Université de Montréal

Extraction et Intégration des Données à partir des Pages WEB

Par
Hicham SNOUSSI

Département d'Informatique et de Recherche Opérationnelle
Faculté des Arts et Sciences

Mémoire présenté à la Faculté des Etudes Supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M. Sc.)
en Informatique

Décembre 2000

Université de Montréal

Faculté des Etudes Supérieures

Ce mémoire intitulé

Extraction et Intégration des Données à partir des Pages WEB

Par

Hicham SNOUSSI

a été évalué par un jury composé des personnes suivantes :

Claude FRASSON, président – rapporteur
Jian-Yun NIE, directeur
Laurent MAGNIN, co-directeur
Jean-Yves POTVIN, membre du jury

Mémoire accepté le : 9 mars 2001

A mes Parents,
mes sœurs.

Remerciements

Une fois ce travail accompli, je ne pouvais m'empêcher de manifester ma reconnaissance à mes deux Directeurs de recherche pour leurs appuis durant mon projet de recherche.

Je tiens à exprimer mes remerciements à Laurent MAGNIN pour avoir dirigé et encadré mon projet de maîtrise. Je lui exprime également ma gratitude pour son soutien moral et financier dans des moments difficiles. Les conditions et l'environnement de travail au sein de l'équipe étaient très motivants.

Je remercie Jian-Yun NIE pour avoir accepté de diriger ce travail, il m'a fournit ses remarques et orientations pertinentes. Ses commentaire et ses indications m'ont beaucoup aidé à accomplir ce travail dans le meilleur des cas.

SOMMAIRE

De nos jours, Internet est un moyen de communication incontournable. On y présente de grandes quantités d'information et de données jamais atteintes auparavant. En particulier, le WEB est le service le plus utilisé. De ce fait, il est devenu la première source d'information de toutes sortes. Les données ne sont pas seulement diversifiées, elles sont aussi mises à jour régulièrement dans la plupart des cas.
La recherche et l'extraction de données dans cette grande masse d'information deviennent alors nécessaires pour repérer et récupérer les données qui intéressent les usagers. Nous ne traitons pas dans ce travail la problématique de la recherche d'information mais plutôt celle de l'extraction et la récupération des données des pages WEB.

L'hétérogénéité des sources d'information rend difficile la récupération automatique et le traitement des données par un programme ou un agent. Les différences de structures et le manque d'une uniformisation des données des sources constituent des obstacles à cette fin. Chaque nouveau site formate et structure ses données de manière différente des autres.

Nous nous proposons dans ce mémoire d'essayer de trouver une approche afin de faciliter la formalisation, l'extraction et le regroupement des données appartenant à des sources différentes. Pour ceci, nous avons développé un utilitaire qui permet de générer des descriptions pour différentes sources afin de faciliter l'extraction de certains types de données. En se basant sur une ontologie descriptive d'un domaine d'intérêt, l'utilisateur utilise ces descriptions pour extraire les données, les mettre en commun et les interroger de manière uniforme par la suite.

Nous avons appliqué cette méthode d'extraction pour alimenter un système multi-agents en données. Cette extraction est un préalable aux traitements et services rendus par les agents.

A travers ce mémoire, nous avons pu appliquer une approche basée sur un modèle commun pour l'intégration aussi bien que pour l'extraction des données à partir des pages WEB. Nous avons montré que cette extraction automatique est réalisable pour certaines données (fréquemment mises à jour) et peut être appliquée à plusieurs applications.

Dans ce mémoire, nous avons proposé une méthodologie pour extraire, intégrer et présenter les données et les rendre compréhensibles en respectant un certain formalisme partagé par une communauté d'un domaine particulier. C'est un moyen pour permettre aux utilisateurs de récupérer automatiquement les données du WEB. Ceci permet également de partager les données plus facilement entre systèmes en éliminant la nécessité de transfert de données lors du passage d'un formalisme de représentation d'un système à un autre.

Mots-clés : extraction de données, WEB, ontologie, agent, XML.

Table des matières

Liste des figures

Liste des Tableaux

Liste des Abréviations

B2B	Business to Business
B2C	Business to Customer
DARPA	Defense Advanced Research Agency
DOM	Document Object Model
DTD	Document Type Definition
EDI	Electronic Data Interchange
HEL	HTML Extraction Language
HTML	Hyper Text Markup Language
HTTP	HyperText Transfer Protocol
ICL	Inter-agent Communication Language
JEDI	Java Extraction and Dissemination of Information
KIF	Knowledge Interchange Format
KQML	Knowledge Query and Manipulation Language
NSL	Nested String List
OAA	Open Agent Architecture
OEM	Object Exchange Model
OQL	Object Query Language
RDF	Resource Description Framework
SIMS	Service and Information Management for decision System
SMA	Système Multi-Agents
SOX	Schema for Object-oriented XML
SQL	Structured Query Language
TSIMMIS	The Stanford-IBM Manager of Multiple Information Sources
URL	Uniform Resource Location
W3C	World Wide Web Consortium
W4F	World Wide Web Wrapper Factory
WMA	Web Mining Agent
WONDEL	Web Ontology Description Language
WWW	World Wide Web
xCBL	XML Common Business Library
XDR	XML-Data Reduced

XML	Extensible Markup Language
XQL	XML Query Language
XSL	XML Style Sheet

CHAPITRE 1

Introduction

L'abondance de l'information et la multitude de sources de données disponibles actuellement sur Internet offre à l'utilisateur une mine quasi inépuisable, variée et continue de données. Si un utilisateur peut se réjouir de ces possibilités, la récupération de données dans cette masse d'information peut s'avérer une tâche fastidieuse. En effet, celle-ci est consommatrice de temps et d'énergie et le résultat final risque de ne pas correspondre aux attentes de l'utilisateur. Une assistance automatique pour l'accès aux ressources Internet et Intranet est de ce fait hautement souhaitable.

Actuellement, les outils d'aide à l'accès aux informations sur le WEB sont pour la plupart sous forme de moteurs de recherche ou de portails. Ces outils se basent sur des stratégies de mot-clé pour identifier les documents qui intéressent l'utilisateur. Les moteurs de recherche permettent sans doute d'effectuer une recherche et présentent les résultats sous forme d'index de liens (ou des pages HTML) dont la vérification de la pertinence revient toujours à l'utilisateur qui la fera manuellement. Les outils de recherche ne permettent pas une structuration et un traitement des données trouvées. L'utilisateur est tenu de parcourir les liens trouvés et en extraire "à la main" les données qu'il désire. La nécessité de disposer d'un système qui rassemble et organise le maximum de données pour un utilisateur croît de jour en jour et au fur et à mesure que de nouvelles sources apparaissent[1].

---

[1] Par exemple, en France, le nombre de sites Web a été multiplié par 2,5 en un an, celui de pages par plus de 4 en 1998. Source : Association Française de la Télématique Multimédia.

La problématique et la recherche relative à l'intégration concernait en particulier les bases de données hétérogènes. Récemment, les sources sont devenues de plus en plus variées suite au développement rapide des moyens de communication notamment Internet, le problème a regagné alors un nouvel intérêt pour les nouveaux types de sources.

1.1 Problématique

La présentation des informations est différente et hétérogène d'un Site WEB à un autre même si ses informations sont similaires. L'utilisation de ces information par un programme est alors difficile dû à la grande différence du format. Extraire et formaliser les données des sources permet de faciliter et favoriser le traitement escompté.

Si pour accéder aux informations des bases de données, le schéma conceptuel de la base est nécessaire et suffisant, l'accès aux données sur les pages WEB est moins évident. Ces données se trouvent à l'intérieur de pages HTML qui ont pour fonction première de les présenter à l'utilisateur. En effet, les pages WEB ne disposent pas de descriptions schématiques (à l'instar des bases de données ou de connaissances). Vouloir récupérer automatiquement les données du WEB exige des connaissances sur la structuration et sur le contenu des pages. On distingue pour cela quelques approches :

- la première vise à utiliser le traitement du langage naturel pour comprendre la sémantique du texte affiché. L'implantation de cette technique est difficile et reste restreinte à un domaine d'application particulier ;
- une deuxième formule vise à ajouter de la sémantique aux pages Web au moment de leurs créations, par exemple en utilisant des balises personnalisées. Les limitations d'un tel procédé sont bien connues étant donné que ces balises restent personnelles et ne sont pas généralisées [Atzeni *et al*, 97] ;

Une autre difficulté qui se pose à l'intégration des données concerne leur mise en commun après extraction. A l'origine, les données étaient organisées selon des structures différentes sur leurs sources respectives. Dans le but de les mettre en commun, les données sont à restructurer selon un modèle commun indépendant des sources.

L'éparpillement des sources de données, le manque de structure uniforme et leur hétérogénéité sont les problèmes qui se posent pour l'intégration et l'extraction des données. Ceci est dans le but d'offrir à l'utilisateur un accès automatique uniforme et rapide aux données sur le WEB.

Le but est d'étudier les possibilités offertes pour résoudre ces difficultés d'extraction et de mise en commun de certaines données sur le WEB.

1.2 Considérations du domaine

La conséquence des coûts élevées et l'absence de normes pour une infrastructure globale de communication n'ont pas aidé autrefois les organismes à développer un système d'information centralisé pour leurs besoins. Les sources d'informations ont ainsi été crées au besoin, et sont maintenant hétérogènes et souvent non liées entre elles. Avec l'avènement de l'Internet et l'Intranet par la suite, les entreprises et les organismes[2] ont commencé à offrir, à leurs employés et aux utilisateurs de manière générale, l'accès à un ensemble très important de données. L'utilisateur se trouve dès lors à devoir gérer plusieurs points d'accès, ce qui implique des modes différents d'interrogation et une gestion répétitives des actions et une perte de temps considérable. Aussi la construction d'applications de traitement de données et d'aide à la décision se voit contrainte par l'éparpillement des sources et l'hétérogénéité des structures des sources.
Le rôle de l'intégration des données est de faciliter justement la création et l'utilisation de ce genre d'applications ou de programmes (eg. des agents) afin de produire des services à valeurs ajoutées. Pour cela, ces applications se baseront sur les informations pertinentes provenant de différents sites et ayant une même présentation synthétique.

1.2.1 Sources d'information[3]

Les différentes sources d'information susceptibles de pouvoir être utilisées pour l'intégration peuvent être classées en trois grandes catégories : sources structurées, semi-structurées et non structurées [Martin et al, 97].

- Une source est structurée si l'information est organisée de telle façon à permettre des requêtes exprimées dans un langage bien défini. Les bases de données et les bases de connaissances sont l'archétype des sources structurées.
- Une source est considérée comme semi-structurée si elle contient une structuration des données suffisante pour pouvoir être traitée comme une base de donnée sans qu'un langage de requête ne soit disponible. [Knoblock *et al*, 98] considèrent une source comme semi structurée si l'information peut être récupérée à l'aide d'une grammaire formelle. Les pages

[2] Sans oublier les particuliers et leurs pages personnelles
[3] On appellera source d'information tout système en mesure de fournir des données quelques soient leurs types.

HTML, les formulaire WEB et les langages de description de textes sont des exemples de sources semi-structurées.

- Une source est non structurée si elle ne possède pas de formes d'organisation ni de relations précises entre ses données.

Le traitement des sources non structurées nécessitent encore plus d'efforts au niveau des techniques du traitement de la langue naturelle. D'autres part, de plus en plus de sites offrent des données structurées ou semi-structurées. Nous allons traité dans ce travail des données structurées et semi-structurées.

1.2.2 Considérations pratiques liées aux sources

Les problèmes essentiels des accès aux données présentes sur Internet sont la répartition, l'hétérogénéité et leurs caractère dynamique [Genesereth *et al*, 97].

Répartition : une source d'information n'est pas souvent en mesure d'offrir une réponse complète pour une requête d'un utilisateur : ce dernier est obligé de consulter différents sites qui fournissent chacun une réponse partielle pour constituer une réponse optimale. Par exemple, prenons un utilisateur qui souhaite avoir "le meilleur prix d'une voiture BMW 740i dans la ville de Montréal" à partir de site WEB de concessionnaires contenant les prix qu'ils appliquent. Afin de répondre adéquatement à une telle requête, nous devons rassembler les données, qui sont réparties sur différents sites, et de construire la réponse finale. Par exemple, la liste de tous les prix est construite à partir des listes de prix de tous les concessionnaires. La requête sera résolue en se basant sur les prix appliqués par l'ensemble des concessionnaires.

Hétérogénéité : Il peut s'agir soit d'une hétérogénéité de notation, soit de conception [Genesereth *et al*, 97]. Les différences de notation concernent le langage et le protocoles d'accès. Par exemple, des bases de données comme Sybase ou Informix utilisent SQL tandis qu'OQL est utilisé par ObjectStore. Ceci dit, même s'il s'agit de bases de données utilisant la même conception et le même langage, il reste néanmoins des différences possibles au niveau du vocabulaire utilisé. Une même appellation peut désigner différents types de données, ou différentes appellations réfèrent au même type de donnée.

Instabilité : Les sources qui existent sur Internet revêtent un caractère dynamique, de telle sorte que des sources disparaissent et d'autres sont créées continuellement. Aussi, les sources changent de contenu et peuvent changer de format de présentation essentiellement pour les pages WEB, à la différence des bases de données qui conservent toujours la même structure pour les données.

Cependant, nous avons constaté que de plus en plus de sources sur le WEB gardent une même structure alors que leur contenu change. Ceci est vrai en particulier pour les sites qui se trouvent en avant des bases de données.

Dans la littérature, on considère souvent que le processus d'intégration des données est à l'image d'un médiateur entre l'utilisateur sollicitant l'information et les sources d'information. La médiation est donc le processus par lequel un client demande une information parmi une variété de sources sans se soucier de l'identité, l'emplacement, les schémas ou les mécanismes d'accès à ces sources. Cette fonction de médiation est la clé de l'intégration des données car elle permet un accès transparent et une interface aux sources [Martin *et al*, 97]. Nous retrouvons les fonctions de médiation dans pratiquement tous les travaux dans le domaine de l'intégration de données.

1.2.3 Agents intelligents pour l'intégration d'information

L'extraction de données telles que les taux de change, les valeurs des actions sur les marchés boursiers sont autant de services qui peuvent être rendus par un agent. L'application de notre approche aux agents vise à montrer ce genre de services dans un environnement multi-agents. La création d'agents spécialisés pour la récupération de données est une nécessité dans un tel environnement impliquant naturellement la communication, la coopération entre agents et la résolution distribuée de problèmes.

L'évolution du paradigme des agents et les différentes recherches théoriques et pratiques menées sur leurs applications laissent à prévoir que les agents seront d'une grande utilité pour la récupération et le traitement de l'information et des données. De telles fonctionnalités correspondent bien aux services que peut offrir un agent ou plus spécifiquement un système multi-agents.

Nous rappellerons ici une définition du concept d'agent. Même s'il n'y a pas de définition commune officielle, la communauté agent s'accorde à dire qu'un agent est une entité logicielle qui exécute des tâches de manière autonome. En général, un agent est caractérisé par des propriétés dont les plus importantes sont : l'autonomie, la mobilité, la réactivité, la communication, la coopération. Dépendamment du contexte de son utilisation, on favorise l'une ou l'autre de ces propriétés.

L'intégration des données est souvent considérée comme une exigence essentielle pour mettre en valeur les fonctionnalités pratiques des agents intelligents. En effet, la récupération et l'obtention d'informations sont des exigences préalables à tout traitement ou service rendu par un agent "intelligent". Des services telles que la planification de voyages, la recherche de documents, les

activités reliées au commerce électronique (achat, vente et négociation), la gestion des titres d'actions et l'observation du marché boursier, nécessitent pour leur mise en œuvre d'être alimentés en données.

1.2.4 Objectifs

A travers cette étude, nous essaierons de trouver des approches pour atteindre nos objectifs :

- Extraire les données des pages WEB : il s'agit de trouver une méthode pour repérer et récupérer les données qui intéressent l'utilisateur à l'intérieur des pages HTML, ces données sont préalablement identifiées par ce dernier.
- Trouver un formalisme de représentation pour les données extraites des différents sites WEB : l'hétérogénéité de la conception, de la présentation et du vocabulaire nous pousse à utiliser un formalisme commun pour représenter les données de même type.
- Utiliser ce formalisme pour intégrer les données extraites et les mettre en commun : ce qui permettra à l'utilisateur d'interroger d'une manière uniforme des données provenant de différentes sources (sites WEB).
- Automatiser le processus d'extraction : il s'agit d'un objectif essentiel. En effet, l'automatisation de ce processus (étapes d'extraction et d'intégration) permettra à l'utilisateur de récupérer automatiquement les données du WEB. Ainsi, à l'aide de ce système, l'utilisateur doit pouvoir facilement définir un extracteur pour un site. Les données extraites seront standardisées et rendues disponibles à d'autres utilisations (utilisateurs). Aussi, notre travail dans ce mémoire peut être vu comme une étape menant à l'intégration des données du WEB aux agents intelligents. Un agent intelligent récupère et utilise au besoin des données (telles que les cours de change, données météorologiques, valeurs des actions, données financières ...) directement des sites WEB sans intervention de l'utilisateur.

Nous ne tenterons pas de traiter tous les types de données sur le WEB. En effet, seulement une partie de ces données peut être extraite de manière automatique (données structurées et semi-structurées). Nous donnerons des exemples de tels sites plus en avant dans ce mémoire.

1.2.5 Plan du mémoire

Nous commençons notre étude par une revue de la littérature sur l'intégration des données suivie de la conception générale de notre approche basée sur les conclusions et la synthèse de l'état de l'art. Les chapitres qui suivent traitent les trois principaux axes de notre approche : une modélisation du domaine d'intérêt, la procédure d'extraction des sources et la construction des

requêtes. Ces trois chapitres constituent notre principale contribution dans le domaine. Dans le dernier chapitre, nous exposons notre point de vue concernant une application directe de l'extraction et de l'intégration des données. En effet, l'opération d'extraction des données de plusieurs sources et leur intégration constituent l'étape nécessaire à tout traitement ultérieur de ces données. Nous considérons en particulier que la récupération (automatique) de données en vue d'exécuter d'autres opérations du commerce électronique (comparaison, achat, vente, statistiques,...) est essentielle pour ce domaine (commerce électronique). Nous finissons ce travail par une conclusion.

CHAPITRE 2

Revue de l'état de l'art

Dans ce chapitre, nous présenterons des recherches et systèmes existants dans le domaine de l'intégration d'information[4]. Nous verrons quelles sont les tentatives antérieures pour traiter cette problématique.

2.1 Le projet SIMS

SIMS a été initié par l'Information Science Institute de l'university of Southern California. SIMS (Service and Information Management for decision System) vise, à l'instar de la plupart des projets présenté dans ce document, à offrir un accès à différentes sources d'informations [Arens *et al*, 93]. SIMS est parmi les premiers travaux qui se sont intéressés à l'intégration des sources d'informations incluant les bases de données et les bases de connaissances.

2.1.1 Approche de SIMS

SIMS applique une variété de techniques d'intelligence artificielle pour construire une interface aux sources d'information. En particulier, il se base sur [Arens *et al*, 93]:

- *Modélisation et la représentation des connaissances* : les structures et les contenus des sources d'information utilisées par SIMS et le domaine auquel se rapporte ces sources sont

[4] Il est à remarquer que les figures et schémas utilisés pour illustrer les architecture et les composantes des systèmes sont inspirés (éventuellement modifiés) des travaux correspondants.

modélisés. Le modèle de chaque source d'information concerne le schéma des données, le langage de requête, la localisation de la source, la taille ... Le contenu est quant à lui décrit en utilisant le modèle du domaine. L'utilisateur formule ses requêtes en terme du modèle du domaine d'application.

- *Recherche/Planification* : SIMS choisit les sources d'information qui seront utilisées pour résoudre une requête et construit des séquences de requêtes pour chacune d'elles. Il détermine où les données intermédiaires sont traitées et quelles sont les requêtes qui peuvent être évaluées en parallèle.

2.1.2 Outils utilisés par SIMS

SIMS s'appuie sur un certain nombre de techniques/outils pour la modélisation et la résolution des requêtes :

- LOOM : SIMS utilise LOOM pour la représentation des connaissances liées au modèle du domaine et des sources d'information. LOOM est un langage de connaissances basé sur les frames et les réseaux sémantiques. Les connaissances déclaratives de LOOM sont sous forme de définitions, de règles et de faits. La partie déductive utilise le chaînage avant, l'unification et les techniques de maintenance des objets pour transformer la partie déclarative en un réseau, ce qui permet de supporter efficacement le traitement déductif des requêtes. LOOM est un descendant de KL-ONE [Arens *et al*, 93] ;
- LIM : Loom Interface Module a été développé pour jouer l'intermédiaire entre LOOM et les bases de données. Etant donnée une requête LOOM, LIM génère automatiquement une requête appropriée exprimée dans le langage de la base de données, le résultat retourné est vue comme des instances de LOOM ;
- PRODIGY : Il est utilisé pour résoudre le problème de planification des requêtes, il utilise le modèle du domaine comme son modèle du monde. La requête est considérée par PRODIGY comme un but à atteindre, il génère une séquence d'opérations (opérateurs) qui font passer le monde de son état initial à un état où le but devient satisfait. PRODIGY facilite la construction des règles pour contrôler la recherche, la planification des opérations et la supervision de l'exécution.

2.1.3 Modélisation des sources d'information

Naturellement, pour pouvoir accéder à une source d'information, SIMS doit disposer de la description de celle-ci. Pour chacune d'elles, SIMS construit un modèle qui regroupe les informations suivantes [Arens *et al*, 96]:

- La nature de la source d'information, base de données ou base de connaissances, ces deux types ont été les seules catégories supportées par SIMS ;
- La taille, les tables et l'emplacement de la base ;
- Les clés utilisées dans les bases (si elles existent) ;
- Une description du contenu des sources.

2.1.4 Modélisation du domaine dans SIMS

La modélisation du domaine est une base de connaissances décrite en une hiérarchie de terminologies du domaine d'application [Arens *et al*, 96]. Les nœuds représentent tous les états, actions et objets du domaine, aussi le modèle décrit les relations possibles entre les nœuds. Les entités du domaine ne correspondent pas forcément à des objets dans des bases de données, elles représentent une modélisation du domaine d'application. Toutefois, les bases de données qui seront exploitées sont décrites en utilisant le modèle du domaine. La figure 2-1 montre une vue hiérarchique du modèle du domaine auquel sont associées des bases de données (ombrées sur la figure) : AFSC Seaport, AFSC Airport, GEO Geoloc et AFSC Airport. La figure (2-2) montre plus en détails les relations entre entités du modèle du domaine et le contenu de la base AFSC Seaport par exemple [Arens *et al*, 93]. Le domaine traité est celui des transports en particulier les ports et les aéroports.

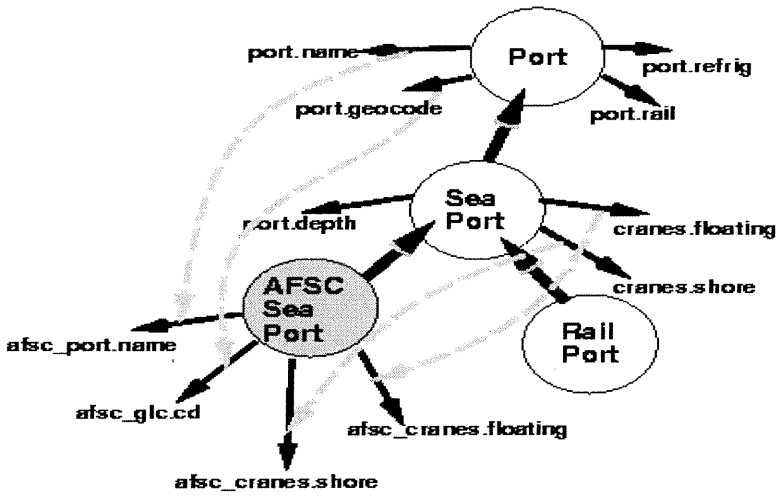


Figure 2-1 : relations entre domaine et contenu des sources

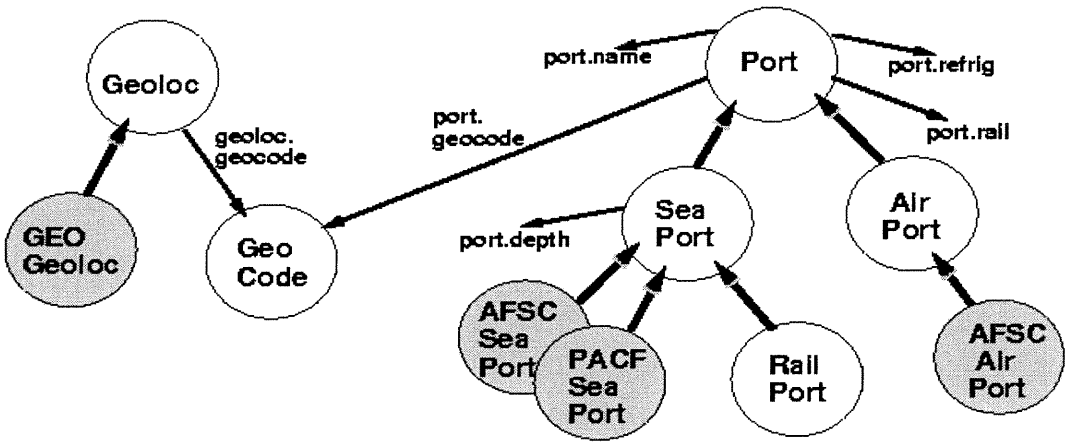


Figure 2-2 : modèle du domaine

2.1.5 Réponse aux requêtes

La réponse aux requêtes nécessite la définition des sources utilisées par celles-ci [Arens et al, 93], Ce qui suppose qu'il faut au préalable déterminer les sources d'information qui contiennent les données pertinentes pour la résolution des requêtes. Une source est sélectionnée si les concepts utilisés pour exprimer la requête peuvent être reliés aux concepts présents au niveau du modèle de la base.

Ensuite, SIMS produit un plan d'exécution. Il s'agit d'un plan pour extraire les données et les manipuler en utilisant des sous-requêtes destinées aux sources sélectionnées. La résolution des dépendances logiques entre les sous-requêtes est assurée par les fonctionnalités de raisonnement de LOOM.

Un prototype se basant sur l'approche de SIMS a été appliqué à la planification du domaine du transport. Il a été utilisé pour intégrer les informations portuaires (ports, bateaux, activités, locaux, ...) stockées au niveau de neuf bases de données Oracle [Arens et al, 93].

Naturellement, et vu que les bases de données qui sont utilisées par le système sont étroitement liées au modèle du domaine, tout changement au niveau de ce dernier entraîne une révision de ces relations.

2.1.6 Ariadne

Le projet SIMS a été conçu pour offrir un cadre de travail pour intégrer des bases de données et des bases de connaissances. Ariadne est une composante de SIMS pour l'extraction de données à

partir des pages WEB, ces dernières sont devenues des sources incontournables [Knoblock et al, 98]. Une page WEB est vue comme une source relationnelle (ou une petite base de données).
Bien que l'approche générale d'Ariadne soit basée sur l'architecture de SIMS, l'extraction des données y est naturellement différente car il s'agit des sources semi structurées contrairement aux sources utilisées précédemment par SIMS.

Ariadne dispose d'une interface graphique (DoUI, Demonstration-oriented User Interface) qui permet d'aider les utilisateurs à désigner les données à extraire des pages WEB [Knoblock *et al*, 98]. L'ensemble est supporté par des techniques de *Machine Learning* pour produire par induction les règles d'extraction pour la même page ou des pages similaires. La génération des règles, pour une donnée, se base sur plusieurs exemples du même champ à l'intérieur de la page.
L'équipe du projet est en train de développer des applications de test pour Ariadne, telles que l'intégration des informations sur des cartes géographiques [Knoblock *et al*, 98].

2.2 Information Broker

Information Broker est une architecture de médiation développée par le centre d'Intelligence artificielle de l'organisme de recherche SRI [URL-2-2]. Il s'appuie sur un environnement défini par l'OAA (Open Agent Architecture) proposé antérieurement par le SRI [Cohen *et al*, 94].

2.2.1 Architecture OAA

Cette architecture met en évidence une société d'agents intelligents, chacun disposant d'une grande autonomie. Cette coopération est supervisée par des facilitateurs. OAA est conçue comme une hiérarchie d'agents. Chaque agent d'application est lié à un facilitateur qui lui même peut être en relation avec un autre. Un facilitateur offre des services tels que le routage des messages, la gestion des données globales et la coordination entre agents (figure 2-3) [Martin et al, 99].

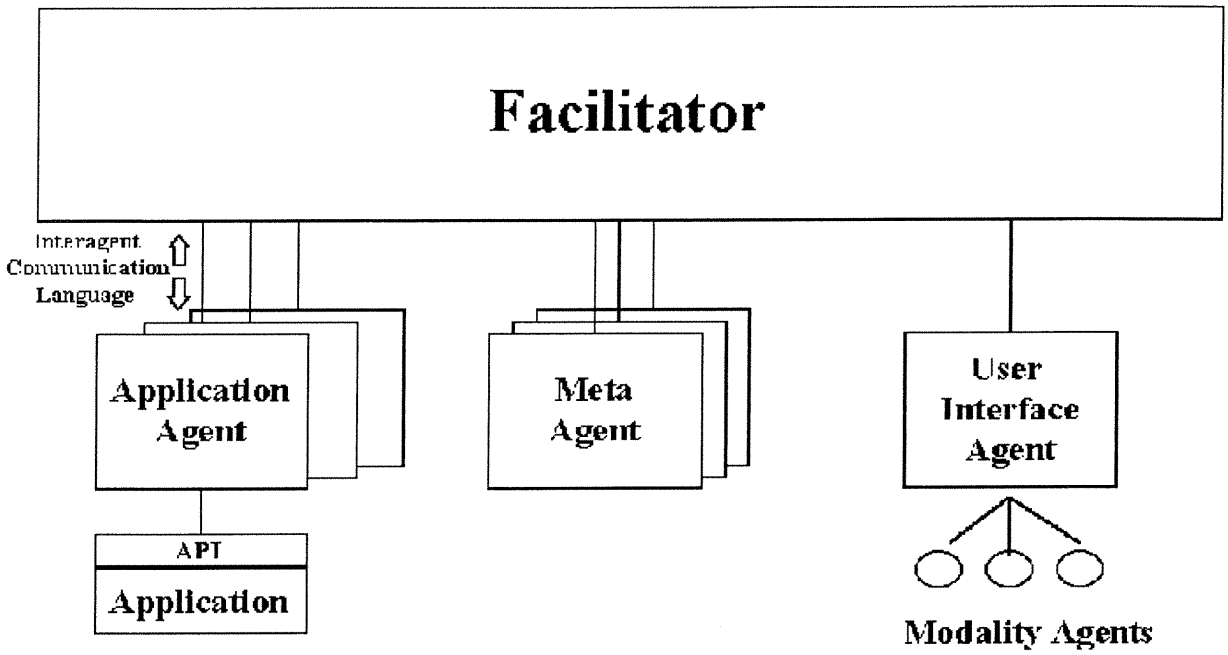


Figure 2-3 : architecture de l'Open Agent Architecture

L'interaction à l'intérieur de l'environnement se fait à l'aide du langage de communication spécifique ICL (*Interagent Communication Language*). ICL est une extension de Prolog, ce qui lui donne la puissance d'unification et de chaînage [Cohen *et al*, 94].

2.2.2 Architecture

La figure 2-4 montre les composants du système Information Broker IB, [Martin *et al*, 97]. *Information Broker Agent* en est la partie principale. Elle fournit un accès aux données de manière transparente aux clients. Les requêtes sont construites en utilisant un schéma commun propre au système. Le schéma de la figure 2-4 désigne la manière dont le *Broker* organise l'accès aux données. La syntaxe d'une requête, exprimée en ICL, ressemble à celle d'un but dans le langage des prédicats. Le *Broker* utilise un schéma de règles de correspondance pour décider quelles sources sont concernées par quelles sous-requêtes. Il réécrit chaque sous-requête en respectant le schéma de la source correspondante.

En respectant l'architecture OAA, Information Broker supporte l'enregistrement (et le désenregistrement) de nouvelles sources sans avoir des connaissances préalables sur celles-ci [Martin *et al*, 97]. Au moment de l'enregistrement, une source fournie un ensemble de règles de passage entre son schéma et celui d'Information Broker. Naturellement, le concepteur de la

source devrait avoir le schéma de représentation commun du Broker (IB). L'ajout de nouvelles sources demande un travail de plus pour les concepteurs (ou responsables) de celles-ci.

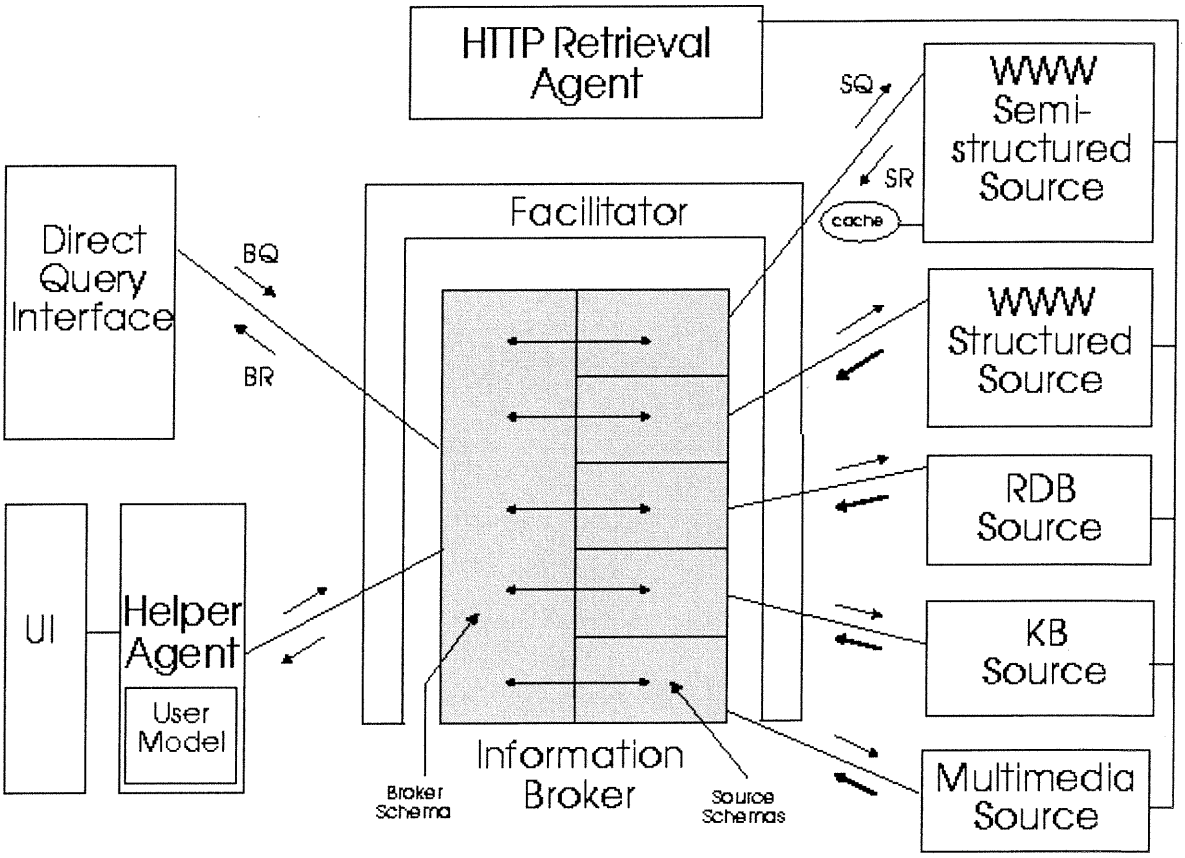


Figure 2-4 : architecture d'Information Broker

Aussi les sources sont vues comme des agents (du point de vue de l'OAA) qui empaquettent ces sources. Sur la figure les sources se trouvent dans la partie droite. BQ (Broker Query) et BR (Broker Response) désignent respectivement la requête du client et la réponse du Broker exprimées dans le schéma de celui-ci. De même, SQ (Source Query) et SR (Source Response) réfèrent aux interrogations et aux réponses pour la source écrites dans le langage d'une source.

2.2.3 Traitement des requêtes

Les règles de correspondance entre schémas ont la forme suivante :

$$B(X) \leftarrow S_1(Z_1), \ldots, S_n(Z_n)$$

Où :

- B(X) est le prédicat dans le schéma du broker.
- chaque S_i peut être :
  - un prédicat dans le schéma d'une source i ;
  - un prédicat du domaine du Broker qui concerne des connaissances spécifiques du domaine traité par le Broker ;
  - un prédicat prédéfini dans ICL
- X et les Z_i sont des variables ou des valeurs instanciées.

La procédure de résolution des requêtes est décrite par les étapes suivantes [Martin *et al*, 97] :

- pour chaque prédicat de la requête, désigner l'ensemble des sources concernées ;
- s'il s'agit d'un prédicat du broker, un test d'unification est enclenché avec les prémisses des règles (constituant le schéma de passage entre le *Broker* et les sources). Si l'unification réussit pour une règle, alors la source (ou les sources) associée avec cette règle est sélectionnée ;
- s'il s'agit d'un prédicat du domaine, le *Broker* est pris pour la source ;
- déterminer les sous-requêtes (pour la même source) qui peuvent être regroupées en une seule et traitées ainsi sans changement de sémantique. Cette procédure optimise les communications entre la source et le *Broker*, et s'il y a lieu de faire les traitements sur les données intermédiaires au niveau de la source au lieu du *Broker*.

Le maintien de la persistance des requêtes est un service intéressant du *Broker*. Etant donnée une requête qui vient d'être résolue, il observe si des changements ont été apportés au niveau des sources qui ont servi pour cette requête. Dans le cas où les données de ces sources ont changé, il reconsidère la requête. Il compare les nouveaux résultats avec ceux précédemment obtenus, dans le cas de non correspondance, il avise le demandeur de la requête de l'existence de nouveaux résultats [Martin et al, 97]. Si dans des cas, cette fonctionnalité est intéressante (requêtes sur des

actions boursières par exemple), elle peut conduire néanmoins à une grande consommation de ressources système.

Les agents sources doivent être capables de traiter les requêtes avec la syntaxe semblable à Prolog qui sont générées par le *Broker*. [Martin et al, 97] affirment que le passage de SQL à un langage similaire à Prolog est facile, mais ils n'indiquent pas de détails sur la démarche à suivre.

2.3 Infomaster

Il s'agit d'un système intégrateur d'information qui offre un accès uniforme à des sources hétérogènes et distribuées sur Internet. Les utilisateurs d'un tel système ont l'illusion d'avoir des informations homogènes centralisées [Duschka *et al*, 97b]. Le noyau d'Infomaster est un facilitateur qui détermine la manière pour répondre à la requête d'un client en se basant sur les informations qu'il détient à propos des sources à travers les *wrappers* associés à chacune d'elles [Genesereth *et al*, 97].

Infomaster gère la transformation de la structure et du contenu pour harmoniser les hétérogénéités des sources. Ces dernières peuvent être des systèmes comme Z39.50 pour l'accès aux bibliothèques digitales, des bases de données SQL, des interfaces EDI ou des données semi-structurées sur des pages WWW. L'implantation et l'expérimentation d'Infomaster a concerné plusieurs applications :

- fournir la possibilité de recherche à travers les annonces classées de location d'immobilier ;
- uniformiser l'accès aux catalogues électroniques des produits de plusieurs vendeurs ;
- intégrer l'accès à des collections de bases de données de l'université Stanford regroupant les données sur les personnes, les cours et les bibliothèques. Le consortium CommerceNet étudie la possibilité d'utiliser Infomaster pour intégrer les catalogues de ses membres [Duschka et al, 97a].

2.3.1 Architecture

La figure montre les composants du système [Genesereth *et al*, 97] :

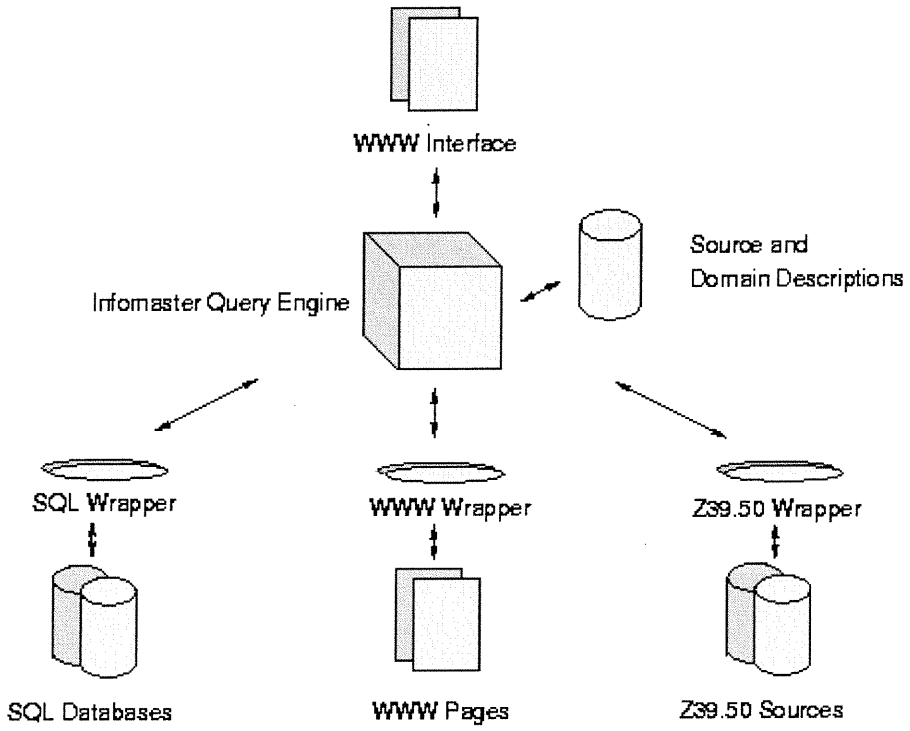


Figure 2-5 : architecture d'Infomaster

Chaque source exige un module, appelé *wrapper* [Duschka *et al*, 97b], qui assure la transformation de la requête entre le langage de requête utilisé par cette source et le langage commun utilisé par le système Infomaster. L'échange entre les composants d'Infomaster se fait à l'aide de KIF (Knowledge Interchange Format).

Infomaster utilise des règles pour décrire les sources d'information et les transformations entre celles-ci. Infomaster ne produit pas les règles de passage pour chaque paire de schémas (source-interface) ; il fournit plutôt un schéma de référence unique qui est utilisé pour décrire les règles de transition aussi bien du côté des clients que du côté des sources.
Les utilisateurs accèdent à Infomaster à travers un browser WWW. L'accès se fait avec des formulaires qui permettent d'éditer les requêtes.

2.3.2 Procédure de la transformation

L'idée de base de la reformulation des requêtes repose sur la modélisation des interfaces clientes et des sources d'information par un ensemble de relations. Par exemple, les formulaires utilisés pour éditer les requêtes sont modélisés par des *Interface Relations.* Les données des sources sont également décrites (et vues) comme des relations, appelées *Site Relations.* Le problème d'intégration est alors restreint à trouver la forme de passage des *Interface Relations* vers *Site Relations* [Duschka *et al*, 97b]. La figure 2-6 montre les deux types de modélisations et le rapport entre elles à travers une *Base Relation.*

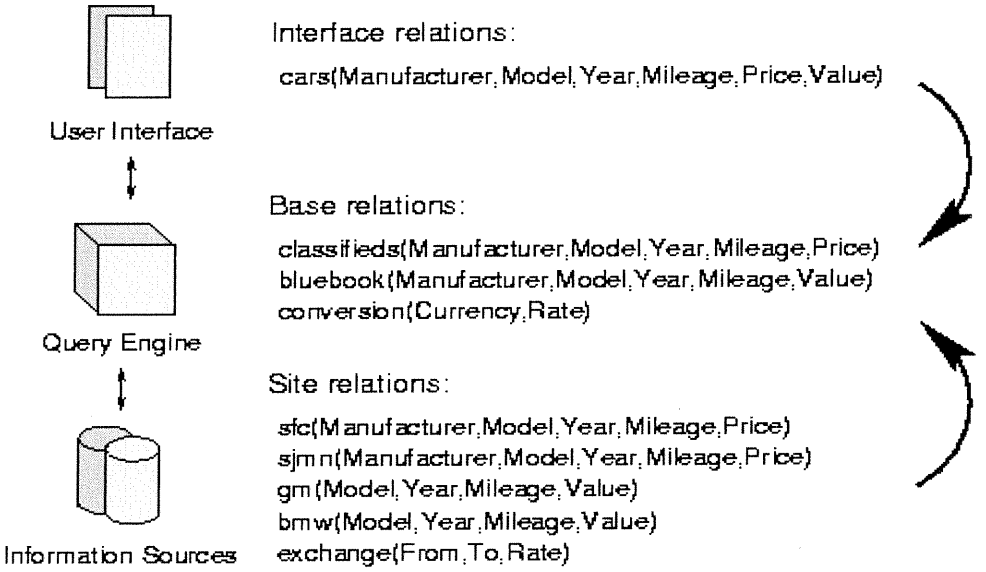


Figure 2-6 : correspondances entre relations

Nous donnerons l'exemple suivant, tel que rapporté dans [Duschka et al, 97b], pour illustrer la structuration, la construction des relations, le schéma interne ainsi que le processus de résolution des requêtes. L'exemple concerne le domaine des annonces classées pour les voitures d'occasion, les sources d'informations disponibles sont les suivantes :

- les données sur les offres de vente sont puisées de deux journaux : San Jose Mercury News et San Francisco Chronicle ;
- les informations techniques sur les véhicules et l'estimation de leurs valeurs sur le marché sont fournies par les constructeurs (ici GM et BMW).

Les données des annonces sont modélisées et représentées par les *Site Relations sfc* et *sjmn* respectivement pour San Francisco Chronicle et San Jose Mercury News. Les relations *gm* et

bmw désignent les informations des deux constructeurs. La relation *cars* est l'*Interface Relation* qui représente le formulaire WWW de la requête.

L'introduction des *Base Relations* permet de relier directement les *Site Relations* et les *Interface Relations*. Les *Site relations* et les *Interface Relations* sont exprimées alors en fonction des B*ases Relations*.

Dans l'exemple, on a choisi comme *Bases Relations* les relations *classifieds*, *bluebook* et *conversion* qui permettent respectivement de regrouper les informations des annonces classées, des données des constructeurs sur leurs modèles et des taux de change pour les devises.

La relation *cars* est une combinaison des *Base Relations* : *classifieds* et *bluebook*, ceci est exprimé comme suit :

$$cars(Manufacturer, Model, Year, Mileage, Price, Value) \equiv$$
$$classifieds(Manufacturer, Model, Year, Mileage, Price)$$
$$\& \; bluebook(Manufacturer, Model, Year, Mileage, Value)$$

Aucun des deux journaux de l'exemple ne fournit toutes les annonces de vente de voitures possibles, les relations *sfc* et *sjmn* constituent chacune une partie de la relation *classifieds* et sont donc contenues dans celle-ci.

$$sfc(Manufacturer, Model, Year, Mileage, Price) \Rightarrow$$
$$classifieds(Manufacturer, Model, Year, Mileage, Price)$$

$$sjmn(Manufacturer, Model, Year, Mileage, Price) \Rightarrow$$
$$classifieds(Manufacturer, Model, Year, Mileage, Price)$$

D'autre part, les *Site Relations gm* et *bmw* fournissent toute l'information correspondante à leurs modèles respectifs, de ce point de vue, elles sont complètes et donc équivalente aux fragments dans la relation *bluebook*. L'ajout d'autres sources d'information sur les modèles de GM et BMW serait inutile.

gm(Model, Year, Mileage, Value) ≡
bluebook(gm, Model, Year, Mileage, Value)

bmw(Model, Year, Mileage_in_km, Value_in_DM) ≡
bluebook(bmw, Model, Year, Mileage, Value)
& conversion(dm, Rate)
*& Mileage = Mileage_in_km * 1.6*
*& Value = Value_in_DM *Rate*

Les données de *bmw* sont exprimées en unités européennes km et DM, elles nécessitent une conversion en mile et USD.

Le processus de résolution d'une requête se fait en trois étapes [Duschka *et al*, 97b] : R*eduction, Abduction, Optimization* (voir annexe1 pour les détails).

2.4 Projet TSIMMIS

TSIMMIS : The Stanford-IBM Manager of Multiple Information Sources est un projet de l'université de Stanford qui a pour but de fournir les outils et mécanismes afin de faciliter l'intégration rapide des sources d'information. Il s'agit essentiellement de sources semi-structurées ou non structurées.

Le projet TSIMMIS fournit les outils pour accéder d'une manière uniforme à de nombreuses sources d'information et s'assurer que le résultat présenté est cohérent.

2.4.1 Eléments de TSIMMIS

L'architecture et les composantes du système se présentent comme suit (figure 2-7) [Chawathe *et al*, 94]:

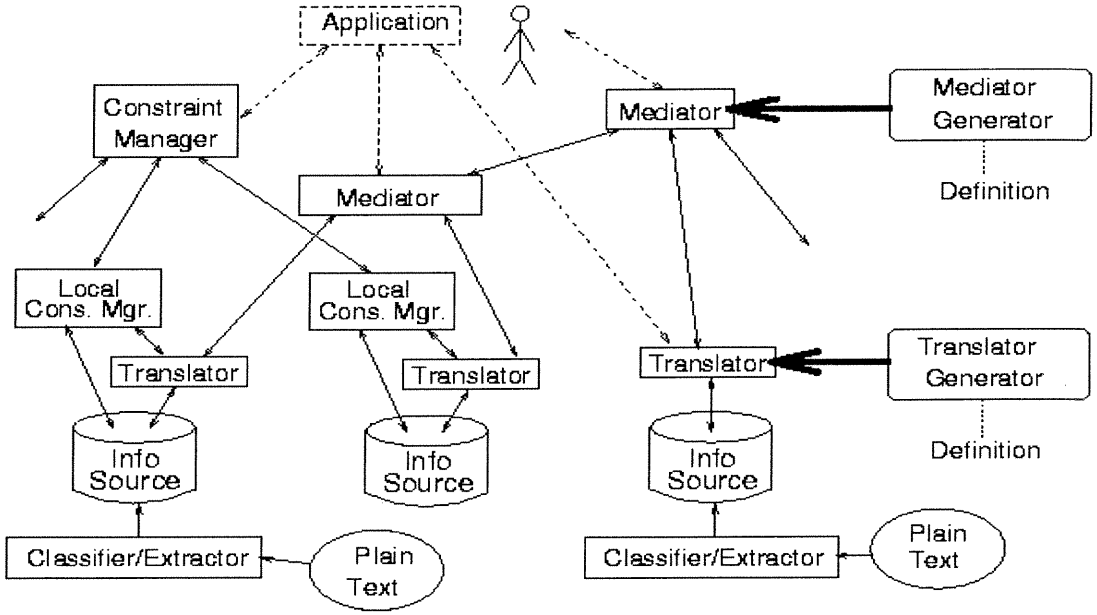


Figure 2-7 : composants du système TSIMMIS

Translator : A chaque source, est associé un traducteur appelé aussi *Wrapper*, qui constitue l'interface avec la source et qui s'occupe de convertir les objets de la source en un modèle de représentation commun. Pour ce faire, les requêtes de l'utilisateur sont traduites en des demandes que la source peut exécuter, les données retournées par la source sont à leur tour représentées dans un modèle commun. Ce dernier n'étant autre que le OEM (Object Exchange Model), un objet est décrit en des sous-objets ayant des labels qui désignent la signification de chacun (voir plus de détail sur OEM plus loin).

Mediator : Au-dessus des *Translators*, se trouvent les médiateurs (*Mediators*). Un médiateur est un système qui raffine l'information d'une ou de plusieurs sources. Il a la connaissance nécessaire pour traiter un type spécifique d'information. Par exemple, un médiateur spécialisé dans les événements et les nouvelles pourrait savoir que les sources de nouvelles les plus appropriées d'information sont l'agence *Reuter*, et le *New York Times*. Quand le médiateur reçoit une requête, il saura où expédier la requête. Le *Mediator* traite l'information survenue de plusieurs sources afin d'éviter par exemple la redondance [Chawathe *et al*, 94]. Cette vérification de la redondance est une tâche complexe. Dans le cas de l'exemple des sources d'information concernant les

nouvelles, il difficile de déterminer si deux articles (écrits par deux auteurs différents) parlent de la même chose.
Les utilisateurs et les *Mediators* obtiennent l'information de la part des *Translators* ou d'autres *Mediators*.

Constraint Manager et le *Local Constraint Manager* ont le rôle de gérer les contraintes dans un environnement distribué. Le respect de certaines contraintes vis-à-vis des requêtes des utilisateurs sont essentielles, surtout si les sources sont faiblement couplées [Chawathe *et al*, 94]. Par exemple, les données sur les édifices en construction doivent être en concordance avec celles de l'architecte qui a conçu les plans ; les deux types de données se trouvant sur des sources distribuées [Chawathe *et al*, 94].

Classifier/Extractor : son rôle est de reconnaître les propriétés clés à partir des informations non structurées pour construire des objets OEM par la suite. Le *Classifier/Extractor* est basé sur le système RUFUS développé au IBM Almaden Research Center [URL-2-1]. RUFUS est concu essentiellement pour le traitement des documents textuels (fichiers texte, courrier, ...). Il extrait et indexe automatiquement le contenu de ces documents. [Chawathe *et al*, 94] n'indiquent pas plus de précisions sur la manière de procéder.

Mediator Generator : L'un des buts du projet TSIMMIS est d'automatiser ou de semi automatiser la création de *Mediators* à partir d'une description formelle du traitement envisagé [Papakonstantinou *et al*, 95]. Cette approche de génération automatique du *Mediator* ainsi que du *Translator* n'ont pas été implantées.

Les requêtes sont envoyées, par des applications clientes ou à partir de formulaires WEB, aux *Mediators* et/ou *Translators* sous forme de OEM-QL, un langage de requêtes semblable à SQL conçu pour les objets OEM. La partie suivante décrit OEM et OEM-QL plus en détail [Papakonstantinou *et al*, 95].

2.4.2 Modèle de représentation des objets dans TSIMMIS

Le modèle de représentation de TSIMMIS sert comme modèle commun de l'information traitée par l'ensemble des composants du système TSIMMIS. Ce modèle est utilisé pour répondre aux requêtes et non pas pour stocker les données. Chaque objet OEM est structuré comme suit :

Label	Type	Value	Object-ID

Par exemple :

(employe, set, {o1, o2, o3})
o1 : (name, string, "Employe_Name")
o2 : (office, string, "Employe_Office")
o3 : (name, bitmap, "Employe_photo")

Label : est une chaîne de caractères qui décrit l'entité représentée par l'objet ;

Type : c'est le type de la valeur *Value* de l'objet. Ce type peut être atomique (tels que Integer, String, real) s'il n'est pas décomposable, il peut être complexe (set/complex) s'il est constitué de plusieurs autres sous-objets ;

Value : ce champ contient la valeur de l'objet ;

Object-ID : c'est l'identification de l'objet qui assure l'unicité de la référence. [Papakonstantinou *et al*, 95] proposent que cet identifiant comporte le nom de la source comme préfixe pour distinguer entre objets de sources différentes ayant le même Label.

OEM est une convention simple de représentation qui permet en particulier l'identification et l'empaquetage d'objets. Il renferme une certaine sémantique sur les données puisque OEM est auto-descripteur [Chawathe *et al*, 94].

Afin de récupérer des objets OEM, le client utilise un langage de requêtes appelé OEM-QL. La structure de OEM-QL est semblable à SQL.

2.4.3 Extraction des données (WEB)

Un module d'extraction pour les informations semi-structurées (WEB) a été développé dans le cadre du Système TSIMMIS. Ce module se base sur une spécification déclarative indiquant l'emplacement de l'information à l'intérieur des pages HTML et la manière avec laquelle les données récupérées vont être stockées sur une base de données [Hammer *et al*, 97].

Les données sont repérées dans le texte à l'aide de patrons textuels délimiteurs, qui annoncent le début et la fin de telles données. Il y a pas de technique d'intelligence artificielle pour la compréhension du texte. L'analyseur syntaxique pour les fichiers HTML est basé sur le langage Python [URL-2-3].

Les spécifications sont écrites dans un fichier structuré en une séquence de commandes, chacune définit une étape de l'extraction. Une commande a la forme suivante [Hammer *et al*, 97]:

[variables, source, patron]

où :

- *variables* sont l'ensemble des variables qui vont recevoir le texte extrait à la fin. Elles peuvent aussi être utilisées pour passer du texte entre les commandes ;
- *source* spécifie le texte à considérer ;
- *patron* indique comment trouver le texte désiré dans la source.

Nous introduisons ci-après un exemple de fichier HTML et un fichier de spécification de commande correspondant [Hammer *et al*, 97]. Il s'agit d'une page qui regroupe un certain nombre de données météorologiques dans certaines villes d'Europe.

```
...

<TABLE CELLSPACING=0 CELLPADDING=0 WIDTH=514>
        <TR ALIGN=left>
                <TH COLSPAN=2><BR></TH>
                <TH COLSPAN=2><I>Tue, Jan 28, 1997</I></TH>
                <TH COLSPAN=2><I>Wed, Jan 29, 1997</I></TH>
        </TR>
        <TR ALIGN=left>
                <TH><I>country</I></TH>
                <TH><I>city</I></TH>
                <TH><I>forecast</I></TH>
                <TH><I>hi/lo</I></TH>
                <TH><I>forecast</I></TH>
                <TH><I>hi/lo</I></TH>
                </TR>
        <TR ALIGN=left>
                <TD>Austria</TD>
                <TD><A HREF=http://www.intellicast.com/weather/vie/>Vienna</A></TD>
                <TD>snow</TD>
                <TD>-2/-7</TD>
                <TD>snow</TD>
                <TD>-2/-7</TD>
        </TR>
        <TR ALIGN=left>
                <TD>Belgium</TD>
                <TD><A HREF=http://www.intellicast.com/weather/bru/>Brussels</A></TD>
                <TD>fog</TD>
                <TD>2/-2</TD>
                <TD>sleet</TD>
                <TD>3/-1</TD>
        </TR>
</TABLE>
...
```

Le fichier de spécifications se présente comme suit :

```
1 ["root", "get('http://www.intellicast.com/weather/europe/')", "#" ],

2 ["temperatures", "root", "*<TABLE*</TR>#</TABLE>*" ],

3 ["citytemp", "split(temperatures,'<TR ALIGN=left>')", "#" ],

4
["country,c_url,city,weath_tody,hgh_tody,low_today,weath_tomorrow,hgh_tomorrow,low_tomorrow
",
  "city_temp",

"*<TD>#</TD>*HREF=#>#</A>*<TD>#</TD>*<TD>#/#</TD>*<TD>#</TD>*<TD>#/#*" ]
```

La première commande indique que la variable root reçoit le contenu du fichier HTML téléchargé à partir de l'adresse fournie par get. La variable *temperatures* de la deuxième commande contiendra un texte extrait de la variable root et qui obéit à la séquence spécifiée par le patron "*<TABLE*</TR>#</TABLE>*". En particulier "*" indique un texte à ignorer et "#" indique le texte voulu. La troisième commande construit un tableau avec les éléments dans *temperatures* qui sont séparés par la séquence "<TR ALIGN=left>". Enfin, la dernière commande extrait des éléments du tableau *city_temp* les valeurs des variables finales.

Le résultat de ce traitement est un objet OEM :

```
root complex {
        temperature complex {
            city_temp complex {
                country          string   ‘‘Austria”
                city_url         url      http://www...
                city             string   ‘‘Vienna”
                weather_today    string   ‘snow”
                high_today       string   ``-2"
                low_today        string   ``-7"
                weather_tom      string    ``snow"
                high_tomorrow    string    ``-2"
                low_tomorrow     string   ``-7"
                        }
            city_temp complex {
                country          string   ``Belgium"
                city_url         url      http://www...
                city              string  ``Brussles" ... } ... } }
```

La syntaxe des commandes d'extraction permet, pour avoir une donnée particulière, de prendre en compte l'existence de plusieurs possibilités quant à l'emplacement d'une donnée dans le texte.

Un prototype de ce système a été testé, (avec le projet TSIMMIS) dans le cadre du DARPA I3 (Intelligent Integration of Information), pour extraire les données météorologique à partir de plusieurs sites WEB [Hammer *et al*, 97].

2.4.4 Traitement des requêtes

Une demande passe par trois étapes : identifier les sources, ordonner les sous requêtes, optimiser les requêtes. À la fin, un plan d'exécution est généré [Li *et al*, 98].

Pour illustrer la procédure de traitement de requêtes par TIMMIS, nous allons reprendre l'exemple tel que décrit dans [Li *et al*, 98]. Il s'agit de deux sources S1 et S2. La première offre des informations sur le titre, l'auteur et le résumé des articles de la conférence SIGMOD-97. La deuxième donne les informations sur les conférences et les titres des articles de SIGMOD-97.
En supposant que S1 et S2 soient les seules sources disponibles, une requête concernant les articles est écrite comme suit :

```
< paper {<title T> <author A> <abs B> <conf C> } > :-
        < entry {<title T> <author A> <abs B> } > @S1,
        < entry {<title T> <conf C> } > @S2
```

Afin de récupérer le titre et le résumé des articles présentés par SMITH dans SIGMOD-97, on écrit :

```
<ans { <title T> <abs B> }> :-
        < paper {<title T> <author "SMITH"> <abs B> <conf "SIGMOD-97"> } >
```

Ce qui revient à écrire le plan logique pour la résolution en réécrivant *paper* en terme de *entry* :

```
<ans { <title T> <abs B> }> :-
        < entry {<title T> <author "SMITH"> <abs B> } > @S1,
        < entry {<title T> <conf "SIGMOD-97"> } > @S2
```

TSIMMIS décrit les capacités des sources S1 et S2 à répondre à cette requête. Pour cela, il génère des *Templates* pour chaque requête susceptible d'être résolue par S1 et S2 :

T11 : <entry { <title $T> <author A> <abs B> } > @S1
T21 : <entry { <title T> <conf $C > } > @S2
T22 : <entry { <title $T> <conf C > } > @S2

Cela signifie en particulier que :

- S1 est mesure de fournir les informations (titre et résumé) pour un article en ayant comme entrée son titre ;
- Étant donné le nom de la conférence, S2 est capable de retrouver tous les titres des articles qui y sont présentés ;
- Étant donné le titre d'un article, S2 est en mesure de fournir le nom de la conférence correspondante.

La résolution finale des requêtes est faite en identifiant les séquences de *Templates* pouvant conduire aux résultats recherchés à partir des données disponibles.

L'implantation et la structuration de OEM (et de TSIMMIS) ont été testées pour intégrer différentes sources d'information bibliothécaires avec des *Translators* pour accéder à des librairies de l'université de Stanford et à d'autres sources comme INSPEC. L'implantation a concerné aussi des *Mediators* qui traitent les données issues de plusieurs sources et les présente comme ayant une seule origine. Par exemple, un *Mediator* fait une union des informations ou une jointure en éliminant la redondance des "entrées" qui réfèrent au même document [Chawathe *et al*, 94].

2.5 Projet InsoSleuth

Infosleuth est un système collaboratif d'agents constituant une infrastructure de communication et de coopération afin de rendre l'accès facile à un ensemble de bases de données à travers le WEB [Nodine *et al*, 98a]. Les agents d'Infosleuth gardent les descriptions des ressources (bases de données) qui leurs sont associées et participent à la résolution des requêtes d'un utilisateur de manière distribuée [Woelk and Tomlinson, 95]. CARNOT est un projet antérieur à InfoSleuth et sur lequel se base ce dernier [Nodine *et al,* 98b]. Le projet CARNOT offre des techniques de modélisation de la sémantique pour l'intégration des informations statiques. CARNOT n'a pas

été conçu pour des environnement dynamiques où les sources d'information changent durant le temps (disparition et apparition). Infosleuth étend les fonctionnalités de CARNOT dans ce sens. InfoSleuth est développé par MCC Microelectronics and Computer Technology Corporation [URL-2-4]

La description d'InfoSleuth montre qu'il s'appuie sur les techniques suivante [Bayardo *et al*, 96] :

- *modèles de domaines* (ou ontologies) : ils définissent une description uniforme et déclarative de la sémantique de l'information, indépendamment de la syntaxe de représentation ou des modèles conceptuels. Les ontologies ont été introduites pour représenter sémantiquement l'information liée à un domaine spécifique. En d'autres termes, une ontologie implique un ensemble de termes et des liens entre eux, elle décrit les termes et la façon avec laquelle il sont (ou ne sont) pas liés entre eux.
- *information Brokerage* : des agents spécialisés (Broker Agents) établissent la correspondance entre les besoins (spécifiés en terme d'ontologies) avec les ressources disponibles. Les requêtes sont alors directement acheminées aux ressources adéquates [Bayardo *et al*, 96].

2.5.1 Architecture

Infosleuth est constitué d'un réseau d'agents qui coopèrent et communiquent entre eux en utilisant le langage KQML (Knowledge Query Manipulation Language). KIF est employé pour la représentation des connaissances.

Les *Ressource Agents* informent les autres agents de leurs capacités de fournir des services particuliers. La décision d'acheminement des requêtes se base sur les ontologies qui décrivent les connaissances des agents et les relations entre eux. Les requêtes sont routées vers les agents ressources appropriés. L'architecture de l'environnement InfoSleuth est décrite par la figure suivante (figure 2-8) [Bayardo *et al*, 96]:

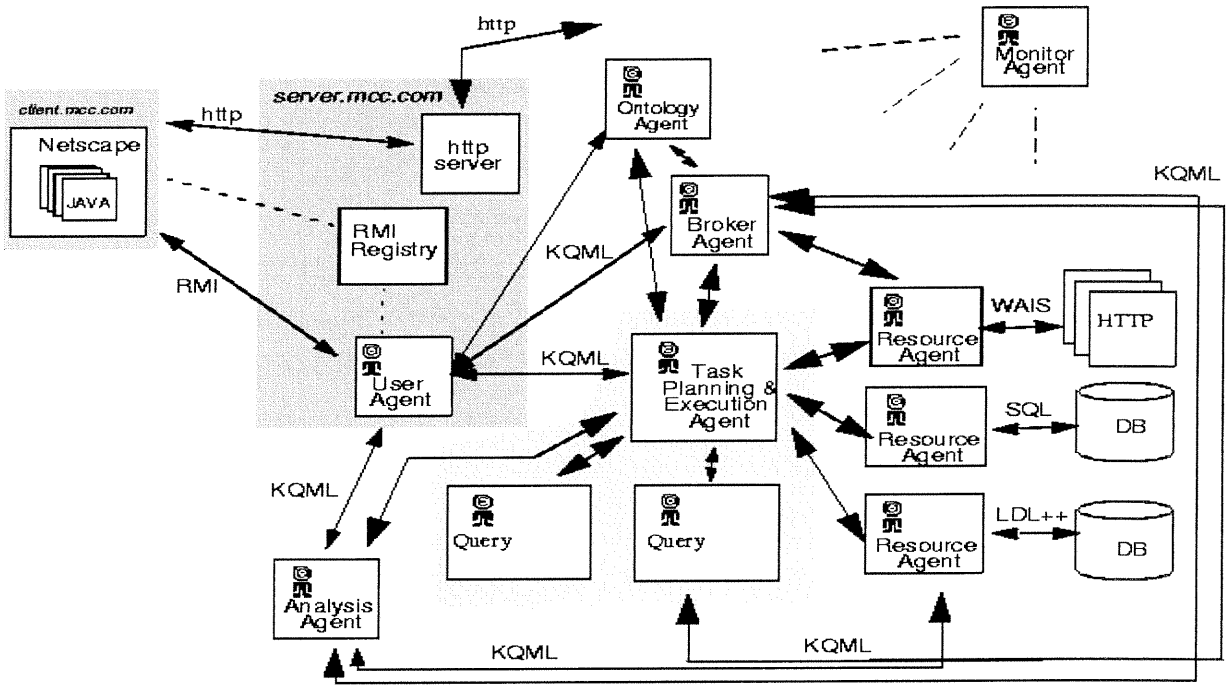


Figure 2-8 : environnement de InfoSleuth

User Agent : il est l'interface de l'utilisateur vers InfoSleuth, il l'assiste pour la formulation des requêtes en se basant sur les ontologies des domaines choisies par l'utilisateur. L'agent présente les résultats une fois obtenus.

Broker Agent : reçoit les annonces des agents concernant leurs capacités à fournir des services. Basé sur ces informations reçues, il indique aux agents demandeurs de services les agents qualifiés pour répondre adéquatement.

Ressource Agent : son rôle est de rendre l'information contenue dans la source disponible pour la récupération et la mise à jour. Il est l'interface entre les données locales et l'ensemble des agents ; il masque les spécificités de l'organisation et de la représentation locale. La description du contenu de la ressource concerne les métadonnées sur l'information, c'est-à-dire l'ensemble des noms des objets connus au *Ressource Agent*, les valeurs ou plages de valeurs de ces objets et l'ensemble des opérations permises sur les données. Le *Ressource Agent* traduit les requêtes (spécifiées en KIF/KQML) en une description locale. Cela est possible grâce au lien entre les termes et concepts de l'ontologie et de la ressource.

Ontology Agent : c'est un *Ressource Agent* spécialisé. Il répond aux requêtes concernant les ontologies : la liste des ontologies, la source d'une ontologie et la recherche d'un concept parmi les ontologies disponibles.

Task Execution Agent : il coordonne les sous-tâches nécessaires pour une requête, il achemine les sous-requêtes vers les *Ressource Agents* désignés par le *Broker Agent,* il assemble les réponses et les renvoie au *User Agent.*

Monitor Agent : Permet de la visualisation des interactions et des étapes de l'exécution.

Les agents communiquent via des primitives KQML [Finin *et al*, 97], cela inclut les requêtes (*evaluate, ask-one, ask-all*), les annonces (*advertise, subscribe, monitor*). KQML est utilisé comme un langage de transport et de support pour un autre langage commun de représentation des connaissances : KIF.

InfoSleuth a été utilisé dans le domaine médical et spécialement pour offrir l'accès aux administrateurs de la santé aux différentes données distribuées au niveau des bases de données des hôpitaux [Bayardo *et al*, 96]. Le but étant de fournir un maximum d'informations sur les expériences (données sur les séjours des patients, des complications, ...) des hôpitaux et des médecins afin de réduire les coûts et améliorer la qualité des soins.

2.6 WEB Mining Agent

WMA Web Mining Agent est un système développé à l'université d'Ottawa et qui a pour objectif d'automatiser l'extraction d'information des sites WEB [Ouahid and Karmouch, 99b] [Ouahid and Karmouch, 99c]. En particulier WMA permet de :

- extraire automatiquement l'information du Web ;
- structurer cette information de telle façon qu'elle soit stockée et manipulée par la suite à l'aide d'une base de données relationnelle ;
- maintenir la base de données à jour en fonction des changements survenus dans les sites Web d'où l'information a été extraite.

WMA se base sur WONDEL Web Ontology DEscription Language (développé aussi dans le cadre du projet), un langage de description qui permet à l'utilisateur de décrire la structure des

pages HTML et qui servira à produire des ontologies pour les sites WEB. WONDEL tire profit de certaines uniformités respectées par les concepteurs des sites WEB pour la production des pages HTML. En d'autre terme, souvent des informations répétitives sont présentées dans un format unique, par exemple : des catalogues de produits, les nouvelles dans les journaux électroniques, ...

2.6.1 L'architecture su système

WMA est constitué de quatre composantes : HTML2XML Converter, WONDEL Interpreter, Database Adapter, Web Change Detector (voir figure 2-9) [Ouahid and Karmouch, 99d].

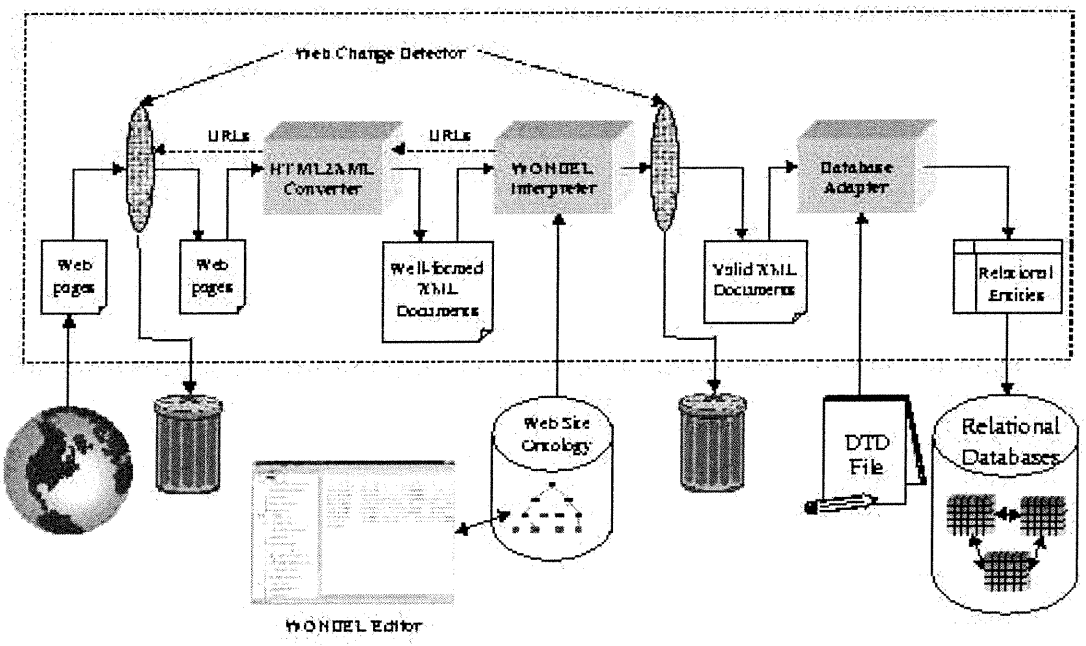


Figure 2-9 : architecture de WMA

HTML2XML Converter s'occupe de la conversion des pages HTML en des documents XML corrects. Vu que la spécification de HTML est différente de celle de XML et que des documents HTML peuvent être visualisés même s'ils contiennent des erreurs de syntaxe, HTML2XML tente de corriger les ambiguïtés et les erreurs pour avoir des documents XML bien écrits et ne pas induire en erreur le composant suivant qui se base sur les spécifications originales de HTML pour traiter les pages [Ouahid and Karmouch, 99a].

WONDEL Interpreter extrait l'information désirée et la regroupe dans un document XML. La connaissance nécessaire pour WONDEL afin d'extraire l'information désirée est décrite dans des documents WONDEL [Ouahid and Karmouch, 99c]. En particulier, ceux-ci montrent

l'emplacement des pages WEB, la position des informations à récupérer et finalement comment organiser les données en sortie. Quand il s'agit d'informations qui sont comprises dans plusieurs pages (dans le cas du traitement d'un site WEB), le regroupement de ces informations fait appel un à arbre de documents WONDEL. L'arbre reflète la hiérarchie du site, chaque feuille contient l'ontologie qui décrit la manière pour traiter une page HTML (ou plusieurs, si elles se ressemblent), un nœud de l'arbre contient les information générales sur les feuilles-documents qui lui sont attachées [Ouahid and Karmouch, 99d].

Database Adapter : utilise un algorithme qui utilise les DTD Document Type Definition pour générer les relations entre les entités du document XML. Ces données sont ensuite stockées dans une base de données relationnelle.

WEB Change Detector : Le dernier composant est responsable de la consistance de la base de données vis à vis des sources d'information.

2.6.2 Application de WMA

Le système a été appliqué pour rassembler des données sur des universités [Ouahid and Karmouch, 99d] : le personnel, les départements, les projets, les groupes de recherche ... Pour chaque université, l'arbre de documents WONDEL est construit à partir des feuilles en remontant vers la racine. Une fois construit, l'arbre est exploré par WMA en *depth-first*. Au fur et à mesure qu'une feuille est traitée (charger la page HTML et en extraire les données), le document XML qui regroupe toutes les informations concernant une université est alimenté.

2.7 Autres projets

2.7.1 WebFindIt

L'architecture proposé par WebFindIt de Queensland University of Technology définit un environnement pour l'intégration des bases de données disponibles sur le WEB [Boughettaya *et al*, 99]. WebfindIt introduit deux nouveaux concepts par rapport aux autres systèmes, il s'agit de la *coalition* et le *service link*. La première réfère à un ensemble de bases de données capables de fournir des informations dans un domaine particulier. Ces *coalitions* sont reliées entre elles par l'intermédiaire d'un *service link*. Ce dernier indique les ressources disponibles pour répondre aux requêtes qui ne peuvent être traitées localement (au niveau de la *coalition*).

A chaque base de données, est associée une co-database qui contient des méta-données sur ladite base. En particulier, elle indique le modèle des données, le système d'exploitation, le langage de requêtes, ...

WebTassili est le langage de requêtes utilisé par WebfindIt. Il permet la manipulation des données (sur les bases de données) et de méta-données (sur les co-databases). Un prototype du système a été testé dans le domaine de la recherche de la santé.

2.7.2 ARANEUS

Le projet ARANEUS conduit par l'universita di Rom Tre vise à gérer et à restructurer les pages WEB. L'approche est basée sur la notion de vues (issue des bases de données) et essaie de généraliser cette notion pour couvrir les document WEB. ARANEUS offre un certain nombre de mécanismes pour transformer un site WEB en un site offrant une vue "hypertextuelle". Le processus passe par trois étapes : 1) l'extraction des données des pages WEB, 2) l'intégration des données issues de différents sites (pour cela, les données sont stockées au niveau d'une base de données, ce qui permet d'utiliser les mécanismes de restructuration, d'intégration et de réorganisation offerts par les bases de données) et enfin 3) la régénération des pages WEB. Durant cette dérnière phase, des méta *Tags* (balises) sont insérés pour décrire la sémantique des éléments des pages afin de supporter les recherches ultérieurement.

2.7.3 Produits / Projets pratiques

PartNet est une compagnie de développement de logiciels, spécialisée dans le commerce sur Internet [URL-2-6]. Son produit PartNet eCommerce permet à plusieurs vendeurs d'être présents à travers une même adresse WEB (figure 2-11). Le département de la défense DOD (USA) utilise ce produit pour son marché virtuel DOD EMALL. Il s'agit de donner l'accès à des acheteurs du DOD et des autres administrations fédérales aux produits de plusieurs vendeurs associés avec le EMALL et présents derrière une seule vitrine. Le système est composé de deux parties : eBroker et ePort. La eBroker reçoit la requête de l'utilisateur et l'achemine vers les bases de données susceptibles d'avoir les données voulues selon le dictionnaire de données qu'il détient sur les sources. Il convertit ensuite les résultats en une page HTML qui combine les données de chaque vendeur. Le ePort interface la base de données du vendeur et exécute les requêtes issues du eBroker sur les données locales.

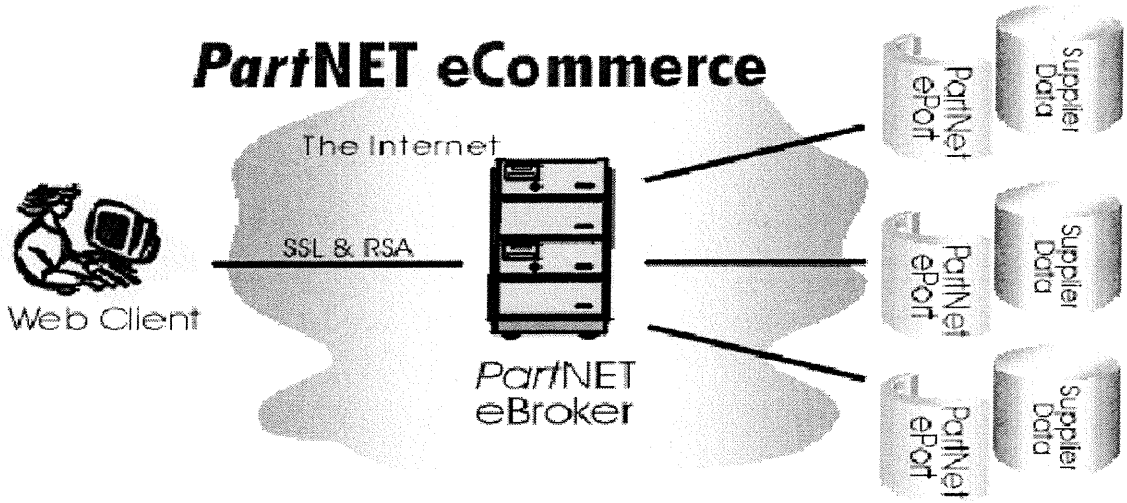


Figure 2-10 : PartNet

CHAPITRE 3

Synthèse et conception générale

3.1 Synthèse de l'état de l'art

Nous allons revoir dans cette partie l'ensemble des travaux et effectuerons une synthèse générale sur les procédés utilisés pour les différentes recherches et les conclusions qui peuvent être tirées afin de s'en servir dans la conception de notre architecture. Cela concernera en particulier les types de sources, la modélisation du domaine, la modélisation des sources et l'interrogation des données extraites.

3.1.1 Types de sources prises en considération

Nous avons remarqué que presque tous les travaux que nous avons examinés prennent les bases de données et les bases de connaissances comme sources principales d'information (voir tableau 3-1). Pour les recherches qui ont débuté vers les années 93/94, les bases de données et les bases de connaissances étaient les seules sources considérées dans leurs systèmes. Par la suite, on a inclut les sources semi-structurées, notamment les pages WEB pour faire partie intégrante des systèmes.

D'autres types de sources ont été aussi incluses telles que des sources multimédia (Information Broker, le détail concernant la gestion de ce type de sources n'a pas été donné) ou Z39.50

(Infomaster). Seul le projet TSIMMIS a pris en considération des sources non structurées comme sources principales dans son architecture.

Type de source	Structurée	Semi structurée	Non structurée
SIMS-Ariadne	DB	Pages WEB	-
Information Broker	DB, KB	Pages WEB	Multimédia
Infomaster	DB, Z39.50	Pages WEB	-
TSIMMIS	-	-	Texte
InfoSleuth	DB	Pages WEB	-
WMA	-	Pages WEB	-

Tableau 3-1 : sources prises en considération

3.1.2 Modélisation des sources

Une condition nécessaire pour pouvoir interroger une source est de disposer d'une description syntaxique et sémantique (Tableau 3-2).

Pour les sources structurées, il s'agit de transformer le schéma conceptuel de la source en un autre qui offre l'accès aux données pertinentes. Pour la plupart des systèmes, un nouveau schéma est une relation ou vue (en terme des bases de données). Cette nouvelle structure offre une description explicite ou déclarative de la source. Lorsqu'il s'agit d'une description déclarative, le système essaie d'identifier les possibilités d'interrogations que la source offre à travers des vues (ou des relations) construites au dessus de cette source. Si une description explicite est utilisée, alors le système décrit aussi le contenu de cette source dont il lie les différentes entités avec une description commune du domaine. Les systèmes qui utilisent cette description (InfoSleuth et SIMS) appliquent en général un modèle du domaine d'application commun aux sources traitées et dont on se sert pour décrire celles-ci [Arens *et al*, 93] [Nodine *et al*, 98a].

Quand il s'agit d'interagir avec des sources non structurées ou semi-structurées, c'est le système qui crée une modélisation pour ces sources (WMA, Ariadne). Cette modélisation est alors compréhensible par l'ensemble des éléments du système. A l'inverse des sources structurées, aucune transformation n'est nécessaire, après cette modélisation, pour les sources non structurés ou semi-structurées.

	Modélisation des sources	Modèle du domaine	Représentation des connaissances/objets
SIMS-Ariadne	LIM	LOOM	LOOM
Information Broker	Prédicats	Prédicats	-
Infomaster	Relations	Relations	KIF
TSIMMIS	RUFUS	-	OEM
InfoSleuth	Ontologies	Ontologies	KIF
WMA	WONDEL	-	XML

Tableau 3-2 : modélisation des sources et du domaine

3.1.3 Représentation des connaissances

Peu de projets revus dans ce document ont développé de nouveaux formalismes pour la représentation et l'échange des objets/connaissances. L'OEM proposé par TSIMMIS offre un formalisme flexible aussi bien pour construire une structure des sources ou pour décrire le résultat d'une interrogation. Le WMA utilise XML pour reconstituer et faciliter la réécriture des données des pages HTML sur une base de données relationnelle. Les autres projets utilisent des formalismes de représentation des connaissances existants, notamment KIF et LOOM.

	Langage de communication	Type d'architecture
SIMS-Ariadne	-	-
Information Broker	ICL	OAA
Infomaster	KQML	Médiation
TSIMMIS	-	Médiation
InfoSleuth	KQML	Agents
WMA	-	Agent

Tableau 3-3 : environnement de communication

3.1.4 Transformations entre schémas

Dans la majorité des cas, l'utilisateur dispose d'une interface HTML pour éditer sa requête. De plus, certains systèmes (par exemple, *Information Broker*) offrent une interface d'interrogation

aux applications clientes en utilisant le formalisme de requêtes interne du système. Ces interfaces de requêtes sont conçues de manière à faciliter la réécriture directement en terme de prédicats ou de relations dépendamment de la nature interne du système. Une transformation est par contre nécessaire pour passer du schéma interne aux vues construites sur les sources. Ce passage est exprimé souvent en des jointures de prédicats ou de relations.

Il est à remarquer que si le schéma interne est écrit en fonction des schémas des sources, la résolution des requêtes est aisée car il s'agirait d'effectuer des déductions directes. Cependant, l'ajout d'une nouvelle source n'est pas automatique car il faut la prendre en charge au niveau du schéma conceptuel commun du système.

Dans le cas où les schémas des sources sont décrits en fonction de celui du système, l'ajout de nouvelles sources est plus aisé. Néanmoins, la résolution des requêtes n'est pas trivial du moment où des techniques comme l'abduction (projet TSIMMIS) sont utilisées [Chawathe *et a*, 94]. Cette méthode fait appel à des hypothèses et n'offre pas une résolution correctement logique de la requête.

Pour les systèmes qui utilisent les bases de données pour stocker les données des sources, les requêtes sont directement adressées à la base et ne posent pas de problème de reformulation de requêtes puisqu'elles ne sont pas adressées aux sources.

L'ajout de nouvelles sources nécessite le développement d'un *Wrapper* ou interface pour interagir avec celles-ci. Dans ce sens, certains systèmes ont développé des outils pour aider à intégrer de nouvelles sources de manière semi automatique. Ces techniques sont destinées principalement pour les sources semi-structurées : les outils employés se basent soit sur l'apprentissage machine, soit sur des formalismes pour décrire une délimitation spatiale des données à extraire. Cette phase d'ajout de nouvelles sources est nécessaire, nous la retrouvons pratiquement dans tous les systèmes ; parfois elle est facilitée par des outils d'aide.

3.1.5 Nécessité d'un domaine d'application pour intégrer des sources

L'intégration des données à partir de plusieurs sources suppose que les informations présentées par ces sources font partie du même domaine. Par exemple, les données de sources qui fournissent les cours des actions en bourse, les taux de change, l'historique financier des compagnies et leurs résultats financiers sont susceptibles d'être intégrées et mises entre elles. De même, les données météorologiques, les destinations touristiques et les informations sur les

moyens de transport et de voyages conviennent d'être intégrées. Ce qui permet de fournir des données complètes sur un même domaine (financier, touristique...). Cette nécessité de définir un domaine d'application est traduite de manière pratique par une modélisation du domaine.
L'application à un domaine spécifique donne également l'occasion de traiter les demandes des utilisateurs de façon plus approfondie et permet de construire des services à valeurs ajoutées pour répondre à des exigences particulières des utilisateurs. Il faut noter aussi que plus le domaine est précis plus on a tendance à avoir une description explicite des sources par le moyen d'un modèle du domaine. L'utilisation d'un domaine restreint donne l'occasion à un traitement de l'information plus spécialisé et plus précis. Ceci est dû au fait que l'on peut supposer et partir de "données sur les données" (méta-données) propres du domaine.

3.2 Architecture et éléments généraux de notre conception

3.2.1 Types de sources

Parmi les types de sources de données qui existent, nous allons nous intéresser particulièrement aux bases de données et aux informations présentées sur des pages/sites WEB. C'est à dire les sources structurées et les sources semi-structurées. Actuellement, il existe beaucoup de sites qui fournissent ce genre d'informations et sur lesquelles on peut faire des traitements élaborés. Par exemple, les sites tels que Yahoo Finance (URL-3-1) et xRates (URL-3-2) affichent des pages WEB qui sont reproduites de bases de données, elles autorisent ainsi ce type de traitement.
Le traitement des sources non structurées permet actuellement de proposer des documents qui nécessitent un traitement ultérieur de la part de l'utilisateur. Elles n'offrent pas des données précises et exactes qui peuvent être utilisées par des systèmes. D'autres part, la tendance actuelle tend à favoriser la création de sources structurées ou semi-structurées plus que des sources non structurées. Nous nous limitons aux sources structurées et semi-structurées.

Les bases de données sont des sources principales de données structurées. Il est donc naturel de les considérer lors de la conception de système pour l'intégration des données.
Actuellement, le WEB est devenu un moyen incontournable pour la publication d'informations et de données. Toute intégration de données doit être en mesure de prendre les pages WEB comme sources de données à intégrer et explorer les possibilités d'extraction relatives à ce type de sources (dites semi-structurées)

Les caractéristiques de ces sources considérées imposent une manipulation différente pour chacune d'elles. Si l'on est en mesure de faire des requêtes sur des bases de données, étant donné

que son schéma est connu, l'extraction des données d'une source semi-structurée est plus difficile. En effet, les pages WEB ne sont pas "interrogeables" et nécessitent une restructuration afin de pourvoir localiser les données sur la pages et de les extraire par la suite.
La conception que nous proposons concernera les deux types de sources, tandis que nous ferons l'implantation pour les sources WEB uniquement. Ce choix est motivé par le fait que l'on sait comment effectuer l'extraction (une étape pour l'intégration) à partir de bases de données, ce qui n'est pas encore évident pour les pages WEB.
Une fois que les données sont extraites, elles sont mises en commun. Le système utilise ces données pour répondre aux utilisateurs sans distinction de leurs origines ; les données semblant provenir d'une seule source.

3.2.2 L'opération d'extraction

Une étape importante pour intégrer les données de différentes sources est de déterminer la méthode pour y accéder et en extraire les informations. Si, à priori, la récupération des données à partir des bases de données est relativement aisée à faire (figure 3-1, partie gauche); les sources semi-structurées, quant à elles, ne possèdent pas un schéma de données. Il y a donc une nécessité pour ces sources d'avoir des outils qui permettent de construire des vues ou modèles qui nous facilitent l'extraction des données recherchées. En d'autres termes, nous devons trouver une manière pour tirer profit du minimum d'organisation que contiennent les sources semi-structurées.

Pour chaque source, nous allons construire une description décrivant comment les données qui intéressent l'utilisateur sont retrouvées et tirées de la page WEB. L'édition de la description sera faite manuellement avec l'assistance et l'aide d'un outil. Cependant, une fois construite, la description pourra être utilisée pour extraire et mettre à jour les données de la source de manière automatique. Si le contenu et les données de la page subissent des mises à jour régulières, la description est d'une grande utilité puisqu'elle permet d'automatiser l'extraction des données. Par contre, si la page change de structure, il se peut que la description correspondante soit appelée à subir quelques changements. Nous étudierons l'impact de ceci ultérieurement.

Le langage HTML n'offre aucun mécanisme de requêtes directes, cependant le consortium W3C a mis en place une spécification pour accéder aux informations contenues dans un document XML en exploitant sa structure hiérarchique.

Il est donc intéressant de pouvoir exploiter cette possibilité en traitant une page HTML comme un document XML. S'il est vrai que HTML peut être considéré comme une instance de XML, les pages WEB contiennent néanmoins certaines imperfections et incomplétudes. Les navigateurs ignorent souvent ces erreurs et permettent de visualiser les documents HTML dans la plupart des cas. Pour cette partie, nous allons considérer des outils de conversion HTML-XML et des outils de requêtes pour les documents XML. Avec ce procédé, nous allons essayer de restructurer les pages WEB pour permettre la récupération directe des données (figure 3-1, partie droite).

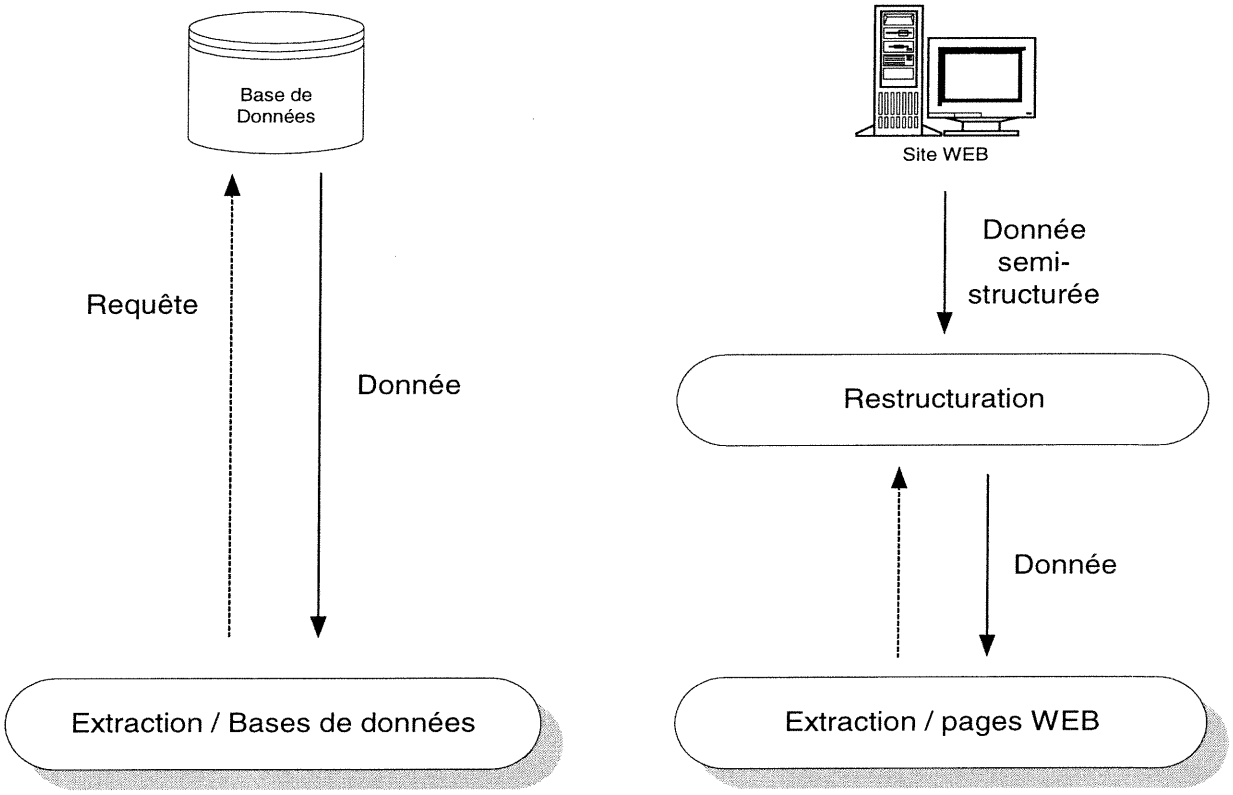


Figure 3-1 : extraction des sources

3.2.3 Architecture globale

D'après l'étude de l'état de l'art dans le domaine et les conclusions que nous en avons tirées, nous considérons que l'intégration des données issues de plusieurs sources font intervenir trois fonctions essentielles : extraction, mise en commun suivant un modèle du domaine et possibilité d'effectuer des requêtes sur les données résultats.

L'extraction des données est l'étape par laquelle les sources sont interrogées. Nous nous intéresserons particulièrement (du point de vue implantation) à récupérer les données des pages WEB. Pour cela, nous ne nous baserons pas sur des technique de délimitation de chaînes de caractères à l'intérieur des documents HTML (comme c'est le cas pour le système TSIMMIS

§2.4.3) mais sur des langages de requêtes appropriés. Notons que l'opération d'extraction nécessite une restructuration de ce type de source avant d'y appliquer des requêtes.

Il ressort aussi du chapitre précédent que la plupart des systèmes/projets utilise un modèle du domaine pour décrire les entités qui représentent une modélisation du domaine dont les données sont extraites des sources. Notre conception fait intervenir aussi une modélisation des concepts (et relations entre eux) auxquels les données des sources font référence. Le modèle du domaine, que nous appellerons : "ontologie", permet de constituer une vue commune des différentes sources et d'indiquer les données auxquelles l'utilisateur est intéressé.

La dernière partie de l'architecture est le module des requêtes qui donne la possibilité d'interroger les données extraites. Les requêtes sont construites en se basant sur l'ontologie, c'est-à-dire sur un modèle commun et indépendant des sources.
Les rôles des trois modules dans notre système sont illustrés dans la figure 3-2.

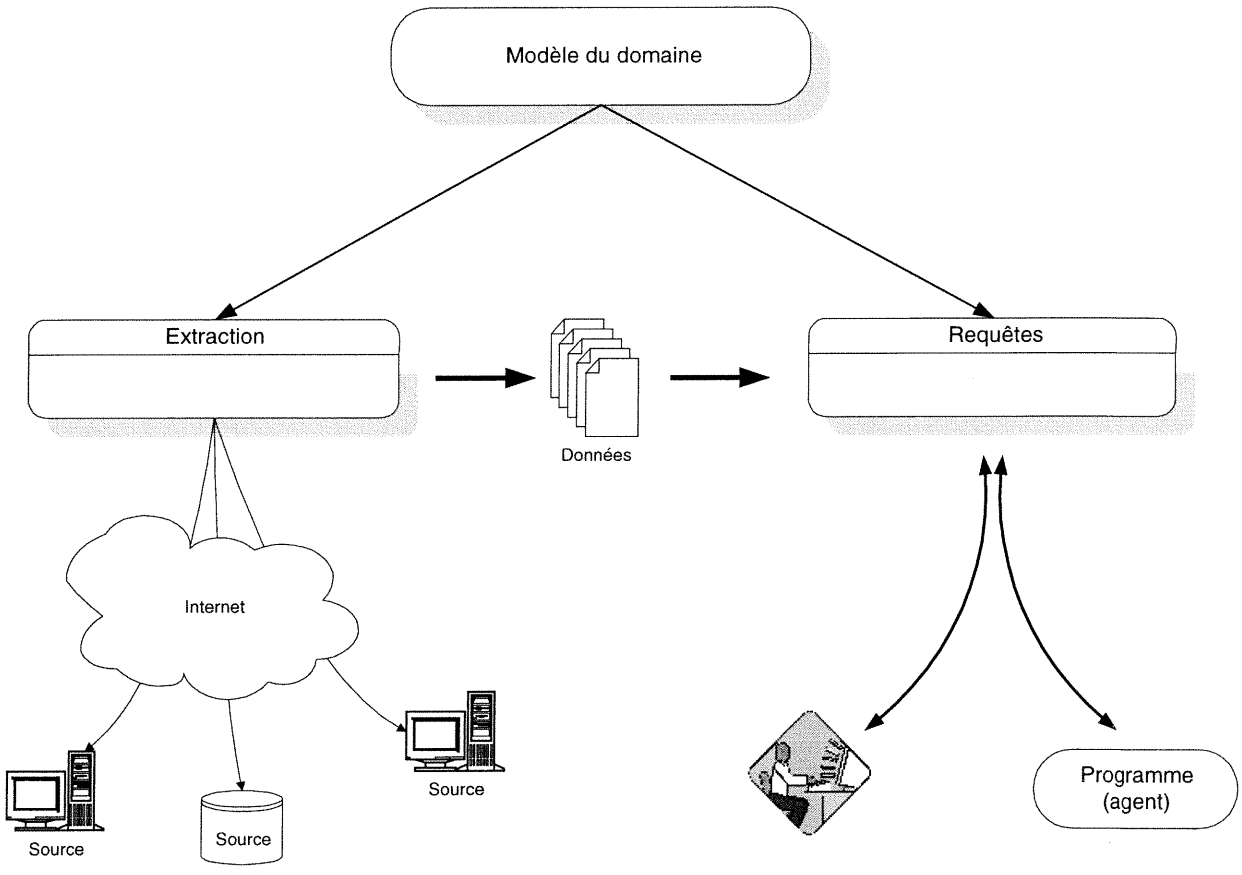


Figure 3-2 : fonctions générales

Les trois fonctions, constituées par la construction de l'ontologie, l'extraction et l'application des requêtes, sont reprises en détail dans les chapitres qui suivent.

3.2.4 L'application du concept au commerce électronique

Une application directe de l'extraction et intégration des données est dans le domaine du commerce électronique. En effet, l'opération d'extraction des données de plusieurs sources et leur intégration constituent l'étape nécessaire à tout traitement ultérieure de ces données pour le commerce électronique (comparaison, achat, vente, statistiques,...). Actuellement, cette discipline est en plein essort et prend deux formes pratiques : B2B (Business to Business), commerce ou échange entre fournisseurs et B2C (Business to Customer) qui reflète les opérations entre un fournisseur et un consommateur final. Le choix du commerce électronique comme application directe n'exclu pas d'autres possibilités.

Le développement de la première catégorie du commerce électronique se fait à l'intérieur de consortuims qui regroupent des partenaires (fournisseurs) qui partagent les mêmes intérêts pour le regroupement. Dans ce cas, l'échange des données (entre fournisseurs) se fait à travers un système central qui joue le rôle d'interface (§2.7). Chaque partenaire qui souhaite mettre à la disposition des autres ses données (par exemple sur ses produits) doit utiliser une passerelle avec le système central. Celle-ci s'occupe de faire les transformations nécessaires pour rendre les données accessibles selon le format commun utilisé. Pour chaque nouveau fournisseur qui souhaite adhérer, on doit implanter cette passerelle. Ce que nous proposons permet de faciliter le processus de récupération, la mise en commun des données et aide à l'introduction de nouveaux partenaires plus facilement.

La deuxième catégorie concerne le B2C qui se caractérise, en général, par des sites-portails offrant des produits et services divers. Le consommateur cherche le produit qui lui convient parmi les propositions du site et éventuellement effectue son achat.
Les informations sur les produits existent localement sur le site et sont fournies par les fournisseurs de ces produits en respectant un format bien défini propre au portail.
L'utilisation du processus d'extraction des données tel que nous le proposons permet d'avoir accès rapidement et directement à de nouvelles sources de données. Ce qui accroît le nombre de pages WEB référencées.

Dans les deux cas de figure du commerce électronique (B2B et B2C), l'extraction et l'intégration des données constituent les deux premières étapes d'un achat et plus précisément du modèle du

consommateur proposé par [Guttnam *et al*, 98] pour le commerce électronique. Ce modèle analyse et découpe le comportement d'un consommateur en plusieurs phases :

Le modèle, introduit par [Guttnam *et al*, 98], sert en particulier à évaluer les possibilités d'application des agents intelligents dans le commerce électronique. [Guttnam *et al*, 98] Ayant étudié plusieurs modèles de comportement du consommateur CBB (Consumer Buying Behavior), ils ont constaté que ces modèles partagent en commun six étapes. Cette décomposition permettra d'identifier les phases du comportement où la technologie peut aider le consommateur dans ses actions d'achat.

- *Need Identification* : le consommateur prend conscience d'un besoin, celui-ci peut être perçu, par exemple, à travers une information sur un produit ;
- *Product Brokering* : cette étape comprend la recherche d'information pour déterminer quoi acheter. Ceci se base sur l'évaluation des produits selon les critères du consommateur ;
- *Merchant Brokering* : il s'agit de combiner les critères précédents aux informations spécifiques de chaque vendeur pour décider de qui acheter ;
- *Negociation* : cette étape vise à déterminer les conditions de la transaction. La négociation varie en durée et complexité, il est possible pour le consommateur dans certains cas de discuter certains aspects de la transaction (prix, garantie, livraison, conditions de paiement ...) ;
- *Purchase and Delivery* : c'est l'indication que l'étape de négociation est terminée, la transaction sera donc effectuée ;
- *Service and Evaluation* : cette étape implique le produit lui même, le service après vente et la satisfaction générale du client quant à sa décision et son expérience d'achat.

Ces étapes sont une approximation du comportement du consommateur vis-à-vis d'une activité d'achat. A partir de cette perspective, on peut identifier le rôle de l'intégration des données pour faciliter la recherche des produits de plusieurs sources pour un consommateur. Ce dernier pourra faire les comparaisons nécessaires pour optimiser et faire les bons choix parmi des propositions (étapes *Product Brokering*, *Merchant Brokering*). La phase de la *Negociation* pourrait survenir ensuite.

Ainsi, nous nous plaçons dans ce domaine (commerce électronique) en particulier pour les premières phases de recherche et d'identification des produits sachant que notre proposition pourra avoir d'autres types d'application qui nécessitent l'extraction et l'intégration des données. Les exemples que nous traitons par la suite s'inscrivent dans cette optique.

Nous abordons dans les prochains chapitres les approches et méthodes que nous avons utilisées pour :

- définir un modèle commun de données sous forme d'ontologie (chapitre 4) ;
- extraire l'information et l'intégrer selon un format respectant l'ontologie (chapitre 5) ;
- effectuer des requêtes sur le résultat de l'extraction en utilisant un langage de requêtes adéquat (chapitre 6) ;
- appliquer cette approche pour récupérer et traiter des données sur les cours des actions à l'aide d'un système multi-agents.

CHAPITRE 4

Construction d'une ontologie pour l'intégration des données

4.1 Introduction

L'intégration des informations recueillies de plusieurs sources suppose l'existence d'un environnement de travail qui facilite la mise en commun des données. Les sources ne représentant pas forcément leurs données de la même manière, l'échange et la communication de ces informations n'est pas évident si les différents acteurs (soit producteurs de l'information ou consommateurs) ne s'entendent pas sur la sémantique et le sens des objets partagés. Il revient alors de définir un "alphabet" pour la bonne interprétation et compréhension des échanges. Le rôle d'une ontologie d'un domaine est de constituer un modèle commun qui fournit le minimum requis pour cette fin. En effet, une référence à un tel modèle permet de définir les éléments auxquels les utilisateurs d'un domaine sont intéressés indépendamment des sources. Celles-ci peuvent représenter leurs données de manière différente les unes des autres. Le modèle de domaine vise à uniformiser la description et l'appellation des éléments et des relations entre éléments. Le besoin d'unification constitue une des raisons pour l'utilisation d'une ontologie.

4.2 Définitions

Il n'est pas possible de trouver dans la littérature une définition unique pour l'ontologie. Chaque définition proposée prend en considération le domaine et l'utilisation envisagée de l'ontologie. Ci-après nous citons quelques définitions possibles :

- [Gruber, 93] [URL-4-10] définit ainsi l'ontologie : "Une ontologie est une spécification explicite d'un domaine particulier. Pour nous, c'est une représentation formelle et déclarative qui inclut le vocabulaire (ou des noms) pour référencer les termes dans ce domaine et les relations logiques qui décrivent ce que sont les termes et comment ils sont associés entre eux. Les Ontologies fournissent donc un vocabulaire pour représenter et communiquer les connaissances par rapport à un domaine ... " ;

- "Une ontologie est une conceptualisation d'un domaine qui introduit et décrit les entités qui peuvent y exister et les relations entre-elles, elle fournit aussi le vocabulaire nécessaire pour représenter et communiquer les connaissances." [Farquhar et al, 95] ;

- "Une conceptualisation est une vue abstraite simplifiée du monde qu'on veut représenter pour des fins particulières. Toute utilisation ou échange de connaissances est soumis (et suppose) une conceptualisation explicite ou implicite. " [Gruber, 93] ;

- "... autrement dit, c'est une définition concise et non ambiguë des entités principales d'un domaine particulier et des relations possibles entre celles-ci. Les entités peuvent être des objets, processus, fonctions, prédicats ou autres types selon le formalisme choisi pour la représentation. " [Schulze-Kremer, 97].

- "L'ontologie est à la réalité ce que la logique formelle est à la vérité, la logique formelle étudie la structure formelle du raisonnement (indépendamment de la vérité), l'ontologie formelle étudie la structure formelle du possible (indépendamment de la réalité). " [Guarino, 96].

L'ontologie est une méthode de découpage du monde en objets, et un mode de description de ces objets. C'est une description partielle du monde. Cette description est dépendante des fins voulues par le concepteur et attendues de l'application/système. De tout domaine, il est possible

de construire de nombreuses ontologies [Bezivin, 97]. L'utilisation des ontologies diffère donc d'une application à l'autre. La conception et la formalisme de représentation diffèrent eux aussi.

4.3 Besoin d'une ontologie

La recherche sur les ontologies a suscité l'intérêt de plusieurs communautés de recherche, notamment celle du partage et d'utilisation des connaissances, des agents intelligents et dernièrement du commerce électronique.
Le besoin de partager les connaissances d'un domaine particulier entre les différents systèmes/applications a fait surgir la nécessité de s'entendre sur une conception commune des entités et concepts du domaine et de leurs sémantiques. La communauté des bases de connaissances s'est alors intéressée aux ontologies dans le but de communiquer, réutiliser et comprendre les connaissances "partagées".

D'un autre coté, dans le cas des agents intelligents (communauté de l'intelligence artificielle), les besoins d'interaction, de collaboration et de coopération entre agents impliquent l'échange de connaissances. Ceci va de l'échange simple de données à la résolution partagée des problèmes. Les agents se réfèrent alors à une même ontologie pour se comprendre mutuellement.
Actuellement, on considère que l'une des barrières principales au commerce électronique se situe dans le besoin des applications de partager clairement l'information en plus des problèmes de fiabilité et de sécurité de l'Internet. Ceci est dû à la variété des systèmes pour la représentation des données utilisés par les entreprises, ce qui ne facilite pas l'utilisation des systèmes de commerce électronique. De ce fait, l'ontologie est d'une grande importance en ce qui concerne les relations avec les partenaires (chaîne d'approvisionnement), la construction de communautés virtuelles, la collaboration, la création des catalogues communs des produits, etc.

On peut affirmer que l'utilisation de l'ontologie a pour but :

- de permettre aux machines (applications) d'utiliser des connaissances du domaine ;
- le partage des connaissances entre machines (applications) ;
- de mieux comprendre le sujet traité, et partager la même compréhension avec les autres ;
- d'aider à trouver un consensus sur les connaissances avec les partenaires (fournisseurs et consommateurs) d'un domaine particulier.

4.4 Critères de conception

[Gruber, 93] a proposé des critères de modélisation d'ontologies. Il est à remarquer qu'ils ont été faits pour des ontologies servant à l'échange de connaissances ; mais ceci n'exclu pas la possibilité de les généraliser.

- *Clarté* : Une ontologie doit communiquer le sens exact des termes définis, les définitions doivent être objectives, les termes choisis doivent véhiculer le sens exact et ne pas entraîner une interprétation arbitraire ou ambiguë ;
- *Cohérence* : S'il y a lieu de procéder à des inférences, l'ensemble doit être cohérent et ne pas générer des contradictions avec les définitions initiales ;
- *Extensibilité et modularité* : Possibilité d'ajouter de nouveaux concepts sans remettre en cause les anciennes définitions et anticiper de nouvelles utilisations ;
- *Indépendance du codage* : La conceptualisation est censée être exprimée dans le niveau des connaissances sans aucune dépendance directe d'un modèle de représentation particulier ;
- *Engagement minimal* : Une ontologie doit pouvoir représenter le minimum de définitions suffisantes pour supporter le partage de connaissances entre systèmes.

Ces critères sont des directives de comparaison et d'évaluation des ontologies plus qu'un cadre théorique pour la modélisation. Celles-ci dépendent du domaine et du type d'application pour l'ontologie [Gruber, 93]. C'est au concepteur de déterminer les critères les plus importants pour lui.

4.5 Formats de représentation

Dans cette section, nous allons présenter quelques efforts et travaux concernant les ontologies et leurs conceptions parmi les recherches majeures entreprises dans le domaine.

4.5.1 Ontolingua

Knowledge Systems Laboratory de l'université de Stanford a développé des outils pour la création, l'utilisation et la maintenance des ontologies. Ces outils utilisant un langage semi-formel ; lequel supporte une description informelle des termes en langage naturel et une description formelle pouvant être interprétée par la machine.

Le langage Ontolingua utilisé pour cette fin est constitué d'une extension de KIF (Knowledge Interchange Format) [URL-4-1]. Ce dernier est un langage de logique de premier ordre avec possibilité de supporter la définition de fonctions, relations et objets et le raisonnement sur ceux-

ci. [Gruber 93] a étendu la syntaxe de KIF pour inclure des idiomes afin de pouvoir spécifier les ontologies en utilisant un style proche de l'orienté-objet : *Class, Subclass, Slot, Slot-value-type, Slot-cardinality et Facet*. Une ontologie est donc exprimée en des axiomes KIF (qui obéissent à la logique de premier ordre) et un ensemble de symboles non logiques (noms des relations, fonctions et objets qui servent de présentation pour l'utilisateur).

Un éditeur est accessible à distance en tant que service WWW, cela permet aux utilisateurs de créer, modifier et utiliser les ontologies de manière coopérative [URL-4-2]. Cet environnement de développement offre la possibilité de construire des ontologies réutilisables et partageables mais nécessite le travail à distance sur l'éditeur ; de plus l'implantation nécessite l'utilisation de LISP, ce qui rend difficile l'exploitation de cet environnement [Schulze-Kremer, 97].

4.5.2 Cyc

Cyc est une ontologie développée initialement pour supporter les connaissances communes de "chaque jour". De ce fait, Cyc n'est pas destiné à une application particulière. Cyc n'a pas d'éditeur mais dispose d'un moteur d'inférence avec utilisation d'un dictionnaire anglais. L'outil n'est pas disponible au public mais il est commercialisé. Voulant être universel et inclure le plus possible de connaissances sur "tous" les domaines, Cyc manque de précision et de spécialisation [Schulze-Kremer, 97].

4.5.3 DARPA Knowledge Sharing Effort

Ce projet est une initiative du DARPA (*Defense Advanced Research Agency*). Il vise à développer l'infrastructure technique afin de permettre l'échange des connaissances entre les systèmes [Gruber, 93]. Le but est de partager les connaissances déclaratives et les techniques de résolution des problèmes. Ces travaux originaux ont mis l'accent sur : les mécanismes pour la transformation des connaissances exprimées dans différents langages, les protocoles de communications entre les bases de connaissances et enfin les bibliothèques d'ontologies (considérées comme fondements pour les applications dans un domaine particulier).
Ce qui nous intéresse dans ce projet est la dernière partie. Le but est de trouver une ontologie commune comme moyen de partage de connaissances

4.5.4 Ontologie pour la biologie moléculaire

L'utilisation d'une ontologie commune pour la communauté de la biologie moléculaire afin de partager et s'entendre sur des connaissances communes est une expérience pratique de

construction et d'édition des ontologies. Ce projet a été initié par Max-Planck Institute for Molecular Genetics [Schulze-Kremer, 98] [URL-4-9].

L'édition d'une ontologie fait intervenir une hiérarchie d'entités de deux sortes : *CONCEPT* et *INSTANCE*. Un *CONCEPT* est équivalent au sens de classe dans l'orienté-objet, une *INSTANCE* est une instantiation d'un concept.
Il y a deux types de relations pour la construction de la hiérarchie : *IS-A* et *PART-OF*. La première relation est utilisée entre *CONCEPTS* et entre *CONCEPTS* et INSTANCES pour indiquer des (superClasses et sousClasses) généralisations et des spécialisations. La relation *PART-OF* exprime une composition physique ou conceptuelle.

4.6 Exemples d'outils

JOE, Java Ontology Editor, est un éditeur d'ontologies de l'université University of South Carolina [URL-4-3]. JOE permet d'éditer des concepts (entités), propriétés (attributs) et des relations. Une version ancienne (1997) de JOE (voir figure 4-1) est disponible sous forme d'applet Java ; cependant l'utilisateur ne peut pas procéder à l'ouverture ou à l'enregistrement de fichiers.

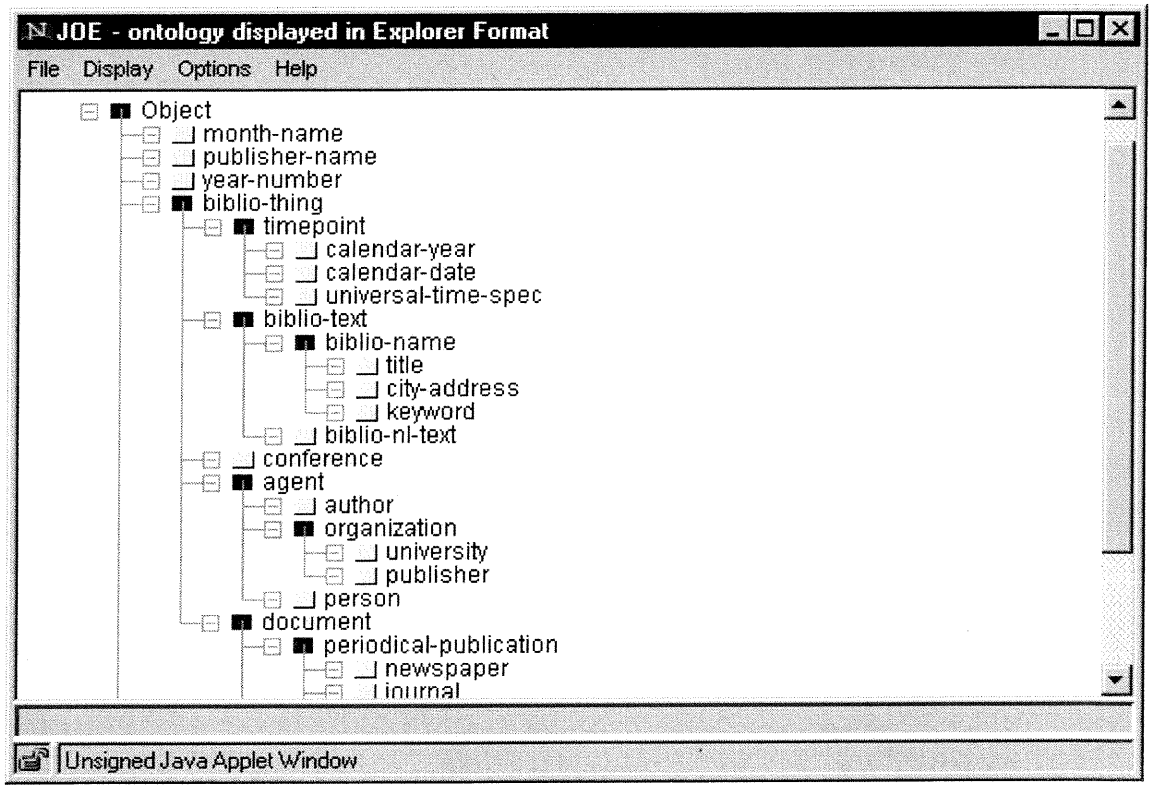


Figure 4-1 : copie d'écran de l'éditeur JOE

Ontology Editor de l'université d'Osaka au Japon [URL-4-4]. Cet outil est également sous la forme d'applet (figure 4-2). Les éléments de l'ontologie sont constitués aussi d'une hiérarchie de nœuds. Chaque nœud étend le super-nœud et hérite de ses attributs. Le nœud dispose aussi de ses propres attributs.

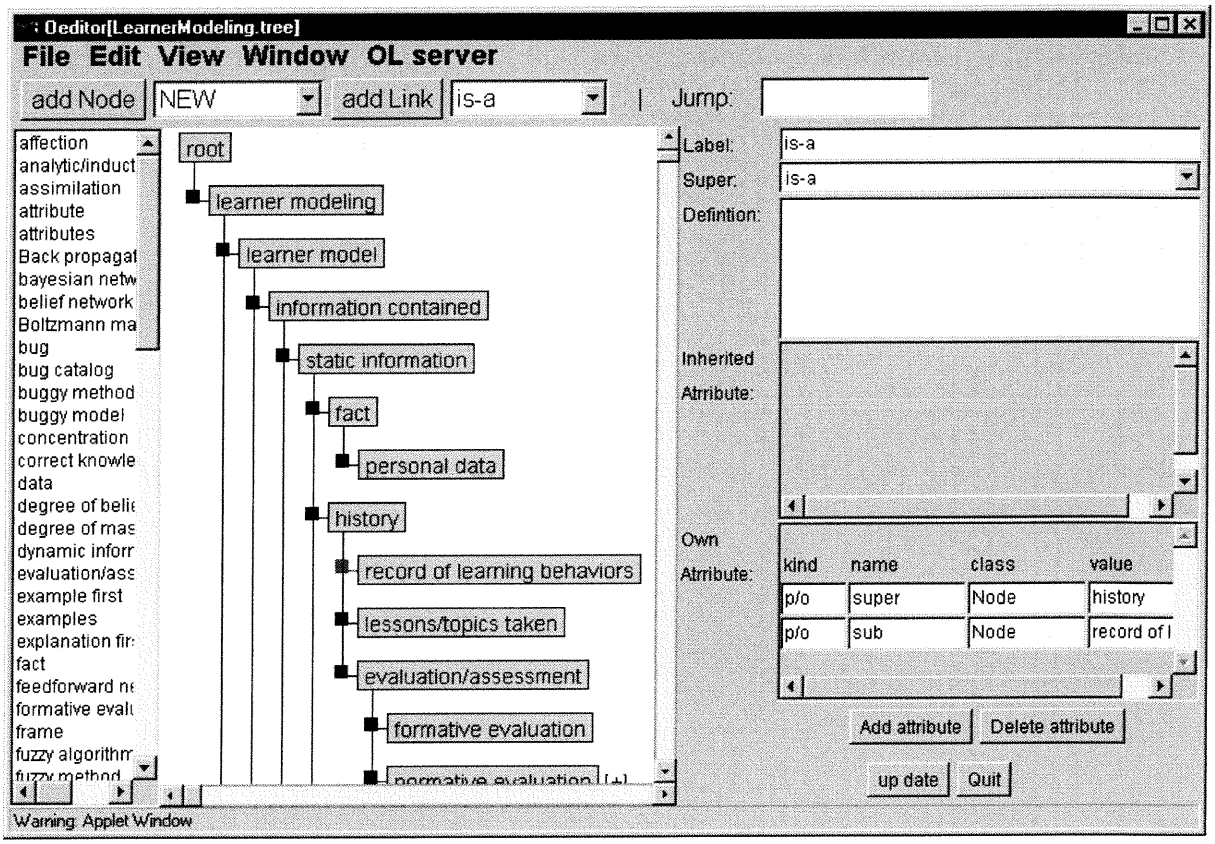


Figure 4-2 : copie d'écran de OEditor

Protégé (figure 4-3) : Une ontologie est structurée en un ensemble de classes, dont l'organisation peut être hiérarchique (une classe a au plus une Superclasse) ou sous forme de graphe général (une classe peut avoir plus d'une Superclasse). Une classe possède des propriétés propres ou hérités de ses Superclasses (directs ou indirects) simultanément le long de voies d'accès multiples. Dans le cas où une classe a plusieurs superclasses, elle hérite de tous leurs attributs et méthodes, l'utilisateur aura la tache de résoudre les problèmes liés à l'héritage multiple.

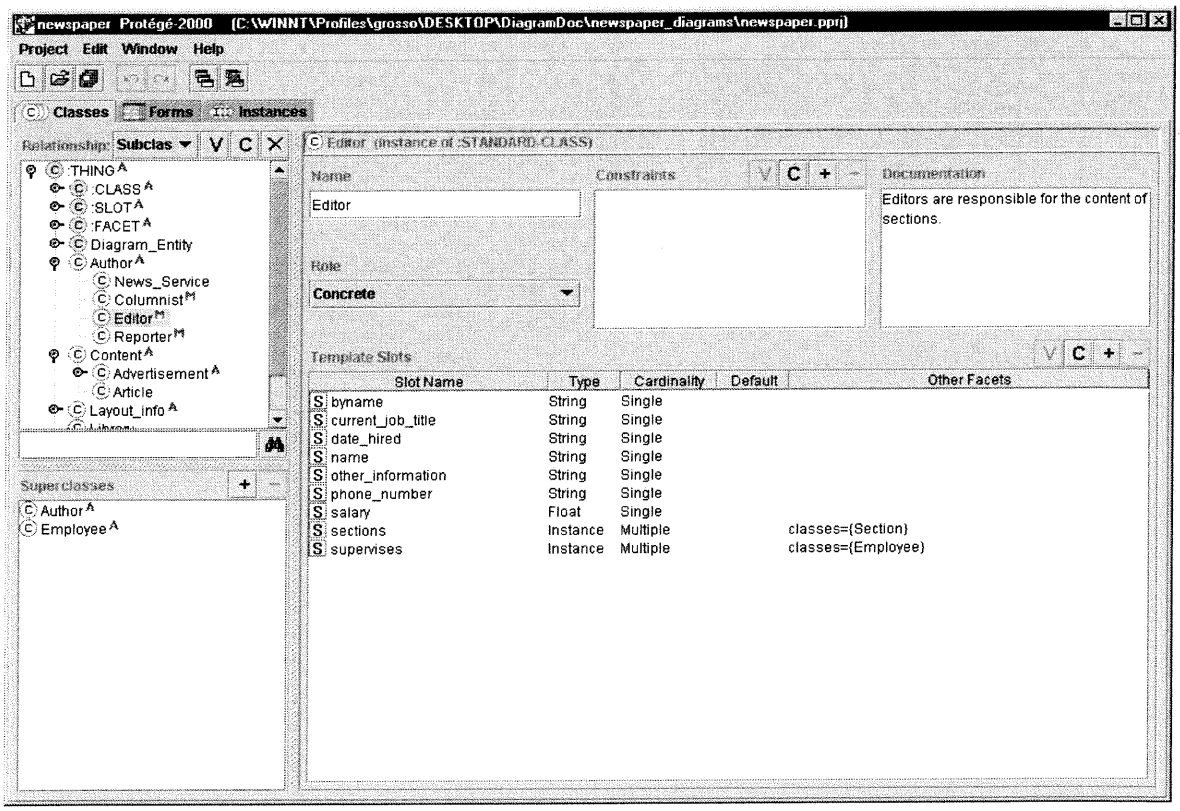


Figure 4-3 : copie d'écran de l'outil *protégé*

4.7 Proposition d'une modélisation

Il ressort des définitions, des systèmes et des outils que nous avons vus dans ce qui précède qu'il n'existe pas une définition ni une conception unique pour une ontologie. Dépendamment des besoins particuliers pour chaque cas, les attentes et la conception d'une ontologie varient.

On peut avoir une ontologie représentée sous forme d'un dictionnaire simple de vocabulaire relié à un domaine particulier. On peut avoir aussi une ontologie qui fait intervenir des relations plus élaborées (autre que des relations de synonymie, hyponymie, hyperonymie ...).

Nous proposons dans ce qui suit une modélisation d'ontologie qui sera basée plus sur la représentation des concepts et des relations entre ceux-ci dans le monde réel.

4.7.1 Modélisation

L'ontologie que nous voulons utiliser est plus pour exprimer un modèle de données que pour décrire des connaissances. De ce fait, il n'est pas nécessaire pour l'instant qu'elle utilise des mécanismes d'inférence et de raisonnement pour produire d'autres connaissances. Comme la plupart des systèmes que nous avons décrits auparavant [§2.1, §2.3, §2.5], nous allons utiliser une

modélisation d'ontologie basée sur les mécanismes proches de l'orienté-objet. Nous estimons qu'avec ce paradigme, nous arrivons à exprimer une ontologie de manière explicite et à pouvoir éventuellement générer plus facilement par la suite des unités logicielles exploitables directement par les applications.

La conception des ontologies repose sur un modèle à trois niveaux : Objets de base, méta-modélisation, modélisation. L'introduction d'une couche de méta-modélisation a été expérimentée par [Staab *et al*, 00] dans le but de pouvoir décrire des axiomes au niveau des ontologies. Il s'agit de créer des catégories d'axiomes qui introduisent des descriptions sémantiques plutôt que syntaxiques qui seraient dépendantes des machines. Les auteurs étendent les définitions fournies du RDF (Ressource Description Framwork) [URL-4-5] pour les utiliser dans la catégorisation des axiomes. Les catégories permettent de distinguer entre les structures qui se répètent dans les axiomes. Une catégorie est ensuite instanciée suivant la description commune des axiomes.
Nous allons utiliser ici le même concept, mais dans un cadre plus général que les axiomes, en permettant une redéfinition des rôles des éléments utilisés pour la modélisation par le concepteur suivant les besoins particuliers de l'application.

4.7.1.1 Objets de base

La plupart des travaux considèrent des entités, attributs et relations pour la modélisation de l'ontologie. Nous allons considérer également ces objets en plus d'un quatrième élément : des contraintes. L'introduction des contraintes permet d'apporter plus de restriction et de contrôle sur les ontologies résultantes. L'outil « Protégé » décrit précédemment, prévoit d'appliquer des contraintes aux éléments et attributs, cependant cette possibilité n'est pas encore implantée.
Ces objets de base constituent les éléments essentiels sur lesquels repose la méta-modélisation et la conception de l'ontologie elle-même.

- **Entité** : une entité décrit des concepts (éléments du domaine étudié) et fournit une représentation logique de ceux-ci.
- **Attribut** : c'est une propriété qui caractérise une entité.
- **Relation** : elle exprime les liens qui peuvent exister entre les objets de la modélisation : les entités et les attributs.
- **Contrainte** : c'est une restriction que le concepteur applique aux entités, attributs et relations afin de décrire ses contraintes.

4.7.1.2 Méta-modélisation

Ce niveau permet de définir les interactions qui régissent les liens entre les objets définis précédemment. Le but étant de fournir la possibilité au concepteur de définir ses propres outils pour construire l'ontologie. La méta-modélisation permet donc d'apporter un raffinement aux objets de base et de définir certaines propriétés sur ceux-ci.

Nous allons donner deux exemples de méta-modélisation et de conception d'ontologies pour illustrer ce niveau.

4.7.2 Premier exemple de méta-modélisation

En considérant les objets de base définis auparavant soit (Entités, Attributs, Relations et Contraintes), le méta-modèle (voir figure 4-4) de cet exemple apporte un raffinement à ces objets et définit les interactions entre eux.

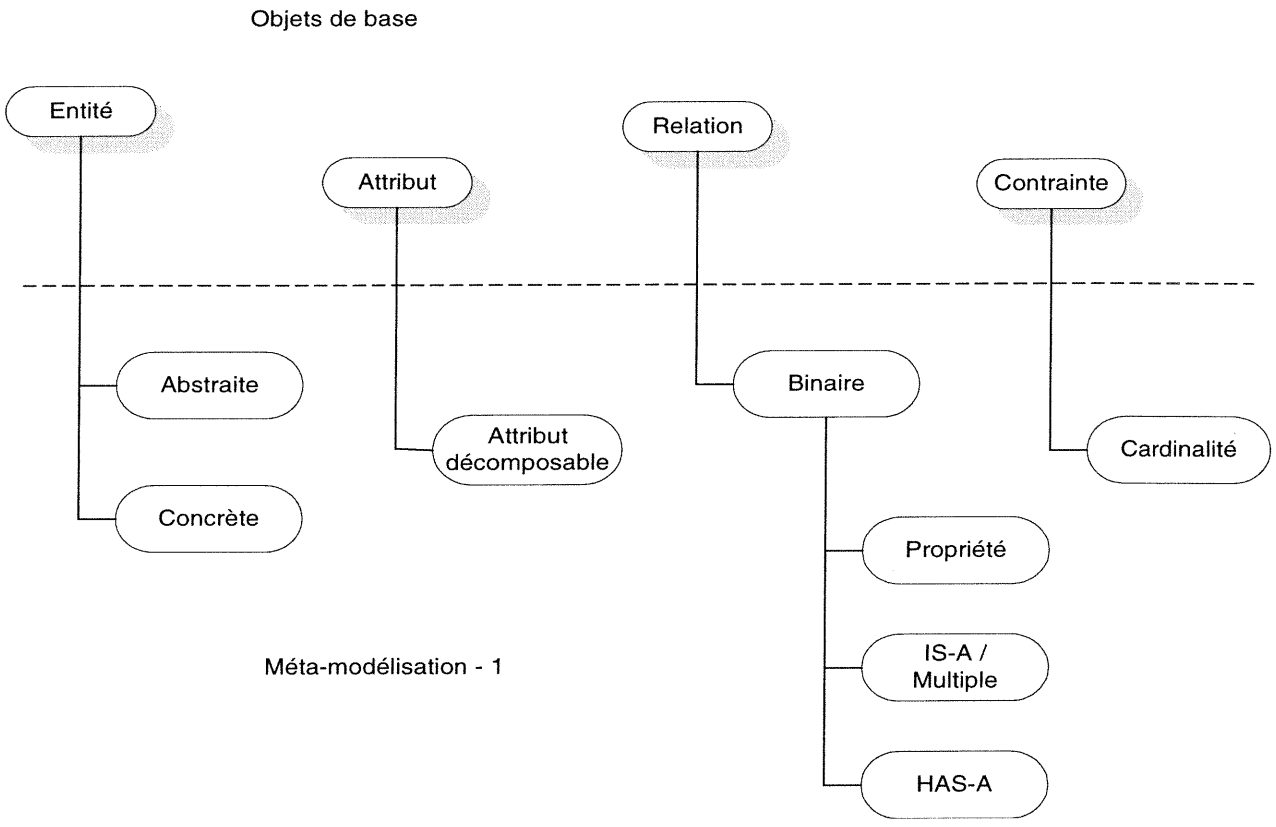


Figure 4-4 : premier exemple de méta-modélisation

On spécifie certaines propriétés et contraintes sur les objets de base (Entité, Attribut, relation et Contraite) pour exprimer des besoins particuliers de modélisation :

- Une entité est abstraite (*Abstract*) ou concrète (*Concrete*). La première est introduite pour supporter la définition des éléments génériques du domaine qui admettent plusieurs formes "instantiation" dans la réalité. Une entité concrète est utilisée pour représenter un objet/élément du domaine considéré comme réel ;
- Un attribut est décomposable. Il peut s'agir d'un objet unitaire (attribut age est unitaire il prend des valeurs d'entiers) ou complexe (il est composé de plusieurs autres attributs).
- Les relations prises en compte dans cet exemple sont de type binaire Entité-Entité ou Entité-Attribut. Cette dernière indique que l'attribut est une propriété (Property) de l'entité en question. La relation Entité-Entité peut être du type IS-A pour exprimer un sous concept représenté par la deuxième entité et qui hérite de la première ; une entité peut hériter de plusieurs autres. Le deuxième type est une relation de composition HAS-A qui indique qu'une entité fait partie d'une décomposition physique ou fonctionnelle de l'autre.
- Les contraintes s'appliquent aux relations et aux entités elles mêmes : dans une relation IS-A, l'entité hérite de tous les attributs de l'entité mère. Les contraintes de cardinalités concernent la relation HAS-A. On spécifie le nombre d'occurrence minimal et maximal de l'entité qui compose la deuxième partie de la relation.

Nous avons utilisé cet exemple de méta-modélisation pour modéliser deux cas d'ontologies, la première concerne les moyens de transport (voir figure 4-5) et la deuxième illustre des données financières (figure 4-6).

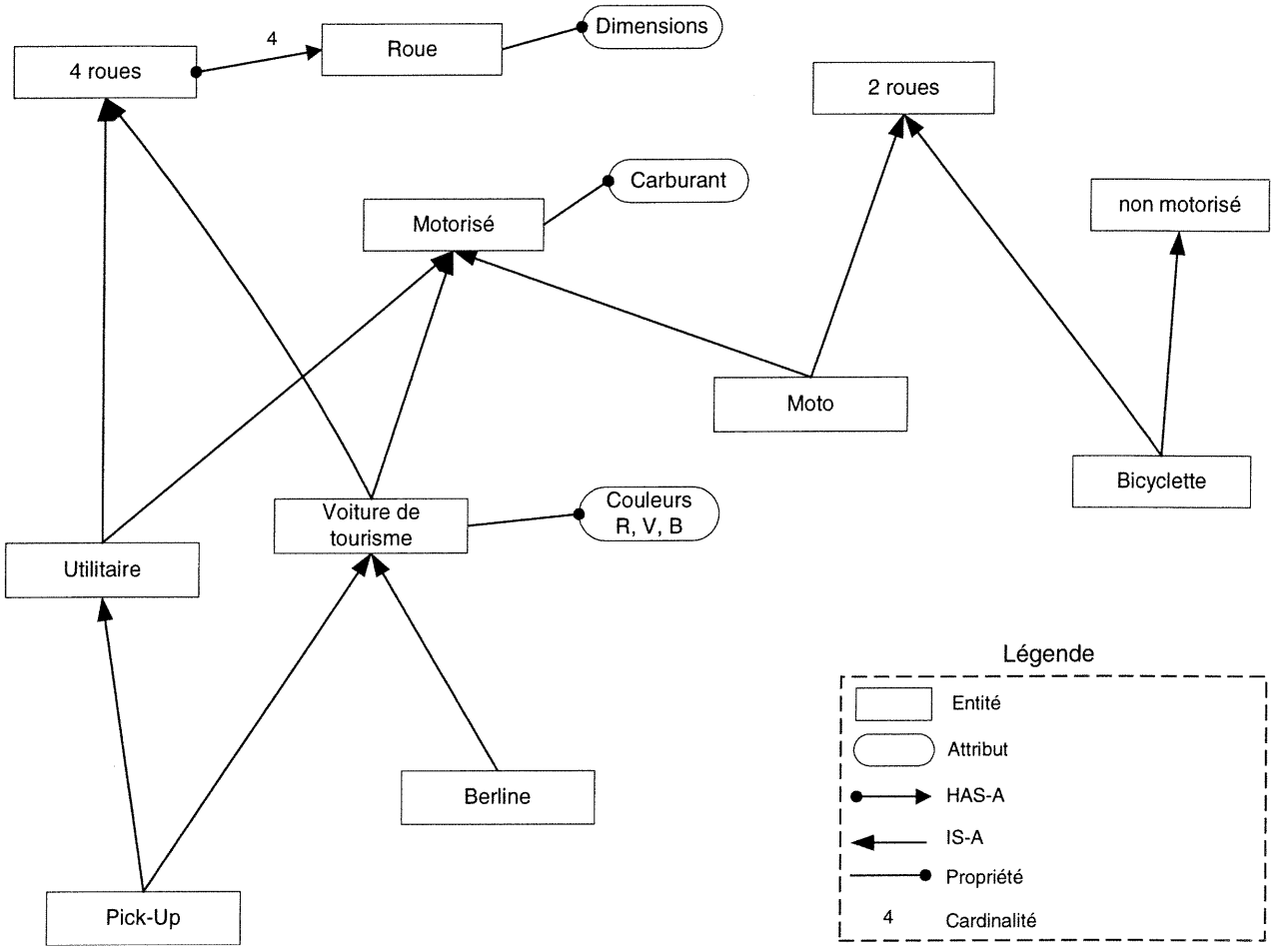


Figure 4-5 : exemple d'ontologie utilisant la première méta-modélisation

Dans la figure précédente, l'entité *Voiture de tourisme* est un véhicule *Motorisé* à *4 Roues*. Ceci est exprimé par deux relations IS-A avec deux autres entités *4Roues* et *Motorisé*. L'entité *Voiture de tourisme* possède une couleur qui la caractérise, ceci est exprimé par la relation Propriété avec l'attribut *Couleur (R,V,B)*.

Ci-après, est un autre exemple de modélisation. Cette fois-ci, il concerne le domaine financier, particulièrement les données sur les cours des actions et des taux de change. La figure 4-6 montre l'utilisation du premier exemple de la métamodélisation. L'entité STOCK est en relation Propriété avec les entités STOCK_DESCRIPTION, DATE, STOCK_VALUE, STOCK_ID et STOCKMARKET qui la composent. L'entité STOCK est en relation IS-A avec FINANCE pour exprimer que c'est un concept de la finance.

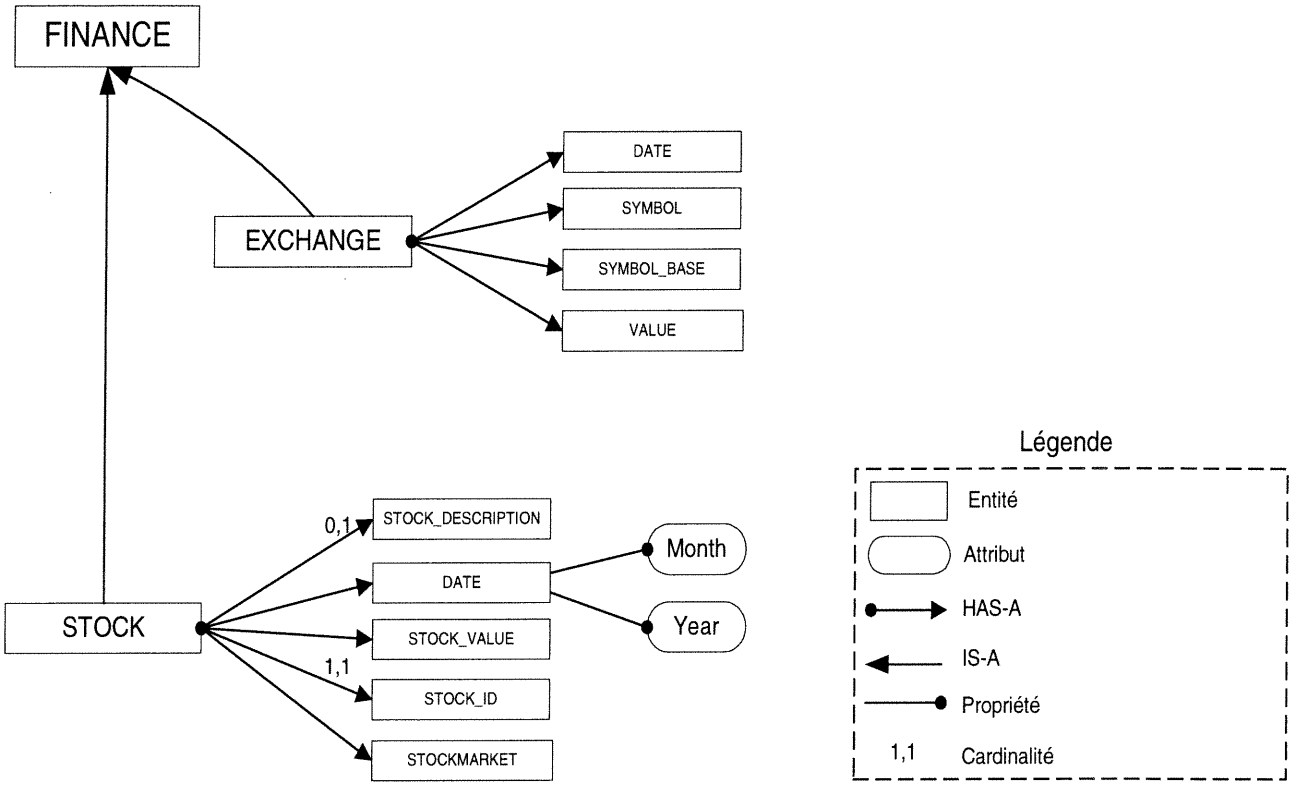


Figure 4-6 : autre exemple utilisant la première méta-modélisation

4.7.3 Deuxième exemple de méta-modélisation

Ce deuxième exemple de méta-modélisation est celle que nous allons prendre en compte par la suite lors de la conception de l'ontologie (figure 4-7).

- Une entité est une représentation d'un concept ou objet du monde réel.
- Un attribut est unitaire, il n'est pas décomposable.
- Nous considérons dans le niveau de méta-modélisation les relations entre les entités et entre les entités et les attributs. Une relation est binaire (entité-entité ou entité-attribut) ou multiple (entité-entités ou entité-attributs). La relation binaire entre une entité et un attribut prend le nom de *Property* et signifie que l'attribut est une propriété de l'entité. Une relation binaire entre une entité et une autre entité est soit une relation de spécification *IS-A* ou une relation de composition *HAS-A*. Une relation multiple *Properties* est une relation qui lie une entité à un ensemble d'attributs. Une relation multiple entre une entité et un ensemble d'entités en même temps est une relation de composition *HAS-A*.
- Les contraintes que nous avons choisies s'appliquent à des relations pour exprimer des besoins élémentaires en terme de nombre d'occurrence (*Cardinality*), une séquence précise de

décomposition (*Sequence*) ou pour faire un choix parmi un ensemble donné (*Choice*). Les contraintes *Sequence* et *Choice* s'appliquent seulement dans le cas des relations multiples afin d'exiger l'existence d'une séquence déterminée ou un choix parmi un ensemble d'entités ou d'attributs. La contrainte *Cardinality* s'applique aux relations binaires et multiples afin de définir le nombre d'occurrences possibles pour les entités ou attributs qui composent cette relation.

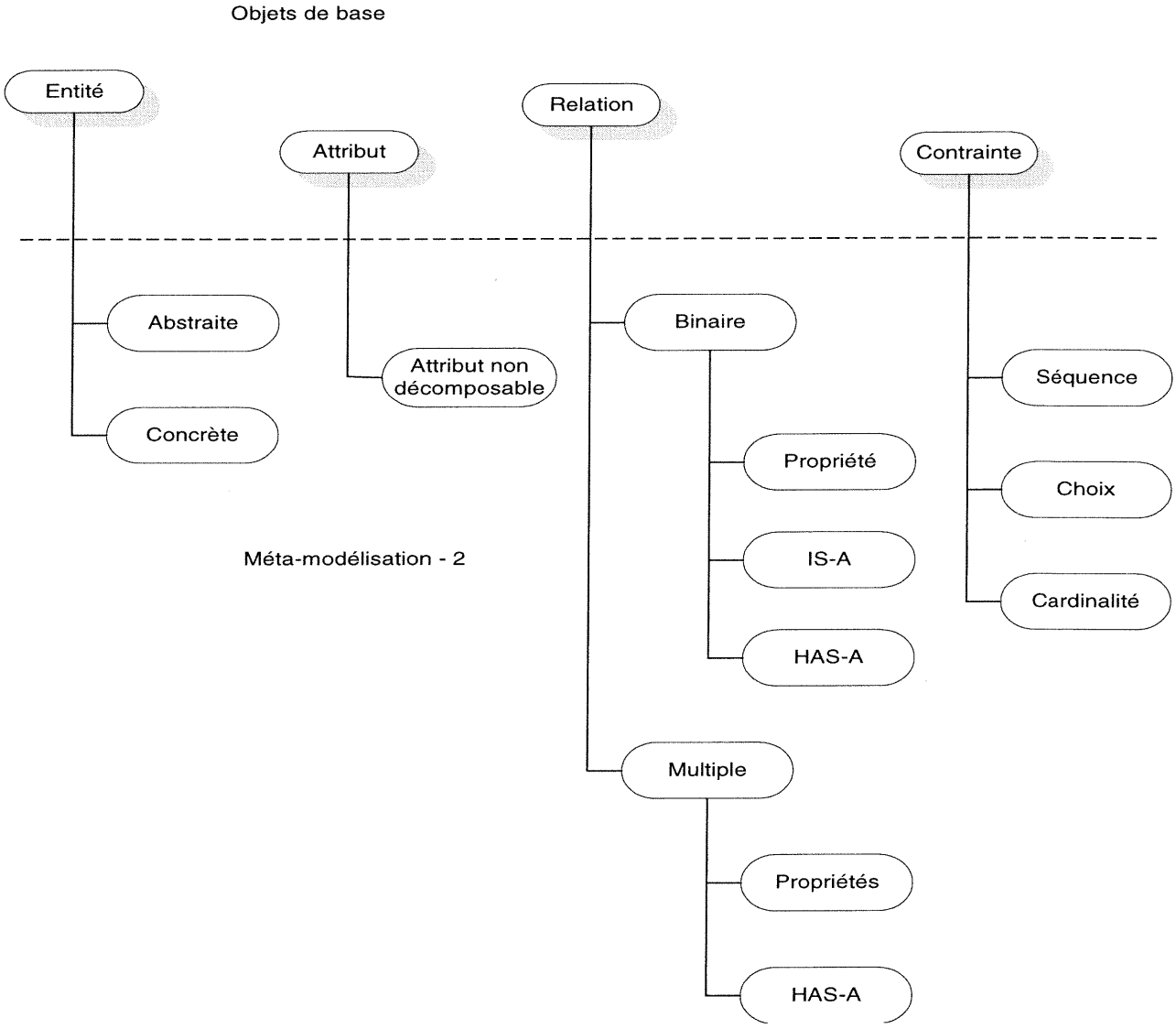


Figure 4-7 : deuxième exemple de méta-modélisation

Dans l'exemple qui suit (voir figure 4-8), nous reprenons l'exemple du domaine financier en utilisant le deuxième exemple de méta-modélisation. Les entités de la figure 4-8 sont pratiquement identiques que ceux de la figure 4-6 avec des différences au niveau des relations qui

sont spécifiées dans les deux exemples de méta-modélisation. Par exemple, l'entité STOCK est en relation HAS-A avec les entités STOCK_DESCRIPTION, DATE, STOCK_VALUE, STOCK_ID et STOCKMARKET. Puisque la présence de ces dernières entités est nécessaire pour décrire et représenter l'entité STOCK, la relation HAS-A qui lie l'entité STOCK et l'ensemble (STOCK_DESCRIPTION, DATE, STOCK_VALUE, STOCK_ID et STOCKMARKET) prend la forme de séquence de composition. En pratique, ceci signifie l'obligation d'avoir ces dernières entités toutes ensemble. La présence d'une entité de l'ensemble est dépendante de la présence des autres entités qui le composent. La création de ce genre de dépendance permet de contrôler la complétude des données extraites et la validation des documents.

Par ailleurs, l'entité STOCKMARKET est en relation de choix avec deux attributs NYSE et EURONEXT, elle prend l'un ou l'autre attribut.

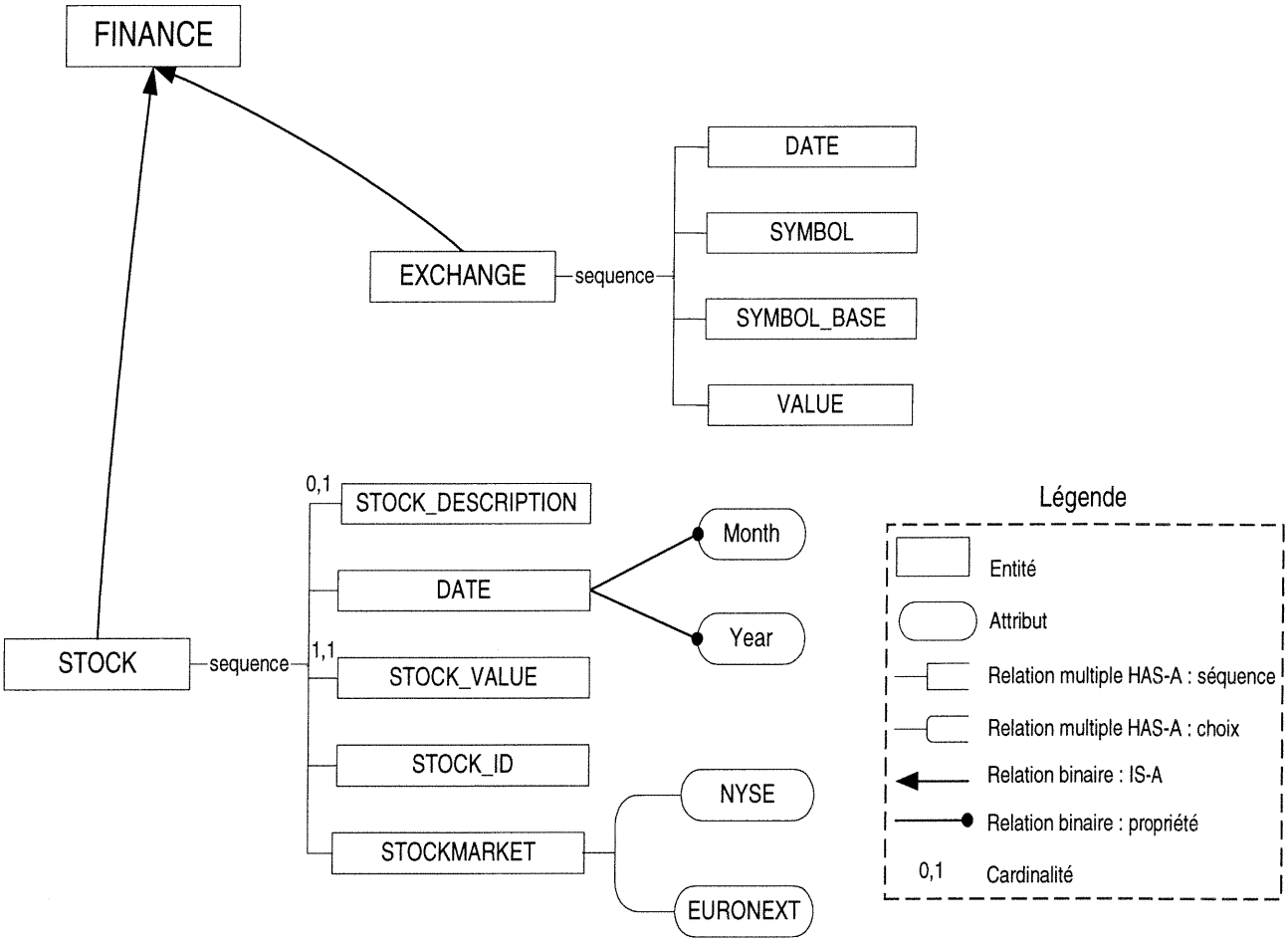


Figure 4-8 : exemple utilisant la deuxième méta-modélisation

Les deux figures 4-6 et 4-8 représentent respectivement deux exemples simplifiés d'ontologie pour les données sur les cours des actions et des taux de change. Nous pouvons remarquer que certaines contraintes telles que des relations multiples entre une entité et un ensemble d'entités ou d'attributs ne peuvent pas être représentées en utilisant le premier exemple de modélisation. Plusieurs relations *HAS-A* (dans la première méta-modélisation) ne peuvent pas exprimer une relation avec l'ensemble des entités puisque chaque relation *HAS-A* se comporte de manière indépendante des autres. Tandis qu'une relation multiple *HAS-A* (dans la deuxième méta-modélisation) permet non seulement d'avoir plusieurs liaisons *HAS-A*, mais implique également une corrélation implicite entre les entités. Par exemple, les entités (DATE, SYMBOL, SYMBOL_BASE et VALUE) doivent exister en même temps. De même une relation impliquant un choix ne peut pas être représentée en utilisant la première méta-modélisation. Pour ce qui est de l'exemple du transport, il peut être modélisé indifféremment avec la première ou la deuxième méta-modélisation.

En conséquence, l'utilisateur choisira la représentation qui lui permet d'exprimer ses contraintes et ses besoins de modélisation en fonction de son domaine.

4.8. Exemple d'une ontologie

L'ontologie que nous avons construite est celle décrite par la figure 4-8 en utilisant la deuxième méta-modélisation 4.7.3. La figure illustre une partie du fichier de représentation de cette ontologie. Dans ce qui reste de ce mémoire, nous nous baserons sur cette dernière pour nos opérations d'extraction, d'intégration et de requêtes.

Nous avons choisi d'expérimenter le concept en prenant un exemple relié au commerce électronique. Dans ce domaine, la phase de la recherche et la consultation des produits est une étape importante dans le processus de l'achat. Les résultats de cette étape sont la plupart du temps décisives pour l'utilisateur. Il est donc intéressant de proposer aux clients la vue la plus complète possible sur les biens ou services recherchés. Le concept de l'intégration dans ce cas est très utile puisqu'il permet à l'acheteur de revoir les propositions de plusieurs vendeurs en même temps et de manière uniforme (exemple : tous les prix en dollars canadiens). Par ailleurs, il existe de nombreux efforts de standardisation et une multitude de recherches reliés au domaine. Il serait judicieux de tenir compte des résultats éventuels de ces travaux dans un souci de test du concept de l'intégration des données.

L'existence des sources de données, (sites WEB et bases de données) qui offrent des informations concernant les produits et services, est un élément essentiel qui a motivé le choix de ce domaine.

4.8.1 Outils pratiques pour l'implantation

4.8.1.1 Langage pour la définition des schémas.

De manière pratique, nous allons faire appel à SOX (Schema for Oriented-Object XML), un langage de définition de schémas pour les documents XML [URL-4-6]. SOX est développé par Commerce One pour l'utilisation de XML en commerce électronique [URL-4-7]. On désigne par schéma un ensemble de règles qui définissent la structure d'un document. Un DTD (Document Type Definition) est un cas particulier de langage de schémas utilisé pour définir des documents XML. Etant donné un schéma, on peut créer des instances de documents XML qui s'y conforment, la vérification est faite par le *parser*. Une ontologie est décrite en terme de ce langage.

Le choix de SOX est motivé par le fait qu'on peut l'utiliser pour exprimer les besoins de méta-modélisation définis précédemment (exemple-2). SOX a été développé pour palier aux insuffisances des DTD, il est proche du paradigme objet et introduit les concepts de ce type de programmation aux documents XML. Il offre ainsi de nombreux avantages [URL-4-6] :

- SOX permet de définir des types de données. En plus des types prédéfinis, l'utilisateur peut spécifier ses propres types de données. Par exemple, on peut spécifier que la valeur d'un attribut doit appartenir à un intervalle donné, ceci n'étant pas possible en utilisant une DTD. Le schéma défini ainsi, permet de garantir un minimum de contrôle et de fiabilité pour les applications.
- SOX met en valeur XML en offrant la possibilité d'étendre des éléments déjà prédéfinis par héritage.
- SOX encourage la définition des types d'éléments et de données en utilisant les *Namespaces*. Ceci signifie qu'un schéma peut faire appel à des définitions d'un autre schéma par des importations.

Les applications du commerce électronique ont besoin d'utiliser des données fortement type, par exemple, les prix, les quantités et les dates. Les DTD ne supportent que le String comme type de données. SOX se propose de remplacer les DTD qui ne sont pas adéquats pour les applications du commerce électronique. Actuellement, XML tend à devenir un des outils essentiels pour le commerce électronique. De ce fait, l'utilisation des schémas deviendra elle aussi importante. L'écriture des schémas introduit moins d'erreurs essentiellement si on se base sur des définitions précédentes au lieu de redéfinir le tout de nouveau. Ceci convient mieux du point de vue

programmation dans un environnement distribué où les documents sont partagés par de nombreuses applications. Ci-après, la figure 4-9 montre un exemple d'ontologie sous forme de schéma SOX.

```
<?xml version ="1.0"?>
<!DOCTYPE schema SYSTEM "urn:x-
commerceone:document:com:commerceone:xdk:xml:schema.dtd$1.0">

    <schema uri = "file:///S:/DataFiles/schemas/n1_0/Finance.sox" soxlang-version =
                                        "V0.2.2">
      <elementtype name = "STOCK">
            <model>
                  <sequence>
                        <element type = "STOCKMARKET"/>
                        <element type = "STOCK_ID"/>
                        <element type = "STOCK_VALUE"/>
                        <element type = "DATE"/>
                  </sequence>
            </model>
      </elementtype>

      <elementtype name = "STOCKMARKET">
            <model>
                  <string/>
            </model>
      </elementtype>

      <elementtype name = "STOCK_ID">
            <model>
                  <string/>
            </model>
      </elementtype>

      <elementtype name = "STOCK_VALUE">
            <model>
                  <string datatype = "int"/>
            </model>
      </elementtype>

      <elementtype name = "DATE">
            <model>
                  <string datatype = "date"/>
            </model>
      </elementtype>
...
</schema>
```

Figure 4-9 : exemple de modélisation sous forme de schéma SOX

Afin de faciliter l'indépendance entre applications et documents, SOX supporte les possibilités d'extension et d'attribution de versions pour les documents. Ainsi, les applications supportent aisément les changements aux schémas et aux documents sans revoir les applications.

4.8.1.2 xCBL, XML Common Business Library: Bibliothèque de documents

Tandis que chaque nouvelle spécification XML d'un domaine particulier est une contribution à l'évolution des modèles de données pour l'échange entre partenaires, elle tend aussi à vouloir réinventer les concepts et les descriptions qui entrent en jeu. Une meilleure approche serait de construire des documents XML à partir de composants communs à un domaine et l'adapter par la suite aux cas particuliers [URL-4-6].

xCBL est une spécification en XML de CommerceOne pour l'échange des documents d'affaires au sein de l'industrie. Elle inclut des descriptions de produit, ordres d'achat, factures, et les échéanciers d'expédition. xCBL est un ensemble de modules XML qui permet la création de documents XML réutilisables pour le commerce électronique.
Le but de xCBL est de fournir un ensemble initial d'éléments que les compagnies peuvent étendre et utiliser afin de développer rapidement leurs applications. Certains des éléments de base sont pris des standards internationaux (ISO 8601 pour la date et l'heure, ISO 639 pour les codes des langues, ISO 3166 pour ce qui concerne les codes de pays, ISO 4217 pour les codes des monnaies). Cependant, les autres formats de données particulières ne sont pas standardisés, c'est le rôle de l'ontologie d'uniformiser ces représentations et de compléter ces standards.

xCBL est, par ailleurs, supporté par des leaders de l'industrie : BizTalk de Microsoft, OASIS, le groupe de travail de méthodologies d'UN/CEFACT et le groupe de projet eCo de CommerceNet.

Dans la mesure du possible, nous allons essayer d'utiliser certains éléments de xCBL afin de rester le plus proche possible des efforts de standardisation de la part des acteurs et consortiums industriels.

4.8.3 Description de l'outil XML Autority

Nous utilisons un éditeur de schémas pour les documents XML : XML Autority. C'est un produit de Extensibility [URL-4-8]. Il prend en compte la définition des schémas en utilisant des DTD, SOX et XDR avec possibilité de validation des documents. L'outil permet de visualiser l'arborescence du document de manière graphique ce qui facilite la conception.
La figure 4-10 montre une prise d'écran de l'outil avec un exemple de construction de schéma.

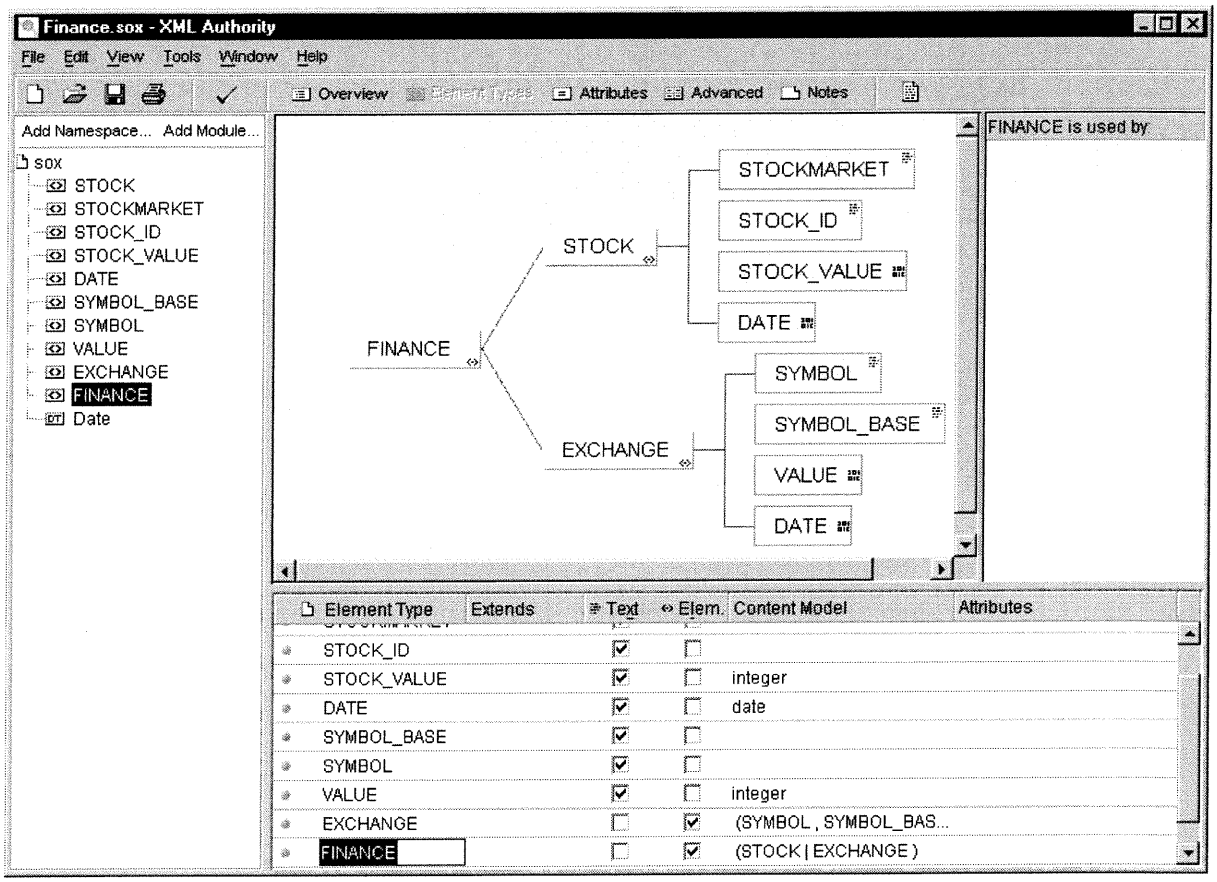


Figure 4-10 : prise d'écran de l'outil XML Autority

4.9 Evaluation par rapport aux critères de conception

- Clarté : en ce concerne la clarté, il suffit de choisir les termes exacts pour désigner les éléments et les attributs. A ce stade, les éléments de l'exemple du §4.8.1.1 véhiculent le sens voulu.
- Cohérence : notre ontologie ne fait pas appel à des inférences, le critère de la cohérence n'a pas à être vérifié.
- Extensibilité : la relation IS-A permet de définir des entités représentants des concepts généraux qui sont à spécifier et à étendre au fur et à mesure que des besoins particuliers surgissent. Cette relation d'héritage offre les possibilités d'extension à l'ontologie.
- Modularité : les éléments et les attributs sont à définir sous forme de parties indépendantes et réutilisables, nous pensons aussi à utiliser et à importer des définitions (modules) déjà développés par CommerceOne : la librairie xCBL.

- Indépendance du codage : l'outil que nous avons utilisé pour éditer l'exemple permet une représentation en différents formats de codage (en SOX et XDR, ce sont des outils qui prennent en compte les besoins et les caractéristiques de notre choix de la méta-modélisation).
- Engagement minimal : l'exemple de l'ontologie reste assez général afin de prendre en compte les spécificités de chaque source pour les cours des actions. Par exemple, l'ontologie accepte de référencer le titre d'une action aussi bien par numéro (Bourse de Paris) que par chaîne de caractères (pour le cas du NYSE, Nasdaq, ...).

CHAPITRE 5

Conception et implantation de l'extraction

Les pages Web qui présentent des données dynamiques gardent en général une même structure du document et d'emplacement des données à l'intérieur de la page HTML. Ceci est dû principalement au fait qu'il existe une base de données (ou autre représentation structurée) en arrière plan du site, et qui l'alimente en informations. Nous allons exploiter cette "propriété" pour récupérer les données.

Dans le domaine de l'extraction de données des pages WEB, les travaux réalisés utilisent des langages de requêtes pour interroger les documents HTML. Une façon de faire serait alors d'exécuter des requêtes directes sur ces documents [A. Sahuguet and F. Azavant, 99a][Huck et al, 98]. Dans ce cas, les langages et les grammaires utilisés sont spécifiques au HTML et prennent aussi en considération les spécificités des pages WEB (erreurs et irrégularités). Nous présenterons dans ce qui suit deux exemples de travaux qui s'inscrivent dans cette perspective. Nous montrerons les raisons qui nous ont poussé à ne pas choisir cette voie et à opter pour une autre approche. Il s'agit de faire les transformations nécessaires sur le document HTML pour être considéré comme un document XML, ce qui nous permet d'appliquer le langage de requêtes de notre choix.

5.1 Travaux et réalisations pour l'extraction des données des pages HTML

5.1.1 W4F : World Wide Web Wrapper Factory

W4F est un environnement de développement qui permet aux utilisateurs de construire un *Wrapper*, de le compiler comme un composant java et de l'inclure dans leurs applications [A. Sahuguet and F. Azavant, 99b].

Un *Wrapper* est une interface pour accéder au contenu des pages WEB. Il charge le document du réseau, le corrige, en extrait les données et lie celles-ci à une structure de données prédéfinie, qui sera utilisée par la suite. Un *Wrapper* contient une spécification, divisée en trois sections, pour renseigner sur la manière de : récupérer le document, d'extraire les données et de les lier avec des variables.

Le téléchargement du document WEB est pris en compte par le "Retrieval Agent" en utilisant le protocole HTTP. Dans ce cas, il joue le rôle d'un *browser* normal. La deuxième étape est celle de la correction du document HTML afin de le transformer en un arbre DOM représentatif du document.

La troisième phase consiste à extraire effectivement des données à partir de l'arbre construit. Pour ce faire, [A. Sahuguet and F. Azavant, 00] ont définit un langage d'extraction spécifique HEL (HTML Extraction Language) pour décrire les règles d'extraction. Une règle d'extraction contient une expression de chemin le long de l'arbre et une spécification des informations à récupérer. Les données extraites sont par la suite stockées temporairement dans une structure de données interne au système NSL (Nested String List). La dernière phase de l'extraction lie les données obtenues avec les structures de données propres à l'application. La transformation de la structure interne vers la structure définie par l'utilisateur se fait de manière déclarative (voir la section SCHEMA de l'exemple de *Wrapper* qui suit).

L'architecture de W4F se présente comme suit (voir figure 5-1)

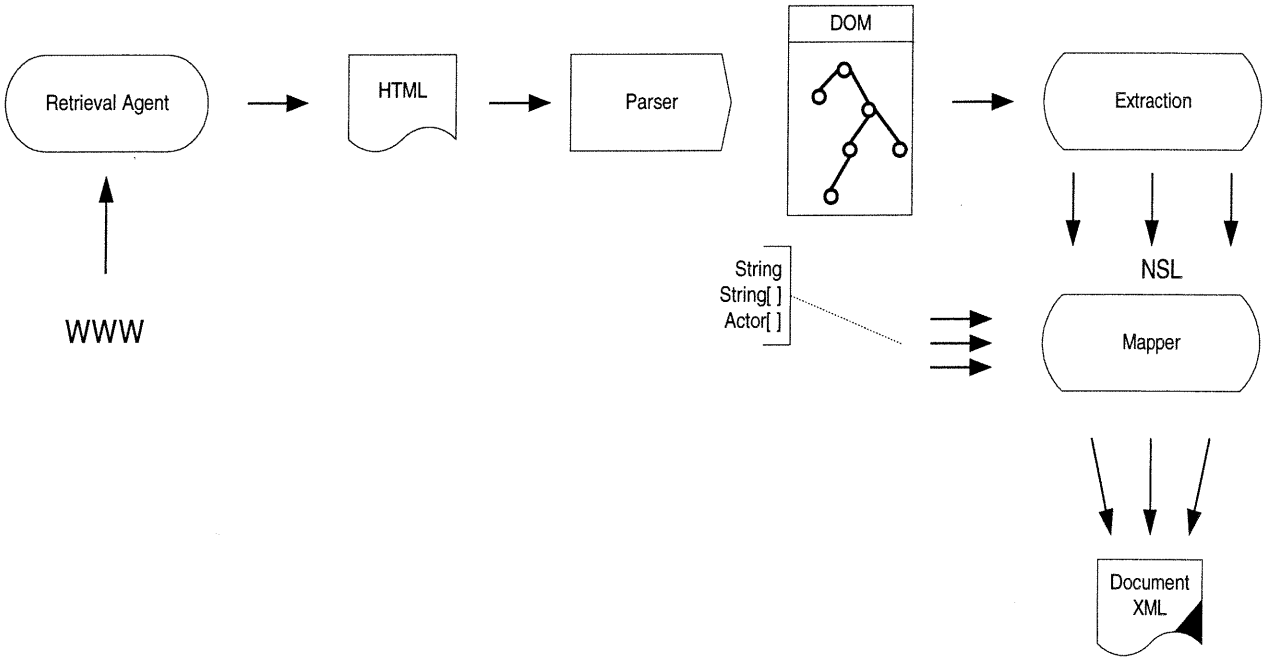


Figure 5-1 : architecture de W4F

La spécification d'un *Wrapper* se fait à l'aide de trois parties déclaratives : Retrieval_Rules, Exctraction_Rules et SCHEMA. L'exemple suivant montre un *Wrapper* utilisé pour l'extraction de données relatives aux films disponibles sur le site de IMDb (Internet Movie Database) [URL-5-1]. Les données récupérées concernent le nom du film, son genre et les acteurs qui y jouent.

Un exemple de spécification est comme suit :

```
SCHEMA
{
        String title;
        int year;
        String[] genres;
        String[][] cast;
}
EXTRACTION_RULES
{
        title = html.body->h1.txt, match/(.*?) [(]/;
        year = html.body->h1.txt, match/.*?[(]([0-9]+)[)]/;
        genres = html.body->td[i:0].a[*].txt
        WHERE html.body->td[i].b[0].txt = "Genre";
                cast = html.body->table[i:0].tr[j:*].td[0].txt, match/(\S+)\s(.*)/
        WHERE html.body->table[i].tr[0].td[0].txt =~ "Cast"
                AND html.body->table[i].tr[j].getNumberOf(td) = 3;
}
RETRIEVAL_RULES
{
        get(String url) { GET "$url$"; }
}
```

- La section Retrival_Rules définit la méthode d'accès à la source WEB à partie d'une adresse URL fournie. La méthode « Get » du HTTP est par défaut utilisée.
- La section Extraction_Rules détermine quelles sont les données à extraire. Une règle d'extraction est composée du nom pour référencer la donnée et d'une expression HEL.
- La section SCHEMA définit comment et sous quel types le *Wrapper* va exporter les données aux applications utilisatrices. Dans l'exemple, les résultats sont : le titre en tant qu'objet String de Java, l'année en tant qu'entier (int), le genre en tant que tableau String[ ] et les acteurs en tant que tableau double String[ ][ ] contenant leurs noms et prénoms.

La version récente de l'implantation du W4F contient des utilitaires visuels pour fournir de l'aide à l'utilisateur dans la spécification des *Wrapper*s [A. Sahuguet and F. Azavant, 00]. Un des ces outils permet d'avoir le chemin d'une donnée à l'intérieur de l'arbre du document. Ce chemin est utilisé dans l'expression des règles. Il apparaît sous forme de "*ToolTips*" quand l'utilisateur pointe la donnée avec la souris sur son *browser*. Si cela est une aide appréciable pour l'utilisateur, encore faut-il que ce dernier écrive la spécification de son *Wrapper* "à la main".

5.1.2 JEDI : Java Extraction and Dissemination of Information

JEDI est un ensemble d'outils pour générer des *Wrapper*s pour extraire les données des sources textuelles. Cela concerne aussi bien les sources WEB que d'autres fichiers de texte. JEDI est un projet de l'institut IPSI (Integrated Publication and Information Systems Institute) en Allemagne [Huck *et al*, 98].

Si JEDI permet de représenter les données extraites en plusieurs formats, la construction de documents XML à partir des sources est l'une des applications les plus intéressantes. Comme W4f, JEDI se base aussi sur des *Wrapper*s.

Le but d'un *Wrapper* XML est d'indiquer comment générer une représentation en XML des données extraites. De manière pratique, un *Wrapper* est un document texte qui contient un ensemble de directives d'extraction et de transformations qui comprend :

- Des règles permettant : (une partie descriptive)
  - Une description déclarative de la structure syntaxique de la source ;
  - L'extraction des données en les assignant à des variables ;
  - La construction de nouvelles données comme résultat correspondant aux règles.
- Code de contrôle :
  - Récupérer les données de la source ;
  - Evaluer les règles sur ces données ;
  - Construire les résultats partiels ;
  - Générer le résultat (document XML) à partir de l'évaluation des règles.

La structure syntaxique d'une source est décrite sous forme d'un ensemble de règles. Chaque règle contient une contrainte que les données (chaînes de caractères) doivent satisfaire de manière syntaxique. Une règle JEDI s'écrit sous forme [Huck *et al*, 98] :

'rule' identifier 'is'
production
'end'

La règle suivante cherche tous les nombres d'un texte donné. Ceux-ci sont stockés dans la variable ***num***.

Pour un texte tel que "le prix de 10 actions de Bell Canada est de 700$", la variable ***num*** récupère '10' et '700'.

```
rule number is
    [0-9]+
end
rule numbers : num is
 (
    num+=number()
 )+
end
```

La construction des données nécessite une partie du code (ou contrôle) pour formater les résultats selon le format voulu. L'exemple suivant montre la génération d'un élément XML contenant le jour, le mois et l'année extraits d'un document. La règle cherche dans le texte une chaîne de caractère de la forme mois/jour/année; ces trois valeurs étant stockées sous forme de trois variables. Le constructeur de la règle utilise alors ces trois variables pour produire l'élément [Huck *et al*, 98].

```
rule data : result is
  m = [0 – 9]+ '/'
  d = [0 – 9]+ '/'
  y = [0 – 9]+

  do
   result=<date>
                <day>d</day>,
                <month>m</month>,
                <year>y</year>
          </date>;
  end
end
```

En plus des spécifications concernant l'extraction des données, la spécification du *Wrapper* XML nécessite du code additionnel écrit en utilisant le langage de script JEDI. Le but de ce code est de :

- Récupérer les données de la source à partir des entrées/sorties en utilisant le nom de fichier ou l'adresse URL ;
- *Parser* le document récupéré ;
- Intégrer les résultats partiels (fournis par les règles) et stocker le résultat final, par exemple, en un document XML.

Les deux outils que nous avons présentés ci-dessus font appel respectivement à deux langages d'extraction propres à ces outils. Ceci implique que l'utilisateur doit apprendre à manipuler le langage d'extraction pour écrire les *Wrapper*s. En effet, ce type de langage utilise sa propre syntaxe qui fait intervenir des déclarations et manipulations de variables et éventuellement de scripts. Ce qui ressemble pratiquement à de la programmation. De ce fait, les utilisateurs peuvent trouver cette contrainte difficile à surmonter. D'autre part, cette difficulté empêche l'utilisation des outils graphiques pour générer automatiquement des *Wrapper*s. Ceci est dû à la complexité du langage utilisé. Nous avons vu que W4F propose un outil pour aider l'utilisateur à écrire un *Wrapper*, cependant, ceci se limite à aider l'utilisateur à trouver le chemin de la donnée à l'intérieur de la source et ne permet pas de produire automatiquement un *Wrapper*.
Pour ces raisons, nous n'avons pas utilisé ce type de langage d'extraction (celui de JEDI ou de W4F), et par conséquent, nous avons pensé à une approche similaire mais plus simple du point de vue utilisation et qui offre plus de possibilités d'améliorations futures.
Dans ce qui reste de ce chapitre, nous discutons la conception générale de l'approche et présentons le prototype que nous avons réalisé pour son implantation.

5.2 Conception de l'extraction des données des pages HTML

5.2.1 Approche pour l'extraction des pages HTML

La méthode que nous allons utilisr consiste à transformer la page HTML en un document XML et y appliquer des requêtes pour récupérer les données voulues. La transformation vise à corriger les erreurs et les irrégularités présentes dans le document HTML, obtenant ainsi un document XML bien formé. Cette opération est nécessaire afin de pouvoir appliquer les requêtes. En effet, la plupart des langages de requêtes sur un document XML exigent une représentation DOM de ce dernier, chose qui n'est pas possible si le document n'est pas bien formé.

5.2.1.1 Transformer HTML en XML

En général, une page HTML est un document essentiellement conçu dans une perspective d'affichage élaboré (graphique, couleurs et texte). La visualisation d'une page WEB sur un *browser* ne signifie pas que l'auteur de la page l'a bien codée. En fait, la plupart des *browser*s arrivent à afficher une page même en présence de beaucoup d'erreurs.
Même si le consortium W3C a produit plusieurs spécifications et directives pour l'écriture des documents HTML, le problème reste posé et la souplesse des *browser*s ne l'arrange pas. Malheureusement, ces erreurs de codage et d'organisation des balises empêchent la construction

du modèle DOM pour la page. Nous montrons dans cette partie des exemples d'erreurs et présentons des outils pour y remédier.

5.2.1.2 Quelques cas des erreurs rencontrées

Les erreurs les plus fréquentes rencontrées dans les documents HTML concernent l'oubli des balises fermantes, l'emboîtement incorrect des balises et l'oubli des attributs.

- Absence de balise fermante

```
<h1>Titre1
<h2> Sous titre 2</h2>
<Li>premier élément
<Li>deuxième élément
```

- Emboîtement incorrect

```
<b>gras<i>gras et italic</b>italic ?</i>
```

5.2.1.3 Outils utilisés pour la conversion

Pour surmonter ce problème d'erreurs dans les documents HTML et qui empêchent sa lecture et son traitement par un *parser*, nous avons fait appel à un outil de correction des documents HTML. Il s'agit de HTML TIDY [URL-5-2]. Il est disponible sur le site de W3C [URL-5-3] sous forme d'une application exécutable ou sous forme d'un programme en langage C. Dans notre cas, nous avons utilisé une version implantée en java par [URL-5-4] dans sa version "30July2000". HTML TIDY est compatible avec la spécification HTML 4.0. Le travail de HTML TIDY consiste à utiliser des heuristiques pour corriger un document. Si une décision ne peut être prise concernant la manière de corriger une erreur, elle sera signalée à l'utilisateur sous forme de WARNING ou ERROR selon son degré d'importance. L'outil permet aussi de générer le résultat (de la correction) sous forme de document XML correctement formé. C'est dans cette option que nous l'avons utilisé.

Ceci dit, le résultat final n'est pas toujours garanti et il est possible que des erreurs demeurent dans le document. En effet, durant le développement de notre prototype, nous avons constaté que dans certains cas nous ne pouvions obtenir un arbre DOM de la page. Les erreurs étaient trop nombreuses et complexes à trouver et à résoudre par des heuristiques. Ce qui donnait comme résultat un document XML non bien formé ne pouvant pas être lu par un *parser* XML. HTML

TIDY s'est révélé insuffisant pour notre prototype car le type de pages HTML que nous pouvions traiter étaient trop limité.

Dans le but de maximiser l'ensemble des sources WEB susceptibles d'être interrogées, nous avons fait appel à un deuxième procédé pour transformer un document HTML en document XML bien formé. Nous avons utilisé une partie du processus d'extraction de W4F (présenté auparavant) qui concerne la correction de la page HTML. Les outils chargent le document HTML en un arbre DOM avant d'en extraire les données à l'aide du langage HEL. Durant cette phase, une procédure de correction du document est engagée. Pour W4F, la correction se base aussi sur des heuristiques, lesquelles ne sont pas détaillées dans la documentation de l'outil.

Cependant, nous avons constaté que HTML TIDY s'occupe surtout des erreurs de la structure (balises manquantes et non proprement emboîtées), tandis que W4F permet de remédier aux erreurs de la syntaxe et du codage.
Nous avons ainsi utilisé les deux outils "en série", ce qui nous a permit d'extraire les données des sources encore non disponibles avec seulement HTML TIDY ou W4F.

Une fois que l'on a pu transformer la page en un document XML, l'accès aux données se fait à l'aide d'un *parser* DOM. Dans notre cas, nous avons utilisé l'implantation de Sun Microsystems du DOM la version 1.0 de JAXP. L'accès à une donnée se fait de manière hiérarchique en suivant son chemin à partir de la racine du document. La figure 5-2 illustre cette procédure de conversion :

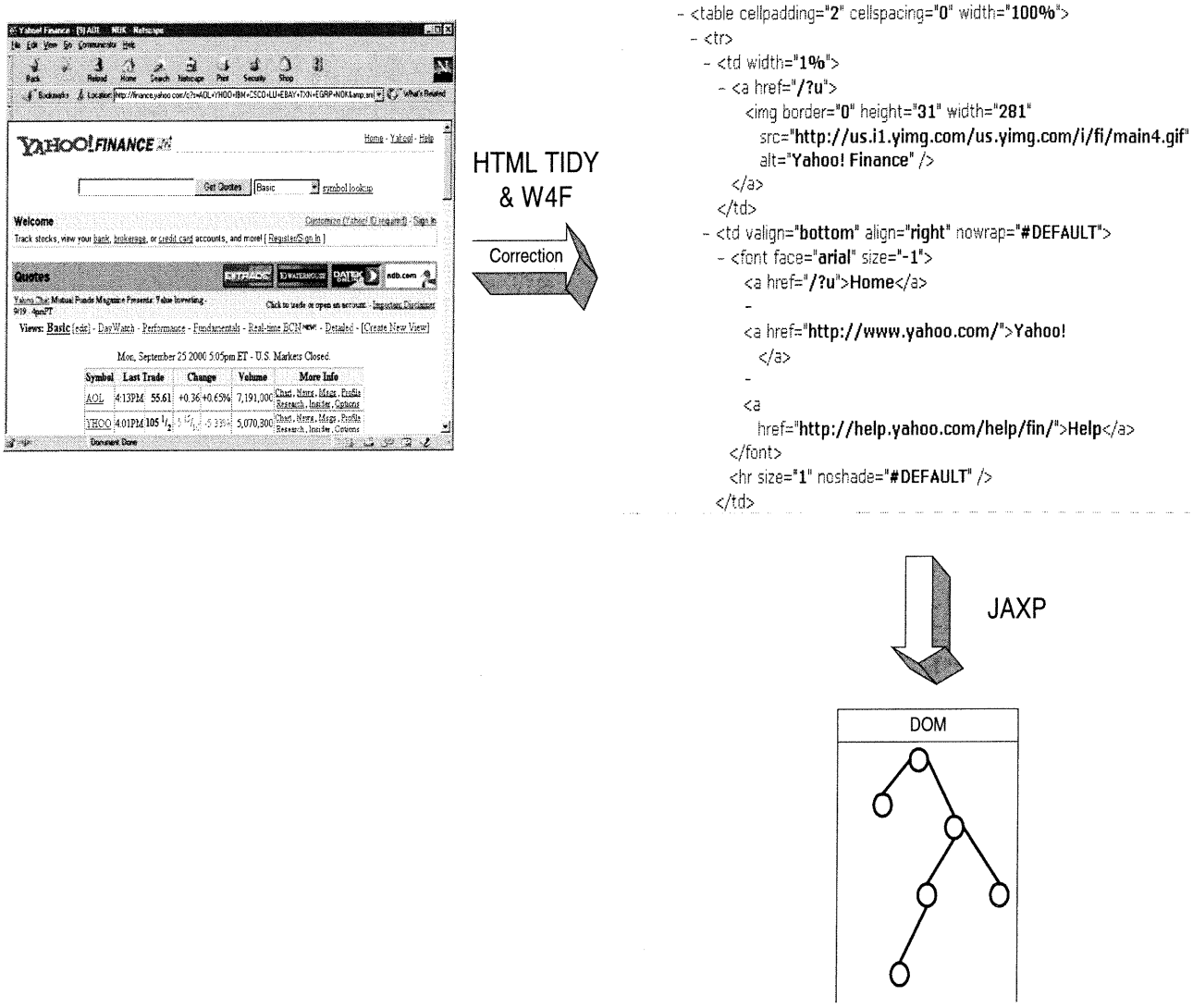
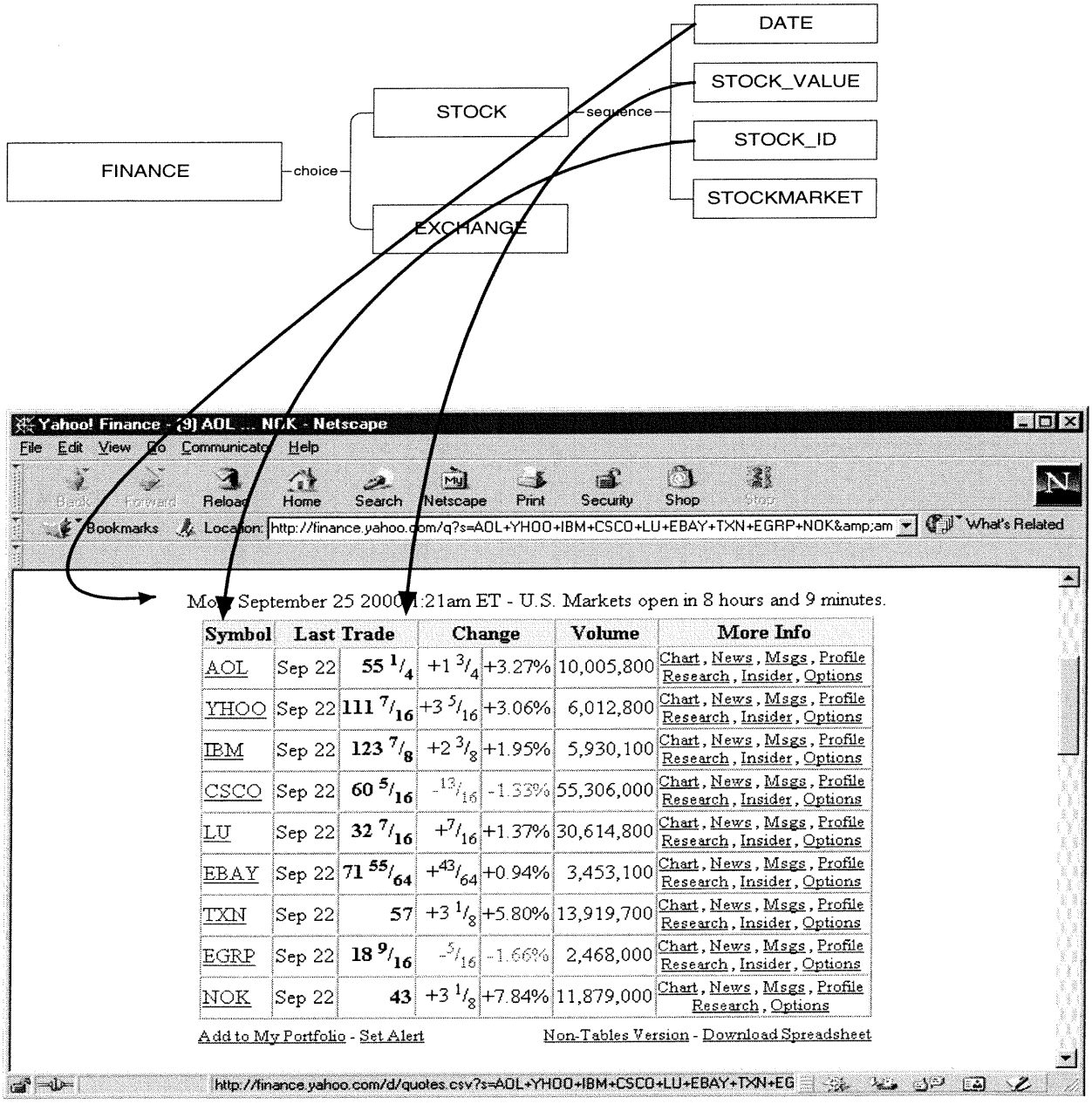


Figure 5-2 : conversion d'une page HTML en XML

5.2.2 Liaison "Mapping" avec l'ontologie

Comme nous en avons discuté dans le chapitre précédent, l'approche générale que nous avons choisie se base sur une description du domaine de travail sous forme d'ontologie. En effet, les données extraites seront liées aux éléments de l'ontologie. Chaque donnée récupérée de la page WEB sera liée avec un élément de l'ontologie qui la représentera.

La figure 5-3 montre la construction de l'élément <STOCK>, qui lui-même fait partie de l'élément <FINANCE>. Elle indique aussi quelles sont les données, sur la pages WEB, qui correspondent aux éléments de l'ontologie.



Symbol	Last Trade		Change		Volume	More Info
AOL	Sep 22	55 1/4	+1 3/4	+3.27%	10,005,800	Chart, News, Msgs, Profile Research, Insider, Options
YHOO	Sep 22	111 7/16	+3 5/16	+3.06%	6,012,800	Chart, News, Msgs, Profile Research, Insider, Options
IBM	Sep 22	123 7/8	+2 3/8	+1.95%	5,930,100	Chart, News, Msgs, Profile Research, Insider, Options
CSCO	Sep 22	60 5/16	-13/16	-1.33%	55,306,000	Chart, News, Msgs, Profile Research, Insider, Options
LU	Sep 22	32 7/16	+7/16	+1.37%	30,614,800	Chart, News, Msgs, Profile Research, Insider, Options
EBAY	Sep 22	71 55/64	+43/64	+0.94%	3,453,100	Chart, News, Msgs, Profile Research, Insider, Options
TXN	Sep 22	57	+3 1/8	+5.80%	13,919,700	Chart, News, Msgs, Profile Research, Insider, Options
EGRP	Sep 22	18 9/16	-5/16	-1.66%	2,468,000	Chart, News, Msgs, Profile Research, Insider, Options
NOK	Sep 22	43	+3 1/8	+7.84%	11,879,000	Chart, News, Msgs, Profile Research, Options

Figure 5-3 : *mapping* avec l'ontologie

La liaison est faite pour construire une occurrence de l'élément <STOCK>. Nous allons voir par la suite qu'on peut avoir aussi autant d'éléments <STOCK> comme résultat d'extraction qu'il y a de données dans la page WEB satisfaisant l'ontologie. Par exemple, on peut avoir autant d'élément <STOCK> extraits qu'il y a de lignes dans le tableau de la page HTML. Le nombre

d'occurrences est déterminé par l'utilisateur à travers sa requête d'extraction (voir détail par la suite).

Nous avons fait appel au langage de requêtes XQL pour récupérer les données du document XML converti. Ce langage a été proposé pour exprimer des requêtes sur des documents XML. Il est simple, concis et facile pour utiliser. XQL et d'autres langages de requêtes sont présentés et comparés en détail dans le chapitre suivant, où nous montrons également les raisons qui ont motivé ce choix.

5.2.3 Transformations sur la donnée extraite

Il est possible que dans certains cas la donnée se trouve dans la page HTML à côté d'autres informations non intéressantes pour l'utilisateur. Par exemple dans le document suivant, l'utilisateur souhaite extraire seulement la date :

```
...
<b>
        <i> Wed, September 27 2000 1:25am ET - U.S.
                Markets open in 8 hours and 5 minutes
        </i>
</b>
...
```

Cela est dû au fait que le niveau de granularité le plus fin que l'on peut atteindre avec le modèle DOM et le langage de requêtes est le texte contenu par un élément feuille. Dans l'exemple précédent, une requête nous permet d'avoir toute la chaîne de caractères (de l'élément <i>) "*Wed, September 27 2000 1:25am ET - U.S. Markets open in 8 hours and 5 minutes*" alors que l'on veut prendre seulement la date. Pour aller plus loin dans la précision, il faut séparer les données à prendre des données à écarter. Pour cela, nous avons prévu une transformation sous forme de filtre à appliquer sur l'élément extrait.

Ce filtre est une expression qui informe l'application de l'opération de séparation à effectuer avant de délivrer la donnée finale. Cette expression est une chaîne de caractères qui contient des caractères spéciaux :

- "$" : indique la données finale à récupérer ;
- "*" : indique n'importe quelle chaîne de caractères à ignorer ;
- En plus, le filtre peut contenir n'importe quelle chaîne de caractères qui sert comme délimiteur pour la donnée finale à extraire.

Par exemple, si l'on considère l'exemple précédent et que l'on y applique le filtre "$- U.S. Makets*", le résultat final sera la date voulue : "*Wed, September 27 2000 1:25am ET* ".

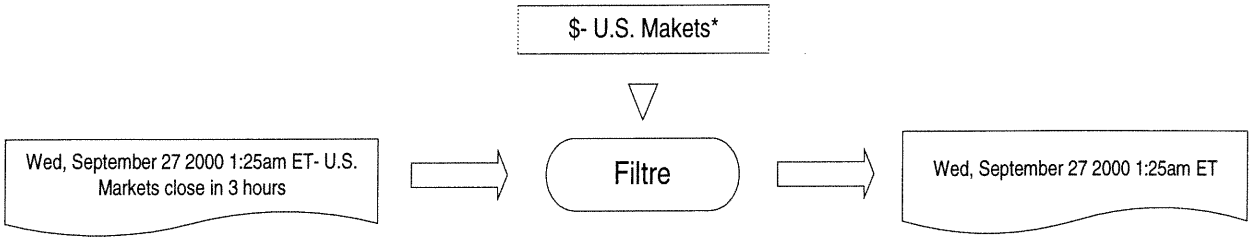


Figure 5-4 : filtre de transformation

Une autre opération, qui serait intéressante pour l'utilisateur, est la recherche de mots particuliers dans la chaîne de caractères extraite. Ceci est particulièrement utile dans le cas où un élément de l'ontologie admet plusieurs valeurs possibles et que celles-ci sont énumérées comme choix dans l'ontologie. Si l'information à prendre ne peut être repérée directement, l'utilisateur pourra chercher l'existence de l'une de ses valeurs (définies dans l'ontologie) et prendre la valeur trouvée comme résultat final. L'utilisateur pourrait spécifier d'autres mots à chercher en plus de ceux qui existent dans l'ontologie. Parmi toutes ces possibilités exprimées par une liste de mots clés, un seul est choisi. Par exemple, on recherche parmi différentes dénominations d'un terme en plusieurs langues ou parmi des valeurs connues pour un standard (on recherche le symbole d'une monnaie USD, CAD, FRF ...).

Il est aussi possible que la donnée à extraire soit éparpillée entre plusieurs éléments en différents endroits du document. Ici également, l'utilisateur ne peut rassembler la donnée en utilisant simplement les requêtes sur le document. Dès lors, une opération de fusion de parties de données est nécessaire.

Durant l'implantation, nous avons prévu les deux transformations de séparation (filtre) et de recherche de mots particuliers. De même, nous pensons que la transformation qui vise à fusionner des morceaux de donnée reste tout à fait réalisable. Cependant, elle n'a pas été prise en compte dans notre outil.

5.3 Description de l'extraction à l'aide de notre outil

5.3.1 Environnement de l'outil

Le prototype que nous avons réalisé se veut un outil d'aide à l'extraction et l'automatisation de cette opération. Le processus d'extraction passe par plusieurs étapes : charger la page HTML, la convertir en un document XML, choisir les éléments de l'ontologie, extraire les données et les lier avec ces éléments et enfin appliquer les transformations éventuelles. Le résultat de ces étapes est une description qui reprend tout le processus sous forme de spécification (description) relative à une source donnée.

Nous allons aborder, dans ce qui suit, le détail de l'implantation de chaque phase. Nous commençons tout d'abord par décrire de manière générale l'environnement d'utilisation de ce prototype. L'outil est développé de telle manière à mettre en relief chacune des étapes énumérées ci-dessus et à faciliter le test des spécifications produites.

L'utilisateur travaille en terme de projets ; chacun concernant une seule ontologie tout en pouvant référencer plusieurs descriptions de sources. Naturellement, une description d'une source doit se conformer à une seule ontologie et peut faire partie de plusieurs projets en même temps à condition d'avoir la même ontologie (figure 5-5). Ceci permet d'avoir des descriptions des sources indépendantes des projets et assure une certaine "réutilisabilité" des descriptions.

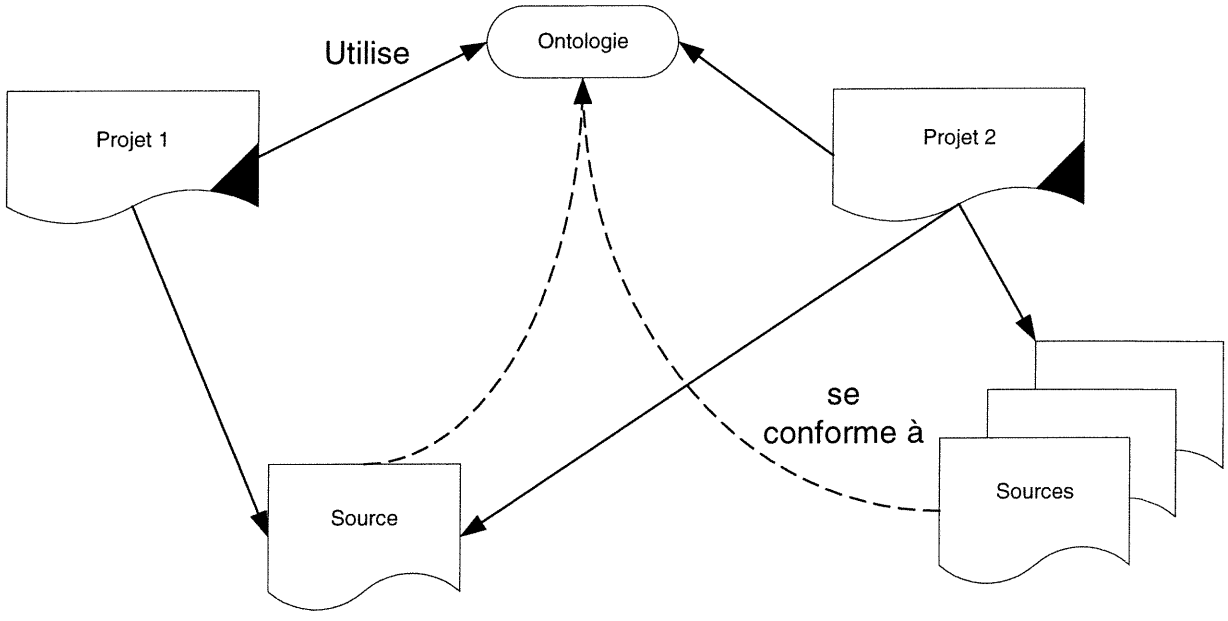


Figure 5-5 : organisation des fichiers de descriptions

5.3.2 Processus d'extraction

Le processus d'extraction automatique passe par trois étapes : chargement de la page WEB, sa conversion en un document XML, choix des éléments à partir de l'ontologie et enfin étalir les liaisons (*mapping*) entre les données de la page et les éléments de l'ontologie.

5.3.2.1 Charger la page WEB

L'outil contient un *browser* qui permet de visualiser la page HTML chargée. Dans ce cas, l'outil se comporte comme un client HTTP qui émet une demande "Get" au serveur indiqué sur l'adresse URL du document (voir figure 5-6).

5.3.2.2 Convertir le document HTML

Durant cette phase, l'utilisateur demande une conversion de son document HTML en un document XML bien formé. La partie à droite de la figure 5-6 montre une représentation graphique sous forme d'arbre du document XML obtenu. Pour cela, nous faisons appel aux librairies de HTML TIDY et de W4F, comme décrit précédemment.

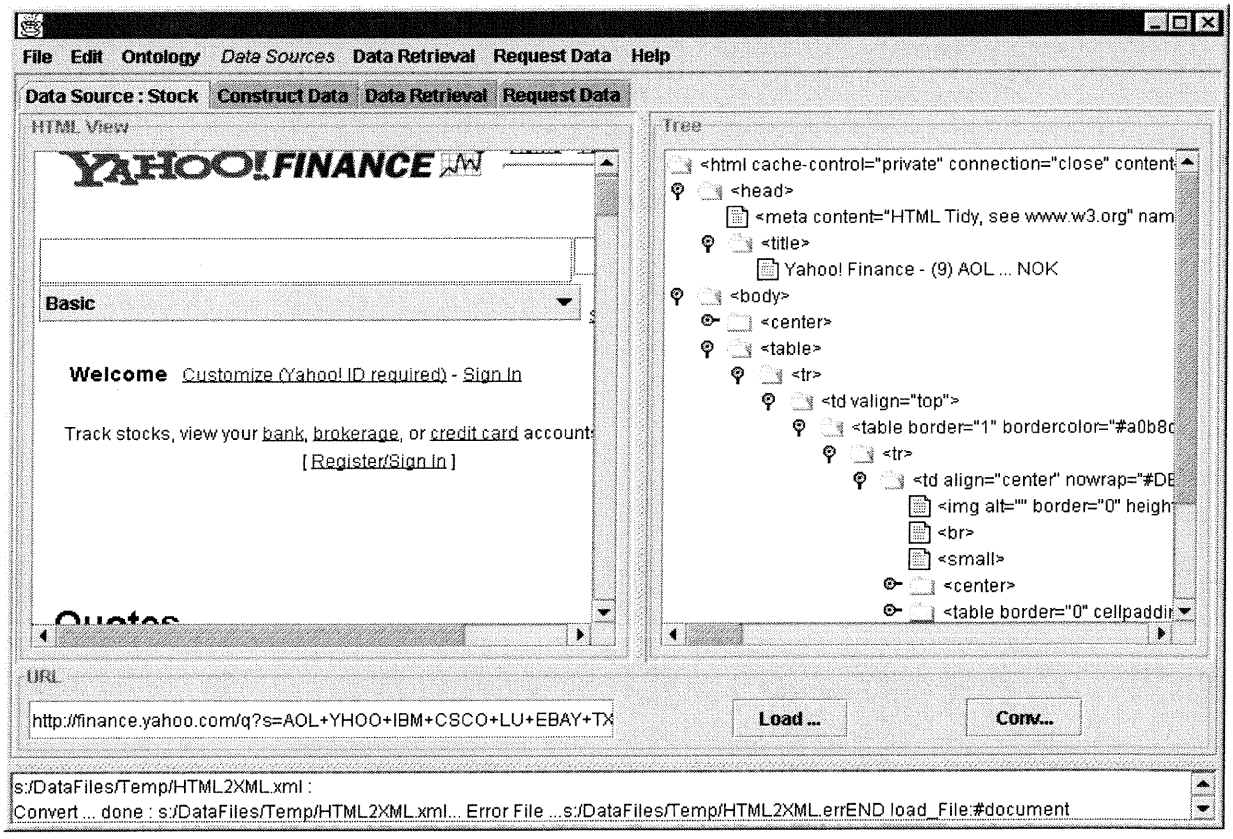


Figure 5-6 : prise d'écran pour la conversion du HTML en XML

5.3.2.3 Choix des éléments de l'ontologie

Nous avons utilisé pour l'édition de l'ontologie un outil externe, XML Authority. Cet outil produit l'ontologie sous forme de fichier "Schema" de type SOX. Puisque l'outil n'offre pas d'interface programmable (API) pour manipuler les fichiers SOX, il a fallu lire ce type de fichier et en déterminer les éléments de l'ontologie que l'utilisateur a défini précédemment. Aussi, il était nécessaire de dégager toutes les relations qui existent entre les éléments et attributs de l'ontologie et d'en constituer une vue en mémoire afin d'être utilisée au besoin le long du processus d'extraction.

L'utilisateur choisit les éléments de l'ontologie qu'il utilisera pour construire la structure de données à extraire d'une source donnée. L'utilisateur n'est pas tenu de prendre l'ontologie entière pour une source donnée, celle-ci n'offrant pas nécessairement toutes les informations voulues. La complétude des informations relevant de l'ontologie pourra par la suite être obtenue lors de la fusion des données extraites des différentes sources. En ce qui concerne la complétude de l'ontologie elle-même, nous considérons que c'est un problème de conception et que nous n'allons pas chercher à le résoudre actuellement.

Les éléments de l'ontologie que l'utilisateur choisit répondent à la question "quoi extraire ? " ou quelles sont les données auxquelles l'utilisateur est intéressé. Les deux phases de conversion et des requêtes (qui sont explicitées par la suite) répondent à la question "comment extraire ? ".
La figure 5-7 ci-après montre une copie d'écran durant la phase de la construction des données. Les colonnes de la table contiennent les informations sur les données à construire :

- le nom de l'élément pour désigner la donnée, celui-ci est naturellement pris parmi les constituants de l'ontologie. Dans la modélisation que nous avons choisie pour l'ontologie, il s'agit des noms des éléments ou des attributs (en terme de modélisation DOM d'un document XML). Lors de l'ajout d'un nouvel élément ou attribut dans la table, l'utilisateur choisit parmi les descendants de l'élément qui précède dans la table. Ce qui fait que les relations entre les éléments construits respectent la structure hiérarchique d'un document XML. Cela permet aux données extraites des sources d'être stockées dans de tels documents. Notamment, l'utilisateur prendra une seule racine pour les données, et lors de la construction des autres éléments il aura à choisir seulement parmi les descendants de l'élément qu'il a pris auparavant. La génération automatique du menu qui présente les choix disponibles aide l'utilisateur à respecter le format hiérarchique ;

- la forme que prendra la donnée extraite : "attribut" ou "élément" (si elle peut admettre d'autres composantes). Cette propriété n'est pas à déterminer par l'utilisateur, elle est déterminée à partir de l'ontologie ;
- "Presence" indique, pour l'élément, le nombre d'occurrences permies. Cette propriété indique pour l'attribut s'il est obligatoire ou optionnel ;
- la colonne "Model/Data Type" indique, pour l'élément, le modèle pour ses descendants (choix, séquence, texte) et pour l'attribut son type (String, int, float,...).

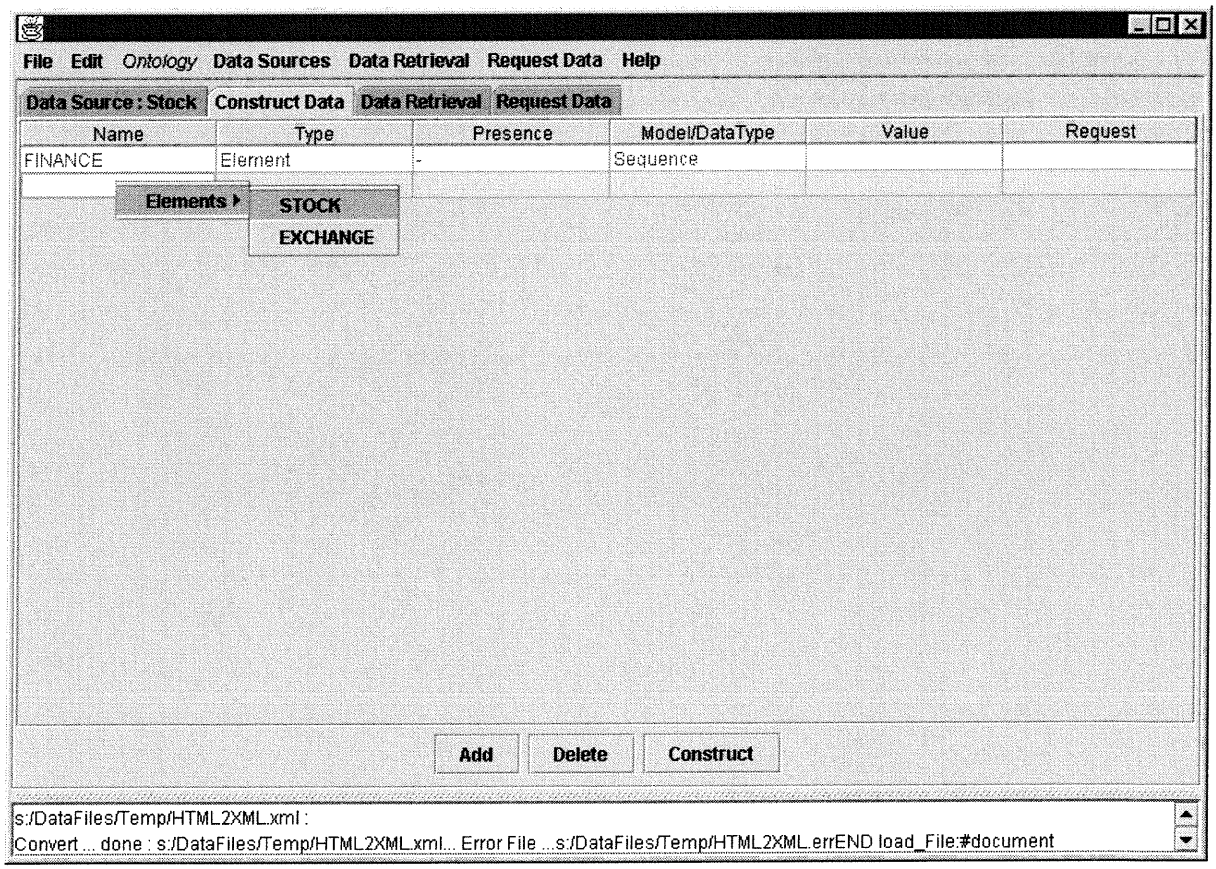


Figure 5-7 : choix des éléments de l'ontologie

5.3.2.4 Construction des liaisons avec les données : Mapping

La table sert aussi à lier les éléments/attributs de l'ontologie aux données proprement dites de la source. Pour cela, l'utilisateur disposera d'une autre fenêtre pour construire la donnée qui correspond à l'élément/attribut de l'ontologie. La figure 5-8 montre cette fenêtre qui contient une représentation en arbre de la source, un champ de saisie pour entrer la requête en XQL. Toutes les occurrences d'un élément/attribut donné qui peuvent être récupérées de la source à l'aide de cette requête sont affichées à droite.

Comme indiqué précédemment, il est possible de spécifier des transformations à effectuer sur les données si nécessaire. La même figure montre deux types d'opérations possible : "Filter" (appliquer un patron) et "Search" (recherche d'un mot clé parmi une liste possible).

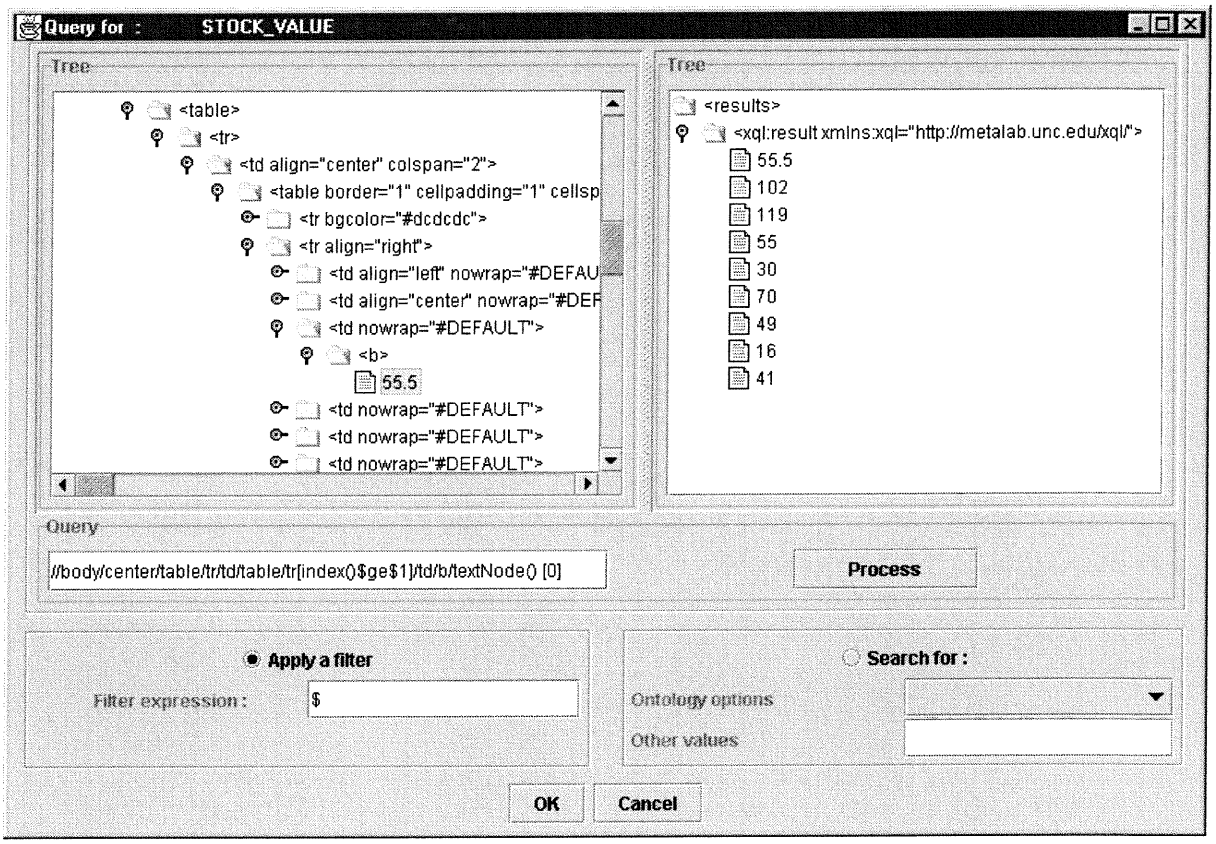


Figure 5-8 : extraction des données pour un élément de l'ontologie

Une fois que l'utilisateur a complété cette dernière phase de "mapping", il produit la spécification pour la source. La spécification reprend en quelque sorte les actions effectuées par l'utilisateur durant le processus d'extraction. Elle contient, entre autre, les noms des éléments/attributs, les requêtes, les transformations et les informations sur la structure du résultat final (le document XML). Un exemple de spécification est fourni par le document suivant (figure 5-9), il est relatif à une page de Yahoo pour les données sur les cours des actions [URL-3-1].

```
<?xml version="1.0" encoding="UTF-8"?>
<SOURCE>
      <NAME>Stock3</NAME>
      <URL>http://finance.yahoo.com/q?s=AOL+YHOO+IBM+CSCO+LU+EBAY+TXN+EGRP
+NOK&d=v1</URL>
      <ONTOLOGYFILE>file:///S:/DataFiles/schemas/n1_0/Finance.sox</ONTOLOG
YFILE>
      <ITEMS>
            <ITEM>
                  <INDEX>0</INDEX>
                  <NAMEITEM>FINANCE</NAMEITEM>
                  <TYPE>element</TYPE>
                  <ITEM>
                        <INDEX>1</INDEX>
                        <NAMEITEM>STOCK</NAMEITEM>
                        <TYPE>element</TYPE>
                        <ITEM>
                              <NAMEITEM>DATE</NAMEITEM>
                              <TYPE>Element</TYPE>
                              <INDEX>2</INDEX>
                              <QUERY>//body/center/p[0]</QUERY>
                              <TRANS TYPE="FILTER">$-*</TRANS>
                        </ITEM>
                        <ITEM>
                              <NAMEITEM>STOCK_VALUE</NAMEITEM>
                              <TYPE>Element</TYPE>
                              <INDEX>3</INDEX>

      <QUERY>//body/center/table/tr/td/table/tr[index()$ge$1]/td/b/textNod
e() [0]</QUERY>
                              <TRANS TYPE="FILTER">$</TRANS>
                        </ITEM>
                        <ITEM>
                              <NAMEITEM>STOCK_ID</NAMEITEM>
                              <TYPE>Element</TYPE>
                              <INDEX>4</INDEX>

      <QUERY>//body/center/table/tr/td/table/tr[index()$ge$1]/td/a[@href$c
ontains$'/q']</QUERY>
                              <TRANS TYPE="FILTER">$</TRANS>
                        </ITEM>
                        <ITEM>
                              <NAMEITEM>STOCKMARKET</NAMEITEM>
                              <TYPE>Element</TYPE>
                              <INDEX>5</INDEX>
                              <QUERY>//body/center/p[0]</QUERY>
                              <TRANS TYPE="FILTER">*-$s *</TRANS>
                        </ITEM>
                  </ITEM>
            </ITEM>
      </ITEMS>
</SOURCE>
```

Figure 5-9 : fichier de description

L'élément ITEM (mis en gras) sur la figure 5-9 contient les éléments suivants :

- NAMEITEM c'est le nom de l'élément spécifié par l'ontologie qui représentera une donnée extraite ;
- TYPE exprime son type Elément ou Attribut ;
- INDEX sert à reconstruire la description lorsqu'elle est relue par l outil ;
- QUERY spécifie la requête en XQL qui permet de récupérer la donnée désignée par l'élément NAMEITEM ;
- TRANS renseigne sur une transformation éventuelle de la donnée après récupération, il contient un attribut TYPE pour spécifier le genre de la transformation FILTER ou SEARCH.

5.3.3 Résultat de l'extraction

Notre prototype permet à l'utilisateur de tester les spécifications d'extraction pour les sources. Ce prototype utilise un module d'extraction qui reconnaît et interprète les "instructions" contenues dans les fichiers de spécifications. Le module peut être utilisé indépendamment du prototype, éventuellement intégré dans une autre application ou utilisé séparément. Le rôle de ce module est de reprendre le processus de l'extraction d'une source et produire les résultats sous forme d'un document XML. Après avoir récupéré et converti la page WEB, il extrait les données relatives à chaque élément/attribut défini dans la spécification et y applique les transformations éventuelles. La mise en commun de ces données extraites se fait comme indiqué par le schéma suivant :

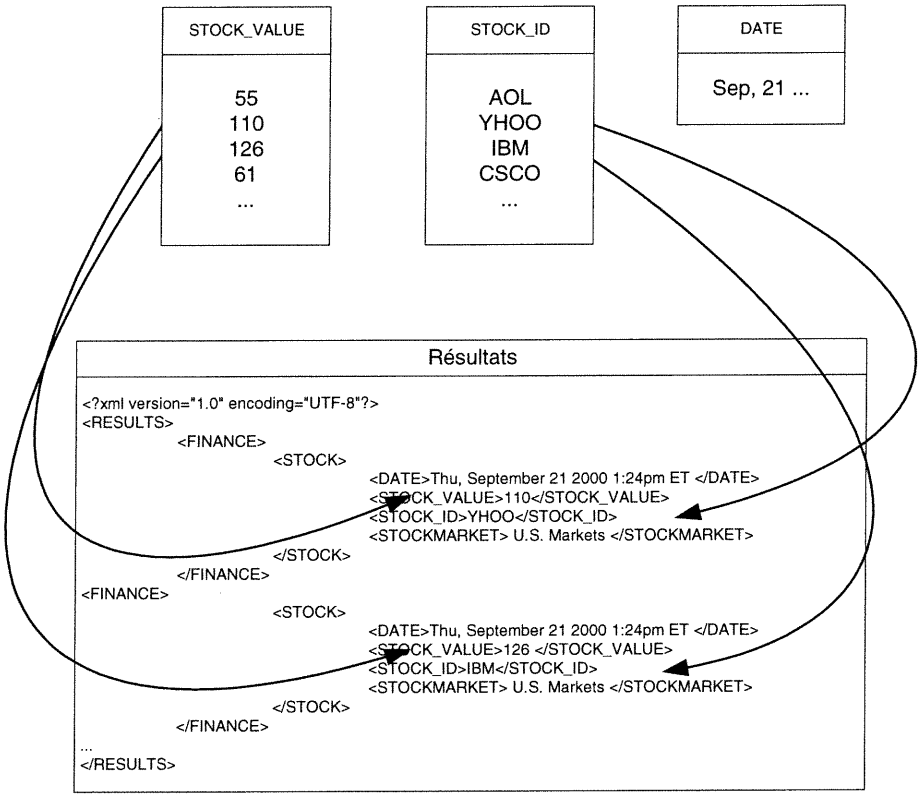


Figure 5-10 : regroupement des résultats de l'extraction

Dans l'exemple illustré par la figure 5-10, la construction de chaque élément <STOCK> se fait en utilisant respectivement les occurrences de <STOCK_ID>, <STOCK_VALUE>. Dans le cas où le nombre d'occurrence de ces éléments diffèrent, le nombre d'occurrences de l'élément final <STOCK> construit est le minimum des nombres de données disponibles pour chaque élément : <STOCK_ID>, <STOCK_VALUE>. Cette règle admet une exception dans le cas où il existe une seule occurrence pour un élément (l'élément <DATE> de l'exemple est valable et repris pour tous les éléments <STOCK>).

Le schéma suivant reprend la conception générale et les étapes d'extraction des données.

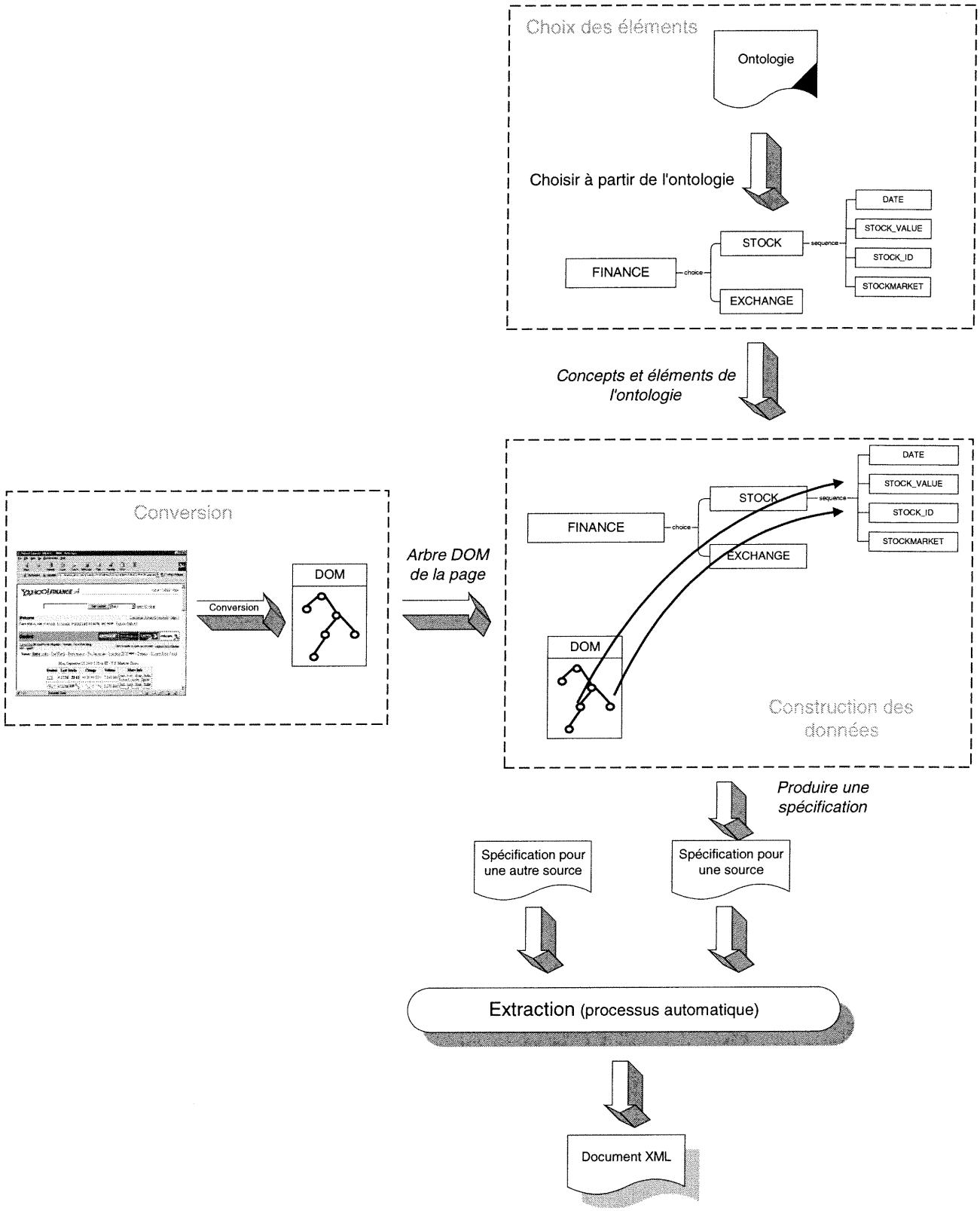


Figure 5-11 : vue globale des étapes de l'extraction

CHAPITRE 6

Requêtes sur les données extraites

6.1 Introduction

L'objectif de cette partie est de fournir la description de l'approche et des outils utilisés par notre prototype pour interroger les données extraites des sources. Nous allons décrire également les raisons qui ont motivé nos choix.

Le résultat de l'étape d'extraction des données est un document XML que l'on peut stocker dans un fichier. Ceci dit, il est possible de les mettre dans une base de données conventionnelle RDBMS ou OODBMS. Cependant, lorsqu'on désire utiliser un tel SGBD, il faut prendre en considération que les données extraites ne partagent pas un même schéma fixe et n'ont pas forcément la même structure. Par exemple, en prenant une ontologie qui contient des données optionnelles, il est possible de trouver deux sources de données différentes dont l'une ne fournit pas ces données optionnelles. Ceci dit, les deux sources obéissent toujours à cette même ontologie. Dans ce cas, la différence se situe au niveau des données optionnelles.
Par exemple l'utilisateur pourrait choisir deux spécifications différentes pour deux sources qui fournissent les valeurs des cours d'actions. Les deux sources obéissent à la même ontologie.

Première possibilité :

```
<FINANCE>
      <STOCK>
            <DATE> </DATE>
            <STOCK_VALUE> </STOCK_VALUE>
            <STOCK_ID> </STOCK_ID>
            <VARIATION> </VARIATION>
            <STOCKMARKET> </STOCKMARKET>
      </STOCK>
</FINANCE>
```

Une deuxième possibilité serait que l'utilisateur omettrait d'inclure la variation de la valeur représentée par l'élément *<VARIATION> </VARIATION>* car la source ne contient pas cette information ou que cette information n'est pas pertinente pour lui.

```
<FINANCE>
      <STOCK>
            <DATE> </DATE>
            <STOCK_VALUE> </STOCK_VALUE>
            <STOCK_ID> </STOCK_ID>
            <STOCKMARKET> </STOCKMARKET>
      </STOCK>
</FINANCE>
```

Néanmoins, nous pourrons envisager de contourner la différence entre les sources par l'assignation de "valeurs par défaut" ou "valeurs non disponibles" aux données manquantes afin de pouvoir stocker les documents XML sur des SGBD. Ces deux possibilités peuvent engendrer des difficultés pour le SGBD pour résoudre des requêtes faisant intervenir ces éléments manquants.

D'autre part, nous envisageons l'utilisation de l'extraction des données au niveau des applications ne disposant pas nécessairement d'une base de données. Le cas des agents mobiles est un exemple qui illustre bien ce genre d'application. Un agent mobile, qui se déplace d'un site à un autre et qui souhaite extraire des données sur des sites WEB, pourrait ne pas disposer d'une base de données pour les stocker. Il est donc essentiel de lui fournir des outils et des mécanismes pour faire des requêtes autrement que sur un SGBD. Le chapitre 7 décrit plus en détails notre exemple d'application faisant intervenir des agents.

Nous avons donc trouvé essentiel d'étudier les requêtes sur des documents XML, ce que nous allons développer dans cette partie.

Etant donné la nature des données recueillies à partir du WEB, les langages de requêtes pour les données relationnelles ou objet-relationnelles ne conviennent pas. Ceci est dû à ce que ces langages de requêtes nécessitent un schéma fixe et défini pour les données et ne tolèrent pas des données manquantes. En fait, le choix du langage de requête et le moyen pour stocker les données sont liés entre eux. Dès lors qu'on a choisi de faire des requêtes sur les données résultats stockées dans des fichiers XML, les langages de requêtes conventionnels ne conviennent pas.

6.2 Langages de requêtes pour documents XML

De nouveaux langages de requêtes ont été proposés pour interroger des documents XML. Certains sont inspirés des langages de requêtes traditionnels pour les bases de données tels que SQL (RDBMS) ou OQL (OODBMS). Les autres langages sont proposés en se basant sur les caractéristiques de XML lui-même. Actuellement, il n'y a pas de langage de requête standard pour XML. Des discussions sont en cours autour de plusieurs propositions à l'intérieur du groupe de travail sur les langages de requêtes du consortium W3 (World Wide Web Consortium)[Miller and Sheth, 00]. Ce groupe de travail réunit plusieurs institutions académiques et industriels du domaine.

Ci-après, un comparatif de quelques propositions de langages : LOREL, XML-QL, XQL.

6.2.1 Langages de requêtes LOREL, XML-QL et XQL

- LOREL a été initialement développé pour interroger les données semi-structurées. Il a été étendu par la suite pour les documents XML. LOREL est conçu à Stanford University. Un prototype est disponible au [URL-6-1]. LOREL est une extension de OQL avec la même structure (SELECT-FROM-WHERE), que celle du SQL/OQL, pour construire les requêtes [Goldman *et al*, 99].

- XML-QL a été proposé par AT&T Labs, University of Pennsylvania et l'INRIA. Il étend le langage SQL en utilisant le même concept SELECT-WHERE ; il introduit une nouvelle clause CONSTRUCT pour construire le document résultat de la requête [Alin *et al*, 98].

- XQL est une extension de XSL (Extensible Stylesheet Language) utilisé pour la transformation de documents XML. XQL est une proposition commune d'industriels (Texcel, WebMethods, Microsoft) [Hiroshi *et al*, 98]. XQL se base sur les caractéristiques de XSL qui

permettent d'identifier les nœuds à l'intérieur d'un document ; il y ajoute la logique booléenne, les filtres et l'indexation de ces nœuds. XQL est concis, simple et ressemble à une navigation à l'intérieur d'une structure de fichiers, ce qui correspond bien à la nature hiérarchique des documents XML.

Dans ce qui suit, nous présentons une comparaison de ces trois langages en terme de leurs modèles de données, résultats des requêtes, jointures et ordonnancement [Bonifati and Ceri, 00].

6.2.2 Le modèle de données des langages

Le modèle de données de LOREL se base sur la paire *<eid, value>* ; où *eid* est un identifiant unique pour chaque élément et *value,* qui peut être atomique ou complexe, désigne la valeur de l'élément. Le modèle est représenté par un graphe, les nœuds correspondant aux éléments XML.
Les données de XML-QL sont modélisées également par un graphe, chaque nœud étant associé avec un identifiant *OID* (*Object Identifier*).
En ce qui concerne XQL, ses concepteurs supposent que XML implique forcément l'existence d'un modèle pour les données. Ainsi, le modèle de données pour XQL est celui du document lui-même. Chaque nœud a un type et une valeur, les relations entre les nœuds sont de type hiérarchique (parent/fils), de position (absolu, relative, rang) et d'ordonnancement (précède). Ce modèle n'est pas un graphe comme pour les autres langages, mais plutôt un arbre. Nous étudierons XQL plus en détail par la suite.

6.2.3 Résultat d'une requête

Le résultat d'une requête en LOREL est un ensemble d'identifiants *eid* des éléments appartenant au document original, référencés par un nouvel élément. Le résultat d'une requête en XML-QL ou en XQL est un nouveau document.

6.2.4 Jointures

Une jointure permet de comparer une condition entre des données de plusieurs documents ou à l'intérieur du même document pour en déduire des résultats.
LOREL permet d'exprimer les jointures à l'intérieur du même document ou à travers plusieurs documents. Elles sont écrites à la manière de SQL en l'exprimant à l'aide de variables explicites.
Les jointures sont également supportées par XML-QL, à l'aide d'OID.
Cependant, l'un des reproches pour XQL est l'impossibilité d'exprimer des jointures complètes à travers plusieurs documents. Néanmoins, il est possible d'appliquer des semi-jointures, c'est-à-dire des jointures qui font intervenir des conditions sur les données d'un même document.

6.2.5 Rang ou intervalle de rang des données

Dans les requêtes exprimées en LOREL ou en XQL, il est possible d'avoir les données qui appartiennent à un rang ou à un intervalle de rang à l'intérieur d'un élément particulier. Par contre, ceci n'est pas possible pour XML-QL.

6.2.6 Ordonnancement

L'ordonnancement du résultat est défini par la manière de présenter les données résultantes, selon un ordre ascendant ou descendant d'une valeur appartenant à ces mêmes données.

Ceci est possible pour LOREL à l'aide d'une clause qui définit l'ordre selon lequel les éléments résultats doivent être retournés. Pour XML-QL, il est aussi possible de spécifier un tel ordonnancement. Cependant le résultat d'une requête XQL préserve l'ordre des éléments dans leurs emplacements originaux dans le document XML initial.

XQL élimine les répétitions des éléments dans les résultats, ce qui n'est pas souhaitable dans plusieurs cas. En effet, l'utilisateur pourrait vouloir garder même les éléments qui se répètent. Ceci dit, il y a un moyen de contourner cette faiblesse. Notons par ailleurs que l'un des avantages de XQL est que l'utilisateur peut faire évaluer sa requête à n'importe quel niveau hiérarchique du document.

6.2.7 Exemples

Ci-après des exemples de requêtes écrites en utilisant les différents langages présentés ci-dessus. Les requêtes concernent le document XML suivant, lequel contient les informations sur les actions des sociétés :

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCFINANCE>
      <FINANCE>
            <STOCK>
                  <DATE>Thu, September 21 2000 1:24pm ET </DATE>
                  <STOCK_VALUE>41 </STOCK_VALUE>
                  <STOCK_ID>NOK</STOCK_ID>
                  <STOCKMARKET> U.S. Markets </STOCKMARKET>
            </STOCK>
      </FINANCE>
      <FINANCE>
            <STOCK>
                  <DATE>Thu, September 21 2000 1:24pm ET </DATE>
                  <STOCK_VALUE>110</STOCK_VALUE>
                  <STOCK_ID>YHOO</STOCK_ID>
                  <STOCKMARKET> U.S. Markets </STOCKMARKET>
            </STOCK>
      </FINANCE>
      <FINANCE>
            <STOCK>
                  <DATE>Thu, September 21 2000 1:24pm ET </DATE>
                  <STOCK_VALUE>18 </STOCK_VALUE>
                  <STOCK_ID>EGRP</STOCK_ID>
                  <STOCKMARKET> U.S. Markets </STOCKMARKET>
            </STOCK>
      </FINANCE>
      <FINANCE>
            <STOCK>
                  <DATE>Thu, September 21 2000 1:24pm ET </DATE>
                  <STOCK_VALUE>61 </STOCK_VALUE>
                  <STOCK_ID>CSCO</STOCK_ID>
                  <STOCKMARKET> U.S. Markets </STOCKMARKET>
            </STOCK>
      </FINANCE>
</DOCFINANCE>
```

La requête pour récupérer les données des actions qui valent au maximum 50$ est écrite ainsi :

En LOREL :

select S
from stoks.STOCK S
where S. STOCK_VALUE <= 50

En XML-QL :

```
WHERE   <FINANCE>
        <STOCK>
        ...
        < STOCK_VALUE >$v</ STOCK_VALUE >
        </STOCK>
        </FINANCE> ELEMENT_AS $s IN stocks.xml, $v
    <=50
CONSTRUCT $s
```

En XQL :

//STOCK[STOCK_VALUEN_lt50]

A remarquer que l'opérateur "N_lt" ne figure pas parmi les opérateurs de comparaison de la spécification originale du langage. La version implantée du langage que nous avons utilisée [URL-6-2] permet à l'utilisateur d'ajouter ses propres expressions de comparaison ou d'évaluation. Cette possibilité rend le langage parfaitement extensible et dès lors pouvant répondre à des besoins particuliers de la part des utilisateurs. Ainsi, nous avons ajouté et défini un certain nombre d'opérateurs qui nous permettent de faire des comparaisons entre nombres entiers ou réels. Les opérateurs de comparaison proposés par défaut considèrent les données en tant que chaînes de caractères.

Voici le résultat de la requête en XQL :

```
<xql:results>
      <STOCK>
            <DATE>Thu, September 21 2000 1:24pm ET </DATE>
            <STOCK_VALUE>41 </STOCK_VALUE>
            <STOCK_ID>NOK</STOCK_ID>
            <STOCKMARKET> U.S. Markets </STOCKMARKET>
      </STOCK>
      <STOCK>
            <DATE>Thu, September 21 2000 1:24pm ET </DATE>
            <STOCK_VALUE>18 </STOCK_VALUE>
            <STOCK_ID>EGRP</STOCK_ID>
            <STOCKMARKET> U.S. Markets </STOCKMARKET>
      </STOCK>
</xql:results>
```

6.2.8 Tableau comparatif

Le tableau suivant résume la comparaison entre LOREL, XML-QL et XQL.

	LOREL	XML-QL	XQL
Modèle de données	Graphe	Graphe	Document XML (arbre)
Résultats	Ensemble de OID	Document XML	Document XML
Jointure	Complète	Complète	Semi-jointure
Requête sur plusieurs Documents	Oui	Oui	Non
Ordonnancement des résultats	Oui	Oui	Ordre du document original
Variables	Oui	Oui	Non
Quantificateur	Oui	Oui	Oui

Tableau 6-1 : tableau comparatif des langages de requêtes LOREL, XML-QL et XQL

S'il paraît que LOREL et XML-QL semblent avoir les qualités d'un langage de requêtes complet puisqu'ils sont inspirés des langages de requêtes traditionnels, XQL convient mieux à la nature et à la structure des documents XML. Il est simple, concis et assez expressif pour les besoins de telles données. Aussi, le fait que les requêtes en XQL soient exprimées en une chaîne de caractères, son intégration au niveau des adresses URL ou des documents HTML ou dans des scripts est tout à fait possible. Par ailleurs, la nature extensible de XML implique que le langage doit être ouvert à ces extensions, ce qui est possible en XQL. La simplicité de XQL permet également de faire des transformations des requêtes construites à l'aide d'un environnement visuel en requêtes XQL.

A ces raisons, s'ajoutent des considérations pratiques. LOREL nécessite l'utilisation d'une base de données *Lore* [Goldman et al, 99] ; quant à XML-QL, l'implantation que nous avons trouvée est compatible seulement avec JDK 1.1 sous Unix, ce qui pose des problèmes de cohérence avec l'ensemble de l'application.

En attendant des améliorations de XQL pour supporter les jointures, nous laisserons aux applications utilisatrices la gestion d'une telle exigence et nous considérons que XQL répond actuellement à nos besoins de prototypage. Actuellement, nous n'avons pas utilisé de jointure dans notre outil et nous espérons que XQL le permettra dans les prochaines versions.

Le choix de XQL pour répondre aux requêtes est motivé par le fait qu'il permet d'écrire des requêtes simple, concises et adéquates pour être incluses comme une chaîne de caractères dans des programmes, scripts ou fichiers de description. Cependant, l'utilisateur pourrait faire appel aux autres langages de requêtes si ses besoins l'exigent.

6.3 Construction des requêtes

Une fois que l'utilisateur a pu extraire les données, il peut les interroger et y appliquer des requêtes. Le but de cette partie est de présenter le langage que nous avons utilisé : XQL. Nous allons décrire aussi un module que nous avons développé pour aider les utilisateurs à construire les requêtes de manière visuelle.

6.3.1 Le langage XQL

6.3.1.1 Critères de conceptions

Initialement, les exigences de XQL n'étaient pas seulement fonctionnelles. En prenant en compte ses éventuelles utilisations futures, certaines contraintes ont été prises en considération. Celles-ci concernent essentiellement les applications sur le WEB. Ainsi, les requêtes peuvent être facilement incluses dans des adresses URL, prises par des langages de programmation ou des scripts, insérées dans des attributs de documents [URL-6-3].

De plus, le choix d'avoir une requête simple et compacte permet de l'utiliser facilement dans des lignes de commande, de la construire visuellement ou de l'insérer dans des programmes.

En reprenant l'exemple de la requête en XQL :

//STOCK[STOCK_VALUEN_lt50]

Cette même requête reprise dans une adresse URL pour être traité éventuellement par un script sur le serveur :

http://www.site-exemple.com/stocks#*//STOCK[STOCK_VALUEN_lt50]*

ou mise à l'intérieur d'un document HTML :

<a href="http://www.site-exemple.com/stocks#*//STOCK[STOCK_VALUEN_lt50]*">

6.3.1.2 Ecriture des requêtes en XQL

XQL se base sur les informations implicites contenues dans le document lui-même, celles-ci constituant son modèle de données. Chaque nœud d'un document a un type et une valeur ou un contenu ; les éléments et les attributs ont un nom. XQL se base sur ces propriétés pour sélectionner les éléments et formuler des conditions sur les données. Il est à noter également que les relations entre les éléments contiennent une bonne partie des informations sur le document, notamment, les relations de hiérarchie et de séquence. L'écriture des requêtes fait intervenir les informations suivantes [URL-6-3] :

- La structure du document :
  - Hiérarchie : Parent / Enfant, Ancêtre / Descendant ;
  - Séquence : Précède, Précède immédiatement ;
  - Position : Absolue, Relative, Rang.

- Contexte de recherche :

Le contexte de recherche est l'ensemble de nœuds d'un documents que la requête évalue. Un processeur de requêtes exécute une requête dans un contexte de recherche donné. Ce dernier peut être la racine du document ou un ensemble de nœuds fournis à l'environnement du processeur.

- Les conditions sur les nœuds concernent :
  - Condition sur le nom de l'élément ou l'attribut (*STOCK_VALUE*, @id) (le "@" désigne un attribut) ;
  - Condition sur le contenu ou la valeur (*STOCK_VALUE* = "50") ;
  - Condition sur le type du nœud (*Element, Attribute, PI, Text*, ...).

- Opérateur de retour :

Les conditions sur les nœuds et sur les relations entre les nœuds sont combinées pour former des expressions de chemin. Une requête cherche les chemins qui satisfont à ces conditions ; les opérateurs de retour indiquent les nœuds à sélectionner parmi ces chemins. XQL définit deux opérateurs de retour :

- "?" pour le nœud seulement ;
- "??" pour récupérer le nœud et ses descendants.

Dans le cas où la requête ne contient pas des opérateurs de retour explicites, les nœuds retournés sont ceux qui ont été évalués. La différence entre les deux est que l'évaluation concerne les nœuds qui figurent dans les opérations de comparaison tandis que les nœuds retournés sont ceux qui constituent le résultat. Les nœuds retournés peuvent être marqués explicitement par un "?" ou "??", sinon les nœuds évalués sont retournés. Les nœuds évalués ne sont pas donc forcément ceux qui sont retournés. Il y a deux manière de sélectionner un nœud :

- Il existe un opérateur de retour qui indique un tel nœud.
- S'il n'y a pas d'opérateur de retour explicite, les nœuds évalué sont sélectionnés ;

Dans l'exemple qui suit, on s'intéresse à l'élément STOCK tout entier

//STOCK[STOCK_VALUE?N_lt50]

Ci-après un tableau qui résume les opérateurs utilisés dans une requête par ordre décroissant de priorité :

Opération	Opérateur
Groupement	( )
Filtre	[ ]
Chemin	/, //
Comparaison	=, !=, <, <=, >, >=, eq, ne, lt, le, gt, ge, ieq, ine, ilt, ile, igt, ige
Intersection	$intersect$
Union	$union$, |
Négation	not
Conjonction	and
Disjonction	or

Tableau 6-2 : opérations et opérateurs de XQL

Les résultats sont des documents XML bien formés. Il est possible que le résultat ne contienne pas un seul nœud racine (dans le cas où la requête fournirait des nœuds indépendants). Ces élément seront mis sous une racine unique (<xql :results> </xql :results>) pour en construire un document bien formé.

Le tableau suivant donne une comparaison sommaire entre SQL et XQL :

	SQL	XQL
Stockage des données	Tables (relations)	Documents XML
Identification des données à récupérer	SELECT attrib1, attrib2,	Nœud_Père/Nœud_Fils
Filtre de requête	WHERE	Nœud[]
Condition	Attrib1=100	Element=100, @Attr=100
Résultat de la requête	Table (relation)	Document XML

Tableau 6-3 : comparaison sommaire entre SQL et XQL

6.3.2 Construction visuelle

Souhaitant simplifier l'interrogation des données extraites pour l'utilisateur, nous avons expérimenté un module de construction visuelle des requêtes (figure 6-1). Ceci déchargera l'utilisateur d'une connaissance préalable du langage de requête. Ainsi, la construction d'une requête est simplifiée au maximum pour ces utilisateurs. L'idée est de permettre une construction visuelle qui sera transformée par la suite en XQL. La construction visuelle (ou graphique) des requêtes et l'opération de transformation qui s'en suit ont été facilitées par la nature du langage XQL.

6.3.2.1 Opérations de base

La construction visuelle se base sur deux opérations Get et Condition. La première indique les éléments que l'utilisateur choisit comme résultats de retour. La deuxième exprime simplement une condition à satisfaire parmi les valeurs des éléments/attributs du document de base ; elle joue le rôle du WHERE dans SQL.

L'expression des requêtes ne se base pas sur les fichiers des données extraites mais au contraire sur l'ontologie ayant servi à la description des données à extraire des sources. Dans le contexte de l'application prototype réalisée, le module de requête lit la spécification (qui est elle-même basée sur l'ontologie) de la source. Il en déduit tous les éléments/attributs qui sont susceptibles d'être demandés par l'utilisateur. Pour chacun de ces derniers, il détermine son type. Dans un souci de simplification, nous avons pris en considération deux type de données primitives : les nombres (entiers et réels) et les chaînes de caractères. La connaissance, au préalable, du type de la donnée

permet d'en déduire automatiquement les opérations de comparaisons possibles sur l'élément/attribut. Du point de vue pratique, nous avons choisi de fournir les expressions "*contains*" et "*equals*" pour comparer et chercher les chaînes de caractères, et les expressions classiques ("<", ">", "<=", ">=", " != ", "=") pour les nombres. La figure 6-1 suivante montre la phase de construction des requêtes de la part de l'utilisateur.

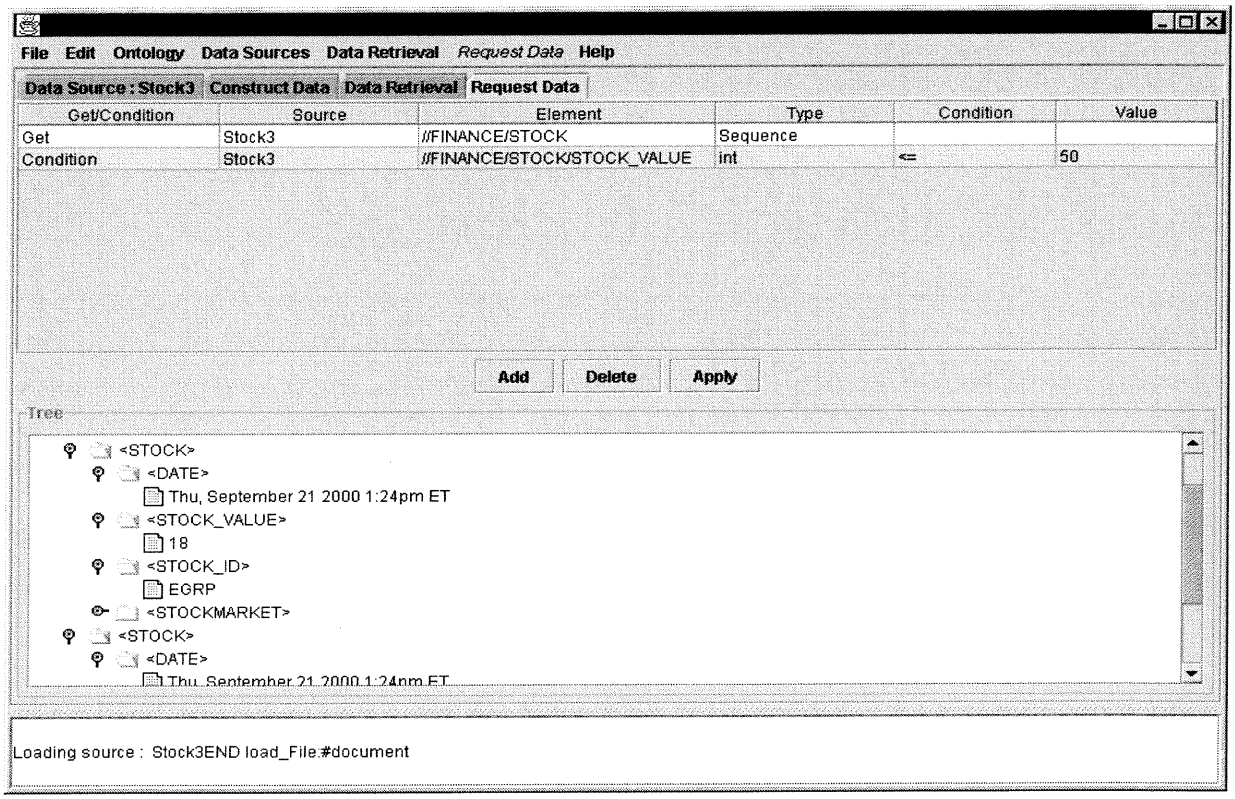


Figure 6-1 : exécution des requêtes à l'aide de notre outil

6.3.2.2 Schéma des requêtes

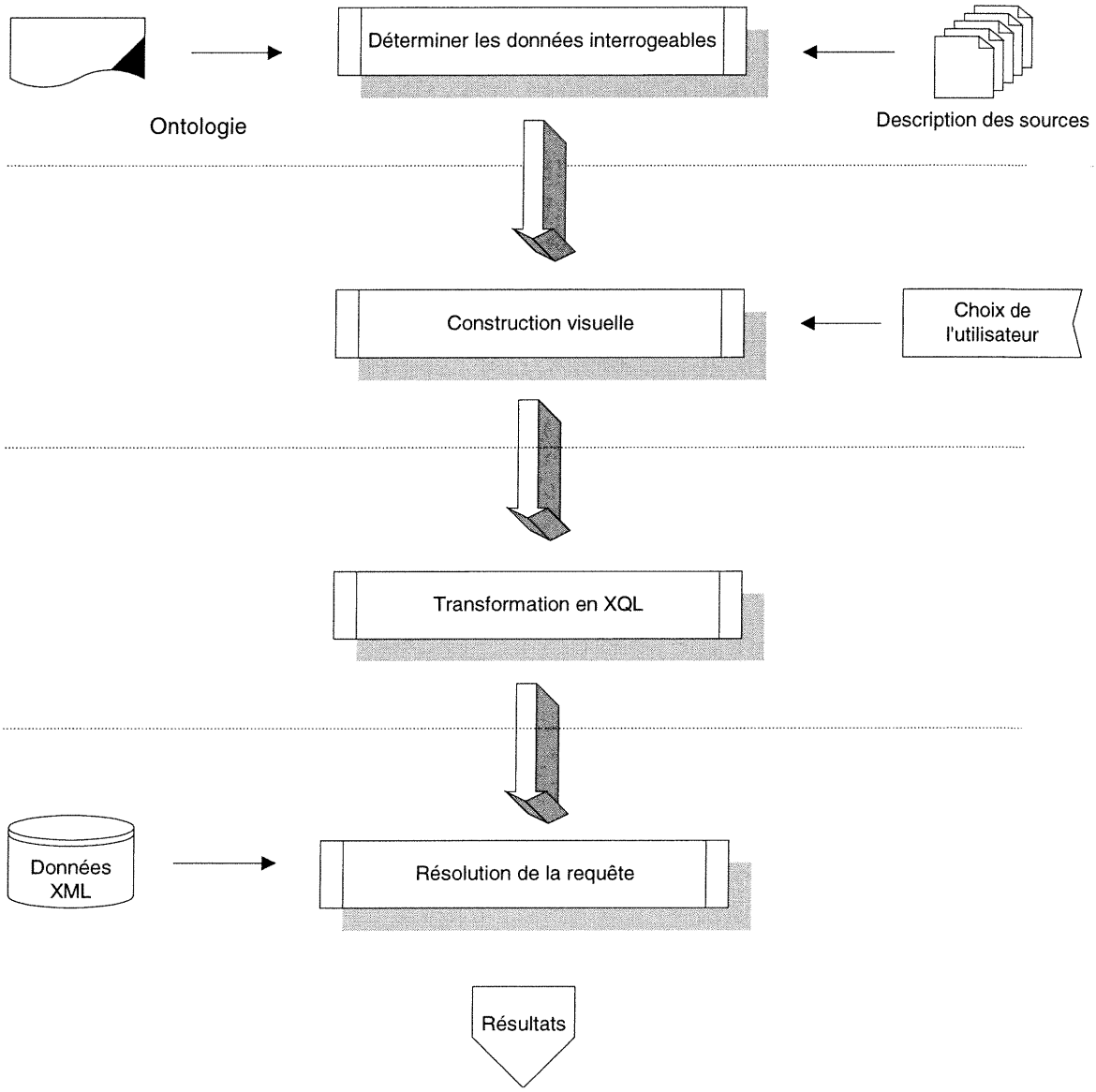


Figure 6-2 : schéma des requêtes

La transformation des spécifications de l'utilisateur en une requête XQL se fait en deux étapes. La première concernant le regroupement des conditions pour former une seule expression qui les exprime et ensuite le regroupement des éléments choisis en une expression les réunissant. Quoiqu'il est possible de choisir le regroupement des conditions par un "and" ou par un "or", dans notre prototype, nous utilisons un "and" implicite. La deuxième étape produit la requête XQL à partir de ces deux expressions (le regroupement des conditions et le regroupement des éléments/attributs à récupérer). Le schéma suivant illustre cette opération.

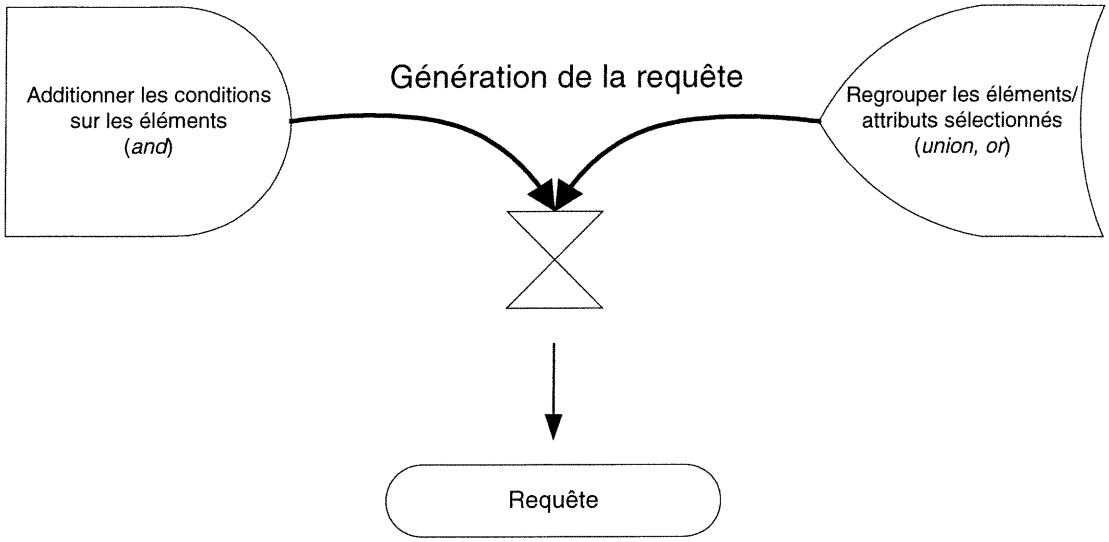


Figure 6-3 : génération de la requête

CHAPITRE 7

Utilisation des données extraites : Exemples d'application

Nous présentons dans cette ce chapitre deux exemples d'application de notre méthode d'extraction et d'intégration des données. Il s'agit d'un exemple qui fait intervenir un Système Multi-Agents (SMA), le deuxième exemple concerne l'exploitation des données extraites à travers une interface WEB. Comme nous l'avons évoqué dans l'introduction, les tâches que les agents doivent remplir exigent de leur fournir un minimum de données. La présentation du premier exemple vise à montrer une façon possible d'intégrer les données extraites dans une application (ici un SMA). Les agents de cet exemple utilisent notre procédure d'extraction pour récupérer et obtenir les données nécessaires. Se basant sur celles-ci, les agents rendent des services tels que demandés par l'utilisateur.
L'exemple vise à montrer une application simple et utile de notre concept, mais il met aussi en valeur les fonctionnalités que peut avoir un SMA de manière générale.

Nous utilisons des agents GUEST développés au sein du CRIM [Magnin et Alikacem, 99]. La particularité de ces agents est qu'ils sont génériques : ne dépendant pas d'une plate-forme particulière, ils peuvent être accueillis sur différentes plates-formes hétérogènes.

7.1 Exemple utilisant les agents GUEST

7.1.1 Description des agents GUEST

Il existe actuellement plusieurs plates-formes et environnement pour agents, chacune offrant les mécanismes nécessaires pour la création et l'exécution des agents. Une fois créé sur une plate-forme particulière, l'exécution de l'agent reste lié à celle-ci. Il ne peut migrer que sur des plates-formes du même type. Par exemple, un agent "Aglets" [URL-7-1] ne peut fonctionner que sur un serveur basé sur la plate-forme Aglets, et non pas sur un serveur "Grasshopper" [URL-7-2]. Et inversement. Tant que l'on envisage des applications multiagents propriétaires et fermées, cette limitation n'est pas trop contraignante. Cependant, dans un environnement hétérogène et ouvert, cette limitation devient rédhibitoire [Magnin, 99].

Afin de contourner ce problème, GUEST offre la possibilité d'exécuter des agents sur des plates-formes hétérogènes. Pour cela, GUEST fournit une couche supplémentaire à celles-ci pour qu'elles soient en mesure d'accueillir des agents qui ne leurs étaient pas destinés initialement.

Le principale rôle de la couche GUEST est de permettre à des agents (dits génériques) de s'exécuter indépendamment de la plate-forme sous-jacente (dite native). Dès lors, ces agents disposent d'un environnement et de fonctionnalités identiques et de manière transparente pour toutes les différentes plates-formes de base. Ainsi, GUEST permet d'avoir un environnement de développement d'applications multiagent indépendantes des spécificités des plates-formes existantes.

Les serveurs supportés actuellement par GUEST (pour lesquels une interface a été développée) sont Aglet d'IBM, Voyager d'ObjectSpace [URL-7-3] et Grasshopper d'IKV++. Tous ces environnements pour agents se basent sur les machines virtuelles de Java ; ce qui offre encore plus d'indépendance au niveau du matériel et des systèmes d'exploitation.

7.1.1.1 Architecture

Les plates-formes disposant d'environnements différents les unes des autres, leurs fonctionnalités de base ne sont pas toujours les mêmes. La couche GUEST, qui constitue l'interface avec les plates-formes natives, permet d'uniformiser l'utilisation et l'interaction avec celles-ci. Pour les services existants (tels que la migration, la communication, la création ...), GUEST offre un ensemble de services et une API uniforme pour toutes les plates-formes. Il est possible que

certaines fonctions ne soient pas disponible dans certaines plates-fomes. Dans ce cas, GUEST implante ces fonctions manquantes pour que de tels services soient partout disponibles.

La figure 7-1 illustre le positionnement de l'interface GUEST entre les agents et les plates-formes.

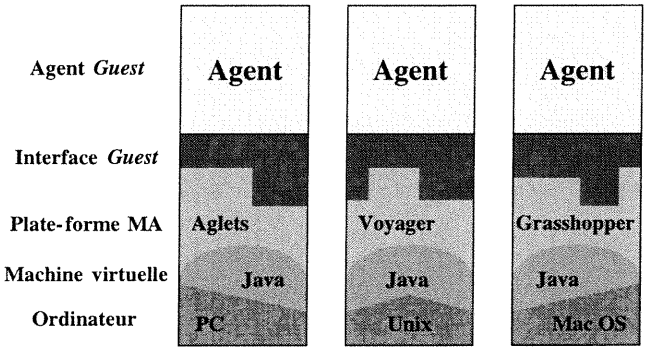


Figure 7-1 : l'interface de GUEST

7.1.1.2 Manipulation des agents GUEST

L'utilisateur dispose de deux API (Application Programming Interface) pour manipuler et interagir avec les agents :

« **Développeur API** » permet aux développeurs d'étendre GUEST, en ajoutant de nouvelles plates-formes natives ainsi que de nouvelles fonctionnalités.

« **Utilisateur API** » permet aux développeurs de créer des agents et de les manipuler. Ces agents GUEST peuvent migrer d'un serveur à un autre et communiquer entre eux de façon transparente vis-à-vis des types des plates-formes qui abritent ces agents. Cette API travaille seulement avec la partie de GUEST indépendante des plates-formes natives.

7.1.2 Exemple d'extraction des données

Ci-après, nous décrivons un exemple qui fait intervenir des agents GUEST spécialisés dans le domaine financier, particulièrement pour l'observation des cotations en bourse. Cet exemple a été réalisé en grande partie par S. YUN et N. BESSON, étudiants au CRIM. Notre contribution se situe au niveau du module pour la récupération des données des pages WEB.

Etant donné que les fluctuations de la bourse sont fréquentes dans une même journée et pour faire un maximum de profit, il faut acheter au plus bas et revendre au plus haut prix, un courtier doit constamment consulter les variations d'une valeur. Mais il est difficile pour une personne de

surveiller plusieurs valeurs tout au long de la journée. S'il se concentre sur une valeur, il prend énormément de risque. C'est là que le rôle des agents devient intéressant. Dans l'exemple qui suit, nous allons faire appel aux agents GUEST pour observer le marché financier. En particulier, ils suivront le cours des indices et informeront l'utilisateur de l'évolution des opérations en laissant à ce dernier plus de temps pour se consacrer aux tâches telles que : *établissement d'une stratégie* et *prise de décisions*.

7.1.2.1 Description sommaire

De manière pratique, un utilisateur possède un certain nombre de titres d'actions appartenant à plusieurs compagnies. L'ensemble de ces titres constituent le portefeuille boursier de l'utilisateur. Notre exemple se base sur la surveillance d'un portefeuille par l'observation des valeurs des actions qui le constituent.

L'exemple fait intervenir trois types d'agents différents *Supervisor*, *Collector* et *Associated* qui ont chacun une mission distincte. L'agent *Collector* a la fonction de gérer un portefeuille, il a sous sa responsabilité plusieurs autres agents : *Associated.* Chacun de ces derniers a la tâche d'observer un titre d'action du portefeuille. L'agent *Supervisor* permet de contrôler plusieurs *Collectors*. Voici un schéma (figure 7-2) pour illustrer les relations entre ces agents :

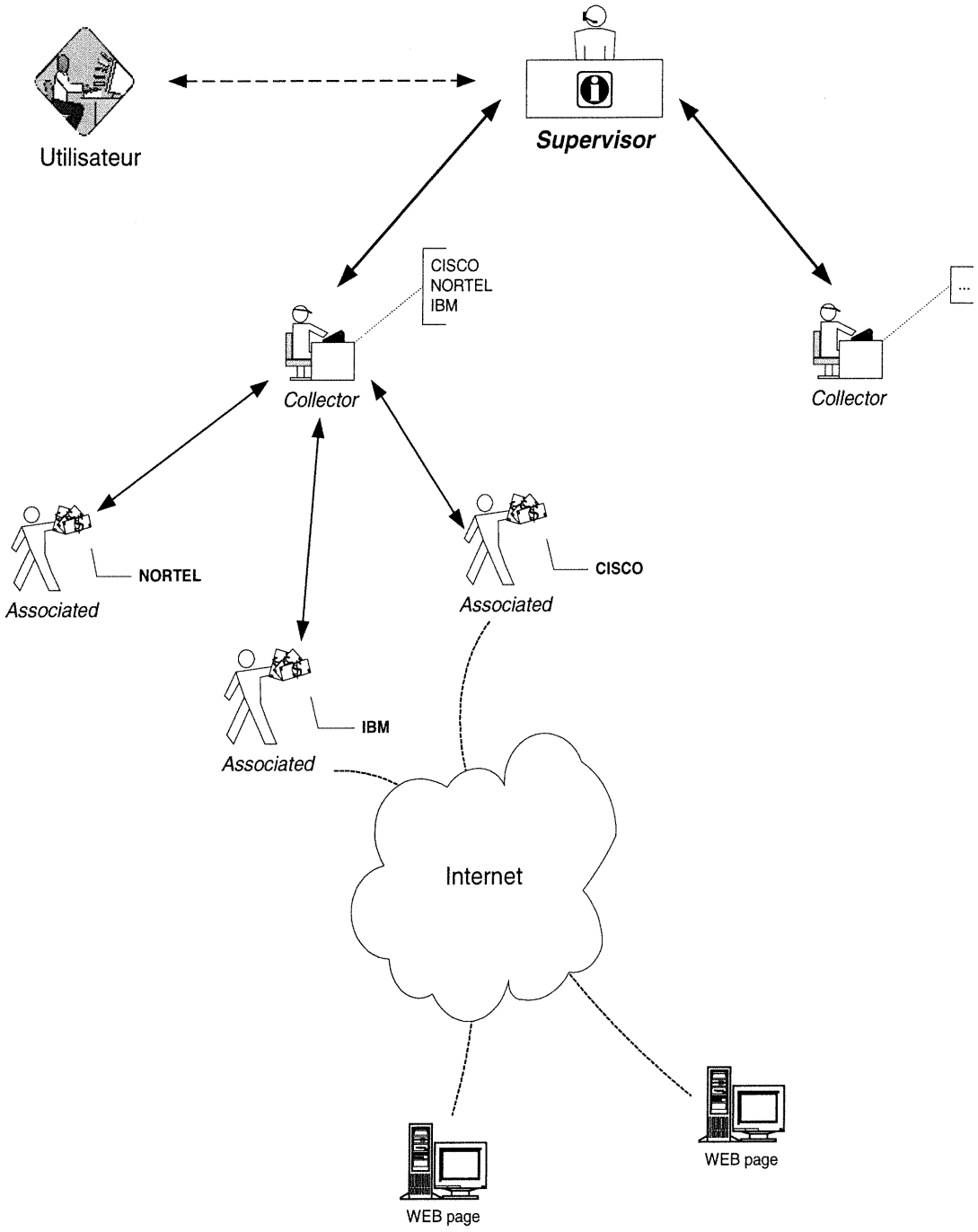


Figure 7-2 : exemple utilisant les agents GUEST

7.1.2.2 Description détaillée et le rôle de chaque agent

- Agent *Supervisor* : il permet de créer un agent *Collector* en lui soumettant les informations sur un portefeuille à gérer. Il joue également le rôle d'interface entre l'utilisateur et l'agent *Collector*. Il transmet au *Collector* les directives de l'utilisateur pour exécuter des opérations particulières. De même, il renvoie les résultats du *Collector* à l'utilisateur ;
- Agent *Collector* : il se charge de l'observation d'un portefeuille composé de plusieurs titres d'actions. Pour chacune de ces dernières, il crée un agent *Associated* spécialisé qui surveillera son cours. L'agent *Collector* émet des avis suite à la variation de la valeur du portefeuille[5]. Le *Collector* contacte régulièrement ses agents *Associated* pour avoir les nouvelles valeurs. Il recalcule sur la base de ces dernières la nouvelle valeur de son portefeuille. Les services rendus actuellement par le *Collector* consistent à émettre des avis sur la variation en gain ou en perte du portefeuille ;
- Agent *Associated* : il a la charge d'observer le cours d'une action donnée. Pour cela, il utilise notre méthode d'extraction de données pour récupérer la valeur en temps réel à partir des pages WEB. Il se base sur le fichier de description d'une source donnée pour extraire la valeur du titre d'action dont il a la charge.

Les agents *Associated*, *Collector* et *Supervisor* sont des agents GUEST. Il peuvent être crées sur n'importe quelle plate-forme d'agents qui possède une interface GUEST. Ils migrent entre ces plates-formes et communiquent de manière transparente.

Nous avons constitué un portefeuille comportant des actions de cinq compagnies : Nortel Networks, IBM, CISCO, YAHOO et Nokia. La valeur du portefeuille est la somme des valeurs de ces actions multipliées par le nombre des actions.

$$P = \sum n_i v_i$$

- n_i est le nombre d'actions de la compagnie i dans le portefeuille ;
- v_i est la valeur de l'action i sur le marché boursier.

Nous avons choisi deux sources YahooFinance (URL-3-1) et Nasdaq (URL-7-5) pour récupérer les valeurs v_i. Chaque agent *Associated* s'occupe d'extraire une ces valeurs en utilisant la description de la source correspondante. De manière régulière, l'agent *Collector* met à jour la valeur de P en se basant sur les nouvelles valeurs communiquées par les agents *Associated*. Il informe l'utilisateur en temps réel des variations observées dans son portefeuille.

[5] Nous nous contentons pour l'instant d'observer les variations de la valeur d'un portefeuille sachant qu'un traitement plus élaboré pour la gestion des portefeuilles est en cours de réalisation par d'autres personnes au sein du groupe multi-agents du CRIM.

7.2 Deuxième exemple d'application sur Internet

Nous avons également écrit un module, indépendant de notre application prototype, spécialisé uniquement dans l'extraction des données à partir des descriptions des sources. Dans le but de faciliter son utilisation par des intéressés, nous l'avons intégré à un serveur WEB.
De manière pratique, les utilisateurs soumettent une description de source à travers le site WEB et reçoivent en retour les données souhaitées.
Sur le site, les utilisateurs peuvent tester quelques exemples de descriptions qui sont déjà prêts [URL-7-4]. Le résultat de l'extraction est un document XML qui contient les données récupérées de la page WEB désignée. Toutefois, l'utilisateur a la possibilité de fournir une autre description pour extraire les données. L'utilisateur spécifie l'emplacement du fichier en utilisant un chemin en forme d'adresse URI.

L'implantation de cet exemple est faite en utilisant le serveur HTTP *Apache* et le module JServ pour supporter l'utilisation des *servlets.* La figure 7-3 montre le schéma pour cet exemple :

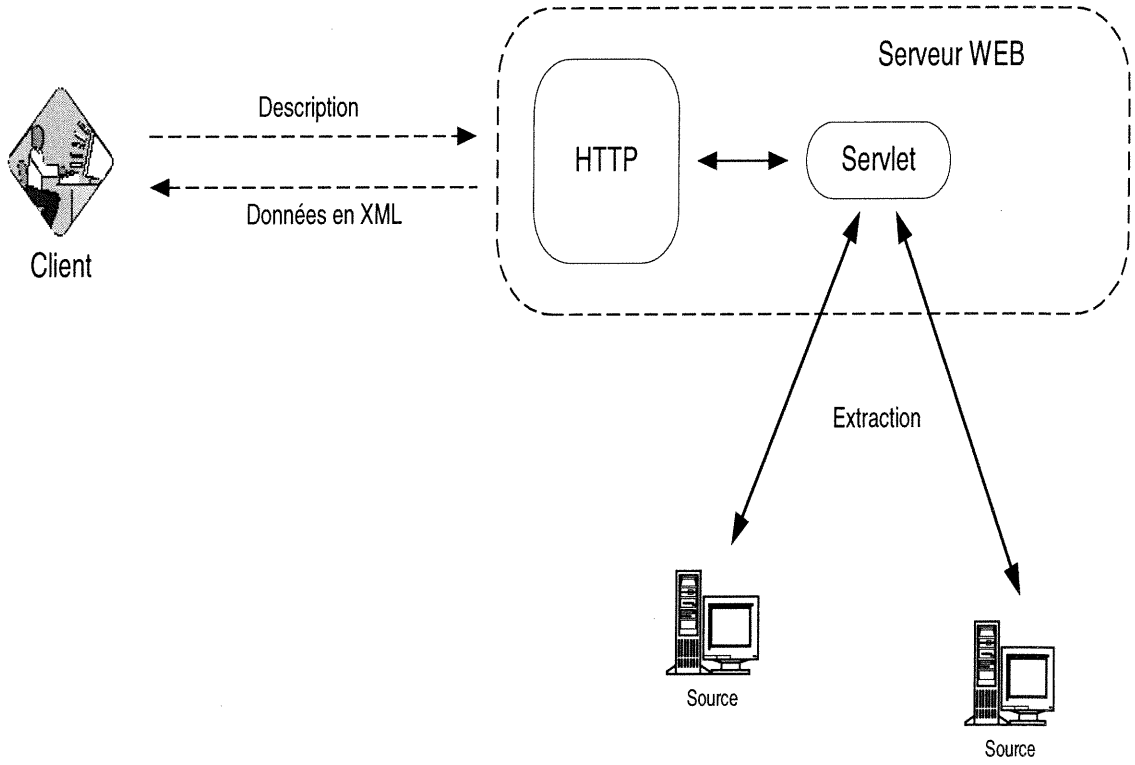


Figure 7-3 : utilisation des servlets comme interface pour l'extraction

7.3 Conclusions

Le premier exemple nous a permis non seulement d'appliquer notre approche d'extraction mais aussi d'intégrer les données à partir de différentes sources hétérogènes sur le WEB. Ce qui offre une vue uniforme sur les données appartenant à un même domaine. Nous avons effectué un traitement (même simplifié) en temps réel après extraction afin de montrer l'utilité et l'apport possible de notre approche. Actuellement, d'autres personnes de l'équipe GLIC du CRIM sont en train de poursuivre dans cette direction pour une gestion plus raffinée des valeurs boursières.
Nous avons pu montrer, à travers le premier exemple, l'apport de notre outil et méthodes dans un domaine pratique d'application.

Le deuxième exemple visait à porter l'utilisation des descriptions des sources et de l'extraction de l'information à travers une interface WEB. Ce qui offre au public la possibilité d'extraire les données d'une source en fournissant la description de la source sans pour autant posséder l'ensemble de l'outil. D'autres part, nous considérons que cela peut aider les gens d'une communauté d'un domaine particulier à publier leurs données (en plus de leurs sites WEB traditionnels) selon un format qui respecte une ontologie partagée par l'ensemble. Ceci a pour effet de favoriser la communication et les échanges entre eux.

L'exemple du §7.2 rend comme résultat d'extraction des documents XML. En utilisant le langage de requêtes qui lui convient (§6.2.1), l'utilisateur pourra interroger ces documents pour avoir les données qu'il désire.

Conclusion

Nous avons traité dans ce mémoire le problème de l'extraction des données à partir de pages WEB et de leur intégration. Après avoir passé en revue un certain nombre de travaux de recherche dans le domaine, nous avons tiré des conclusions à partir de ces expériences et proposé notre démarche pour extraire et intégrer les données des pages WEB.

Particulièrement, il s'agissait de trouver un moyen pour récupérer les données puis de les mettre en commun par la suite. Concernant l'extraction des données, nous avons utilisé une méthode consistant à appliquer des requêtes XQL sur la page HTML après l'avoir convertie en un document XML bien formé. Grâce à l'utilisation des fichiers de description, l'utilisateur récupère les données telles qu'il les perçoit sur les pages WEB. Ce qui offre à l'utilisateur des données valides qui pourront alimenter directement des bases de données ou un système multi-agents. Ceci dit, et en comparaison avec les systèmes classiques de recherche à base de mots clés, ces systèmes nécessitent l'intervention ultérieure de l'utilisateur pour vérifier les résultats trouvés.

L'intégration de données extraites de plusieurs sources se base sur une ontologie qui constitue un modèle commun entre ces sources. Les données extraites obéissent à une même ontologie (d'un domaine particulier), ce qui facilite leur mise en commun. Dans ce sens, nous avons proposé une modélisation pour représenter l'ontologie.

Sur le plan des réalisations, nous avons développé un outil qui assiste l'utilisateur pour la construction des descriptions de sources. Cette étape se fait manuellement, mais elle est

grandement facilitée par les fonctionnalités offertes par l'outil. Le grand intérêt d'une telle description est qu'elle est générée une seule fois pour un site et reste utilisable tant que ce dernier ne change pas le formatage de ses informations. L'étape de récupération utilisant cette description se fait alors automatiquement sans intervention de l'utilisateur. Ce dernier pourra non seulement récupérer les données d'une page WEB mais également les mettre à jour en utilisant la même description. Ce processus se fait naturellement de manière automatique.

Ceci dit, une page WEB a un caractère dynamique, elle pourrait subir des modifications, il est alors légitime d'évaluer la durée de validité de sa description. En d'autres termes, quelles sont les possibilités qui font que la description d'une page WEB ne fournisse pas les données escomptées suite à des changements opérés sur celle-ci. Dans cette optique, il est nécessaire de noter que :

- La plupart des pages WEB qui fournissent des données dynamiques (pour lesquelles l'utilisateur a construit une description) ne changent pas d'emplacement sur la page. C'est-à-dire que la hiérarchie (ou l'arborescence XML) de la page ne change pas même si les données sont mises à jour. Notre méthode de construction de pages WEB se basant sur cette structure des pages pour extraire les données, cette description n'est donc pas affectée dans ce cas.

- Les pages WEB qui présentent des données mises à jour régulièrement se trouvent en général en avant des bases de données. La génération des pages WEB en utilisant les informations de ces bases de données ce fait alors automatiquement ; ce qui fait que ces pages conservent une structure uniforme de présentation.

- D'autre part, si des changements sont effectués sur une page WEB, ils ne sont pas effectués généralement de manière à modifier profondément cette page. Ceci est dû au fait que les utilisateurs sont en général familiarisés avec la présentation de la page. Les responsables des sites WEB essaient de fidéliser les utilisateurs et font en sorte que ces derniers gardent le même comportement et retrouvent facilement ce qui les intéresse parmi les informations sur la page. Si des modifications sont à effectuer, elles concernent principalement l'aspect visuel (couleurs, images, ...) et ne concernent pas l'emplacement des données (sauf modifications majeures qui ne surviennent pas souvent).

- Enfin, nous avons utilisé des descriptions pour extraire les données à partir de plusieurs pages WEB. D'après notre expérience et pendant la durée (cinq mois) où nous avons utilisé

plusieurs exemples de descriptions, nous n'avions pas eu à les modifier. Elles sont restées valides pour la récupération des données souhaitées. Ceci montre que ces sites n'effectuent pas souvent des changements profonds de structure.

Nous considérons que passé cette durée de temps dans la vie d'une description, si un changement est nécessaire à effectuer pour s'adapter aux modifications de la structure de la page, il sera acceptable. De plus, et afin de minimiser les incidences de tels changements, notre outil pourrait être fourni à chaque site qui désire être intégré dans une communauté. Le concepteur du site pourra alors modifier le processus d'extraction pour répondre à un changement.

Cette étude montre qu'il est possible d'intégrer des données provenant de différentes sources. Cependant, une phase de préparation manuelle est nécessaire. Dans l'état actuel du WEB, il est difficile d'envisager la suppression de cette phase. Cela serait possible quand les concepteurs de sites suivront certains standards du domaine, lesquels domaines devront posséder alors une ontologie bien établie. Même avec cette phase de récupération manuelle, le processus d'extraction est souhaitable car les données extraites peuvent être facilement intégrées dans d'autres systèmes tels qu'un système d'agents pour le commerce électronique. Le point qui reste le plus critique est la définition d'une ontologie. Mais on ne peut pas envisager un système ouvert de commerce électronique sans faire appel à une norme du domaine. Même si on n'établit pas une ontologie complète, un standard jouant son rôle sera toujours nécessaire.

Nous avons utilisé un outil externe pour la construction et la conception de l'ontologie, ce dernier ne prenant pas en charge une gestion distribuée des ontologies impliquant plusieurs utilisateurs. Dans ce contexte, l'édition et la maintenance réparties d'une ontologie en utilisant une interface WEB serait appréciable. Cette option constitue une amélioration à prendre en compte dans les prochaine versions de l'outil.

L'outil, tel que nous l'avons réalisé, admet des améliorations quant au niveau de l'aide fournie à l'utilisateur pour la construction des descriptions. Jusqu'à présent, il revenait à l'utilisateur d'écrire les requêtes XQL pour extraire les données. Même si XQL est simple et facile à apprendre et à utiliser, il serait intéressant pour l'utilisateur de pouvoir générer automatiquement la requête. Ceci est envisageable à partir des données qu'il aurait sélectionné sur la page. Nous avions déjà fait du travail dans ce sens en proposant de choisir les données à partir d'une représentation en arbre de la page. Cependant, la requête générée par cette méthode reste non optimisée et l'utilisateur doit parcourir et sélectionner toutes les données qu'il désire.

Ce que nous proposons comme amélioration, est que l'outil soit capable de déduire la requête pour toutes les données qui ont le même patron que celle qui a été sélectionnée par l'utilisateur sur la page WEB. Il serait alors nécessaire d'extraire des règles de génération de requêtes. Dans notre cas, puisque les données ont un format bien défini, cette génération serait possible.

D'autre part, l'architecture générale que nous avons proposée, qui comprend la définition d'une ontologie, la démarche pour l'extraction et les requêtes, reste applicable à d'autres sources de données (eg. des bases de données). Ceci est possible en adaptant l'étape de l'extraction aux types de sources.

Le module d'extraction des données que nous avons réalisé est le seul composant qui interface les sources de sonnées (ici des pages WEB) et qui agit en conséquence. Si l'on veut prendre en compte d'autres types de sources de données telles que des bases de données, il suffira de récupérer les données de ces bases (chose que l'on sait faire par des requêtes SQL par exemple) et de les lier avec les éléments de l'ontologie. Par contre, l'extraction des données à partir des sources non structurées reste difficile à réaliser. Ces sources nécessitent un traitement de la langue naturelle ce qui diminue l'exactitude et la précision des informations récupérées.

Le module d'extraction est le seul composant de l'architecture à modifier pour prendre en considération les nouveaux types de sources, les autres composants restent inchangés. Ainsi, on pourra intégrer différentes sources aussi bien que différents types de sources de données.

Annexe 1 : Traitement des requêtes dans le système Infomaster

Afin d'illustrer le modèle de transformations entre les *Interface Relations*, les *Base Relations* et les *Site Relations,* introduites au §2.3.2, nous donnons ci-après un exemple.

Nous supposerons qu'un utilisateur cherche des voitures BMW construites en 1996 et dont le prix de vente est inférieur à la moyenne des prix sur le marché. Cet exemple se base sur les informations données en exemple dans §2.3.2.

q(Model, Mileage, Price) ≡
cars(bmw, Model, 1996, Mileage, Price, Value)
& Price < Value

Le processus de résolution d'une requête se fait en trois étapes [Duschka *et al*, 97b] : R*eduction, Abduction, Optimization.*

Reduction : étant donnée une requête q exprimée en des *Interface Relations*, cette étape consiste en la réécriture de q en fonction des *Bases Relations*. Ceci est simple puisque les *Interface Relations* qui servent à exprimer q sont remplacées par leurs définitions en fonction des *Base Relations*.

q(Model, Mileage, Price) ≡
classifieds(bmw, Model, 1996, Mileage, Price)
& bluebook(bmw, Model, 1996, Mileage, Value)
& Price < Value

Abduction : Cette phase du traitement est la plus délicate puisque les *Site Relations* sont exprimées dans le système Infomaster, par des *Base Relations* et non pas l'inverse. La requête q est donc réexprimée de la manière suivante :

q(Model, Mileage, Price) ≡
sfc(bmw, Model, 1996, Mileage, Price)
& bmw(Model,1996, Mileage_in_km, Value_in_DM)
& exchange(dm, dollar, Rate)
*& Mileage = Mileage_in_km * 1.6*
*& Value = Value_in_DM *Rate*
& Price < Value

q(Model, Mileage, Price) ≡
sjmn(bmw, Model, 1996, Mileage, Price)
& bmw(Model,1996, Mileage_in_km, Value_in_DM)
& exchange(dm, dollar, Rate)
*& Mileage = Mileage_in_km * 1.6*
*& Value = Value_in_DM *Rate*
& Price < Value

Optimization : Cette étape concerne essentiellement l'élimination de la redondance d'accès aux sources. Si on sait par exemple que le journal de San Francisco Chronicle publie toujours toutes les annonces du San Jose Mercury News, alors l'accès à ce dernier s'avère redondant et inutile.

La description des interfaces utilisateurs et des sources en terme de Base Relations rend le système plus flexible, en ce sens qu'il permet d'ajouter de nouvelles sources et de modifier l'interface utilisateur sans pour autant changer le schéma interne de représentation. Toutefois, cela rend la tâche difficile quand il s'agit de changer ce dernier, car les correspondances entre relations sont à revoir.

Bibliographie

[Alin et al, 98] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, Dan Suciu "XML-QL", QL'98 - The Query Languages Workshop, Boston, December, 1998

[Arens *et al,* 93] Y. Arens, C. Y. Chee, C. N. Hsu, and C. A. Knoblock, Retrieving and Integrating Data from Multiple Information Sources, International Journal of Intelligent and Cooperative Information Systems. Vol. 2, No. 2, 1993.

[Arens *et al,* 96] Y. Arens, C. A. Knoblock and C. N. Hsu, Query Processing in the SIMS Information Mediator, Advanced Planning Technology, editor, Austin Tate, AAAI Press, Menlo Park, 1996.

[Atzeni et al, 97] P. Atzeni, G. Mecca, P. Merialdo To Weave the Web - In Proceedings of the 23rd International Conference on Very Large Databases (VLDB'97), 1997

[Bayardo *et al*, 96] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, D. Woelk, Semantic Integration of Information in Open and Dynamic Environments, MCC Technical Report MCC-INSL-088-96, October, 1996.

[Bezivin, 97] J. Bezivin, LES NOUVELLES CONVERGENCES : OBJETS, COMPOSANTS, MODÈLES ET ONTOLOGIES, JICAA'97, Roscoff France, Mai 1997.

[Bonifati and Ceri, 00] Angela Bonifati, Stefano Ceri: Comparative Analysis of Five XML Query Languages. SIGMOD Record 29(1): 68-79 (2000)

[Boughettaya *et al*, 99] A. Boughettaya, B. Benatallah, M. Ouzzani and L. Hendra, WEBFINDIT : An Architecture and System for Querying Web Databases, IEEE Internet Computing, July-August 1999.

[Chawathe *et a*, 94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources, Proceedings of IPSJ Conference, Tokyo, October 1994.

[Cohen *et al*, 94] P. R. Cohen, A. J. Cheyer, M. Wang, and S. C. Baeg, An Open Agent Architecture, in AAAI Spring Symposium, March 1994.

[Duschka *et al*, 97a] O. M. Duschka and M. R. Genesereth, Query Planning in Infomaster, Proceedings of the ACM Symposium on Applied Computing, SAC '97, San Jose, February 1997.

[Duschka *et al*, 97b] O. M. Duschka and M. R. Genesereth, Infomaster - An Information Integration Tool, Proceedings of the International Workshop "Intelligent Information Integration", German Annual Conference on Artificial Intelligence, Freiburg, Germany, September 1997.

[Farquhar *et al*, 95] A. Farquhar, R. Fikes, W. Pratt, & J. Rice. Collaborative Ontology Construction for Information Integration. Knowledge Systems Laboratory, Department of Computer Science, Technical Report KSL-95-63, August 1995.

[Finin et al, 97] Tim Finin, Yannis Labrou, and James Mayfield, KQML as an agent communication language, "Software Agents", in Jeff Bradshaw (Ed.), MIT Press, Cambridge, 1997.

[Genesereth *et al*, 97] M. R. Genesereth, A. M. Keller, and O. M. Duschka, Infomaster: An Information Integration System, Proceedings of 1997 ACM SIGMOD Conference, May 1997.

[Goldman *et al*, 99] R. Goldman, J. McHugh, and J. Widom. From Semistructured Data to XML: Migrating the Lore Data Model and Query Language. Proceedings of the 2nd International Workshop on the Web and Databases (WebDB '99), Philadelphia, Pennsylvania, June 1999.

[Gruber, 93] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing, The International Workshop on Formal Ontology, March 1993.

[Guarino, 96] N. Guarino, Understanding, Building, and Using Onologies, International Journal of Human and Computer Studies, 1996.

[Hammer *et al*, 97] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting Semistructured Information from the Web". In Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, May 1997.

[Hiroshi *et al*, 98] Hiroshi Ishikawa, Kazumi Kubota, Yasuhiko Kanemasa, XQL: A Query Language for XML Data, QL'98 - The Query Languages Workshop, Boston, December, 1998

[Huck et al, 98] Gerald Huck, Peter Fankhauser, Karl Aberer, Erich J. Neuhold: JEDI: Extracting and Synthesizing Information from the Web, Conference on Cooperative Information Systems CoopIS'98, New York, August, 1998, IEEE Computer Society Press.

[Keller, 97] A. M. Keller, Smart Catalogs and Virtual Catalogs, Readings in Electronic Commerce, Editors : Ravi Kalakota and Andrew Whinston, Addison-Wesley, 1997.

[Knoblock *et al*, 98] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. G. Philpot, and S. Tejada, Modeling Web Sources for Information Integration. Proceedings of the National Conference on Artificial Intelligence, Madison, 1998.

[Lange et Oshima, 99] D. B. Lange and M. Oshima, Seven Good Reasons for Mobile Agents, Communications of The ACM, vol. 42 No. 3, March 1999.

[Li *et al,* 98] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. Ullman and M. Valiveti. Capability Based Mediation in TSIMMIS". SIGMOD 98 Demo, Seattle, June 1998.

[Magnin et Alikacem, 99] Laurent Magnin et El Hachemi Alikacem, GenA : Multiplatform Generic Agents, MaTa'99 First International Workshop on Mobile Agents for Telecommunication Applications, Ottawa, October 1999.

[Magnin, 99] Laurent Magnin, "Internet, environnement complexe pour agents situés", conference on the Intelligence Artificielle Située (IAS), Paris, October 25-26, 1999

[Martin *et al,* 97] D. L. Martin, H. Oohama, D. Moran, and A. Cheyer, Information Brokering in an Agent Architecture, Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, April 1997.

[Martin *et al,* 99] D. L. Martin, A. J. Cheyer, and D. B. Moran, The open agent architecture: A Framework for Building Distributed Software Systems, Applied Artificial Intelligence, vol. 13, January-March 1999.

[Miller and Sheth, 00] John A. Miller and Sonali Sheth, "Querying XML Documents," IEEE Potentials (IEEE-STM), Vol. 19, No. 1 (February/March 2000) pp. 24-26. IEEE Press

[Nodine *et al,* 98a] M. Nodine, J. Fowler and B. Perry, An Overview of Active Information Gathering in InfoSleuth, Technical Report, October 1998, http://www.mcc.com/projects/infosleuth/publications/TR98/INSL-114-98.ps

[Nodine *et al,* 98b] M. Nodine, B. Perry and A. Unruh, Experience with the InfoSleuth Agent Architecture, Proceedings of AAAI-98 Workshop on Software Tools for Developing Agents, 1998.

[Ouahid and Karmouch, 99a] H. OUAHID and A. KARMOUCH, Converting WEB Pages into Well-formed XML Documents, Proceeding of IEEE International Conference on Communications, Vancouver, June 1999.

[Ouahid and Karmouch, 99b] H. OUAHID and A. KARMOUCH, A WEB Ontology Description Language, GlobeCom'99, Rio de Janeiro, July 1999.

[Ouahid and Karmouch, 99c] H. OUAHID and A. KARMOUCH, WONDEL : An Efficient Way to Add Semantics to the WEB Pages, Metastractures'99, Montreal, August 1999.

[Ouahid and Karmouch, 99d] H. OUAHID and A. KARMOUCH, An XML-Based WEB Mining Agent, Proceeding of MATA'99, Ahmed KARMOUCH and Roger IMPEY edts., World Scientific, Ottawa, October 1999.

[Papakonstantinou *et a*, 95] Y. Papakonstantinou, H. Garcia-Molina and J. Widom, Object Exchange Across Heterogeneous Information Sources, IEEE International Conference on Data Engineering, Taipei, March 1995.

[Sahuguet and Azavant, 99 a] Arnaud Sahuguet and Fabien Azavant, Looking at the Web through XML glasses, Conference on Cooperative Information Systems CoopIS'99, Edinburgh Scotland, September 2-4 1999.

[Sahuguet and Azavant, 99 b] Arnaud Sahuguet and Fabien Azavant, Building light-weight wrappers for legacy Web data-sources using W4F, International Conference on Very Large Databases (VLDB), Edinburgh - Scotland - UK September 7 - 10 1999.

[Sahuguet and Azavant, 00] Arnaud Sahuguet and Fabien Azavant, Building Intelligent Web Applications Using Lightweight Wrappers, Data and Knowledge Engineering (to appear) (2000)

[Schulze-Kremer, 97] S. Schulze-Kremer, Integrating and Exploiting Large-Scale, Heterogeneous and Autonomous Databases with an Ontology for Molecular Biology, In Molecular Bioinformatics, Sequence Analysis - The Human Genome Project, Eds R. Hofestaedt and H. Lim eds, Shaker Verlag, Aachen, 1997, pp. 43-56

[Schulze-Kremer, 98] S. Schulze-Kremer, Ontologies for Molecular Biology, Proceedings of the Third Pacific Symposium on Biocomputing, Hawaii, World Scientific Publishers, Singapor, 1998.

[Staab *et al*, 00] S. Staab, M. Erdmann and A. Maedche, An extensible approach for Modeling Ontologies in RDF(S), Submitted to the 12th International Workshop on Knowledge Engineering and Knowledge Management (EKAW'2000), Juan-les-Pins, French Riviera, October 2-6, 2000

[Woelk and Tomlinson, 95] D. Woelk and C. Tomlinson, "Carnot and InfoSleuth: Database Technology and the World Wide Web", ACM SIGMOD International Conference on the Management of Data, May 1995.

URL-2-1 A. Luniewski, P. Scwars, J. Thomas and M. Tresch, RUFUS : Managing Semi-Structured Information, IBM-Almaden Research Center, http://www.almaden.ibm.com/cs/showtell/rufus/overview.html

URL-2-2 http://www.sri.com

URL-2-3	http://www.python.org
URL-2-4	http://www.mcc.com/projects/infosleuth/
URL-2-6	http://www.part.net
URL-3-1	http://finance.yahoo.com/q?s=AOL+YHOO+IBM+CSCO+LU+EBAY+TXN+EGRP+NOK&d=v1
URL-3-2	http://www.x-rates.com/tables/USD.html
URL-4-1	http://www.cs.umbc.edu/kse/kif/
URL-4-10	http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/what-is-an-ontology.html
URL-4-2	http://www.ksl.Stanford.EDU/software/ontolingua/
URL-4-3	Http://www.engr.sc.edu/research/CIT/demos/java/joe/joeBeta.html
URL-4-4	Http://www.ei.sanken.osaka-u.ac.jp/oe/Oe_guest/oeapp.html
URL-4-5	Http://www.w3.org/TR/PR-rdf-schema/
URL-4-6	http://www.commerceone.com/xml/cbl/docs/
URL-4-7	http://www.commerceone.com/
URL-4-8	http://www.extensibility.com/
URL-4-9	http://igd.rz-berlin.mpg.de/~www/oe/mbo.html
URL-5-1	http://www.imdb.com
URL-5-2	http://www.w3.org/People/Raggett/tidy/
URL-5-3	http://www.w3.org
URL-5-4	http://www3.sympatico.ca/ac.quick/jtidy.html
URL-6-1	http://www-db.stanford.edu/lore
URL-6-2	http://xml.darmstadt.gmd.de/xql/
URL-6-3	http://www.texcel.no/whitepapers/xql-design.html
URL-7-1	"Aglets Workbench by IBM Tokyo RL", http://www.trl.ibm.co.jp/aglets/
URL-7-2	Grasshopper, "Grasshopper by IKV++", http://www.ikv.de/products/grasshopper/index.html

URL-7-3 Voyager, "Voyager by ObjectSpace", http://www.objectspace.com/voyager/

URL-7-4 http://troi.crim.ca:8008/

URL-7-5 http://quotes.nasdaq.com/quote.dll?page=nasdaq100