REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE ABOU BEKR BELKAID – TLEMCEN

Faculté des Sciences de l'Ingénieur
Département d'Electronique

Mémoire

Pour l'obtention du Diplôme de Magister en API
(Automatique-Productique-Informatique)

Option : Informatique

Thème

MEDIATION DE SOURCES DE DONNEES HETEROGENES DANS LE CADRE DU WEB SEMANTIQUE

Présenté par

Djelloul BOUIDA

Soutenu en 02 juillet 2005 devant le Jury composé de :

M. **M.BOUFAIDA**	Président du jury	(Prof. à l'Université de Constantine)
M. **H.HAFFAF**	Examinateur	(MC à l'université d'Es-senia Oran)
M. **A.CHIKH**	Examinateur	(MC à L'université de Tlemcen)
Mme **Z.BOUFAIDA**	Encadreur	(Prof. à l'Université de Constantine)

Remerciement

Par l'intermédiaire de ces quelques lignes, je tiens à adresser mes plus sincères remerciements à Mme Zizette BOUFAIDA pour ses conseils, ainsi que pour son encadrement tout en m'accordant une grande confiance et une souplesse de manœuvre.

J'adresse également mes remerciements aux membres du jury, M. Mahmoud BOUFAIDA pour avoir accepté de présider le jury, mais également à M. Hafid HAFFAF et M. Azeddine CHIKH pour avoir été examinateurs de ce mémoire.

Je tiens également à remercier toute personne ayant participé de loin ou de prés, ne serait qu'une proposition, un encouragement ou une orientation à l'élaboration de ce présent travail.

Merci également à tous ceux que j'aurais pu oublier, désole ☺

A ma fiancée....

Sommaire

Chapitre 1

Introduction générale

1.1 Introduction

L'intégration de sources d'information hétérogènes dans le cadre du Web sémantique pourra s'appuyer sur de multiples systèmes de médiation. Certains pourront suivre une approche centralisée telle l'approche décrite dans le chapitre suivante. D'autres pourront suivre une approche décentralisée consistant à considérer une coalition de serveurs d'information. Chaque serveur peut jouer indifféremment le rôle de serveurs de données ou de médiateurs avec ses pairs en participant de manière distribuée et collective au traitement des requêtes des 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[1].

On peut penser que dans le cadre du Web sémantique ces systèmes de médiation distribués seront mieux adaptés par leur flexibilité. Dans ce contexte de médiation décentralisée, apparaissent de nouveaux problèmes liés à la façon de connecter les différents systèmes mis en relation. Il faut trouver et définir des correspondances sémantiques entre les ontologies manipulées par chacun des systèmes. Il faut pouvoir disposer d'une approche simple et naturelle de description de telles correspondances sémantiques entre ontologies. Le passage à l'échelle du Web en matière de connexion entre ontologies n'est envisageable que si elle est en partie automatisée. Il est donc nécessaire d'étudier comment cette automatisation est possible, sachant qu'elle devra pouvoir être établie entre des ontologies qui sont locales à des sources et qui sont hétérogènes. Il est en effet illusoire de penser que toute ontologie ou schéma local à une source d'information sera toujours exprimée dans les termes d'un schéma global (ou ontologie globale) pré-existant(e)[1].

1.2 Motivation

1.2.1 La diversité des sources de données

L'évolution constante en matière de réseaux et de bases de données ces trente dernières années a mené à une demande toujours croissante d'accès rapides à une large quantité d'informations variées.
Certaines de ces informations sont enregistrées dans des bases de données traditionnelles et accessibles par les langages de requête très puissants, autres sont simplement stockées dans des systèmes de fichiers ou de simples tableurs, autres encore sont les résultats d'application plus ou moins complexes. Enfin, certains sont les données changeantes et arborescentes des pages du Web. La diversité de ces sources de données conduit à des modes de consultations qui peuvent être très différents. Ainsi, une base de données relationnelles sera interrogée par l'intermédiaire d'une requête SQL, une page Web sera consulté par une adresse Web (URL) particulière, et un tableur par une formule

[1] Laublet, P., Reynaud, C., Charlet, J. 2002. " Sur quelques aspects du web sémantique ". *Assises du GDR 13* (décembre 2002). Nancy : Editions Cépadues. [14.10.2003]. Disponible sur : <URL : http://www.lalic.paris4.sorbonne.fr/stic/articles/03-webSemantique.pdf>

spécifique. Une telle variété de sources implique diverses façons de les interroger – c'est-à-dire de formuler une requête- mais aussi plusieurs manières pour la source pour présenter un résultat.

Pour obtenir un résultat, la requête fait appel aux méthodes d'accès aux sources de données. Ces méthodes d'accès peuvent se faire localement par l'intermédiaire d'une interface spécifique ou d'une interface de programmation, ou de manière distante via des protocoles de communications. L'apparition de protocoles comme ODBC, JDBC, ou IIOP simplifient considérablement l'accès à nombre de ces sources de données. L'émergence d'architectures de médiation « trois-tiers » permet de plus en plus de regrouper l'accès à différentes sources par l'intermédiaire d'une interface unique.

1.2.2 Hétérogénéité schématique

Les conflits schématiques entre les données apparaissent lorsque des concepts équivalents sont représentés différemment dans des bases de données locales. Les conflits schématiques sont associés aux noms, types de données, attributs, et unités, comme l'absence d'un attribut dans l'une des deux entités équivalentes appartenant à des schémas locaux différents. Les conflits dans la représentation d'un attribut se présentent lorsque le domaine de types similaires est incompatibles (p.ex. une adresse définie comme une chaîne de caractères et comme un n-uplet <numéro, rue, ville, code postal>. Afin de masquer ces différences, un schéma global doit être construit sur un modèle suffisamment expressif.

1.2.3 Hétérogénéité sémantique

Ce terme caractérise les différences de signification, d'interprétation ou d'utilisation d'une même donnée. Il est important de distinguer l'hétérogénéité sémantique et structurelle (schématique). En fait les différences schématiques n'ont intérêt que s'il existe des similarités sémantiques entre des objets. Quatre types de relations sémantiques peuvent exister entre deux représentations R1 et R2 : R1 identique à R2, lorsque les mêmes constructeurs sont utilisés pour représenter le même concept ; R1 équivalente à R2, lorsque des constructeurs différents sont utilisés pour représenter le même concept ; R1 compatible avec R2, si elles ne sont ni identiques et ni équivalentes ; R1 incompatible avec R2, si elles sont contradictoires, à cause de différences fondamentales dans la représentation de l'information sous-jacente.

Le problème d'avoir des différences dans la compréhension par rapport à la même information vient parfois de la diversité des aspects géographiques et organisationnels qui les utilisent. Par exemple, l'attribut prix d'un produit est compris en termes d'euros dans la communauté européenne et en dollars aux Etats-Unis, le dollar américain n'a pas la même valeur que le dollar canadien, etc.

1.3 Problématique

1.3.1 Web sémantique :

Sous l'expression Web sémantique, attribuée le plus souvent à Tim Berners-Lee, se regroupe un ensemble de programmes de recherche et de travaux variés. Leur objectif commun est de permettre aux machines d'exploiter automatiquement les contenus de sources d'information accessibles par le Web pour réaliser des tâches variées. La réalisation de cet objectif repose sur l'existence de données, accessibles par le Web, structurées ou semi-structurées, représentées dans un formalisme autorisant des traitements automatisés allant au delà des traitements liés à la présentation des données et mettant en oeuvre des mécanismes d'inférence puissants.

De tels traitements favorisent l'intégration des données issues de sources multiples et hétérogènes et rendent possible leur utilisation dans des applications variées. Cet objectif doit aujourd'hui être réalisé dans un contexte d'informatique distribuée marqué par la généralisation de la diversité des modes d'accès à l'information et par la recherche générale d'interopérabilité. Ce contexte suscite la création de standards et, combiné au nouveau contexte du Web sémantique, soulève des problèmes nouveaux dont les solutions passent par l'adaptation de technologies déjà connues ou par l'exploitation de technologies qui restent encore à inventer.
On peut en attendre, entre autres, des recherches d'information plus exactes et plus efficaces, une meilleure compréhension des sites par des agents intelligents (commerce électronique, ….), différents types d'interopérabilité entre systèmes mais aussi une assistance « intelligente » à l'utilisation d' applications complexes.

1.3.2 Médiation entre ontologies

Les ontologies vont jouer un rôle central dans le marquage sémantique des données du web. Si on peut anticiper la construction d'ébauches d'ontologies émanant de groupes de standardisation de différents domaines d'application, de par la nature profondément distribuée du web, il est inévitable que différentes ontologies cohabiteront et pourront être utilisées pour marquer sémantiquement des données relatives à un même domaine.

Des outils permettant de définir et de trouver des correspondances sémantiques entre ontologies seront donc au cœur de la recherche d'informations pertinentes sur le web. C'est l'ensemble des ontologies réparties sur le web et l'ensemble des correspondances existant entre elles qui formeront un tout cohérent qui pourra être qualifié de "web sémantique".
Pour passer à l'échelle du web, seule une approche déclarative, formelle et générique nous paraît possible. Une approche déclarative pour définir des "mappings" entre ontologies présente l'avantage d'être simple et naturelle pour qui a à décrire "manuellement" ces mappings. Définir ces mappings dans un langage formel rend possible l'inférence de nouvelles correspondances qui découlent logiquement des mappings fournis au départ ou de mappings partiels repérés automatiquement.

1.4 Objectif et contexte du mémoire

Les systèmes d'intégration de données hétérogènes s'appuyant actuels sur une approche centralisée dont les données restent stockées dans les sources et sont exprimées dans les termes d'un schéma global (ou ontologie globale), ce schéma global (ou ontologie globale) représente un point faible dans le cadre du Web sémantique. En fait, il faut envisager une architecture décentralisée baser sur un réseau dynamique et flexible, or que le l'approche centralisée utilise une architecture client/serveur qui montre des limitations diverses (le cas de blocage du serveur immobilise tout le système), l'utilisation des réseaux pair-à-pair (peer-to-peer) devient indispensable.

Un des objectifs majeur est de proposer un système de médiation pouvant intégrer le plus grand nombre de sources différentes possibles et s'appuyant sur une architecture décentralisé pour éviter la conception d'une schéma globale (Ontologie globale), et ceci pour remédier au problème du web sémantique (l'échelle mondial). Ce système s'appuyant sur XML comme modèle d'échange vu leurs avantages (structures de données complexes et irrégulières, pas de schéma obligatoire,...).

Et pour aller vers un Web Sémantique, il est nécessaire de disposer d'un langage d'ontologie ayant une sémantique formelle bien définie, DAML-OIL est mieux adapté vu leurs avantages [chap. 3].

Nous utilisons également le langage DAML+OIL comme langage de représentation des règles de mapping pour le système de mapping qui capture les correspondances entre les déférentes sources de données.

Nous présentons aussi l'implémentation d'un moteur d'évaluation des règles de mapping, qui est un couplage du moteur d'inférence JESS avec DAML API. Ceci est nécessaire pour permettre l'implémentation.

Afin de présenter notre contribution nous proposons le plan de lecture suivant :

Chapitre 2 Nous introduirons la notion de données semi-structurées. Puis nous montrerons comment ont été modélisées de telles données. Nous verrons ensuite comment des métadonnées peuvent être définies ou extraites. Il faut un langage permettant d'interroger ce nouveau type de données, et plusieurs langages ont été proposés. Nous étudierons les caractéristiques des principaux langages portant sur les données semi-structurées dans la section 2.6. L'utilisation d'un nouveau type de données ne doit pas faire oublier les autres types de données existants, il s'agit de montrer comment intégrer ces données avec d'autres données réparties sur un réseau aussi vaste que l'Internet. Cette intégration sera décrite dans la section 2.7. Divers systèmes d'intégration de données existent déjà. Les plus intéressants sont analysés par la suite.

Chapitre 3 Nous avons présenté dans ce chapitre Web du futur, qui est nommé le *Web Sémantique*. Nous avons passé en revue comment les chercheurs voient le Web intelligent du futur, et quel pourrait être son impact. Nous décrivons en suite les principales langages du Web sémantique, avec une présentation des concepts de chaque langage.

Chapitre 4 Dans ce chapitre nous abordons la notion d'ontologie en prenant une position affirmée sur ce que nous pensons que c'est et que ça n'est pas. Cette position nous semble nécessaire à la compréhension de ce que l'on peut faire ou non avec les ontologies comme avec de nombreux produits terminologiques tels que les thésaurus. Nous prenons appui sur les travaux développés au sein de la communauté Ingénierie des connaissances, s'appuyant eux-mêmes sur des travaux plus anciens comme la terminologie ou la recherche d'information, puis nous développons la conception des ontologies.. Nous terminons ce chapitre avec des exemples d'utilisations des ontologies.

Chapitre 5 Ce chapitre s'attache à présenter les aspects techniques de notre étude sur le peer-to-peer. Nous livrons tout d'abord un bref historique de l'informatique pour y situer l'apparition du peer-to-peer. Une étude technique introduite le concept tel que nous le connaissons aujourd'hui et l'analyse. Nous décrivons ensuite les principaux architecture possibles, pour finalement considérer les possibilités d'optimisation dont peuvent bénéficier les protocoles peer-to-peer.

Chapitre 6 Dans ce chapitre, nous avons présenté un système de mapping qui capture les correspondances entre le schéma global et les schémas sources. Ce système est conçu de manière indépendante du modèle de représentation des connaissances. En effet, ses règles sont exprimées en fonction de concepts et de relations. Nous avons utilisé pour cela, les ontologies comme modèle de représentation des bases de données au niveau conceptuel. Un exemple a été illustré pour montrer comment utiliser les règles de mapping dans le processus de réécriture de requêtes. Aussi, nous avons présenté comment représenter les informations sémantiques tant au niveau conceptuel qu'au niveau de langage de représentation.
Nous avons également utilisé le langage de représentation des ontologies et des règles de mapping qui est le DAML+OIL, ainsi qu'un couplage du moteur d'inférence JESS avec DAML API.

Chapitre 7 Dans ce chapitre, nous présentons une étude de cas qui permet de mieux comprendre les différentes parties de notre solution. Cette étude est appliquée au domaine du transport.

Et en fin, on termine notre mémoire avec une conclusion générale dans laquelle nous faisons le point sur le travail réalisé et donnons quelques idées pour une éventuelle amélioration.

Chapitre 2

Médiation de données

2.1 Introduction

Les données du World Wide Web se caractérisent par des structures irrégulières dynamiques ou inconnues. Ces types de données sont communément appelés données *semi-structurées*. L'apparition du concept des données semi-structurées est considérée comme une révolution dans le monde des bases de données. En effet, en intégrant le monde documentaire et le monde des bases de données relationnelles et objets, les données semi-structurées ont pour objectif de permettre une meilleure représentation des entités du monde réel.

En contrepartie, de telles données sont complexes à manipuler par des traitements automatiques. Les langages de requête, et les techniques d'optimisation et de stockage utilisés pour les données classiques ne s'appliquent plus. Il a donc fallu adapter, voire inventer de nouveaux concepts et algorithmes pour manipuler des objets semi-structurés.

Il faut pouvoir représenter ces données. Pour cela des *modèles de données* propres à ce nouveau type de données ont été proposés. Il faut aussi pouvoir *échanger* ces données, aussi des *formats de documents* représentant ces données ont dû être définis.

Lorsque le modèle de donnée a été formalisé, il s'agit d'*interroger* les données. Il faut donc définir un *langage* de requête applicable aux données semi-structurées. Ce langage doit être défini de sorte à exploiter au maximum les spécialités de ces nouveaux types de données.

Un changement de technologie ne peut se faire sans tenir compte de l'historique et des technologies déjà en place. Il est donc indispensable de permettre l'échange et la coopération entre des données provenant de sources classiques (hiérarchiques, relationnelles, objets, annuaires LDAP) et de ces sources semi-structurées. Les techniques de *médiation* ont dû être adaptées pour prendre en compte ces nouveaux types de données.

Partant de ces bases, nous présentons un état de l'art sur l'intégration de données dans un contexte hétérogène.

2.2 Notions de données semi-structurées

Les bases de données traditionnelles s'appuient sur la régularité des structures des objets qu'elles manipulent. En effet, l'une des caractéristiques principales des bases de données relationnelles est la définition de schémas fixes auxquels les données sont ensuite obligées de se conformer. Il en va de même pour les bases de données objets. Cela simplifie les traitements informatiques (stockages, interrogations) et permet des accès sur critères très rapides. En revanche, une telle régularité ne permet pas de reproduire la pensée de l'utilisateur ou le monde réel. Ceci car les bases de données traditionnelles (structurées) ne peuvent pas travailler sur des données dont le schéma n'est pas fixé *a priori*. Cette opposition entraîne par la suite les problèmes bien connus suivant :

La structure de données peut évoluer : cela nécessite alors la modification du schéma. Cette modification peut s'avérer complexe à réaliser ;

Les données peuvent ne pas se conformer exactement au schéma : ceci nécessite le plus souvent de surdimensionner le nombre de colonnes et d'employer beaucoup de valeurs nulles ;

- *Le même attribut peut avoir des types différents suivants les données* : cela nécessite le plus souvent l'emploi du type le plus englobant (souvent le type « chaîne de caractères ») ; ceci a pour conséquence de réduire la finesse de la description et les possibilités d'interrogation ;
- *Une instance d'attributs peut être mono-valuée ou multi-valuée :* les SGBD traditionnels traitent ces deux cas comme deux cas bien distincts, or il faudrait pouvoir gérer ces cas uniformément.
- *Les données peuvent être faiblement structurées (texte brut)* : ceci entraîne l'utilisation de blocs de données brutes (BLOB) qui ne permettent pas des techniques très fines d'interrogation.

Or la majorité de données non-structurées (texte brut, données binaires) reste limitée ou trop spécifique et globale. Les requêtes sur objets longs (BLOB ou CLOB) consistent surtout en la recherche par mots-clefs. Celles-ci sont les plus souvent basées sur des techniques d'indexation dans le cas des données textuelles. IL existe aussi d'autres méthodes de recherche spécifique (par exemple, pour les images, suivant le format d'encodage, la reconnaissance de formes, etc.) dans les données non-structurées. Ces méthodes spécifiques peuvent permettre des interrogations très poussées (utilisation de thésaurus), mais restent propres aux données manipulées et se basent sur une analyse globale des données.

- Les données semi-structurées ne sont pas contraintes par l'utilisation de structures aussi rigides que dans le cas du relationnel ou de l'objet. Elles bénéficient néanmoins d'une structure flexible restant assez cohérent pour pouvoir être manipulées.

- Les caractéristiques (Abiteboul, 1998) (Abiteboul, 1997) des données semi-structurées sont décrites ci-dessous :

- *La structure est irrégulière* : une collection de données semi-structurées peut comporter des éléments hétérogènes de différents types pour représenter la même information. Un même attribut peut être mono-valué dans certaines instances et multi-valuées dans d'autres. Des informations supplémentaires (annotation, détails) peuvent apparaître à certains endroits. Enfin des éléments peuvent aussi manquer dans certaines instances ;
- *La structure est implicite* : même si une certaine structuration peut sembler présente (indiquée par l'intermédiaire d'étiquettes, de balises, de champs, etc.), il peut ne pas exister de description explicite. L'extraction et l'interprétation de la structure est un processus difficile puisqu'il s'agit à la fois d'analyser, d'interpréter les données et d'effectuer des correspondances logiques pour enfin en déduire la structure ;
- *La structure est partielle* : une partie des données peut être constituée d'information non structurée (images, textes bruts) ;
- *Le typage est irrégulier* : il n'y a pas de typage strict dû à l'hétérogénéité des données ;
- *Le schéma est lâche* : dans les SGBD traditionnels une politique de typage strict est imposée pour protéger la consistance des données. Dans des données semi-structurées, des transgressions sont tolérées et conduisent à une altération du schéma ;
- *Le schéma peut être antérieur ou postérieur* : la notion de schéma peut être antérieure ou postérieure à l'existence des données. Les systèmes de gestion de bases de données traditionnelles sont basés sur l'hypothèse d'un schéma fixe défini avant toute insertion de

données. Dans le cas de données semi-structurées, la notion de schéma est souvent postérieure à l'existence de données ;

- *Le schéma est large* : en conséquence de l'hétérogénéité, le schéma est la plupart du temps très large pour englober toutes les informations des différentes instances des données. Cela peut poser des problèmes dans les formulations de requêtes par l'utilisateur ;

- *Le schéma est parfois ignoré* : les requêtes exploratoires ou à base de recherches de mots sans indications précises sont très usitées. Dans ce type de requête, le schéma n'est pas utilisé ;

- *Le schéma évolue rapidement* : les sources de données semi-structurées sont habituellement dynamiques. Leurs données et leur organisation changent fréquemment. En conséquence, leurs schémas sont souvent mis à jour ;

- *Le type des éléments de donnée est éclectique* : la structure d'une information peut varier suivant le point de vue. Le type des objets peut changer lors de leurs traitements. Par exemple pour décrire un wagon fumeur / non fumeur, on peut utiliser soit l'attribut catégorie de type chaîne, et où les valeurs possibles seront les chaînes « fumeur » ou « non fumeur », soit l'attribut booléen fumeur dont les valeurs seront vrai ou faux ;

- *La distinction entre schéma et donnée est floue* : dans les SGBD conventionnels, la différence entre schéma et donnée est très marquée. Nous avons vu qu'en semi-structurée ces différences s'estompent : les schémas se modifient, sont larges, les requêtes portent aussi bien sur les données que sur les schémas. De plus, en semi-structuré, la différence entre schéma et donnée n'a parfois logiquement que peu de sens.

2.3 Représentation des données semi-structurées

Une des façons de représenter une donnée semi-structurée est le graphe. Les éléments du schéma sont alors représentés sous forme d'étiquettes attachées aux arcs ou aux nœuds du graphe.

La figure 2.1 illustre le graphe semi-structuré d'une personne .

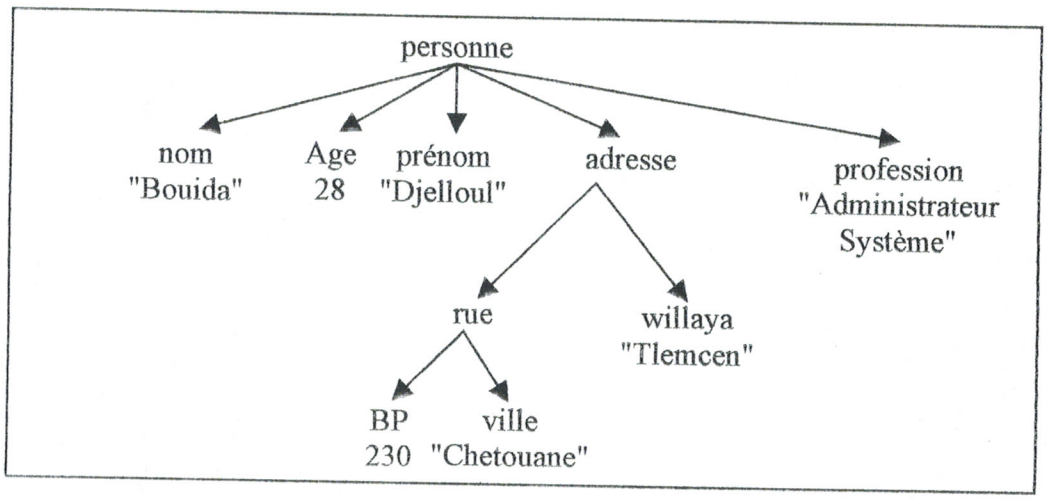


Fig. 2.1 Exemple de graphe représentant deux données semi-structurées

Ainsi, la personne est décrite par un nom, un age, une adresse et une profession. L'adresse est elle-même décrite par une rue et une ville et une rue est définie par un numéro et un nom.

2.3.1 OEM

La modélisation la plus commune des données semi-structurées est le modèle OEM présenté dans le projet d'intégration TSIMMIS (Papakonstantinou et al, 1995). Dans ce projet, les données semi-structurées sont modélisées sous forme de graphe orienté. Une donnée OEM est représentée par une collection d'objets. Chaque objet peut être atomique ou complexe. La valeur d'un objet atomique répond à un type de base (entier, chaîne de caractères, image, son, etc.). La valeur d'un objet complexe est un ensemble de couples (nom d'attribut, objet). Le graphe comporte une racine.

La figure 2.2 est un exemple de donnée OEM. Les chiffres 1, 2,3, etc. sont des identifiant d'objets (OID). La racine 1 désignée dans le schéma par l'étiquette « divertissement », regroupe deux objets de type complexe étiqueté « restaurant ».

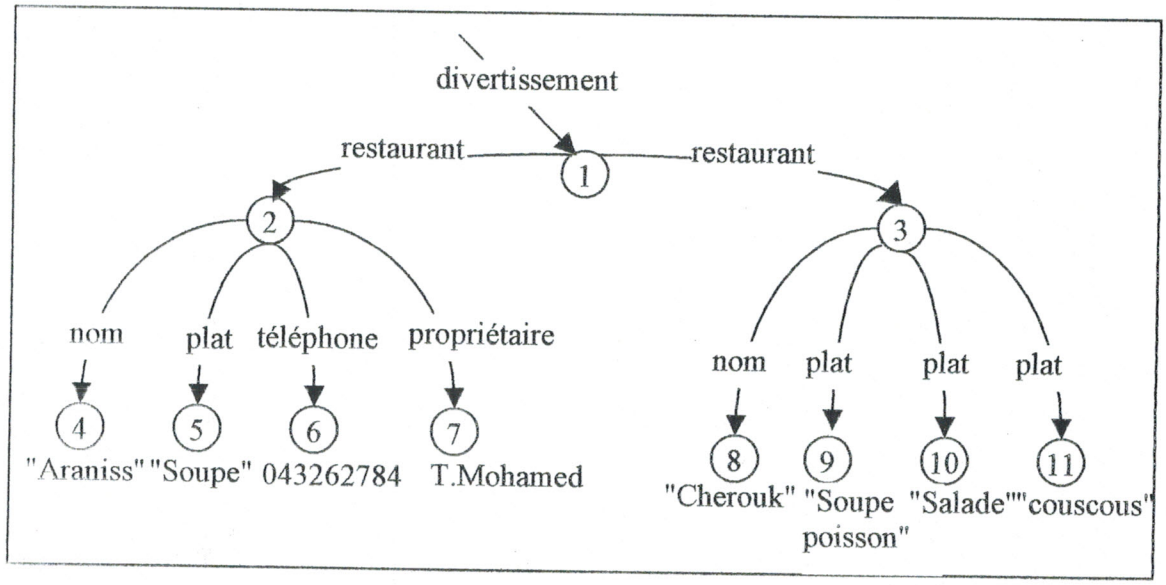


Fig. 2.2 Exemple de graphe OEM

2.3.2 XML

XML (Bray et al, 1998) (eXtended Markup Language) est un format textuel extensible de description de document défini par le W3C. De la famille des langages de marquages SGML (Goldfarb, 1991) (ISO 8879 :1986), il permet de s'adapter à quasiment tous les domaines où l'on a besoin de structurer de l'information de façon portable.

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

XML est conçu de façon à faciliter l'intégration et l'échange de données entre applications. Il isole le formatage et le rendu des documents par rapport à se structure. C'est à des langages des style spécifiques tels que XSL (eXtended Style Sheet) (Clark &

Deach, 2001) qu'on laisse le soin de s'occuper du rendu de la page XML lors de la publication.

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

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

- *bien formé* : quand il respecte la syntaxe du langage XML définie par le W3C ;
- valide : quand il est associé à une définition de type de document et qu'il la respecte (nom des éléments, type, répétition et ordre d'apparition dans le document).

Un document XML *bien formé* est un document XML qui respecte certaines règles simples :

1. Il existe un et un seul élément racine qui contient tous les autres éléments.
2. Les balises sont correctement imbriquées : chaque balise ouvrante a une balise fermante associée et il n'y a pas de chevauchement.
3. Le nom des balises est libre mais il contient au moins une lettre.
4. Les attributs des balises, lorsqu'ils existent, doivent comporter obligatoirement une valeur qui doit toujours apparaître entre doubles apostrophes.
5. Quand un élément est vide, les balises peuvent être simplifiées :<balise></balise> est identique à <balise/>.

La représentation XML du graphe de données semi-structurées de l'exemple de la figure 2.2 est donnée dans le document 2.2. Un attribut catégorie (prenant la valeur '3' puis '5') a été aussi rajouté à l'élément restaurant (les attributs n'ont pas d'équivalents en OEM).

```
<?xml version="1.0" encoding=="ISO-8859-1" standalone=="yes">
<divertissement>
      <restaurant, categorie=="3">
            <nom>Araniss</nom>
            <plat>Soupe</plat>
            <téléphone>043262784</téléphone>
            <propriétaire>T.Mohamed</propriétaire>
      </restaurant>
      <restaurant, categorie=="5">
            <nom>Cherouk</nom>
            <plat>Soupe poisson</plat>
            <plat>Salade</plat>
            <plat>couscous</plat>
      </restaurant>
</divertissement>
```

Document 2.2

XML est à présent le format standard utilisé pour représenter des données semi-structurées, et (Goldman et al,1999) montre que les projets utilisant OEM peuvent migrer aisément vers XML.

2.4 Métadonnées pour données semi-structurées

Les données semi-structurées sont auto-descriptives : le schéma est défini dans les données, et aucune structure n'est précisée *a priori*. Ceci permet une grande flexibilité dans le traitement (requêtes, stockage, chargement), les mises à jour et les changements structurels de telles données. En contrepartie, une telle souplesse comporte les inconvénients suivants (Suciu, 1998) :

- *le stockage des données inefficace* : le schéma doit être pour chaque donnée ;
- *les requêtes sont complexes à évaluer de façon optimale* : même une simple expression de chemin rationnelle implique souvent le parcours complet du graphe ;
- *les requêtes sont complexes à formuler* : l'utilisateur ne peut s'appuyer que sur des informations incomplètes ou inexistantes pour pouvoir formuler une requête pertinente.

Des observations sur les applications exploitant les données semi-structurées montrent que, malgré tout, les données possèdent souvent une certaine structure régulière. Il conviendrait de tirer parti de cette dernière afin de résoudre les problèmes énumérés ci-dessus. Des recherches ont été effectuées afin de décrire et d'exploiter cette régularité sans toutefois utiliser un schéma prédéfini rigide. Deux principales approches ont été proposées :

- **Un schéma déduit** : Calculé automatiquement *a posteriori* à partir des données. Ces schémas sont rigides et décrivent la structure de données de façon assez précise, et sont recalculés ou mis à jour régulièrement. On peut citer les *guides de données* (Goldman & Widom, 1997), les T-indexes (Milo & Suciu, 1999) et les types datalog unaires (Nestorov et al, 1997) ;
- **Un formalisme de schéma flexible** : Défini *a priori*. Il permet de décrire les données en offrant un degré de précision modulable sur la structure de données. On doit à l'aide de ce formalisme, aussi bien, décrire un schéma rigide et typé, qu'un schéma n'imposant aucune structuration dans les données. De telles formulations ont fait l'objet d'étude et de standardisation, notamment DTD (Bosak et al,1998) et XML-Schéma (Thompson et al, 2001) (Biron & Malhotra, 2000).

On appellera *forêt* une collection de documents XML de même nature stockés ensemble.

2.4.1 Guide de données

Dans un SGBD relationnel ou objet, le schéma est considéré comme *a priori.* Les tables sont définies avant d'insérer les données. Les données semi-structurées présentées sous forme de documents XML peuvent répondre à un certain format prédéterminé, notamment afin de définir le type de certains éléments (chaîne de caractères, entier, type flottant, type complexe). La DTD et son évolution vers XML-Schéma sont des formats permettant de spécifier les types. Que ce soit un schéma défini au niveau SGBD, une DTD ou un XML-Schéma, même si les définitions de types peuvent être très lâches (XMLèSchéma permet des descriptions optionnelles ou alternatives), La structure de données garde une certaine rigidité quand à l'insertion de nouvelles données dont le structure ne s'adapte pas à celle définie pour son type. C'est pour cette raison que (Goldman & Widom, 1997) a défini un *guide de donnée* basé sur le principe de « factorisation » des chemins communs *a posteriori.*

Le guide de donnée est un schéma faible généré à partir d'un ensemble de document par union des arbres de structure décrivant tous les cheminements possibles dans la collection et par typage de données en texte.

2.4.2 DTD

Un DTD (Document Type Definition) permet aux utilisateurs de définir leurs propres balisages, attributs et entités pour des documents de types SGML ou XML.
Un document DTD est signalé par une déclaration de type de document (Document Type Declaration) par l'intermédiaire de la chaîne <!DOCTYPE. Il peut être défini explicitement ou en référence vers une entité externe. Un DTD définit une structure de document par l'intermédiaire d'un ensemble de déclarations de balisage. Une déclaration de balisage peut être une déclaration de type d'élément, une déclaration de liste d'attributs, une déclaration d'entité ou une déclaration de notation.

2.4.3 XML-Schema

XML-Schema (Thompson et al, 2001) (Biron & Malhotra, 2000) est un standard permettant de définir et de typer un document XML dans un formalisme proche de celui des bases de données. Il devient alors possible de définir des schémas de bases de données comme dans un modèle objet. Les éléments et les attributs globaux sont créés par des déclarations qui apparaissent directement à l'intérieur de l'élément schéma.

2.5 Intégration de données hétérogènes

Les données accessibles sur le réseau peuvent prendre diférentes formes et ont chacune leurs spécificités propres.

Un système de médiation est un outil puissant permettant un accès simple aux diférentes informations collectées de sources de données pouvant être très disparates. Il

doit intégrer des données diverses afin de pouvoir offrir à l'utilisateur une vue centralisée et uniforme des données en masquant les caractéristiques spécifiques à leur localisation, méthode d'accès et formats.

2.5.1 Diversité des sources de données

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

La localisation d'une source de données englobe tout aussi bien le référencement du site sur lequel se situe la source (URL, adresse IP + port, annuaire LDAP), que le protocole de communication utilisé (TCP/IP, IPX, Appletalk), les moyens d'accès à la base (ODBC, JDBC) ainsi que le support (pages web, SGBD).

Le type de données géré par une source peut être structuré (base de données relationnelles), semi-structuré (sources XML, OEM) ou non-structuré (images, multimédia, texte libre). Les possibilités d'interrogation sont aussi nombreuses, et vont de langages de requêtes évolués et standardisés (SQL, OQL) ou propriétaires (Lorel) à de simples interfaces de programmation ou encore des recherches par motifs (moteur de recherche web). Nous ferons abstraction des interfaces graphiques de requêtes utilisateurs, trop spécifiques et inadaptables dans le cadre d'une intégration.

Enfin, les formats des résultats complètent les disparités qui peuvent exister entre les diférentes sources de données. Celles-ci peuvent être formatées suivant divers standards (XML, HTML) ou encore accessibles par des structures de programmation variées (ResultSet, OEM).

Les sources de données auxquelles nous avons accès dans un contexte interconnecté via le web se sont diversifié selon les directions que nous avons citées, de sorte que nous pouvons à présent parler véritablement de sources de donn!! ees hétérogènes.

2.5.2 Principes généraux de la médiation

La diversification des sources de données telles que nous venons de les décrire, a conduit tout naturellement à l'idée d'offrir à l'utilisateur un système permettant d'avoir une vue centralisée uniforme des données.

Les sources de données sont encapsulées par un adaptateur spécifique permettant de pallier les disparités -voire les limitations des différentes bases- et d'offrir une interface uniforme au composant d'évaluation ; le traitement de la requête est ainsi réparti sur le médiateur et sur les adaptateurs.

Afin que l'utilisateur ait l'illusion d'un accès à une seule et unique base de données virtuelle, ce système doit posséder un langage de requête unique et retourner à l'utilisateur un résultat sous un format lui aussi unique.

2.5.3 Intégration de schéma

On peut classifier (Halevy, 2000) les systèmes d'intégration de données suivant la relation entre les schémas des sources locales par rapport au schéma unifié global sur le médiateur. On distingue trois types de système : un schéma global comme vue sur des schémas locaux GAV (Global As View) ou des vues locales comme vues du schéma global LAV (Locale As View) (figure 2.3), et un système hybride GLAV qui regroupe les deux approches.

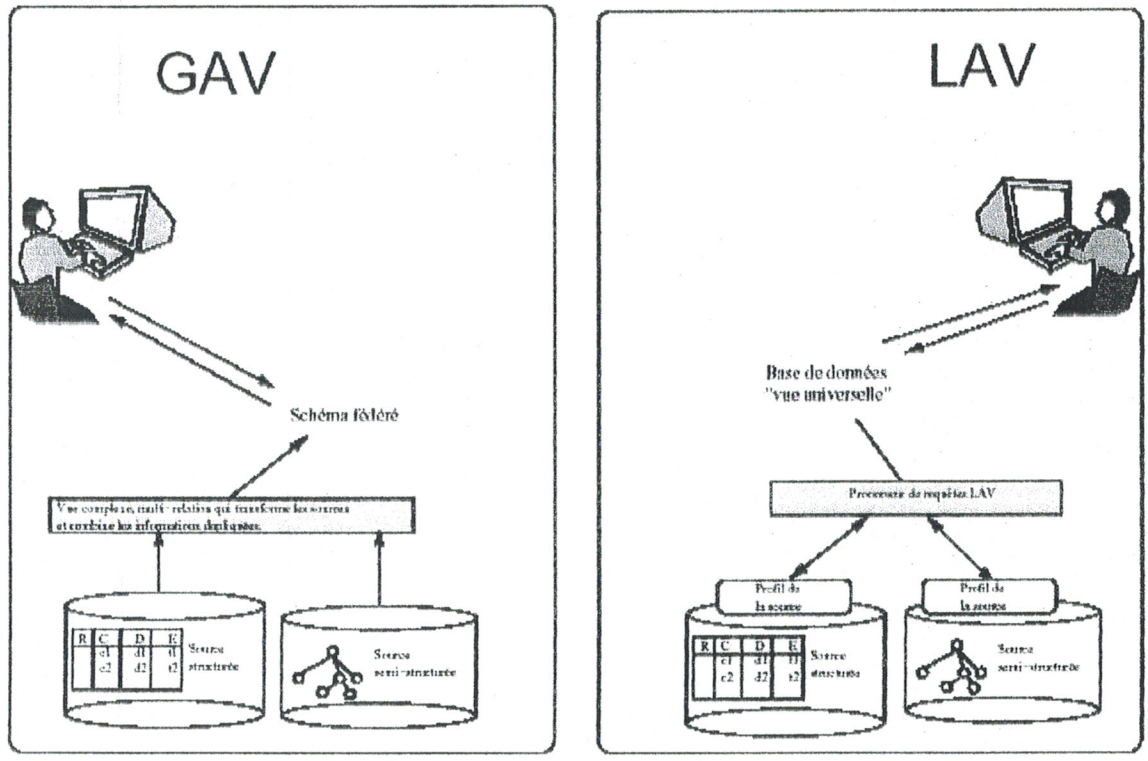


Fig. 2.3 - Comparaison d'architectures GAV et LAV.

Dans l'approche GAV, la transformation d'une requête sur le schéma global en requête sur le schéma local est une simple opération faite par le gestionnaire de vue.
Dans le cas d'une approche LAV, la requête sur le schéma global doit être reformulée suivant les schémas des sources locales. D'un autre côté, dans une architecture GAV, une modification sur l'ensemble des sources locales ou sur leur schéma entraîne une reconsidération complète du schéma global. Dans l'architecture LAV, chaque source est spécifiée de manière indépendante. Un changement local de schéma est considéré en mettant à jour la vue locale. De plus, si les données des sources locales n'ont pas le même format (relationnel, semi-structuré), il est difficile de définir le schéma global comme vue des sources de diférents formats. En utilisant une approche LAV, chaque source peut être décrite séparément par un mécanisme de vue spécifique à son format.

Les architectures les plus souvent utilisées sont des architectures GAV (TSIMMIS, GARLIC, DISCO, YA T, SilkRoute, Xperanto). Très peu utilisent l'approche LAV (Information Manifold, AGORA).

2.5.4 Evaluation de requête

Le médiateur présente des vues intégrées des sources de données. Ainsi une requête formulée au médiateur est posée indépendamment de la localisation des différentes données intervenant pour calculer le résultat.
Cela introduit trois difficultés :

La décomposition d'une requête : il s'agit à partir d'une requête posée sur une vue intégrée, de localiser les données intervenant dans sa résolution, de produire des sous-requêtes spécifiques à chacune des sources, d'ordonner ces sous-requêtes et éventuellement d'introduire des opérateurs au niveau du composant de médiation afin de compléter cet ensemble de sous-requêtes. La localisation des sous-requêtes nécessite des structures spécifiques de gestion de méta-données ;
La recomposition des résultats : une fois les sous-requêtes soumises à chacune des sources, il s'agit de savoir recomposer les différents résultats entre eux. Les résultats de chacune des sous-requêtes peuvent éventuellement faire l'objet d'un traitement additionnel soit parce que l'évaluateur de sous-requête n'a pas la capacité nécessaire pour traiter entièrement cette dernière, soit parce que les sous-requêtes comportent des dépendances entre elles ;
Au niveau de l'optimisation : le médiateur a rarement une vision sur la façon dont sont traitées les sous-requêtes au niveau des sources (placement des données, type de stockage, indexation, stratégie d'évaluation). De plus la distribution des données sur des sources disjointes ne permet pas d'utiliser directement les algorithmes employés normalement dans le cas d'un SGBD centralisé.

2.5.5 Cas particulier des données semi-structurées

Les données semi-structurées ont une structure de graphe. Ces données ayant par nature une structure qui n'est pas encore complètement connue, leurs manipulations peut s'avérer complexe dans le cas d'une restructuration suivant un autre schéma, ou encore dans le cas d'une composition de deux objets semi-structurées. Enfin lorsqu'on prend en compte dans les opérations précédentes, le cas des attributs multivalués. Les problèmes lors de l'évaluation de la requêtes sont alors liés à :
La restructuration : comment transformer un graphe suivant une structure cible, sans perdre d'information. ;
La composition : comment composer plusieurs graphes en tenant compte des branches communes. Soit la figure 2.4 (A) décrivant le schéma d'un objet de type personne et le schéma d'un objet de type voiture. Comment composer deux données répondant à ce schéma suivant le schéma représenté en (B), sachant que les attributs ***Personne/Voiture/Imm*** et ***Voiture/Numero*** correspondent, et les attributs ***Personne/Voiture/Caract/Annee*** désigne le même objet que ***Voiture/Annee*** ;
Aux objets multivalués : comment tenir compte des attributs multivalués lors de

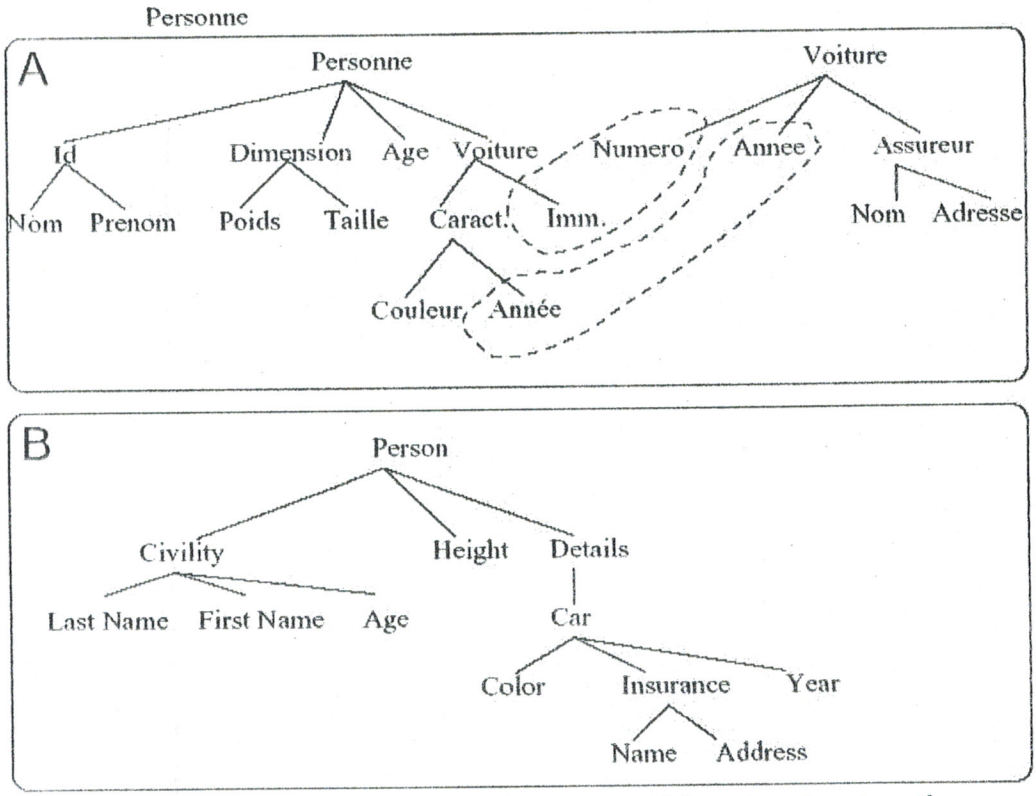


Fig. 2.4 -Schéma avant et après recomposition et restructuration

la composition, de l'évaluation ou de l'élimination des doublons.
Soit le schéma initial des données représenté en (a) dans la figure 2.5. Supposons que l'on veuille restructurer les données suivant le schéma représenté en (b) sur la même figure.

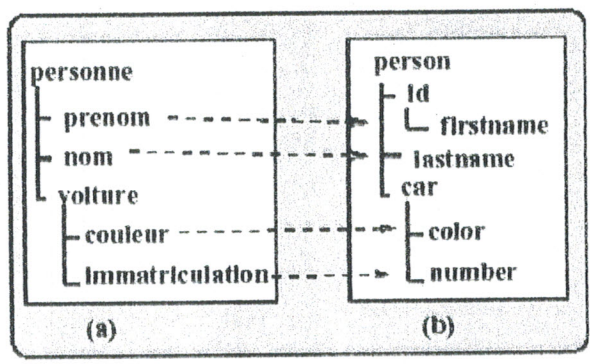


Fig. 2.5 - Schéma avant et après recomposition des données

Soit les deux données représentées en (c) et (d) de la figure 2.6 correspondant au schéma (a) de la figure 2.5.
Le nœud fils ***voiture*** de ***personne*** de l'objet représenté en (d) est multi-valué.
Lors de la restructuration de (d), doit-on éclater la structure initiale suivant en autant d'objets que d'instances de nœuds multivalués comme représenté dans la figure 2.7 (e) ou conserver le groupement comme représenté en (f) ?
Le nœud fils ***prenom*** de ***personne*** de l'objet représenté en (c) est multivalué. Un nouveau nœud ***id*** est placé comme parent du nœud associé à ***prenom***. Dans le résultat final, doit-on dupliquer le nœud ***id*** (figure 2.8 (g)) ou le nœud ***prenom*** (figure 2.8 (h)) ?

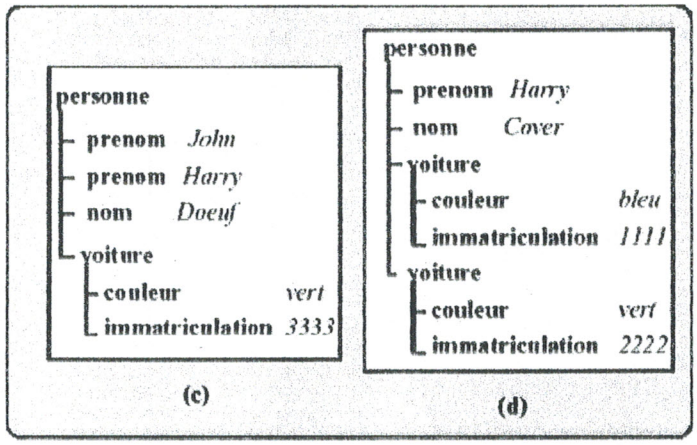


Fig. 2.6 - Données à traiter

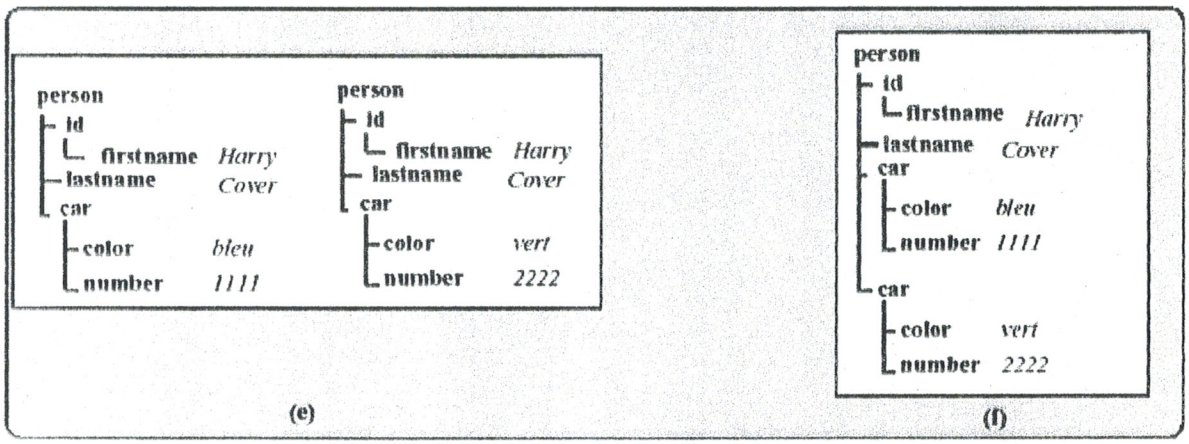


Fig. 2.7 - Restructuration de la donnée (d)

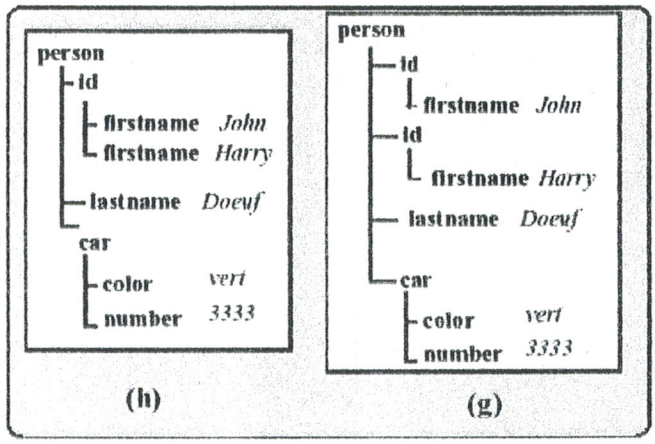


Fig. 2.8 - Restructuration de la donnée (c)

Les données semi-structurées ont une structure arborescente mal connue. Elles n'ont de ce fait aucune description précise du nombre de niveaux à partir d'une racine, ni de la largeur maximum du graphe renvoyé. Par exemple, si on ne limite pas le nombre de

niveaux ramenés, et si on choisit par exemple de récupérer la racine correspondante à la base de données semi-structuré, on peut être amené à charger la base entière.

Enfin, le dernier problème induit par le concept semi-structuré est lié à la décomposition et à la recomposition d'arbres suivant un modèle, sachant que certains attributs peuvent être multivalués.

2.5.6 Architecture de médiation

L'intégration des données hétérogènes distribuées a donné lieu à un modèle d'architecture (DARPA I3) proposée par DARPA et Gio Wiederhold (Wiederhold, 1992) représentée par la figure 2.9. Ce modèle distingue deux composants principaux : les médiateurs et les adaptateurs.
Les adaptateurs sont dédiés à l'hétérogénéité des données : ils convertissent l'interface d'entrée/sortie de la source gérée en une interface commune à tous les adaptateurs et au médiateur.
Les médiateurs sont dédés à la distribution des données : leur rôle est de regrouper les informations données par les adaptateurs.

Les architectures des bases de données fédérées ont progressivement convergé vers une vue unifiée

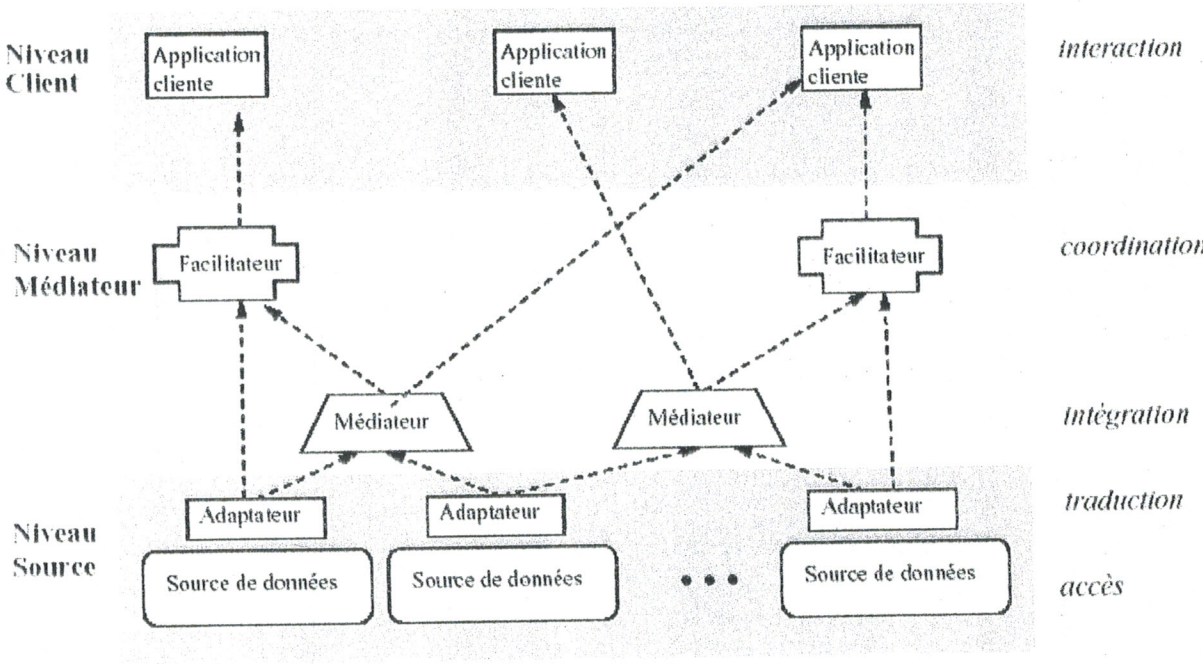


Fig. 2.9 - Architecture DARPA I3

Cette architecture est composée de trois niveaux comme suit :

Le niveau source : comporte les différentes sources de données ; à l'aide d'un adaptateur (wrapper), il est capable de communiquer avec les médiateurs et facilitateurs du niveau supérieur, en leur fournissant une vue homogène de la source à laquelle il est associé. Un adaptateur accepte une requête donnée dans le langage commun du médiateur, la transcrit dans le langage natif de la source et exécute la requête. Le résultat de la requ^ete

transmise sous forme native est alors transformé suivant le modèle de données global du médiateur et renvoyé à celui-ci.

Le niveau médiateur : comporte des médiateurs permettant d'intégrer les données en provenance de différentes sources afin de répondre aux requêtes des utilisateurs. Ce module joue un rôle actif dans la couche entre les applications utilisateurs et les sources de données. Son rôle est de fournir à la couche supérieure une vue centralisée des sources qui sont hétérogènes et distribuées. On trouve aussi à ce niveau des facilitateurs permettant d'identifier les sources qui peuvent être des sources de données ou des médiateurs, et de composer les réponses pour les utilisateurs.

Le niveau client : comporte les applications clientes (navigateurs, programmes d'application, interface graphique).

L'intégration d'une nouvelle source se résume à développer un adaptateur pour cette dernière et le rendre accessible par le médiateur (Cluet, 1998).

Un langage de requête commun ainsi qu'un modèle de données commun entre les facilitateurs, le médiateur et les adaptateurs doivent être définis pour toute l'architecture. Le rôle d'un adaptateur consiste essentiellement à permettre la traduction du langage de requête commun dans le langage natif de la source qu'il gère, et la traduction du modèle de donnée de la source en modèle de données commun.

Les sources de données accessibles peuvent être très différentes : certaines sont des SGBD relationnelles, d'autres des SGBD objets, certains encore sont des systèmes de chiers ou des pages web et d'autres des moteurs de recherche, ou encore même des médiateurs.

Ces disparités doivent être prises en compte dans le développement de l'adaptateur spécifique, mais cela n'est pas toujours évident. En effet, certaines sources (par exemple un moteur de recherche, ou une page web) offrent des capacités limitées d'interrogation (par rapport à un SGBD relationnel par exemple). Le médiateur doit pouvoir en tenir compte.
Ensuite, la répartition des sources sur un réseau à l'échelle de l'Internet entraîne l'accroissement de situations d'indisponibilité des sources dues à des problèmes de communication ou de serveurs inaccessibles.

2.7 Architectures de médiations existantes

La plupart des architectures de médiation se sont orientées vers l'architecture DARPA I3. Les premières se sont appuyées sur des sources de données relationnelles (MULTIBASE (Landers & Rosenberg, 1982), MERMAID (Templeton et al, 1987), Information Manifold (Kirk et al, 1995)). Avec l'avènement des systèmes de bases de données objets ou orientées objets, il a fallu tenir compte de ce nouveau modèle de données dans les architectures de médiation (IRO-DB (Gardarin, 1997), PEGASUS (Ahmed et al, 1987), GARLIC (Haas et al, 1997), DISCO (Tomasic et al, 1996)). Enfin, récemment, sont apparues des sources à base de données semi-structurées, et les nouvelles architectures de médiation ont dû s'adapter, aussi bien dans des projets de recherche (TSIMMIS (Chawathe et al, 1994), MIX (Bornhovd, 1998), YAT (Cluet, 1998)

et AGORA (Manolescu et al, 2001)) que dans des produits industriels (NIMBLE (Nimble, 2002a) (Nimble, 2002b) (Nimble, 2002c) et Xperanto (IBM, 2002)).

2.7.1 TSIMMIS, GARLIC et MIX

TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) (Chawathe et al, 1994) (Garcia-Molina et al, 1994) est un projet permettant de développer des outils qui facilitent l'intégration de sources d'informations hétérogènes comportant des données structurées et semi-structurées.

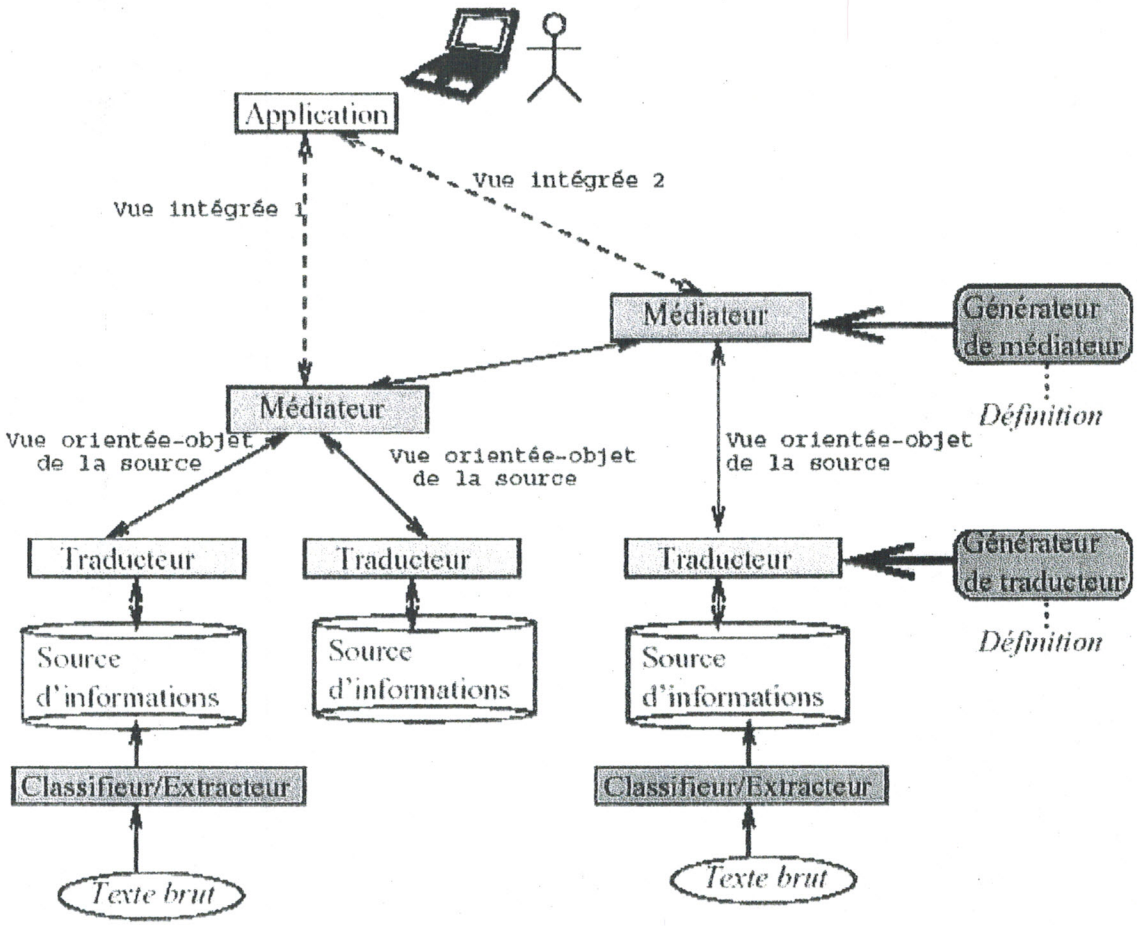


Fig. 2.10 - Architecture de TSIMMIS

De ce fait, TSIMMIS est une architecture composée des modules suivants (voir figure 2.10) :

- *classifieur/ extracteur de données sur le Web* : depuis du texte brut, des pages HTML ou des pages XML. Le classieur/extracteur identie et classie de tels objets (ex. est-ce un fichier texte ? une page web HTML ? un courrier électronique ?), et en extrait des propriétés (ex. date, auteur). Le classieur/extracteur est basé sur le système RUFUS (Shoens et al, 1993) développé au centre de recherche d'IBM Almaden ;
- *traducteur (ou adaptateur)* : permettant la transformation des requêtes et des données. Le traducteur convertit les requêtes OEM-QL sur des données du modèle commun OEM en requêtes spécifiques à la source. Il convertit ensuite les données résultats de la source en données du modèle commun OEM ;
- *médiateur* : permettant de combiner les informations des différentes sources ;

- *applications clientes* : le médiateur tout comme les traducteurs, prend des requêtes OEM-QL en entrée et renvoie des données résultats OEM. Des applications clientes (MOBIE MOsaic-Based Information Explorer) permettant de naviguer dans les données à travers le web ont été développées afin d'offrir une interface conviviale à l'utilisateur final.

Un des objectifs de TSIMMIS est d'intégrer des sources qui sont très hétérogènes, qui peuvent être peu structurées et qui sont susceptibles d'évoluer rapidement. TSIMMIS a introduit le formalisme OEM (Papakonstantinou et al, 1995) comme modèle de données interne et a adopée un langage de requête dérivé d'OQL : OEM-QL. Afin de pouvoir utiliser une source de données purement semi-structurées, un SGBD semi-structuré natif LORE basé sur OEM a aussi été développé.

Le projet GARLIC (Haas et al, 1997) est un projet semblable développé au centre de recherche d'IBM Almaden. Son objectif est l'intégration de diverses sources multimédia (images, vidéo, texte, objets médicaux, cartes géographiques).

L'architecture de GARLIC (figure 2.11), tout comme TSIMMIS, adopte l'architecture à base de médiateurs et d'adaptateurs de DARPA I3. Le modèle de données commun de GARLIC est orienté-objet, et le langage de requête sur ces données est une extension de SQL appelée GQL (Garlic Query Language).

Outre le modèle de données utilisé, la différence fondamentale entre l'approche de GARLIC et celle de TSIMMIS est dans la prise en compte des capacités des sources. Le médiateur de TSIMMIS assume que les adaptateurs auxquels il adresse les requêtes décomposées savent tout faire (Li et al, 1998), et c'est aux adaptateurs de pallier aux défaillances des sources. Le médiateur de TSIMMIS ne prend pas en compte les capacités des sources, et donc, la décomposition et l'optimisation des requêtes est plus simple à réaliser au niveau médiateur. Il y a une bonne répartition des tâches mais l'implémentation des adaptateurs est rendu plus lourde.

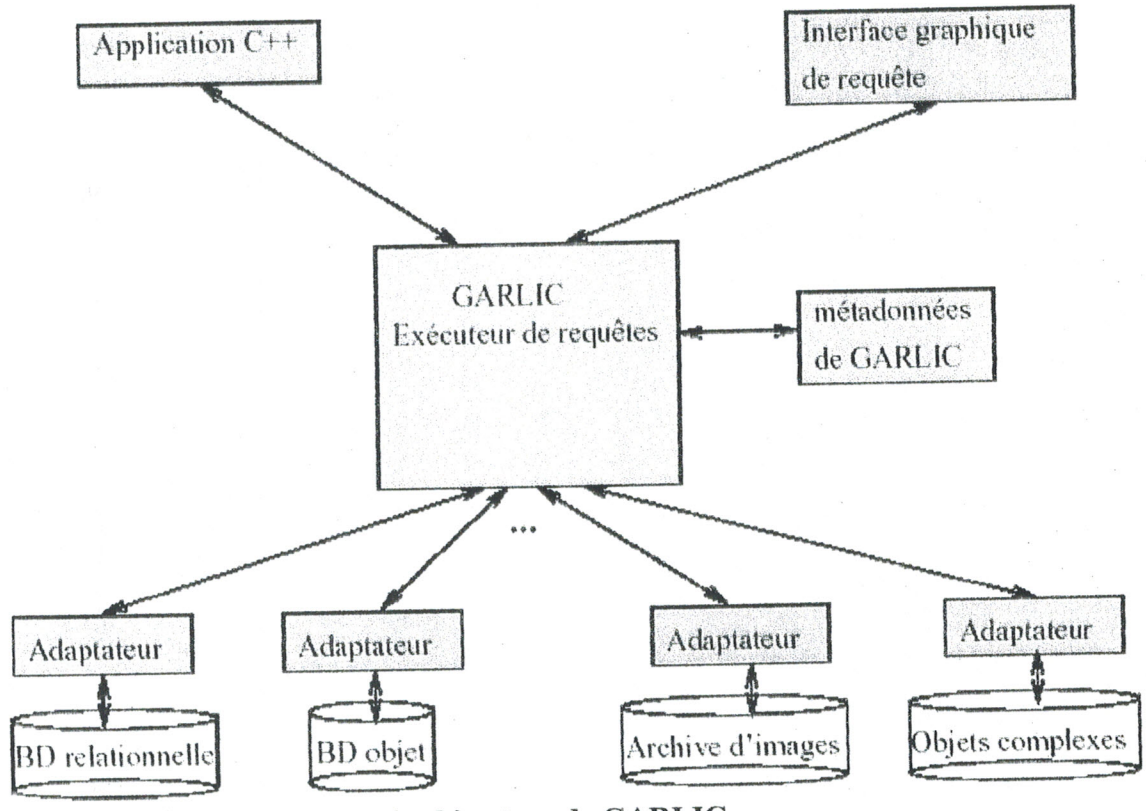


Fig. 2.11 - Architecture de GARLIC

Avec GARLIC, le médiateur prend en considération les capacités des sources aussi bien pour les calculs de coûts que pour les décompositions des requêtes envoyées. Son approche est plus fine et permet de traiter certaines requêtes impossibles à TSIMMIS. Les algorithmes de traitement du médiateur sont en revanche plus complexes et la centralisation entraîne un travail supplémentaire pour le médiateur. Un langage de description des capacités des sources est aussi nécessaire. Les pré-requis fonctionnels demandés aux adaptateurs de GARLIC se base sur le plus petit ensemble commun de propriétés. Au niveau des opérations d'exécution, il est demandé à l'adaptateur de savoir au minimum retrouver les objets d'une collection, et accéder aux attributs d'un objet donné.

GARLIC n'est pas un médiateur de données semi-structurées, mais l'intégration de sources multimédias (peu ou pas structurées) avec des sources relationnelles et objet a permis à GARLIC de détailler la définition des adaptateurs et créer un modèle d'architecture pouvant intégrer des sources très hétérogènes.

Il est à noter que TSIMMIS a ensuite adopté l'approche de GARLIC (Vassalos & Papakonstantinou, 1997) et LORE a fait évoluer son format de données semi-structurées OEM vers XML (Goldman et al, 1999). Cela a permis la conception de MIX (Bornhovd, 1998). MIX utilise donc XML comme modèle d'échange et XMAS (Xml Matching And Structuring language) comme méta-langage de requête. De la sorte, tous les langages basés sur du semi-structuré (YATL, XML-QL, UnQL et MSL) peuvent être convertis en XMAS qui peut être ensuite être utilisé par le médiateur.

2.7.2 STRUDEL

STRUDEL (Fernandez et al, 1998) est un constructeur de sites web permettant de définir quelles sont les données qui seront disponibles sur le site. L'idée principale est de séparer l'administration des données du site, la création et la gestion de la structure du site, et la représentation graphique des pages web nales. Pour cela, le constructeur de site crée un modèle uniforme de toutes les données disponibles sur le site. Puis, le constructeur de site utilise ce modèle pour définir la structure du site web en appliquant aux données, une requête de « définition de site ». Enfin, le constructeur de site spécifie la représentation graphique des pages en utilisant le langage de STRUDEL pour la définition de modèle (*template*) HTML. Les données résident soit sur des sources externes (SGBD, fichiers structurés), soit dans l'entrepôt (*repository*) interne de TRUDEL.

A chaque niveau du système STRUDEL, toutes les données (qu'elles soient internes ou externes) sont modélisées par un graphe orienté similaire à OEM. Un ensemble d'adaptateurs spécifiques se charge de convertir la représentation externe en graphe. La vue intégrée des données est appelée graphe de données.

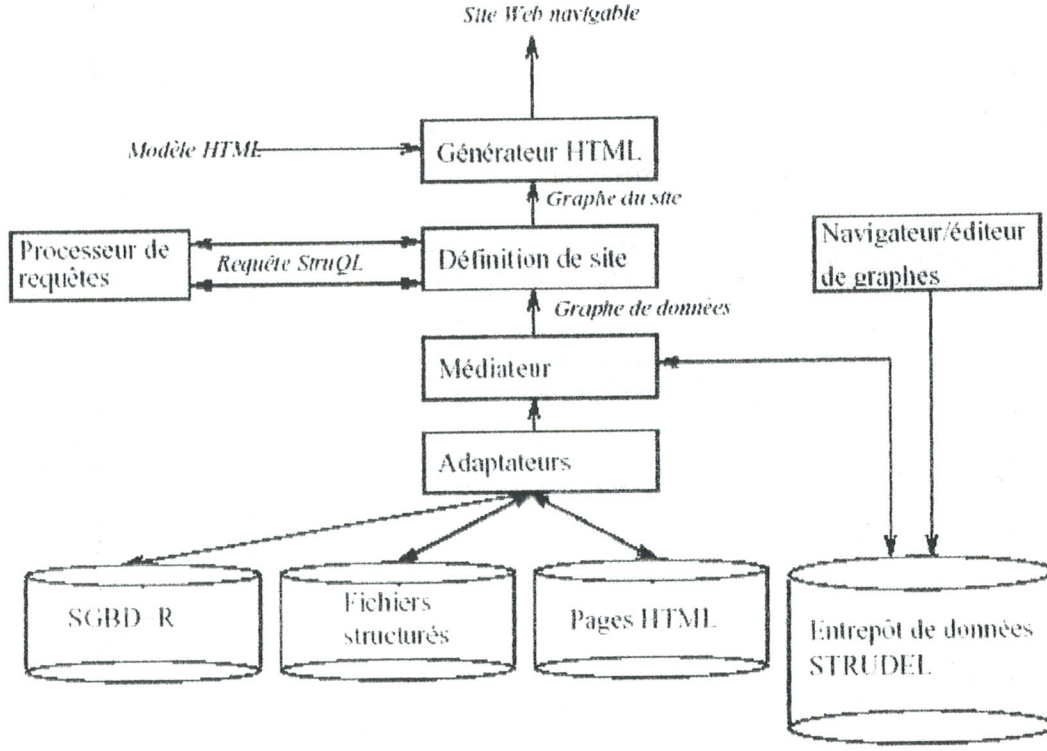


Fig. 2.12 - Architecture de STRUDEL

L'architecture de STRUDEL est représentée à la figure 2.12. Elle est composée de :

- *Entrepôt de données STRUDEL* : le graphe de données d'un site web est stocké dans l'entrepôt de données STRUDEL. Les données sont obtenues des adaptateurs qui convertissent les données des sources externes en des données au format interne semi-structuré utilisé par STRUDEL ;
- *Navigateur/ éditeur de graphe* : il permet à l'utilisateur de créer, mettre à jour et visualiser les graphes pouvant être utilisés pour le graphe de données et le graphe du site ;

- *médiateur* : il fournit une vue uniforme des données sous-jacentes. Plutôt que d'interroger les sources externes à la demande au moment de l'exécution de la requête, son approche est intégrer les données en stockant préalablement les données des sources externes dans l'entrepôt de données STRUDEL ;
- *Processeur de requêtes* : STRUDEL définit le langage STRUQL (STRUdel Query Language) pour réaliser des requêtes et restructurer des données semi-structurées. L'interpréteur de requête de STRUDEL utilise les opérateurs physiques traditionnels (jointure, restriction, sélection) ainsi que des opérateurs nécessaires pour l'interrogation de schéma (ex. trouver tous les noms d'attributs dans un graphe) ;
- *Générateur HTML* : pour produire la représentation graphique de chaque page du site web, un modèle HTML est associé à chaque nœud du graphe du site. Le résultat est un site web navigable.

2.7.3 YAT

De la même façon que STRUDEL, YAT intègre des sources de données dans un environnement Web. Il se base sur une représentation de graphes afin de représenter les données semi-structurées provenant des diférentes sources. Le langage YATL est un langage déclaratif de conversion/intégration à base de règles ; ce n'est pas un langage de requête et il ne fournit pas de possibilités de requête comme des expressions de chemins généralisées. Cependant, il permet des primitives de restructuration et des fonctions de Skolem pour manipuler des identiants et créer des graphes complexes.

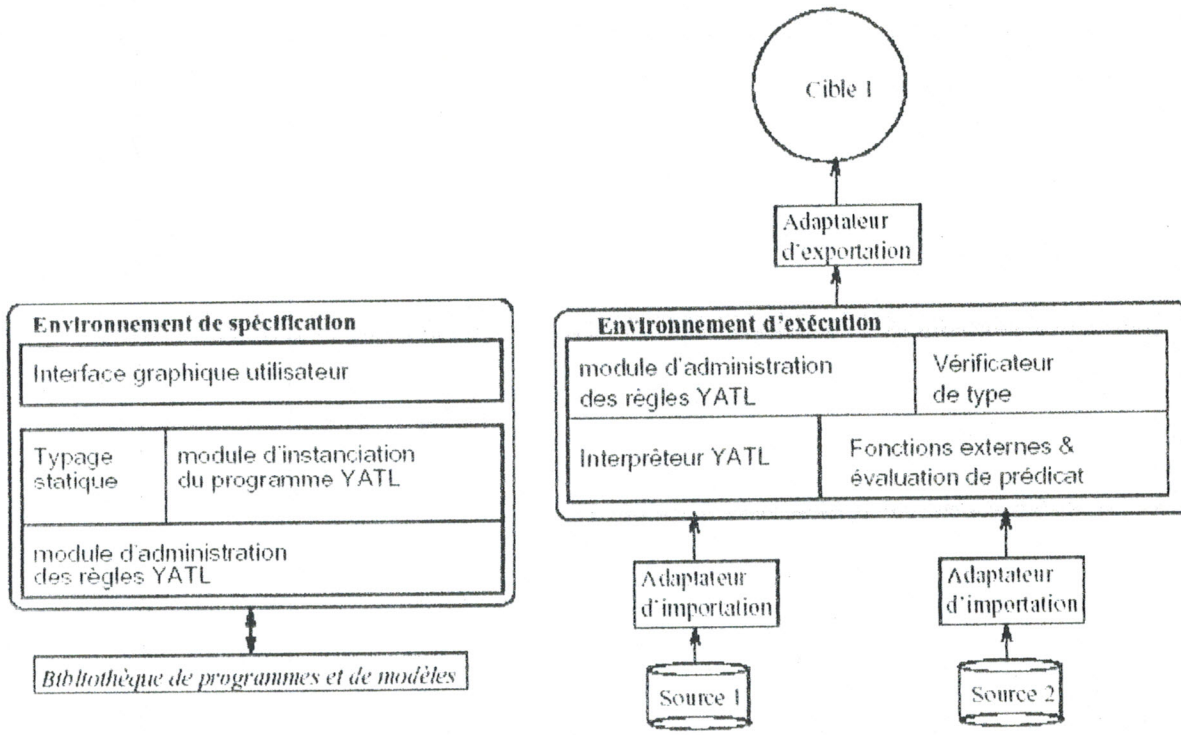


Fig. 2.13 - Architecture du système YAT

La figure 2.13 décrit l'architecture de YAT. Il y a trois parties principales : (1) l'environnement de spécication, (2) l'environnement d'exécution, et (3) une bibliothèque de programmes et format. Chacun des environnements repose sur le module de modèle

YAT et d'administration de règles YATL. L'environnement de spécification offre (i) un module permettant de vérifier ou déduire le type d'un programme, (ii) une interface graphique permettant à l'utilisateur de définir les traductions en utilisant (iii) le module d'instanciation pour personnaliser les programmes existants si besoin est.

L'environnement d'exécution utilise des adaptateurs pour importer (resp. exporter) des données depuis (resp. vers) YAT et un interpréteur pour réaliser la traduction.
L'interpréteur se base sur un module séparé pour traiter des fonctions externes et des prédicats. Si cela est requis, il vérifie les types à l'aide du vérificateur de type.

2.7.4 AGORA/LeSelect

AGORA (Manolescu et al, 2001) est le seul projet avec Information Manifold (Levy et al, 1996) à utiliser une approche LAV. A la différence de ce dernier qui utilise uniquement des données relationnelles, AGORA manipule aussi des données semi-structurées. Pour cela, il s'appuie sur le médiateur LeSelect (Caravel, 1998). LeSelect est un médiateur dont le modèle de donnée est de type modèle relationnel étendu et dont le langage d'interrogation commun est de type SQL. La motivation de AGORA (Florescu et al, 2000) est de compléter les fonctionnalités de LeSelect en :

- définissant un schéma virtuel, générique et relationnel décrivant le contenu de documents XML ;
- traduisant des requêtes spécifiques à XML dans le schéma d'intégration relationnel ;
- étendant l'optimisation de requêtes afin de pouvoir gérer les cheminements ;
- utilisant l'indexation textuelle pour améliorer les performances des recherches plein texte.

L'architecture d'AGORA est représentée à la figure 2.14.
L'objectif d'AGORA est de supporter les requêtes et l'intégration de sources relationnelles et semi-structurées. Le schéma global de AGORA est une DTD XML. Les sources relationnelles et XML sont décrits comme vues sur ce schéma global. Un schéma relationnel générique est utilisée comme interface entre ce schéma et les sources. Pour chaque type de nœud XML (élément, attributs, textes, commentaire), une table relationnelle est associée dans ce schéma.
De cette façon, les requêtes XQuery sont transformées en requêtes relationnelles et les résultats sous formes de tuples sont ensuite retransformés en XML.

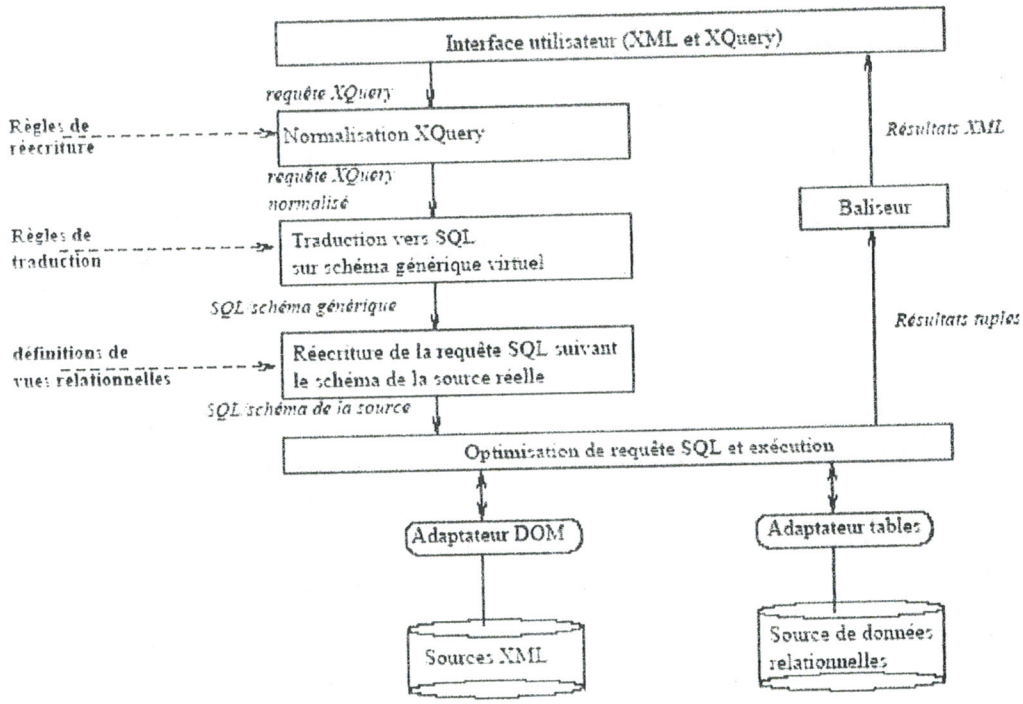


Fig. 2.14 - Architecture du système AGORA

2.7.4 PICSEL

Dans le projet PICSEL au Laboratoire de Recherche en Informatique de Paris-Sud, c'est le problème de l'intégration d'un grand nombre de sources de données ayant le format de documents XML qui est étudié. Un premier prototype, OntoMedia, a été développé pour extraire des composants d'une ontologie à base de classes à partir de DTD spécifiques à un domaine d'application donné (Giraldo & Reynaud, 2002). Une expérimentation réalisée à partir de DTD élaborés par un organisme de standardisation de transactions commerciales laisse penser que l'approche peut permettre la construction de systèmes médiateurs plus « ouverts ». Ces derniers pourraient être capables de regrouper a priori tous les systèmes dont l'interface est conforme aux standards ayant permis la construction de l'ontologie mais qui, au moment de la construction de cette ontologie, ne sont pas forcément identifiés. Une telle ouverture est intéressante dans une optique Web sémantique même s'il ne s'agit pas d'une approche complètement générale, s'appliquant à toutes les ressources identifiables via le Web, quelle qu'elles soient(Reynaud & Safar, 2002).

L'architecture du PICSEL est représentée à la figure 2.15. (Rousset et al, 2002)

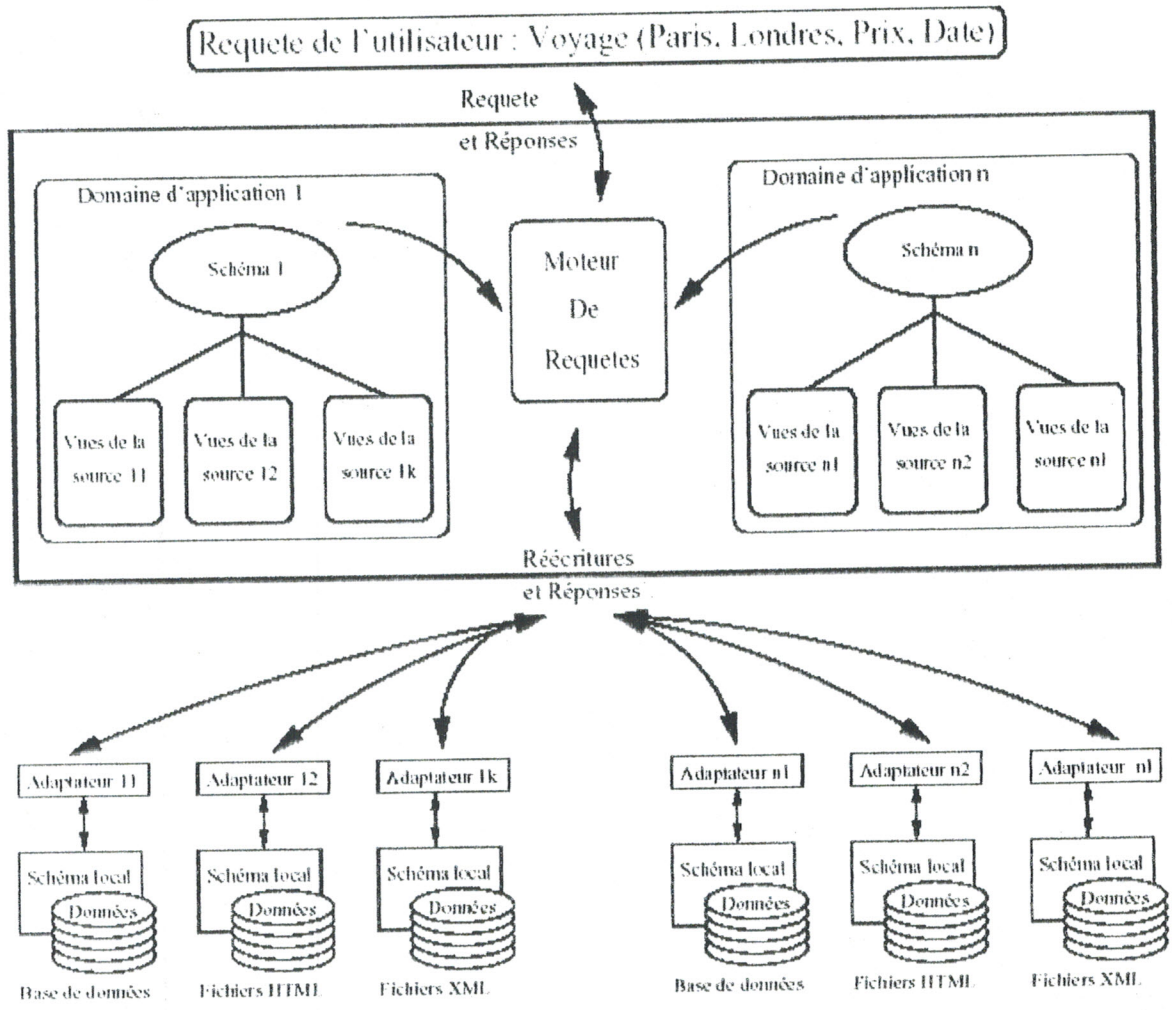


Fig. 2.15 - Architecture du PICSEL

Conclusion

Nous avons présenté dans ce chapitre une étude sur les systèmes d'intégration de données hétérogènes s'appuyant sur une approche centralisée dont les données restent stockées dans les sources et sont exprimées dans les termes d'un schéma global (ou ontologie globale), ce schéma global (ou ontologie globale) représente un point faible dans le cadre du Web sémantique décrit dans le chapitre suivant.

On peut penser que dans le cadre du Web sémantique, les systèmes de médiation distribués seront mieux adaptés par leur flexibilité. Dans ce contexte de médiation décentralisée, on va remédier au problème du schéma global. Il faut, donc, trouver et définir des correspondances sémantiques entre les ontologies manipulées par chacun des systèmes. Il faut pouvoir disposer d'une approche simple et naturelle de description de telles correspondances sémantiques entre ontologies qui est but de notre projet.

Références

ABITEBOUL .S, WIDOM .J, et LAHIRI .T. "An Unified Approach for Querying Structured Data and XML", *http ://www.w3.org/TandS/QL/SL98/pp/serge.html.* 1998.

ABITEBOUL .S. "Querying Semistructured Data". *In Proceeding of the 6th International Conference on Database Theory, Delphi, Greece,* 1997.

ABITEBOUL .S. "Semistructured Data Tutorial", 1998.

AHMED . R, ALBERT . J, DU . W, KENT . W, LITWIN .W, et SHAN . M. "An Overview of Pegasus". *In IEEE Conference on Data Engineering, Vienna,* April 1987.

BIRON . P et MALHOTRA . A. "XML Schema Part 2 : Datatypes", October 2000.

BONIFATI . A et CERI . S. "Comparative Analysis of Five XML Query Languages". *SIGMOD Record, 29(1) :68-79,* 2000.

BORNHOVD .C . "MIX - A Representation Model for the Integration of Web-based Data". *Technical report, Dep. CS, Darmstadt University of Technology, Germany,* 1998.

BOSAK . J, BRAY .T, CONNOLLY . D, MALER . E, NICOL .G, SPERBERG-MCQUEEN .C, WOOD . L, et CLARK .J . "W3C XML Specification DTD", 1998.

BRAY .T, PAOLI . J, et SPERBERG-MACQUEEN . C. "Extensible Markup Language (XML) 1.0 (W3C Recommendation) ", 1998.

CARAVEL. "LeSelect", 1998.

CHAWATHE .S, GARCIA-MOLINA .H, HAMMER .J, IRELAND .K, PAPAKONSTANTINOU .Y, ULLMAN .J, et WIDOM .J. "The TSIMMIS Project - Integration of Heterogeneous Information Sources". *In 10th Anniversary Meeting of Information Processing Society of Japan, Tokyo, Japan,* 1994.

CHRISTOPHIDES .V, ABITEBOUL .S, CLUET .S, et SCHOLL . M. "From structured documents to novel query facilities". *In In Proc. of ACM SIGMOD Conf. on Management of Data, pages 313-324, Minneapolis, Minnesota,* Mai 1994.

CHRISTOPHIDES .V, CLUET .S, et SIMEON .J. "On Wrapping Language and efficient XML Integration". *In ACM SIGMOD Conference on Management of Data,* 2000.

CLARK .J et DEACH .S. "Extensible Stylesheet Language (XSL) ", 2001.

CLARK .J et DEROSE .S. "XML Path Language (XPath) Version 1.0", 1999.

CLUET .S. "Your Mediators need Data Conversion ! " *In International Conference on Management of Data ACM SIGMOD, Seattle,* 1998.

DEUTSCH .A, FERNANDEZ .M, FLORESCU .D, LEVY .A, et SUCIU .D. "XML-QL : A Query Language for XML", 1998.

FERNANDEZ .M.F, FLORESCU .D, KANG .J, LEVY .A.Y, et SUCIU .D. "Catching the Boat with Strudel : Experiences with a Web-Site Management System". *In SIGMOD Conference, pages 414-425*, 1998.
FLORESCU .D, MANOLESCU .I, KOSSMANN .D, et XHUMARI .F.
"Agora : Living with XML and Relational". *In Proceeding of the Conference on Very Large Data Base, Cairo, Egypt*, Février 2000.

GARCIA-MOLINA .H, HAMMER .J, IRELAND .K, PAPAKONSTANTINOU .Y, ULLMAN .J, et WIDOM .J. "Integrating and Accessing Heterogeneous Information in TSIMMIS". *In 10th Anniv. Meeting of Information Processing Society of Japan, pages 7-18, Tokyo, Japan*, 1994.

GARDARIN .G et YOON .S. "Hyoql : A query language for structured hypermedia documents". *The International Journal of Microcomputer Applications*, 1996.

GARDARIN .G. "Multimedia Federated Databases on Intranets : Web-Enabling IRO-DB". *In 8th International Conference on Database and Expert Systems Applications (DEXA'97), France*, 1997.

GARDARIN .G. "XML". *Dunod*, 2002.

Giraldo G & Reynaud C. "Construction semi-automatique d'ontologies à partir de DTDs relatives à un même domaine", *IC'02 (Ingénierie des Connaissances 2002), Rouen, France*, 2002.

GOLDFARB .C. "The SGML Handbook". *Clarendon Press*, 1991.

GOLDMAN .R, MCHUGH .J, et WIDOM .J. "From Semistructured Data to XML : migrating the Lore Data Model and Query Language". *In Proceeding of the 2nd International Workshop on the Web and Databases (WebDB'99), Philadelphia, Pennsylvania , USA*, June 1999.

GOLDMAN .R et WIDOM .J. "DataGuides : Enabling Query Formulation and Optimization in Semistructured Databases". *In Proceeding of the Conference on Very Large Data Base (VLDB), Athens, Greece*, 1997.

HAAS .L.M, KOSSMANN .D, WIMMERS .E.L, et YANG .J. "Optimizing Queries Across Diverse DataSources". *In 23th International Conference on Very Large Data Bases, pages 276-285, Athens, Greece*, 1997.

HALEVY .A. "Logic-based techniques in Data Integration". *Logic Based Artificial Intelligence*, 2000.

IBM. "XTables : Bridging Relation Technology and XML", 2002.

KIRK .T, LEVY .A, et SRIVASTAVA .D. "The Information Manifold". *Technical report, AT&T Bell Laboratories*, 1995.

LANDERS .T et ROSENBERG .R.L. "An Overview of MULTIBASE". *Distributed Databases*, 1982.

LEVY .A, RAJARAMAN .A, et ORDILLE .J. "Querying Heterogeneous Information Sources Using Source Description". *In Proceeding of International Conference on Very Large Data Base, Bombay*, 1996.

LI .C, YERNENI .R, VASSALOSV.V, GARCIA-MOLINA .H, PAPAKONSTANTINOU .Y, ULLMAN .J.D, et VALIVETI .M. "Capability Based Mediation in TSIMMIS". *In Proceedings ACM-SIGMOD International Conference on Management of Data, pages 564-566, Seattle, Washington,* 1998.

LUASCHER .B, PAPAKONSTANTINOU .Y, VELIKHOV .P, et VIANU .V. "View definition and DTD inference for XML", Janvier 1999.

LE MAITRE .J, MURISASCO .E, et ROLBERT .M. "From annotated corpora to databases : The SgmlQL language". *In John Nerbonne, editor, Linguistic Databases, pages 37-58. CSLI Publications, Stanford, California,* 1997.

MANOLESCU .I, FLORESCU .D, et KOSSMANN .D. "Answering XML Queries over Heterogeneous Data Sources". *In Proceedings of 27th International Conference on Very Large Data Bases (VLDB), pages 241-250, Rome, Italie,* Septembre 2001.

MILO .T et SUCIU .D. "Index Structures for Path Expressions". *In Intl Conf. on Database Theory,* 1999.

NESTOROV .S, ABITEBOUL .S, et MATWANI .R. "Inferring Structure in Semistructured Data". *In Proceeding of the Workshop on Management of Semistructured Data, Arizona, USA,* May 1997.

NIMBLE (a). "Next Generation Data Integration", 2002.

NIMBLE (b). "Nimble Datasheet", 2002.

NIMBLE (c). "Nimble Integration Suite", 2002.

PAPAKONSTANTINOU .Y, GARCIA-MOLINA .H, et WIDOM .J. "Object Exchange across Heterogeneous Information Sources". *In Int. Conf. on Data Engineering, Taipei,* 1995.

REYNAUD .C et SAFAR .B, "Aide à la formulation de requêtes dans un médiateur", *RFIA'02 (Reconnaissance des Formes et Intelligence Artificielle 2002), Angers, France,* 2002.

ROBIE .J, LAPP .J, et SCHACH .D. "XML Query Language (XQL) ", 1998.

ROBIE .J, CHAMBERLIN .D, et FLORESCU .D. "Quilt : an XML Query Langage", March 2000.

ROUSSET .M, BIDAULT A, FROIDEVAUX .C, GAGLIARDI H, GOASDOUE .F, REYNAUD .C ET SAFAR . B, "Construction de médiateurs pour intégrer des sources d'information multiples et hétérogènes : le projet PICSEL", *Revue I3, Vol.2, N°1, p. 9-59.* 2002

SHOENS .K, LUNIEWSKI .A, SCHWARZ .P, STAMOS .J, et THOMAS .J. "The Rufus System : Information Organization for Semi-Structured Data". *In R. Agrawal, S. Baker, et D. Bell, editors, 19th International Conference on Very Large Data Bases, pages 97-107, Dublin, Ireland,* August 1993.

SUCIU .D. "Semistructured Data and XML". *In International Conference on Foundations of Data Organization*, 1998.

TEMPLETON .M, BRILL .D, DAO .S.K, LUND .E, WARD .P, CHEN .A.L, et MACGREGOR .R. "Mermaid - a Front-End to Distributed Heterogeneous Databases". *In IEEE Conference on Data Engineering, Vienna*, April 1987.

THOMPSON .H, BEECH .D, MALONEY .M, et MENDELSOHN .N. "XML Schema Part 1 : Structures", May 2001.

TOMASIC .A, RASCHID .L, et VALDURIEZ .P. "Scaling Heterogeneous Databases and the Design of DISCO". *In International conference on Distributed Computing Systems, Hong Kong*, 1996.

VASSALOS .V et PAPAKONSTANTINOU .Y. "Describing and Using Capabilities of Heterogeneous Sources". *In Proceeding of International Conference on Very Large Data Base, Athens, Greece,* August 1997.

W3C. "An XML Query Language" (XQuery 1.0), 2001.

WIEDERHOLD .G. "Mediators in the Architecture of Future Information System". *Computer*, 25 (3) :38-49, 1992.

Chapitre 3

Le Web sémantique

3.1 Introduction

La majorité du contenu du web actuellement produit est conçu pour être lu par des êtres humains, et pas pour être manipulé symboliquement par des programmes informatiques. Certes un document HTML est manipulé par un programme pour que la mise en page soit correcte. Mais ce traitement se limite à interpréter les balises de présentation HTML présentes dans le document. Ces balises se limitent à décrire la manière dont le document doit être présenté. La signification du contenu du document reste implicite et le document ne peut donc pas être manipulé sur base de cette signification. De manière générale les ordinateurs n'ont aucune méthode systématique pour traiter le contenu d'un document web sur base de leur sémantique.

La capacité à manipuler le contenu des documents web sur base de leur sémantique permettrait à des programmes de réaliser des tâches qui doivent être actuellement réalisées à la main et ouvre la voie à de nouvelles possibilités d'automatisation. Le web sémantique va structurer le contenu sémantique des documents web et ainsi permettre à des programmes d'interpréter leur contenu et de raisonner dessus. Le web sémantique va créer un environnement où des programmes, que l'on appelle des agents logiciels, pourront exécuter des tâches complexes aux noms d'utilisateurs qui leur auront délégué ces tâches. Pour les accomplir les agents logiciels devront communiquer entre eux, et interpréter le contenu échangé de la même manière, c'est à dire en interprétant les termes décrivant le contenu de la même manière.

3.2 Le web sémantique

3.2.1 Introduction

L'expression Web sémantique, due à Tim Berners-Lee (Berners-Lee et al., 2001)au sein du W3C, fait d'abord référence à la vision du Web de demain comme un vaste espace d'échange de ressources entre êtres humains et machines permettant une exploitation, qualitativement supérieure, de grands volumes d'informations et de services variés. Espace virtuel, il devrait voir, à la différence du Web que nous connaissons aujourd'hui, les utilisateurs déchargés d'une bonne partie de leurs tâches de recherche, de construction et de combinaison des résultats, grâce aux capacités accrues des machines à accéder aux contenus des ressources et à effectuer des raisonnements sur ceux-ci.

Le Web actuel est essentiellement syntaxique, dans le sens que la structure des documents (ou ressources au sens large) est bien définie, mais que son contenu reste quasi inaccessible aux traitements machines. Seuls les humains peuvent interpréter leurs contenus. La nouvelle génération de Web – Le Web sémantique – a pour ambition de lever cette difficulté. Les ressources du Web seront plus aisément accessibles aussi bien par l'homme que par la machine, grâce à la représentation sémantique de leurs contenus.

Le Web sémantique, concrètement, est d'abord une infrastructure pour permettre l'utilisation de connaissances formalisées en plus du contenu informel actuel du Web, même si aucun consensus n'existe sur jusqu'où cette formalisation doit aller. Cette infrastructure doit permettre d'abord de localiser, d'identifier et de transformer des ressources de manière robuste et saine tout en renforçant l'esprit d'ouverture du Web avec sa diversité d'utilisateurs. Elle doit s'appuyer sur un certain niveau de consensus

portant, par exemple, sur les langages de représentation ou sur les ontologies (Chap. 4) utilisés. Elle doit contribuer à assurer, le plus automatiquement possible, l'interopérabilité et les transformations entre les différents formalismes et les différentes ontologies. Elle doit faciliter la mise en œuvre de calculs et de raisonnements complexes tout en offrant des garanties supérieures sur leur validité. Elle doit offrir des mécanismes de protection (droits d'accès, d'utilisation et de reproduction), ainsi que des mécanismes permettant de qualifier les connaissances afin d'augmenter le niveau de confiance des utilisateurs.

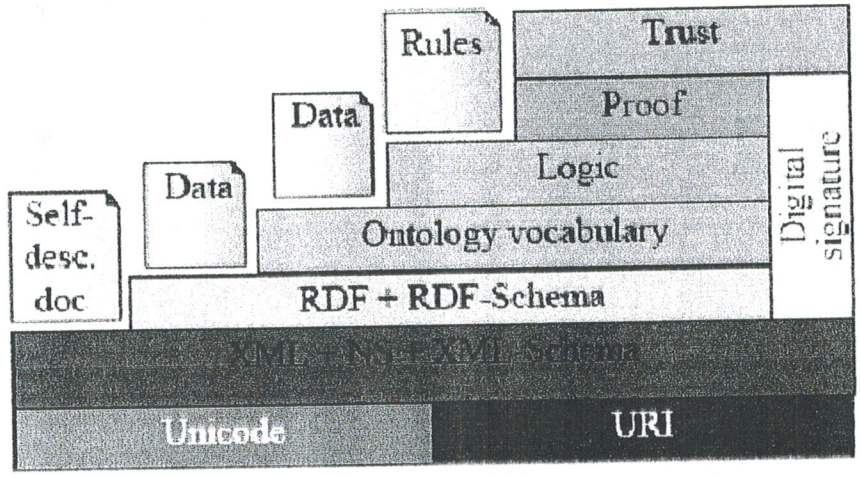


Figure 3.1: les couches du Web Sémantique

XML (W3C, 1998) : fournit une surface syntaxique pour les documents structurés mais ne fournit aucune contrainte sémantique sur le sens de ces documents.
XML Schema (W3C, 2001) : est un langage pour restreindre la structure des documents XML et étendre aussi XML avec des types de données.
RDF (W3C, 2004, c) : est un modèle de données pour les objets (« ressources ») et les relations entre eux, fournissant des sémantiques simples pour ce modèle de données qui peuvent être représentés en XML.
RDF Schema : est un vocabulaire pour décrire les propriétés et les classes des ressources RDF.
DAML-OIL : comme RDF schema, DAML+OIL supporte les notions de classes et de propriétés, mais DAML+OIL vient enrichir l'expressivité de cette notion par la logique de description (LD).

OWL : ajoute plus de vocabulaire pour décrire les propriétés et les classes entre autres, les relations entre les classes, cardinalité, égalité, typage de propriétés plus riche, caractéristiques des propriétés et les hiérarchies des propriétés et des classes.

Mais restreindre le Web sémantique à cette infrastructure serait trop limitatif. Ce sont les applications développées sur celle-ci qui font et feront vivre cette vision et qui seront, d'une certaine manière, la preuve du concept . Bien sûr, de manière duale, le développement des outils, intégrant les standards du Web sémantique, doit permettre de réaliser plus facilement et à moindre coût des applications ou des services développés aujourd'hui de manière souvent ad-hoc.

3.2.2 Les lacunes du web actuel

Le Web contient de vastes bases de données avec plusieurs buts et de multiples sources non homogènes. Ceci prouve qu'il y a nécessité d'améliorer l'accessibilité à cette importante masse d'informations et disposer d'outils plus sophistiqués pour une meilleure recherche et organisation au sein du Web (Stuart et Eric, 2000). Il est clair que le Web actuel doit changer d'orientation d'un Web "présentable" qui se contente de présenter ou d'afficher les informations disponibles, à un Web "intelligent" ou compréhensible. Cette réorientation doit s'effectuer pour ne pas ruiner les moteurs de recherche et le e-business, surtout avec la quantité croissante des informations sur le Web (Claude, 2001). Ceci nous montre que le Web dans son état actuel souffre de plusieurs lacunes, le problème qu'on doit résoudre est : "Comment trouver, de façon précise et rapide, l'information que l'on recherche sur la toile ?" (INRIA, 2001).

La réponse à cette question passe par la nécessité de changer la structure du Web actuel. Cette dernière se contente "d'établir des liens entre des fichiers et de les formater en vue de les afficher"(FING, 2001). En changeant cette structure en une autre, qui permet de relier le sens des informations se trouvant à l'intérieur des fichiers, on va permettre aux machines de mieux exploiter ces informations. Cette vision correspond bien au principe du Web Sémantique.
En plus des points évoqués plus haut et qui ont poussé les chercheurs à vouloir améliorer le Web actuel, une autre raison a fait son apparition durant l'année 2001. Pendant cette année, un nouveau terme a vu le jour et a eu beaucoup de succès, il s'agissait du terme "portail". Par ce mot, on désigne une énorme base de données de ressources documentaires (textes, images, sons, vidéos, etc.) comme Lycos, Yahoo, etc. Ces bases de données ne peuvent pas être exploitées efficacement avec les techniques standard de la recherche d'informations (RI). Ce besoin a renforcé l'utilité du Web Sémantique dans le domaine de la RI (TuanAnh, 2001).

3.2.3 Le web sémantique et le web du futur

Actuellement, les gens partagent leur savoir (informations) sur le Web dans des langages propres à eux, donc inconnus à d'autres gens. Il ne fait pas de différences entre un texte qui doit être vu comme une information commerciale, ou comme étant une information académique. La première source est plutôt destinée à la compréhension humaine, et la deuxième aux machines (traitement automatique). Le *SW* vise à être un pont entre le raisonnement de la machine (logique) et celui des humains (qui se base sur le langage humain)(Berners-Lee et al., 2001). Il sera possible d'exprimer nos besoins ou donner des informations dans des termes que nos ordinateurs peuvent interpréter pour nous et les faire échanger. Non seulement, il sera simple et utile, mais aussi il disposera du raisonnement et de la puissance nécessaire.

Dans la conclusion qui a été faite en juin 2001, *l'atelier Semantic Web Technologies Workshop* a fait part de sa vision future concernant le Web, et tout ce qui a été dit dans cette conclusion coïncide fortement avec les objectifs du *SW*, il a été dit : "Permettre aux utilisateurs d'accéder, de récupérer et de filtrer des informations depuis le Web, correspondant à leurs intérêts et besoins, et répondant à leurs attentes qualitatives. Ceci

exige des méthodes, des modèles et des outils nouveaux et avancés, et des systèmes fournissant des services relatifs à *l'accès*, à la *récupération* et au *filtrage* de contenus basés sur le Web, en particulier le développement de nouveaux systèmes et machines de recherche, en liaison avec les standards existants et émergents. Ces services devraient typiquement être rendus par des agents intelligents". Le rôle de ces agents est "d'automatiser le dialogue entre applications distantes, que ce soit en termes de services ou de contenus." Cela veut dire "analyser les réseaux sémantiques de mots clés et de méta-données afin de repérer les liens entre informations."((Dardailler, 2002), (Crochet Damais, 2001)).

Cela peut être résumé en quatre points :

. Comment et avec quoi structurer le contenu, et comment définir et déclarer sa sémantique ?

. Comment et avec quoi déduire les attributs sémantiques du contenu du Web ?

. Comment faire un filtrage et une analyse intelligente ?

. Interfaces intelligentes et claires qui se basent sur les structures sémantiques.

Pour essayer de résoudre ce genre de problèmes, les chercheurs ont proposé d'ajouter une couche de méta-données (description sémantique) aux informations existantes(Bordage, 2002). C'est à dire "optimiser la structuration et la formalisation des données pour qu'elles puissent être comprises, puis exécutées par les machines" (Dardailler, 2002).

3.2.4 Quelques applications dans le domaine du web sémantique

Le *SW* peut être intégré dans des outils de veille stratégique qui sont issus du milieu de l'Intelligence économique "Competitive Intelligence". Cette dernière a pour but de localiser, de trier, de n'apporter que les informations pertinentes existantes sur Internet, pouvant aider une société à prendre des décisions qui devanceront ses concurrents. Ces outils fonctionnent comme des robots scrutant la toile régulièrement afin d'indexer et de retrouver les informations adéquates. On peut aussi employer le *SW* dans des outils d'analyse ou décisionnels. Comme dans des moteurs de recherche OLAP (OnLine Analytical Processing), qui donnent à l'utilisateur la possibilité d'analyser rapidement une quantité d'informations, les faire interagir afin de prendre la meilleure décision(Benchmark, 2001). Ce qui rend l'utilisation du *SW* encore plus difficile, est le manque de Sémantique dans le Web actuel.

Pour résoudre ce problème, plusieurs solutions existent : "sémantiser" par des moyens logiciels ou manuels. La première solution a le mérite d'accroître le nombre de documents traités et aussi d'accélérer le processus de sémantisation du Web, mais son inconvénient est la complexité du développement des algorithmes. La deuxième solution demandera certainement beaucoup de ressources humaines mais aura l'avantage d'être plus riche. La solution idéale est une *combinaison* des deux possibilités, "une vérification et correction humaine des résultats produits par automatisation logicielle."(Bernard, 2001).

Il existe aussi un autre problème, étant donné que les documents sur Internet sont créés, modifiés ou supprimés à une vitesse incroyable, il sera très difficile de sémantiser le Web sans la coopération des éditeurs on-line. La cause est la non adoption d'un seul standard pour écrire les sources, par exemple les standards émergeant du W3C. Or, il existe tellement de sources qui ne contiennent pas d'informations sémantiques, que faire alors ? Ne plus tenir compte de ces informations ou trouver un autre moyen ?

La première étape consiste à créer une sémantique liée à ces sources dans des documents séparés qui possèdent un lien vers ladite source. La deuxième étape consiste à développer des outils de bas niveau (moteurs de règles pour faire de la déduction), et des outils de haut niveau (moteurs de recherche évolués, agents intelligents)(Bernard, 2001).

3.3 Représentation des connaissances

Le Web contient un nombre ingérable de documents. Pour que l'utilisateur ne se perde pas dans cet immense gisement, il faut impérativement donner une signification au contenu. Donner du sens ou interpréter les ressources du Web implique trois prérequis : " Un codage universel des caractères –Unicode-, une structure standard de documents -XML "eXtensible Markup Language" est bien parti pour répondre à ce besoin -, et donner un sens aux documents. Le troisième point conduit au *SW*"(Van der Vlist, 2000). Le troisième point peut être conceptualisé en reliant au Web (plus exactement en reliant aux données) des méta-données, qui ont pour rôle d'ajouter un sens au contenu du Web. Ces méta-données sont connues sous le nom de RDF "Resource Description Framework". Or, relier toutes les ressources du Web entre elles n'est pas chose simple à faire, surtout quand il y a une diversité dans les formats des documents. La solution consiste à structurer ou à standardiser le contenu en utilisant le langage XML.

Le challenge du SW est de pouvoir fournir un langage exprimant en même temps les données et les règles, afin de fournir un sens au contenu, et que ces règles soient le plus possibles universelles.

Le Web Sémantique et l'XML

XML reste un langage très important pour structurer le contenu de Web malgré le nombre assez réduit des documents XML actuellement disponibles. Or, cette tendance va s'inverser, car d'après une étude de Gartner, "l'XML dépassera tous les autres formats de publication de données, et cela pour tous les contenus multi-supports", car c'est un langage capable d'apporter de la sémantique aux ressources(Nacfaire, 2002).

Parmi les avantages que XML peut apporter au Web en général et au *SW* en particulier :

.Indépendance par rapport aux plate-formes.
. Souplesse et puissance.
. Gestion de plusieurs formats de données.
. Normalisation des données.
. Grâce à la sémantique incluse dans ses balises, on peut effectuer des recherches plus pertinentes.

3.4 Les langages du Web Sémantique

3.4.1 Introduction

Il a été bien reconnu dans la communauté du Web Sémantique que les ontologies jouent un rôle clé dans la livraison du Web Sémantique en facilitant le partage d'information entre les communautés humaines et des agents logiciels. Les DTDs et les schémas XML peuvent être utilisés pour échanger les données entre les parties qui ont convenu les définitions avant l'échange. Cependant, leur manque de sémantique ne permet pas aux personnes de créer de nouveaux vocabulaires XML. Le même terme peut être utilisé avec différents sens dans différents contextes et des termes différents peuvent être utilisés pour des articles ayant le même sens. RDF et RDF Schema ont essayé de résoudre ce problème en permettant d'associer des sémantiques simples aux identificateurs. Avec RDF Schema, une personne peut définir des classes qui ayant plusieurs sous-classes et super classes, et peut aussi définir des propriétés pouvant avoir des sous propriétés, de domaines ou d'intervalle.
Dans ce sens RDF Schema est un langage d'ontologie pour le Web primitif. Pour atteindre l'interopérabilité entre les divers schémas développés de façon autonome, des sémantiques plus riches sont nécessaires.

OWL,DAML-OIL et RDF sont devenus des recommandations du W3C. RDF est utilisé pour représenter l'information et pour échanger la connaissance sur le Web. OWL est utilisé pour publier et partager les ensembles de termes (appelées les ontologies), en supportant les recherches Web avancées, la gestion de la connaissance et des agents logiciels (W3C 2004).

3.4.2 RDF (Resource Description Framework)

RDF

RDF est une création pour le traitement des métadonnées ; il fournit l'interopérabilité entre les applications qui échangent de l'information non compréhensible par les machines sur le Web. RDF augmente la facilité de traitement automatique des ressources Web. RDF peut être utilisé dans une variété de champs d'application ; par exemple : dans la découverte de ressources pour fournir une meilleure efficacité aux moteurs de recherche, dans le catalogage pour décrire le contenu et les relations entre les contenus disponibles sur un site Web particulier, sur une page, ou sur une bibliothèque numérique, dans l'évaluation du contenu, en décrivant des ensembles de pages qui représente un simple et unique "document", pour décrire les droits sur la propriété intellectuelle des pages Web, et pour indiquer les préférences de confidentialité d'un utilisateur aussi bien que les politiques de confidentialité d'un site Web. RDF avec les signatures numériques sera la clé pour construire un "Web de confiance" pour le commerce électronique, les collaborations, et d'autres applications.

RDFS (pour RDF Schéma (Brickley et Guha, 2000))

RDFS a pour but d'étendre le langage en décrivant plus précisément les ressources utilisées pour étiqueter les graphes. Pour cela, il fournit un mécanisme

permettant de spécifier les classes dont les ressources seront des instances, comme les propriétés. RDFS s'écrit toujours à l'aide de triplets RDF, en définissant la sémantique de nouveaux mots-clés
comme :

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

Ce besoin de spécifier davantage les classes est à l'origine du langage dédié aux définitions de classes : OWL.

3.4.3 Cartes topiques

Les cartes topiques ("Topic maps" (Biezunski et al., 1999)) sont un standard ISO issu de HyTime dont le but était d'annoter les documents multimédia. Issu de SGML, il s'est vu récemment attribuer une syntaxe XML (XTM (Pepper et Moore, 2001)). Par ailleurs, un groupe de l'ISO s'occupe de définir un langage de requêtes pour les cartes topiques (TMQL).

Les cartes topiques sont bâties autour de quatre notions primitives (nous faisons ici abstraction des sujets) :

- Les "topics" que l'on peut comprendre comme des individus des langages de représentation de connaissances ;
- Les noms donnés aux topics : l'une des originalités des cartes topiques est la séparation des concepts et de leurs noms. Cela permet d'avoir plusieurs noms pour le même concept (et donc d'avoir des cartes topiques multilingues) et des noms partagés par plusieurs concepts ;
- Les occurrences sont des "proxis" d'entités externes qui peuvent ainsi être indexés par les topics (ou les entités littérales lorsque celles-ci sont représentables) ;
- Les portées, qui sont parfois vues comme une quatrième dimension, permettent de spécifier le contexte dans lequel une relation est valide.

Par exemple, le topic de vol est instancié par myFlight, il a pour nom « vol pour Boston » dont la portée est celle de mes discussions au déjeuner avec les collègues et « flight AF322 » lors de discussions avec l'immigration américaine.

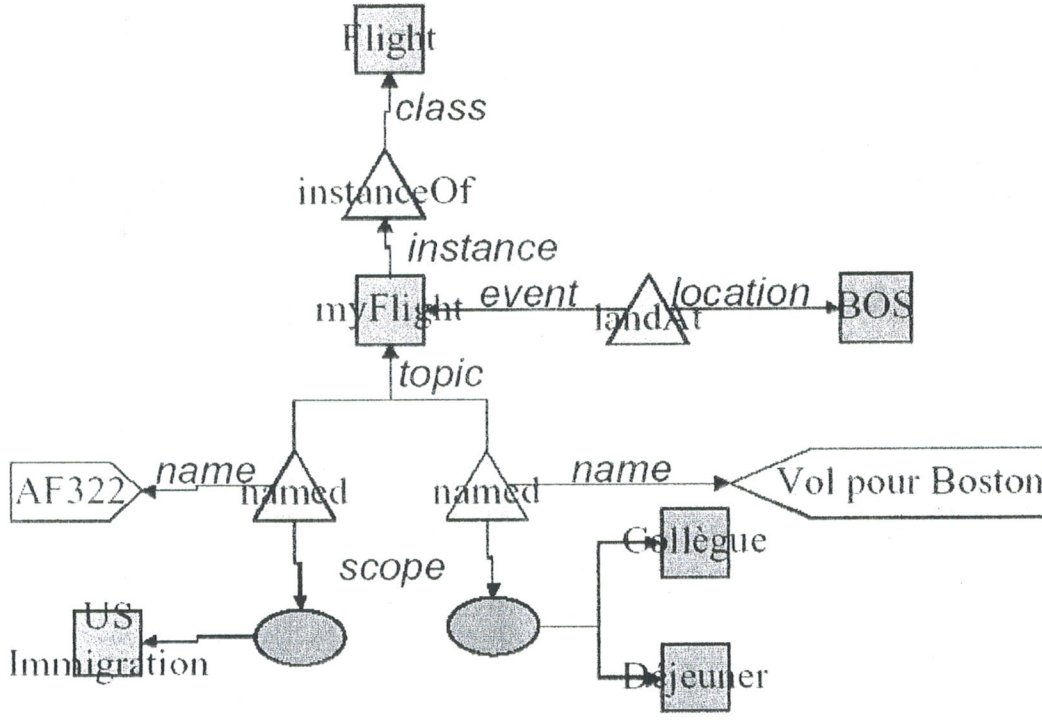


Figure 3.4: Une carte topique.

Si ces quatre dimensions sont spécifiées de manière indépendante, elles sont en réalité interdépendantes : les topics et les noms ont des portées, les topics ont des noms, les portées sont des ensembles de topics…).

Dans la nouvelle syntaxe des cartes topiques, celles-ci sont représentées par des graphes comprenant 3 types de nœuds (topic, association, portée) et un certain nombre de types d'arcs (instance, occurrence, portée, nom). Les relations sont représentées par des nœuds dont les arcs sortants portent des étiquettes identifiant leur rôle. Par ailleurs, différentes interprétations sont données à ces primitives suivant les étiquettes placées sur les arcs et les nœuds. Autant dire que les cartes topiques ne disposent pas d'une sémantique claire et que, au contraire, ses concepteurs ont tendance à considérer que la richesse du langage tient dans les interprétations multiples que l'on peut en faire.

Ceci ne le rend pas un candidat très souhaitable pour le Web sémantique malgré ses qualités indéniables. Il existe cependant des outils permettant de tirer parti de manière utile des cartes topiques qui sont utilisées dans un certain nombre d'applications.

3.4.4 DAML-OIL

Pour ajouter une plus grande expressivité au langage RDF(S), DAML+OIL utilise les primitives de la logique de description. DAML+OIL est issu de l'union de deux langages: DAML-ONT (DARPA Agent Markup Langage) développé par les EU et OIL (Ontology Inference Layer) qui provient d'Europe. À l'origine DAML-ONT était un programme du gouvernement américain en vue de développer un langage pour le Web

sémantique. OIL provient du projet européen « On-To-Knowledge », ce langage ajoute les primitives de langage de frame à RDF(S). DAML+OIL fut soumis en décembre 2000 ensuite il y eut une version en mars 2001.

Tout comme RDF schema, DAML+OIL supporte les notions de classes et de propriétés, mais DAML+OIL vient enrichir l'expressivité de cette notion par la logique de description (LD). Dans DAML+Oil, tout comme en logique de description, les classes peuvent être des noms (un URI) ou des expressions. Des constructeurs sont utilisés pour créer des expressions de classe.

Par exemple pour illustrer le constructeur d'intersection de Humain et de Mâle, DAML+OIL utilise la syntaxe RDF suivante :

```
<daml :Class>
   <daml :intersectionOf rdf :parseType= »daml :collection »>
     <daml:Class rdf:about="#Humain"/>
     <daml:Class rdf:about="#Mâle"/>
   </daml :intersectionOf>
</daml:Class>
```

L'exemple du constructeur « intersectionOf » indique qu'une nouvelle classe est créée à partir des classes Humain et Mâle pour former la classe des mâles qui sont des humains. Le constructeur « unionOf » représente le ou logique ainsi une nouvelle classe qui est soit les comptables ou les pompiers qui est issue des classes Comptable et Pompier. Le constructeur «complementOf » est la négation, ainsi la classe de tout ce qui n'est pas un mâle est faite à partir de la classe Mâle.

Le constructeur « oneOf » permet de définir une classe en énumérant ses éléments. Dans l'exemple la classe des pays d'Amérique du Nord pourrait être la liste Canada,E.U et Mexique. Le constructeur « toClass » indique que toutes instances rencontrant les valeurs associées à la propriété appartiennent à la classe. Dans l'exemple la classe de tous les enfants de pompier forme la classe. Tandis que « hasClass » spécifie qu'au moins une seule valeur de la propriété doit être rencontrée pour définir la classe. Dans l'exemple il suffit qu'au moins un seul enfant de comptable soit présent pour créer la classe. Le constructeur « hasValue » est une combinaison de « hasClass » et « oneOf ». Ainsi dans l'exemple la classe est composée d'au moins un membre qui habite Montréal.

Les constructeurs « minCadinalityQ », « maxCardilanityQ » et « cardinalityQ » imposent des restrictions de cardinalité à respecter.. Ils représentent la classe de toutes les instances qui sont reliées par une propriété P à au moins n différentes ressources de type C . Le constructeur « minCardinalityQ » dans l'exemple décrit la classe de ceux qui ont au minimum 2 enfants blonds. Tandis que l'exemple du constructeur « maxCardinalityQ » décrit la classe de ceux qui ont au plus un enfant mâle. Finalement le dernier constructeur « cardinalityQ » indique la cardinalité à respecter,ainsi dans l'exemple la classe est définie par ceux qui ont une auto japonaise. Ces constructeurs peuvent s'imbriquer pour former des classes plus complexes.

Axiomes	Syntaxe de Logique Desc.	Exemple
subClassOf	$C_1 \sqsubseteq C_2$	Felin ⊑ Mammifère
sameClassAs	$C_1 = C_2$	Homme ≐ Humain⊓ Mâle
subPropertyAs	$R \sqsubseteq P$	Crawler⊑ Nager
samePropertyAs	$R = P$	ConstitueDe ≡ ComposéDe
sameIndividualAs	$\{x1\} = \{x2\}$	{Rambo} = {Stallone}
disjointWith	$C_1 \sqsubseteq \neg C_2$	Mâle⊑ ¬Femelle
differentIndividualFrom	$\{x1\} \sqsubseteq \neg \{x2\}$	{Jacques}:c⊑ ¬{Paul}
inverseOf	$R = P^-$	Lancer ≡ Attraper-
transitiveProperty	$P^+ \sqsubseteq P$	HabitantDe+⊑HabitantDe
uniqueProperty	$\top \sqsubseteq \leq 1 P$	T⊑≤1aMère
unambiguousProperty	$\top \sqsubseteq \leq 1 P^-$	T⊑≤1estMèrede-

Tableau 4.4: Axiomes de DAML+OIL, source (Horrocks, 2001).

Une ontologie en DAML+OIL est composé d'axiomes qui alléguent des caractéristiques au sujet des classes et des propriétés (voir tableau 4.4). Les axiomes peuvent exprimer la subsomption (c-à-d de montrer une hiérarchie entre les classes ou les propriétés) ou d'équivalence par les axiomes de « subClassOf », « sameClassOf », « subPropertyOf » et « samepropertyAs ».

Une des particularités de DAML+OIL c'est que les axiomes de « subClassOf » et « sameClasseOf » peuvent s'appliquer à des expressions de classe (class expression) arbitraire. Les restrictions sur les propriétés peuvent ainsi créer des expressions de classe. Ensuite une classe peut devenir une sous-classe de cette expression de classe arbitraire. Par exemple nous pouvons créer une classe par la restriction suivante :

```
<daml :Restriction>
    <daml :OnProperty ref :ressource= « #se_nourrit_de »/>
    <daml :toClass rdf :ressource= « #vegetal »/>
</daml :Restriction>
```

Ce qui signifie que nous venons de créer une classe anonyme, c'est à dire la classe des objets qui satisfont la restriction, dans ce cas ci, ce sont les instances qui ont la propriété «se_nourrit_de » appliquée à la classe « #vegetal ». Ainsi par exemple nous pourrions définir la classe girafe comme une sous-classe de cette classe anonyme ou autrement dit comme faisant partie de la classe de ceux qui se nourrissent de végétal:

```
<daml :Class ref :ID = « Girafe »>
      <daml:SubClassOf>
         <daml :Restriction>
           <daml :OnProperty ref :ressource= « #se_nourrit_de »/>
           <daml :toClass rdf :ressource= « #vegetal »/>
         </daml :Restriction>
      </daml:SubClassOf>
  </daml:Class>
```

En procédant ainsi on demande que chaque instance de la classe Girafe satisfasse cette restriction, autrement dit chaque girafe doit respecter la restriction de se nourrir uniquement de végétal. Cela permet à la classe d'hériter des caractéristiques de la classe anonyme. Cette façon de proceder permet plus de flexibilité pour définir les particularités d'une classe.

La fidélité de DAML+OIL à la logique de description permet de supporter des inférences au niveau de l'ontologie grâce au mécanisme de subsomption. Le raisonnement peut être utile entre autres :

- Pour valider la relation entre les classes lors du design de l'ontologie. Ceci est d'autant plus important pour les ontologies comportant plusieurs auteurs.
- Pour faciliter l'intégration d'information entre ontologies et éviter entre autres des inconsistances au niveau des hiérarchies entre les classes.
- Pour son déploiement : vérifier si un individu est une instance d'une classe, déterminer si un ensemble de faits peut être consistant avec l'ontologie.
- Pour permettre aux agents intelligents de mieux exploiter le contenu de l'ontologie.

Voici quelques exemples de raisonnement qui peuvent être permis avec DAML+OIL.

La transitivité permet de capturer la causalité, les conséquences etc. ex :

$$\text{Vitesse} \subseteq \exists \text{cause.Accident plus Accident} \subseteq \exists \text{cause.Blessure}$$
$$\Rightarrow \text{Vitesse} \subseteq \exists \text{cause.Blessure}$$

Cet exemple indique que la vitesse peut être une cause d'accident et que les accidents peuvent être des causes de blessure par le fait même on peut déduire que la vitesse peut être une cause de blessure.
L'inversion des rôles permet de saisir les relations de causes par exemple.

$$\text{Blessure} \wedge \exists \text{causéPar.Accident} \subseteq \text{Traumatisme} \Rightarrow \text{Vitesse} \subseteq \exists \text{cause.Traumatisme}$$

Comme la classe des blessures causée par accident sont inclus dans la classes des traumatisme alors DAML+OIL peut déterminer que la vitesse peut être une cause de traumatisme.
Plusieurs axiomes d'égalité ou d'inclusion permettent d'effectuer des raisonnements d'égalité par exemple.

$$\text{Joueur-deHockey} = \text{Joueur} \wedge \text{Pratique\_leHockey plus}$$
$$\text{Joueur-deHockey} \subseteq \exists \text{ utilisent\_desPatins}$$
$$\text{Joueur} \wedge \text{Pratique\_leHockey} \subseteq \text{Patineur.}$$

À partir des deux premiers axiomes qui indiquent qu'un joueur de hockey est la même chose qu'un joueur qui pratique le hockey et qu'un joueur de hockey est une sous-

classe de ceux qui utilisent des patins. DAML+OIL peut considérer qu'un Joueur qui pratique le hockey est membre de la classe des patineurs.

3.4.5 OWL

RDF, langage dédié à l'expression d'assertions sur les relations entre objets, s'est heurté à la nécessité de définir les propriétés des classes dont ces objets sont instances. Cependant, l'extension à RDFS ne fournit que des mécanismes primitifs pour spécifier ces classes. Le langage OWL (Dean et Schreiber 2003), quant à lui, est dédié aux définitions de classes et de types de propriétés, et donc à la définition d'ontologies. Inspiré des logiques de descriptions (et successeur de DAML+OIL (van Harmelen et al., 2001)), il fournit un grand nombre de constructeurs permettant d'exprimer de façon très fine les propriétés des classes définies. La rançon de cette expressivité est l'indécidabilité du langage obtenu en considérant l'ensemble de ces constructeurs. C'est pour cela que OWL a été fractionné en trois langages distincts :

- OWL LITE ne contient qu'un sous-ensemble réduit des constructeurs disponibles, mais son utilisation assure que la comparaison de types pourra être calculée (un problème de NP, donc « simple » en représentation de connaissances) ;
- OWL DL contient l'ensemble des constructeurs, mais avec des contraintes particulières sur leur utilisation qui assurent la décidabilité de la comparaison de types. Par contre, la grande complexité de ce langage (un de ses fragments est P-SPACE-complet) semble rendre nécessaire une approche heuristique ;
- OWL FULL, sans aucune contrainte, pour lequel le problème de comparaison de types est vraisemblablement indécidable.

La syntaxe d'un document OWL est donnée par celle des différents constructeurs utilisés dans ce document. Elle est le plus souvent donnée sous la forme de triplets RDF. La sémantique de chaque constructeur est donnée en théorie des modèles (Patel-Schneider et al., 2003). Elle est directement issue des logiques de descriptions. La sémantique associée aux mots-clés de OWL est plus précise que celle associée au document RDF représentant une ontologie OWL (elle permet plus de déductions).

Nous donnons ici l'ensemble des constructeurs utilisés dans OWL, dans une syntaxe simplifiée (les mots-clés réservés de OWL, habituellement préfixés de OWL : sont soulignés), ainsi que leur « sémantique intuitive ». Les constructeurs de OWL LITE sont cités les premiers.

OWL LITE

- Reprend tous les constructeurs de RDF (c'est-à-dire fournit des mécanismes permettant de définir un individu comme instance d'une classe, et de mettre des individus en relation),
- Utilise les mots-clés de RDFS (rdfs:subClassOf, rdfs:Property, rdfs:subPropertyOf, rdfs:range, rdfs:domain), avec la même sémantique,
- Permet de définir une nouvelle classe (owl:Class) comme étant plus spécifique ou équivalente à une intersection d'autres classes,
- owl:sameIndividualAs et owl:differentIndividualFrom permettent d'affirmer que deux individus sont égaux ou différents,
- Des mots-clés permettent d'exprimer les caractéristiques des propriétés :

`owl:inverseOf` sert à affirmer qu'une propriété p est l'inverse de p' (dans ce cas, le triplet $<s\ p\ o>$ a pour conséquence $<o\ p'\ s>$) ; d'autres caractéristiques sont par

exemple la transitivité (`owl:TransitiveProperty`), la symétrie (`owl:SymmetricProperty`),
- `owl:allValuesFrom` associe une classe *C* à une propriété *P*. Ceci définit la classe des objets *x* tels que si <*x P y*> est une relation, alors la classe de *y* est *C* (quantification universelle de rôle en logique de descriptions). `owl:someValuesFrom` encode la quantification existentielle de rôle,
- `owl:minCardinality` (resp. `owl:maxCardinality`) associe une classe *C*, une propriété *P*, et un nombre entier *n*. Ceci définit la classe des objets *x* tels qu'il existe au moins (resp. au plus) *n* instances différentes *y* de *C* avec <*x P y*>.
Pour des raisons d'efficacité algorithmique, OWL LITE ne permet d'utiliser que des entiers égaux à 0 ou 1. Cette restriction est levée dans OWL DL.

OWL DL

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

OWL FULL

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

Nous n'avons pas cité ici certains constructeurs, qui peuvent être trivialement implémentés grâce à ceux que nous avons évoqués (par exemple owl:sameClassAs, servant à affirmer que deux classes sont identiques, peut être écrit grâce à deux rdfs:subClassOf). Il serait intéressant d'identifier quels sont les constructeurs primitifs nécessaires pour ces langages, et ceux qui ne sont que des macros.
Des moteurs d'inférence ont déjà été implémentés pour des sous-ensembles significatifs de OWL DL (dans le cadre des logiques de descriptions) et peuvent être utilisés dans divers outils (OilEd, Protégé…).

Conclusion

Nous avons présenté ce que tout le monde définit comme étant le Web du futur, qui est nommé le *Web Sémantique*. Nous avons passé en revue comment les chercheurs voient le "Web intelligent" du futur, et quel pourrait être son impact. Nous pensons que ce type de Web aura un avenir très brillant vu les avantages qu'il pourrait apporter à la communauté scientifique et aux industriels, bien sur avec une architecture réseau dynamique et flexible, et d'éviter l'aspect Client/Serveur. Nous présentons au chapitre 5 les réseaux Pair-à-pair qui nous prévoyons pour le web sémantique, pour remédier à ce problème.

Références

BENCHMARK Group. "Business intelligence : des concepts mais peu d'usage". *Management & budget informatique no36*, juin 2001.

BERNARD .J. "Le Web Sémantique n'attend plus que vous !". *http ://xmlfr.org/documentations/articles/010601-0001*, juin 2001.

BIEZUNSKI .M, BRYAN .M et NEWCOMB .S. "ISO/IEC 13250:2000 Topic Maps: Information Technology — Document Description and Markup Languages." *http://www.y12.doe.gov/sgml/sc34/document/0129.pdf*, 1999

BORDAGE .F. "Le Web Sémantique affine la recherche sur le Net". *www.01net.com/rdn ?oid=181230&rub=3349, Internet Professionnel*, avril 2002.

BRICKLEY .D et GUHA .R.V. "Resource Description Framework (RDF) Schema Specification 1.0". *W3C Recommentdation*, Mar. 2000.

CLAUDE .C. "Semantic Web : contenu avec référenceS", *http ://xmlfr.org/actualites/decid/010402-0001*, mai 2001.

CROCHET DAMAIS .A. "Mondeca met le Web sémantique au service de l'organisation des connaissances". *JDNet*, novembre 2001.

DHENIN .C. "Profium ou la gestion du contenu selon RDF". *JDNet, http ://solutions.journaldunet.com/0108/010831 profium.shtml*, août 2001.

DARDAILLER .D. "Du Web sémantique aux Web services, nous automatisons le travail des applications". *JDNetsolutions, http ://solutions.journaldunet.com/printer/020208 it w3c.shtml*, février 2002.

DEAN .M et SCHREIBER .G.. "OWL web Ontology Language: Reference". *W3C Working Draft. http://www.w3.org/TR/owl-ref/* , 2003.

VAN DER VLIST .E. "Le Web Sémantique verra le jour avec ou sans RDF". *01net., www.01net.com/rdn ?oid=151859&rub=3556*, juillet 2001.

VAN DER VLIST .E. "RDF vous guide sur XMLfr". *http://xmlfr.org/actualites/xmlfr/000919-0001*, septembre 2000.

Fondation Internet Nouvelle Génération (FING)."Le Web sémantique : trop malin pour les humains ?". *www.fing.org/index.php ?num=2115,3,119,4*, octobre 2001.

HORROCKS I. DAML+OIL and Description Logic Reasoning. *Talk given at HP Labs, Bristol*, October, 2001;

INRIA. "Le Web Sémantique : du sens sur la Toile". *www.inria.fr/actualites/inedit/inedit29 rega.fr.html*, avril 2001.

EUZENAT .J. "Contribution aux réflexions de l'action spécifique Web Sémantique, *INRIA Rhône-Alpes, Action Exmo, www.inrialpes.fr/exmo/cooperation/asws/exmo-sw.html*, Janvier 2002.

KLYNE .G et CARROLL .J. "Resource Description Framework (RDF): Concepts and Abstract Syntax". *W3C Working Draft, 2003 http://www.w3.org/TR/rdf-concepts/* , 2003.

LASSILA .O et SWICK .R. "Resource Description Framework (RDF) Model and syntax specification". *Recommendation, W3C. http://www.w3.org/TR/REC-rdf-syntax* , 1999.

PATEL-SCHNEIDER .P, HAYES .P et HORROCKS Ian. "OWL web Ontology Language: Abstract Syntax and Semantics". *W3C Working Draft. http://www.w3.org/TR/owlsemantics/* , 2003

PEPPER .S et MOORE .G. "XML Topic Maps (XTM) 1.0. TopicMaps.Org Specification". *http://www.topicmaps.org/xtm/1.0/* , 2001.

NACFAIRE .R. "Xyleme se lance sur le marché de la recherche et de l'intégration de contenus XML". *Digital Business Globe, www.xyleme.com/fr/presse/doc1.pdf*, mars 2002.

STUART .W et ERIC .M. "An Introduction to Dublin Core",
www.xml.com/pub/a/2000/10/25/dublincore/, October 2000.

T. P. Martin. " Searching and Smushing on the Semantic Web—Challenges for Soft Computing ".
University of Bristol, 2001 BISC International Workshop on Fuzzy Logic and the Internet, 2001.

TIM Berners-Lee, JAMES Hendler, et ORA Lassila. "The Semantic Web".
www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809 EC588EF21& catID=2 , May 2001.

TUANANH Ta. "Web Sémantique et portails —un état de l'art".
www.infres.enst.fr/people/saglio/etudes/e-parcours/portails/SemWebintro.pdf, 2001.

VAN HARMELEN .F, PATEL-SCHNEIDER .P et HORROCKS .I. "Reference description of the DAML+OIL ontology markup language". *W3C.*
http://www.daml.org/2001/03/reference.html , 2001

Chapitre 4

Les ontologies

4.1 Introduction et définitions

4.1.1 Les ontologies en IC

Les ontologies sont apparues en Ingénierie des connaissances et plus largement en Intelligence artificielle, avec l'idée de construire mieux et plus rapidement des SBC en réutilisant le plus possible des composants génériques, que ce soit au niveau du raisonnement ou des connaissances du domaine. Ce qui nous a amené incidemment à une définition simple qui prend déjà acte d'une décision personnelle quant au choix des objets :

Définition d'ontologie: Ensemble des objets reconnus comme existant dans le domaine. Construire une ontologie c'est aussi décider de la manière d'être et d'exister des objets (Gruber, 2003).

Pour avancer sur la question de ce qu'est une ontologie, il nous semble indispensable de rappeler que les travaux sur les ontologies sont développés dans un contexte informatique
- que ce soit celui de l'ingénierie des connaissances, de l'Intelligence artificielle ou des sciences de gestion et de leurs systèmes d'information ou plus récemment le contexte du Web sémantique - où le but final est de spécifier un artefact informatique.

Ce contexte est important pour comprendre les buts poursuivis par les concepteurs d'ontologie et sur les contraintes qui se posent à eux et seront développées au long de ce chapitre. En particulier, la question de la conceptualisation devient centrale dans le but de construire un artefact puisqu'on a besoin, dans ce contexte, de définir et spécifier les concepts à prendre en compte. Cette recherche des définitions au moment de l'élaboration de l'ontologie nous situe au niveau du sens, au niveau de l'intention.

4.2 Les constituants d'une ontologie

4.2.1. Connaissances et domaines de connaissance

Comme précisé précédemment, une ontologie ne peut être construite que dans le cadre d'un domaine précis de la connaissance, ne serait-ce que parce que beaucoup de termes n'ont pas le même sens d'un domaine à l'autre, et qu'une sémantique non ambiguë doit être intégrée à l'ontologie. Un domaine de connaissances est donc constitué par les objets du domaine et par un contexte d'usage de ces objets (Bachimont, 1999). Délimiter rigoureusement un domaine de connaissances peut par contre se révéler ardu, à cause de la nature holistique de la connaissance. Certaines connaissances, qui peuvent constituer en elles-mêmes un domaine, sont utilisées dans tous les autres domaines.

C'est le cas des notions générales liées à la causalité, au temps, à l'espace, etc. De plus, les connaissances humaines se déploient suivant plusieurs dimensions : des connaissances peuvent être développées non seulement sur la réalité mais également sur un domaine de connaissance. On parle alors de méta connaissances, ou connaissances sur les connaissances. Savoir borner le domaine des connaissances à représenter demande donc une délimitation extrêmement précise de l'objectif opérationnel de l'ontologie.

D'autre part, dans le domaine de l'Ingénierie des Connaissances, le terme de connaissance a un sens forcément restreint : ne sont considérées que les connaissances (au sens large) susceptibles d'être formalisées, c'est-à-dire les connaissances peu ou prou techniques (Bachimont, 2000) : « For knowledge-based systems, what exists is exactly that which can be represented » (Gruber, 1993). En outre, la manipulation automatique de connaissances ne parait pertinente que dans le cas des connaissances objectives, c'est-à-dire dont le sens est l'objet d'un consensus large. Plus précisément, on peut considérer qu'il n'y a connaissance que quand l'information présente dans la machine prend un sens pour l'utilisateur, c'est-à-dire qu'il peut établir un lien entre cette information et celles qu'il possède déjà, et ce sens doit être le même pour tous les utilisateurs (Charlet, 2001). Cette notion de lien sémantique explique d'ailleurs le développement des modèles de représentation de type réseaux sémantiques.

Enfin, les connaissances manipulées dans les SBC se doivent d'être des informations «actives», c'est-à-dire susceptibles d'influencer le déroulement de processus, de produire de nouvelles informations ou de permettre de prendre des décisions (Kayser, 1997). Seul ce type de connaissances « productives » parait offrir un intérêt pour l'automatisation. En conclusion, la construction d'une ontologie doit se faire à partir d'un champ de connaissances bien délimité par un objectif opérationnel clair, et portant sur des connaissances objectives dont la sémantique puisse être exprimée rigoureusement et formellement.

Types d'ontologies

Partant de ce que nous avons dit auparavant, plusieurs types d'ontologies peuvent être distinguées en fonction des différents objectifs opérationnels recensés. Comme il a été dit plus haut, les ontologies vont permettre de spécifier les connaissances d'un domaine, de façon aussi indépendante que possible du type de manipulation qui vont être opérées sur ces connaissances. Ces ontologies sont appelées ontologies de domaine, puisqu'elles sont construites sur un domaine particulier de la connaissance. De nombreuses ontologies de domaine existent déjà, telles que MENELAS dans le domaine médical (Zweigenbaum, 1999), ENGMATH pour les mathématiques (Gruber & Olsen, 1994), TOVE dans le domaine de la gestion des entreprises (Gruninger & Fox, 1994), etc.

Une distinction est établie entre les ontologies de domaine portant sur des concepts renvoyant à des objets matériels ou à des concepts d'assez bas niveau (c'est-à-dire n'offrant que des possibilités limitées de raffinement) et les ontologies portant sur des concepts de haut niveau (upper-ontologies). Ces dernières décrivent des notions générales comme les notions d'objet, de propriété, d'état, de valeur, de moment, d'événement, d'action, de cause et d'effet (Sowa, 2000). Parmi les ontologies de haut niveau, on trouve des ontologies qui vont décrire les notions utilisées dans toutes les ontologies pour spécifier les connaissances, telles que les sortes, les substances, les concepts, les relations, etc. Ces ontologies de représentation sont en fait indépendantes des différents domaines de connaissance, puisqu'elles décrivent des primitives cognitives communes aux différents domaines (Guarino et al, 1994).

On considère également que les processus de raisonnement appliqués aux connaissances forment eux-mêmes un domaine de connaissance, que l'on peut

représenter à l'aide d'une ontologie. En particulier, des ontologies de raisonnement ont été développées afin de représenter les connaissances génériques mises en oeuvre lors de la résolution automatique de problèmes. Par exemple, B. CHANDRASEKARAN propose, dans (Chandrasekaran et al, 1998a), une ontologie portant sur la résolution de problème par la décomposition en tâches et méthodes, ontologie indépendante des domaines d'application potentiels ; seules sont décrites les connaissances portant sur la façon d'utiliser d'autres connaissances, elles-mêmes non précisées. B. CHANDRASEKARAN considère d'ailleurs que, dans un système à base de connaissances utilisant des mécanismes inférentiels (c'est-à-dire un système non exclusivement destiné à la consultation de connaissances stockées), deux ontologies distinctes spécifiant les connaissances à manipuler et les connaissances de raisonnement doivent être intégrées.

Une présentation plus fine des différents types d'ontologie présentés dans la figure 2 est donnée dans (Mizoguchi & Ikeda, 1997). Mais au sein d'une même ontologie, différents types de connaissances cohabitent également. En effet, un domaine de connaissance intègre sa propre terminologie, des faits connus, des règles intangibles, des heuristiques, etc. Les ontologistes s'accordent cependant pour considérer que les primitives cognitives de base d'une ontologie sont les concepts et les relations entre ces concepts. Ces notions nécessitent cependant d'être explicitées.

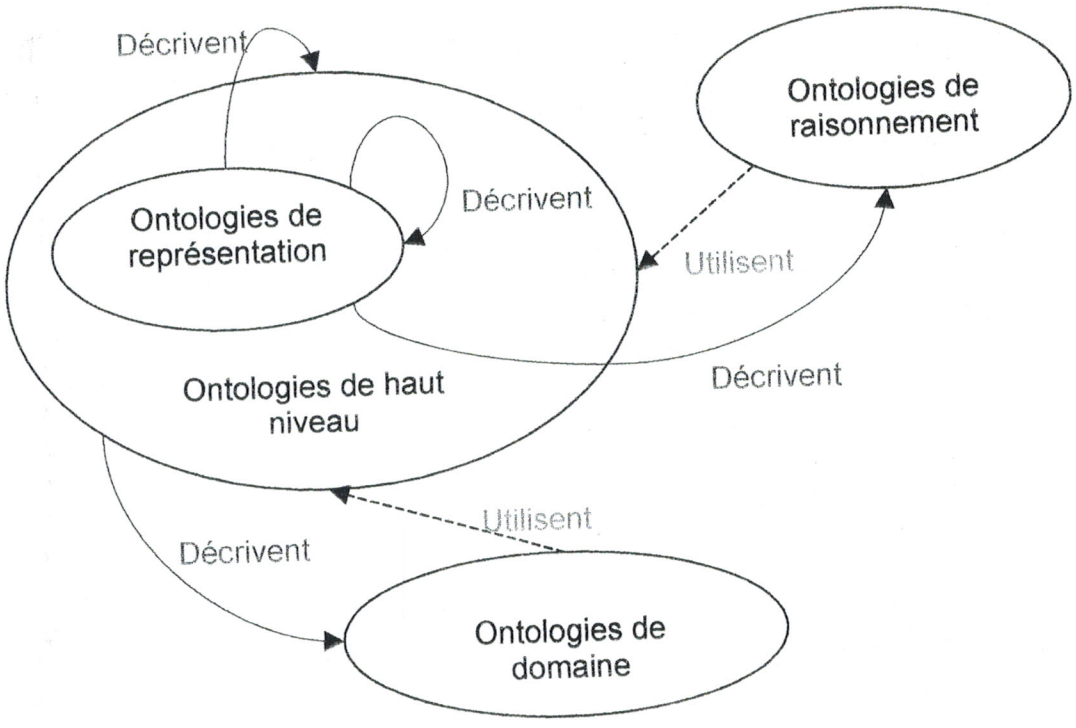


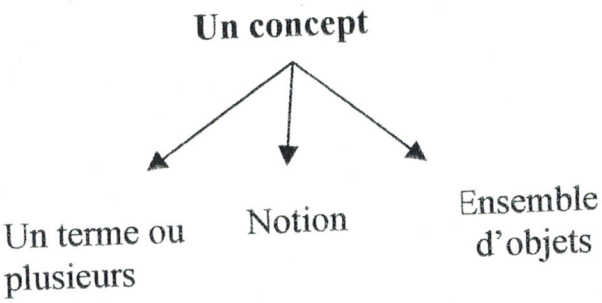
Fig. 4.2 - Les différentes ontologies

4.2.2. Les concepts et les relations

4.2.2.1. Les concepts

Un concept peut représenter un objet matériel, notion, une idée. (Uschold, 1996).

Il peut être divisé en trois parties :



Notion : appelée également intention du concept contient la sémantique du concept, exprimé en termes de propriétés et d'attributs, de règles et de contraintes.

Ensemble d'objets : également appelés extension du concept regroupe les objets manipulés à travers le concept (ces objets sont appelés instances du concept)

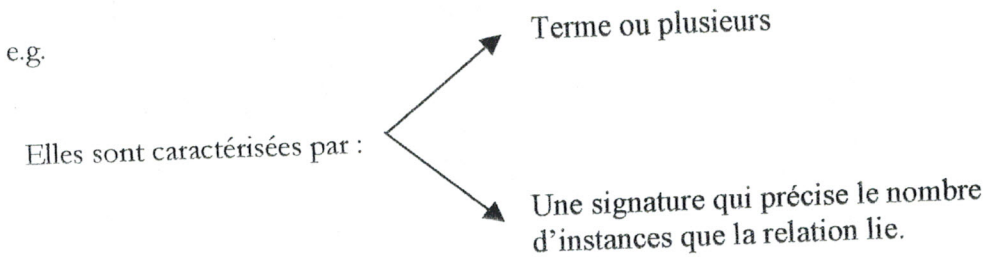
Exemple : le terme « Table »
A comme notion : un objet de type « meuble » qui a un plateau et des pieds…
Ensemble d'objets de ce type : table scolaire, table de cuisine, table de salon…

4.2.2.2. Les relations

Si certains liens conceptuels existant entre les concepts peuvent s'exprimer à l'aide de propriétés portées par les concepts, d'autres doivent être représentés à l'aide de relations autonomes. Une relation permet de lier des instances de concepts, ou des concepts génériques. Elles sont caractérisées par un terme (voire plusieurs) et une signature qui précise le nombre d'instances de concepts que la relation lie, leurs types et l'ordre des concepts, c'est–à–dire la façon dont la relation doit être lue. Par exemple, la relation « écrit » lie une instance du concept « personne » et une instance du concept « texte », dans cet ordre.

Trancher entre la représentation d'une notion sous forme d'un concept ou d'une relation peut parfois s'avérer nécessaire. Par exemple, l'écriture d'un texte peut être vue comme un concept et on exprimera le fait qu'une personne écrit en disant qu'elle est en relation de type -
« mener une action », l'action étant elle-même l'écriture qui sera en relation avec un « texte ».

Le choix dépend alors essentiellement de l'objectif opérationnel de l'ontologie et du contexte d'utilisation de la notion. De même, il convient de choisir entre l'utilisation de relations ou d'attributs pour représenter les liens entre concepts. Tout comme les concepts, les relations peuvent être spécifiées par des propriétés dont une liste, non exhaustive, est donnée ci-après. Les relations sont organisées de manière hiérarchisée à l'aide de la propriété de subsomption décrite précédemment. Les remarques faites au sujet de l'organisation des propriétés s'appliquent également dans ce cas.



Par exemple : Ali écrit une lettre.
On a 3 concepts : C1 : ALI,
C2 : ECRIRE
C3 : LETRRE
et 2 relations : R1 : AGENT,
R2 : THEME
Cette relation permet de lier des instances des concepts ou des concepts génériques.
AGENT (C1, C2) et THEME (C2, C3)

4.2.3 Les connaissances inférentielles

Décrire les connaissances en terme de concepts, de relations entre ces concepts et de propriétés sur ces concepts et relations ne suffit généralement pas pour atteindre l'objectif opérationnel d'un Système à Base de Connaissances.
« De façon un peu caricaturale, on pourrait dire que ce ne sont pas les connaissances en elles-mêmes qui sont intéressantes [..]. En fait, ce qui doit être le centre d'intérêt pour le modélisateur, c'est davantage leur mise en oeuvre, leur concrétisation dans une action pour atteindre un but car c'est bien l'activité qu'on cherche à assister et non les connaissances en elles-mêmes » (Teuliet & Girard, 2001). Il s'agit également de tirer au maximum parti de ce qui fait la spécificité du support informatique par rapport au support écrit traditionnel, c'est-à-dire son aspect dynamique. Comparé à l'oral qui n'offre pas de support, et à l'écrit traditionnel, un système informatique peut manipuler les connaissances pour en inférer de nouvelles (Bachimont, 1999).

Ces connaissances opérationnelles peuvent être des faits, des règles, ou des contraintes. Un fait est un énoncé vrai, non implicatif, un axiome qui participe à la description du monde cognitif dans lequel s'inscrit le SBC.
L'énoncé « l'entreprise E compte 20 salariés » est un exemple de fait. Une règle permet d'inférer de nouvelles connaissances et contient donc une implication. La règle « si une entreprise compte X salariés, alors elle paye X*100 euros de charges » permet de calculer les charges d'une entreprise. Les contraintes permettent de spécifier des impossibilités ou des obligations. Par exemple, on peut vouloir spécifier que toute société importante possède obligatoirement un conseil d'administration et qu'une société ne peut avoir qu'un seul PDG.

4.3 La construction des ontologies

4.3.1 Le cycle de vie des ontologies

Les ontologies étant destinées à être utilisées comme des composants logiciels dans des systèmes répondant à des objectifs opérationnels différents, leur développement

doit s'appuyer sur les mêmes principes que ceux appliqués en génie logiciel. En particulier, les ontologies doivent être considérées comme des objets techniques évolutifs et possédant un cycle de vie qui nécessite d'être spécifié. Les activités liées aux ontologies sont d'une part des activités de gestion de projet (planification, contrôle, assurance qualité), et d'autre part des activités de développement (spécification, conceptualisation, formalisation) ; s'y ajoutent des activités transversales de support telles que l'évaluation, la documentation, la gestion de la configuration (Blazquez et al, 1999). Un cycle de vie inspiré du génie logiciel est proposé dans (Dieng et al, 2001). Il comprend une étape initiale d'évaluation des besoins, une étape de construction, une étape de diffusion, et une étape d'utilisation. Après chaque utilisation significative, l'ontologie et les besoins sont réévalués et l'ontologie peut être étendue et, si nécessaire, en partie reconstruite.

La phase de construction peut être décomposée en 3 étapes : conceptualisation, ontologisation, opérationalisation . L'étape d'ontologisation peut être complétée d'une étape d'intégration au cours de laquelle une ou plusieurs ontologies vont être importées dans l'ontologie à construire (Fernandez et al, 1997). FERNANDEZ insiste sur le fait que les activités de documentation et d'évaluation sont nécessaires à chaque étape du processus de construction, l'évaluation précoce permettant de limiter la propagation d'erreurs. Le processus de construction peut être intégré au cycle de vie d'une ontologie comme indiqué en figure 4.2 (Gandon, 2002). La section suivante va être plus spécifiquement consacrée aux méthodologies mises en œuvre lors de la phase de construction afin, en particulier, de guider les choix délicats de conceptualisation et de représentation.

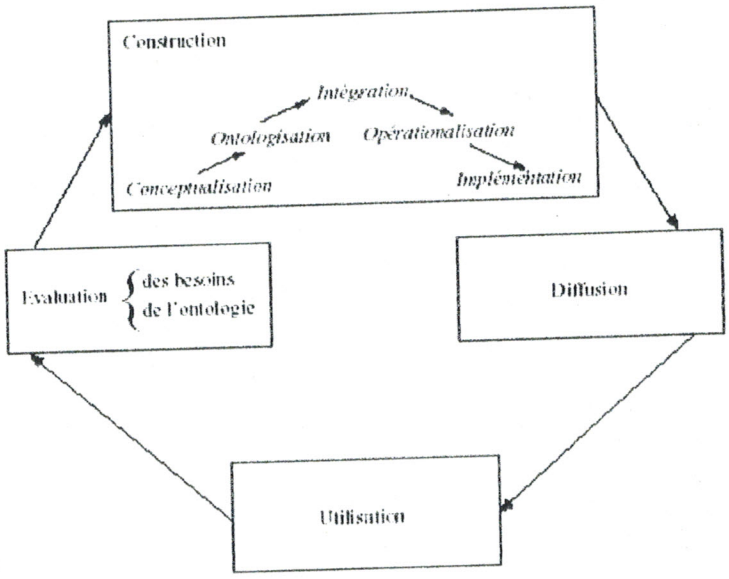


Fig. 4.2 - Le cycle de vie d'une ontologie

4.3.2 Les méthodologies de construction d'ontologies

Bien qu'aucune méthodologie générale n'ait pour l'instant réussi à s'imposer, de nombreux principes et critères de construction d'ontologies ont été proposés. Ces méthodologies peuvent porter sur l'ensemble du processus et guider l'ontologiste à toutes les étapes de la construction. C'est le cas de METHONTOLOGY, élaborée en 1998 par A. GOMEZ-PEREZ, qui couvre tout le cycle de vie d'une ontologie (Fernandez et al, 1997). M. USCHOLD et M. KING ont également proposé une

méthodologie générale, inspirée de leur expérience de construction d'ontologies dans le domaine de la gestion des entreprises (Uschold, 1996). La méthodologie présentée par M. GRUNINGER et M. S. FOX dans (Gruninger & Fox, 1994) est elle aussi issue d'une expérience de construction d'ontologie sur ce domaine.

Mais quelque soit la méthodologie adoptée, le processus de construction d'une ontologie est une collaboration qui réunit des experts du domaine de connaissance, des ingénieurs de la connaissance, voire les futurs utilisateurs de l'ontologie (Farquhar et al, 2000). Cette collaboration ne peut être fructueuse que si les objectifs du processus ont été clairement définis, ainsi que les besoins qui en découlent.

4.3.2.1 L'évaluation des besoins

Le but visé par la construction d'une ontologie se décline en 3 aspects (Uschold, 1996) :

L'objectif opérationnel : il est indispensable de bien préciser l'objectif opérationnel de l'ontologie, en particulier à travers des scénarios d'usage (Biebow & Szulman, 1999);
Le domaine de connaissance : il doit être délimité aussi précisément que possible, et découpé si besoin en termes de connaissances du domaine, connaissances de raisonnement, connaissances de haut niveau (communes à plusieurs domaines) ;
Les utilisateurs : ils doivent être identifiés autant que faire se peut, ce qui permet de choisir, en accord avec l'objectif opérationnel, le degré de formalisme de l'ontologie, et sa granularité.

Une fois le but défini, le processus de construction de l'ontologie peut démarrer, en commençant par la phase de conceptualisation.

4.3.2.2 La conceptualisation

La conceptualisation consiste à identifier, dans un corpus[1] , les connaissances du domaine. Cette conceptualisation peut se décomposer en deux étapes. Tout d'abord, il faut faire le tri entre connaissances spécifiques au domaine et celles qui, bien que présentes dans le corpus, ne participent qu'à l'expression des connaissances du domaine.

En outre, s'il est prévu d'intégrer d'autres ontologies, les connaissances spécifiées dans ces ontologies ne doivent pas être prises en compte. La nature conceptuelle (concepts, relations, propriétés des concepts et relations, règles, contraintes, etc.) des connaissances ainsi extraites du corpus doit ensuite être précisée. Des choix liés aux contextes d'usage de l'ontologie doivent donc être effectués dès cette étape.

4.3.2.3 L'ontologisation

Une formalisation partielle, respectant l'intégrité du modèle conceptuel, va permettre, à cette étape, de construire une ontologie proprement dite. Afin de respecter les objectifs

[1] Un corpus peut être constitué par des interviews d'experts du domaine, des documentations techniques, etc.

généraux des ontologies, T. GRUBER propose 5 critères permettant de guider le processus d'ontologisation (Gruber, 1993) :

- la clarté et l'objectivité des définitions, qui doivent être indépendantes de tout choix d'implémentation ;
- la cohérence (consistance logique) des axiomes ;
- l'extensibilité d'une ontologie, c'est-à-dire la possibilité de l'étendre sans modification ;
- la minimalité des postulats d'encodage, ce qui assure une bonne portabilité ;
- la minimalité du vocabulaire, c'est-à-dire l'expressivité maximum de chaque terme.

De plus, il faut bien voir que l'ontologisation est une traduction dans un certain formalisme de connaissances exprimées a priori en langage naturel. Le respect de la sémantique du domaine doit être assuré par un engagement ontologique, notion proposée initialement par T. GRUBER comme un critère pour utiliser une spécification partagée d'un vocabulaire (Gruber, 1993). Respecter l'engagement ontologique revient à donner à chaque concept son extension et à manipuler ce concept conformément au sens prescrit par cette extension.

L'ontologisation doit mener à la construction de hiérarchies de concepts, de relations mais aussi d'attributs de concepts. Après cette phase, et une fois le modèle conceptuel structuré, il faut le traduire en langage semi-formel de représentation d'ontologies.
Le travail de structuration peut d'ailleurs être mené en même temps, les langages en question offrant tous la possibilité de représenter des hiérarchies de concepts et de relations.

Parmi les langages de représentation développés au niveau conceptuel, trois grands modèles sont distingués : les langages à base de frame, les logiques de description et le modèle des graphes conceptuels.

Quelques-uns uns de ces langages ou des langages utilisant ces modèles sont déjà opérationnels et les ontologies exprimées dans ces formalismes peuvent être directement utilisées en machine. Dans les autres cas, une opérationnalisation de l'ontologie est nécessaire.

Le modèle à base de frame

Introduit dès les années 70 en IA, le modèle des frames a depuis été adapté à d'autres problématiques puisqu'il a donné naissance au modèle objet, qui envahit peu à peu les différentes branches de l'informatique. Une frame représente n'importe quelle primitive conceptuelle et est dotée d'attributs (slots), qui peuvent porter différentes valeurs (facets), et d'instances (Kifer et al, 1995). La sémantique de la subsomption est purement référentielle : une frame
F1 est plus spécifique qu'une frame F2 si toute instance de F1 est instance de F2. FLOGIC est l'exemple le plus connu de langage à base de frames (Kifer et al, 1995). L'OPEN KNOWLEDGE BASE CONNECTIVITY (OKBC (Chandrasekaran et al,

1998b)), protocole et API de requête et d'interfaçage entre bases de connaissances, utilise également le modèle de frame.

Propriétés du système de frame :

- Connaissances sur une entité, ses parties et ses constituants.
- Parties et constituants représentés comme des slots (attributs)
- Liaisons avec des frames à travers des hiérarchies de types
- Mécanisme d'inférence : subsomption, classification
- Systèmes des frames représentent souvent la même entité vue sous forme de différentes perspectives.

Exemple : [Tweety est_un Canari with
Couleur Jaune
Sexe Male
Age 2
Santé excellente
.....
.....]

forme générale :
[<Nom forme> est-un <Type>
<Nom slot> <Valeur Slot>
<Nom slot> <Valeur slot>
.....
.....]

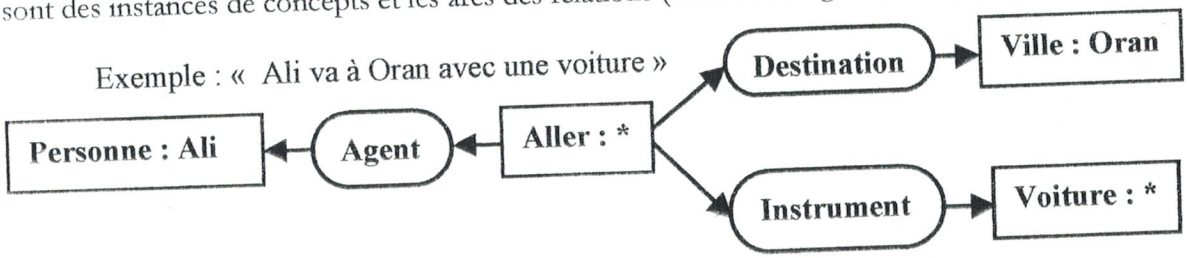
Les logiques de description

Les logiques de description permettent de représenter les connaissances sous forme de concepts, de rôles et d'individus (Kayser, 1997). Les rôles sont des relations binaires entre concepts et les individus sont les instances des concepts. Les propriétés des concepts, rôles et individus sont exprimées en logique des prédicats, en particulier les propriétés de subsomption. Au niveau terminologique sont définis les concepts, les rôles et leurs propriétés. Les faits portant sur des individus (types des individus et relations entre individus) sont exprimés au niveau factuel. LOOM (Macgregor, 1995) et KL-ONE (Brachman & Schmolze, 1985) sont des exemples de systèmes implémentant ce modèle. Il est de plus utilisé dans le langage de représentation de connaissance OIL développé pour le Web.

Le modèle des graphes conceptuels

Introduit par SOWA au début des années 80 (Sowa, 1984), le modèle des graphes conceptuels se décompose en deux niveaux : le niveau terminologique où sont décrit les concepts, les relations et les instances de concepts, ainsi que les liens de subsomption entre concepts et entre relations, et le niveau assertionnel où sont représentés les faits, les règles et les contraintes sous forme de graphes où les sommets sont des instances de concepts et les arcs des relations (Chein & Mugnier, 2001).

Exemple : « Ali va à Oran avec une voiture »



- Traduit par une fonction ϕ :
  $(\exists x)(\exists y)$ (Personne (Ali) ^ Aller(x) ^ville (Oran) ^ voiture(y) ^ Agent(x, Ali) ^ Destination(x, Oran) ^ Instrument(x, y)
- En notation linéaire :
  [Aller :*]- {(agent) ⟶ [personne : Ali] ;
  (Destination) ⟶ [Ville : Oran] ;
  (Instrument) ⟶ [Voiture :*] ;
  }

Ce formalisme est implémenté, entre autres, dans COGITANT, une plateforme de développement de SBC utilisant les graphes conceptuels (Genest & Salvat, 1998) et PROLOG+CG, une extension de PROLOG basée sur les graphes conceptuels (Kabbaj, 2000).

4.3.2.4 L'opérationnalisation

L'opérationnalisation consiste à outiller une ontologie pour permettre à une machine, via cette ontologie, de manipuler des connaissances du domaine. La machine doit donc pouvoir utiliser des mécanismes opérant sur les représentations de l'ontologie. Or, si beaucoup de langages réifiant les modèles cités précédemment autorisent l'expression de connaissances inférentielles, peu sont outillés pour rendre possible la manipulation de ces connaissances.

Le modèle des graphes conceptuels fait exception car la représentation des connaissances sous forme de graphes permet de mettre en œuvre des raisonnements par des opérations formelles sur les graphes (comparaison, fusion, etc.). La façon de mener ces opérations dépend cependant de l'objectif opérationnel du système envisagé.

Dans le cas où le langage d'ontologisation n'est pas opérationnel, il est nécessaire, soit d'outiller ce langage, dans la mesure du possible, soit de transcrire l'ontologie dans un langage opérationnel. Mais certains langages offrent des possibilités de raisonnement limitées qui peuvent convenir à certaines applications limitées. Par exemple, les langages à base de frames et les logiques de description permettent de savoir si une connaissance donnée, ou une connaissance plus spécifique qu'une connaissance donnée, est présente dans une base de connaissances en utilisant la relation de subsomption. Dans le cas d'un simple système de stockage et de consultation de connaissances, de tels langages sont donc suffisants.

Finalement, l'ontologie opérationnalisée est intégrée en machine au sein d'un système manipulant le modèle de connaissances utilisé via le langage opérationnel choisi.

Mais avant d'être livrée aux utilisateurs, l'ontologie doit bien sur être testée par rapport au contexte d'usage pour lequel elle a été bâtie.

4.4 Quelques exemples d'utilisation des ontologies

Loin de n'être qu'un objet de laboratoire, les ontologies trouvent aujourd'hui des applications dans tous les systèmes à base de connaissances et d'autres domaines plus précisément le WEB sémantique.

- **Projet MENELAS** : mené dans les services informatiques de l'assistance publique HOPITAUX DE PARIS et qui vise la gestion des rapports médicaux et leur analyse par un système qui utilise le modèle des graphes conceptuels. L'utilisation des mécanismes de raisonnement adaptés permet alors la consultation interactive de la connaissance, le système disposant des moyens d'aiguiller la recherche de l'utilisateur par des questions et/ou des propositions.

- **Projet TOVE** (Toronto Virtual Entreprise) a pour but de créer un modèle d'entreprise exprimé dans une ontologie

- **Projet CoMMA** mené à l'INRIA vise également à permettre la gestion d'une mémoire partagée des connaissances à l'intérieur d'une entreprise.

- **Projet GINA** qui représente une ontologie dans le domaine de la géométrie.

- Et d'autres projets dans différents domaines

Conclusion

Au long de ce chapitre, nous avons essayé d'éclaircir la notion d'ontologie en recherchant son origine, puis en mettant en perspective un certain nombre de travaux de la communauté Ingénierie des connaissances. Nous espérons avoir montré que la notion d'ontologie a évolué dans un continuum de réflexions cohérentes pour que puisse se tracer un chemin d'évolution vers la stabilisation du concept. Alors une définition d'une méthodologie unifiée de construction et de validation des ontologies est nécessaire, en particulier pour faciliter la fusion d'ontologie. Cette unification doit porter sur les principes de structuration sémantique des connaissances mais également sur les langages opérationnels de représentation. Ces langages doivent d'autre part être enrichi pour permettre l'expression et la manipulation de connaissances inférentielles. De véritables raisonnements pourraient alors être menés au niveau sémantique et ainsi élargir substantiellement les capacités des systèmes d'information.

Bien que les ontologies partagées et l'extension des ontologies autorisent un certain degré d'interopérabilité entre des organisations et des domaines différents. Et afin que les machines puissent intégrer les informations engagées sur des ontologies hétérogènes(web sémantique(chap. 4)), il est nécessaire de disposer de primitives qui permettent aux ontologies de relier des concepts à leurs équivalents dans d'autres ontologies, ou autrement dit, un système qui capture les correspondances sémantiques entre ces différentes ontologies qui représentent des sources de données hétérogènes, en utilisant une architecture distribuée, et pour cela on va présenter un tel réseau qui supporte sa au prochain chapitre.

Références

BACHIMONT B., L'intelligence artificielle comme écriture dynamique : de la raison graphique à la raison computationnelle, *Grasset, Paris*, 1999.

BACHIMONT B., Engagement sémantique et engagement ontologique : conception et réalisation d'ontologies en ingénierie des connaissances, in CHARLET J., ZACKLAD M., KASSEL G. & BOURIGAULT D., eds., *Ingénierie des connaissances : évolutions récentes et nouveaux défis*, Eyrolles, pages 305-323, 2000.

BIEBOW M. & SZULMAN S., TERMINAE : a method and a tool to build of a domain ontology, in *Proceedings of the 11th European Knowledge Acquisition Workshop (EKAW'99)*, Springer, 1999.

BLAZQUEZ M., FERNANDEZ M., GARCIA-PINAR J. M. & GOMEZ-PEREZ A., Building Ontologies at the Knowledge Level using the Ontology Design Environment, in *Proceedings of the Banff Workshop on Knwoledge Acquisition for Knwoledge-based Systems*, 1998.

BRACHMAN R. & SCHMOLZE J., An overview of the KL-One knowledge representation system, in *Cognitive Science 9(2)*, pages 171-216, 1985.

CHANDRASEKARAN B., JOSEPHSON J. & BENJAMINS V., The Ontology of Tasks and Methods, in *Proceedings of the 11th workshop on Knowledge Acquisition, Modeling and Management (KAW'98)*, 1998a.

CHAUDHRI V., FARQUHAR A., FIKES R., KARP P. & RICE J., OKBC : a programmatic foundation for knowledge base interoperability, in *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI'98)*, MIT Press, 1998b.

CHARLET J., L'ingénierie des connaissances : une science, un enseignement ?, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2001)*, Presse Universitaire Grenobloise, pages 233-252, 2001.

CHEIN M. & MUGNIER M., Conceptuals graphs : fundamental notions, *Revue d'Intelligence Artificielle (RIA)*, 6(4), pages 365-406, 1992.

DIENG R., CORBY O., GANDON F., GIBOIN A., GOLEBIOWSKA J., MATTA N. & RIBIÈRE M., *Méthodes et outils pour la gestion des connaissances : une approche pluridisciplinaire du knowledge management (2ième édition)*, Dunod Edition Informatiques Séries Systèmes d'Information, 2001.

FARQUHAR A., FIKES R. & RICE J., Ontolingua server : a tool for collaborative ontology construction, in *International journal of Human-Computer studies* (46), pages 707-727, 2000.

FERNANDEZ M., GOMEZ-PEREZ A. & JURISTO N., METHONTOLOGY : from ontological art towards ontological engineering, in *Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97)*, AAAI Press , 1997.

GANDON F., Ontology Engineering : a survey and a return on experience, rapport de recherche 4396, INRIA, 2002.

GENEST D. & SALVAT E., A platform allowing typed nested graphs : how CoGITo became CoGITaNT, in *Proceedings of the International Conference on Conceptual Structures (ICCS'98)*, Springer-Verlag LNAI 1453, pages 154-161, 1998.

GRUBER T., A translation approach to portable ontology specifications, *Knowledge Acquisition* 5(2), pages 199-220, 1993.

GRUBER T. & OLSEN G., An ontology for enginering mathematics, in J. DOYLE F. S.&TORANO P., eds., *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasonning*, Morgan-Kauffmann, 1994.

GRUNINGER M. & FOX M. S., Methodology for the design and evaluation of ontologies, in *Proceedings of the Workshop on Basic Ontological Issues on Knowledge Sharing, IJCAI'95*, 1995.

GUARINO N., CARRARA C. & GIARETTA P., An ontology of meta-level categories, in J. DOYLE F. S.&TORANO P., eds., *Principles of Knowledge representation and Knowledge Reasonning*, Morgan-Kauffmann, pages 270-280, 1994.

GUARINO N. & GIARETTA P., Ontologies and knowledge bases, towards a terminological clarification, inMARS N., eds., *Towards very large knowledge bases : knowledge building and knowledge sharing*, IOS Press, pages 25-32, 1995.

HORROCKS I. DAML+OIL and Description Logic Reasoning. *Talk given at HP Labs, Bristol*, October, 2001;

KABBAJ M., From Prolog++ to Prolog+CG : a CG objet-oriented logic programming language, in *Proceedings of the International Conference on Conceptual Structures (ICCS'00)*, Springer LNAI 1867, pages 540-554, 2000.

KASSEL G., ABEL M., BARRY C., BOULITREAU P., IRASTORZA C.&PERPETTE S., Construction et exploitation d'une ontologie pour la gestion des connaissances d'une équipe de recherche, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2000)*, 2000.

KASSEL G. La gestion des concepts et du vocabulaire dans l'entreprise ; *Terminologies et Ontologies*. Rapport interne IIIA-01-VT11, IIIA. Rapport de veille technologique IIIA, 2001.

KASSEL G., OntoSpec : une méthode de spécification semi-informelle d'ontologies, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2002)*, pages 75-87, 2002.

KAYSER D., *La représentation des connaissances*, Hermès, 1997.

KIFER M., LAUSEN G. & WU J., Logical foundations of object-oriented and frame-based languages, in *Journal of the ACM*, 1995.

MACGREGOR R., Inside the LOOM classifier, in *SIGART bulletin* (2)3, pages 70-76, 1991.

MIZOGUCHI R., & IKEDA M., Towards ontolgy engineering, in *Proceedings of the Joint Pacific Asian Conference on Expert Systems*, 1997.

NOBÉCOURT J. & BIÉBOW B., MDOS : a modelling language to build a formal ontology en either Description Logics or Conceptual Graphs, in *Proceedings of the 12th International*

Conference on Knowledge Engineering and Management (EKAW'2000), Springer-Verlag LNAI 1937, pages 57-64, 2000.

SOWA J., Conceptual structures : information processing in mind and machine, Addison-Wesley, 1984.

SOWA J., Ontology, Metadata and semiotics, in *Proceedings of the 8th International Conference on Conceptual Structures (ICCS'2000)*, Springer-Verlag, pages 55-81, 2000.

TEULIET R. & GIRARD N., Des connaissances pour l'action dans les organisations. Quelle ingénierie des connaissances pour assister l'activité ?, in *Actes des journées francophones d'Ingénierie des Connaissances (IC'2001)*, Presse Universitaire Grenobloise, pages 253-272, 2001.

USCHOLD M., Building ontologies : towards a unified methodolgy, *in Proceedings of the 16th conference of the British Computer Society Specialist Group on Expert Systems*, 1996.

USCHOLD M. & GRUNINGER M. Ontologies : Principles, methods and applications. *Knowledge Engineering Review*, 1996.

ZWEIGENBAUM P., Encoder l'information médicale : des terminologies aux systèmes de représentation des connaissances, in *Innovation stratégique en information de santé (ISIS)* (2-3), pages 27-47, 1999.

Chapitre 5

Réseaux pair-à-pair

5.1 Introduction

Ce chapitre s'attache à présenter les aspects techniques de notre étude sur le peer-to-peer. Nous livrons tout d'abord un bref historique de l'informatique pour y situer l'apparition du peer-to-peer. Une étude technique introduit le concept tel que nous le connaissons aujourd'hui et l'analyse. Nous décrivons ensuite les principales architectures possibles, pour finalement considérer les possibilités d'optimisation dont peuvent bénéficier les protocoles peer-to-peer.

5.2 Historique

5.2.1 L'hégémonie du client/serveur

Le monde universitaire, fonctionne selon ses propres logiques internes, mais il subit également des influences issues des autres acteurs sociaux. Ainsi, chaque secteur cherche avant tout à conquérir le plus d'autonomie possible par rapport aux autres. Et, c'est dans ce contexte de conquête de l'autonomie qu'il fa ut comprendre l'origine d'Arpanet, l'ancêtre d'Internet.

5.2.2 L'apparition du peer-to-peer

Garantissant un bon contrôle de l'information le client/serveur s'est montré capable de répondre parfaitement aux attentes des acteurs de l'Internet. Sa suprématie s'est longtemps montrée la seule solution acceptable pour intégrer les clients légers au réseau. Mais le modèle client/serveur s'éloigne de la philosophie égalitaire qui avait donné naissance à l'Arpanet.

L'originalité dans un réseau pair-à-pair réside dans le fait que les fichiers transférés ne le sont pas au travers d'un serveur mais directement d'un utilisateur vers un autre utilisateur, définissant ainsi le principe d'échange du « peer-to-peer » ; c'est à dire d'usager à usager.
En fait ce système redécouvre l'Internet dans son essence fondamentale, lorsque l'Internet offrait une structure symétrique pour faciliter le partage de l'information à l'intérieur de la communauté scientifique des premières universités connectées.

5.3 Présentation

Si le mot peer-to-peer a pris toute son ampleur très récemment, l'informatique peer-to-peer, elle, ne date pas d'hier. Il y a 30 ans, certaines entreprises travaillaient déjà avec des architectures qui seraient aujourd'hui qualifiées de pair à pair.

5.3.1 définition

Dans son essence, l'informatique pair à pair se définit comme le partage des ressources et des services informatiques par échanges direct entre systèmes. Ces échanges peuvent porter sur les informations, les cycles de traitement, la mémoire cache ou encore

le stockage sur disque des fichiers(Gribble et al, 2001). Contrairement au modèle client/serveur, le modèle de communication peer-to-peer fait de chacun des nœuds du réseau une entité complète qui remplit à la fois le rôle de client et de serveur.

Aujourd'hui, néanmoins, plusieurs facteurs participent à l'explosion du phénomène : la puissance, la largeur de bande passante et la capacité de stockage, désormais disponibles à bas prix. Le peer-to-peer tire parti de la puissance existante des PC et de la connectivité des réseaux afin de mettre le potentiel collectif des nœuds au service de l'ensemble du réseau.
Notons que le terme peer-to-peer n'aide pas à clarifier les choses. Le jeu en réseau Doom, l'email ou même le téléphone peuvent être vus comme des systèmes peer-topeer (on échange des informations directement entre pairs) alors que Napster, qui a popularisé le concept, est paradoxalement construit autour d'un serveur central.
Le vrai changement, c'est la nature des éléments qui constituent les réseaux peer-to-peer. Dans le passé, les centaines de millions d'ordinateurs connectés à Internet de manière intermittente ne faisaient pas partie du réseau. Avec le peer-to-peer, les ordinateurs personnels ont le pouvoir de devenir une partie intégrante du réseau. Le peer-to-peer désigne donc une classe d'applications qui tirent partie des ressources matérielles ou humaines qui sont disponibles en bordure du réseau de l'internet.

5.3.2 Caractéristiques

Comme ces ressources ont une connectivité instable ou des adresses IP variables, elles fonctionnent de manière autonome, indépendamment de systèmes centraux comme les DNS. Ce qui a rendu Napster et des systèmes similaires populaires, c'est le fait de tirer partie des ressources qui était auparavant inutilisées en tolérant d'une connectivité aléatoire(Kantere, 2002).

Un vrai système peer-to-peer peut se reconnaître en se posant la question des deux caractéristiques suivantes :

- Est-ce que le système permet à chaque pair de se connecter de manière intermittente avec des adresses IP variables ?
- Est-ce que le système donne à chaque pair une autonomie significative ?

Si la réponse à ces deux questions est oui, le système est un système peer-to-peer. Notons au passage que le modèle client/serveur apporte une réponse négative aux deux questions.

Une autre manière de distinguer un système peer-to-peer est de raisonner en terme de « propriété ». La question à se poser est alors : « Qui possède les ressources qui font tourner le système ? ». Dans un système comme celui de Yahoo! – un grand architecte des services web -, l'essentiel des ressources est possédé par la société et ses partenaires, qui les mettent à la disposition des internautes, tandis que dans un système comme celui de Napster, les ressources sont un bien commun des membres du réseau.

Dans les vrais systèmes peer-to-peer, on ne se contente pas de décentraliser les fonctions mais aussi les coûts et les charges d'administration. Si l'on fait un calcul basique, 100 millions de PC connectés à l'Internet avec 100 Mo d'espace disque représentent une puissance potentielle de 10 000 téraoctets de stockage. C'est de ces ressources que le peer-to-peer permet de tirer parti.

Une caractéristique importante des réseaux de peers décentralisés est que la valeur perçue est directement liée à la quantité et la qualité des données qui y sont disponibles. Les ressources sont ajoutées à mesure que le nombre de peers du réseau augmente. Donc la valeur du réseau augmente avec sa popularité.

Enfin, la pertinence d'un système peer-to-peer réside dans sa capacité à localiser les ressources efficacement quelle que soit la taille du réseau. Aussi, ces systèmes doivent reposer sur des méthodes efficaces de découverte des ressources désirées. En général, la solution employée est l'utilisation de méta-données.

5.3.3 Avantages et inconvénients

Les systèmes peer-to-peer, en plus d'élargir le nombre de ressources disponibles pour les services, permettent d'en faire un usage plus efficace. Ces ressources inexploitées, se trouvant en bordure du réseau. Le peer-to-peer trouve parfaitement sa place dans le travail collaboratif. Malgré l'utopie de pouvoir éviter l'usage de serveurs centralisés, peu d'applications (ou services) pourront effectivement se passer de points d'entrée centralisés. La combinaison des services web (centralisés) et du peer-to-peer apporte une collaboration totale, en temps réel ou asynchrone. Certaines applications peer-to-peer seront mieux adaptées pour fonctionner en Intranet afin d'exploiter et de partager l'information entre les utilisateurs (partage des fichiers, recherche dans des formats de fichiers différents : mail, tableurs, etc).
Les réseaux peer-to-peer doivent être capables d'identifier uniquement les ressources disponibles. Ce que le DNS tente de réaliser pour les machines, le peer-to-peer doit le faire au niveau des ressources. Par conséquent, les systèmes peer-to-peer ont dû adopter leurs propres schémas de nommage et de localisation indépendamment des adresses IP. Cette caractéristique leur permet de tirer habilement parti de la disponibilité intermittente des ressources se trouvant en bordure de l'Internet (celles des machines personnelles).

Un réseau peer-to-peer présente la particularité de ne fonctionner correctement que lorsqu'un nombre suffisant de nœuds participe au partage de ressources. Un réseau peer-to-peer peu populaire ne sera pas capable de survire sans participants. A l'opposé, un réseau peer-to-peer mal adapté à la montée en charge risque de souffrir de cette popularité. La nature décentralisée des réseaux peer-to-peer implique un trafic important pour localiser les autres participants, supporter l'intermittence des connexions et acheminer les données. Certaines études (Eytan & Bernardo, 2002) .montrent qu'un système fortement décentralisé peut générer un trafic plus que néfaste pour la bande passante.

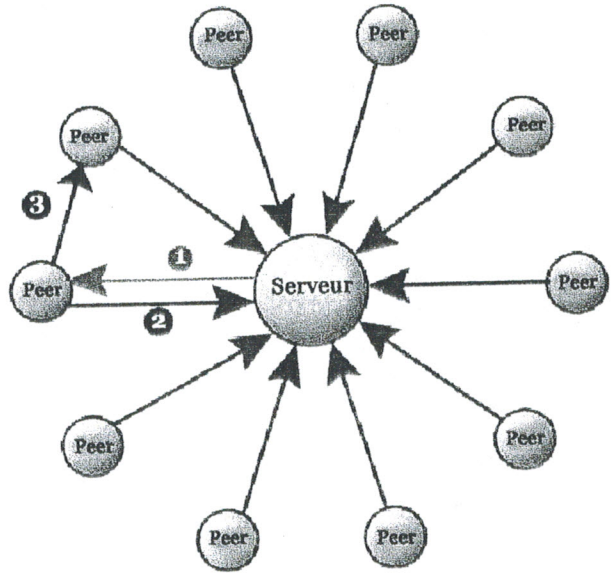
En facilitant le partage et la diffusion d'information, les systèmes peer-to-peer créent une certaine redondance des données les rapprochant physiquement des bordures du réseau. Ce mécanisme de réplication de l'information est profitable aux nœuds disposant d'une faible bande passante. Le contrôle de l'information dans une architecture décentralisée rend l'administration et la sécurité difficiles à maîtriser. La problématique de la sécurisation peut être résolue par des bibliothèques de fonctions peer-to-peer dédiées faisant intervenir des techniques de cryptage et d'authentification à clé publique. Néanmoins, comme l'a montré le cas Napster, l'échange des contenus protégés par la propriété intellectuelle (ouvrages, musique, vidéos, jeux, logiciels) est très difficilement contrôlable dans une communauté peer-to-peer native.

5.4 Architectures

Derrière la définition largement répandue de peer-to-peer coexistent deux modèles d'architecture. D'un côté, le peer-to-peer assisté, construit autour d'un serveur central. De l'autre, le peer-to-peer décentralisé qui repose sur des nœuds de réseaux. Sur ces deux modèles se greffent un certain nombre d'optimisations et de variantes topologiques dont les applications peer-to-peer peuvent tirer parti en fonction de leurs besoins(Yang & Garcia-Molina, 2002).

5.4.1 Le peer-to-peer assisté

Dans ce modèle, on emploie un serveur central qui permet d'indexer les peers connectés au réseau et les ressources disponibles. Cet index est utile pour fournir aux autres peers une cartographie des ressources disponibles sur le réseau. Le serveur assure le service de localisation des ressources et les peers peuvent alors communiquer directement entre eux sans l'assistance du serveur.



1 : l'utilisateur (peer) envoie une requête. Le serveur cherche dans sa base de données centrale.
2 : le serveur renvoie une liste de fichiers disponibles.
3 : l'utilisateur consulte le fichier directement depuis la machine d'un autre utilisateur.

Figure 5.1: peer-to-peer assisté

Un peer A se connecte au réseau en annonçant à un serveur quelles ressources il met à disposition. Le serveur maintient un index des ressources disponibles sur le réseau à chaque instant. Si le peer A a besoin d'une ressource, il interroge le serveur qui localise la ressource sur un autre peer B connecté au réseau. Le serveur n'intervient plus à ce moment là et le transfert de la ressource s'effectue directement entre A et B.

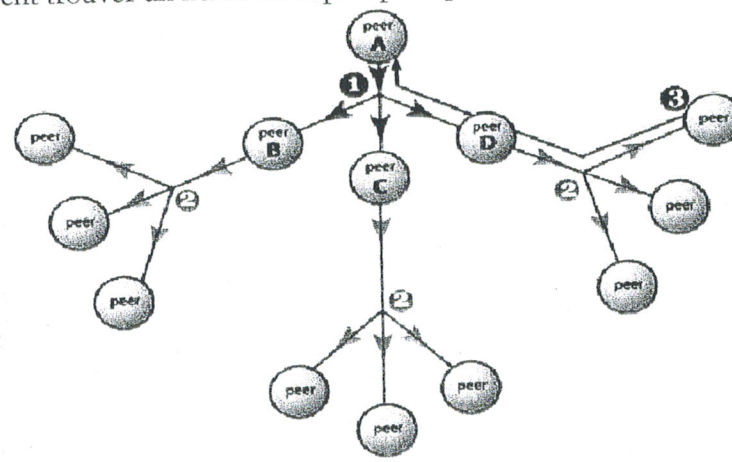
Certains systèmes reposent complètement sur un serveur, comme c'est le cas de SETI@Home, d'autres utilisent l'assistance d'un serveur pour réaliser une partie de leurs tâches. Le service de recherche de Napster est centralisé alors que le service de

téléchargement est décentralisé. L'un des principaux avantages de ce modèle est l'utilisation d'un index central qui
permet de localiser les ressources rapidement et efficacement. Cet index doit être mis à jour en permanence en fonction de l'activité du réseau. Les peers peuvent avoir une connexion intermittente sans que cela ne nuise à l'intégrité du système.

Principal problème, en revanche, ce type de système n'autorise qu'un seul point d'entrée sur le réseau, et n'est pas à l'abri d'une coupure de serveur susceptible de bloquer toute l'application. Des architectures hybrides (voir §5.4.3) permettent de contourner cette faiblesse.

5.4.2 Le peer-to-peer décentralisé

Le deuxième modèle repose entièrement sur les nœuds du réseau plutôt que sur un serveur. Chacun des nœuds met à disposition de l'ensemble du réseau un certain nombre de ressources. Le service de localisation comme le routage des requêtes est pris en charge par l'ensemble du réseau au lieu de faire appel à un serveur. Ce modèle rend l'accès au réseau plus difficile à utiliser que le premier, car les nœuds qui désirent joindre le réseau doivent trouver un nœud de départ pour pouvoir accéder au service.



1 : A envoie une première requête vers B,C et D.
2 : B,C et D envoie la requête vers la couche suivante d'ordinateurs connectés.
3 : le fichier recherché est localisé et une réponse à A selon le même chemin.

Figure 5.2 : peer-to-peer décentralisé

Le principe est le suivant : un peer A, équipé d'un programme spécifique, se connecte à un peer B, lui aussi équipé de ce programme. A lui annonce ainsi sa présence sur le réseau.
B relaie cette information à tous les peers auquel il est connecté. Ceux-ci signifient alors leur présence à A et relaient l'information à leur tour aux peers auxquels ils sont connectés, et ainsi de suite jusqu'à « l'horizon visible » du peer A. Une fois que A a annoncé sa présence aux autres membres du réseau de peers, il peut s'appuyer sur les peers qu'il connaît pour effectuer ses recherches. Pour obtenir une ressource, A lance une requête vers certains peers du réseau qui la relaient vers les peers auxquels ils sont connectés, qui eux même la transmettent.
Si l'un des peers dispose d'une ressource qui pourrait convenir à A, il transmet l'information vers A. Ce dernier pourra ainsi ouvrir une connexion directe vers cet ordinateur et obtenir la ressource.

Ce modèle, en étant décentralisé, est beaucoup plus robuste qu'un modèle centralisé puisqu'il n'est pas dépendant d'un serveur, qui est le point de défaillance potentiel d'un réseau centralisé. Il tire partie de l'intermittence des connexions des nœuds car si l'un des noeuds se déconnecte du réseau, la requête pourra être poursuivie vers les autres ordinateurs connectés.
En revanche, en raison de la façon dont sont transmises les requêtes (broadcast), la bande passante né cessaire pour chaque requête croît exponentiellement quand le nombre de peers croît linéairement. De plus, ce type de mécanisme est très facilement victime d'activités malicieuses.
Des membres malintentionnés peuvent envoyer en grande quantité des requêtes erronées qui produisent une lourde charge sur le réseau, réduisant ainsi son efficacité.

5.4.3 Alternatives et améliorations

Il existe un certain nombre de techniques et d'architectures alternatives qui tentent d'apporter des solutions aux problèmes que connaissent ces deux modèles.

Architectures hybrides

Le modèle assisté apporte certainement la meilleure des solutions en terme de rapidité de résolution des requêtes. Le modèle décentralisé, lui, apporte la robustesse mais doit consommer beaucoup de ressources pour acheminer les requêtes. Des modèles hybrides font leur apparition afin de tirer parti des points forts des deux modèles de base. La gamme des topologies hybrides imaginable est assez vaste et permet de répondre à de nombreuses problématiques. C'est notamment le cas dans un environnement d'entreprise où des points d'entrée centralisés sont nécessaires pour des raisons de maintenance et de sécurité (Nejdl et al, 2002).

La combinaison de plusieurs services peut nécessiter plusieurs serveurs organisés de façon hiérarchique. Cette topologie peut être nécessaire lorsqu'il existe une chaîne d'actions bien déterminée entre les services. Ces systèmes offrent généralement une grande possibilité d'extension comme l'a démontré le DNS conçu il y a 15 ans et qui supporte aisément les millions de nœuds que comporte l'Internet aujourd'hui. Ce modèle est une alternative aux systèmes centralisés et permet d'intégrer facilement de nombreux services. La racine de la structure hiérarchique représente cependant un point potentiel de défaillance et la sécurisation du système est difficile.

Architecture Hiérarchique

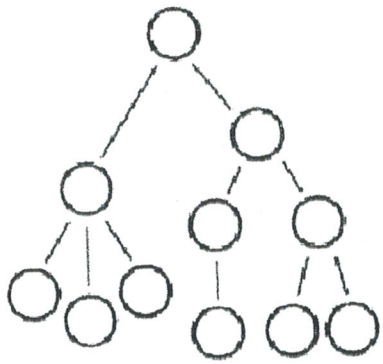


Figure 5.3 : Architecture hiérarchique

L'organisation de grappes de serveurs en anneau est largement utilisée dans le web. L'anneau permet de réaliser la répartition de charge et dilue le risque de défaillance localisée. En connectant le réseau de peers à ces anneaux, on bénéficie de la simplicité d'un système assisté avec la robustesse d'un système décentralisé.

Architecture en Anneau

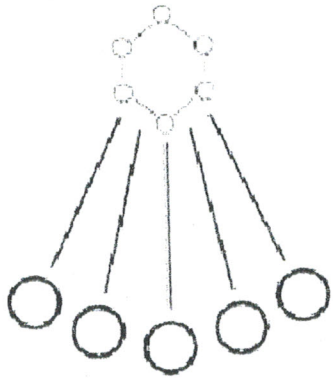


Figure 5.4 : Architecture en anneau

Une nouvelle vague de systèmes peer-to-peer utilise une architecture combinant des systèmes centralisés au sein de systèmes décentralisés. Ce type de réseau remplace les serveurs dédiés centraux qui réalisent l'indexation du contenu par un grand nombre de « super-nœuds » disposant de bande passante et de puissance de calcul au dessus de la moyenne.

Les super-nœuds peuvent aussi être utilisés pour relayer des requêtes lorsque les ressources trouvées sont insuffisantes. Ce principe est populaire et améliore grandement les performances des réseaux décentralisés. Il a notamment été intégré au réseau Gnutella et dans le système de FastTrack (FastTrack, Site web). Les peers sont reliés à au moins un super-nœud qui n'est pas isolé mais intégrés dans le réseau décentralisé.

Architecture avec super-nœuds

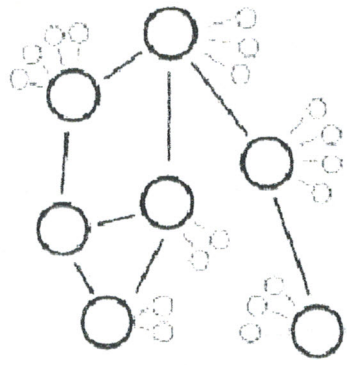


Figure 5.3 : Architecture avec super-nœuds

C'est un modèle qui se rapproche de celui de l'email. Les clients mail sont rattachés à un ou plusieurs serveurs mail qui se partage la tâche de routage des emails. La croissance phénoménale d'Internet depuis la conception initiale du système d'email n'a en rien dégradé ses performances. Un tel système est donc capable d'équilibrer la charge sur l'ensemble du réseau, et de limiter le risque d'interruption de service puisque si un super-nœud n'est pas disponible, les autres réaliseront ses tâches. Cependant dans les systèmes actuels le choix de fonctionner en tant que super-nœud est une option laissée à la générosité de chaque utilisateur.

Optimisations

Un certain nombre d'optimisations du réseau peut être intégré aux systèmes peer-to-peer (DRDLPBN, Site web). Une des pratiques les plus néfastes aux réseaux peer-to-peer d'intérêt commun est celle du freeloading. Celle-ci consiste à utiliser les ressources du réseau sans partager les sienne s. La charge du réseau s'en trouve augmentée sans que sa valeur n'augmente.
La lutte contre le freeloading est un point crucial que certains systèmes comme Mojo Nation (Mojo Nation, Site web) combattent en attribuant plus de crédits aux peers qui favorisent le fonctionnement du réseau.

Les systèmes actuels utilisent généralement TCP comme protocole de transport. De par sa nature non connectée, il est difficile de reprendre une connexion perdue entre deux sessions d'utilisation du réseau. Rendre les connexions persistantes permettrait de maintenir un historique des sessions et de construire des profils locaux sur les autres peers. Ces profils seraient alors utilisés pour déterminer la valeur d'un peer ainsi que pour monitorer les ressources qu'utilise.

Au sein d'une communauté, les membres n'ont pas les mêmes affinités. Etablir des profils de peer donne l'avantage de permettre au réseau de se réorganiser en groupes d'intérêts différents. Par exemple un amateur de science- fiction partagera des ressources qui ne conviendront pas nécessairement à un amateur de littérature du 18ème siècle.
Favoriser les requêtes vers les peers disposant, d'après leurs profils, de ressources plus intéressantes permet de rendre les recherches plus pertinentes. Des groupes de peers ayant des intérêts similaires s'organiseront alors spontanément comme ils le feraient dans

le monde réel. C'est une approche sociale que préconise le réseau ALPINE (ALPN, Site web).

Dans un environnement aussi hétérogène que l'Internet, les peers disposant d'une faible connexion sont souvent des goulets d'étranglement. Pour contrer ce problème, des profils de connexions peuvent être intégrés au système. Un peer disposant d'une connexion à grande bande passante est favorisé pour fournir la ressource. Les peers disposant d'une connexion plus faible sont alors repoussés en bordure du réseau, évitant ainsi sa congestion, tout en bénéficiant d'une qualité de service similaire. Une extension de ce principe a été introduite par clip2 sur le réseau Gnutella. Il s'agit d'utiliser les peers disposant d'une grande bande passante comme proxies pour les peers plus lents.

Certains réseaux comme Freenet (Freenet Project, Site web), utilisent des identifiants obtenus par hashage de chaînes plutôt que des méta-données. L'utilisation de catalogues et de méta- index permet d'augmenter la pertinence et la rapidité des requêtes. Un projet nommé Espra (Espra, Site web) stocke des catalogues à l'intérieur de Freenet. Le catalogue fournit une description pour chaque identifiant (hash key). Mais ces méta-index restent difficiles à maintenir sur l'ensemble du réseau.

D'autres optimisations portent sur la diminution de la taille des paquets internes au protocole, notamment par l'utilisation d'UDP. Ces paquets forment l'essentiel du trafic dans des protocoles de flood broadcast comme Gnutella. Rendre le protocole de communication compact et léger diminue la bande passante gaspillée. Les systèmes d'échange de fichier sont particulièrement concernés par la taille des paquets puisque des gigaoctets de données sont transportées en permanence sur le réseau. Aussi, une variante, notamment utilisée par Mojo Nation la technologie SwarmCast de OpenCola (Open Cola, Site web), consiste à découper les fichiers partagés en petits morceaux. Le système permet alors d'agréger la bande passante de nombreux peers pour transférer ces portions de fichier. Le résultat est une grande rapidité de téléchargement et une fluidification des points de congestion. Permettre à chaque peer de contrôler les peers avec qui il communique, en terme de bande passante consommée, et d'utilisation du réseau fournit la possibilité de résister aux abus de réseau, d'allouer de la bande passante en fonction des préférences de l'utilisateur, et aussi d'optimiser le « processus de découverte du réseau ».

5.5 Les applications

Le poids d'une technologie ou d'un paradigme peut se mesurer par rapport au nombre d'applications y faisant appel. Nous donnons un aperçu des idées qui ont cours autour du peer-to-peer. Nous verrons que bien que le concept date, les communautés d'intérêt l'ont redécouvert récemment. Certains spécialistes y voient désormais un grand potentie l qui pourrait bien redéfinir l'usage fonctionnel des réseaux.

5.5.1 Le calcul distribué

Les communautés scientifiques ont toujours nécessité de grandes capacités de calcul pour leurs travaux de recherche. Citons, par exemple, la recherche des nombres premiers de Mersenne. Les solutions proposées sont généralement d'énormes supercalculateurs tel que l'IBM ASCI White fournissant une capacité de calcul de 12 téraflops. En comparaison, un simple PC à 1 gigahertz fourni mois d'un gigaflop, soit mille fois moins, ces calculateurs sont rares et non destinés aux départements de recherche modestes ou peu populaires.

SETI@Home

L'un des premiers projets scientifique à bénéficier du calcul distribué par les machines personnelles des internautes a été le projet SETI (Search for Extraterrestrial Intelligence).

Depuis, de nombreuses initiatives internationales ont vu le jour, notamment la NASA qui lança en 1992 un programme SETI intensif d'écoute d'émission radios dans l'espace dans l'espoir de trouver des séquences redondantes non aléatoires, signe d'une émission « intelligente ». Ce programme a été abandonné un an plus tard. Mais de nombreux enthousiastes n'ont pas voulu en rester là et en 1997 le projet SETI@Home (SETI@HOME, Site web) a vu le jour à l'UC Berkeley. L'initiative consiste à utiliser les capacités de calculs des machines personnelles volontaires pour analyser les énormes quantités de données radio enregistré par le télescope Arecibo à Porto Rico.

5.5.2 Le corporate peer-to-peer

Le peer-to-peer ne se limite pas aux applications de partage de fichiers. La possibilité de se dégager des coûts engendrés par les infrastructures centralisées actuelles a de quoi intéresser les entreprises.

Le travail collaboratif

Une des applications les plus en vogues dans le domaine du corporate peer-to-peer est Groove, de la société Groove Networks (Groove Networks, Site web). Ce logiciel permet à ses utilisateurs de collaborer en temps réel depuis n'importe où. Groove offre de nombreux services comme la messagerie instantanée, la conférence, le partage de documents, la navigatio n simultanée, des outils de planification de tâche et de gestion de projet. Mais la liste est longue et d'autres extensions sont en cours de développement afin de couvrir un large éventail d'activités professionnelles.

L'objectif est de diminuer les coûts de communication interne, spécialement dans le cas où les collaborateurs sont disséminés à travers le monde. Groove peut travailler de manière complètement décentralisée, mais généralement ce mode de fonctionnement ne convient qu'aux environnements personnels, ou aux petits réseaux d'entreprise dans lesquels le mécanisme de découverte des noeuds par broadcast reste efficace. Plus généralement, Groove fait appel à une variété de services centralisés transparents pour l'utilisateur.

De plus, Groove est basé sur des technologies XML comme XML-RPC ou SOAP (Simple Object Access Protocol). Cette caractéristique le rend parfaitement compatible avec les web services et Mircosoft l'a bien compris puisque la firme de Redmond a récemment acquis près de 20% du capital de Groove Networks. La plateforme .NET de Microsoft supporte aussi le protocole, ce qui rend l'intégration de nouveaux services parfaitement flexibles. Groove adopte COM comme modèle de composants ce qui le rend compatible avec la plateforme windows. Les utilisateurs de cette plateforme pourront donc facilement intégrer Groove dans leur environnement de travail sans que cela ne nécessite d'utiliser des outils spécifiques. Groove fournit simplement une plateforme d'échange peer-to-peer destinée à faciliter les échanges entre collaborateurs..

Conclusion

Issu d'une volonté d'accessibilité équitable à l'information et à la connaissance, l'Internet a rapidement démontré son potentiel économique. Pour répondre à une logique marchande, les services web n'ont cessé de progresser en s'appuyant sur le modèle client/serveur.

En intégrant les fonctions de client et de serveur sur le même poste, le peer-to-peer permet à tous de tirer partie des ressources globales du réseau. Ce concept, redécouvert récemment, a pourtant connu des applications bien avant Napster, sans qu'il ne soit clairement identifié. Aujourd'hui encore, les avis divergent pour distinguer ce qui est de ce qui n'est pas peer-to-peer. Néanmoins, les experts s'accordent à dire qu'il n'existe pas une architecture unique mais plusieurs, répondant chacune à des problématiques différentes.

Et pour ces raisons ; l'architecture de décentralisé de notre approche est fortement inspirer des réseaux P2P, et bien sur ça dynamicité et flexibilité qui donnent au web sémantique toutes ces pouvoirs.

Références

ABERER .K & HAUSWIRTH .M. "Peer-to-Peer Information Systems: Concepts and Models, state-of-the-art, and Future Systems", *Tutorial IEEE ICDE*, 2002.

ERICSON .J. "Peer to Peer is Here", *Line56 News*, juillet 2001.

EYTAN A & BERNARDO A.H, "Free Riding on Gnutella", FIRSTMONDAY.DK *http://www.firstmonday.dk/issues/issue5_10/adar/* ,2002.

GRIBBLE .S, HALEVY .A, IVES .Z, RODRIG .M, et SUCIU .D. "What can databases do for peer-to-peer? "
WebDB, Workshop on Databases and the Web, June 2001.

KANTERE .V. "A rule mechanism for peer-to-peer data management". *M.Sc. thesis, University of*
Toronto, Canada, 2002.

KEMENTSIETSIDIS .A, ARENAS .M, et MILLER .R. "Data mapping in peer-to-peer systems". *Technical*
Report CSRG-456, University of Toronto, July 2002.

NEJDL .W et al. "EDUTELLA: A P2P Networking Infrastructure based on RDF". In *Proceedings of WWW11*, May 2002, Hawaii.

YANG .B & GARCIA -MOLINA .H, "Designing a Super-Peer Network", *Proc. ICDE Conf.*, 2003

SWEENY .J, HAYWARD .S, DRAKOS .N et BATCHELDER .R, "The five Peer to Peer models : Toward
the new Web", *Gartner, COM-12-4447*, février 2001.

SITE WEB

ALPN, Adaptive Large-scale Peer2peer Information Networking,
http://cubicmetercrystal.com/alpine/

DRDLPBN,Decentralized Ressource Discovery in Large Peer Based Networks,
http://cubicmetercrystal.com/alpine/discovery.html

FastTrack, *http://www.fasttrack.nu*

Freenet Project, *http://freenetproject.org*

Espra, *http://www.espra.net*

Open Cola, *http://www.opencola.com*

SETI@HOME, *http://setiathome.ssl.berkeley.edu/*

Groove Networks, *http://www.groove.net*
Mojo Nation, *http://www.mojonation.net*

Chapitre 6

Description du système proposé

6.1. Introduction

L'approche de médiation que nous présentons ici est fortement inspiré de l'architecture pair à pair (P2P) (Parameswaran et al., 2001) que nous prévoyons pour le web sémantique.
Dans cette section, il sera question de montrer le rapport qu'il y a entre le web sémantique et cette architecture P2P.

Si on examine l'architecture du world wide web actuelle, on distinguera deux architectures : client/serveur et l'architecture p2p (Peer-to-peer). A l'échelle du World Wide Web n'importe quelle forme de centralisation va immédiatement créer un goulot d'étranglement au niveau des sorties et de la capacité des serveurs. Pour ces raisons, l'architecture décentralisée des serveurs est encore plus forte dans le web sémantique.
Dans ce contexte, il semble très peu raisonnable de parler d'un schéma global. La raison conceptuelle principale de rejeter cette notion est que nous ne pouvons pas penser à un ensemble de bases de données de P2P, juste comme une implémentation particulière d'une source de données (virtuelle) simple, ça sera la prétention fondamentale qui motive la définition d'un schéma global. D'un point de vue fondamental, toute théorie développée dans la supposition d'un schéma global, et que le schéma global est fixe, nous empêche d'étudier la dynamicité d'un réseau de P2P.

6.2. Approche proposée

Notre approche s'appuiera nécessairement sur de multiples systèmes de médiation. Cette approche décentralisée consistant à considérer une coalition de serveurs d'information, chaque serveur jouant indifféremment le rôle de serveurs de données ou de médiateurs avec ses pairs, et participant de manière distribuée et collective au traitement des requêtes des utilisateurs. Une telle architecture sera plus adaptée grâce à sa flexibilité.
Dans ce contexte de médiation décentralisée et afin de décrire le système proposé de façon plus détaillée, nous suggérons ce schéma qui représente ses différents pairs.

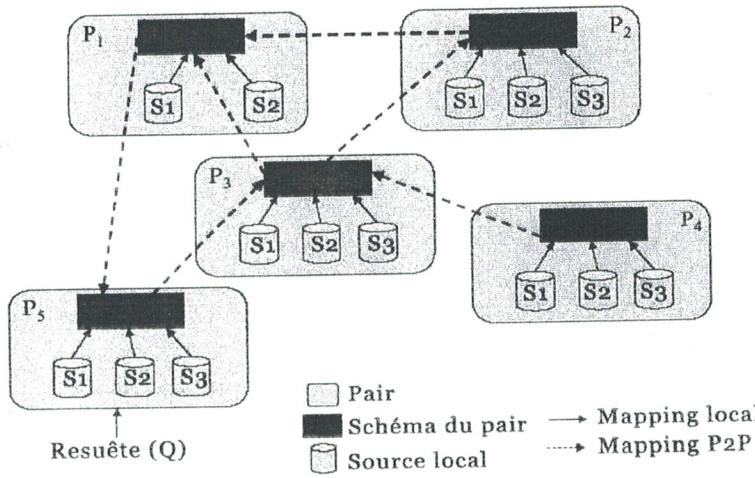


Figure 6.1 Architecture du système

Avec :
P(i) : les Super-pairs ou bien les médiateurs,

S(j) : les sources de données,

Maintenant on va détailler la structure de chaque Pair :

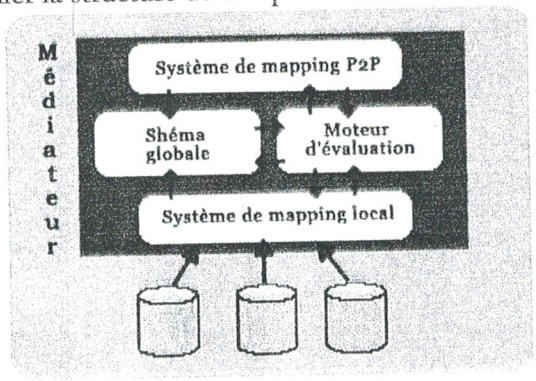


Figure 6.2 Structure d'un pair du système

Dans notre système, les différentes vues sémantiques sont exprimées par des ontologies, qui seront fourni par plusieurs paires dans un large réseau de paires, chaque un des pairs emploie son propre local ontologie, et le schéma global sa sera calculer a travers des règles de mapping.

Notre travail consiste à concevoir un système de mapping tout en assurant une représentation de l'information des SIG dans un format qui s'inscrit dans l'activité Semantic Web du W3C. Pour comprendre en quoi consiste un système de Mapping, on peut dire que c'est la solution au problème suivant : à partir de deux schémas sources et un schéma intégré, comment ce dernier lors du traitement d'une requête posé par l'utilisateur, peut-il savoir l'appartenance des informations demandées à telle ou telle autre source de donnée ?

Parmi les travaux déjà faits sur le mapping, on peut citer, (Miller et al, 2000), (Yan et al, 2001), où on considère la création du mapping seulement entre des schémas relationnels et ignorent les contraintes qui lient les attributs entre les différents schémas. Une amélioration de ces derniers est dans les travaux du projet Clio (Miller et al, 2001), (Popa et al, 2002a), (Popa et al, 2002b), (Popa et al, 2002c) qui consiste en un outil de mapping semi-automatique, développé par le centre de recherche d'Almaden d'IBM. Cet outil permet le mapping depuis et vers des schémas relationnels, et prend en considération les contraintes qui lient les attributs dans les schémas sources et schéma global. Mais, ce système n'effectue le mapping, que s'il dispose auparavant de correspondances entre les attributs des schémas sources et ceux du schéma global. Ses règles sont exprimées en langage logique de premier ordre et sont dépendantes du modèle des schémas source et du schéma global.

Le système de mapping que nous devons concevoir, est indépendant du modèle de représentation des schémas sources et de schéma global. Nous nous basons pour cela, sur le concept d'ontologie. Nous verrons comment ces règles permettent d'adresser touts les objets du schéma global à partir des schémas sources et comment elles facilitent le processus de réécriture des requêtes. Ensuite, nous abordons l'implémentation d'un moteur d'évaluation des règles de mapping.

6.3. Conception du système de mapping

6.3.2 L'utilisation des ontologies

Le problème qui se pose maintenant, est comment peut-on préserver cette information sémantique dans le schéma global? et comment définir celle-ci d'une façon formelle, et qui sera exploitable tant par des humains que par des programmes ?

C'est dans le concept d'ontologie que ce problème trouve sa solution. En effet, une ontologie est composée de concepts et de relations (rôles) effectivement utilisés pour modéliser un domaine donné. Ainsi, on peut exprimer des relations de généralisation/spécialisation (qu'on note "is-a" ou "isa"). Par exemple, la relation "Noeud isa Extrémité" correspond à la règle "BD1.Noeud _ BD2.Extrémité".

Définition Une ontologie est un quintuplet O = (C, R, Source, Dest, isa), où :

- C est un ensemble de concepts ;
- R est un ensemble de rôles ;
- source et dest font correspondre pour chaque rôle, un concept début et un concept fin dans C, respectivement ;
- isa est une relation d'héritage entre les concepts dans C.

Propriété Nous considérons les propriétés suivantes :

- L'ontologie est symétrique : chaque rôle r ∈ R a un rôle inverse, noté r- ∈R (sont exprimés entre parenthèses, voir figure 1 et 2). Ainsi, dest(r-) = source(r), et source(r-) = dest(r).
- Chaque instance d'un concept de source(r) peut être reliée à 0 ou plusieurs instances de concept de dest(r) par le rôle r.
- La relation isa est interprétée comme une relation de sous-ensembles (ou généralisation/ spécialisation) et d'héritage.

Nous avons choisit à titre d'illustration, de représenter deux bases de données sources par des ontologies. Ainsi, O1 représente les connaissances de la base de données BD1, et O2 représente les connaissances de la base de données BD2. tel que :

O_1 = (C1;R1; Source1;Dest1; isa1) ;
C1 = {Tronçon, geom_T, numéro_route, nb_voies_direct, nb_voies_indirect, Nœud ,geom_N, types_nœud} ;
R1 = { T a une géométrie, T appartient à la route n_, T a nb_voies_direct, T a nb_voies_indirect, N a une géométrie, N a un type, T débute par, T fini par} ;

On considère que le schéma intégré BDSI est représenté par l'ontologie OSI , tel que :
O_{SI} =(C_{SI} ;R_{SI} ; $Source_{SI}$;$Dest_{SI}$; isa_{SI} ; $part_{SI}$) ; tel que :

- C_{SI} = C1 ∪ C2
- R_{SI} = R1 ∪ R2
- $Source_{SI}$ = Source1 ∪ Source2
- $Dest_{SI}$ = Dest1 ∪ Dest2

Lors de la résolution des conflits de fragmentation, on a vu qu'un Noeud de BD1 est une Extrémité dans BD2. Donc, dans l'ontologie OSI , on illustre cela par la relation isa. Ainsi, on a :
La relation Noeud isa Extrémité correspond à la règle : BD1.Noeud ⊂ BD2.Extrémité .
La relation Noeud.(type_noeud ="tunnel") isa Entrée_tunnel, correspond à la relation SELECTION BD1.Noeud(type_noeud ="tunnel") ⊂ BD2.Entrée_tunnel.

En considérant la relation isa précédente, on a la relation :
Noeud.geom isa Entrée_tunnel.geom, qui correspond à la relation
BD2.Entrée_tunnel.geom INSIDE BUFFER (BD1.Noeud.geom, résolutionBD2).

On retrouve aussi les deux dernières relations isa si on prend Carrefour et Péage comme types de Nœud.

isa_{SI} = isa1 ∪ isa2 ∪ { Noeud isa Extrémité, Noeud.(type_noeud ="tunnel") isa Entrée_tunnel, Noeud.geom isa Entrée_tunnel.geom, Noeud.(type_noeud "carrefour") isa carrefour, Noeud.geom isa carrefour.geom, Noeud.(type noeud ="Péage") isa Péage, Noeud.geom isa Péage}

Nous allons voir maintenant quelques concepts liés aux ontologies, et qui nous seront utiles dans la conception des règles de mapping.

Définition : Soient c, ć deux concepts, on dit que c et ć sont isa-reliés si l'une des conditions suivantes est vérifiée : c = ć, c isa ć ou ć isa c.

Chemin de rôle : Un chemin de rôle de longueur n ($n \geq 1$) est une séquence r = ri…rn, où ri sont des rôles, tel que pour tout $1 \leq i < n$, dest(ri) et source(ri+1) sont isa-reliés.
la source et la destination d'un chemin de rôle sont définis par la source et la destination de leur extrémités : source(r) = source(r1) et dest(r) = dest(rn).

Par exemple, "T.fini_par.a_une_géométrie" défini un chemin de rôle entre le concept Tronçon et le concept Geom_N.

Chemin de concept : Un chemin de concept p est soit de le forme c, ou bien une séquence c.r, où c est un concept et r est un chemin de rôle, tel que source(r) et c sont isa-reliés.

La longueur de p est 0 dans le premier cas, et la longueur du chemin de rôle r dans le deuxième cas. La source et la destination du chemin de concept c est c lui même. La source et la destination de p = c.r sont définis tel que : source(p) = c et dest(p) = dest(r). Par exemple, P=Chaussée.débute_par_une.a_une_géométrie, est un chemin de concept, tel que c=Chaussée et r=débute_par_une.a_une_géométrie. La source de p est Chaussée et la destination est Geom_E.

6.3.3 Règles de mapping

Lorsque l'utilisateur pose une requête sur le schéma intégré, cette requête ne peut être satisfaite que lorsqu'il existe un système de mapping, qui fait la correspondance entre les structures de données qui existent dans les différentes bases de données source

(dont l'intégration donne le schéma intégré) d'une part, avec la structure des données dans le schéma intégré de l'autre part. Voir fig. 6.3.

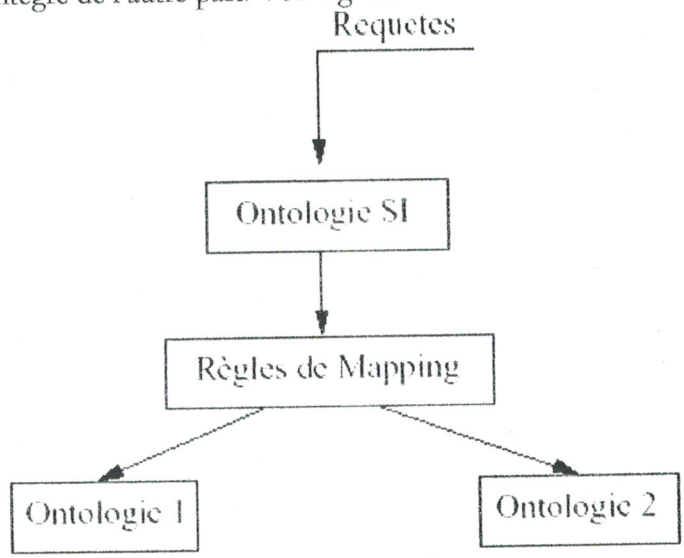


Figure 6.3 Position du système de mapping

Nous allons voir maintenant la description formelle des règles de mapping que nous nous sommes inspiré de (Bernd et al,2001) .

Description formelle des règles de Mapping

Soit V un ensemble de variables, et O un ensemble de racines d'ontologies sources. Une règle de Mapping est une expression de la forme :

R : a/q [$\rightsquigarrow$ v] $\leftarrow$ p, où :

– R est le nom de la règle,

– a $\notin$ V $\cup$ O, la racine de la règle, est soit une variable ou bien la racine d'une ontologie source.

– q est un chemin de concept dans une ontologie source,

– [$\rightsquigarrow$v] est une agrégation (optionnel) par v, où v $\in$ V est une variable,

– p est un chemin de schéma : plus précisément, p est un chemin de rôle si a est une variable et un chemin de concept dans le cas où a est une racine d'ontologie.

Une règle de mapping R est appelé règle relative si sa racine est une variable v. Elle est appelé règle absolue dans le cas contraire.

Concaténation des règles de mapping

Supposons maintenant, qu'un utilisateur s'interroge sur la géométrie d'un Noeud qui est un début d'un Tronçon donné. Cela s'exprime par le chemin conceptuel OSI. Tronçon.débute_par_un.a_une_géométrie. Or, ce chemin est une concaténation des chemins de schéma suivants : OSI.Tronçon, OSI.T.débute_par_un et OSI.N.a_une_géométrie. Puisque cela est un chemin de schéma, existant dans l'ontologie OSI , alors il existe un chemin de source qui lui correspond.
Ce chemin est : O1/Tronçon/Noeud début/Geom_N, qui n'est autre que la concaténation des chemins sources qui correspondent au différents chemins de schémas vus ci-dessus.

Ainsi, on peut voir la règle : Rc1 = R1.R12.R7 :

O1/Tronçon/Noeud début/Geom_N ← OSI.
Tronçon.débute_par_un.a_une_géométrie.
Comme une concaténation des règles R1, R12 et R7 successivement.

Nous allons voir maintenant quelles sont les conditions qui s'appliquent pour qu'une concaténation de règles puisse être possible.
Etant donnée deux règles R et Ŕ tel que :

R : a/q1 ⇝ v1 ← p1

Ŕ : v1/q2 [⇝ v2] ← p2

peuvent être concaténés, si :

- La racine de Ŕ est la variable d'agrégation dans R
- p2 est un chemin de rôle.
- dest(p1) et source(p2) sont isa-reliés.

Le résultat de la concaténation est la règle :

Rc = R. Ŕ: a/q1/q2 [⇝ v2] ← p1.p2

6.3.4 Cas d'utilisations des règles de mapping

Nous allons voir maintenant à travers un exemple, comment les règles de mapping peuvent être exploitées pour la réécriture des requêtes posées par l'utilisateur. Cela ne constitue qu'une étape intermédiaire dans le processus de réécriture.
Supposons que l'utilisateur s'interroge sur le nombre de voies de la chaussée qui appartient au tronçon dont le numéro de route est "RN1". La requête est illustrée en figure 5. Nous avons choisit SQL comme langage de requêtes juste pour illustrer l'exemple. Nous considérons que cela n'intervient pas dans le choix du modèle de représentation des bases de données en question.

```
select   c.nb_voies
from     Chaussée c, Tronçon t
where    c.numéro_route = t.numéro_route AND
         t.numéro_route = "RN1"
```

Figure 6.4 Requête r_1

Pour pouvoir répondre à la requête de l'utilisateur, le système de réécriture à besoin de réécrire la requête originale en une ou plusieurs requêtes. A une étape donnée, les requêtes résultantes sont des requêtes, qui utilise les chemins de sources définis dans les règles de mapping.
Le système de réécriture a la charge de savoir, à partir de la requête originale, quels sont les chemins de schéma qui correspond aux chemins décrit dans la requête. Par exemple, la requête r1 dans la figure 6.4, est composé de cinq chemins de schéma : Tronçon, Chaussé, c.a_nb_voies, c.appartient_à_la_route_n° et t.appartient_à_la_route_n°. En remplaçons les chemins de schéma par les chemins de source correspondant, nous obtenons la requête réécrit illustré en figure 6.5.

```
select   C/nb_voies
from     O2/Chaussée C, O1/Tronçon T
where    C/numéro_route = T/numéro_route AND
         T/numéro_route = "RN1"
```

Figure 6.5 Requête r_2

6.4. Moteur dévaluation des règles de mapping

Pour pouvoir implémenter un moteur d'évaluation des règles de mapping, le choix d'un langage de représentation pour la modélisation d'ontologie, ainsi que pour les règles de mapping, parait indispensable.

6.3.1 Langage de représentation d'ontologie

Il existe plusieurs modèles et langages de représentation pour la modélisation d'ontologie (Chap. 4), Nous avons choisit de représenter les ontologies O1, O2 et OSI avec DAML+OIL, et cela pour toutes les fonctionnalités sémantiques que permet DAML+OIL et qui sont plus riche que celle d'RDF. En effet, DAML+OIL permet :

- Définition de (intersection, union, complément, ...) sur les classes.
- Définition de nouveau types de données avec XML schéma
- Différents types de propriétés, sous-propriétés, ...
- Restrictions des propriétés (cardinalités, restrictions de valeurs, ...)
- Logique de description (and, or, not, rôle transitif, rôle inverse, ...)

Nous donnons maintenant un aperçu sur l'utilisation du DAML+OIL, pour la définition de OSI . La définition complète étant décrite en Annexe (Annexe 1).

- Définition de Tronçon comme sous classe de BDSI :

```
<daml:Class rdf:ID="Tronçon">
     <rdfs:label>Tronçon</rdfs:label>
     <rdfs:subClassOf rdf:resource="#BD_SI"/>
</daml:Class>
```

- Un Noeud est équivalent à une Extrémité (Noeud isa Extrémité) :

```
<daml:Class rdf:ID="Noeud">
     <daml:sameClassAs rdf:resource="#Extremité">
</daml:Class>
```

6.4.2 Représentation des règles de mapping

Pour pouvoir exploiter les règles de mapping par des programmes, notamment le moteur d'évaluation des règles, il faut trouver un language de représentation pour ces règles.
Nous avons choisit de représenter ces règles avec DAML+OIL. ce choix a été fait pour les raisons suivantes :

- DAML+OIL permet d'exprimer certaines types de règles.
- Les règles de mapping en question sont décrites par des concepts et relations d'ontologies.
- Le langage de représentation d'ontologies que nous avons choisi est DAML+OIL.

Nous allons voir maintenant, comment on peut représenter les règles de mapping avec DAML+OIL :
En analysons les règles, on peut voir qu'elles sont décrites de la manière suivante : (chemin de schéma) → (chemin de source), où la flèche signifie que le chemin de schéma correspond (ou se map vers) le chemin de source. Ainsi, on peut voir cela comme (Objet) "propriété" (Objet) dans le langage de DAML+OIL, tel que " propriété " est "se_map_vers".

commençons par définir la propriété (ou la relation) "se_map_vers" :

```
<daml:ObjectProperty rdf:ID="se_map_vers">
     <rdfs:domain rdf:resource="#BD_SI"/>
     <rdfs:range rdf:resource="#BD1"/>
     <rdfs:range rdf:resource="#BD2"/>
</daml:ObjectProperty>
```

"domain" de la propriétée est BD_SI, cela signifie que la relation "se_map_vers" est applicables sur l'objet BD SI.
"range" de la propriétée est une conjonction de BD1 et BD2. cela signifie que la propriété prend ces valeurs seulement dans BD1 et BD2.
Prenons par exemple, la règle R1 des règles de mapping, qui spécifie que les tronçons du schéma global (noté ici "Tronçon_SI") correspond aux tronçons de la base BD1 (noté "Tronçon_source").
Définition de Tronçon du BD 1 :

```
<daml:Class rdf:ID="Tronçon_source">
     <rdfs:subClassOf rdf:resource="#BD_1"/>
     <daml:disjointWith rdf:resource="#Noeud_source"/>
</daml:Class>
```

Définition de Tronçon dans ontologie Osi :

```
<daml:Class rdf:ID="Tronçon_SI">
     <rdfs:subClassOf rdf:resource="#BD_SI"/>
     <daml:disjointWith rdf:resource="#Noeud_SI"/>
</daml:Class>
```

Les deux définitions qu'on vient de voir, doivent être au préalable décrite dans la définition de BD_1 et BD_SI, respectivement. Elles ne sont décrites ici que pour expliquer la traduction de R1.

Maintenant, nous verrons comment traduire la relation "se_map_vers" entre les deux objets Tronçon_SI et Tronçon_source :

```
<daml:Class rdf:about="#Tronçon_SI">
     <rdfs:subClassOf>
          <daml:Restriction>
               <daml:onProperty rdf:resource="#se_map_vers"/>
               <daml:toClass rdf:resource="#Tronçon_source"/>
          </daml:Restriction>
     </rdfs:subClassOf>
</daml:Class>
```

Cette définition spécifie que tout élément qui est dans la classe Tronçon_SI, qui est relié par la relation "se_map_vers" est un élément de la classe Tronçon_source.
Avec la description que nous venons de voir, celle-ci englobe toutes les règles de mapping où la racine du chemin source d'une règle est Tronçon_source, c'est à dire les règles R2, R3, R4, R5, R12, R13. Pour représenter les autres règles, on fait de même avec les Noeud, Chaussée,...etc.

Conclusion

Dans ce chapitre, nous avons présenté un système de mapping qui capture les correspondances entre le schéma global et les schémas sources. Ce système est conçu de manière indépendante du modèle de représentation des connaissances. En effet, ses règles sont exprimées en fonction de concepts et de relations. Nous avons utilisé pour cela, les ontologies comme modèle de représentation des bases de données au niveau conceptuel. Un exemple a été illustré pour montrer comment utiliser les règles de mapping dans le processus de réécriture de requêtes. Aussi, nous avons présenté comment représenter les informations sémantiques tant au niveau conceptuel qu'au niveau de langage de représentation.
Nous avons également utilisé le langage de représentation des ontologies et des règles de mapping qui est le DAML+OIL, ainsi qu'un couplage du moteur d'inférence JESS avec DAML API. Ceci est nécessaire pour permettre l'implémentation d'un moteur d'évaluation des règles de mapping. La réalisation de ce moteur, faisait partie de nos ambitions initiales, ceci n'a malheureusement pas pu être fait pour des contraintes de temps.

Références

BERND .A, BEERI .C, FUNDULAKI .I, SCHOLL .M et VERCOUSTRE .A-M. "Rewriting Tree Queries into XPath Expressions". In *Proceedings Bases de Données Avancées,* 2001.

BOLEY .H, TABET .S ET WAGNER .G. "Design Rationale of RuleMl : A Markup Language for Semantic Web Rules". *International Semantic Web Working Symposium.* 30 July - 1 August 2001.

CONNOLLY .D, HARMELEN .F.V, HORROCKS .I, MCGUINNESS .D.L, STEIN .L.A et LUCENT TECHNOLOGIES. "DAML+OIL (March 2001) reference Description". *W3C* Note 18 décembre 2001. *http ://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218/*

DEVOGELE .T, PARENT .C, SPACCAPIETRA .S. "Intégration de bases de données spatiales" . *Inst. De rech. Ecole Navale, U. de Lausanne, Ecole Poly. Fédér.* de Lausanne. 2000

E-PARCOURS "L'avenir du Web sémantique" .
http ://www.infres.enst.fr/people/saglio/etudes/e-parcours/portails/SemWebintro.pdf

HARMELEN .F.V, MCGUINNESS .D.L, STEIN .L.A, et LUCENT TECHNOLOGIES. "Annotated DAML+OIL Ontology Markup. " *W3C* Note 18 December 2001.
http ://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/

JESS, "the Rule Engine for the JavaTM Platform. " *http://herzberg.ca.sandia.gov/jess/*

JOE .K, "DAMLJESSKB".
http://plan.mcs.drexel.edu/projects/legorobots/design/software/DAMLJessKB/

MEHDI .E. "Intégration des systèmes d'information géographique" *Mémoire de DEA Informatique,* Marseille, 2001.

MILLER .R.J, HASS .L.M, et HERNANDEZ .M.A. "Schema Mapping as Query Discovery." *In Proc. of the International Conf. on Very Large Data Bases (VLDB), pages 77-88, Cairo, Egypt,* September 2000.

MILLER .R.J, HERNANDEZ .M.A, HASS .L.M, YAN .L, HOWARD HO .C.T, FAGIN .R, POPA .L. "The Clio Project : Managing Heterogeneity U. of Toronto, " *IBM Almaden Research Center.*

PARAMESWARAN, M., SURSALA, A., ET WINSTON, A. . "P2p networking: An information sharing alternative". *IEEE Computing, 34(7)* ,2001.

POPA .L, HERNANDEZ .M.A, VELEGRAKIS .Y, MILLER .R.J, NAUMANN .F, et HO .H. " Mapping XML and Relational Schemas with CLIO, " *System Demonstration. In ICDE,* 2002a.

PHILIPPE .L. "Vers le Web sémantique. " *Séminaire Ontosaurus,* Mars 2002.
http ://www.lalic.paris4.sorbonne.fr/stic/tutor/Ontosaurus29032002.pdf

POPA .L, VELEGRAKIS .Y, HERNANDEZ .M, MILLER .R.J et FAGIN .R. "Mapping Generation Data Translation of Heterogeneous Web Data". *Cet article est un résumé de (Popa et al, 2002c)* ,2002b.

POPA .L, VELEGRAKIS .Y, HERNANDEZ .M, MILLER .R.J et FAGIN .R."Translating Web Data" *Technical Report CSRG-441, U. of Toronto, Dept. of CS,* February 2002c. *ftp ://ftp.cs.toronto.edu/cs/ftp/pub/reports/csri/441.*

SOUAD .B. " Etude et réalisation d'un système d'intégration a base d'agents de l'information distribué hétérogène". *Mémoire de DEA MCAO*, Marseille, 2002.

YAN .L, R. MILLER .R.J, HASS .L.M et FAGIN .R. " Understanding and Refinement of Schema Mappings". *In ACM SIGMOD Conference, pages 485-496,* 2001.

Chapitre 7

L'étude de cas

7.1. Introduction

L'intégration de données résidant dans des systèmes hétérogènes pose de nombreux problèmes sémantiques. Par exemple, dans le cas d'intégration de données géographiques, les échelles de représentation des données peuvent être différentes, l'organisation des données peut être différente d'une source à l'autre, etc. Dans ce chapitre, nous nous intéressons plus particulièrement à l'élaboration d'un système de mapping qui capture les correspondances d'intégration, tout en assurant une représentation de l'information sémantique liée aux sources de données, dans un format qui s'inscrit dans l'activité Semantic Web du W3C. Nous traitons aussi, l'implémentation d'un moteur d'évaluation de ces règles de correspondance.

Système d'Information Géographique

Un Système d'Information Géographique (SIG) est un ensemble organisé de matériels informatiques, de logiciels, de données géographiques et de personnel capable de saisir, stocker, mettre à jour, manipuler, analyser et présenter toutes formes d'informations géographiquement référencées.

La technologie du SIG possède toutes les fonctionnalités du gestionnaire de base de données comme par exemple l'analyse statistique ou les requêtes, mais qui s'appliquent dans un environnement géoréférencé pour l'analyse et la visualisation.

Ces caractéristiques distinguent le SIG des autres systèmes d'analyse et offrent de nombreuses perspectives d'application. La majorité des problèmes auxquels l'humanité se trouve confrontée - surpopulation, pollution, déforestation, risques naturels - possède une dimension géographique essentielle. De même, l'implantation industrielle, la localisation du meilleur site pour une production agricole ou la définition d'itinéraires pour l'organisation des interventions de la sécurité civile sont des problèmes qui peuvent être traités à l'aide de la technologie SIG. Le SIG va produire des cartes, intégrer les informations, visualiser des scénarios, résoudre des problèmes complexes et proposer des solutions pertinentes.

7.2. Conception du système de mapping

7.2.1 Exemple de base de donnée cartographique

Pour donner un aperçu du problème dans le cas des SIG, nous nous sommes inspirés de l'exemple sur les bases de données cartographiques de l'institut Géographique National (Devogele et al, 2000). Cet exemple comporte deux sources BD1 à échelles 1/250 000 et BD2 à échelles 1/10 000 représentants le même réseau routier sur la même zone géographique.

La source BD1 (Annexe 3) contient un ensemble de tronçons. Un nouveau tronçon est créé à chaque fois qu'il y a un changement du nombre de voies dans une route. Chaque tronçon contient un nœud de début et un nœud de fin. Ces nœuds décrivent les points où les conditions de trafic changent. Un nœud peut être de type carrefour, rond-point, péage, extrémité d'une impasse ou un point où la valeur d'au moins un attribut de la route change.

La source BD2 (Annexe 3) contient un ensemble de chaussées composées par une ou plusieurs voies contiguës de même direction. Dans le cas où un séparateur divise une ou plusieurs voies de même direction, chaque groupe de voies serait représenté par une chaussée. Une instance de l'objet séparateur est alors créée pour séparer les deux chaussées. L'attribut position d'une chaussée définit sa position par rapport au sol : surface ou tunnel. Chaque chaussée admet deux extrémités qui peuvent être de type Carrefour, Péage ou Entrée Tunnel.

Pour intégrer ces deux sources de données, une identification des corrélations au niveau des populations et des instances est nécessaire. En effet, il faut arriver à une représentation globale et unifiée qui décrit toutes les informations des deux sources. Ces corrélations sont définies à l'aide des Assertions de Correspondances Inter-schéma (ACI), dont le format général inclut quatre clauses définissant respectivement quelles sont les deux populations en correspondance, quel prédicat lie leurs instances qui se correspondent, si elles ont des attributs en commun et comment leurs géométries se correspondent :

Les populations en correspondance :

- BD1.Noeud $\subset$ BD2.Extrémité
- BD1.Tronçon $\subseteq$ BD2.SET(Chaussée, Séparateur)

Les correspondances d'instances de ces populations (CI) :

SELECTION BD1.Noeud (type_noeud = "carrefour") $\subset$ BD2.Carrefour
– CI BD2.Carrefour.geom INSIDE BUFFER (BD1.Noeud.geom, résolutionBD2)

Où SELECTION est l'opérateur algébrique usuel qui extrait le sous-ensemble des instances répondant à un prédicat donné. L'opérateur INSIDE décrit si un objet est à l'intérieur de l'autre et l'opérateur BUFFER transforme le point représentant le nœud de BD1 en une surface plus ou moins grande en fonction de la résolution de BD2.

Les attributs-communs (AC) :

L'ACI reliant BD1.Tronçon à BD2.Chaussée inclut la clause :
– AC Tronçon.num_route = Chaussée.num_route,
– AC Tronçon.nb_voies_directes = compte_voie({c∈Chaussée/COR (c, Tronçon)}, direction(Tronçon)),
– AC Tronçon.nb_voies_indirectes = compte_voie({c∈Chaussée/COR (c, Tronçon)}, direction_opposée (Tronçon)) ;

où la méthode compte_voie compte le nombre de voies d'un ensemble de chaussées qui sont dans une certaine direction, et l'opérateur booléen COR(x,y) détermine si les deux instances x et y se correspondent.

Géométrie commune (GC) :
La clause GC pour la même ACI est :
– GC Tronçon.geom = MERGE UNSPLIT ({x.geom / x ∈(Chaussée ∪ Séparateur) ∧COR(x,Tronçon)})

Où l'opérateur UNSPLIT regroupe des lignes ayant une extrémité commune en une ligne continue. L'opérateur MERGE transforme des lignes continues, quasi-parallèles, en une ligne.

Par ailleurs, un des problèmes les plus fréquents qui se pose lors du processus d'intégration des données, est celui des conflits de fragmentation. Ceci est du au fait qu'à une instance d'un élément d'une source BD1 peuvent correspondre plusieurs instances d'un élément d'une source BD2. Ce conflit est du type 1:n. La modélisation du conflit se fait à l'aide de deux type de règles :

1. Des règles dites de correspondances qui expriment la correspondance entre les objets en précisant les cardinalités minimum et maximum d'instances des différents objets. BD1.Tronçon $\subseteq$ BD2.SET ([1 :N] Chaussée, [0 :N] Séparateur) , Cette règle précise qu'un tronçon de BD1 peut correspondre à un ou plusieurs Chaussée et Séparateur de la BD2.
2. Des règles de résolution de conflit qui définissent les corrélations entre les instances des éléments.

CI Tronçon.num_route = Chaussée.num_route $\wedge$ BD2.chaussée.geom INSIDE BUFFER (BD1.tronçon.geom, résolutionBD2)

CI BD2.Séparateur.geom INSIDE BUFFER (BD1.tronçon.geom, résolutionBD2) $\wedge$ $\exists$c $\in$ BD2.Chaussée (COR(c,BD1.Tronçon) $\wedge$ sépare (BD2.Séparateur, c))

La première règle spécifie qu'à chaque instance de Tronçon correspond l'ensemble d'instances de Chaussées portant le même numéro de route et dont la géométrie est incluse dans la zone construite autour de la géométrie du Tronçon. La deuxième règle définit les instances de Séparateur dont la géométrie est incluse dans la zone tampon de celle du Tronçon et qui sont reliées à au moins une instance de Chaussée.

7.2.2 L'utilisation des ontologies

Comme nous venons de le voir, lors de l'identification des règles de résolution de conflit de fragmentation et les différents ACI, on s'aperçoit que la principale information sémantique qu'on puisse tirer, c'est l'existence d'objets de BD1 qui correspondent à un ensemble d'objets de BD2. Cela s'explique à cause des échelles géographiques, car les deux bases de données ont des résolutions différentes mais représentent le même réseau routier. Or, le problème qui se pose maintenant, est comment on peut préserver cette information sémantique dans le schéma global ? et comment définir celle-ci d'une façon formelle, et qui sera exploitable tant par des humains que par des programmes ?

C'est dans le concept d'ontologie que ce problème trouve sa solution. En effet, une ontologie est composée de concepts et de relations (rôles) effectivement utilisés pour modéliser un domaine donné. Ainsi, on peut exprimer des relations de généralisation/spécialisation (qu'on note "is-a" ou "isa"). Par exemple, la relation "Noeud isa Extrémité" correspond à la règle "BD1.Noeud _ BD2.Extrémité".

Nous avons choisit pour les raisons évoquées précédemment, de représenter les deux bases de données sources par des ontologies. Ainsi, O1 représente les connaissances de la base de données BD1, et O2 représente les connaissances de la base de données BD2. tel que :

O1 = (C1;R1; Source1;Dest1; isa1) ;
O2 = (C2;R2; Source2;Dest2; isa2) ;

C1 = {Tronçon, geom_T, numéro_route, nb_voies_direct, nb_voies_indirect, Nœud ,geom_N, types_nœud} ;
R1 = { T a une géométrie, T appartient à la route n_, T a nb_voies_direct, T a nb_voies_indirect, N a une géométrie, N a un type, T débute par, T fini par};
C2 = { Chaussé, geom_c, numéro_route, nb_voie, position, Séparateur, geom_s, hauteur, Extrémité, geom_e, Carrefour, geom_ca, Péage, geom_p, toponyme, Entrée tunnel, geom_e} ;

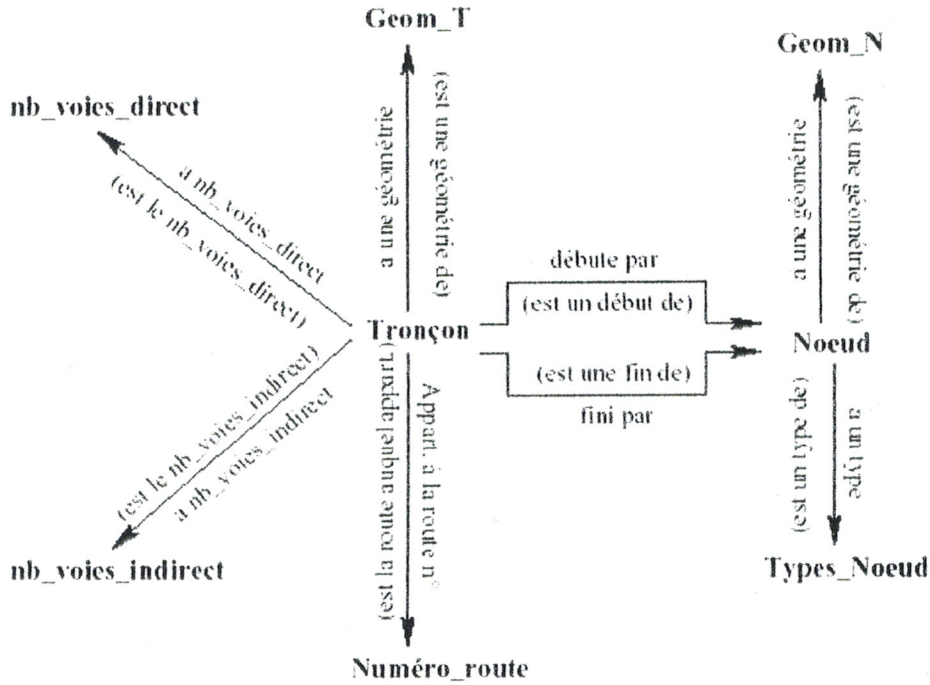


Figure 7.1 Ontologie O1

R2 = {C a une géométrie, C appartient à la route n_, C a nb_voies, C a une position, S a une géométrie, S a une hauteur, E a une géométrie, Ca a une géométrie, P a une géométrie, Et a une géométrie, S Sépare, C débute par, C fini par, E est de type} ;
Source1; Source2;Dest1 et Dest2 sont définit selon la définition vue précédemment.
isa1 = isa2 = ϕ

Il faut noter que les rôles inverses exprimés entre parenthèses, (voir Fig. 7.1 et 7.2) appartiennent aussi à l'ensemble R1 et R2, respectivement. Nous ne les avons pas inclus ici pour plus de clarté. Les règles de mapping expriment les correspondances, qui existent

entre les données du schéma intégré avec ceux des schémas source. Dans ce cas, une représentation du schéma intégré est nécessaire.

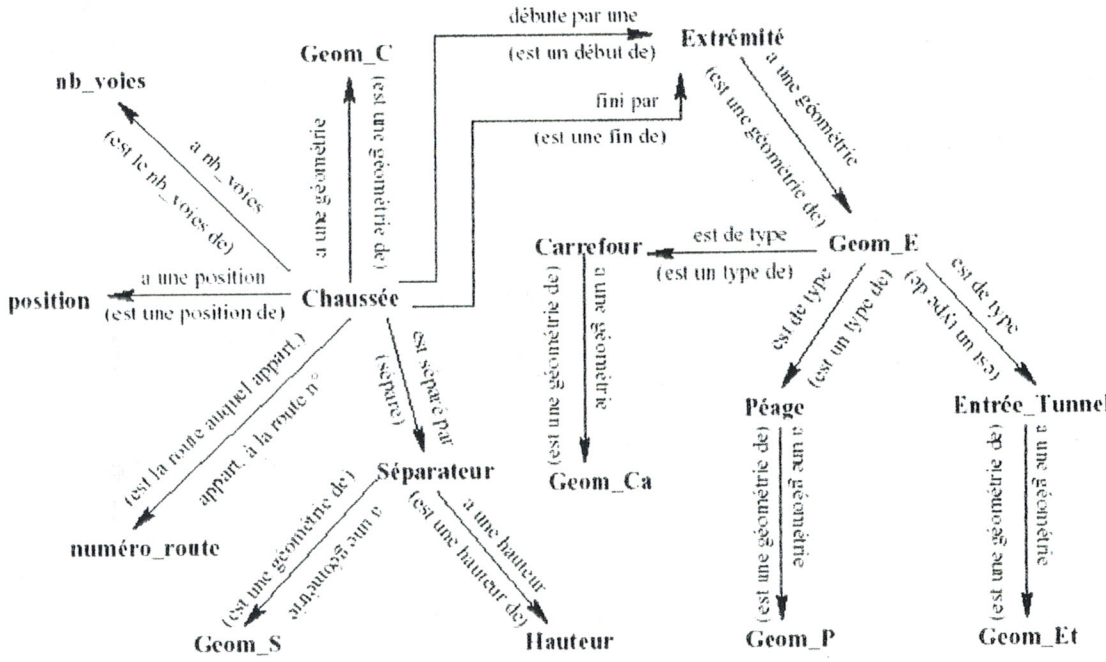


Figure 7.2 Ontologie O_2

On considère que le schéma intégré BDSI est représenté par l'ontologie OSI , tel que :
OSI =(CSI ;RSI ; SourceSI ;DestSI ; isaSI ; partSI) ; tel que :

- CSI = C1 ∪ C2
- RSI = R1 ∪ R2
- SourceSI = Source1 ∪ Source2
- DestSI = Dest1 ∪ Dest2

Lors de la résolution des conflits de fragmentation, on a vus qu'un Noeud de BD1 est une Extrémité dans BD2. Donc, dans l'ontologie OSI , on illustre cela par la relation isa. Ainsi, on a :
La relation Noeud isa Extrémité correspond à la règle : BD1.Noeud ⊂ BD2.Extrémité .
La relation Noeud.(type_noeud ="tunnel") isa Entrée_tunnel, correspond à la relation SELECTION BD1.Noeud(type_noeud ="tunnel") ⊂ BD2.Entrée_tunnel.
En considérant la relation isa précedante, on a la relation :
Noeud.geom isa Entrée_tunnel.geom, qui correspond à la relation
BD2.Entrée_tunnel.geom INSIDE BUFFER (BD1.Noeud.geom, résolutionBD2).

On retrouve aussi les deux dernières relations isa si on prend Carrefour et Péage comme types de Nœud.

isaSI = isa1 ∪ isa2 ∪ { Noeud isa Extrémité, Noeud.(type_noeud ="tunnel") isa Entrée_tunnel, Noeud.geom isa Entrée_tunnel.geom, Noeud.(type_noeud "carrefour")
isa carrefour, Noeud.geom isa carrefour.geom, Noeud.(type noeud ="Péage") isa Péage, Noeud.geom isa Péage}

Une autre règle de résolution des conflits de fragmentation est :
BD1.Tronçon ⊆ BD2.SET ([1 :N] Chaussée, [0 :N] Séparateur) qui fait correspondre à chaque instance de Tronçon de BD1 plusieurs instances des Chaussée et Séparateur de la BD2. Cette relation ne peut être représentée par une relation isa dans l'ontologie OSI. Cependant, on peut la représenter par un lien d'agrégation (appelé aussi lien de composition, ou lien "Part of"), qui en modélisation de données relie un objet à ses objets composants.
Suivant les conventions UML, une agrégation est symbolisée par un losange touchant la classe d'objet agrégat (Devogele et al, 2000), voir fig. 7.3. Ainsi, la relation d'agrégation partSI est :
partSI = Chaussée part-of Tronçon ∪ Séparateur part-of Tronçon

L'ontologie OSI est représenté en figure 6.3. Pour des raisons de clarté dans le schéma qui représente OSI et pour mettre l'accent sur les relations d'agrégation et d'isa, nous avons décidé de ne pas représenter certains concepts et rôles que nous avons déjà détaillés dans les schémas de O1 et O2. Mais ces rôles et concepts sont bien évidemment définis dans l'ontologie OSI .

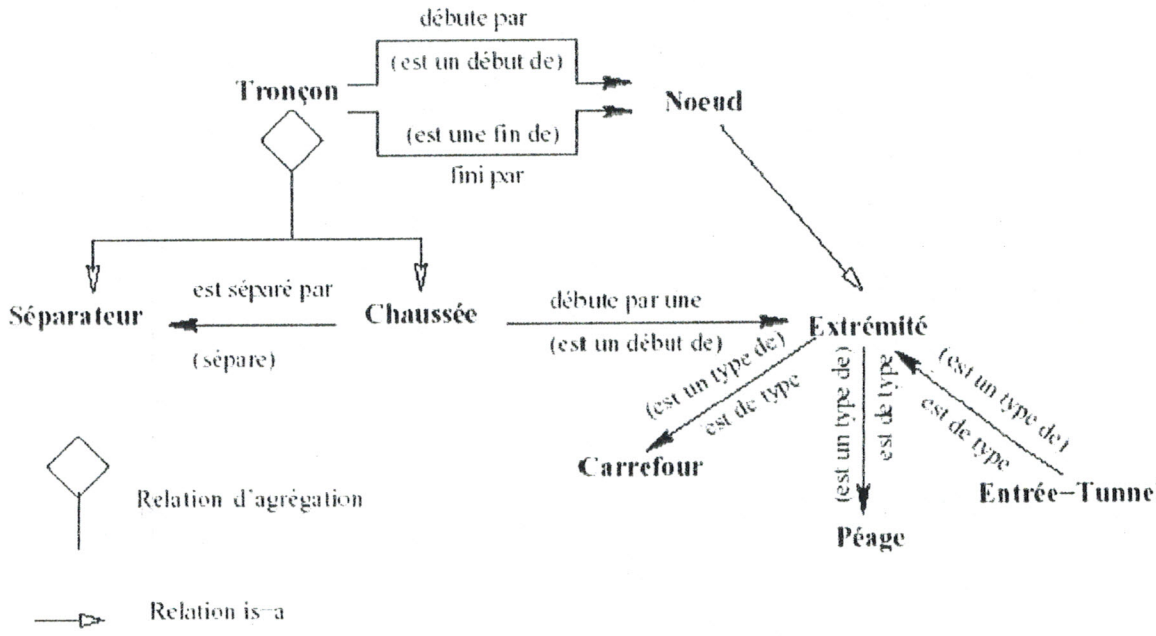


Figure 7.3 L'ontologie OSI

7.2.3 Règles de mapping

Lorsque l'utilisateur pose une requête sur le schéma intégré, cette requête ne peut être satisfaite que lorsqu'il existe un système de mapping, qui fait la correspondance entre les structures de données qui existent dans les différentes bases de données source (dont l'intégration donne le schéma intégré) d'une part, avec la structure des données dans le schéma intégré de l'autre part.

Nous allons voir maintenant les règles de mapping, qui mappent les chemins conceptuels dans l'ontologie OSI avec les chemins conceptuels dans l'ontologie O1 et O2.

Les règles décrites en TAB. 7.1 ne sont qu'une partie de l'ensemble des règles de mapping, entre l'ontologie OSI et les deux ontologies O1 et O2. La totalité des règles de mapping sont décris en Annexe (Annexe 2).
En analysons les règles décris en TAB. 7.1, on remarque qu'une règle de mapping est divisée en deux parties. La partie gauche correspond au chemin de concept dans les ontologies sources, qu'on appelle chemin de source de la règle. Ce chemin est évalué soit à partir de la racine de l'ontologie source, soit à partir d'une variable. La partie droite est le chemin de concept ou chemin de rôles de l'ontologie OSI qu'on appelle chemin de schéma de la règle.

R_1 :	O_1.Tronçon $\rightsquigarrow$ T	—	O_{SI}.Tronçon
R_2 :	T.geom_T	—	O_{SI}.T.a_une_géométrie
R_3 :	T.numéro_route	—	O_{SI}.T.appartient_à_la_route_n°
R_4 :	T.nb_voies_direct	—	O_{SI}.T.a_nb_voies_direct
R_5 :	T.nb_voies_indirect	—	O_{SI}.T.a_nb_voies_indirect
R_6 :	O_1.Noeud $\rightsquigarrow$ N	—	O_{SI}.Noeud
R_7 :	N.geom_N	—	O_{SI}.N.a_une_géométrie
R_8 :	N.types_noeud $\rightsquigarrow$ Tn	—	O_{SI}.N.a_un_type
R_9 :	Tn.Carrefour	—	O_{SI}.N.est_de_type_Carrefour
R_{10} :	Tn.Peage	—	O_{SI}.N.est_de_type_Peage
R_{11} :	Tn.Tunnel	—	O_{SI}.N.est_de_type_Tunnel
R_{12} :	T.Noeud_début $\rightsquigarrow$ N	—	O_{SI}.T.débute_par_un
R_{13} :	T.Noeud_fin $\rightsquigarrow$ N	—	O_{SI}.T.fini_par_un
R_{14} :	N.début_T $\rightsquigarrow$ T	—	O_{SI}.N.est_un_début_de
R_{15} :	N.fin_T $\rightsquigarrow$ T	—	O_{SI}.N.est_une_fin_de

Table 7.1 Règles de mapping

La première règle de mapping spécifie que tous les élements de type Tronçon appartenant à O1 sont agrégés par la variable T et sont des instances du concept Tronçon de l'ontologie OSI. De la même manière, la règle R6 spécifie que tous les éléments de type Noeud appartenant à O2 sont agrégés par la variable N et sont des instances du concept Noeud de l'ontologie OSI.
Les règles de mapping définissent aussi les instances de rôle. Par exemple, la règle R12 crée des instances du rôle "T.débute par un" qui font associer chaque élément Tronçon obtenu par la règle R1 à tout les éléments Noeud qui peuvent être obtenus depuis T en suivant le chemin de concept "Noeud début".

Il faut noter qu'une bonne utilisation des variables aide à réduire la taille des règles de mapping. Pour illustrer cela, considérons les règles R6, R12 et R13 qui sont toutes les trois agrégés par la variable N. Par ces multiples et même agrégations, les règles R7, R8, R14 et R15 peuvent être appliquées à tous les éléments trouvés en utilisant les règles R6, R12 et R13. Par contre, si par exemple seulement R6 été agrégé à une nouvelle variable N1 $\neq$ N, alors les règles R7, R8, R14 et R15 ne peuvent être utilisées pour trouver la géométrie, types_noeud, début_Tronçon et fin_Tronçon pour les éléments trouvés par R6. Pour remédier à cela, nous serions obligés d'ajouter des règles qui sont similaire à R7, R8, R14 et R15, mais qui utilise N1 comme noeud de contexte. Cela augmente considérablement la taille des règles de mapping.

Nous allons voir maintenant la description formelle des règles de mapping que nous nous sommes inspiré de (Bernd et al,2001) .

Concaténation des règles de mapping

Supposons maintenant, qu'un utilisateur s'interroge sur la géométrie d'un Noeud qui est un début d'un Tronçon donné. Cela s'exprime par le chemin conceptuel OSI. Tronçon.débute_par_un.a_une_géométrie. Or, ce chemin est une concaténation des chemins de schéma suivants : OSI.Tronçon, OSI.T.débute_par_un et OSI.N.a_une_géométrie. Puisque cela est un chemin de schéma, existant dans l'ontologie OSI , alors il existe un chemin de source qui lui correspond.
Ce chemin est : O1/Tronçon/Noeud début/Geom_N, qui n'est autre que la concaténation des chemins sources qui correspondent aux différents chemins de schémas vus ci-dessus.
Ainsi, on peut voir la règle : Rc1 = R1.R12.R7 :

O1/Tronçon/Noeud début/Geom_N ← OSI.
Tronçon.débute_par_un.a_une_géométrie.
Comme une concaténation des règles R1, R12 et R7 successivement.

7.3. Moteur dévaluation des règles de mapping

Pour pouvoir implémenter un moteur d'évaluation des règles de mapping, le choix d'un langage de représentation pour la modélisation d'ontologie, ainsi que pour les règles de mapping, parait indispensable.

7.3.1 Langage de représentation d'ontologie

Nous avons choisit de représenter les ontologies O1, O2 et OSI avec DAML+OIL, et cela pour toutes les fonctionnalités sémantiques que permet DAML+OIL et qui sont plus riche que celle d'RDF. En effet, DAML+OIL permet :

- Définition de (intersection, union, complément, ...) sur les classes.
- Définition de nouveau types de données avec XML schéma
- Différents types de propriétés, sous-propriétés, ...
- Restrictions des propriétés (cardinalités, restrictions de valeurs, ...)
- Logique de description (and, or, not, rôle transitif, rôle inverse, ...)

Nous donnons maintenant un aperçu sur l'utilisation du DAML+OIL, pour la définition de OSI . La définition complète étant décrite en Annexe (Annexe 1).
- Affirmer que c'est une ontologie

```
<daml:Ontology rdf:about="">
```

- Définition de Tronçon comme sous classe de BDSI :

```
<daml:Class rdf:ID="Tronçon">
     <rdfs:label>Tronçon</rdfs:label>
     <rdfs:subClassOf rdf:resource="#BD_SI"/>
</daml:Class>
```

- Un Noeud est équivalent à une Extrémité (Noeud isa Extrémité) :

```
<daml:Class rdf:ID="Noeud">
     <daml:sameClassAs rdf:resource="#Extremité">
</daml:Class>
```

- Définir le rôle "sépare" :

```
<daml:ObjectProperty rdf:ID="sépare">
          <rdfs:label>sépare</rdfs:label>
</daml:ObjectProperty>
```

- Un séparateur sépare deux Chaussées :

```
<daml:Class rdf:about="#Separateur">
     <rdfs:subClassOf>
          <daml:Restriction daml:Cardinality="2">
               <daml:onProperty rdf:resource="#sépare"/>
               <daml:hasClassQ rdf:resource="#Chaussée"/>
          </daml:Restriction>
     </rdfs:subClassOf>
</daml:Class>
```

- Tronçon ∈ SET ([1 :N] Chaussée, [0 :N] Séparateur) :

```
<daml:Class rdf:about="#Tronçon">
     <rdfs:subClassOf>
          <daml:Restriction daml:maxCardinality="n">
               <daml:onProperty rdf:resource="#correspond"/>
               <daml:hasClassQ rdf:resource="#Chaussée"/>
          </daml:Restriction>
     </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Tronçon">
     <rdfs:subClassOf>
          <daml:Restriction daml:minCardinality="1">
               <daml:onProperty rdf:resource="#correspond"/>
               <daml:hasClassQ rdf:resource="#Chaussée"/>
          </daml:Restriction>
     </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Tronçon">
     <rdfs:subClassOf>
          <daml:Restriction daml:maxCardinality="n">
               <daml:onProperty rdf:resource="#correspond"/>
               <daml:hasClassQ rdf:resource="#Séparateur"/>
          </daml:Restriction>
     </rdfs:subClassOf>
</daml:Class>
```

7.3.2 Représentation des règles de mapping

Pour pouvoir exploiter les règles de mapping par des programmes, notamment le moteur d'évaluation des règles, il faut trouver un language de représentation pour ces règles.
Nous avons choisit de représenter ces règles avec DAML+OIL. ce choix a été fait pour les raisons suivantes :

- DAML+OIL permet d'exprimer certaines types de règles.
- Les règles de mapping en question sont décrites par des concepts et relations d'ontologies.
- Le langage de représentation d'ontologies que nous avons choisi est DAML+OIL.

Nous allons voir maintenant, comment on peut représenter les règles de mapping avec DAML+OIL :
En analysons les règles, on peut voir qu'elles sont décrites de la manière suivante : (chemin de schéma) → (chemin de source), où la flèche signifie que le chemin de schéma correspond (ou se map vers) le chemin de source. Ainsi, on peut voir cela comme (Objet) "propriété" (Objet) dans le langage de DAML+OIL, tel que " propriété " est "se_map_vers".

commençons par définir la propriété (ou la relation) "se_map_vers" :

```
<daml:ObjectProperty rdf:ID="se_map_vers">
     <rdfs:domain rdf:resource="#BD_SI"/>
     <rdfs:range rdf:resource="#BD1"/>
     <rdfs:range rdf:resource="#BD2"/>
</daml:ObjectProperty>
```

"domain" de la propriétée est BD_SI, cela signifie que la relation "se_map_vers" est applicables sur l'objet BD SI.
"range" de la propriétée est une conjonction de BD1 et BD2. cela signifie que la propriété prend ces valeurs seulement dans BD1 et BD2.
Prenons par exemple, la règle R1 des règles de mapping, qui spécifie que les tronçons du schéma global (noté ici "Tronçon_SI") correspond aux tronçons de la base BD1 (noté "Tronçon_source").
Définition de Tronçon du BD 1 :

```
<daml:Class rdf:ID="Tronçon_source">
     <rdfs:subClassOf rdf:resource="#BD_1"/>
     <daml:disjointWith rdf:resource="#Noeud_source"/>
</daml:Class>
```

Définition de Tronçon du schéma global :

```
<daml:Class rdf:ID="Tronçon_SI">
        <rdfs:subClassOf rdf:resource="#BD_SI"/>
        <daml:disjointWith rdf:resource="#Noeud_SI"/>
</daml:Class>
```

Les deux définitions qu'on vient de voir, doivent être au préalable décrite dans la définition de BD_1 et BD_SI, respectivement. Elles ne sont décrites ici que pour expliquer la traduction de R1.

Maintenant, nous verrons comment traduire la relation "se_map_vers" entre les deux objets Tronçon_SI et Tronçon_source :

```
<daml:Class rdf:about="#Tronçon_SI">
      <rdfs:subClassOf>
            <daml:Restriction>
                  <daml:onProperty rdf:resource="#se_map_vers"/>
                  <daml:toClass rdf:resource="#Tronçon_source"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>
```

Cette définition spécifie que tout élément qui est dans la classe Tronçon_SI, qui est relié par la relation "se_map_vers" est un élément de la classe Tronçon_source. Avec la description que nous venons de voir, celle-ci englobe toutes les règles de mapping où la racine du chemin source d'une règle est Tronçon_source, c'est à dire les règles R2, R3, R4, R5, R12, R13. Pour représenter les autres règles, on fait de même avec les Noeud, Chaussée,...etc.

7.3.3 DAML API et moteur d'inference

Maintenant que les bases de données et les règles de mapping sont bien représentées, comment peut-on exploiter cela pour implémenter le moteur d'évaluation des règles ? Ce dernier devra permettre à l'utilisateur de poser des requêtes sur les données.
Le DAML+OIL actuel, n'inclut pas encore de spécifications de règles d'inférences explicites. Donc, une combinaison avec un moteur d'inférence est nécessaire.
Parmi les outils qui se basent sur DAML, nous avons choisi DAMLJessKB [JKO01], qui utilise JESS (Java Expert System Shell) [JES02], comme moteur d'inférence et Java comme plate forme de programmation.
Cette librairie permet la lecture des fichiers DAML, interprète les informations selon le langage DAML, et permette l'utilisateur de poser des requêtes que ces informations.

Les principales fonctionnalités de cette librairie sont :
– Utilisation d'un API RDF pour la lecture des fichier DAML
– Création des triplets RDF ‹Sujet›‹Propriété›‹Objet›
– Traduction des triplets RDF en triplets JESS.
– Application des règles du langage de JESS aux données.
– Application des agents de règles et de requêtes
– Traduction des faits appropriés en DAML.

Conclusion

Dans ce chapitre, nous avons proposé une étude de cas basé sur 'architecture proposée (Une architecture d'intégration décentralisée basée sur l'architecture P2P), et nous nous intéressons plus particulièrement à l'élaboration d'un langage de règles qui capture les correspondances d'intégration entre ontologies, tout en assurant une représentation de l'information sémantique liée aux sources de données, dans un format (DAML-OIL) qui s'inscrit dans l'activité Semantic Web du W3C. Nous avons traité aussi, l'implémentation d'un moteur d'évaluation de ces règles de correspondance.

Conclusion générale

L'objectif principal de notre travail, est de proposer une architecture de médiation décentralisée pour remédier aux problèmes rencontrés dans l'approche centralisée et ceci dans le cadre du web sémantique. Nous avons réalisé un système de médiation de données qui est inspiré de l'architecture pair-à-pair (P2P) que nous prévoyons pour le web sémantique. Ce système est composé de trois parties :

La première est la présentation du système de mapping qui capture les correspondances entre le schéma global et les schémas sources. Ce système est conçu de manière indépendante du modèle de représentation des connaissances. En effet, ses règles sont exprimées en fonction de concepts et de relations. Nous avons utilisé pour cela, les ontologies comme modèle de représentation des connaissances au niveau conceptuel.

Dans la deuxième, nous avons évoqué la représentation des informations sémantiques tant au niveau conceptuel qu'au niveau de langage de représentation.
Nous avons également utilisé le langage de représentation des ontologies et des règles de mapping qui est le DAML+OIL.

Et en fin, nous avons présenté l'implémentation d'un moteur d'évaluation des règles de mapping, qui est un couplage du moteur d'inférence JESS avec DAML API. Ceci est nécessaire pour permettre l'implémentation.

Ce mémoire constitue une base de travail à partir de laquelle de nouveaux travaux de recherche peuvent être lancés pour atteindre les perspectives suivantes :

1. Renforcer le système proposé par une implémentation qui valide les solutions proposées.
2. S'intéresser à l'automatisation des mises en correspondance entre ontologies. Il faut ensuite pouvoir raisonner sur les correspondances entre ontologies. Il faut s'attendre à une explosion du nombre d'ontologies utilisées. Beaucoup décriront des domaines similaires mais n'utiliseront pas forcément les mêmes termes, d'autres décriront des domaines qui pourront se recouvrir. Il est nécessaire pour cela de développer des recherches portant sur la représentation explicite des mises en correspondance entre ontologies ainsi que sur la conception d'algorithmes de raisonnement efficaces et adaptés au traitement des mises en correspondance de différentes sortes : égalité, inclusion, recouvrement.
3. Enfin, ces systèmes distribués reposent sur l'exploitation d'ontologies elles-aussi distribuées. Un champ de recherches à favoriser concerne alors la gestion à grande échelle de ce nombre très important d'ontologies pouvant couvrir des domaines identiques ou se recouvrant.

Annexe

1. l'ontologie O_{SI}

```
<daml:Ontology rdf:about="">
   <daml:versionInfo>
       $Id: daml+oil-ex.daml,v 1.0 2005/03/27 16:38:38 Exp $
   </daml:versionInfo>
  <rdfs:comment>
  Un exemple d'ontologie, avec des types de données pris du schéma de XML
  </rdfs:comment>
  <daml:imports rdf:resource="http://www.daml.org/2001/03/daml+oil"/>
</daml:Ontology>
```

Description de l'ontologie O_{SI}

```
<daml:Class rdf:ID="BD_SI">
      <rdfs:label>BD_SI</rdfs:label>
      <rdfs:comment>Base de donn´ees g´eographique</rdfs:comment>
</daml:Class>
```

Création des classes de l'ontologie O_{SI}

```
<daml:Class rdf:ID="Tronc¸on">
      <rdfs:label>Tronc¸on</rdfs:label>
      <rdfs:subClassOf rdf:resource="#BD_SI"/>
</daml:Class>

<daml:Class rdf:ID="Noeud">
      <rdfs:label>noeud</rdfs:label>
      <rdfs:subClassOf rdf:resource="#BD_SI"/>
      <rdfs:comment>
        Noeud decrit les point ou les condition du trafic  change.
      </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Chaussée">
      <rdfs:label>Chaussée</rdfs:label>
      <rdfs:subClassOf rdf:resource="#BD_SI"/>
      <rdfs:comment> chaussée est un groupe de voies
      </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Séparateur">
      <rdfs:label>séparateur</rdfs:label>
      <rdfs:subClassOf rdf:resource="#BD_SI"/>
      <rdfs:comment>
  séparateur décrit la séparation entre deux chaussées d'un même tronçon.
      </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Extremit´e">
      <rdfs:label>Extremité</rdfs:label>
      <rdfs:subClassOf rdf:resource="#BD_SI"/>
      <rdfs:comment>
        Extremité est un point ou les conditions de trafic changent.
      </rdfs:comment>
</daml:Class>

<daml:Class rdf:ID="Carrefour">
      <rdfs:subClassOf rdf:resource="#Extremité"/>
</daml:Class>

<daml:Class rdf:ID="Péage">
      <rdfs:subClassOf rdf:resource="#Extremité"/>
</daml:Class>

<daml:Class rdf:ID="Entrée_Tunel">
      <rdfs:subClassOf rdf:resource="#Extremité"/>
</daml:Class>
```

Création des DatatypeProperty

```
<daml:DatatypeProperty rdf:ID="Numéro_route">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_T">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="nb_voies_direct">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="nb_voies_indirect">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_N">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Types_Noeud">
</daml:DatatypeProperty>

<daml: DatatypeProperty rdf:ID="Numéro_route_c">
</daml:DatatypeProperty>

<daml: DatatypeProperty rdf:ID="nb_voies">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_C">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Position">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_S">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Hauteur">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_E">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_Ca">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_P">
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="Geom_Et">
</daml:DatatypeProperty>
```

Création des ObjectProperty

```
<daml:ObjectProperty rdf:ID="débute par">
      <rdfs:label>débute par</rdfs:label>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="fini par">
      <rdfs:label>fin par</rdfs:label>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="sépare">
      <rdfs:label>s´epare</rdfs:label>
</daml:ObjectProperty>
```

Une extrémité peut être un carrefour, péage ou entrée-tunnel

```
<daml:Class rdf:about="#Extremité">
      <daml:disjointUnionOf rdf:parseType="daml:collection">
            <daml:Class rdf:about="#Carrefour"/>
            <daml:Class rdf:about="#Péage"/>
            <daml:Class rdf:about="#Entrée_Tunnel"/>
```

```
      </daml:disjointUnionOf>
</daml:Class>
```

Un Noeud peut être enum par carrefour, péage ou entrée -tunnel

```
<daml:Class rdf:ID="Noeud">
      <daml:oneOf rdf:parseType="daml:collection">
            < noeud rdf:ID="carref"/>
            < noeud rdf:ID="péag"/>
            < noeud rdf:ID="tunnel"/>
      </daml:oneOf>
</daml:Class>
```

Les relations entre les classes :

```
<daml:Class rdf:about="#Tronçon">
      <rdfs:subClassOf>
            <daml:Restriction daml:Cardinality="1">
                  <daml:onProperty rdf:resource="#débute par"/>
                  <daml:hasClassQ rdf:resource="#Noeud"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Tronçon">
      <rdfs:subClassOf>
            <daml:Restriction daml:Cardinality="1">
                  <daml:onProperty rdf:resource="#fin par"/>
                  <daml:hasClassQ rdf:resource="#Noeud"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Noeud">
      <rdfs:subClassOf>
            <daml:Restriction daml:maxCardinality="n">
                  <daml:onProperty rdf:resource="#débute par"/>
                  <daml:hasClassQ rdf:resource="#Tronc,on"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Noeud">
      <rdfs:subClassOf>
            <daml:Restriction daml:maxCardinality="n">
                  <daml:onProperty rdf:resource="#fin par"/>
                  <daml:hasClassQ rdf:resource="#Tronçon"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Chaussée">
      <rdfs:subClassOf>
            <daml:Restriction daml:Cardinality="1">
                  <daml:onProperty rdf:resource="#d´ebute par"/>
                  <daml:hasClassQ rdf:resource="#Extremit´e"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Chaussée">
      <rdfs:subClassOf>
            <daml:Restriction daml:Cardinality="1">
                  <daml:onProperty rdf:resource="#fin par"/>
                  <daml:hasClassQ rdf:resource="#Extremit´e"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Extremité">
```

```
      <rdfs:subClassOf>
            <daml:Restriction daml:maxCardinality="n">
                  <daml:onProperty rdf:resource="#débute par"/>
                  <daml:hasClassQ rdf:resource="#Chaussée"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Extremité">
      <rdfs:subClassOf>
            <daml:Restriction daml:maxCardinality="n">
                  <daml:onProperty rdf:resource="#fin par"/>
                  <daml:hasClassQ rdf:resource="#Chaussée"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Séparateur">
      <rdfs:subClassOf>
            <daml:Restriction daml:Cardinality="2">
                  <daml:onProperty rdf:resource="#sépare"/>
                  <daml:hasClassQ rdf:resource="#Chaussée"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>
```

Représentation des ACIs :

Tronçon ⊆ SET ([1 :N] Chaussée, [0 :N] Séparateur) :

```
<daml:Class rdf:about="#Tronçon">
      <rdfs:subClassOf>
            <daml:Restriction daml:maxCardinality="n">
                  <daml:onProperty rdf:resource="#correspond"/>
                  <daml:hasClassQ rdf:resource="#Chaussée"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Tronçon">
      <rdfs:subClassOf>
            <daml:Restriction daml:minCardinality="1">
                  <daml:onProperty rdf:resource="#correspond"/>
                  <daml:hasClassQ rdf:resource="#Chaussée"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:about="#Tronçon">
      <rdfs:subClassOf>
            <daml:Restriction daml:maxCardinality="n">
                  <daml:onProperty rdf:resource="#correspond"/>
                  <daml:hasClassQ rdf:resource="#Séparateur"/>
            </daml:Restriction>
      </rdfs:subClassOf>
</daml:Class>
```

Noeud ⊂ Extrémité

```
<daml:Class rdf:ID = "noeud">
      <daml:sameClassAs rdf:resource="#Extremité">
</daml:Class>
```

Noeud.Types Noeud=”carrefour” = Carrefour

```
<daml:Class rdf:ID="Carrefour">
      <daml:sameClassAs>
            <daml:Restriction>
                  <daml:onProperty rdf:resource="#Types_Noeud"/>
                  <daml:hasValue rdf:resource="#carref"/>
```

```
            </daml:Restriction>
      </daml:sameClassAs>
</daml:Class>
```

Noeud.Types Noeud=”peage” = Péage

```
<daml:Class rdf:ID="P´eage">
      <daml:sameClassAs>
            <daml:Restriction>
                  <daml:onProperty rdf:resource="#Types_Noeud"/>
                  <daml:hasValue rdf:resource="#p´eag"/>
            </daml:Restriction>
      </daml:sameClassAs>
</daml:Class>
```

Noeud.Types Noeud=”entrée-tunel” = Entrée Tunnel

```
<daml:Class rdf:ID="Entrée_Tunel">
      <daml:sameClassAs>
            <daml:Restriction>
                  <daml:onProperty rdf:resource="#Types_Noeud"/>
                  <daml:hasValue rdf:resource="#tunnel"/>
            </daml:Restriction>
      </daml:sameClassAs>
</daml:Class>
```

2.L'ensemble des règles de mapping

R_1 :	$\mathcal{O}_1$/Tronçon $\rightsquigarrow$ T	$\longleftarrow$	$\mathcal{O}_{SI}$.Tronçon
R_2 :	T/geom_T	$\longleftarrow$	$\mathcal{O}_{SI}$.T.a_une_géométrie
R_3 :	T/numéro_route	$\longleftarrow$	$\mathcal{O}_{SI}$.T.appartient_à_la_route_n°
R_4 :	T/nb_voies_direct	$\longleftarrow$	$\mathcal{O}_{SI}$.T.a_nb_voies_direct
R_5 :	T/nb_voies_indirect	$\longleftarrow$	$\mathcal{O}_{SI}$.T.a_nb_voies_indirect
R_6 :	$\mathcal{O}_1$/Noeud $\rightsquigarrow$ N	$\longleftarrow$	$\mathcal{O}_{SI}$.Noeud
R_7 :	N/geom_N	$\longleftarrow$	$\mathcal{O}_{SI}$.N.a_une_géométrie
R_8 :	N/types_noeud $\rightsquigarrow$ Tn	$\longleftarrow$	$\mathcal{O}_{SI}$.N.a_un_type
R_9 :	Tn/Carrefour	$\longleftarrow$	$\mathcal{O}_{SI}$.N.est_de_type_Carrefour
R_{10} :	Tn/Peage	$\longleftarrow$	$\mathcal{O}_{SI}$.N.est_de_type_Péage
R_{11} :	Tn/Tunnel	$\longleftarrow$	$\mathcal{O}_{SI}$.N.est_de_type_Tunnel
R_{12} :	T/Noeud_début $\rightsquigarrow$ N	$\longleftarrow$	$\mathcal{O}_{SI}$.T.débute_par_un
R_{13} :	T/Noeud_fin $\rightsquigarrow$ N	$\longleftarrow$	$\mathcal{O}_{SI}$.T.fini_par_un
R_{14} :	N/début_T $\rightsquigarrow$ T	$\longleftarrow$	$\mathcal{O}_{SI}$.N.est_un_début_de
R_{15} :	N/fin_T $\rightsquigarrow$ T	$\longleftarrow$	$\mathcal{O}_{SI}$.N.est_une_fin_de
R_{16} :	$\mathcal{O}_2$/Chaussée $\rightsquigarrow$ C	$\longleftarrow$	$\mathcal{O}_{SI}$.Chaussée
R_{17} :	C/geom_C	$\longleftarrow$	$\mathcal{O}_{SI}$.C.a_une_géométrie
R_{18} :	C/numéro_route	$\longleftarrow$	$\mathcal{O}_{SI}$.C.appartient_à_la_route_n°
R_{19} :	C/nb_voies	$\longleftarrow$	$\mathcal{O}_{SI}$.C.a_nb_voies
R_{20} :	C/position $\rightsquigarrow$ Cp	$\longleftarrow$	$\mathcal{O}_{SI}$.C.a_une_position
R_{21} :	Cp/surface	$\longleftarrow$	$\mathcal{O}_{SI}$.Cp.est_de_type_surface
R_{22} :	Cp/tunnel	$\longleftarrow$	$\mathcal{O}_{SI}$.Cp.est_de_type_tunnel
R_{23} :	$\mathcal{O}_2$/Séparateur $\rightsquigarrow$ S	$\longleftarrow$	$\mathcal{O}_{SI}$.Séparateur
R_{24} :	S/geom_S	$\longleftarrow$	$\mathcal{O}_{SI}$.S.a_une_géométrie
R_{25} :	S/hauteur	$\longleftarrow$	$\mathcal{O}_{SI}$.S.a_une_hauteur
R_{26} :	S/sépare $\rightsquigarrow$ C	$\longleftarrow$	$\mathcal{O}_{SI}$.S.sépare
R_{27} :	C/est_séparé $\rightsquigarrow$ S	$\longleftarrow$	$\mathcal{O}_{SI}$.C.est_séparé_par
R_{28} :	$\mathcal{O}_2$/Extrémité $\rightsquigarrow$ E	$\longleftarrow$	$\mathcal{O}_{SI}$.Extrémité
R_{29} :	E/geom_E	$\longleftarrow$	$\mathcal{O}_{SI}$.E.a_une_géométrie
R_{30} :	C/Extrem_début $\rightsquigarrow$ E	$\longleftarrow$	$\mathcal{O}_{SI}$.C.débute_par_une
R_{31} :	C/Extrem_fin $\rightsquigarrow$ E	$\longleftarrow$	$\mathcal{O}_{SI}$.C.fini_par_une
R_{32} :	E/début_C $\rightsquigarrow$ C	$\longleftarrow$	$\mathcal{O}_{SI}$.E.est_un_début_de
R_{33} :	E/fin_C $\rightsquigarrow$ C	$\longleftarrow$	$\mathcal{O}_{SI}$.E.est_une_fin_de
R_{34} :	E/types_Extrem $\rightsquigarrow$ Te	$\longleftarrow$	$\mathcal{O}_{SI}$.E.est_de_type
R_{35} :	Te/Carrefour $\rightsquigarrow$ Ca	$\longleftarrow$	$\mathcal{O}_{SI}$.E.est_de_type_Carrefour
R_{36} :	$\mathcal{O}_2$/Carrefour $\rightsquigarrow$ Ca	$\longleftarrow$	$\mathcal{O}_{SI}$.Carrefour
R_{37} :	Ca/geom_Ca	$\longleftarrow$	$\mathcal{O}_{SI}$.Ca.a_une_géométrie
R_{38} :	Te/Péage $\rightsquigarrow$ P	$\longleftarrow$	$\mathcal{O}_{SI}$.E.est_de_type_Péage
R_{39} :	$\mathcal{O}_2$/Péage $\rightsquigarrow$ P	$\longleftarrow$	$\mathcal{O}_{SI}$.Péage
R_{40} :	P/geom_P	$\longleftarrow$	$\mathcal{O}_{SI}$.P.a_une_géométrie
R_{41} :	Te/Entrée_Tunnel $\rightsquigarrow$ Et	$\longleftarrow$	$\mathcal{O}_{SI}$.E.est_de_type_Entrée_Tunnel
R_{42} :	$\mathcal{O}_2$/Entrée_Tunnel $\rightsquigarrow$ Et	$\longleftarrow$	$\mathcal{O}_{SI}$.Entrée_Tunnel
R_{43} :	Et/geom_Et	$\longleftarrow$	$\mathcal{O}_{SI}$.Et.a_une_géométrie

3.Les sources de données BD_1 (S1) et BD_2 (S2) :

```
Source S1
---------
TRONCON(Geom : ligne, NumRoute : String, NbVoiesDirect : entier,
        NbVoiesIndirect : entier, NoeudDebut : String, NoeuFin: String)
NOEUD(IdNoeud : String, Geom : point, Type : enum)

Source S2
---------
CHAUSSEES(IdChaussee : String, Geom : ligne, NumRoute : String, NbVoies : entier
          Position : enum , ExtremDebut : String, ExtremFin : String)
EXTREMITE(IdExtremite : String, Geom : String)
SEPARATEUR(Geom : String, Chaussee1 : IdChaussee, Chaussee2 : IdChaussee )
CARREFOUR(IdCar : String, Geom : point)
PEAGE(IdPeage : String, Geom : surface, Toponvme : String)
ENTREETUNEL(IdEntree : String, Geom : point)
```