'activité traitement d'une requête	75
Figure40	diagramme de classes	79
Figure41	L'architecture globale du système	82
Figure42	Modules du médiateur.	84
Figure43	Table concepts globaux	87
Figure44	Table concepts locaux	87
Figure45	Table correspondance	87
Figure46	Modules de L'adaptateur OWL	88
Figure47	Modules de l'adaptateur GC.	89
Figure48	l'ontologie OWL créer par l'éditeur protégé	92
Figure49	ontologie créer par l'outil CoGui	93

Liste des tableaux

Figure50	requête d'extraction de classes et de relation à partir de l'ontologie GC	94
Figure51	exemple de résultat de la requête SPARQL sous format d'URIs	94
Figure52	la méthode getClassesOWL qui permet d'extraire les concepts OWL	95
Figure53	résultat de la méthode getClassesOWL	95
Figure54	authentification des utilisateurs	96
Figure55	interface de choix d'ontologie	96
Figure56	fenêtre d'annotation de concepts	97
Figure57	fenêtre d'annotation de relations	97
Figure58	fenêtre d'annulation d'annotation de concepts	98
Figure59	Ontologie Globale générée par le médiateur.	98
Figure60	Fenêtre de requête.	99
Figure61	Fenêtre de résultat global.	99
Figure62	Fenêtre de résultats locaux.	100

Tableau 1	La table « Catalogue_produit » du système	15
Tableau 2	La table « Tprd » du système	16
Tableau3	Description du cas d'utilisation choix d'ontologies	61
Tableau4	Description du cas d'utilisation annotation de concepts.	62
Tableau5	Description du cas d'utilisation annotation de relation	63
Tableau6	Description du cas d'utilisation création du schéma global	64
Tableau7	Description du cas d'utilisation lancement d'une requête	66
Tableau8	Description du cas d'utilisation fermeture du système	67
Tableau9	Description de la classe Médiateur	79
Tableau10	Description de la classe ConceptG	80
Tableau11	Description de la classe ConceptL	80
Tableau12	Description de la classe Concept	80
Tableau13	Description de la classe SourceConnL	80
Tableau14	Description de la classe SourceOWL	80

Tableau15 Description des classes 80
Tableau16 Description de la classe Adaptateur 80
Tableau17 Description de la classe AdaptateurGC 81
Tableau18 Description de la classe AdaptateurOWL 81
Tableau19 Description de la classe Requete 81
Tableau20 Description de la classe SReqOWL 81
Tableau21 Description de la classe SReqOWL 81

Liste des abréviations

OWL	Ontology Web Language
W3C	Consortium World Wide Web
XML	eXtensible Markup Language
RDF	Resource Description Framework
SGBD	Système de gestion de bases de données
GAV	Global As Views
LAV	Local As Views
BAV	Both-As-View
GC	Graphe Conceptuel
SQL	Structured Query Language
OQL	Object Query Language
XQuery	XML Query
HTTP	Hypertext Transfer Protocol
SI	Système d'information
URI	Uniform Resource Identifier

Introduction générale

L'intégration de l'information pose le problème de combiner et de questionner des données de plusieurs sources autonomes et hétérogènes d'une manière homogène. Un grand nombre d'applications ont été développées et de sources d'informations créées tout au long des décennies de développement de systèmes, que ce soit sur le Web ou dans des entreprises de toutes tailles. L'intégration de plusieurs sources locales dans un cadre unique et global est la principale réponse de la communauté informatique pour satisfaire ce besoin.

Plusieurs technologies ont été développées à cette fin dans le domaine de la recherches aussi bien que dans la pratique. Les premières approches d'intégration de ces sources pour les faire coopérer ont été réalisées dans le cadre de système de bases de données (BDs) relationnelles, objets/relationnelles ou objets.

Cependant, dans le contexte du Web, les sources de données ont pour but principal de fournir des capacités d'interrogation et non d'exportation des données. Ce contexte nous oblige à repenser la façon dont nous devons intégrer ces différentes sources d'informations pour connaître au plus tôt la sémantique des données mémorisées.

Connaître cette sémantique, c'est pouvoir « filtrer » les sources qui sont les plus adéquates pour répondre à une demande d'information. Pour ce faire une approche d'intégration de sources hétérogènes a été mise en œuvre, qui consiste à garder les sources d'informations telles qu'elles sont et de les interroger sans les charger et les stocker.

Comment peut-on intégrer cette sémantique ? Comment peut-on en déduire les concepts présents partiellement ou totalement dans plusieurs sources de données ?

Pour répondre à ces questions une approche de médiation basée sur la notion de schéma global à été développée.

Problématique

Quotidiennement, nous faisons appel à différentes sources d'information (journaux, livres, personnes, télévisions,...) pour obtenir des éléments de réponse à des questions qui nous intéressent ou nous sont utiles. En général, nos recherches d'information aboutissent rapidement car nous savons à quelles sources nous adresser et comment interagir avec elles.

De plus en plus de sources d'information sont stockées sur des supports informatiques sous forme de fichiers, bases de données ou pages Web. Les moteurs de recherche par exemple sont de simples outils de localisation d'adresses URL de pages Web contenant des mots-clés fournis par l'utilisateur pour exprimer sa demande d'information (sa requête). Ils ne permettent pas de fournir une réponse claire à une question précise comme « quelles sont les adresses de restaurants gastronomiques dans le quartier latin ? ».

L'intégration d'informations est un domaine de recherche qui a pour but de proposer des architectures pour la construction de systèmes de questions/réponses sur des informations hétérogènes provenant de multiples sources de données. Ces architectures de données permettent aux utilisateurs d'accéder à travers un schéma global unifié à plusieurs sources de données ayant chacune un schéma local. Ces sources de données sont le plus souvent autonomes et hétérogènes.

Les sources d'informations sont réparties : de plus en plus d'informations sont crées partout dans le monde et publiées, 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 des formats divers (HTML, PDF,... etc.) et des modèles de donnés différents sont utilisés pour les représenter (Relationnel, Objet, Semi structure).

De ce fait, l'accès « transparent » aux ressources et de manière plus générale à l'information constitue à être un des challenges actuels majeurs de l'informatique. L'hétérogénéité, la quantité, des ressources constituent autant de verrous que les systèmes d'intégration doivent lever.

Pour notre travail nous nous intéressons à concevoir et réaliser un système d'intégration des connaissances hétérogènes représentées par le langage OWL et les graphes conceptuel. L'hétérogénéité de connaissances concerne à la fois la syntaxe et la sémantique. L'hétérogénéité syntaxique provient du fait que les sources de connaissances peuvent avoir différents formats pour les stocker. L'hétérogénéité sémantique, par contre, présente un défi majeur dans le processus d'élaboration d'un système d'intégration. Elle est due aux différentes interprétations des objets du monde réel. En effet, les sources de connaissances sont conçues indépendamment, par des concepteurs différents, ayant des objectifs applicatifs différents. Chacun peut donc avoir un point de vue différent sur le même concept.

Objectif

Notre objectif est de concevoir et de réaliser un système d'intégration de données hétérogènes, présentés par le langage OWL et Graphes conceptuels. Pour permettre à un simple utilisateur de trouver des réponses à ses requêtes en lui cachant l'hétérogénéité syntaxique et sémantique de ces sources.

Notre système d'intégration doit fournir les objectifs suivants :

(1) fournir une vue globale intégrée des données représentées à travers différentes conceptualisations.

(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 que l'utilisateur n'ait le souci de la provenance des informations ni de leur format d'origine.

Notre système d'intégration sera basé sur l'approche médiateur, en construisant d'une manière automatique un schéma global qui sera crée en traitant le problème de l'hétérogénéité sémantique.

Organisation du mémoire

Ce mémoire est organisé en cinque chapitres. La première partie d'état de l'art comporte deux chapitres. Le premier chapitre est consacré à une étude détaillée des problèmes d'intégrations des sources de données hétérogènes. Le deuxième chapitre présente une étude détaillée des sources de connaissances OWL et GC afin de connaitre leurs syntaxe et sémantique. La deuxième partie

du mémoire est consacrée à la conception et la réalisation de notre système d'intégration. Le troisième chapitre permet l'expression et l'analyse des besoins de notre système. Le quatrième chapitre concerne la conception du système ; dans cette étape nous allons fixer l'architecture de notre système. Le dernier chapitre représente la dernière phase ou notre système est implémenté et testé.

Chapitre I

L'intégration des données hétérogènes par l'approche médiateur

1 Introduction

Avec le développement des différents outils de stockage le nombre de sources de données dispersées et hétérogènes ne cesse de croître. D'où l'intégration de ses différentes sources de données devient un challenge afin de pouvoir les interroger simultanément et combiner les résultats obtenus.

D'où le besoin de développer des systèmes d'intégration de données permettant aux utilisateurs d'accéder à travers un schéma global unifié à plusieurs sources de données ayant chacune un schéma local.

Pour cela différentes approches d'intégrations ont été mises en œuvre pour assurer la cohérence des données et d'homogénéiser les différents formats trouvés. Parmi ces approches « l'approche médiateur » sur la quelle notre projet est basé.

2 L'intégration des Données

2.1 Problème de l'intégration de données

L'autonomie des sources de données à intégrer représente différents défits comme la création d'une interface appropriée pour l'accès à ces sources de données, faire le mapping des vues à intégrer avec les schémas sources, faire différentes requêtes sur ces sources de données selon leurs schéma.

- L'autonomie

Les données qui participent au scenario d'intégration sont autonomes ce qui veut dire que les outils qui gèrent et contrôlent ces sources sont indépendants. En conséquence, pour la création d'un système d'intégration d'information, le problème d'autonomie doit être pris en considération.

Les différents aspects de l'autonomie sont:

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

Autonomie de communication : chaque source d'information locale décide quand elle va communiquer avec le système d'intégration c.à.d. quand elle répond à la requête.

Autonomie d'exécution: c'est pas le système d'intégration qui gère et contrôle les transactions locales ou les opérations externes de la source locale. En d'autre terme, une source d'information dans un scenario d'intégration de données n'as pas besoin d'informer les autres sources de l'ordre de son exécution et ses opérations.

Autonomie d'association : les sources locales ont la possibilité de s'associer ou de se désassocier du système d'intégration c.à.d. elles décident comment faire participer leurs fonctionnalités et données dans le système d'intégration. [SHO,09]

- Hétérogénéité

Un des problèmes les plus complexes lors de l'intégration de plusieurs sources de données autonome 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. L'hétérogénéité est indépendante de la distribution physique des données.

Selon Elmagarmid et Sheth un système d'information est homogène si le même logiciel gère les données sur tous les sites, les données ont le même format et la structure (même modèle de données) et appartiennent au même univers de discours. Au contraire, un système d'information est hétérogène s'il n'a pas accès à toutes les caractéristiques d'un système homogène. Cela signifie qu'il utilise différents modèles de données, langages de requêtes différents, SGBD ou un autre langage.

L'hétérogénéité provient des différents choix qui sont faits pour représenter les informations issues du monde réel dans un format informatique. [SHO, 09]

Elle se décline en trois catégories :

a. ***hétérogénéité syntaxiques :*** se retrouve dans les formats de stockage des données (XML, relationnel, objet, etc.), dans les langages d'interrogation (XQuery, SQL, OQL, etc.), dans les protocoles d'accès (HTTP, etc.), dans les interfaces, etc. par exemple la même information est représentée par des syntaxes et des concepts différents d'un modèle à un autre (Type d'enregistrement dans codasyl, relation dans le modèle relationnel, classe dans le modèle objet, balise dans XML, etc.) [BAL, 07]

b. ***hétérogénéité structurele :*** résultent d'une structuration et classification différente des informations. Ils sont étroitement liés aux choix de conception. Plusieurs types de conflits *structurels* apparaissent :

- *Les conflits de schémas :* résultent de l'utilisation de différents concepts pour représenter le même objet. Une information peut être représentée par une entité dans un système (S1) et par une relation dans un autre système (S2)
- *Les conflits de généralisation / spécialisation* : résultent des différences de hiérarchisation des informations.

Exemple : Dans un système (S1) les enseignants sont regroupés dans un seul objet ENSEGNANT alors que dans (S2) on utilise deux objets PERMANENTS et VACATAIRES pour représenter les enseignants.

- *Les conflits d'agrégation :* résultent d'un niveau de granularité différent de deux systèmes. La valeur d'un attribut sur un système correspond à une agrégation des valeurs de plusieurs attributs sur un autre.

Exemple : Nous disposons des moyennes modulaires des étudiants dans un système (S1) et le détail des notes obtenues au cours des examens dans le deuxième (S2)

- *Les conflits de typage :* résultent des différences de typage pour le même objet dans les différents systèmes de la coopération.

Exemple : Le salaire d'un employé est représenté comme numérique (99999.99) sur le système (S1) et comme monétaire (99.999,99 DA) sur le système (S2).

- *Les conflits de complétude :* apparaissent lorsque des objets se correspondent partiellement sur les différents systèmes. C'est-à-dire qu'une partie d'un objet du système (S1) trouve une correspondance sur le système (S2).

Exemple : Dans (S1) l'attribut Adresse contient Rue, Ville et pays alors que dans (S2) l'adresse contient seulement la Rue. Dans (S2) le nom contient à la fois nom et prénom. (S1) : ETUDIANT (matricule, Nom, Prénom, Adresse)(S2) : STAGIAIRE (N°ins, Nom, Adresse, Ville, Pays) [BAL, 07]

c. ***hétérogénéité sémantiques:*** proviennent des différences d'interprétation des informations partagées entre différents domaines d'application. Plusieurs types de conflits sémantiques existent entre les sources qui n'ont pas la même interprétation de l'information. D'après Goh, il y a principalement trois conflits sémantiques; le conflit d'indéterminisme, le conflit d'échelle et le conflit de noms.

• *Le conflit d'indéterminisme* est une confusion qui surgit entre les concepts qui semblent avoir le même sens, mais ils sont différents. Par exemple, dans des contextes temporels différents.

• *Le conflit d'échelle et d'unité :* se présente dans le cas où on utilise des unités de mesures différentes dans les différentes sources. Nous avons évoqué ce conflit lors de notre discussion sur les faiblesses de l'intégration structurelle, dans le cas des unités monétaires différentes. Un autre exemple de ce conflit est l'utilisation des unités de mesure pour la température. Bien que les deux valeurs *37* et *98.6* soient totalement différentes, elles ont la même interprétation sémantique si on considère que *37* est mesuré en *Celsius* et *98,6* en *Fahrenheit*.

• *Le conflit de noms :* résident dans les différentes désignations pour une valeur donnée. Les relations entre ces désignations sont souvent de type : *synonyme*, *homonyme*, *hyperonyme* et *hyponyme*. Par exemple, les trois termes suivants sont synonymes (*LIRE*, *L.I.R.E*, et *Laboratoire d'informatique répartie*). Pour le cas d'*homonyme*, on peut constater que dans « *mémoire* informatique » et « *mémoire* de thèse » le terme *mémoire* n'a pas le même sens, malgré qu'on utilise le même terme. Les deux dernières relations entrent dans le cadre d'une hiérarchie de concepts. Le terme *personne* est l'*hyperonyme* du terme *étudiant* car il a un sens plus général. [DJA, 03]

2.2 Taxonomie de conflits d'intégration

La résolution de l'hétérogénéité sémantique est une des questions fondamentales à résoudre quand on veut réaliser l'intégration entre plusieurs sources. Elle consiste en l'identification d'objets sémantiquement lies dans différentes sources de données et la résolution de la différence de schémas entre eux. Dans le domaine des bases de données fédérées, Sheth et Larson soulignent que l'hétérogénéité sémantique survient lorsqu'il n'y a pas entente

sur la signification, l'interprétation, ou sur l'utilisation souhaitée des données semblables ou liées.

D'un point de vue plus général, traiter le problème de l'hétérogénéité sémantique revient à traiter le problème de la correspondance entre schémas. Dans le domaine de l'intégration c'est un problème crucial, d'autant qu'intégrer des sources différentes (souvent développées dans des contextes différents) nécessite l'identification des éléments qui peuvent être liés entre les différents schémas. On parle alors de correspondances (ou mapping) entre schémas.

Plusieurs types de conflits dus à l'hétérogénéité peuvent être considérés dans l'établissement des correspondances entre schémas lors de l'intégration de données. Pour cela Plusieurs conflits de données apparaissent dans le processus d'interopérabilité et ont connu des classifications différentes. [BOU, 08]

Une version constituée de six types de conflits, est reprise par Parent et Spaccapietra.

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

2. *Conflits descriptifs* : ils surviennent dès qu'il y a une différence entre les propriétés des types d'objet en correspondance (les types d'objets peuvent différer selon leurs : noms, clés, attributs, les attributs peuvent aussi différer selon leur noms, structures, etc.). Un exemple de ce type de conflits (différence selon le nom) est l'utilisation de termes synonymes ou homonymes

dans la désignation d'un même type d'entité dans deux schémas différents : Thesard dans l'un et Doctorant dans l'autre.

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

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

5. *Conflits données/meta-donnees* : surviennent lorsqu'une donnée dans une base est en correspondance avec une meta-donnée (le nom d'un type) dans le schéma d'une autre base. En prenant comme exemple deux schémas de bases de données de praticiens avec praticien1 (id, neurologue, ...) et praticien2 (id, fonction, ...), certaines valeurs de l'attribut fonction de praticien2 peuvent correspondre au nom de l'attribut neurologue de praticien1.

6. *Conflits de données* : surviennent au niveau des instances, lorsque des occurrences en correspondance ont des valeurs en conflit pour des attributs en correspondance. [BOU, 08]

Une autre classification des conflits qui peuvent apparaître lors de la mise en correspondance entre schémas est proposée par Goh et ses collègues. Ce sont les conflits de nommage (naming), les conflits de graduation (scaling), les conflits de confusion (confounding conflicts) et les conflits de représentation.

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

2. *Les conflits de graduation :* apparaissent lorsque différents systèmes de graduation sont utilisés pour mesurer une valeur. On peut citer en exemple la mesure du poids d'un malade peut être exprimé en Kg dans une ressource et en pound dans une autre.

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

4. *Les conflits de représentation :* se produisent quand deux schémas sources décrivent le même concept de manière différente. Par exemple, dans une source l'adresse peut être désignée par une chaîne de caractères tandis que dans une autre, l'adresse est une structure composée du numéro et du nom de la rue, du code postal et de la ville. [BOU, 08]

Pour illustrer les différents conflits présentés ci-dessus, considérons l'exemple suivant. Soient deux sources de données, source 1 et source 2, modélisant les assurés sociaux. La Source 1 utilise cinq attributs pour décrire un assure, tandis que la Source 2 en utilise seulement quatre (l'attribut Mutuel n'est pas défini).

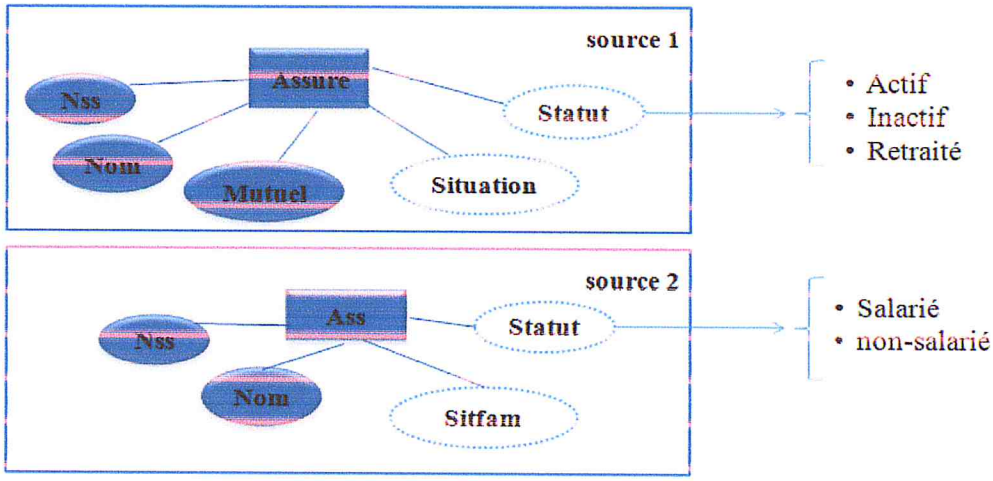

Figure1 : Exemple de sources hétérogènes. [BOU, 08]

Commentaire :

- Pour définir la situation familiale de chaque assure, la source 1 utilise la propriété Situation tandis que la deuxième utilise Sitfam→ Conflit structurels.

- Dans la source 1, le domaine de l'attribut Statut est un type énumère d'entiers (1 : Actif, 2 : Inactif, 3 : Retraite) tandis que le domaine de l'attribut Statut de la source 2 est un type énumère de caractères (S : Salarie, N : Non salarie)→ Conflit de noms.

- L'attribut Statut dans les deux sources a le même nom (Statut); mais avec deux sens différents. Source1.statut signifie la catégorie de l'assure, alors que dans la Source2, signifie la position de l'assure→ Conflit de noms. [BOU, 08]

Sheth et Kashyap ont proposé une autre classification qui est la suivante :

1. *Problème d'incompatibilité de domaine:* ce type de conflits survient lorsque deux objets aient différente définition pour des attributs de domaine similaire sémantiquement.

2. *Problème d'incompatibilité de définition d'entités:* cette incompatibilité lorsque l'utilisation d'un descripteur d'entité par deux objets similaires sémantiquement est partiellement compatible.

3. *Problème d'incompatibilité des valeurs de données:* ce type de conflits est dû aux différentes valeurs présentes dans différentes sources de données.

4. *Problème d'incompatibilité des niveaux d'abstraction :* ce problème ce produit quand deux entités similaires sémantiquement sont représentées sur différents niveaux d'abstraction. La généralisation et l'agrégation sont des cas de ce type d'incompatibilité.

5. *Problème de description schématique:* cette incompatibilité survient dans le cas où on a une donnée dans une source qui correspond à une métadonnée dans une autre. [SHO,09]

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

L'objectif de cette idée est de donner aux utilisateurs une interface uniforme pour les différentes sources de données, dite système d'intégration de données. Ainsi, les utilisateurs vont se focaliser pour spécifier ce qu'ils veulent et non pas perdre du temps en réfléchissant à la façon d'obtenir la réponse. Ils

n'ont pas à chercher les sources adéquates, à interagir avec chaque source en isolation en utilisant une interface particulière et combiner les différents résultats obtenus. [BOU, 08]

2.3 Systèmes d'intégration de données

Les systèmes d'intégration de données offrent des architectures d'interopérabilité sur une fédération de sources de données distribuées, autonomes et hétérogènes. Ils permettent d'accéder à ces sources de données de façon uniforme et transparente, en transformant par réécriture les requêtes d'un utilisateur en sous requêtes envoyées aux sources de données les plus appropriées. [XAV, 03]

2.3.1 Définition et Composante

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. [XAV,03]

Formellement, un système d'intégration de données I est un triplet G, S,M où :

- G est le schéma global, exprimé dans un langage L_G plus un alphabet A_G. L'alphabet comporte un symbole pour chaque élément de G (c.-à-d la relation « si G est relationnel », la classe « si G » est orienté objet, etc.)
- S est le schéma source, exprimé dans un langage L_S.
- M est un mapping entre G et S. [MAU, 02]

Pour interroger le système intégré, les requêtes sont exprimées en termes de constructions du schéma global G.

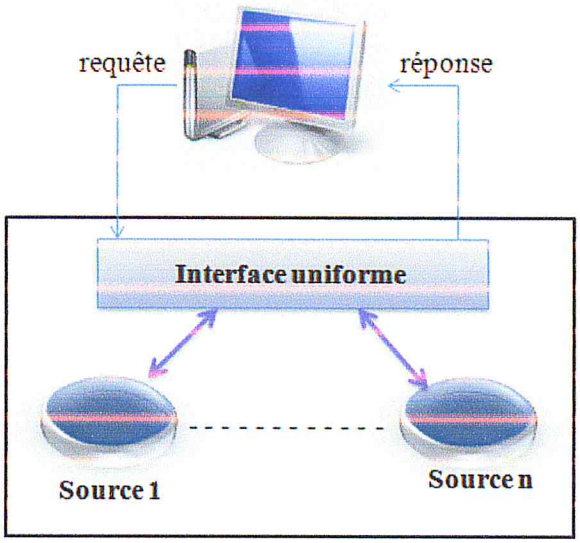

Figure 2: Système d'intégration d'information. [BOU, 08]

2.3.2 Processus d'intégration

Etant donné un ensemble de sources de données hétérogènes {S1, S2, …, Sn}, le problème d'intégration consiste à construire un schéma intègre (ou schéma global) qui sera utilisé comme interface d'accès aux sources de données. La construction du schéma global à partir des schémas locaux est une tâche difficile. Cette difficulté est liée au fait que les sources stockent différentes sortes de données, en différents formats, ayant différentes significations et associées aux différents noms. [BOU, 08]

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

1. *La préintégration* : Cette phase vise à préparer l'intégration des schémas en les rendant plus homogènes. Elle consiste principalement à traduire les schémas initiaux dans un modèle de données commun (réduction de l'hétérogénéité syntaxique).

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

3. *L'intégration :* Cette phase finale produit le schéma intégré et fournit les règles de traduction permettant de passer des schémas source au schéma intègre et inversement « mapping ». [BOU, 08]

2.3.3 Tâche d'un système d'intégration

On peut 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. *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.

2. *Traduction de requêtes:* La traduction de requêtes d'un langage (exemple : XQuery) en un autre langage (exp. 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.

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.

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, 08]

2.4 Panorama des approches d'intégration de données

L'hétérogénéité constitue un problème majeur à résoudre dans l'intégration de données. 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.

2.4.1 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. En général, un processus d'intégration structurel suit principalement ces phases :

• La phase *pré intégration* analyse les schémas de chaque système. Puis, elle détermine la technique et l'ordre d'intégration.

• La phase *comparaison des schémas* compare les schémas en vue de trouver les conflits et les propriétés de chaque source de données. Dans cette phase on détermine les éléments qui peuvent entrer ou non dans l'intégration.

• La phase *conformité des schémas* consiste à résoudre les conflits existants.

• La phase *fusionner et restructurer* permet de raffiner le schéma final et d'assurer sa conformité vis à vis des critères posés.

L'objectif final est de proposer un schéma global qui permet d'avoir une seule vue sur les différents schémas locaux. La création de ce schéma global va permettre la réalisation de deux tâches indispensables pour l'intégration, à savoir le mapping et la transformation de requêtes.

• Le mapping permet d'établir les correspondances entre les éléments des schémas locaux et le schéma global.

• La transformation de requêtes a pour rôle de créer des requêtes adaptées aux sources locales à partir d'une requête globale. Ainsi, chaque système local est invoqué par une requête qui peut l'exécuter. La technique de transformation de requêtes se base essentiellement sur le mapping déjà établi.

L'exemple suivant illustre l'idée de l'intégration structurelle. On considère deux systèmes pour la gestion commerciale des produits et les tables des produits de chaque système, la table « Catalogue_Produit » du système 1 et la table «Tprd » du deuxième système.

Ref_Produit	Nom_Produit	Prix_Produit
ICDuo6600	Intel Core Duo E 6600	220
P4V32	Pentium 4 3.2 GHZ	120
AA64V4800+	AMD Athlon 64 4800+	180

Tableau 1 : La table « Catalogue_produit » du système 1.

Code_Prd	Désignation_Prd	Prix_Vente	Prix_Achat
1	Intel Core Duo 2.13 Ghz	27000	25000
2	Pentium 4 3.2 GHZ	9000	8000
3	Imprimante Pixma 1500	4500	4000

Tableau 2 : La table « Tprd » du système 2

En analysant les deux schémas, on peut constater les conflits suivants :

• Les conflits de noms entre les attributs (*Ref_Produit, Code_Prd*), (*Nom_Produit, Désignation_Prd*) et (*Prix_Produit, Prix_Vente*).

L'attribut *Prix_Achat* de la table *Tprd* n'a pas de correspondance dans la première table.

• Un conflit de type de données entre *(Ref_Produit)* et *(Code_Prd)*. Le premier est alphanumérique tandis que le deuxième est numérique.

Pour résoudre les conflits de noms, il faut proposer un schéma global dans lequel chaque attribut correspond à un attribut local. Ainsi, on peut avoir trois attributs globaux. La figure 3 montre un mapping entre les attributs globaux et locaux.

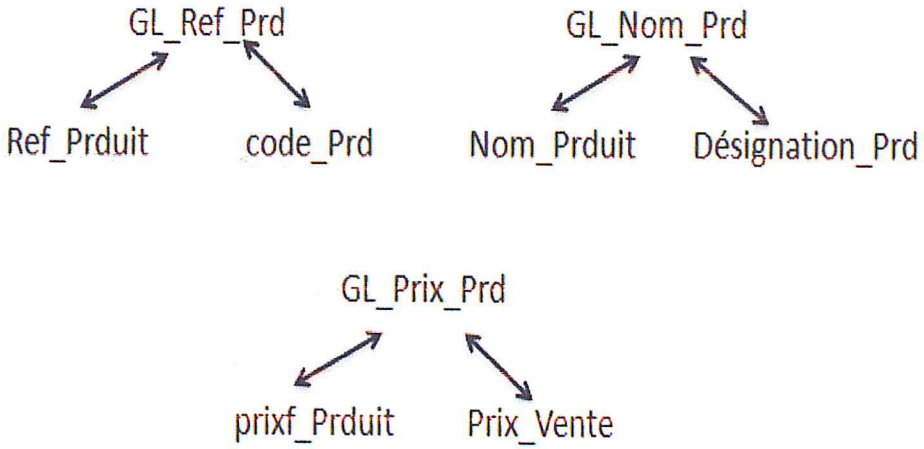

Figure 3: La correspondance entre le schéma global et les schémas locaux

Après la création du schéma global et l'établissement des mapping adéquats, le transformateur de requêtes permet de générer les requêtes locales appropriées.

Néanmoins, cette approche souffre de plusieurs faiblesses qui sont principalement :

• La création du schéma global nécessite d'avoir la structure exacte des différents schémas locaux. Or, cette connaissance n'est pas garantie.

• La structure des données n'est pas aussi simple que dans l'exemple. On peut avoir des structures plus complexes, par exemple : dans une source, on peut avoir un attribut *Adresse*, mais dans une autre, on trouve plusieurs attributs (*Cite, Rue, Ville*) qui ont le même rôle que *Adresse*.

• Les sources d'informations peuvent utiliser des modèles différents (relationnel, objet, semi structuré…). Si la création du mapping est complexe dans le cas où les sources d'informations utilisent le même modèle, elle est encore plus complexe dans le cas d'utilisation de plusieurs modèles de représentation.

• Même si l'extraction de la structure de chaque source est réalisable, celle de la sémantique est plus complexe. Cela rend le mapping entre les attributs une tâche très difficile.

• Dans l'intégration structurelle, l'aspect sémantique des données à intégrer n'est pas pris en charge. On ne peut pas déterminer les équivalences sémantiques entre les différents attributs des schémas. Ce problème est encore plus délicat dans le cas des conflits entre les données.

Ainsi, on remarque que la prise en charge de l'aspect sémantique est très importante dans le cadre d'une intégration de données. [DJA, 03]

2.4.2 L'intégration sémantique

Dans cette approche, on considère 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.

D'après Stuckenschmidt et Wache deux approches de transformation de contextes existent, a) L'approche basée sur les médiateurs et les règles de transformation, b) l'approche basée sur la classification des concepts.

Dans la première approche de transformation, les médiateurs se basent sur des règles de transformations appropriées, afin d'assurer la transformation de la structure locale vers des canevas spécifiques. Les règles de transformations constituent le cœur de cette approche, elles permettent d'avoir les différents mappings et les différentes transformations intermédiaires.

Dans l'approche de transformation basée sur la classification, l'idée est d'utiliser un formalisme capable de proposer une classification de concepts. Le formalisme utilisé est souvent la logique de description. La relation de subsumption associée aux capacités de raisonnements des logiques de description permet de fournir un nouveau contexte sémantique pour les sources locales. [DJA, 03]

3 L'approche médiateur

L'approche médiateur a fait l'objet de nombreux travaux. Les résultats obtenus à ce jour sont intéressants mais ne peuvent être mis en œuvre en l'état à l'échelle du Web. Dans le cadre du Web sémantique, l'intégration de sources d'information devra s'appuyer sur de *multiples* systèmes de médiation, ces systèmes participant de manière *distribuée et collective* au traitement des requêtes utilisateurs. Les connexions entre systèmes de médiation donneront au Web toute sa puissance, autorisant la recherche de données dans des sources non directement connectées aux sources du serveur interrogé. [HAC et al, 03]

3.1 Présentation générale

L'approche médiateur consiste à définir une interface entre l'agent (humain ou logiciel) qui pose une requête et l'ensemble des sources accessibles.

L'approche médiateur présente l'intérêt de pouvoir construire un système d'interrogation de sources de données sans toucher aux données qui restent stockées dans leurs sources d'origine. Ainsi, le médiateur ne peut pas évaluer directement les requêtes qui lui sont posées car il ne contient pas de données, ces dernières étant stockées de façon distribuée dans des sources indépendantes. [HAC et al,03]

3.2 Type de médiation

Selon la manière avec laquelle le médiateur traite les requêtes et utilise la sémantique des sources de données, nous pouvons distinguer deux types de médiation : médiation de schéma et médiation de contexte

Médiation de schéma : La médiation de schéma associe au médiateur un ensemble de connaissances qui localisent et intègrent les informations pertinentes à un contexte d'utilisation. Les requêtes sont exécutées sur ces connaissances (pré-établies) qui indiquent au médiateur la localisation physique des données et les correspondances entre les données pour permettre leur combinaison et transformation afin de restituer des résultats homogènes et exploitables.

Médiation de contexte : Cette approche repose sur une intégration dynamique des informations. Chaque application et chaque source de données locale doivent exprimer leur domaine sous forme de contexte d'utilisation des informations et ce pour permettre au médiateur, lors du traitement d'une requête, de comparer le contexte de l'application aux contextes des systèmes participant à la coopération. Le médiateur est guidé alors par des informations à caractère sémantique pour résoudre dynamiquement une requête sans connaissance préalable des SI participants. [BAL, 07]

3.3 Les composants d'un système de médiation

La médiation repose sur un composant essentiel, appelé médiateur, chargé de répondre à des besoins à partir de connaissances mises à sa disposition. Le médiateur permet de localiser l'information et de résoudre les conflits schématiques et sémantiques.

Un composant secondaire, appelé Wrapper, sert d'interface avec les sources de données.

Le Wrapper résout les conflits syntaxiques en présentant les données dans le modèle de médiation.

Médiateur : Un médiateur est un logiciel qui offre une interface unique et transparente à plusieurs bases de données hétérogènes et distribuées. Il est chargé de la localisation des données pertinentes et de la résolution des conflits (structurels et sémantiques).

Un ensemble de connaissances est mis à la disposition du médiateur pour lui permettre la génération d'un plan d'exécution pour traiter une requête exprimée dans le langage de médiation. Le médiateur constitue l'interface entre les utilisateurs et les données partagées par la coopération. Pour accéder aux bases de données locales, le médiateur fait appel à des Wrappers.

Wrapper : Un Wrapper de données est un logiciel qui convertit les requêtes exprimées dans le modèle de médiation provenant d'un ou de plusieurs médiateurs vers le modèle des bases de données locales. Il convertit les données, résultats d'une requête, du modèle des bases de données locales vers le modèle de médiation. Un Wrapper de données offre une interface homogène locale d'accès aux données sources (résolution des conflits syntaxiques).

Langage de médiation : le médiateur adopte un langage unique appelé langage de médiation pour permettre aux différents utilisateurs et applications d'interroger les différentes sources de données de la coopération de manière uniforme en leur masquant les détails d'hétérogénéité et de localisation. [BAL, 07]

3.4 Architecture d'un système de médiation

Le concept de médiateur dans le domaine des systèmes d'information est apparu pour intégrer des sources d'informations hétérogènes. "Gio

Wiederhold" a proposé une architecture logicielle à trois niveaux et a défini un médiateur de la façon suivante: "un *médiateur* est un module logiciel qui exploite la connaissance de certains ensembles ou sous-ensembles de données pour créer de l'information pour des applications à un niveau supérieur". Cette architecture, qui s'est largement imposée dans les systèmes d'information, repose sur les trois couches suivantes: sources, médiateurs et clients. [WIE, 91]

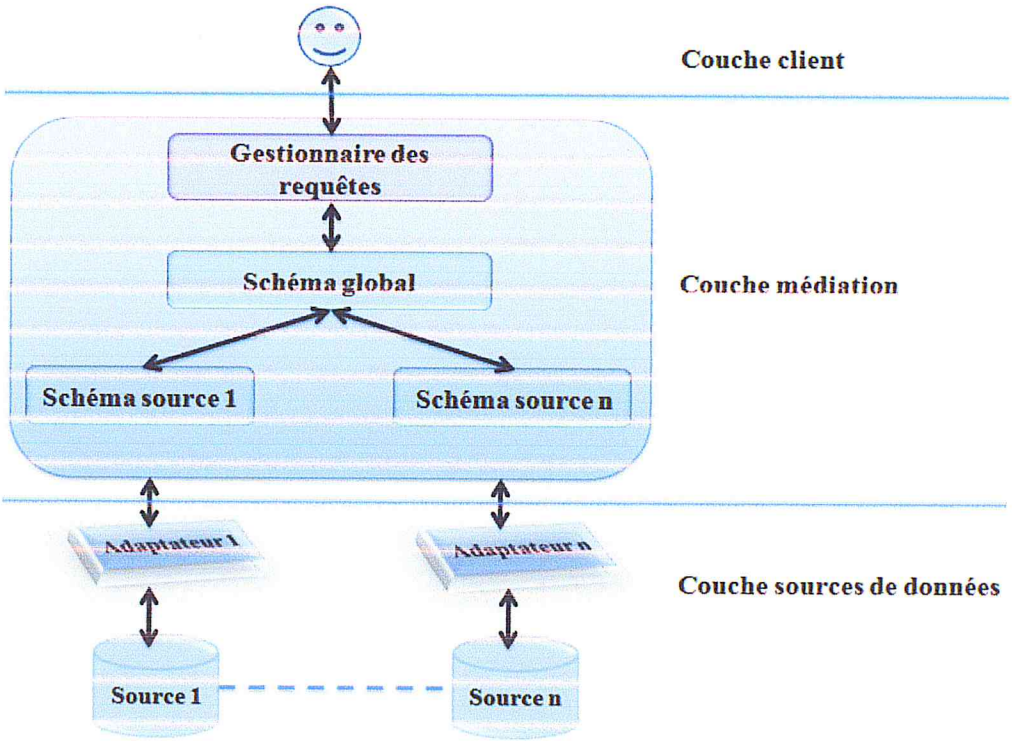

Figure 4 : Architecture d'un système médiateur

3.5 Les couches de l'architecture d'un système médiateur

Couche client : elle comprend des agents utilisateurs qui interagissent avec les utilisateurs ou les applications clientes pour les aider à formuler leurs requêtes. Ces agents transitent les requêtes vers l'agent médiateur de la couche de médiation et retournent les résultats à l'utilisateur.

Couche de médiation : c'est un lieu de rencontre des agents mobiles ; elle traite les requêtes venant des agents utilisateurs et les dirige vers les agents de requêtes correspondants. Elle transmet également les résultats venant de la couche des sources de données vers la couche client (par l'intermédiaire des agents utilisateurs).

Couche de sources de données : elle représente les différentes sources de données et leurs adaptateurs. Elle reçoit les requêtes à travers les agents de requêtes et renvoie les résultats obtenus de ces requêtes vers la couche de médiation à travers les mêmes agents via les adaptateurs appelés Wrapper en anglais, qui traduisent les requêtes réécrites en termes de vues dans le langage de requêtes spécifique accepté par chaque source. [BAK ,06]

3.6 Les approches d'intégration dans la médiation

Les différents systèmes d'intégration d'informations à base de médiateurs se distinguent par :

- La façon dont est établie la correspondance entre le schéma global et les schémas des sources de données à intégrer: on distingue l'approche GAV, l'approche LAV et l'approche BAV.

- Les langages utilisés pour modéliser le schéma global, les schémas des sources de données à intégrer et les requêtes des utilisateurs. [HAC et al, 03]

3.6.1 L'approche GAV

Dans l'approche GAV, la correspondance M associe à chaque élément g dans G une requête q_S sur S. En d'autres termes, le langage de requête $L_{M,G}$ permet uniquement les expressions constituées par un symbole de l'alphabet A_G. Par conséquent, un mapping GAV est un ensemble d'assertions, une pour chaque élément g de G, de la forme $g \rightarrow q_S$.

Du point de vue modélisation, l'approche GAV est basée sur l'idée que le contenu de chaque élément g du schéma global devrait être caractérisé en termes de vue q_S sur les sources. Dans un certain sens, le mapping dit très clairement au système comment récupérer les données lorsque l'on veut évaluer les différents éléments du schéma global. Cette idée est efficace lorsque le système d'intégration de données est basé sur un ensemble de sources qui est stable. Notons que, en principe, l'approche GAV favorise le système dans la réalisation de traitement de la requête, car il indique au système comment utiliser les sources pour récupérer des données.

Toutefois, l'extension du système avec une nouvelle source est maintenant un problème: la nouvelle source peut en effet avoir un impact sur la définition des différents éléments du schéma global qui associent les vues doivent être redéfinies. [MAU, 02]

3.6.2 L'approche LAV

Dans un système d'intégration de données I : <G, S, M> basée sur l'approche LAV, la correspondance M associe à chaque élément s du schéma source S une requête q_g sur G. En d'autres termes, le langage de requête $L_{m,S}$

permet uniquement les expressions constituées par un symbole de l'alphabet A_S. Par conséquent, un mapping LAV est un ensemble d'assertions, un pour chaque élément s de S, de la forme $s \rightarrow q_G$.

Du point de vue modélisation, l'approche LAV est basée sur l'idée que le contenu de chaque source s doit être caractérisé en termes de vue q_G sur le schéma global. le cas le plus important de ce type est lorsque le système d'intégration de données est basée sur un modèle d'entreprise, ou une ontologie. Cette idée est efficace lorsque le système d'intégration de données est basé sur un schéma global qui est stable et bien établie dans l'organisation. Notons que l'approche LAV favorise l'extensibilité du système, l'ajout d'une nouvelle source tout simplement, des moyens d'enrichir le mapping avec une nouvelle affirmation, sans autre modification.

Afin de mieux caractériser chaque source à l'égard du schéma global, plusieurs auteurs ont proposé des assertions plus sophistiquées dans le mapping LAV, en particulier avec l'objectif d'établir l'exploitation d'hypothèse pour les extensions de source diverses. Formellement, cela signifie que dans le mapping LAV, une nouvelle spécification, *noté (s)*, est associée à chaque élément *de* source s. La *spécification (s)* détermine le degré de précision est la connaissance sur les données répondant aux sources, à savoir, le degré de précision est la source par rapport à la vue associée q_G.[MAU,02]

3.6.3 L'approche BAV

BAV, est une autre méthode de mapping de schéma qui est basé sur une procédure de transformation. BAV peut capturer l'information sémantique qui est présente dans LAV et des règles de dérivation GAV, et il est possible de tirer règles exactes ou complètes de LAV et GAV à partir de BAV. Dans cette méthode, le traitement de la transformation est fait dans deux directions: des schémas locaux vers le schéma global et des schémas globaux vers chaque schéma local. Schéma de transformation est fait progressivement en appliquant

une séquence de transformations primitives consistant à ajouter, supprimer ou renommer un seul schéma de construction. Le nouvel élément est supprimé ou ignoré avec une requête basée sur le reste de la construction du schéma présenté dans [MCB et al, 03]

Le système d'intégration que nous proposons est un système à base de médiation en utilisant l'approche GAV car les deux autres approches ne suscitent pas notre intérêt vue que leur implémentation est complexe dans un système d'intégration d'autant plus que nos sources sont prédéfinies à l'avance et qu'on n'a pas besoin de la notion de flexibilité supportée par l'approche LAV.

3.7 L'aide apporté par les systèmes de médiation

L'aide apportée par les systèmes de médiation peut recouvrir différentes formes :

- ✓ Découvrir les sources pertinentes étant donnée une requête posée, puis aider à accéder à ces sources pertinentes.

- ✓ Éviter à l'utilisateur d'interroger lui-même chacune des sources selon leurs propres modalités et leur propre vocabulaire.

- ✓ Combiner automatiquement les réponses partielles obtenues de plusieurs sources de façon à délivrer une réponse globale. [MOS et al, 03]

Parmi les différentes grandes catégories d'applications de ces systèmes de médiation, on peut citer les applications:

- ✓ De recherche d'information.
- ✓ D'aide à la décision en ligne.
- ✓ De gestion de connaissances. [MOS et al, 03]

Chapitre II
Modèles de représentation de connaissance OWL, Graphe Conceptuel

1 Introduction

Durant les dernières années, la question de la conceptualisation des connaissances est devenue un thème central. Plusieurs raisons expliquent la nécessité de créer une base de connaissances fiable et utilisable. Parmi ces raisons, on peut citer : la structuration des données, l'enrichissement des données existantes par une sémantique appropriée, la proposition d'un vocabulaire commun dans un domaine.

Ainsi, la conceptualisation est un des concepts majeurs pour satisfaire ces besoins. Dans ce contexte, la notion d'ontologie est considérée comme une approche intéressante pour assurer une gestion des connaissances, et elle est susceptible d'être utilisée dans le domaine de l'intégration des sources de connaissances. [DJA, 03]

Dans la suite de ce chapitre nous citons les origines et les caractéristiques de l'ontologie en général puis nous présentons les fondements de l'ontologie. Ensuite nous présentons les modèles utilisés pour la représentation des ontologies, sur les quels notre thème se base, à savoir le langage OWL et les Graphe Conceptuels.

2 L'ontologie

Il est difficile de définir ce qu'est une ontologie d'une façon définitive. Le mot est en effet employé dans des contextes très différents touchant la philosophie, la linguistique ou l'intelligence artificielle. [DAM, 08]

Le terme « ontologie », est un mot emprunté de la philosophie qui signifie « Discours sur l'être en tant qu'être » selon Aristote. [COR et al, 06]

Le terme d'ontologie en informatique est apparu au début des années 1990, notamment dans le projet ARPA Knowledge Sharing Effort, en 1993

Une définition générale a été donnée par Thomas R.Gruber où il décrit une ontologie comme *« une spécification explicite d'une conceptualisation »* [MEL, 07]. Cette définition a été approfondie dans le travail de Borst où il donne la définition suivante : *« Une ontologie est définie comme étant une spécification formelle d'une conceptualisation partagée ».* Les concepts annoncés dans ces deux définitions ont été expliqués comme suit ; explicite signifie que le type de concepts utilisés et les contraintes liées à leur usage sont définis explicitement, conceptualisation réfère à un modèle abstrait d'un phénomène dans le monde, en ayant identifié les concepts appropriés à ce phénomène, formel réfère au fait que l'ontologie doit être traduite en langage interprétable par une machine. [DJA, 03]

Les ontologies contribuent essentiellement à résoudre le problème d'hétérogénéité sémantique entre les applications. L'utilisation des ontologies permet d'exprimer explicitement la sémantique inhérente aux applications, ce qui permet aux systèmes de coopérer sans avoir aucune confusion concernant les objets, concepts, appels, méthodes, etc. provenant de systèmes différents. [BAL, 07]

On distingue généralement deux entités globales au sein d'une ontologie. La première, à objectif terminologique, définit la nature des éléments qui composent le domaine de l'ontologie en question, un peu comme la définition d'une classe en programmation orientée objet définit la nature des objets que l'on va manipuler par la suite. La seconde partie d'une ontologie explicite les relations entre plusieurs instances de ces classes définies dans la partie terminologique. Ainsi, au sein d'une ontologie, les concepts sont définies les uns par rapport aux autres (modèle en graphe de l'organisation des connaissances), ce qui autorise un raisonnement et une manipulation de ces connaissances. [COR et al, 06]

2.1 Les composants d'une ontologie

Les connaissances traitées par l'ontologie sont modélisées à travers des éléments propres à cette dernière. Ces composants sont principalement concepts, relations, fonctions, axiomes et instances.

- *Les concepts :* appelés aussi classes d'ontologie, modélisent une abstraction pertinente d'un segment du domaine traité. Un concept peut être abstrait ou concret, atomique ou composé, réel ou fictif.

- *Les relations :* constituent les associations qui peuvent exister entre les concepts. Elles assurent une certaine interaction entre les différents concepts. La relation *sous-classe* est un exemple d'une relation binaire entre deux classes.

- *Les fonctions :* sont un cas particulier des relations dont un élément d'une relation est unique par rapport aux éléments qui le précèdent. Un exemple d'une fonction binaire est *Mère-de* qui donne la mère d'un individu. Ce dernier, doit avoir une seule mère.

- *Les axiomes :* modélisent les connaissances considérées comme vrais dans le domaine traité.

- *Les instances :* constituent des valeurs concrètes et des occurrences pour les concepts et les relations. [DJA, 03]

2.2 Rôles de l'ontologie

Les rôles de l'ontologie dans le domaine informatique les plus importants sont :

- *Fournir un vocabulaire commun :* afin de décrire un domaine particulier, un vocabulaire doit être proposé. La proposition d'un tel vocabulaire constitue le rôle fondamental d'une ontologie.

- *Structuration des données :* L'ontologie peut être considérée comme une base de concepts. Ainsi elle fournit une classification et une structuration des données du domaine.

- *Explication de l'implicite :* souvent les connaissances sont implicites. Cette situation peut causer des ambiguïtés dans un éventuel traitement. Dans ce contexte, l'ontologie explicite les connaissances implicites.

- *Proposition d'un méta-modèle :* Par définition, un modèle constitue une abstraction d'une réalité. Comme la notion d'ontologie se base sur la définition des concepts et des relations, elle s'intègre parfaitement dans une solution pour la création des modèles.

- *Interopérabilité sémantique :* L'interopérabilité entre les systèmes d'informations hétérogènes constitue un défi majeur. Dans ce contexte, l'ontologie constitue une solution intéressante pour ce problème. Elle permet de définir clairement les différents concepts utilisés, en vue d'assurer une communication sans ambiguïté entre les systèmes hétérogènes. [DJA, 03]

2.3 Classification des ontologies

La classification des ontologies permet de positionner chacune d'elles dans un contexte défini. Les principales classifications des ontologies en investissant trois dimensions, à savoir ; objet de conceptualisation, niveau de détail et niveau de formalisme de représentation.

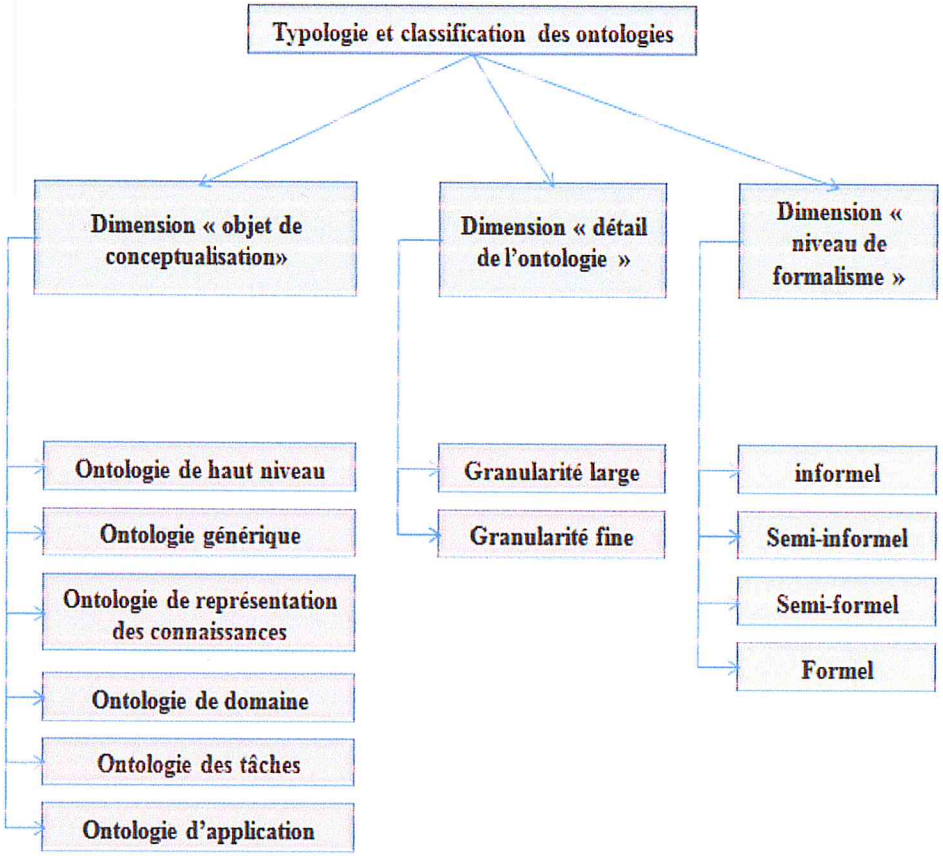

Figure 5 : Typologie et classification des ontologies. [DJA, 03]

En ce qui nous concerne nos ontologies sont des ontologies de domaine cette dernière est définie comme suit :

L'ontologie de domaine propose l'ensemble des termes et des concepts d'un domaine, comme scalpel ou scanner dans la médecine. Selon Mizoguchi, l'ontologie de domaine caractérise les connaissances du domaine où la tâche est réalisée. [DJA, 03]

3 L'OWL

3.1 Généralité

Le 10 février 2004 le W3C publie le langage des ontologies Web OWL qui est un dialecte XML basé sur une syntaxe RDF.

Le langage OWL est basé sur la recherche effectuée dans le domaine de la logique de description d'où il possède deux caractéristiques importantes. La première est sa base théorique consistante, vu sa relation avec les logiques de descriptions. La deuxième est la possibilité qu'OWL soit utilisé dans un environnement pratique tel que le Web [DJA, 03]. OWL peut être vu en quelque sorte comme un format de fichier pour certaines logiques de descriptions. Il permet de décrire des ontologies, c'est-à-dire qu'il permet de définir des terminologies pour décrire des domaines concrets. Une terminologie se constitue de concepts et de propriétés (aussi appelés rôles en logiques de description). Un domaine se compose d'instance de concepts. [COR et al, 06]. Il est basé sur une sémantique formelle définie par une syntaxe rigoureuse. [BEN, 08]

3.2 Les espèces de l'OWL

Le langage OWL offre trois sous-langages d'expression croissante destinés à des communautés de développeurs et d'utilisateurs spécifiques.

- Le langage OWL Lite est destiné aux utilisateurs ayant besoin principalement d'une hiérarchie de classifications et de contraintes simples.

- Le langage OWL DL est destiné aux utilisateurs demandant une expressivité maximale tout en gardant la complétude du calcul (toutes les

inférences sont garanties calculables) et la décidabilité (tous les calculs s'achèveront dans un intervalle de temps fini).

- Le langage OWL Full est destiné aux utilisateurs voulant une expressivité maximale et la liberté syntaxique de RDF sans garantie de calcul. [W3C]

Il existe entre ces trois sous langage une dépendance de nature hiérarchique : toute ontologie OWL Lite valide est également une ontologie OWL DL valide, et toute ontologie OWL DL valide est également une ontologie OWL Full valide.

3.3 Comparaison de OWL avec RDF et RDFS

OWL est, tout comme RDF, un langage XML profitant de l'universalité syntaxique de XML. Il est Fondé sur la syntaxe de RDF/XML, et il offre un moyen d'écrire des ontologies web. OWL se différencie du couple RDF/RDFS en ceci que, contrairement à RDF, il est justement un langage d'ontologies. Si RDF et RDFS apportent à l'utilisateur la capacité de décrire des classes (i.e. avec des constructeurs) et des propriétés, OWL intègre, en plus, des outils de comparaison des propriétés et des classes : identité, équivalence, contraire, cardinalité, symétrie, transitivité, disjonction, etc.

Ainsi, OWL offre aux machines une plus grande capacité d'interprétation du contenu web que RDF et RDFS, grâce à un vocabulaire plus large et à une vraie sémantique formelle. **[XAV, 05]**

3.4 La structure des ontologies OWL

Le langage OWL est un composant de l'activité Web Sémantique. Cet effort vise à rendre les ressources Web plus volontiers accessibles aux

processus automatisés en ajoutant des informations sur les ressources qui décrivent ou fournissent un contenu Web. Comme le Web sémantique est fondamentalement réparti (ou distribué), le langage OWL doit permettre de réunir des informations en provenance de sources réparties. On y parvient en partie en permettant aux ontologies d'être reliées, y compris en important explicitement des informations d'autres ontologies.

En outre, le langage OWL présuppose un monde ouvert. C'est-à-dire que les descriptions de ressources ne se réduisent pas à un seul fichier ou à une seule influence. Bien que la définition originale d'une classe C1 puisse se trouver dans l'ontologie O1, on peut l'étendre dans d'autres ontologies. Les conséquences de ces propositions supplémentaires sur C1 sont mono-toniques. Les nouvelles informations ne peuvent pas désavouer les précédentes. Elles peuvent être contradictoires mais les faits et les inférences peuvent seulement s'ajouter, et ne jamais s'annuler.

Pour écrire une ontologie interprétable sans ambiguïté et utilisable par des agents logiciels, nous avons besoin d'une syntaxe et d'une sémantique formelle pour OWL. OWL est une extension [RDF Semantics] du vocabulaire de RDF. [W3C]

3.4.1 Espaces de nommage

Afin de pouvoir employer des termes dans une ontologie, il est nécessaire d'indiquer avec précision de quels vocabulaires ces termes proviennent. C'est la raison pour laquelle, comme tout autre document XML, une ontologie commence par une déclaration d'espace de nom (parfois appelée « de nommage ») contenue dans une balise rdf:RDF. Supposons que nous souhaitons écrire une ontologie sur une population de personnes ou, d'une manière plus générale, sur l'humanité. Voici la déclaration d'espace de nom qui pourrait être employée : [XAV, 05]

```
<rdf:RDF
xmlns = "http://domain.tld/path/humanite#"
xmlns:humanite= "http://domain.tld/path/humanite#"
xmlns:base = "http://domain.tld/path/humanite#"
xmlns:vivant = "http://otherdomain.tld/otherpath/vivant#"
xmlns:owl = "http://www.w3.org/2002/07/owl#"
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">
```

Figure 6: declaration de l'espace de nommage de l'ontologie OWL.

Les deux premières déclarations identifient l'espace de nommage propre à l'ontologie. La première déclaration d'espace de nom indique à quelle ontologie se rapporter en cas d'utilisation de noms sans préfixe dans la suite de l'ontologie. La troisième déclaration identifie l'URI de base de l'ontologie courante.

La quatrième déclaration signifie simplement que, au cours de la rédaction de l'ontologie *humanité*, on va employer des concepts développés dans une ontologie *vivant*, qui décrit ce qu'est un être vivant. Les quatre dernières déclarations introduisent le vocabulaire d'OWL et les objets définis dans l'espace de nommage de RDF, du schéma RDF et des types de données du Schéma XML. [XAV, 05]

3.4.2 En-têtes d'une ontologie

On peut écrire, à la suite de la déclaration d'espaces de nom, un en-tête décrivant le contenu de l'ontologie courante. C'est la balise owl:Ontology qui permet d'indiquer ces informations. [XAV, 05]

```
<owl:Ontology rdf:about="">
<rdfs:comment>Ontologie décrivant l'humanité</rdfs:comment>
<owl:imports
rdf:resource="http://otherdomain.tld/otherpath/vivant"/>
<rdfs:label>Ontologie sur l'humanité</rdfs:label>
...
```

Figure 7: En-têtes d'une ontologie OWL.

3.4.3 Eléments du langage

3.4.3.1 Les classes

Une classe définit un groupe d'individus qui sont réunis parce qu'ils ont des caractéristiques similaires. L'ensemble des individus d'une classe est désigné par le terme « extension de classe », chacun de ces individus étant alors une « instance » de la classe. Les trois versions d'OWL comportent les mêmes mécanismes de classe, à ceci près que OWL FULL est la seule version à permettre qu'une classe soit l'instance d'une autre classe (d'une métaclasse). A l'inverse, OWL Lite et OWL DL n'autorisent pas qu'une instance de classe soit elle-même une classe. [XAV, 05]

a. Déclaration de classe

La déclaration d'une classe se fait par le biais du mécanisme de «description de classe», qui se présente sous diverses formes. Une classe peut ainsi se déclarer de six manières différentes :

- l'indicateur de classe : la description de la classe se fait, dans ce cas, directement par le nommage de cette classe. Une classe « humain » se déclare de la manière suivante :

```
<owl:Class rdf:ID="Humain" />
```

Figure 8: description directe de la classe.

- l'énumération des individus composant la classe : ce type de description se fait en énumérant les instances de la classe, à l'aide de la propriété owl:oneOf :

```
<owl:Class>
<owl:oneOf rdf:parseType="Collection">
<owl:Thing rdf:about="#Damien" />
<owl:Thing rdf:about="#Olivier" />
<owl:Thing rdf:about="#Philippe" />
<owl:Thing rdf:about="#Xavier" />
<owl:Thing rdf:about="#Yves" />
</owl:oneOf>
</owl:Class>
```

Figure 9: énumération des individus composant une classe.

Si ce mécanisme est utilisable avec OWL DL et OWL FULL, il ne fait cependant pas partie d'OWL Lite.

- La restriction de propriétés : la description par restriction de propriété permet de définir une classe anonyme composée de toutes les instances de owl:Thing qui satisfont une ou plusieurs propriétés.

Ces contraintes peuvent être de deux types : contrainte de valeur ou contrainte de cardinalité. Une contrainte de valeur s'exerce sur la valeur d'une certaine propriété de l'individu (par exemple, pour un individu de la classe Humain, sexe = Homme), tandis qu'une contrainte de cardinalité porte sur le nombre de valeurs que peut prendre une propriété (par exemple, pour un individu de la classe Humain, aPourFrere est une propriété qui peut ne pas avoir de valeur, ou avoir plusieurs valeurs, suivant le nombre de frères de l'individu. La contrainte de cardinalité portant sur aPourFrere restreindra donc la classe décrite aux individus pour lesquels la propriété aPourFrere apparaît un certain nombre de fois).

Enfin, il existe également une classe nommée noThing, qui est sous-classe de toutes les classes OWL. Cette classe ne peut avoir aucune instance. [XAV, 05]

3.4.3.2 Les instances de classe

a. Définition d'un individu

La définition d'un individu consiste à énoncer un « fait », encore appelé «axiome d'individu».

On peut distinguer deux types de faits :

- les faits concernant l'appartenance à une classe : la plupart des faits concerne généralement la déclaration de l'appartenance à une classe d'un individu et les valeurs de propriété de cet individu. Un fait s'exprime de la manière suivante : [XAV, 05]

```
<Humain rdf:ID="Pierre">
<aPourPere rdf:resource="#Jacques" />
<aPourFrere rdf:resource="#Paul" />
</Humain>
```

Figure 12: déclaration d'un fait.

Le fait écrit dans cet exemple exprime l'existence d'un Humain nommé « Pierre » dont le père s'appelle « Jacques », et qu'il a un frère nommé « Paul ». On peut également instancier un individu anonyme en omettant son identifiant : [XAV, 05]

```
<Humain>
<aPourPere rdf:resource="#Jacques" />
<aPourFrere rdf:resource="#Paul" />
</Humain>
```

Figure 13: déclaration d'un fait anonyme.

Ce fait décrit, dans ce cas, l'existence d'un Humain dont le père se nomme « Jacques » et qui a un frère nommé « Paul ».

- les faits concernant l'identité des individus : une difficulté qui peut éventuellement apparaître dans le nommage des individus concerne la non-unicité éventuelle des noms attribués aux individus. Par exemple, un même individu pourrait être désigné de plusieurs façons différentes.

C'est la raison pour laquelle OWL propose un mécanisme permettant de lever cette ambiguïté, à l'aide des propriétés owl:sameAs, owl:diffrentFrom et owl:allDifferent. L'exemple suivant permet de déclarer que les noms « Louis_XIV » et « Le_Roi_Soleil » désignent la même personne. [XAV, 05]

```
<rdf:Description rdf:about="#Louis_XIV">
<owl:sameAs rdf:resource="#Le_Roi_Soleil" />
</rdf:Description>
```

Figure 14: déclaration d'un synonyme d'un fait.

3.4.3.3 Les propriétés

OWL fait la distinction entre deux types de propriétés :

- les propriétés d'objet permettent de relier des instances à d'autres instances

- les propriétés de type de donnée permettent de relier des individus à des valeurs de données.

Une propriété d'objet est une instance de la classe owl:ObjectProperty, une propriété de type de donnée étant une instance de la classe owl:DatatypeProperty. Ces deux classes sont elles même des sous-classes de la classe rdf:Property. [XAV, 05]

```
<owl:ObjectProperty rdf:ID="aPourParent" />
```

Figure 15: définition d'une propriété d'objet.

a. Définition d'une propriété

Afin de spécifier une propriété, il existe différentes manières de restreindre la relation qu'elle symbolise. Par exemple, si on considère que l'existence d'une propriété pour un individu donné de l'ontologie constitue une fonction faisant correspondre à cet individu un autre individu ou une valeur de donnée, alors on peut préciser le domaine et l'image de la propriété. Une propriété peut également être définie comme la spécialisation d'une autre propriété. [XAV, 05]

```
<owl:ObjectProperty rdf:ID="habite">
<rdfs:domain rdf:resource="#Humain" />
<rdfs:range rdf:resource="#Pays" />
</owl:ObjectProperty>
```

Figure 16: définition d'une propriété entre deux classes.

Dans l'exemple ci-dessus, on apprend que la propriété *habite* a pour domaine la classe *Humain* et pour image la classe *Pays* : elle relie des instances de la classe *Humain* à des instances de la classe *Pays*. Dans le cas d'une propriété de type de donnée, l'image de la propriété peut être un type de donnée. Par exemple, on peut définir la propriété de type de données anneeDeNaissance :

```
<owl:Class rdf:ID="dateDeNaissance" />
<owl:DatatypeProperty rdf:ID="anneeDeNaissance">
<rdfs:domain rdf:resource="#dateDeNaissance" />
<rdfs:range rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

Figure 17: définition d'une propriété type de donnée.

Dans ce cas, anneDeNaissance fait correspondre aux instances de la classe de dateDeNaissance des entiers positifs.

On peut également employer un mécanisme de hiérarchie entre les propriétés, exactement comme il existe un mécanisme d'héritage sur les classes[XAV, 05]

```
<owl:Class rdf:ID="Humain" />
<owl:ObjectProperty rdf:ID="estDeLaFamilleDe">
<rdfs:domain rdf:resource="#Humain" />
<rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="aPourFrere">
<rdfs:subPropertyOf rdf:resource="#estDeLaFamilleDe" />
<rdfs:range rdf:resource="#Humain" />
...
</owl:ObjectProperty>
</owl:Class>
```

Figure 18: définition d'une hiérarchie de propriétés .

La propriété « aPourFrere » est une sous-propriété de « estDeLaFamilleDe », ce qui signifie que toute entité ayant une propriété « aPourFrere » d'une certaine valeur a aussi une propriété « estDeLaFamilleDe » de même valeur.

b. Caractéristiques des propriétés

En plus de ce mécanisme d'héritage et de restriction du domaine et de l'image d'une propriété, il existe divers moyens d'attacher des caractéristiques

aux propriétés, ce qui permet d'affiner grandement la qualité des raisonnements liés à cette propriété. Parmi ces caractéristiques, on trouve la transitivité, la symétrie, la fonctionnalité, l'inverse, etc.

```
<owl:ObjectProperty rdf:ID="aPourPere">
<rdfs:domain rdf:resource="#Humain" />
<rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="aPourFrere">
<rdf:type rdf:resource="&owl;SymmetricProperty" />
<rdfs:domain rdf:resource="#Humain" />
<rdfs:range rdf:resource="#Humain" />
</owl:ObjectProperty>
<Humain rdf:ID="Pierre">
<aPourFrere rdf:resource="#Paul" />
<aPourPere rdf:resource="#Jacques" />
</Humain>
```

Figure 19: caractériqtiques d'une propriété.

L'Humain Pierre a pour frère Paul, de même que (symétrie) l'Humain Paul a pour frère

Pierre. Par contre, si Pierre a pour père Jacques, l'inverse n'est pas vrai (aPourPere n'est pas symétrique). [XAV, 05]

4 Les graphes conceptuels

4.1 Généralités

Les graphes conceptuels ont été introduits par John Sowa en 1984. Ils sont un système de logique basé sur les réseaux sémantiques de l'intelligence artificielle. Ils sont la synthèse de plusieurs représentations sémantiques des connaissances comme par exemple la grammaire de cas de Fillmore, la grammaire de structure de phrase de Heidorn, la dépendance conceptuelle et les scripts de Schank. C'est la raison pour laquelle le modèle des graphes

conceptuels a montré son adaptabilité et ses preuves dans plusieurs domaines. J.F. Sowa propose que sa théorie soit un langage universel de représentation des connaissances pour tout système intelligent. Depuis 1984, Plusieurs travaux ont été faits sur les graphes conceptuels pour supporter différentes applications en traitement du langage naturel. Nous pouvons citer les domaines et les systèmes suivants :

- le traitement du langage naturel : génération, analyse du discours, génération du texte, temps et aspect…,

- la logique et raisonnement : moteur d'inférence, déduction et induction,..,

- l'ingénierie de connaissances : acquisition de connaissances, modèle sémantique de données, extraction de connaissance,...,

- application : système expert, recherche d'information, robotique,..,

- le système d'information et les graphes conceptuels (PUGET, 1993),

- l'interprétation du dialogue homme-machine dans un contexte multimodal (AZEM, 1993),

- le système intelligent destiné à contrôler la qualité et le niveau d'acquisition de concepts abstraits pour des étudiants (NICA, 1991)

- la génération automatique du texte (NOGIER, 91 ; FARGUES, 1984, 86, 88),

- le système d'interrogation d'une base de logiciel (CHEVALLIER, 1991,92). [CHA, 03]

Les graphes conceptuels sont très proches de la langue naturelle comme l'illustrent les exemples ci-dessous.

Les phrases *Le « Canada a pour capitale Ottawa »* et « *Une compagnie emploie une personne »* peuvent être modélisées par les graphes conceptuels suivant : [GEO]

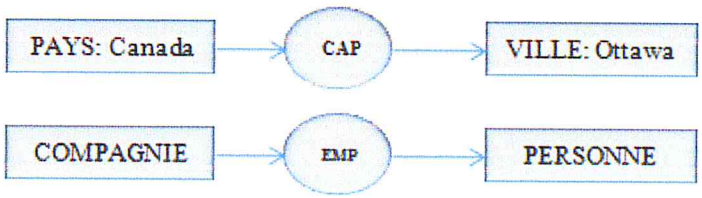

Figure 20: exemple de graphe conceptuel. [GEO]

Dans le premier graphe la ville d'Ottawa est représentée par le concept VILLE : Ottawa, le Canada est représenté par le concept PAYS : Canada et la relation de capitale, qui existe entre Ottawa et le Canada, est représentée par un arc joignant les deux concepts et porte le type de la relation, ici CAP pour capitale.

Dans le deuxième graphe la compagnie et la personne sont représentées respectivement par les concepts COMPAGNIE et PERSONNE et la relation d'emploi qui existe entre la personne et la compagnie est représentée par un arc dont le libelle est EMP pour emploie. [GEO]

4.2 Graphe conceptuel

Les graphes conceptuels ont été introduits par John-F Sowa (Sowa, 1984). Ils ont été conçus pour représenter la sémantique du langage naturel ; ils ont évolué pour devenir des systèmes complets au sens de la logique.

De façon générale, un graphe conceptuel est défini comme un graphe qui a deux sortes de nœuds :

- Les nœuds concepts qui représentent des entités, des attributs, des états, des événements...

- Les nœuds relations conceptuelles qui symbolisent les liens qui existent entre deux concepts.

Les concepts sont représentés graphiquement par des boites ou entre crochets, et les nœuds relations conceptuelles sont représentés entre parenthèses avec un et un seul arc entrant et un seul arc sortant ou par des cercles.

Un arc entrant relie un concept à une relation conceptuelle, et un arc sortant relie une relation conceptuelle à un concept.

Ainsi, un graphe conceptuel est :

- orienté,

- fini : tout graphe dans une mémoire d'un ordinateur ne peut avoir qu'un nombre fini de nœuds,

- connexe : si deux parties n'étaient pas connectées entre elles on aurait deux graphes conceptuels,

- bipartie : il ne possède que deux sortes de nœuds - les concepts et les relations conceptuelles- chaque arc reliant une sorte de nœud à l'autre sorte de nœud. [CHA, 03]

Un graphe conceptuel G=(R,C,U,etiq) définit un support S et un multi graphe non orienté bipartie tel que :

- R et C sont les deux classes de sommets, appelées relation et sommets concepts
- U est l'ensemble des arrêtes. Des arrêtes disjointes à un sommet relation «r » est totalement ordonné (de 1 à l'arête de r). On note Gi(r) l'Ième voisin de r dans G.
- Etiq est une application qui associe à chaque sommet une étiquète.

Les sommets concepts sont représentés par des rectangles et les sommets relations par des ellipses. [GEN,99]

4.3 Concepts

Les concepts représentent les objets de l'univers du discours. Ils sont représentés par des rectangles dans la forme graphique. Un concept est composé de deux parties. Il est l'assemblage d'un libellé de type et d'un référent optionnel qui identifie l'objet représenté. Suivant que l'objet référencé est connu ou inconnu, les concepts sont dits individuels ou génériques.

Un concept *individuel* représente un objet déterminé. Dans le langage naturel cela correspond à un nom propre ou un nom commun précédé par un article défini. Le concept est représenté par un identifiant précédé de son type dans le rectangle.

Un concept *générique* représente un objet indéterminé Dans le langage naturel cela correspond à un nom commun précédé par un article indéfini. Il introduit le quantificateur existentiel et correspond à "Il existe un objet ...". Il est représenté par une variable précédée de son type dans le rectangle ou uniquement par le type. [GER, 06]

Par exemple "MEDECIN" est un type qui représente la classe de tous les médecins.

En outre, les concepts obéissent à la notation suivante : [<Type>: < Référent>] Par exemple :

[PERSONNE : MAX]

[PERSONNE : # 804]

Il arrive que le référent du concept ne soit pas indiqué. C'est le cas des concepts génériques: [< Type générique>].

Le référent peut être:

✓ « * » qui indique un individu de la classe du concept, c'est-à-dire comme un concept générique. Par exemple:

[HOMME: *]

✓ « # » suivi d'un numéro indique un concept individuel. Par exemple :

[HOMME: # 124], pour désigner l'homme dont l'identification est 124.

✓ Soit une instanciation du concept, un individu est donné par son nom. Par exemple :

[HOMME: JEAN]

✓ « @ » pour indiquer une mesure

✓ « SET » pour indiquer un ensemble. Par exemple :

- SET (X1&X2 & ...Xn) signifie un ensemble de conjonctions X1,.., Xn

[PERSONNE: SET ('JEAN', 'MARIE')]

- SET (X1/X2/.../Xn) signifie un ensemble de disjonctions de X1,..., Xn.

❖ ***La « hiérarchie » des concepts***

Un modèle sémantique permet de généraliser ou au contraire de spécialiser. C'est le rôle de la hiérarchie des concepts. Elle classe par groupes les différents concepts du plus particulier au plus général.

La hiérarchie (ou treillis) des concepts est composée de diverses familles de concepts du même type, c'est-à-dire dépendant du même hyperonyme. Par exemple, prenons un type « Meuble » qui désigne l'ensemble des concepts relatifs au mobilier. « Meuble est un hyperonyme de chaise, bureau, armoire, etc... ».

Les concepts sont ordonnés selon leur degré de généralité par une relation d'ordre partiel notée «≤». Par exemple « Table < Meuble » signifie «une table est une sorte de Meuble».

Les termes de sur-type et de sous-type sont employés pour désigner la position respective de deux concepts dans la hiérarchie:

∀ t et s deux types de concepts si t ≤ s alors

- t est un sous-type de s

- s est un sur-type de t

Tout type de concept est à la fois sous-type et sur-type de lui même :

∀ t un type de concepts t ≤ t.

La relation £ est donc une relation d'ordre au sens mathématique du terme. En effet :

- elle est réflexive : ∀ le type de concept t, t ≤ t,

- elle est antisymétrique : ∀t et s deux types de concepts, si t ≤ s et s ≤t alors t = s.

Un type t ne peut être sur-type et sous type d'un autre type s sauf si nous avons le même type : t = s.

- elle est transitive : ∀ t, s et u trois types de concepts, si t ≤ s et s ≤ u alors t ≤ u comme t est sous-type de s, lui même sous-type de u, on peut en conclure que t est sous-type de u.

La hiérarchie des concepts décrite par Sowa (1984) comporte deux types de concepts qui assurent la complétude du système:

- le type universel : UNIV est le sur-type de tous les types de concepts,

- le type absurd : noté ABSURD, est le sous type de tous les types de concepts.

De ce fait, ∀ t un type de concepts, on a :

ABSURD ≤..... ≤ T ≤ .... ≤ UNIV

4.4 Relations conceptuelles

Les relations conceptuelles permettent d'assembler les concepts pour construire une phrase, une idée, une proposition. Les relations conceptuelles sont la deuxième sorte de nœuds qui constituent un graphe conceptuel. [GEO]

4.4.1 Notation

D'une manière générale, une relation se lit toujours dans le sens des flèches : [C1] (RELATION) [C2] signifie que « C1 a pour RELATION C2 »

Dans l'exemple suivant :

Figure 21: un exemple de relation entre concepts concepts. [CHA, 03]

La relation (AGT) du graphe lie l'action « jouer » à la personne qui l'exécute « MAX ». Et le graphe se lit « Jouer a pour agent Max ».

D'une manière générale, une relation entre concepts spécifie le rôle joué par ces concepts dans le graphe [CHA, 03]

Sowa distingue trois sortes de relations conceptuelles :

- Primitive : la relation binaire LINK est la seule relation faisant partie de la théorie des graphes conceptuels. Toutes les autres relations peuvent être construites à partir de celle-ci.

Kit de départ : c'est l'ensemble des relations les plus couramment utilisées. Plusieurs types de relations ont été définis par Sowa, Nogier et Chawk (2000). Voici une liste non exhaustive de ces types : [GEO]

- AGT : agent (entité intervenant de façon active et directement dans le procès),

- PAT : patient (entité intervenant de façon passive dans le procès,

- OBJ : objet (entité affectée par le procès),

- INST : instrument (moyen par lequel un agent agit pour un résultat ou une cause),

- LOC : lieu,

- TEM : temps,

- DEST : destination (aboutissement qui peut être de nature spatiale),

- ORIG : origine (provenance spatiale ou abstraite),

- CRC : caractéristique,

- MNR : manière,

- APP : appartenance,

- POSS: possession...etc. [CHA, 03]

- Définies : ce sont les relations définies pour des besoins spécifiques et qui concernent un domaine application.

Pour les relations dont les libellés sont des noms, les conventions de lecture suivantes sont souvent utilisées. Quand le graphe est lu dans le sens des flèches, l'arc dirige vers la relation est lu "a un(e)", et l'arc qui est issu de la relation est lu "qui est". Quand le graphe est lu dans le sens contraire des flèches, l'arc issu de la relation est lu "est un(e)" et l'arc dirigé vers la relation est lu "de". [GEO]

Bien que la plupart des relations conceptuelles soient binaires, il est possible de définir des relations n-aires. Par exemple, la relation ENTRE est une relation tertiaire. Les deux premiers arcs sont issus des objets qui encadrent le troisième. Le graphe suivant représente la phrase *John est entre un arbre et une voiture*.

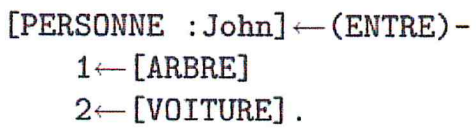

Figure 22 : exemple de relation n-aire. [GEO]

4.5 Référents

Les concepts représentent le sens des mots du discours. Les concepts sont des interprétations des entités, des actions, des propriétés ou des événements du monde réel.

Une interprétation est constituée d'un type et d'un référent. Le type catégorise l'objet représenté. Le référent représente et identifie l'objet de l'univers du discours dont le concept est une interprétation.

Les référents des concepts VILLE : Ottawa et PAYS : Canada sont respectivement Ottawa et Canada. [GER, 06]

5 Conclusion

Cette étude des deux modèles de représentation des connaissances et une étape importante pour comprendre la syntaxe afin de pouvoir réaliser un système d'intégration des deux modèles de connaissances.

Chapitre III
expression des besoins et analyse

1 Introduction

L'objectif de notre travail est de concevoir et de réaliser un système d'intégration de sources de connaissances. Au premier point nous présentons le cycle de vie que nous avons suivit pour la réalisation de ce projet. Par la suite et dans le but d'élucider les besoins que doit vérifier notre système nous allons décrire la solution de notre système, en se basant sur le langage UML pour représenter les différentes spécifications statiques et dynamiques. Cette partie sera organisée comme suit :

Nous commençons d'abord par citer les fonctionnalités réalisées par notre système puis nous détaillerons à l'aide des cas d'utilisation, les besoins définitifs que nous devons prendre en charge durant la phase de conception et réalisation.

Pour montrer la dynamique du système nous spécifions quelques diagrammes de séquences et nous clôturons cette spécification par des diagrammes d'activités.

2 Cycle de vie suivi

Le cycle de vie d'un logiciel est un ensemble séquentiel de phases, dont le nom et le nombre sont déterminés en fonction des besoins du projet, permettant généralement le développement d'un service ou d'un produit, en ce qui concerne notre projet nous avons suivit le modèle en cascade, ce dernier est un cycle de vie linéaire, séquentiel, il a été défini dans les années 70. Ce cycle de vie est basé sur la production d'élément livrable.

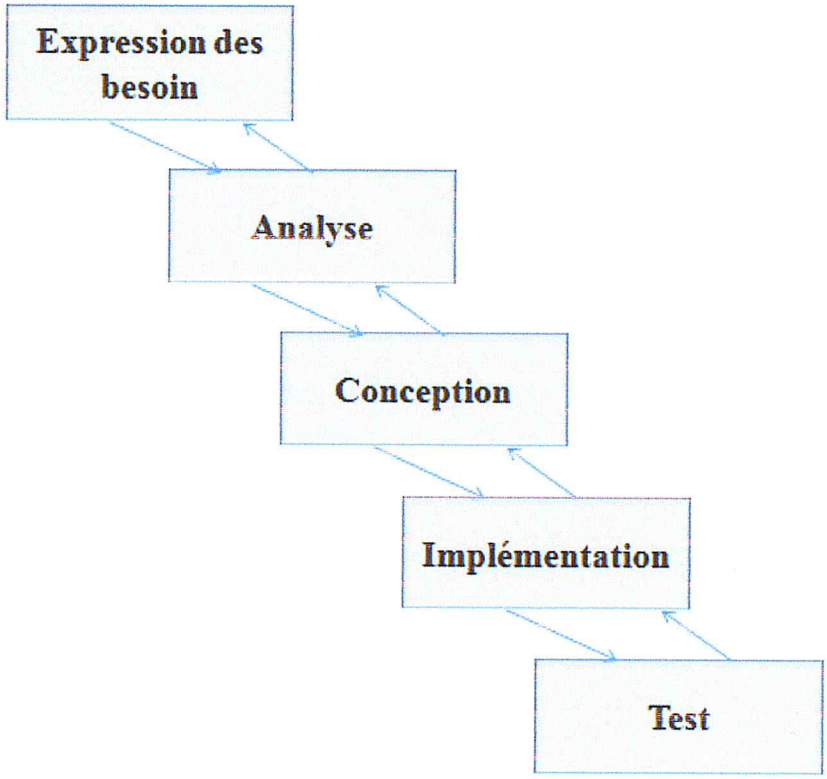

Figure 23 : Modèle du cycle de vie en cascade.

3 Les fonctionnalités générales du système

Notre système devra assurer sa fonctionnalité principale qui consiste en l'intégration des sources de connaissances pour offrir à l'utilisateur une vue uniforme sur ces sources. Pour se faire il doit prendre en charge les exigences suivantes :

- Traitement de la requête de l'utilisateur posée sous format du schéma global.
- Localisation des sources de données pertinentes.
- Reformulation de la requête utilisateur en des requêtes sur les schémas locaux.
- Récupération des résultats obtenus à partir de l'interrogation des sources.

- Recomposition des résultats retournés.

4 Expression des besoins

Cette partie a pour but, la modélisation du système existant. A ce niveau d'étude, la modélisation comprendra les diagrammes suivants :

- Les diagrammes de cas d'utilisation
- Les diagrammes de séquence
- Les diagrammes d'activités

4.1 Diagrammes des cas d'utilisations

Les acteurs

L'acteur est, par définition le rôle joué par une personne qui interagit avec le système. Il est en principe extérieur au système, délimité par ses bornes. L'acteur a un nom, qui le définit, ou qui précise son rôle dans la transaction décrite.

Notre système est composé de deux acteurs :

L'utilisateur simple : il a le droit de poser des requêtes sur les sources de connaissances hétérogènes et de recevoir des réponses homogènes et uniformes.

L'expert : il assiste le système pour l'intégration des sources de connaissances hétérogènes.

Les cas d'utilisation

Un cas d'utilisation est une unité cohérente représentant une fonctionnalité visible de l'extérieur. Il réalise un service de bout en bout, avec un déclenchement, un déroulement et une fin, pour l'acteur qui l'initie. Un cas d'utilisation modélise donc un service rendu par le système, sans imposer le mode de réalisation de ce service.

- ✓ **diagramme de cas d'utilisation choix d'ontologies**

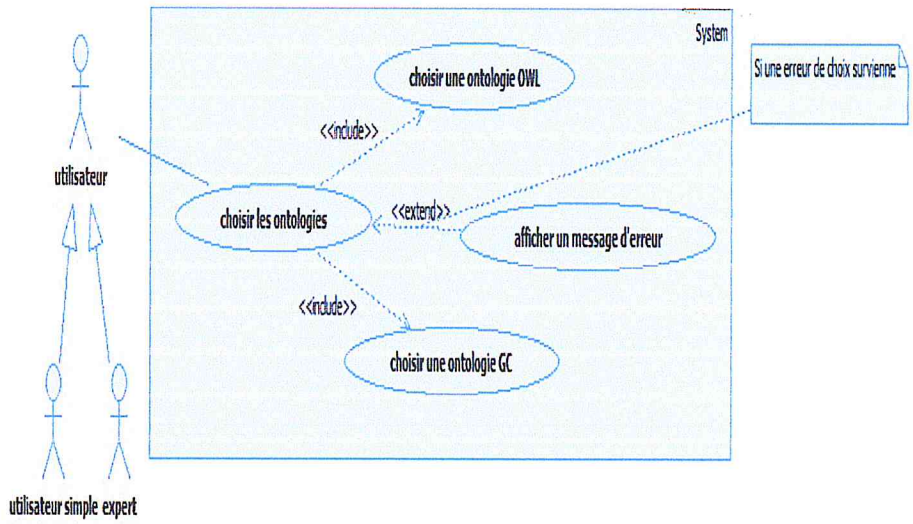

Figure 24 : diagramme de cas d'utilisation choix d'ontologies.

✓ **Description du diagramme du cas d'utilisation choix d'ontologies**

Cas d'utilisation	Description
Choisir les ontologies	L'utilisateur choisit les ontologies à interroger
Choisir une ontologie OWL	L'utilisateur sélectionne l'ontologie OWL.
Choisir une ontologie GC	L'utilisateur sélectionne l'ontologie GC.
Afficher un message d'erreur	Dans le cas ou l'utilisateur choisit la même ontologie ou bien une seule ontologie le message d'erreur est affiché.

Tableau 3:Description du cas d'utilisation choix d'ontologies.

✓ **diagramme de cas d'utilisation annotation de concepts**

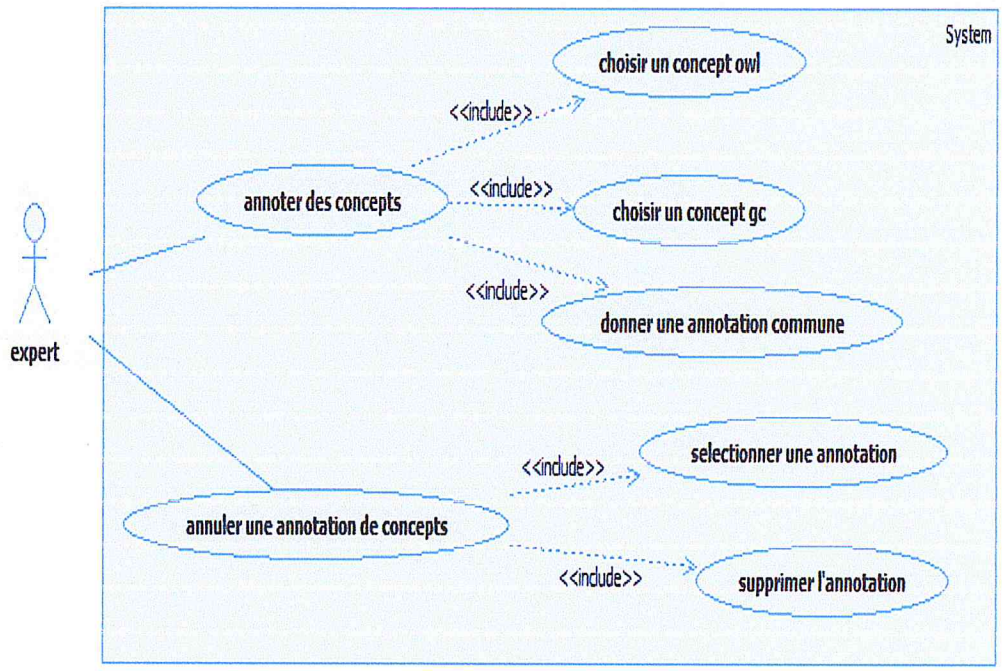

Figure 25 : diagramme de cas d'utilisation annotation de concepts.

✓ **Description du diagramme du cas d'utilisation annotation de concepts**

Cas d'utilisation	Description
Annoter des concepts	L'expert annote les concepts qui les juge qu'ils ont la même signification.
Choisir un concept owl	L'expert sélectionne le concept OWL qu'il veut annoter.
Choisir un concept gc	L'expert sélectionne le concept graphe conceptuel qu'il veut annoter.
Donner une annotation commune	L'annotation effectue par l'utilisateur est utiliser pour la construction du schéma global.
Annuler une annotation de concepts	L'expert a le droit d'annuler une annotation déjà effectuée.
Sélectionner une annotation	L'expert sélectionne l'annotation qu'il veut supprimer.
Supprimer l'annotation	L'expert supprime l'annotation

Tableau 4:Description du cas d'utilisation annotation de concepts.

✓ **diagramme du cas d'utilisation annotation de relations**

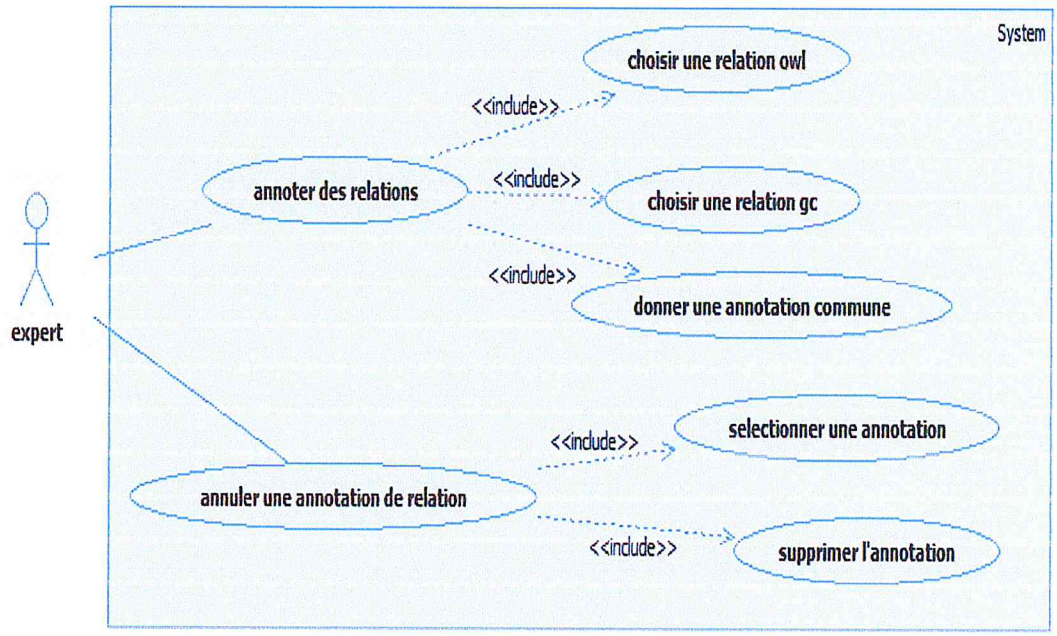

Figure 26 : diagramme de cas d'utilisation annotation de relations.

✓ **Description du diagramme du cas d'utilisation annotation de relations**

Cas d'utilisation	Description
Annoter des relations	L'expert annote les relations qui les juge qu'ils ont la même signification.
Choisir une relation owl	L'expert sélectionne la relation OWL qu'il veut annoter.
Choisir une relation gc	L'expert sélectionne la relation graphe conceptuel qu'il veut annoter.
Donner une annotation commune	L'annotation effectue par l'utilisateur est utiliser pour la construction du schéma global.
Annuler une annotation de relations	L'expert a le droit d'annuler une annotation déjà effectuée.
Sélectionner une annotation	L'expert sélectionne l'annotation qu'il veut supprimer.
Supprimer l'annotation	L'expert supprime l'annotation.

Tableau 5:Description du cas d'utilisation annotation de relation.

✓ **Diagramme de cas d'utilisation création du schéma global**

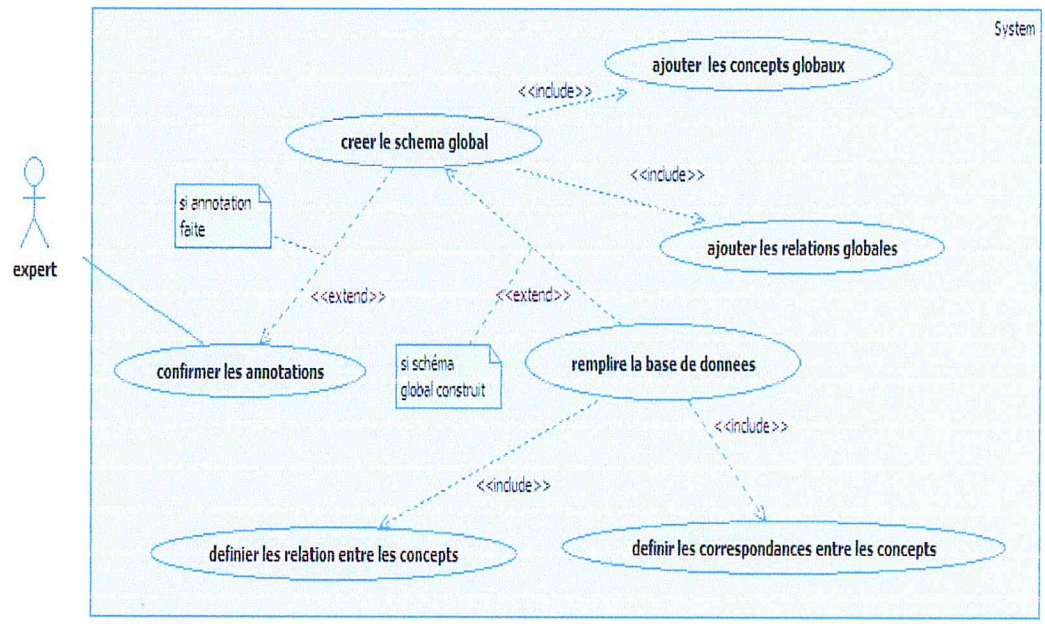

Figure 27: diagramme de cas d'utilisation création du schéma global.

✓ **Description du diagramme du cas d'utilisation création du schéma global**

Cas d'utilisation	Description
Confirmer les annotations	L'expert r confirme ses annotations afin.
Créer le schéma global	Le système crée le schéma global qui est une ontologie OWL.
Ajouter les concepts globaux	Le système ajoute les concepts globaux.
Ajouter les relations globales	Le système ajoute les relations globales.
Remplir la base de données	pour pouvoir décomposer et trouver les valeurs des concepts et des relations dans les sources de connaissances.
Définir les correspondances entre les concepts	Définit les correspondances entre les concepts des sources locales et les sources globales pour remplir la base de données.
Définir les relations entres les concepts	Définit les relations entres les différents concepts locaux (respectivement globaux).

Tableau 6:Description du cas d'utilisation création du schéma global.

✓ **Diagramme de cas d'utilisation lancement d'une requête**

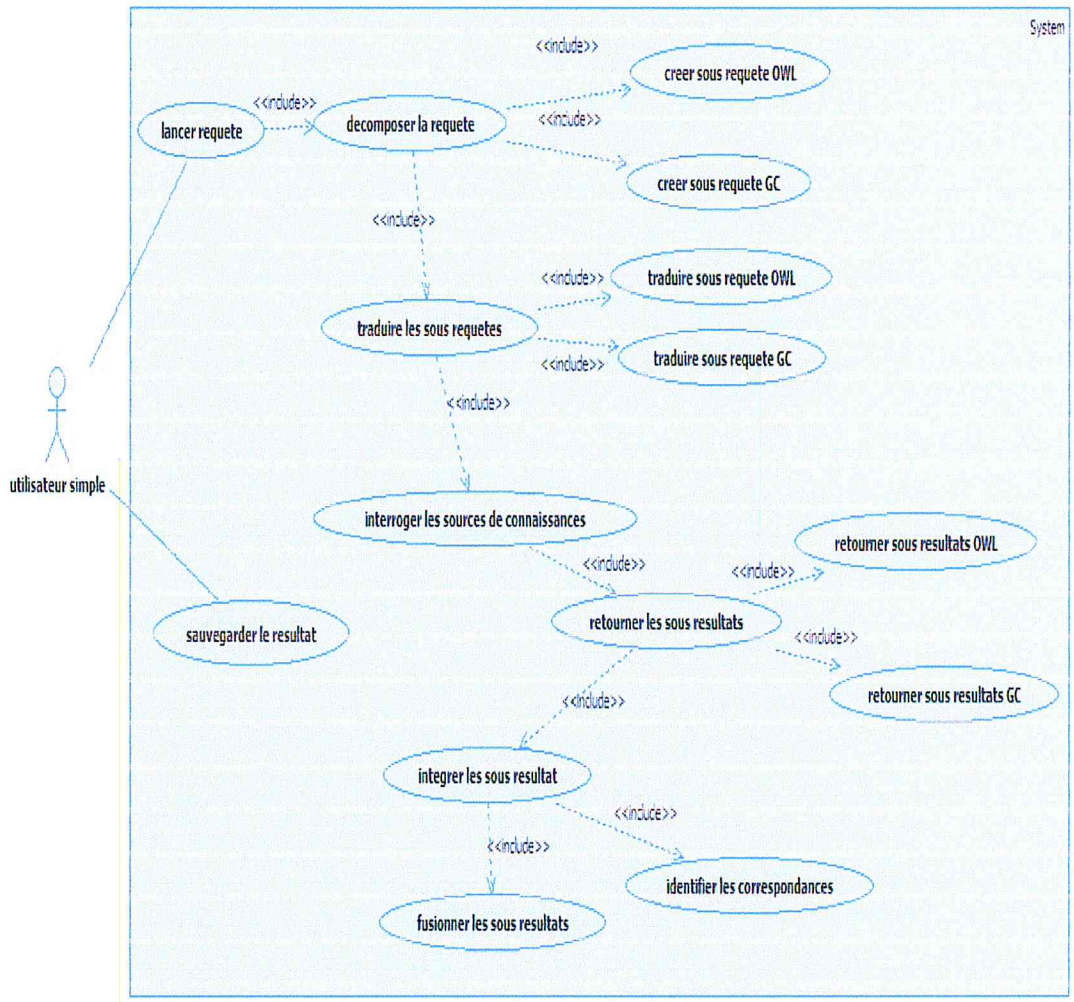

Figure 28 : diagramme de cas d'utilisation lancement d'une requête.

✓ **Description du diagramme du cas d'utilisation lancement d'une requête**

Cas d'utilisation	Description
Lancer requête	L'utilisateur simple lance une requête globale.
Décomposer la requête	Le système décompose la requête en sous requête owl et gc
Créer sous requête OWL	Le système ajoute les concepts globaux.
Créer sous requête GC	Le système ajoute les relations globales.
Définir les concepts OWL	Définir les concepts globaux qui se trouvent au niveau de la source OWL.

Définir les relations OWL	Définir les relations globales qui se trouvent au niveau de la source OWL.
Définir les concepts GC	Définir les concepts globaux qui se trouvent au niveau de la source GC.
Définir les relations GC	Définir les relations globales qui se trouvent au niveau de la source GC.
Interroger les sources de connaissances	Appliquer les requêtes sur les sources est extraire les résultats.
Intégrer les sous résultats	A ce niveau le système définit les correspondances entre les valeurs des résultats pour les fusionner.
Retourner le résultat	L'utilisateur simple reçoit le résultat uniforme.
Retourner le sous résultat OWL	Le système retourne le résultat de l'interrogation de la source OWL par la requête OWL.
Retourner le sous résultat GC	Le système retourne le résultat de l'interrogation de la source GC par la requête GC.
Identifier les correspondances	Le système définit les résultats qui représentent les mêmes valeurs.
Fusionner les sous résultats	Le système regroupe les sous résultats en un seul.

Tableau 7:Description du cas d'utilisation lancement d'une requête.

✓ **Diagramme de cas d'utilisation fermeture du système**

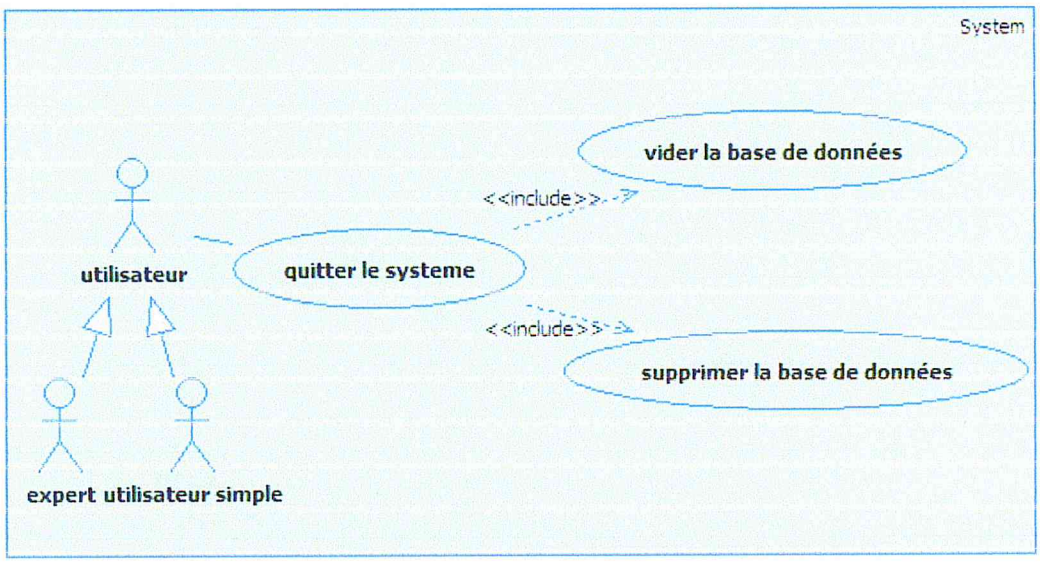

Figure 29 : diagramme de cas d'utilisation fermeture du système.

✓ **Description du diagramme du cas d'utilisation fermeture du système**

Cas d'utilisation	Description
Quitter le système	L'utilisateur quitte le système après qu'il finit d'effectuer ses requêtes.
Supprimer le schéma global	Le système supprime automatiquement le schéma global.
Vider la base de données	Le système vide automatiquement la base de données.

Tableau 8:Description du cas d'utilisation fermeture du système.

4.2 Diagrammes de séquence

Les principales informations contenues dans un diagramme de séquence sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique. Ainsi le temps est représenté explicitement par une dimension (la dimension verticale) et s'écoule de haut en bas.

✓ **Diagramme de séquence annotation des concepts**

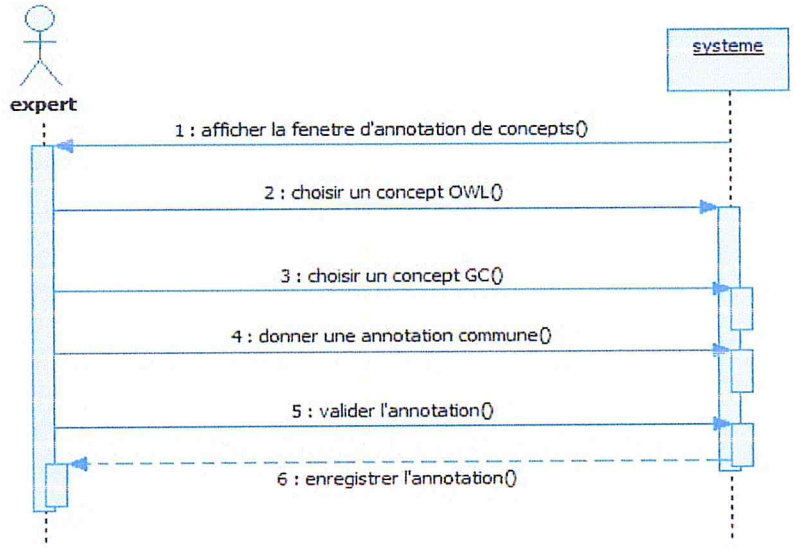

Figure 30 : diagramme de séquence annotation des concepts.

✓ **Diagramme de séquence annotation des concepts**

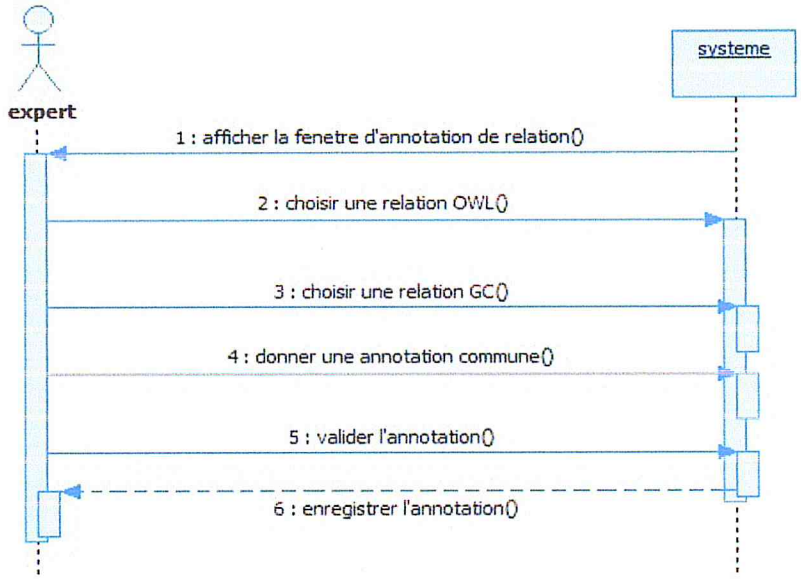

Figure 31 : diagramme de séquence annotation des relations.

✓ Diagramme de séquence lancement d'une requête

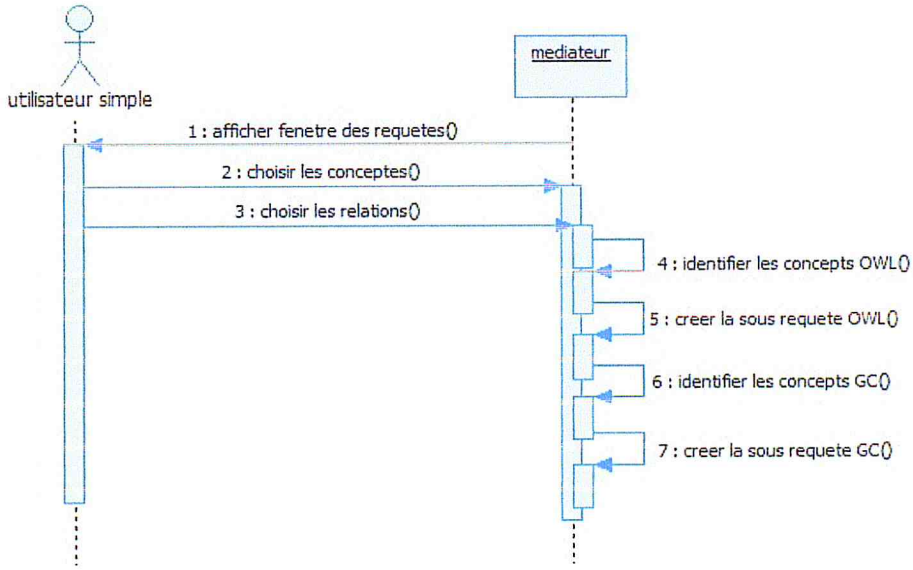

Figure 32: diagramme de séquence de lancement d'une requête.

✓ Diagramme de séquence récupération des sous résultats

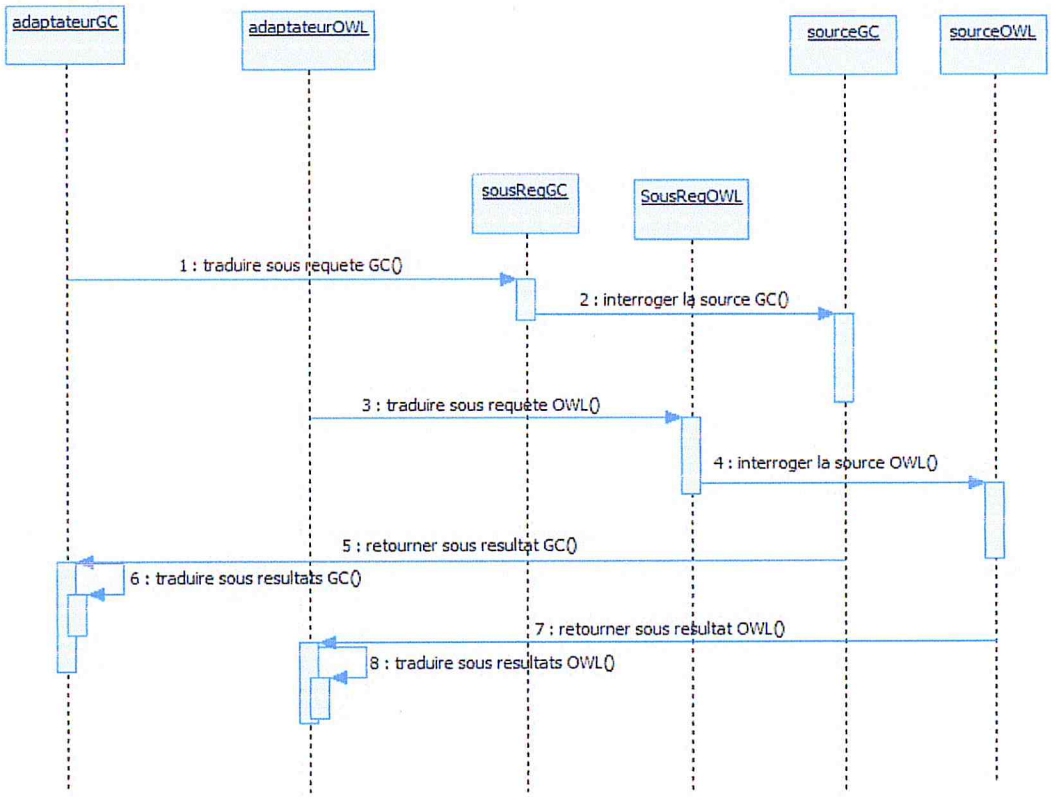

Figure 33: diagramme de séquence récupération des sous résultats.

✓ Diagramme de séquence fusionnement des sous résultats

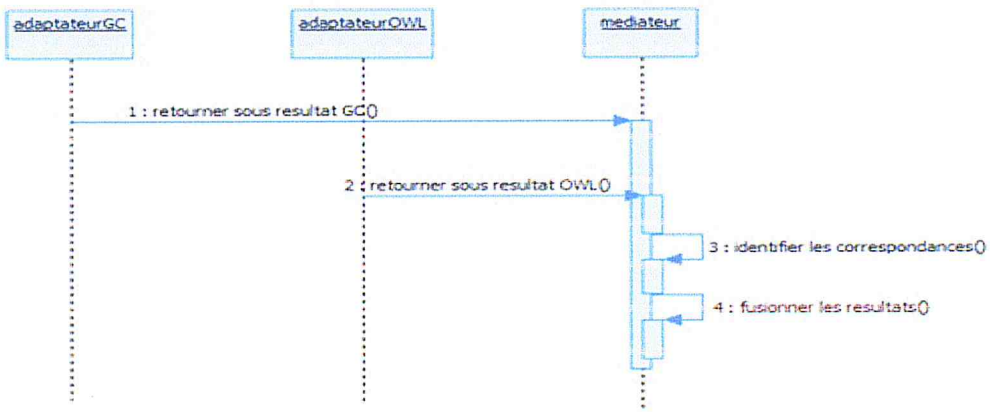

Figure 34: diagramme de séquence fusionner les sous résultats.

4.3 Diagrammes d'activités

Les diagrammes d'activités permettent de mettre l'accent sur les traitements. Ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.

✓ Diagramme d'activité annotation des concepts

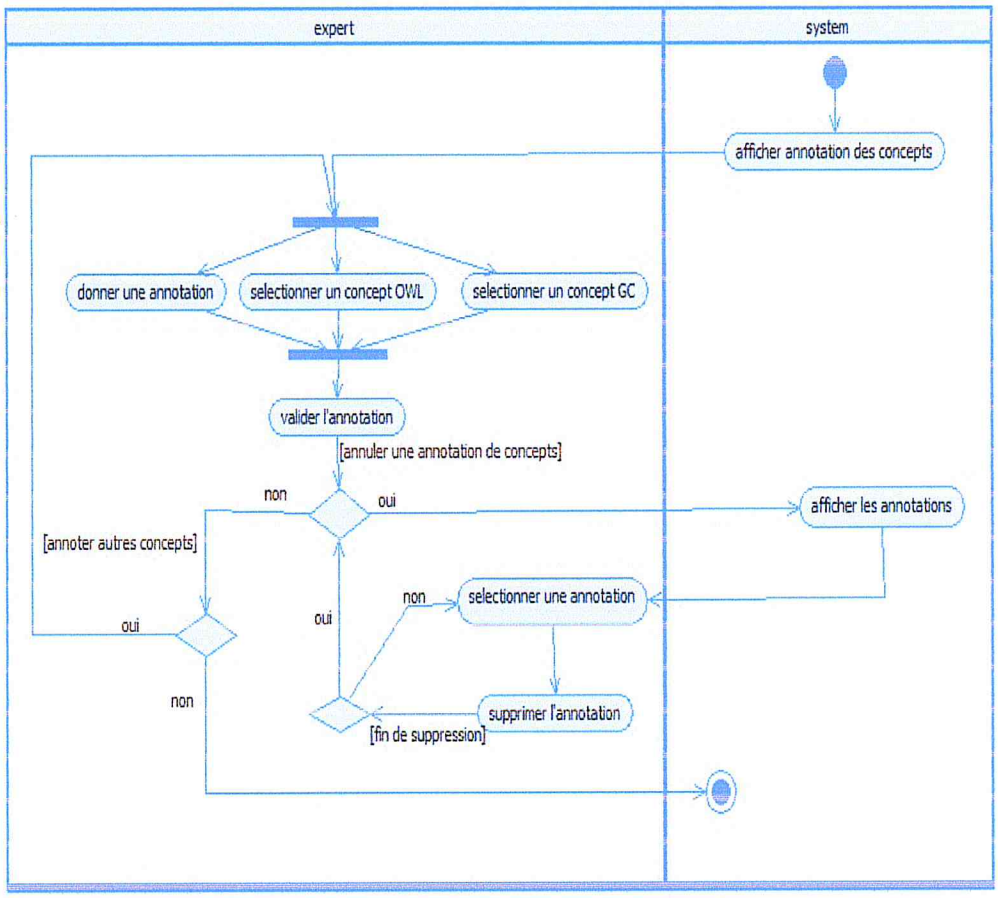

Figure 35: diagramme d'activité annotation des concepts.

✓ Diagramme d'activité annotation des relations

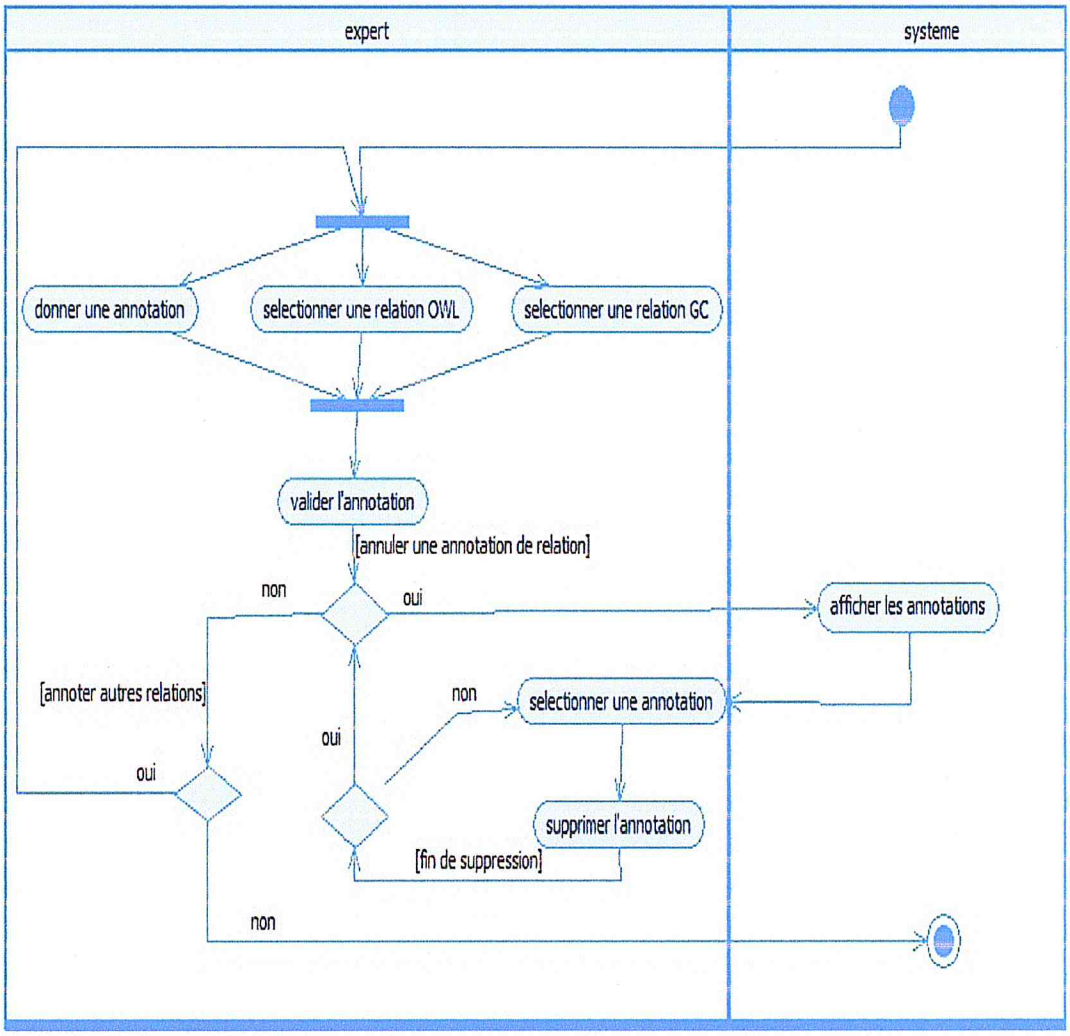

Figure 36: diagramme d'activité annotation des relations.

✓ Diagramme d'activité création du schéma global

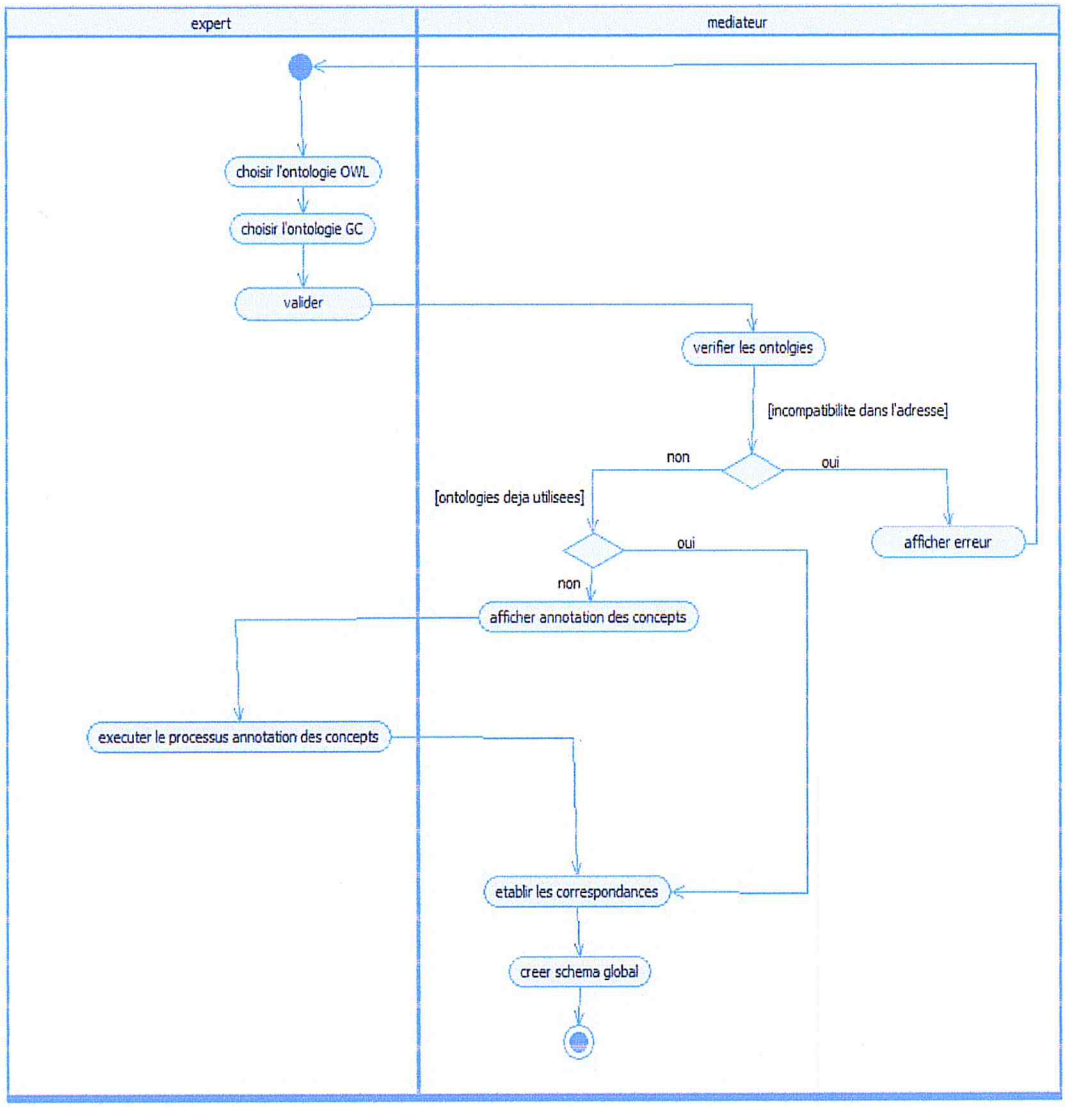

Figure 37: diagramme d'activité création du schéma global.

✓ Diagramme d'activité lancement d'une requête

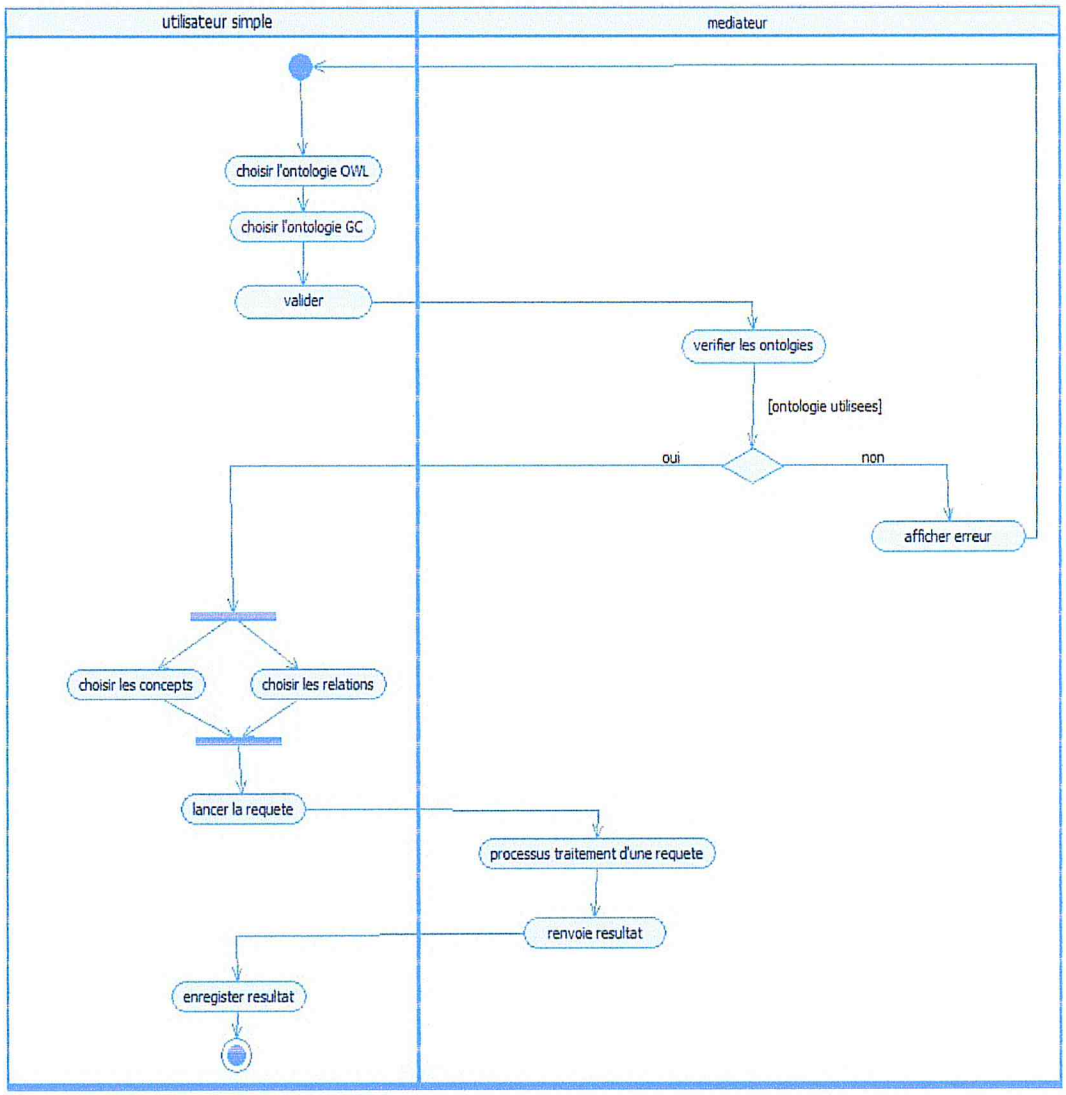

Figure 38: diagramme d'activité lancement d'une requête.

✓ Diagramme d'activité traitement d'une requête

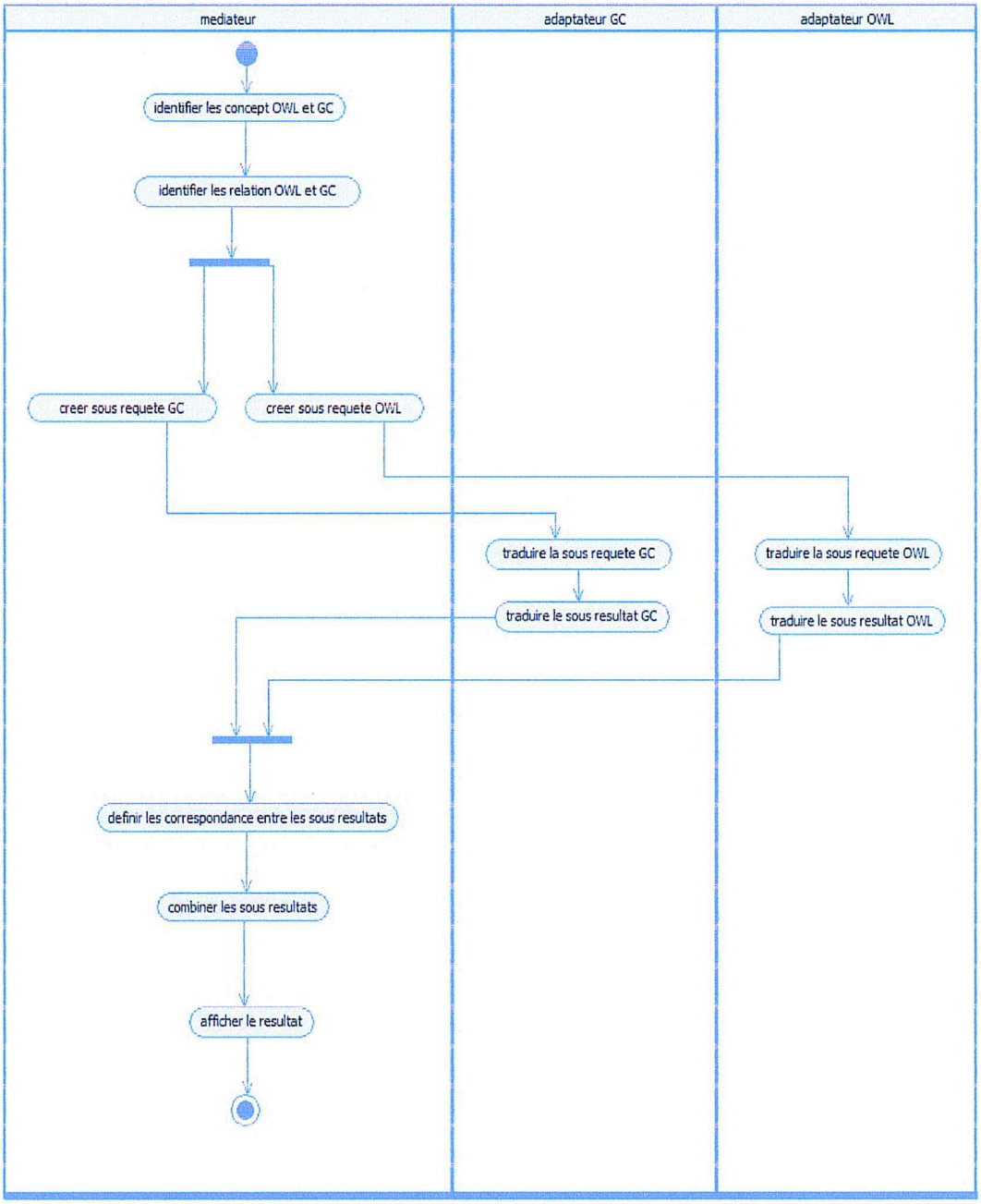

Figure 39: diagramme d'activité traitement d'une requête.

5 Conclusion

L'analyse ainsi présentée nous a permis :

- D'acquérir une expérience dans le domaine de l'intégration des sources hétérogènes
- De comprendre plus profondément la notion de médiateur et de l'approche GAV ;
- De détailler les objectifs que nous avons fixés au départ ;
- D'avoir tous les ingrédients pour commencer la phase conception.

Chapitre IV
conception

1 Introduction

La conception a pour but de définir de façon très précise les fonctions du logiciel, à partir des besoins exprimés et des contraintes générales définies en phase d'analyse et de spécification des besoins.

Dans cette partie nous allons décrire la solution système, en se basant sur le diagramme de classe et nous terminerons par l'architecture de notre Système.

2 Diagramme de classes

Après avoir présenté les différents cas d'utilisation. Nous allons utiliser un diagramme de classe pour montrer les différentes classes du système concerné par l'étude. Ceci permettra aussi, la visualisation des relations entre ces classes.

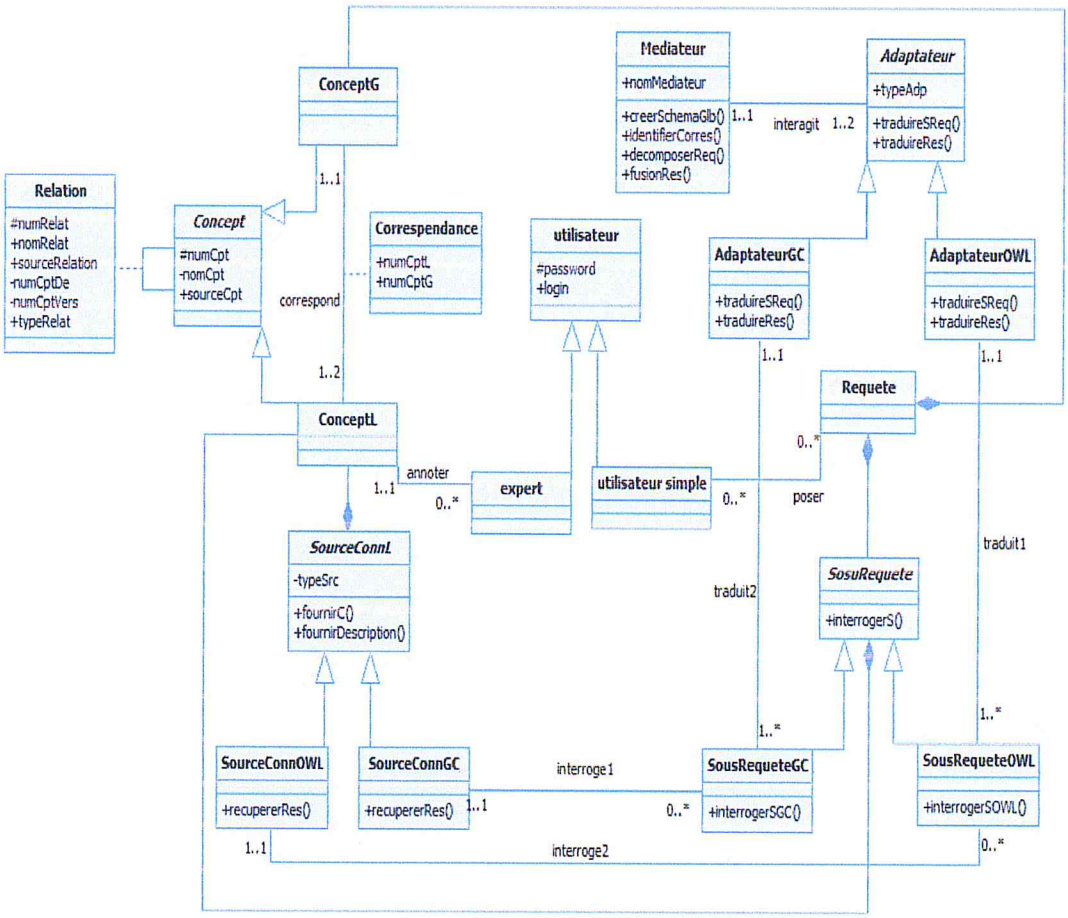

Figure 40: diagramme de classes.

2.1 Description des classes existantes

A ce niveau, nous spécifions les classes et leurs descriptions.

Classe Mediateur	
La classe regroupe toutes les actions pour l'interrogation et l'intégration	
Opération	**Signification**
+creerSchemaGlb	Génère une ontologie OWL à partir des concepts globaux
+ decomposerReq()	Décompose la requête en sous requête.
+ fusionRes()	Fusionne les résultats des sous requêtes
+ identifierCorres()	Identifie les correspondances entre les concepts pour permettre la fusion et la décomposition

Tableau 9 : Description de la classe Mediateur.

Classe ConceptG
Une classe qui hérite de la classe abstraite Concept. Elle représente les concepts du schéma global

Tableau 10 : Description de la classe ConceptG.

Classe ConceptL
Une classe qui hérite de la classe abstraite Concept. Elle représente les concepts des schémas locaux

Tableau 11 : Description de la classe ConceptL.

Classe Concept		
Attribut	**Type**	**Signification**
# numCpt	Int	Identifiant du concept
+nomCpt	String	Le concept
+sourceCpt	String	La source ou se trouve le concept

Tableau 12 : Description de la classe Concept.

Classe SourceConnL		
Attribut	**Type**	**Signification**
+ type	String	Type de la source

Tableau 13 : Description de la classe SourceConnL.

Classe SourceOWL	
Opération	**Signification**
+ recupererRes()	Recupere le resultat de la source OWL

Tableau 14 : Description de la classe SourceOWL.

Classe SourceGC	
Opération	**Signification**
+ recupererRes()	Recupere le resultat de la source GC

Tableau 15 : Description de la classe SourceGC.

Classe Adaptateur		
Une classe abstraite		
Attribut	**Type**	**Signification**
+ nomAdp	String	Nom de l'adaptateur.
Opération		**Signification**
+traduireSReq()		Traduit la sous requete
+ traduireSReq()		Traduit le sous resultat.

Tableau 16 : Description de la classe Adaptateur .

Classe AdaptateurGC	
Opération	**Signification**
+ traduireSReq()	Traduit la sous requete GC vers le schema source GC
+ traduireSReq()	Traduit le sous resultat GC vers le

	schema local GC du mediateur

Tableau 17 : Description de la classe AdaptateurGC.

Classe AdaptateurOWL	
Opération	**Signification**
+ traduireSReq()	Traduit la sous requete OWL vers le schema source OWL
+ traduireSReq()	Traduit le sous resultat OWL vers le schema local OWL du mediateur

Tableau 18 : Description de la classe AdaptateurOWL.

Classe Requete		
La classe regroupe toutes les concepts vertuels spécifique a une requête global		
Attribut	**Type**	**Signification**
+ conceptV	Set	Ensemble des concepts de la requête

Tableau 19 : Description de la classe Requete.

Classe SReqOWL		
La classe represente la sous requete OWL avec ces concepts et ses opérations		
Attribut	**Type**	**Signification**
+ conceptLOWL	Set	Ensemble des concepts locaux OWL
Opération		**Signification**
+ interrogerSOWL()		Interroger la source OWL

Tableau 20 : Description de la classe SReqOWL.

Classe SReqGC		
La classe represente la sous requete GC avec ces concepts et ses opérations		
Attribut	**Type**	**Signification**
+ conceptLGC	Set	Ensemble des concepts locaux GC
Opération		**Signification**
+ interrogerSGC()		Interroger la source GC

Tableau 21 : Description de la classe SReqOWL.

3 L'architecture générale du système

Comme nous l'avons déjà cité ou par avant notre système se compose de trois couches.

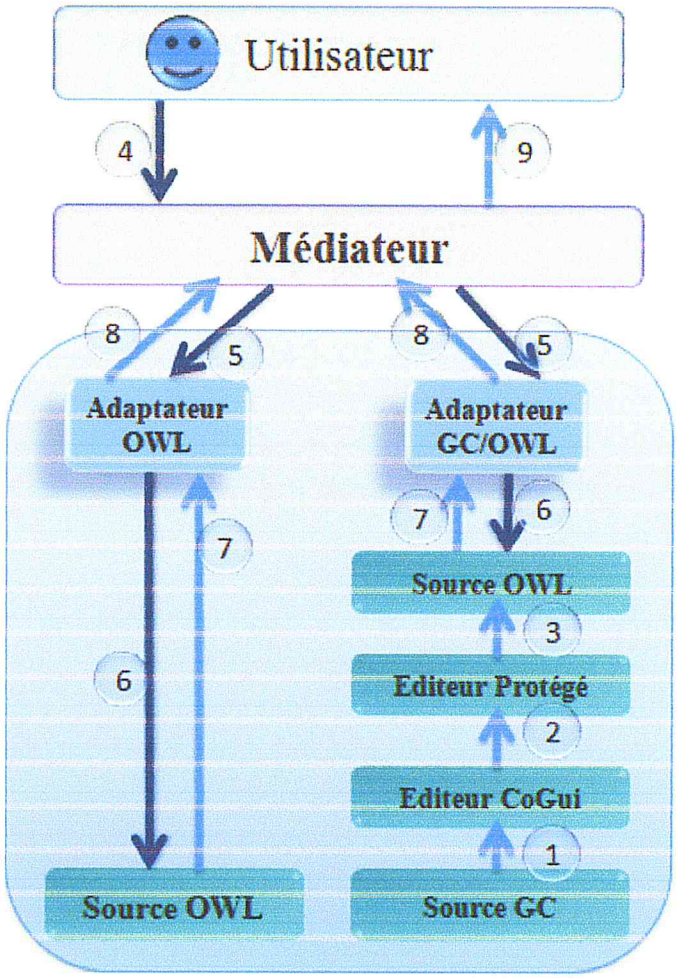

Figure 41 :L'architecture globale du système.

1 : CoGui exporte la source GC sous format RDF.
2 : Protégé enregistre le résultat RDF sous extension OWL.
3 : Obtention d'une ontologie OWL résultante de la traduction d'une ontologie GC.
4 : Lancement d'une requête posée sous format du schéma global.
5 : Envoie des sous requêtes globales OWL et GC
6 : Interrogation des sources de connaissances locales en envoyant des sous requêtes locales.
7 : Retours des sous résultats locaux.
8 : Retours des sous résultats traduits sous format du schéma global.
9 : Renvoie d'une réponse uniforme sous format du schéma global.

3.1 Description des modules de l'architecture

Couche Utilisateur : Représente l'application cliente qui donne l'accès aux utilisateurs à notre système, qui permet d'une part à l'expert d'annoter les concepts et relations différents ou poser des requêtes, et d'autre part permet

aux utilisateurs simples de poser leurs requêtes et recevoir le résultat en donnant l'impression d'interroger une seule source de connaissance, à la fin elle donne à l'utilisateur le choix d'enregistrer le résultat obtenu ou juste le consulter et quitter le système.

Couche médiateur : représente la partie intermédiaire entre l'utilisateur et les sources de connaissances. Avant de commencer à détailler les différents modules de cette couche on doit préciser tout d'abord que sera le langage de notre schéma global, après comparaison des deux langages des schémas sources on est arrivé à la conclusion que le langage OWL est beaucoup plus riche et plus expressif que le Graphe Conceptuel et aussi à la traduction de la source Graphe Conceptuel en OWL on garde l'intégralité de l'ontologie mais si on essai le contraire plusieurs informations qui pourrons êtres pertinentes pour notre utilisateur peuvent être perdu et c'est sur ces critères la qu'on a fini par choisir OWL comme langage pivot de notre system.

Apres le choix du langage du schéma global et après avoir étudier les approches utilisées dans l'intégration au premier chapitre nous avons jugé que le système d'intégration que nous proposons sera suivant l'approche GAV car les deux autres approches ne suscitent pas notre intérêt vue que leur implémentation est complexe dans un système d'intégration d'autant plus que nos sources sont prédéfinies à l'avance et qu'on n'a pas besoin de la notion de flexibilité supportée par l'approche LAV. Et de ce fait notre système portera le nom de MedGAV qui est l'acronyme de **Med**iation par l'approche **GAV.**

Maintenant qu'on connait le langage pivot de notre système d'intégration et l'approche suivi pour sa réalisation on peut détailler les différents modules réalisés par cette couche:

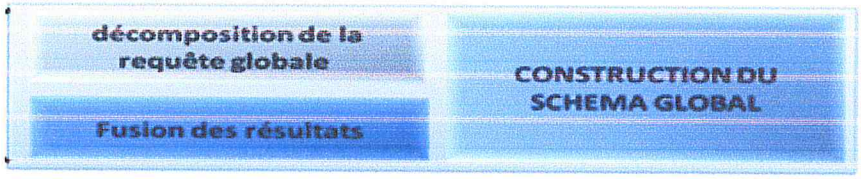

Figure42 :Modules du médiateur.

Le module *« **Création du schéma global** »* ce module est chargé de la création de l'ontologie global qui regroupe les concepts globaux et les relations globales en plus du chargement de la base de données. Ceci peut mieux s'expliquer avec l'algorithme de la construction de notre schéma global suivant :

Tout d'abord, l'expert annote les concepts et les relations différents, en leur donnant une annotation pour chaque deux concepts (respectivement deux relations) qui ont la même signification mais une syntaxe différente. Avec cette solution proposée par MedGav nous traitons l'hétérogénéité sémantique qui peut s'avérée handicapante pour le système d'intégration si elle n'est pas traitée.

Maintenant que tout est prêt, la construction du schéma global peut commencer :

Apres l'annotation des concepts et relations notre système récupère tout les concepts globaux et les relations globales, pour faciliter la réalisation de ce module on il sera diviser en deux phases, une qui traite les concepts et l'autre les relations, pour celle qui traitera les concepts son algorithme général sera comme suit :

Pour chaque concept global on voit s'il n'est pas déjà ajouté ensuite on l'ajoute dans le schéma global ensuite on voit1) si c'est une annotation, 2) s'il

existe dans les deux sources (avec la même syntaxe), 3) existe dans l'une des sources : donc plusieurs cas se présentent :

1) Est dans annotation : on récupère les fils des concepts locaux correspondant au concept global qu'on traite. Ensuite pour chaque fils on vérifie s'il est une annotation, existe juste dans une des sources ou existe avec la même syntaxe dans les deux sources, dans le cas où c'est une annotation on vérifie qu'on ne l'a pas déjà l'ajoute autant que fils du concept global qu'on traite, s'il est dans l'une des deux sources et qu'il n'est pas annoté on l'ajoute comme fils du concept global qu'on traite toujours on vérifie s'il n'est pas déjà ajouté, si il est annoté on ajoute son annotation comme fils, et dans le cas où il existe dans les deux sources avec la même syntaxe on ajoute l'un des concepts locaux comme fils. Ceci est dans le cas où le concept global qu'on traite est dans les annotations. Le cas où il n'est pas dans annotation on vérifie les deux cas restants :

2) Existe dans les deux sources avec la même syntaxe: on récupère ses fils des deux sources locales et on leur fait les même tests qu'on a fait pour les fils du concept appartenant aux annotations, donc nous remarquons que le traitement des fils du concept se répète quel que soit le concept global (annoté, appartient à annotation ou le même dans les deux sources) donc on leur fera une fonction qu'on appellera à chaque fois qu'on veut traiter les fils du concept.

3) Le troisième cas c'est quand il existe dans l'une des sources :on récupère ses fils dans cette source locale, et on traite ses fils ensuite selon le cas qu'on rencontre soit on ajoute le fils directement soit on ajoute son annotation.

Et comme ça on aurait fini la première phase qui constitue l'ajout des concepts au schéma global ainsi que la hiérarchie des concepts fils. Maintenant il est temps d'entamer la deuxième phase qui constitue l'ajout des relations dans le schéma global en respectant leur hiérarchie sans oublier leur rang et leur domaine. Pour chaque relation globale on vérifie si la relation n'a pas déjà été ajoutée au

schéma global on l'ajoute, ensuite on vérifie si cette relation 1) est une annotation,2) existe dans une des sources,3) existe dans les deux sources (la même syntaxe).

1) Dans le cas où c'est une annotation: on l'ajoute comme relation dans le schéma global, on récupère les relations locales correspondant à la relation qu'on traite, on récupère leurs domaines et rangs, ensuite on compares les rang de chaque source si les deux rangs sont identiques (syntaxiquement) on désigne l'un de ces rangs comme rang global de la relation.si ils sont différents c'est qu'ils ont forcément la même annotation donc on ajoute leur annotation comme rang global de la relation, le domaine subira le même traitement que le rang. Ensuite on récupère ses fils, même traitement que pour la relation mère, on passe par les trois conditions et ainsi de suite. dans le cas où ce n'est pas une annotation, on vérifie :

2) existe dans une des sources: on récupère le domaine et rang de la source où existe la relation et on les traite de la même manière suivie dans le cas précédent. Et par la suite on traite les relations filles.

3) Si la relation n'est pas une annotation et existe dans les deux listes c'est que forcement elle a la même syntaxe, on récupère les rangs et domaines dans les deux sources, on compare les rangs (et les domaines respectivement) et test s'ils sont les mêmes on ajoute l'un d'eux comme rang global de la relation (domaine global de la relation respectivement) sinon on ajoute leur annotation comme rang global (domaine globale respectivement).et on traite les fils comme dans les cas précédant.

De ce fait on aurait construit la deuxième partie de notre schéma global qui constitue les relations, leurs rangs et domaines en respectant la hiérarchie existante dans les schémas locaux.

Automatiquement après la création du schéma global le médiateur remplis notre base de donnée avec les concepts locaux, les concepts globaux et la correspondance entre chaque concept global et concept local, ceci facilitera

ensuite le déroulement des deux modules, décomposition de la requête globale et la fusion des résultats.

Dans ce qui suit nous présentons quelques impressions écran de notre ontologie globale et des tables de la base de données rempli par le médiateur :

	num_concept	nom_concept	source_concept
	4	femme	OWL/GC
	1	Human	OWL/GC
	3	homme	OWL/GC
	2	Thing	OWL/GC

Figure43 :Table concepts globaux.

	num_concept	nom_concept	source_concept
	8	Man	GC
	7	Woman	GC
	6	Human	GC
	5	Top	GC
	4	Female	OWL
	3	Human	OWL
	2	Male	OWL
	1	Thing	OWL

Figure 44 :Table concepts locaux.

	num_conceptL	num_conceptG
	7	4
	4	4
	8	3
	2	3
	5	2
	1	2
	6	1
	3	1

Figure45 : Table correspondance.

Le module « ***décomposition de la requête globale*** » ce module traite la requête remise par l'utilisateur sous format du schéma global qui sera décomposé en sous requêtes OWL et sous requêtes GC écrites sous format du schéma global du médiateur. En définissant les concepts globaux et les relations globales qui existent dans la source OWL et ceux qui existent dans la source GC.

Le module « ***fusion des résultats*** » : ce modules est chargé de combiner les résultats retournés par les adaptateurs OWL et GC et de fournir à l'utilisateur une réponse homogène et uniforme écrite sous format du schéma global.

Couche sources: A ce niveau se situe les adaptateurs OWL et GC qui gèrent chacun ses propres modules.

Les adaptateurs OWL et GC sont composés de deux modules chacun :

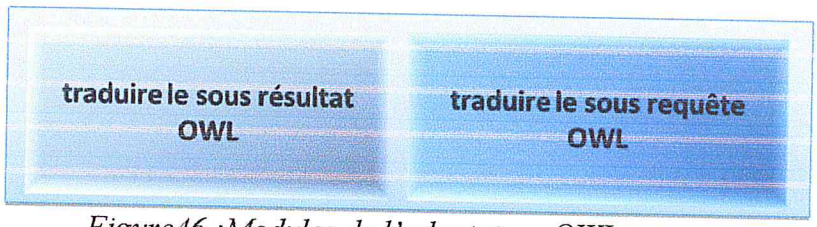

Figure46 :Modules de l'adaptateur OWL

« Traduire la sous requête OWL » : ce module permet de définir les valeurs des concepts et des relations choisis dans la source OWL est de les remplacer pour pouvoir interroger les sources locales

« Traduire le sous résultat OWL » qui permettent la traduction du sous résultat local venant de la source OWL sous forme du schéma global OWL du médiateur.

Respectivement :

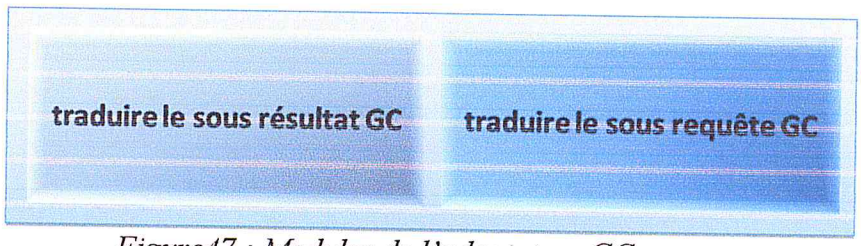

Figure47 : Modules de l'adaptateur GC

« Traduire la sous requête GC » **:** ce module permet de définir les valeurs des concepts et des relations choisis dans la source GC est de les remplacer pour pouvoir interroger les sources locales

« Traduire le sous résultat GC » qui permettent la traduction du sous résultat local venant de la source OWL sous forme du schéma global OWL du médiateur.

4 Conclusion

Apres avoir définie l'architecture du système avec ses composants nous allons adopter l'approche GAV pour la réalisation de notre système d'intégration. En ce qui suis nous allons faire des testes sur une ontologie OWL crée avec Protégé et une ontologie GC créer avec l'éditeur CoGui est exportée sous format OWL.

Chapitre V
Implémentation et test

1 Introduction :

Dans ce chapitre nous allons définir les outils de développement choisis pour l'implémentation de notre système. Ensuite, nous présentons notre application.

2 Le langage de programmation choisi

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

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 besoin pour nos ontologies comme Jena et le langage de requête SPARQL.

3 Les outils utilisés

3.1 Protégé

Protégé est un éditeur d'ontologie et un composant de base de connaissance open source basé sur java.

Protégé permet la visualisation et la manipulation d'ontologies qui peuvent être exportés sous différents formats incluant RDF(S), OWL, et XML schéma.

L'éditeur Protégé-OWL permet aux utilisateurs de construire des ontologies pour le Web sémantique, en particulier dans le langage du W3C le

OWL. "Une ontologie OWL peut inclure des descriptions des classes, des propriétés et de leurs instances.

Protégé est supporté par une forte communauté de développeurs et d'académiques, gouvernements et des utilisateurs en entreprise, qui utilisent des solutions Protégé pour des solutions dans divers domaines comme la biomédecine, la modélisation d'entreprise...etc.[SCB ,11]

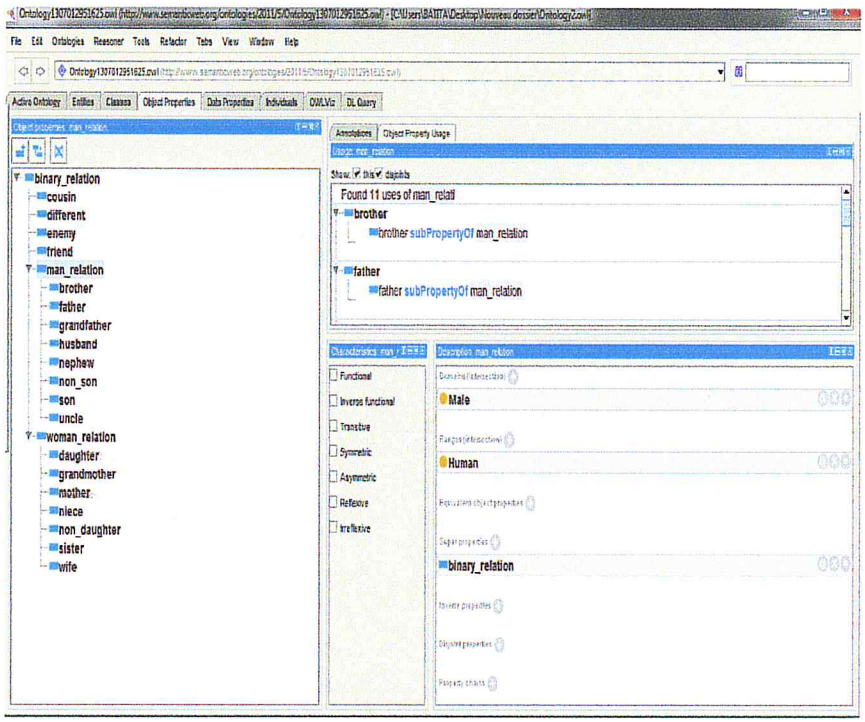

Figure 48 : l'ontolgie OWL créer par l'éditeur protégé.

3.2 CoGui

Développé par l'équipe de RCR au LIRMM (http://www.lirmm.fr/RCR/) CoGui est un outil développé en java (peut être utilisé sur toute plate-forme sur les quelles il est installé un environnement d'exécution Java) pour la construction des bases de connaissances graphes conceptuels représentées sous format COGXML. [LIR,11]

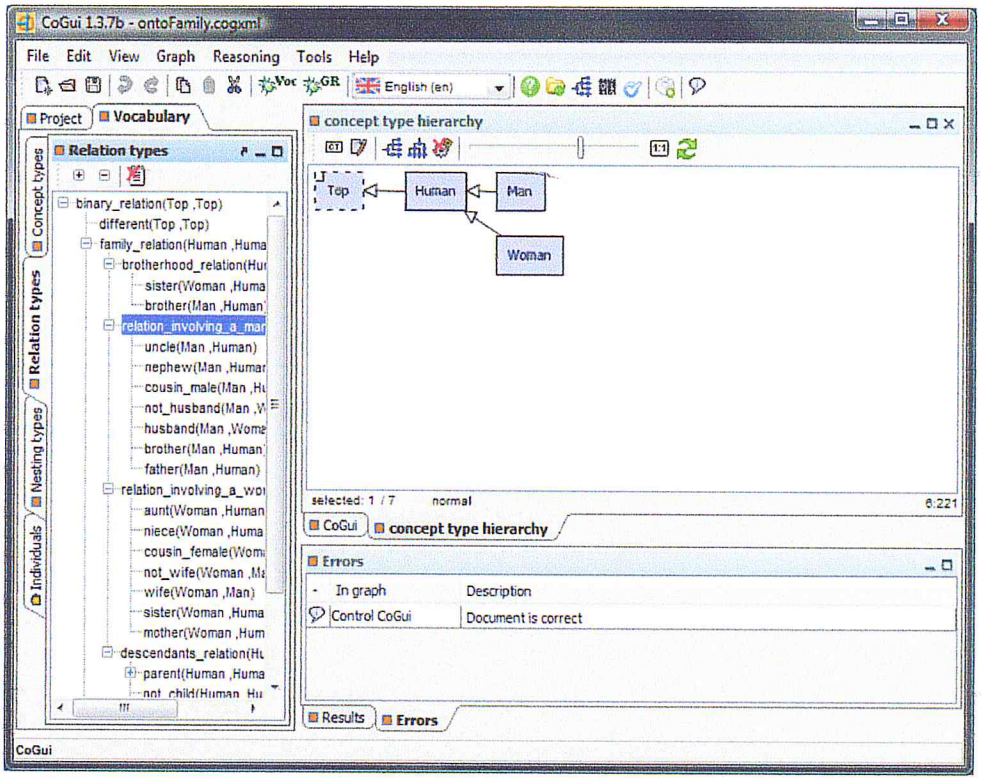

Figure 49 : ontologie créer par l'outil CoGui.

3.3 SPARQL

SPARQL est un langage de requête et un protocole qui permet l'accès en RDF.il a été crée par le W3C RDF Data Access Working Group.

Le langage de requête SPARQL peut être utilisé pour avoir des informations à partir d'un RDFs.

Il fournit certaines facilités l'extraction des informations sous forme d'URIs. [W3C, 08]

```
String urlt = "file:"+chemGC;
OntModel model = ModelFactory
        .createOntologyModel(OntModelSpec.OWL_DL_MEM_RDFS_INF);
InputStream reader = FileManager.get().open(urlt);
if (reader == null) {
    throw new IllegalArgumentException("File: " + urlt + " not found");
}
model.read(reader, "", "RDF/XML");
String queryString =
          "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
        + "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>" +
                "PREFIX owl:<http://www.w3.org/2002/07/owl#>"
        + " SELECT ?x ?fname WHERE {?x  rdfs:label  ?fname}";
Query query = QueryFactory.create(queryString);
Dataset dataset = DatasetFactory.create(model);
QueryExecution qexec = QueryExecutionFactory.create(query, dataset);
ResultSet resultset = qexec.execSelect();
while (resultset.hasNext()) {
    QuerySolution row = (QuerySolution) resultset.next();
    RDFNode c = row.get("fname");
    nomConcept.add(c.toString());
    RDFNode cc = row.get("x");
    codeConcept.add(cc.toString());
}
```

Figure50 : requête d'extraction de classes et de relation à partir de l'ontologie GC.

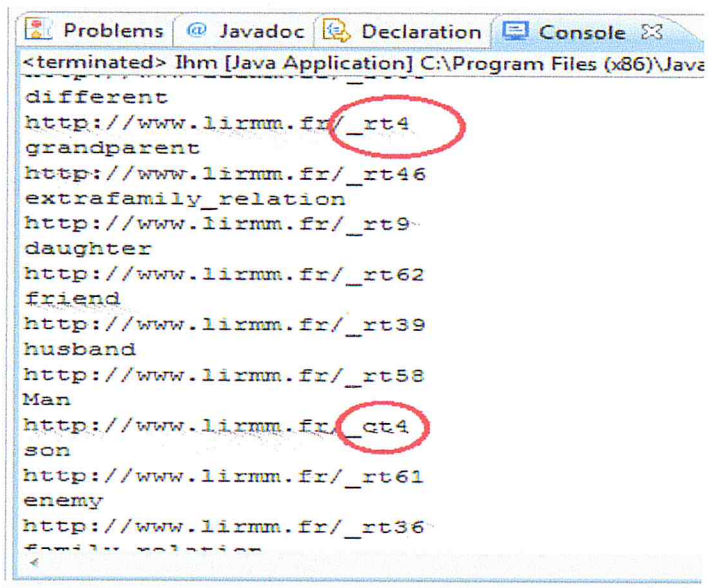

Figure51 : exemple de résultat de la requête SPARQL sous format d'URIs

3.4 Jena

Jena est un API java qui peut être utilisé pour créer et manipuler des graphes RDF. Le package pour RDF est com.hp.hpl.jena.rdf.model, comme il permet de traiter des ontologie OWL.[JEN, 11]

```
//**\//**\//**\//**\//**\//**\//**\//**\//**\//**\//**\//*
public static ArrayList<String> getClassesOWL(){
    String urlt = "file:"+chemOWL;
    String urit = "http://www.owl-ontologies.com/generations.owl#";
    OntModel model = importOnt(urit, urlt);
    ExtendedIterator<OntClass> in = model.listNamedClasses();
    ArrayList<String> result=new ArrayList<String>();
    while (in.hasNext()) {
        String a=in.next().getLocalName().toString();
        result.add(a);
    }
    System.out.println(result);
 return result;
}
//**\//**\//**\//**\//**\//**\//**\//**\//**\//**\//**\//*
```

Figure 52 : la méthode getClassesOWL qui permet d'extraire les concepts OWL.

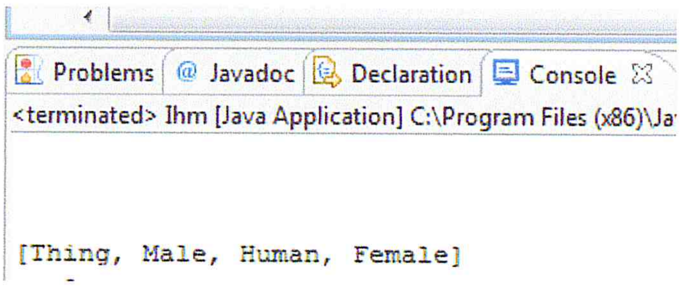

Figure53 : résultat de la méthode getClassesOWL.

4 Présentation du système MedGav

Pour cette partie nous présentons les interfaces qui permettent aux utilisateurs d'avoir des interactions avec le système

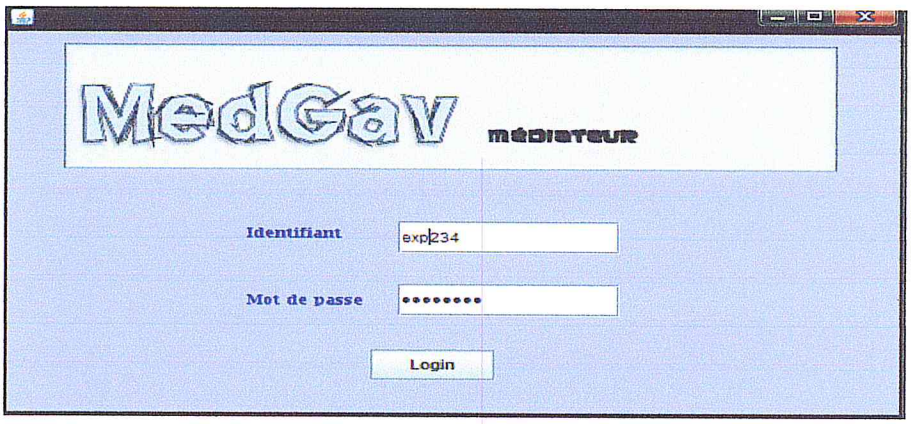

Figure54 : Authentification des utilisateurs.

L'utilisateur doit choisir l'ontologie OWL et l'ontologie GC

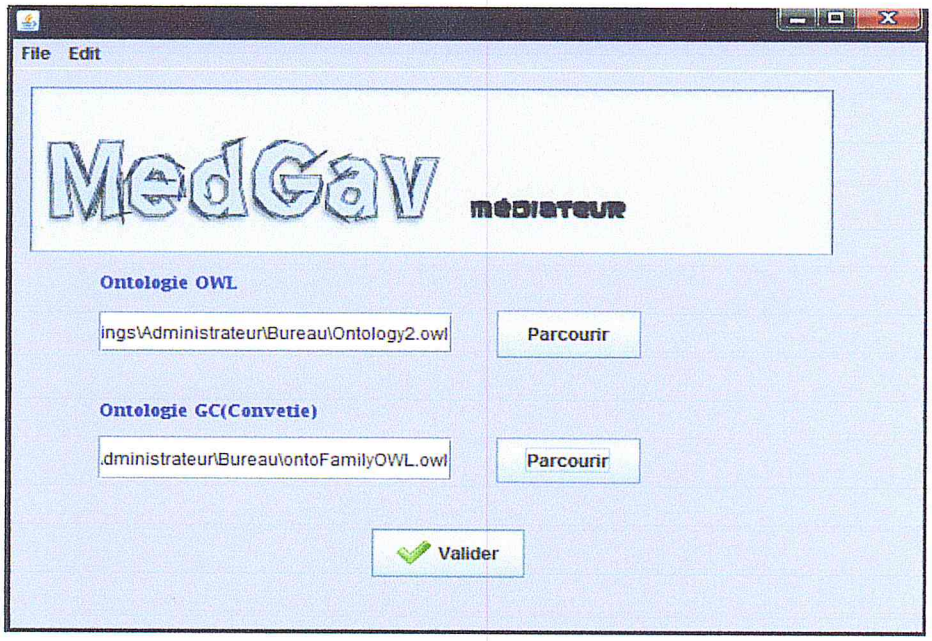

Figure 55 : interface de choix d'ontologie.

L'utilisateur est un expert, il sélectionne les concepts qui les juge avoir la même signification et les annote.

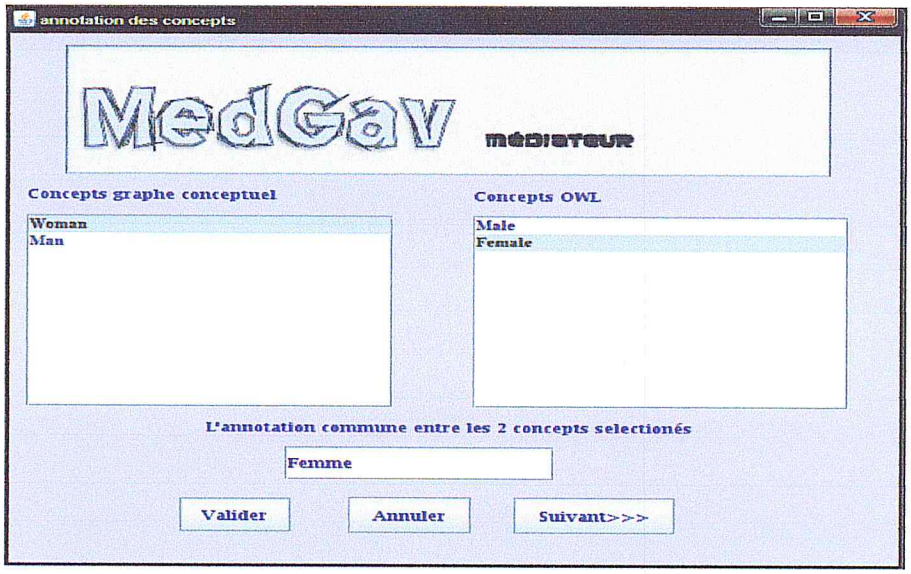

Figure 56 : fenêtre d'annotation de concepts.

L'expert doit sélectionner les relations qui les juge avoir la même signification et les annoter.

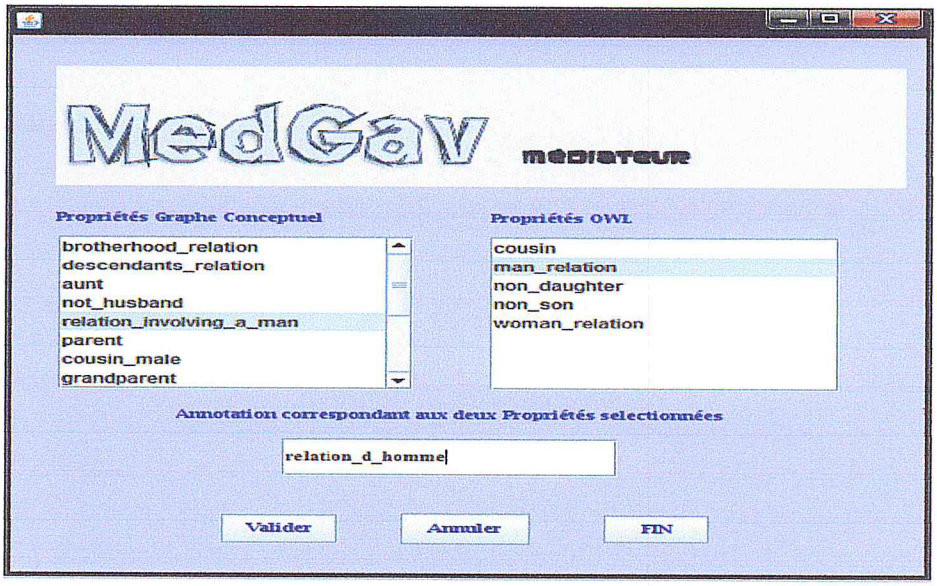

Figure 57 : fenêtre d'annotation de relations.

L'expert a la possibilité d'annuler certaines annotations déjà effectuées pour les concepts.

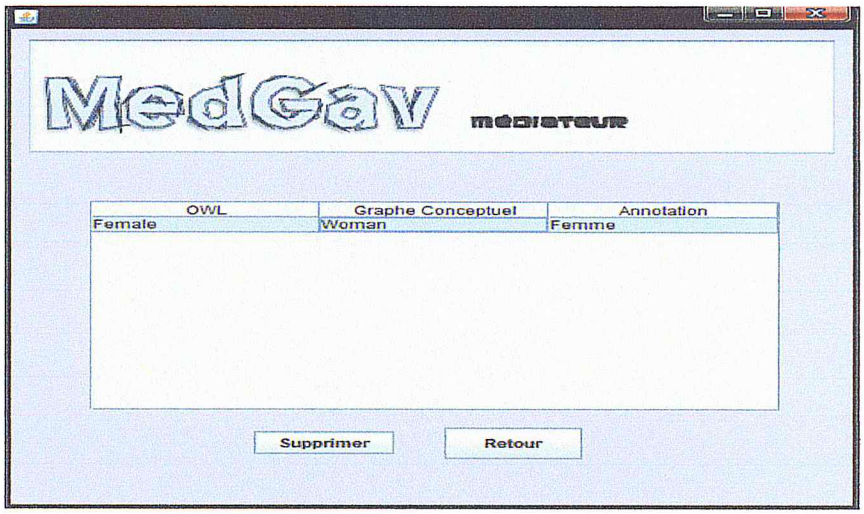

Figure 58 : fenêtre d'annulation d'annotation de concepts.

Le schéma global créer par notre système regroupe tous les concepts et les relations des deux ontologies.

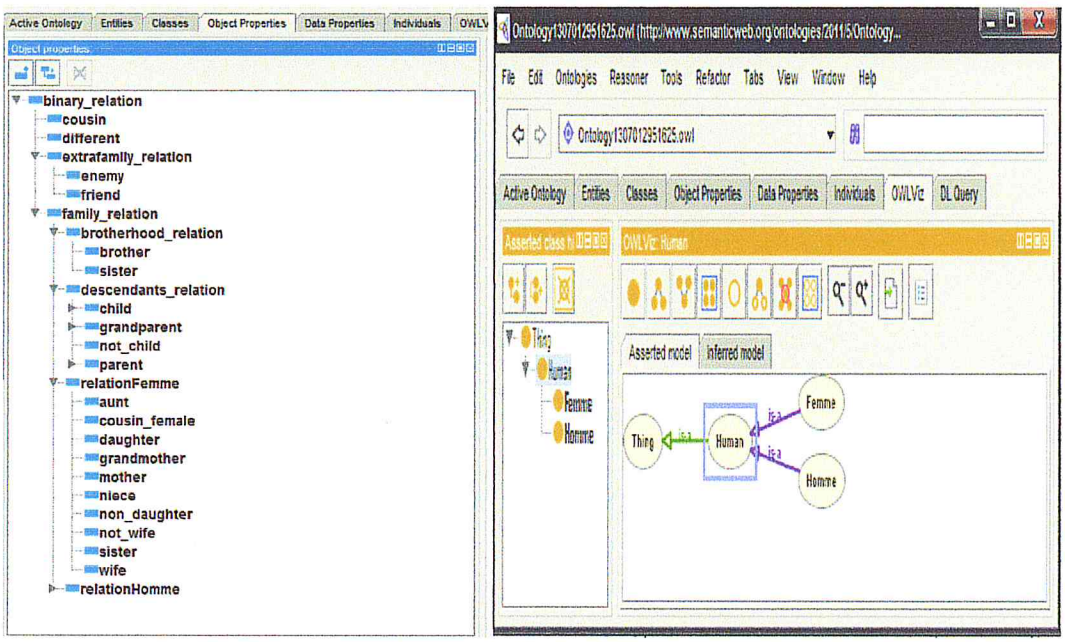

Figure 59 :Ontologie Globale générée par le médiateur.

L'utilisateur simple choisit un concept et une relation comme il peut donner une instance. Dans cet exemple l'utilisateur cherche tous les humains qui ont « fatima » pour sœur.

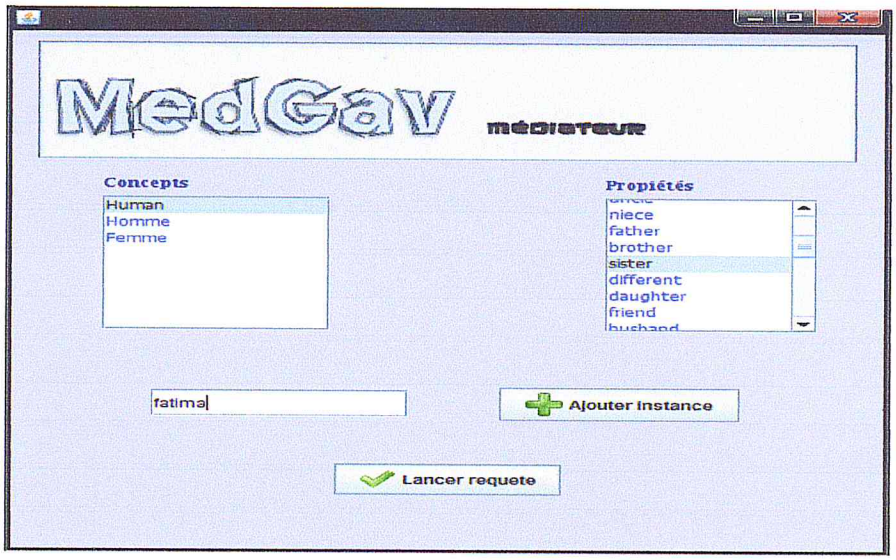

Figure 60 : fenêtre de requête.

Résultats de l'exemple chercher tous les humains qui ont « fatima » pour sœur.

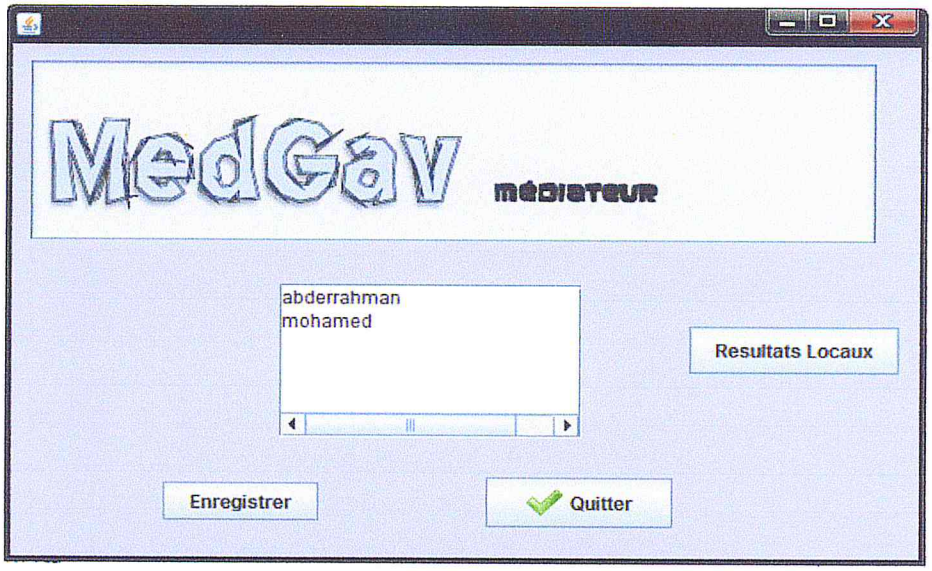

Figure 61 : fenêtre de résultat global.

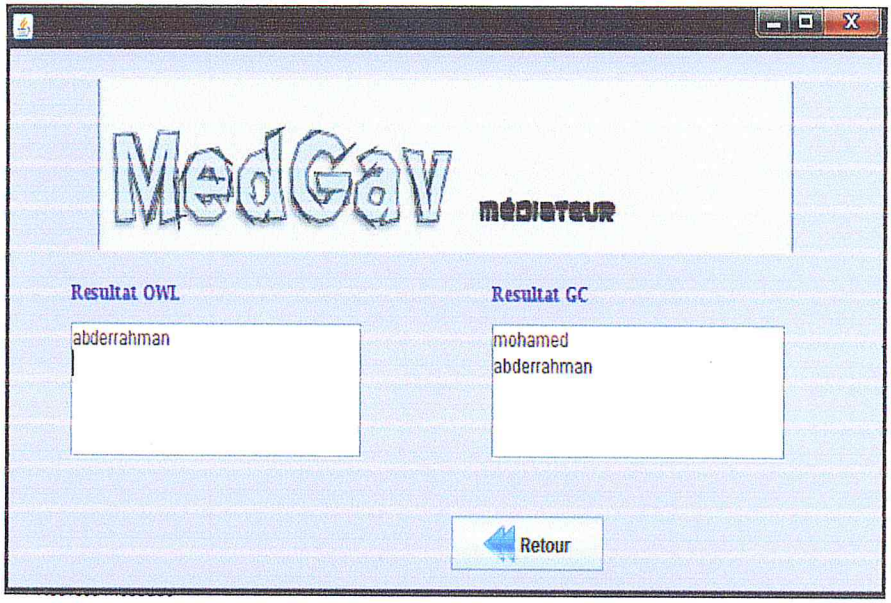

Figure 61 : fenêtre des résultats locaux.

5 Conclusion

Pendant la réalisation de l'application nous sommes arrivé à produire un système d'intégration de sources de connaissances hétérogènes via l'approche médiateur.

Conclusion générale

Au fil des années, de grandes quantités d'informations ont été crées sur les systèmes d'informations des entreprises. La diversité de ces sources de données produites par des systèmes hétérogènes rend difficile leur exploitation et leur mixage pour générer des informations pertinentes. Vu le développement des technologies de l'information, une nouvelle génération d'applications apparaît et consiste à fournir une interface pour l'accès à ces sources de données distribuées et hétérogènes.

Le thème traité par ce mémoire est l'intégration des sources de connaissances hétérogène : OWL et GC par l'approche médiateur. Beaucoup de travaux ont été réalisés dans ce domaine en se basant sur cette approche. Nous avons commencé par une étude détaillée du problème d'intégration des données en suite nous avons étudié la syntaxe et la sémantiques des langages de source de connaissances hétérogènes ; ceci nous a permis d'adopter l'approche GAV qui est une approche d'intégration des sources hétérogènes basée sur la notion de schéma global crées à partir des schémas sources ce schéma global est représenté par le modèle pivot OWL que nous avons choisit , ce modèle a une puissance d'expression des sources de connaissances par rapport au modèle GC.

Nous avons ensuite passé par l'expression des besoins et l'analyse cette partie nous a permis de bien déterminer nos objectifs. Enfin nous avons réalisé une application assistée par un expert, ce dernier donne des annotations aux concepts et aux relations existants dans nos ontologies ce qui permet d'éliminer la notion d'hétérogénéité sémantique et de créer automatiquement un schéma global qui représente une vue générale sur les sources locales et qui permet d'intégrer les sources de connaissances hétérogènes. Notre teste était appliqué sur des ontologies de domaine « Famille ».

Cette étude nous a permis de découvrir une nouvelle génération de système et de modèle pour gérer et présenter les sources hétérogènes et de résoudre les conflits et les problèmes dûs à cette hétérogénéité.

Annexes

```
<?xml version="1.0"?>


<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY Ontology1307012951625 "http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#" >
]>


<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#"
     xml:base="http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:Ontology1307012951625="http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
```

```
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#binary_relation -->

    <owl:ObjectProperty rdf:about="#binary_relation">
        <rdfs:range rdf:resource="&owl;Thing"/>
        <rdfs:domain rdf:resource="&owl;Thing"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#brother -->

    <owl:ObjectProperty rdf:about="#brother">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#man_relation"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#cousin -->

    <owl:ObjectProperty rdf:about="#cousin">
        <rdfs:domain rdf:resource="#Human"/>
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#binary_relation"/>
    </owl:ObjectProperty>



    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#daughter -->

    <owl:ObjectProperty rdf:about="#daughter">
        <rdfs:domain rdf:resource="#Female"/>
```

```
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#different -->

    <owl:ObjectProperty rdf:about="#different">
        <rdfs:range rdf:resource="&owl;Thing"/>
        <rdfs:domain rdf:resource="&owl;Thing"/>
        <rdfs:subPropertyOf rdf:resource="#binary_relation"/>

    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#enemy -->

    <owl:ObjectProperty rdf:about="#enemy">
        <rdfs:domain rdf:resource="#Human"/>
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#binary_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#father -->

    <owl:ObjectProperty rdf:about="#father">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#man_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#friend -->

    <owl:ObjectProperty rdf:about="#friend">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#binary_relation"/>
    </owl:ObjectProperty>



<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#grandfather -->

<owl:ObjectProperty rdf:about="#grandfather">
    <rdfs:range rdf:resource="#Human"/>
    <rdfs:domain rdf:resource="#Male"/>
    <rdfs:subPropertyOf rdf:resource="#man_relation"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#grandmother -->

<owl:ObjectProperty rdf:about="#grandmother">
    <rdfs:domain rdf:resource="#Female"/>
    <rdfs:range rdf:resource="#Human"/>
    <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#husband -->

<owl:ObjectProperty rdf:about="#husband">
    <rdfs:range rdf:resource="#Female"/>
    <rdfs:domain rdf:resource="#Male"/>
    <rdfs:subPropertyOf rdf:resource="#man_relation"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#man_relation -->

<owl:ObjectProperty rdf:about="#man_relation">
    <rdfs:range rdf:resource="#Human"/>
    <rdfs:domain rdf:resource="#Male"/>
    <rdfs:subPropertyOf rdf:resource="#binary_relation"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#mother -->

<owl:ObjectProperty rdf:about="#mother">
    <rdfs:domain rdf:resource="#Female"/>
    <rdfs:range rdf:resource="#Human"/>
```

```
        <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#nephew -->

    <owl:ObjectProperty rdf:about="#nephew">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#man_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#niece -->

    <owl:ObjectProperty rdf:about="#niece">
        <rdfs:domain rdf:resource="#Female"/>
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#non_daughter -->

    <owl:ObjectProperty rdf:about="#non_daughter">
        <rdfs:domain rdf:resource="#Female"/>
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#non_son -->

    <owl:ObjectProperty rdf:about="#non_son">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#man_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#sister -->

    <owl:ObjectProperty rdf:about="#sister">



        <rdfs:domain rdf:resource="#Female"/>
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#son -->

    <owl:ObjectProperty rdf:about="#son">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#man_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#uncle -->

    <owl:ObjectProperty rdf:about="#uncle">
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:domain rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#man_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#wife -->

    <owl:ObjectProperty rdf:about="#wife">
        <rdfs:domain rdf:resource="#Female"/>
        <rdfs:range rdf:resource="#Male"/>
        <rdfs:subPropertyOf rdf:resource="#woman_relation"/>
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#woman_relation -->

    <owl:ObjectProperty rdf:about="#woman_relation">
        <rdfs:domain rdf:resource="#Female"/>
        <rdfs:range rdf:resource="#Human"/>
        <rdfs:subPropertyOf rdf:resource="#binary_relation"/>
    </owl:ObjectProperty>
```

```
<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////////
 -->
 <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#Female -->

<owl:Class rdf:about="#Female">
    <rdfs:subClassOf rdf:resource="#Human"/>
</owl:Class>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#Human -->

<owl:Class rdf:about="#Human"/>
-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#Male -->

<owl:Class rdf:about="#Male">
    <rdfs:subClassOf rdf:resource="#Human"/>
</owl:Class>
<!-- http://www.w3.org/2002/07/owl#Thing -->

<owl:Class rdf:about="&owl;Thing"/>
<!--
///////////////////////////////////////////////////////////////////////////////////////
//
// Individuals
//
///////////////////////////////////////////////////////////////////////////////////////
 -->


 <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#abderrahman --

 <Male rdf:about="#abderrahman">
     <rdf:type rdf:resource="&owl;Thing"/>
 </Male>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#fatima -->

 <owl:Thing rdf:about="#fatima">
     <rdf:type rdf:resource="#Female"/>
     <sister rdf:resource="#abderrahman"/>
     <cousin rdf:resource="#hamza"/>
     <friend rdf:resource="#imene"/>
     <sister rdf:resource="#oumelkheir"/>
 </owl:Thing>
 <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#hamza -->

 <Male rdf:about="#hamza">
     <rdf:type rdf:resource="&owl;Thing"/>
 </Male>
 <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#imene -->

 <Female rdf:about="#imene">
     <rdf:type rdf:resource="&owl;Thing"/>
     <friend rdf:resource="#fatima"/>
     <daughter rdf:resource="#mohamed"/>
     <cousin rdf:resource="#nawel"/>
     <sister rdf:resource="#sofiane"/>
 </Female>
<!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#mohamed -->

 <Male rdf:about="#mohamed">
     <rdf:type rdf:resource="&owl;Thing"/>
 </Male>
```

```
        <rdf:type rdf:resource="&owl;Thing"/>
    </Male>


    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#nawel -->

    <Female rdf:about="#nawel">
        <rdf:type rdf:resource="&owl;Thing"/>
    </Female>



    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#oumelkheir -->

    <Female rdf:about="#oumelkheir">
        <rdf:type rdf:resource="&owl;Thing"/>
    </Female>



    <!-- http://www.semanticweb.org/ontologies/2011/5/Ontology1307012951625.owl#sofiane -->

    <Male rdf:about="#sofiane">
        <rdf:type rdf:resource="&owl;Thing"/>
    </Male>
</rdf:RDF>
```

```
<?xml version="1.0"?>


]<!DOCTYPE rdf:RDF [
    <!ENTITY cogui "http://www.lirmm.fr/cogui#" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<rdf:RDF xmlns="http://www.lirmm.fr/cogui#"
     xml:base="http://www.lirmm.fr/cogui#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
]    xmlns:cogui="http://www.lirmm.fr/cogui#">
    <owl:Ontology rdf:about=""/>
]   <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Object Properties
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->
    <!-- http://www.lirmm.fr/cogui#_rt0 -->

]   <owl:ObjectProperty rdf:about="_rt0">
        <rdfs:label xml:lang="en">binary_relation</rdfs:label>
```

```
        <rdfs:range rdf:resource="_ct0"/>
        <rdfs:domain rdf:resource="_ct0"/>
    </owl:ObjectProperty>
    <!-- http://www.lirmm.fr/cogui#_rt36 -->

    <owl:ObjectProperty rdf:about="_rt36">
        <rdfs:label xml:lang="en">enemy</rdfs:label>
        <rdfs:domain rdf:resource="_ct1"/>
        <rdfs:range rdf:resource="_ct1"/>
        <rdfs:subPropertyOf rdf:resource="_rt9"/>
    </owl:ObjectProperty>
    <!-- http://www.lirmm.fr/cogui#_rt39 -->

    <owl:ObjectProperty rdf:about="_rt39">
        <rdfs:label xml:lang="en">friend</rdfs:label>
        <rdfs:domain rdf:resource="_ct1"/>
        <rdfs:range rdf:resource="_ct1"/>
        <rdfs:subPropertyOf rdf:resource="_rt9"/>
    </owl:ObjectProperty>
    <!-- http://www.lirmm.fr/cogui#_rt4 -->

    <owl:ObjectProperty rdf:about="_rt4">
        <rdfs:label xml:lang="en">different</rdfs:label>
        <rdfs:domain rdf:resource="_ct0"/>
        <rdfs:range rdf:resource="_ct0"/>
        <rdfs:subPropertyOf rdf:resource="_rt0"/>
    </owl:ObjectProperty>
    <!-- http://www.lirmm.fr/cogui#_rt40 -->

    <owl:ObjectProperty rdf:about="_rt40">
        <rdfs:label xml:lang="en"
            >brotherhood_relation</rdfs:label>
        <rdfs:domain rdf:resource="_ct1"/>
```

```
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:subPropertyOf rdf:resource="_rt8"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt41 -->

<owl:ObjectProperty rdf:about="_rt41">
    <rdfs:label xml:lang="en"
        >relation_involving_a_man</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt8"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt42 -->

<owl:ObjectProperty rdf:about="_rt42">
    <rdfs:label xml:lang="en"
        >relation_involving_a_woman</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt8"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt43 -->

<owl:ObjectProperty rdf:about="_rt43">


    <rdfs:label xml:lang="en"
        >descendants_relation</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct1"/>
    <rdfs:subPropertyOf rdf:resource="_rt8"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt44 -->

<owl:ObjectProperty rdf:about="_rt44">
    <rdfs:label xml:lang="en">parent</rdfs:label>
    <rdfs:domain rdf:resource="_ct1"/>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:subPropertyOf rdf:resource="_rt43"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt45 -->

<owl:ObjectProperty rdf:about="_rt45">
    <rdfs:label xml:lang="en">not_child</rdfs:label>
    <rdfs:domain rdf:resource="_ct1"/>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:subPropertyOf rdf:resource="_rt43"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt46 -->
```

```
<owl:ObjectProperty rdf:about="_rt46">
    <rdfs:label xml:lang="en">grandparent</rdfs:label
    <rdfs:domain rdf:resource="_ct1"/>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:subPropertyOf rdf:resource="_rt43"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt47 -->

<owl:ObjectProperty rdf:about="_rt47">
    <rdfs:label xml:lang="en">child</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct1"/>
    <rdfs:subPropertyOf rdf:resource="_rt43"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt48 -->

<owl:ObjectProperty rdf:about="_rt48">
    <rdfs:label xml:lang="en">aunt</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt42"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt49 -->


<owl:ObjectProperty rdf:about="_rt50">
    <rdfs:label xml:lang="en">cousin_female</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt42"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt51 -->

<owl:ObjectProperty rdf:about="_rt51">
    <rdfs:label xml:lang="en">not_wife</rdfs:label>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:range rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt42"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt52 -->

<owl:ObjectProperty rdf:about="_rt52">
    <rdfs:label xml:lang="en">wife</rdfs:label>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:range rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt42"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui# rt53 -->
```

```
<owl:ObjectProperty rdf:about="_rt53">
    <rdfs:label xml:lang="en">sister</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt40"/>
    <rdfs:subPropertyOf rdf:resource="_rt42"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt54 -->

<owl:ObjectProperty rdf:about="_rt54">
    <rdfs:label xml:lang="en">uncle</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt41"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt55 -->

<owl:ObjectProperty rdf:about="_rt55">
    <rdfs:label xml:lang="en">nephew</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt41"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt56 -->


<owl:ObjectProperty rdf:about="_rt56">
    <rdfs:label xml:lang="en">cousin_male</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt41"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt57 -->

<owl:ObjectProperty rdf:about="_rt57">
    <rdfs:label xml:lang="en">not_husband</rdfs:label>
    <rdfs:range rdf:resource="_ct2"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt41"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt58 -->

<owl:ObjectProperty rdf:about="_rt58">
    <rdfs:label xml:lang="en">husband</rdfs:label>
    <rdfs:range rdf:resource="_ct2"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt41"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt59 -->
```

```
<owl:ObjectProperty rdf:about="_rt59">
    <rdfs:label xml:lang="en">brother</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt40"/>
    <rdfs:subPropertyOf rdf:resource="_rt41"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt61 -->

<owl:ObjectProperty rdf:about="_rt61">
    <rdfs:label xml:lang="en">son</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt47"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt62 -->

<owl:ObjectProperty rdf:about="_rt62">
    <rdfs:label xml:lang="en">daughter</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt47"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt63 -->


<owl:ObjectProperty rdf:about="_rt63">
    <rdfs:label xml:lang="en">grandfather</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct4"/>
    <rdfs:subPropertyOf rdf:resource="_rt46"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt64 -->

<owl:ObjectProperty rdf:about="_rt64">
    <rdfs:label xml:lang="en">grandmother</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt46"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt65 -->

<owl:ObjectProperty rdf:about="_rt65">
    <rdfs:label xml:lang="en">mother</rdfs:label>
    <rdfs:range rdf:resource="_ct1"/>
    <rdfs:domain rdf:resource="_ct2"/>
    <rdfs:subPropertyOf rdf:resource="_rt42"/>
    <rdfs:subPropertyOf rdf:resource="_rt44"/>
</owl:ObjectProperty>


<!-- http://www.lirmm.fr/cogui#_rt66 -->
```

```
    <owl:ObjectProperty rdf:about="_rt66">
        <rdfs:label xml:lang="en">father</rdfs:label>
        <rdfs:range rdf:resource="_ct1"/>
        <rdfs:domain rdf:resource="_ct4"/>
        <rdfs:subPropertyOf rdf:resource="_rt41"/>
        <rdfs:subPropertyOf rdf:resource="_rt44"/>
    </owl:ObjectProperty>


    <!-- http://www.lirmm.fr/cogui#_rt8 -->

    <owl:ObjectProperty rdf:about="_rt8">
        <rdfs:label xml:lang="en">family_relation</rdfs:label>
        <rdfs:domain rdf:resource="_ct1"/>
        <rdfs:range rdf:resource="_ct1"/>
        <rdfs:subPropertyOf rdf:resource="_rt0"/>
    </owl:ObjectProperty>


    <!-- http://www.lirmm.fr/cogui#_rt9 -->

    <owl:ObjectProperty rdf:about="_rt9">
        <rdfs:label xml:lang="en"
            >extrafamily_relation</rdfs:label>
        <rdfs:domain rdf:resource="_ct1"/>
        <rdfs:range rdf:resource="_ct1"/>
        <rdfs:subPropertyOf rdf:resource="_rt0"/>
    </owl:ObjectProperty>


<!--
////////////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
////////////////////////////////////////////////////////////////////////////////////////
 -->


<!-- http://www.lirmm.fr/cogui#_ct0 -->

<owl:Class rdf:about="_ct0">
    <rdfs:label xml:lang="en">Top</rdfs:label>
</owl:Class>


<!-- http://www.lirmm.fr/cogui#_ct1 -->

<owl:Class rdf:about="_ct1">
    <rdfs:label xml:lang="en">Human</rdfs:label>
    <rdfs:subClassOf rdf:resource="_ct0"/>
</owl:Class>


<!-- http://www.lirmm.fr/cogui#_ct2 -->

<owl:Class rdf:about="_ct2">
    <rdfs:label xml:lang="en">Woman</rdfs:label>
    <rdfs:subClassOf rdf:resource="_ct1"/>
```

```
    </owl:Class>



    <!-- http://www.lirmm.fr/cogui#_ct4 -->

    <owl:Class rdf:about="_ct4">
        <rdfs:label xml:lang="en">Man</rdfs:label>
        <rdfs:subClassOf rdf:resource="_ct1"/>
    </owl:Class>



    <!--
    ///////////////////////////////////////////////////////////////////////////////////////
    //
    // Individuals
    //
    ///////////////////////////////////////////////////////////////////////////////////////
     -->



    <!-- http://www.lirmm.fr/cogui#abdou -->

    <_ct1 rdf:about="abdou">
        <rdf:type rdf:resource="_ct4"/>
        <rdf:type rdf:resource="&owl;Thing"/>
        <_rt59 rdf:resource="fatima"/>
    </_ct1>



      <!-- http://www.lirmm.fr/cogui#dalila -->

      <_ct1 rdf:about="dalila">
          <rdf:type rdf:resource="&owl;Thing"/>
      </_ct1>



      <!-- http://www.lirmm.fr/cogui#fatima -->

      <_ct2 rdf:about="fatima">
          <rdf:type rdf:resource="_ct1"/>
          <rdf:type rdf:resource="&owl;Thing"/>
          <_rt53 rdf:resource="abdou"/>
          <_rt39 rdf:resource="dalila"/>
      </_ct2>
</rdf:RDF>



<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
  <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
  <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<rdf:RDF xmlns="http://www.semanticweb.org/ontologies/2011/5/1/try.owl#"
xml:base="http://www.semanticweb.org/ontologies/2011/5/1/try.owl"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
xmlns:owl="http://www.w3.org/2002/07/owl#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:about="#Human"/>
  <owl:Class rdf:about="#Femme">
  <rdfs:subClassOf rdf:resource="#Human"/>
  </owl:Class>
  <owl:Class rdf:about="#Homme">
  <rdfs:subClassOf rdf:resource="#Human"/>
  </owl:Class>
  <owl:ObjectProperty rdf:about="#nephew">
  <rdfs:range rdf:resource="#Homme"/>
  <rdfs:domain rdf:resource="#Human"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#grandfather">
  <rdfs:range rdf:resource="#Homme"/>
  <rdfs:domain rdf:resource="#Human"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#mother">
  <rdfs:range rdf:resource="#Femme"/>
```

```
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandmother">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#wife">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Homme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#cousin">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#binary_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#different">
<rdfs:domain rdf:resource="&owl;Thing"/>
<rdfs:range rdf:resource="&owl;Thing"/>
<rdfs:subPropertyOf rdf:resource="#binary_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#family_relation">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#binary_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#extrafamily_relation">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#binary_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#uncle">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#niece">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#father">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brother">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#sister">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#daughter">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#husband">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Femme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#son">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brother">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#niece">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#father">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brother">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#sister">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#daughter">
<rdfs:range rdf:resource="#Femme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#husband">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Femme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#son">
<rdfs:range rdf:resource="#Homme"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brother">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#father">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandfather">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#husband">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Femme"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#nephew">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#non_son">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#son">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#uncle">
<rdfs:domain rdf:resource="#Homme"/>
```

```
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#cousin_male">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#not_husband">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Femme"/>
<rdfs:subPropertyOf rdf:resource="#relHomme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#daughter">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandmother">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#mother">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#niece">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#non_daughter">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#sister">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#wife">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Homme"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#aunt">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#cousin_female">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#not_wife">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Homme"/>
<rdfs:subPropertyOf rdf:resource="#RelFemme"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brotherhood_relation">
<rdfs:range rdf:resource="#Human"/>
<rdfs:domain rdf:resource="#Human"/>
```

```
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#sister">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#brotherhood_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brother">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#brotherhood_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#descendants_relation">
<rdfs:range rdf:resource="#Human"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#parent">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#descendants_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#not_child">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#descendants_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandparent">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#descendants_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#child">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
```

```
<rdfs:subPropertyOf rdf:resource="#descendants_relation",
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#parent">
<rdfs:range rdf:resource="#Human"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#mother">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#parent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#father">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#parent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandparent">
<rdfs:range rdf:resource="#Human"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandfather">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#grandparent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#grandmother">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#grandparent"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#enemy">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
```

```
<rdfs:subPropertyOf rdf:resource="#extrafamily_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#friend">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#extrafamily_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#brotherhood_relation">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#family_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#relHomme">
<rdfs:domain rdf:resource="#Homme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#family_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#RelFemme">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#family_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#descendants_relation">
<rdfs:domain rdf:resource="#Human"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#family_relation"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#child">
<rdfs:range rdf:resource="#Human"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#son">
<rdfs:domain rdf:resource="#Homme"/>
```

```
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#child"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#daughter">
<rdfs:domain rdf:resource="#Femme"/>
<rdfs:range rdf:resource="#Human"/>
<rdfs:subPropertyOf rdf:resource="#child"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="#not_child">
<rdfs:range rdf:resource="#Human"/>
<rdfs:domain rdf:resource="#Human"/>
</owl:ObjectProperty>
</rdf:RDF>
<!-- Generated by the OWL API (version 2.2.1.1138) http://owlapi.sourceforge.net -->
```

REFERENCES BIBLIOGRAPHIQUE

Signe	Titre d'ouvrage (Mémoire)	Auteur	Maison (organisme)	Année
[AUB,07]	UML2	laurent audibert	centre universitaire de technologie de villetaneuse dpt informatique	2007
[BAK,06]	Etude et proposition d'une architecture de médiation entre source de données hétérogènes	Bakhtouchi abd-elgjani	INI Alger	2006
[BEN et al, 06]	Vers l'interopérabilité des systèmes d'information hétérogènes	Laïla Benhlima , Dalila Chiadmi	Ecole Mohammadia d'Ingénieurs BP 765 Agdal, Rabat, Maroc	2006
[BEN,08]	http://websemantique.org/OWL	Ben Ameur Houcine	Université de montréal	2008
[BOU, 08]	Intégration de sources de données à base d'ontologique dans un environnement P2P	Amel Boussis	INI Alger	2007/ 2008
[GUE, 04]	Intégration de données	Cédric Gueydan		2004
[COR et al, 06]	Introduction à OWL Langage XML d'ontologies	Pol Coret, Julien Richard, Eric Talavet, Tetiana Trofimova	Ecole Centrale de Lyon	2006
[DAM,08]	http://websemantique.org/OWL	Olivier Dameron	Université rennes1	2005
[FAN,06]	Vers une integration des differentes approaches de modelisation a base ontologique: application au modele PLIB et OWL	Fankam nguemkam chimene jeannette	Université poitiers	2006
[GEN,99]	Vers un système de recherche documentaire basé sur les graphes conceptuels	David Genest	CNRS Université de MontpellierII	1999
[GER,06]	Introduction au formalisme des graphes	Olivier Gerbe	HEC - Montreal	2006

	conceptuels			
[HAC et al,03]	L'intégration de sources de données	Mohand-Saïd Hacid , Chantal Reynaud	Université Claude Bernard Lyon	2003
[JEN,11]	http ://jena.sourceforge.net/	JENA SOURCEFORGE		2011
[LIR,11]	http://www.lirmm.fr/cogui/	LIRMM		2011
[MAU,02]	Data Integration: A theoretical perspective	Maurizio Lenzerini	Université La Sapienza Italy	2002
[MCB et al, 03]	Data integration bi-directional schema transformation rules	Peter Mcbrien, Alexendra poulovassilis	Imperial college, Birkberck college London	2003
[MEL,07]	Realisation de l'interroperabilité semantique des systems bases sur les ontologies et les flux d'informations	Mellal nassima	Polytech'Savoie	2007
[HAC et al,03]	L'integration de sources de données	Mohand-Said Hacid,Chantal Reynaud	Université Claude Bernard Lyon	2003
[PAR et al,96]	Intégration de bases de données :panorama des problèmes et des approches	Christine Parent,Stefano Spaccapietra	Ecole polytechnique fédérale de Lausanne	1996
[SHO,09]	IXIA (IndeX-based Integration Approach) A Hybrid Approach to Data Integration	Shokoh kerman-shahani	Universite Joseph fourier Grenope I	2009
[SCB,11]	http://protege.stanford.edu	center for biomedicalninformatics research		2011
[W3C]	www.w3c.org	W3C	W3C	2004
[W3C,08]	www.w3c.org/TR/rdf-sparqlquery	W3C		2008
[WIE,91]	Mediators in the architecture of future information systems	Gio Wiederhold	Universite stanford	1991

[XAV,03]	Un modèle de vue pour l'intégration de données XML	Xavier Baril	Université de Montpellier II	2003
[XAV, 05]	Introduction à OWL, un langage XML d'ontologies Web	Xavier lacot	Ecole Nationale Supérieure des Télécommunications	2005
[DJA, 03]	Intégration des ressources Web dans un environnement P2P, basée sur les ontologies et la gestion de la confiance	Djaghloul Younes	université Mentouri Constantine	2003
[BAL, 07]	Interopérabilité Sémantique des Systèmes d'Information Distribués	Bala Mahfoud	INI Alger	2007
[CHA, 03]	Modèle de graphes conceptuels et représentation sémantique du langage naturel	Chawk Mohamad	ERSICO Université Jean-Moulin Lyon 3	