UNIVERSITÉ PIERRE ET MARIE CURIE – PARIS 6

Document de Synthèse

Présenté pour obtenir

L'HABILITATION A DIRIGER DES RECHERCHES

Mention Informatique

par

BERND AMANN

TITRE DE L'HABILITATION

Du partage centralisé de ressources Web à l'échange de documents intensionnels

Soutenu le 18 Novembre 2003 devant la Commission d'Examen

Composition du jury

Rapporteurs :	Anne Doucet, Université Pierre et Marie Curie, Paris 6
	Georg Gottlob, Technische Universität Wien, Autriche
	Mohand-Saïd Hacid, Université Claude Bernard Lyon 1
Examinateurs :	Michel Scholl, Conservatoire National des Arts et Métiers, Paris
	Eric Simon, INRIA Rocquencourt
	Victor Vianu, U.C. San Diego, USA

Habilitation préparée au sein de l'équipe Vertigo, Cedric-CNAM et l'équipe Verso/Gémo, INRIA

Résumé : Ce rapport est un résumé de mon activité de recherche depuis la fin de ma thèse de doctorat en Février 1994 jusqu'en Juillet 2003. La problématique générale de ma recherche pendant cette période est le *partage de données et de connaissances sur le Web*. Mon hypothèse de départ est que ce partage est possible grâce à l'intégration de trois technologies : (1) la technologie des *bases de données* qui a prouvé ses capacités pour modéliser et gérer des grands volumes de données structurées, (2) le langage *XML* comme solution au problème de la représentation et l'échange d'informations sur le Web, et (3) les *ontologies* comme moyen d'exprimer des connaissances partagées sur un domaine d'intérêt.

A partir de ces trois technologies complémentaires pour l'échange, la gestion et la description de ressources Web, j'ai décliné deux axes de recherche principaux. Premièrement je me suis intéressé à l'intégration de données afin d'étendre les possibilités de découverte et d'interrogation de ressources d'informations hétérogènes sur le Web. XML comme modèle de données semi-structurées joue un rôle primordial dans les solutions proposées. Le deuxième axe de mon activité s'adresse aux besoins de publication et de partage de connaissances sur le Web. Les problèmes étudiés concernent l'architecture et la modélisation d'entrepôts de métadonnées sémantiques, la définition d'ontologies spécialisées à un domaine ainsi que l'évaluation de requêtes "sémantiques" qui prennent en compte la présence de thésaurus pour la description de ressources.

Le rapport présente en 5 chapitres le contexte général de ma recherche avec un bilan succinct de mes activités en termes de publications scientifiques, contrats de recherche et domaines d'applications (chapitre 1), le problème de la gestion de données et de connaissances sur le Web (chapitre 2), les résultats obtenus concernant la spécification et l'implantation d'entrepôts de métadonnées (chapitre 3), le problème et les solutions pour l'intégration de données XML (chapitre 4) et une conclusion avec les perspectives de ma recherche actuelle (chapitre 5).

Mots-clés : XML, Web Sémantique, médiation de requêtes, entrepôts de métadonnées, construction d'ontologies

Abstract : This report describes my research activity since the end of my PhD thesis in February 1994. My research interests during this period concerned the *sharing of data and knowledge on the Web* and the chosen approach was based on the integration of three technologies : (1) *database technology* which has proven its capacities in modelling and managing huge volumes of structured data, (2) *XML* as a standard solution to the problem of representing and exchanging information on the Web, and (3) the usage of *ontologies* for the formal representation of shared knowledge concerning a specific domain.

Starting from these three technologies which represent complementary tools for the exchange, the management and the description of Web resources, I have declined two principal research axes. First, I was interested in the problem of data integration for extending the ways of discovering and querying heterogeneous information on the Web. XML as a semi-structured data model plays a fundamental role in the proposed solutions. The second axes of my activity addresses the issue of knowledge sharing on the Web. The studied problems concern the construction of semantic metadata repositories, the definition of specialised ontologies as well as the evaluation of "semantic" queries exploiting semantic relationships between thesauri terms used for the description of Web resources.

This report presents in 5 chapters the general context of my research with a succinct overview of my activities in terms of scientific publications, research contracts and application domains (chapter 1), the problem of managing data and knowledge on the Web (chapter 2), the obtained results concerning the specification and implementation of semantic metadata repositories (chapter 3), the problem and solutions for the integration of XML resources (chapter 4) and, finally, a general conclusion and the perspectives of my current research activity (chapter 5).

Keywords : XML, semantic Web, query mediation, metadata repositories, ontology construction

Remerciements

Je voudrais remercier sincèrement toutes les personnes avec lesquelles j'ai pu collaborer et qui m'ont soutenu pour accomplir le travail présenté dans ce rapport.

Je dis un grand *MERCI*

À Michel. *Merci de m'avoir toujours aidé avec ta grande expérience et ta confiance en moi.*

À Anne Doucet, Georg Gottlob et Mohand-Saïd Hacid *pour m'avoir fait l'honneur d'accepter le travail de rapporteur*

À Michel, Eric Simon et Victor Vianu *pour faire partie du jury.*

À mes amis de l'équipe Vertigo Dan, David, Valérie et Cédric *pour avoir partagé avec moi le bureau, les pauses café et d'autres moments vertigineux.*

À Philippe *sans qui je n'aurais pas eu le courage et le plaisir de me lancer dans l'écriture d'un livre sur XSLT.*

À Irini *pour m'avoir fait le plaisir d'être ma première étudiante en thèse et de la terminer avec succès.*

À Serge et Sophie *pour m'avoir accueilli si chaleureusement dans le projet Verso pendant toutes ces années.*

À Tova, Ioana, Luc, Marie-Christine, Chantal, Laurent, Benjamin, Omar et tous les autres Gémos et ex-Versos dont le nombre est devenu trop important pour pouvoir les nommer tous.

À Vassilis, mon ami grec avec qui j'ai pu partager mes premiers pas dans la recherche.

À mes collègues du laboratoire CEDRIC et du Département Informatique du CNAM qui m'ont aidé à mieux comprendre et accomplir le métier d'enseignant-chercheur.

À tous mes étudiants qui ont fait preuve de patience pendant mes cours finissant souvent très tard le soir.

À Alain Michard qui m'a invité dans l'équipe SAMIE et souvent donné des bons conseils.

À mes parents. Papa, leider habe ich mich nicht genug beeilt damit Du diese Zeilen lesen kannst, die Dich sicher sehr mit Stolz erfüllt hätten.

À Lydie, Caroline, Fabienne et Emmanuelle *pour avoir donné un autre sens à mon travail.*

Table des matières

Chapitre 1

Introduction

1.1 Le partage d'informations sur le Web

Le Web est un outil universel pour la publication d'informations sous forme de ressources accessibles de n'importe quel poste informatique connecté au réseau Internet. La simplicité de la mise en place d'un serveur Web et les coûts d'investissement très limités a rapidement attiré un grand nombre d'utilisateurs de cette technologie pour publier des informations à l'échelle mondiale. En même temps, le Web a été aussi une victime de son succès. La taille du Web a été estimée à 1 milliard de pages en Mai 2000 [1] et le nombre de noms de stations enregistrés sur Internet croit d'une manière exponentielle et se situe à environ 160 millions en Juillet 2002 [2] dont presque 1 million commencent par `www`.

Les outils pour gérer et exploiter cette information doivent continuellement s'adapter face à cette croissance. Un exemple qui illustre bien ce besoin d'adaptation est la gestion de listes favoris (bookmarks) par le navigateur Web. Ce mécanisme s'avère insuffisant quand le nombre de ressources à gérer dépasse plusieurs dizaines d'URLs et il est aujourd'hui souvent remplacé par d'autres mécanismes plus performants comme par exemple l'utilisation d'un moteur de recherche : au lieu de stocker l'URL d'une ressource comme par exemple la page personnelle d'une personne dans la liste des favoris, on envoie le nom et le prénom de la personne au moteur de recherche qui trouve rapidement la page correspondante (indépendamment de son adresse qui a éventuellement changé entre temps).

Les limites du Web comme outil de partage d'information s'expliquent en partie par un manque d'abstraction dans le modèle Web qui représente l'information par un ensemble de ressources Web reliées par des liens hypertextes. En effet, les notions de partage et échange d'information concernent plusieurs niveaux. Le premier niveau est implanté par Internet et permet un accès physique aux données dans un environnement hétérogène et distribué. Le deuxième niveau permet un traitement syntaxique de l'information grâce à XML comme nouveau standard de-facto pour la représentation de la structure de documents et de données en général. Le troisième niveau concerne la sémantique de l'information. Ce niveau est essentiel pour un vrai partage des ressources non seulement au niveau de la structure mais également au niveau de la compréhension par l'utilisateur. La solution consiste dans la création d'un "Web sémantique" qui contient des informations sémantiques sur des ressources Web sous forme de métadonnées descriptives.

L'objectif du *Web Sémantique*[BLHL01] est de fournir des outils et services nouveaux pour faciliter l'organisation, la localisation et le partage de ressources sur le Web. L'idée principale appliquée est illustrée dans la figure 1.1 qui montre un espace d'informations à deux niveaux : (1) le *niveau Web* qui contient l'information sous forme de ressources Web (niveau inférieur dans la figure) et (2) le *niveau sémantique* qui

1. Source: Nature, Mai 2000
2. Source: Internet Software Consortium (http://www.isc.org/)

représente un ensemble de *connaissances* sur les ressources et leur contenu.

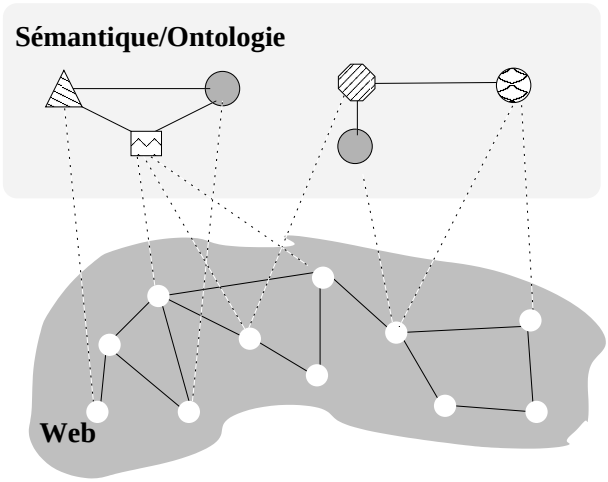

FIG. 1.1 – *Web et Sémantique*

Les ressources Web sont reliées par des liens hypertextes qui permettent de naviguer dans l'espace d'informations qu'elles représentent. La sémantique est exprimée par deux graphes dont chacun correspond à une *ontologie* qui est une représentation formelle des connaissances partagées sur un domaine. Les différentes ressources sont reliées aux deux ontologies par des arêtes en pointillés. L'ensemble de ces arêtes correspond à une *description* des ressources dans les termes de l'ontologie utilisée. Ce premier modèle est évidemment simplifié et nous allons voir qu'il existe une multitude d'approches et de solutions pour la descriptions de ressources.

Le besoin de rendre la sémantique des ressources Web plus explicite a donné naissance au développement de *portails sémantiques* qui fournissent les services nécessaires pour maintenir et exploiter cet espace à deux niveaux. Nous considérons un système comme un portail sémantique s'il satisfait les deux caractéristiques suivantes :

- Les ressources Web accessibles à travers le portail sont *autonomes* et gérées indépendamment.
- La sémantique de l'information gérée par le portail est représentée d'une manière explicite sous forme d'une ontologie qui sert comme interface (ou pivot) entre l'utilisateur, le système et les ressources.

Le premier point signifie qu'il existe une séparation logique et physique entre le portail et les sources qui doit être surmontée par le système. Le deuxième point met l'ontologie dans le centre du système. D'un côté, l'ontologie doit servir comme interface utilisateur pour accéder à l'information (nous reviendrons sur cet aspect plus tard). De l'autre côté, les ressources doivent être *décrites* dans les termes de l'ontologie du portail.

Pour la description des ressources, on peut distinguer deux approches complémentaires qui se différencient au niveau architecture et fonctionnalités :

L'approche "*métadonnées*" considère une ressource Web comme une information abstraite qui peut être référencée par une URL. Cette information est décrite dans les termes de l'ontologie, indépendamment de sa représentation physique (bases de données, document, image). Plus précisément, une description est un ensemble de *métadonnées* reliées à la ressource Web par des *liens de description*.

On obtient ainsi une architecture comme celle représentée dans la Figure 1.2. Les ontologies se transforment en *schémas de métadonnées* qui sont instanciés par des *entrepôts de métadonnées* reliant les ressources aux concepts de l'ontologie. L'utilisateur interagit avec le système à deux niveaux. Tout d'abord,

le portail peut être considéré comme une base de données qu'on peut interroger et dans laquelle on peut naviguer. A ce niveau, le résultat obtenu est toujours un ensemble de métadonnées avec des liens de description. Chacun de ces liens sert ensuite comme lien hypertexte vers la source, qui peut être une page HTML statique, une image, une base de données etc.

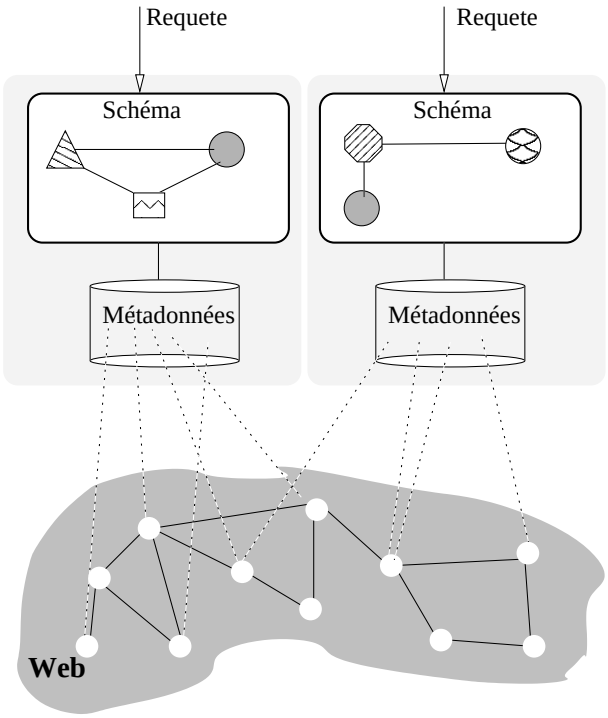

FIG. 1.2 – *Entrepôt de métadonnées*

L'approche *"vue sémantique"* prend en compte la représentation de l'information et/les fonctionnalités d'une ressource Web. Par exemple, si la ressource est une base de données relationnelle, le portail peut exploiter son interface d'interrogation pour accéder directement à l'information stockée dans la ressource. Cette approche est illustrée dans la Figure 1.3. Les sources de données [3] sont intégrées dans le systèmes grâce à la définition de *vues* ou *mises en correspondance* (mapping) entre le schéma global du système et les schémas des sources (voir section 4.1). La différence importante entre cette approche et l'approche *métadonnées* est que l'utilisateur peut directement interroger les informations dans les sources à travers une interface unique représentée par le schéma global. Cette architecture générique peut être adaptée à différents besoins et environnements et elle sert comme point de départ d'une multitude de travaux de recherche effectués pendant les dernières années (voir section 4.1).

Même si ces deux types de portails sémantiques offrent des fonctionnalités complémentaires (découverte de ressources versus interrogation directe), elles partagent un certain nombre de caractéristiques qui les rapprochent :

– Dans les deux cas, l'utilisateur interroge une base de données pour découvrir l'information recherchée.
– Le schéma de cette base de données est créé à partir d'une ontologie indépendante des ressources disponibles.

3. Nous utilisons le terme *source (de données)* pour une ressource Web qui peut être interrogée.

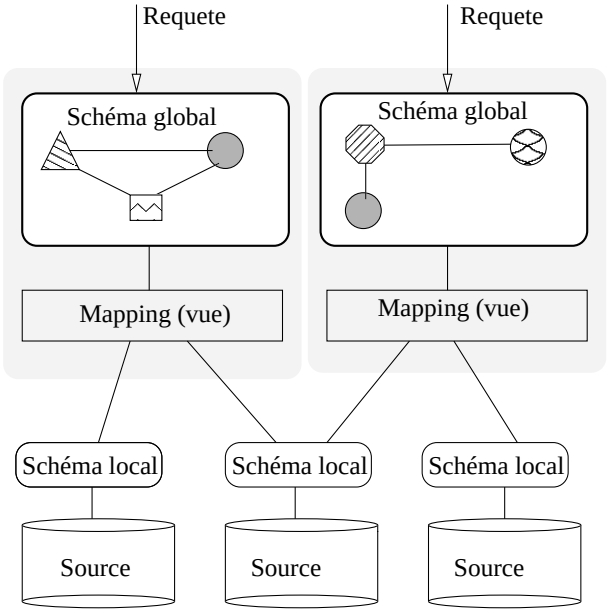

FIG. 1.3 – *Intégration de Données*

– Le portail doit continuellement s'adapter à l'évolution du Web et prendre en compte l'apparition et la disparition de nouvelles ressources en mettant à jour les métadonnées ou les vues.

1.2 Approche générale de ma recherche

Le contexte général de mon activité de recherche des dernières années concerne la problématique du partage de données sur le Web. Mon hypothèse de départ est que ce partage est aujourd'hui possible grâce à l'intégration de trois technologies complémentaires :

1. la technologie des *bases de données* qui a prouvé ses capacités à modéliser et à gérer des grands volumes de données structurées,
2. le langage *XML* comme solution au problème de l'échange des données sur le Web, et
3. les *ontologies* comme moyen d'exprimer des connaissances partagées sur un domaine d'intérêt.

Les bases de données représentent un outil important pour une meilleure utilisation des informations sur le Web [FLM98]. Ainsi, en 2000 une étude a estimé que le volume d'informations disponibles sur le Web et générées dynamiquement à partir de données dans une bases de données est 400 à 500 fois plus important que le total de la taille des pages HTML statiques [Ber00]. Cette séparation entre le traitement de l'information et sa publication permet une meilleure gestion de l'information. Mais la longue expérience dans la modélisation, le stockage et l'interrogation efficace de grand volumes de données structurées a surtout permis à la technologie des bases de données de servir comme point de départ pour des outils nouveaux qui interviennent à tous les niveaux de l'infrastructure Web. Ces outils participent non seulement à la génération et la gestion de sites Web, mais également à l'échange, la transformation et l'intégration de ressources existantes.

Les avancées théoriques et les premières expériences dans la modélisation des données sur le Web [ABS99] ont pu être mises en pratique à une plus grande échelle grâce à XML [BPSMM00]. Initialement conçu pour la création de documents structurés, le langage de balisage XML est également un modèle de données qui facilite l'échange de données entre applications. Grâce à la facilité de générer et de manipuler des données

structurées ou semi-structurées en XML et la possibilité de définir des structures de données adaptées à différents types d'applications, XML est devenu le langage universel pour la représentation de données sur le Web.

La publication et l'échange d'une information nécessite un minimum de connaissances partagées entre le producteur et le consommateur. L'expression explicite de cette connaissance est appelée une *ontologie* [Gru93] et représente un outil indispensable pour une multitude d'applications dans le contexte du Web Sémantique. Par exemple les ontologies linguistiques comme WordNet [Mil95], Mikrokosmos [Mah96] et Pangloss [KL94] expriment les différentes relations sémantiques entre les mots d'une langue et facilitent ainsi l'analyse, le traitement et l'organisation de pages HTML ou d'autres ressources récupérées sur le Web [NVVH03]. D'autres types d'ontologies plus spécialisées sont utilisés pour décrire explicitement l'information publiée par une ressource. Le rôle de l'ontologie s'approche ainsi du rôle d'un schéma de base de données qui décrit la structure des données pour faciliter leur interrogation et mise-à-jour. Dans les deux cas, un rôle essentiel d'une ontologie dans le cadre de l'intégration de données consiste à clarifier la sémantique de l'information traitée et de résoudre des conflits créés par l'hétérogénéité sémantique des sources d'informations [Hul97, KS98].

Mon activité de recherche pendant ces dernières années est fondée sur la complémentarité de ces trois technologies – bases de données, XML et ontologies – pour l'échange, la gestion et la description de données ou de ressources Web. Plus particulièrement, je me suis intéressé à la modélisation et la mise en oeuvre d'entrepôts de métadonnées et l'intégration de données XML. Dans le reste de ce chapitre, je donnerai une brève description de ces activités.

1.3 Entrepôts de métadonnées

La compréhension des besoins et des caractéristiques d'un entrepôt de métadonnées est actuellement encore incomplète et le problème de la gestion de métadonnées sémantiques se situe dans un triangle formé par les SGBD, les entrepôts de données et la gestion des connaissances. Les métadonnées sont tout d'abord des données qu'on veut stocker et interroger efficacement. Mais la richesse des connaissances d'un domaine ne permet généralement pas une traduction directe sous forme d'un schéma de base de données relationnel ou orienté-objet et on est obligé de définir des modèles nouveaux avec des langages de requêtes adaptés pour exploiter ces connaissances d'une façon naturelle et efficace. Les métadonnées représentent des connaissances sur des ressources d'information autonomes et indépendantes et, comme un entrepôt de données classique, un entrepôt de métadonnées n'est pas un monde fermé qui peut être complètement contrôlé.

Pour mieux comprendre ces besoins, nous avons étudié différentes approches et architectures et implanté deux prototypes dans le cadre de plusieurs contrats nationaux et Européens.

1.3.1 ELIOT : Un entrepôt de métadonnées orienté-objets

Le prototype ELIOT [Rad00] a été développé sous ma responsabilité dans le cadre d'un contrat entre le Conservatoire National des Arts et Métiers (CNAM) et la Direction de L'Architecture et du Patrimoine (DAPA) du Ministère de la Culture français. Les objectifs de ce prototype étaient multiples. Premièrement, nous avons voulu valider notre approche de création de schémas de métadonnées (voir section 1.3.3 et section 3.2) à travers une application réelle dans le domaine de la culture. Cette application consistait à trouver un schéma de métadonnées qui permet de décrire des ressources documentaires de la Sous-Direction des Études, de la Documentation et de l'Inventaire (C. Dessaux et G. Pinçon) et la Sous-Direction de l'Archéologie (A.M. Cottenceau). Notre deuxième objectif était de montrer que la technologie des bases de données objet est bien adaptée à la création d'entrepôts de métadonnées. Ceci nécessitait la définition d'un modèle de métadonnées orienté-objet qui permet de représenter des métadonnées sous forme d'objets complexes

qui peuvent être interrogées avec le langage de requête OQL (voir section 3.3). Troisièmement, il s'agissait d'étudier des techniques d'optimisation de requêtes sur des hiérarchies de termes (thesaurus). L'idée de ces techniques est l'utilisation de structures d'indexation standard comme les arbres B+ (pour une dimension) et les arbres R (pour deux dimensions) par la traduction des relations hiérarchiques entre termes en relation d'ordre dans un espace à une dimension. Ainsi un parcours d'arbre pour trouver par exemple tous les descendants d'un terme devient une requête intervalle sur les codes obtenu par la traduction [AFS00]. Ce travail a fait l'objet de deux stages de DEA [Rad00, Laf00] et d'une thèse [Fun03] (voir section 3.3).

1.3.2 Les projets C-WEB et MESMUSES

Le projet C-Web [cwe99] a été lancé en 1999 sous forme d'un contrat Européen IST (No. 1999-13479) de douze mois. Il réunissait comme partenaires l'INRIA (équipes Verso et Acacia), ICS/Forth et EDW, une PME italienne spécialisée dans les systèmes fondés sur SGML/XML pour la gestion d'informations d'entreprise. L'objectif était *l'évaluation et la définition d'une plate-forme pour la génération d'entrepôts de métadonnées*. Les problèmes adressés étaient multiples et concernaient essentiellement la création de schémas de métadonnées et l'interrogation de descriptions RDF [ACK+00]. Le modèle d'intégration que j'ai proposé dans [AF99] a été retenu par C-Web (voir section 1.3.3).

Cette plate-forme a été ensuite confrontée aux besoins d'utilisateurs dans le cadre du projet Européen successeur MesMuses (Metaphor for Science Museums, IST-2000-26074, Octobre 2000 - Juillet 2003). Le consortium réunissait à nouveau tous les partenaires du projet C-Web où s'ajoutaient l'ENST de Bretagne, deux autres partenaires industriels (Valoris/Euroclid, Paris et Finsiel Multimedia Services, Italie) et deux musées scientifiques comme utilisateurs (Cité des Sciences et de l'Industrie à Paris et Istituto E Museo Di Storia Della Scienza à Florence).

L'objectif de MesMuses était l'*implantation de la plate-forme C-Web* et l'installation de deux applications chez les utilisateurs. Il s'agissait de montrer qu'il est possible de réaliser les services C-Web avec la technologie XML et le standard RDF ainsi que d'autres outils génériques comme les bases de données relationnelles pour le stockage et l'interrogation de données RDF (RDFSuite), et les serveurs et protocoles Web (HTTP et WebDAV) pour la gestion de ressources sur le Web.

Dans le cadre des projets C-Web et MesMuses, j'ai été pendant un an conseiller scientifique dans l'action SAMIE dirigé par A. Michard à l'INRIA Rocquencourt. Ma contribution consistait dans l'élaboration du modèle et de l'architecture C-Web (voir chapitre 3) ainsi que l'encadrement de deux ingénieurs de recherche qui étaient chargés du développement du prototype. Le projet MesMuses a récemment donné naissance à une nouvelle entreprise AM^2 Systems[4] dirigé par A. Michard.

1.3.3 Création de schémas de métadonnées

Le rôle d'un schéma de métadonnées est double. D'un côté, il représente une ontologie des connaissances partagées sur un domaine et, de l'autre côté, il joue le rôle d'un schéma de bases de données qui sert comme interface pour la formulation de requêtes structurées sur des métadonnées ou des vues. Le Web est un espace ouvert et évolutif et la création d'un entrepôt de métadonnées ne doit pas être trop pénalisée par l'effort nécessaire à la définition de son schéma. Afin de répondre à ce besoin, j'ai proposé une nouvelle méthodologie pour la conception de schémas de métadonnées fondée sur l'intégration d'ontologies. L'idée principale est de définir une ontologie de haut niveau sous forme d'un schéma conceptuel et d'intégrer des thesaurus spécialisés déjà existants [AF99, AFS00]. Ceci permet non seulement de créer et d'adapter des schémas de métadonnées en "branchant" des thesaurus choisis par rapport aux besoin de l'utilisateur, mais également de revaloriser les efforts considérables de développement de thesaurus.

4. http://aqua.inria.fr/

Notre approche de conception de schéma de métadonnées a prouvé son utilité dans deux applications de portails sémantiques pour *ressources culturelles et scientifiques* (voir sections 1.3.1 et 1.3.2). Une particularité commune de ces applications est l'existence d'un grand nombre de thésaurus concernant les différents domaines pertinents. Ces thésaurus ont été créés par l'utilisateur même dans le cas du ministère de la culture, ou sont publiquement accessibles dans le cas de MesMuses. La deuxième particularité était le besoin de définir d'abord une ontologie de haut niveau adaptée aux besoins de l'application.

Nos expériences ont montré que notre approche est un bon compromis entre le besoin de simplicité pour l'utilisateur non-informaticien et le besoin de richesse pour la description d'informations riches. Il sera décrit plus en détail dans la section 3.2.

1.4 Intégration de données XML

L'apparition de XML comme nouveau format de publication et d'échange de données Web a révolutionné la vision du Web comme source d'information. Grâce la séparation entre le contenu et la présentation, la structure d'une ressource Web n'est plus uniquement guidée par les besoins d'affichage et peut facilement s'adapter aux besoins d'autres applications. Mais la flexibilité de XML pour la représentation de données semi-structurées a surtout ouvert des voies nouvelles pour la création de services d'interrogation et d'intégration de données sur le Web. Dans ce contexte, j'ai pu participer à plusieurs projets de recherche que je décrirai rapidement.

1.4.1 Entrepôt de données XML : Xylème

Le projet Xylème a été initié en 1999 par S. Abiteboul, S. Cluet (INRIA) et F. Bancilhon (alors Directeur Général de Arioso). Il fonctionnait comme un réseau ouvert de chercheurs de l'équipe Verso de l'INRIA, du groupe IASI du LRI à Orsay, de l'Université de Mannheim, et de l'équipe Vertigo du CNAM. L'ambition de tous les participants était de créer le premier entrepôt dynamique pour stocker *toutes les ressources XML* du Web. A partir de cet objectif commun se déclinaient les différents thèmes de recherche suivants :

- stockage efficace de données (arbres) XML [KM00];
- évaluation de requêtes avec indexation au niveau des éléments [ACW01];
- stratégies d'acquisition et maintenance de données [MPA+00];
- contrôle de changement et requêtes continues [MACM01];
- intégration sémantique de données [CVV01].

La nouveauté du projet se situait dans la volonté de créer un système qui traite l'ensemble de ces thèmes *à l'échelle du Web*. Cette nouvelle dimension (le volume des ressources à traiter et le nombre de connexions clients simultanées) est importante dans chacun des thèmes et exclut dès le départ un certain nombre de solutions traditionnelles [ACFR01].

Une tâche importante à laquelle j'ai contribué consistait à définir une architecture distribuée qui est capable de passer à l'échelle avec la vitesse de croissance du Web XML. Un composant important de cette architecture était le module d'acquisition et de maintenance des données XML. J'étais responsable du développement du premier prototype de ce module qui a été développé dans le cadre d'un stage Polytechnique (S. Ailleret) et d'un stages de DEA (A. Galland). Le développement de ce prototype était crucial pour démontrer la faisabilité et garantir le succès du système. Les résultats de ce travail et des améliorations apportées dans une version suivante (M. Preda et L. Mignet) sont décrits dans [MPA+00, MAAV01].

Une autre problématique du projet Xylème était la conception d'une interface utilisateur pour l'interrogation des documents XML stockés dans l'entrepôt. Le rôle principal de cette interface était de cacher

l'hétérogénéité structurelle des documents XML tout en gardant la possibilité de poser des requêtes structurées. La solution adoptée est fondée sur le principe de la *médiation* qui décrit chaque type de document (DTD) dans les termes d'un schéma d'intégration (DTD abstraite) et traduit les requêtes utilisateurs (posées dans les mêmes termes) en requêtes sur les documents sources [CVV01].

Le premier prototype était prêt début 2001 et servait ensuite comme point de départ pour la création de la société Xylème (http://www.xyleme.com) qui garde toujours des relations étroites avec les équipes de recherche fondatrices (LRI/Paris XI, Gémo/INRIA, Vertigo/CNAM, Univ. Mannheim).

1.4.2 Médiation de requêtes XML : STYX

Le modèle de médiation de Xylème a été conçu pour l'intégration semi-automatique d'un grand nombre de documents XML (échelle du Web). Le prix de cette automatisation est un certain nombre de limitations qui peuvent s'avérer trop restrictives dans certains cas. En particulier, l'utilisation de XML comme modèle de médiation restreint la richesse et la précision des descriptions de ressources (règles de mapping) [CVV01]. Dans le prototype STYX, j'ai étudié une autre approche d'intégration de ressources XML. L'idée était de concevoir un modèle de médiation plus riche qui permette de décrire des ressources XML en termes d'une ontologie composée de concepts et de rôles [ABFS02a].

Le langage d'ontologie est un schéma conceptuel et les requêtes utilisateurs correspondent à un sous-ensemble de OQL. Grâce à des règles de traduction et un algorithme de réécriture de requêtes, le médiateur traduit ces requêtes ensuite en requêtes XPath [CD99] ou XQuery [CFR+01] qui peuvent être évaluées par les ressources. Les réponses obtenues sont ensuite intégrées grâce à des clés globales définies au niveau de l'ontologie [ABFS02b, Fun03].

Le modèle d'intégration de STYXa été élaboré en collaboration avec C. Beeri de l'Université de Jérusalem et fait partie de la thèse d'I. Fundulaki que j'ai co-encadrée avec M. Scholl. Le projet STYX a donné lieu à un nombre de publications internationales [AFS+01, ABFS02a, ABFS02b] et une démonstration à la conférence EDBT'02 [FABS02]. Il sera décrit plus en détail dans le Chapitre 4.

1.4.3 Intégration de données XML et services Web : ActiveXML

L'émergence des services Web [SOA] comme nouveau moyen d'accès à des informations sur le Web a donné naissance à un nouveau type de documents XML que nous appelons *documents intensionnels*. Ces documents sont des documents XML où des parties de données sont définies "intensionnellement" sous forme d'appels de services Web.

ActiveXML est un environnement pair-à-pair (peer-to-peer) centré autour du concept de document XML intensionnel. Chaque pair ActiveXML contient un entrepôt de documents et peut contrôler la matérialisation des données intensionnelles en appelant les services correspondants. Quand des documents intensionnels sont échangés entre les pairs, le choix des fragments à matérialiser peut être influencé par différents paramètres comme la performance ou la sécurité. Nous proposons d'utiliser – comme pour les données XML standards – des schémas XML (DTD, XML schema) pour contrôler l'échange de données/documents intensionnels et plus particulièrement, pour guider le processus de matérialisation [MAA+03]. Nous avons formalisé ce problème et fourni des algorithmes pour le résoudre. Une réalisation avec les standards XML Schema et SOAP/WSDL a été effectuée sous ma responsabilité dans le cadre du stage DEA de F. Dang Ngoc [DN02]. Une description plus détaillée du système ActiveXML est donnée dans la section 2.1.4.

1.4.4 Construction d'un entrepôt thématique pour le risque alimentaire : Edot

Le projet RNTL e.dot a démarré en Janvier 2003 et comme but le développement d'un entrepôt thématique pour le risque alimentaire. Cet entrepôt doit enrichier des bases de données existantes avec des

informations découvertes sur le web et fournies par des fournisseurs spécialisés. Le projet s'appuie sur XML et des fonctionnalités avancées comme des requêtes de haut niveau et le monitorage du web. Les expérimentations ont pour cadre la création d'un entrepôt sur le risque alimentaire. Le travail est organisé en trois groupes de travail qui autour de (1) la spécification de l'entrepôt, (2) la définition de l'architecture et (3) l'intégration des données.

Le projet réunit trois équipes de recherche en informatique (IASI/LRI, Gémo/INRIA et BIA/INRA) et une start-up (Xyleme) qui propose des services sur le web autour de XML.

1.5 Autres projets

Pendant les huit dernières années, j'ai pu participer à un certain nombre d'autres projets, dont la plupart s'effectuaient dans le cadre d'un contrat Européen ou national.

1.5.1 Médiation de requêtes spatiales : JaGo

Après ma thèse, que j'ai soutenue en Février 1994 et qui portait sur l'interrogation d'hypertexte [Ama94], je me suis intéressé aux nouvelles architectures d'intégration de données distribuées dans le contexte des Systèmes d'Information Géographique (SIG) [RSV01]. Le prototype JaGo [Ama97] met en oeuvre une architecture Corba pour l'intégration de données spatiales. La contribution majeure de ce travail est l'illustration de l'approche médiateur/traducteur pour l'intégration de données hétérogènes et l'implantation d'une interface graphique (stage de maîtrise de H. Khou [Kho96]) pour l'affichage de cartes thématiques sur le Web [5].

1.5.2 Interrogation de schémas et données

L'objectif du projet WIRE [6] (projet ESPRIT de 1996 à 1998) était le développement d'une infrastructure pour le partage d'informations dans un Intranet sécurisé. Le consortium du projet se constituait de quatre industriels (OSF RI, O2 Technology, AIS, Zanussi) et trois laboratoires de recherche (INRIA, FIZ et Fraunhofer IGD). OSF RI était le coordinateur du projet et fournissait l'infrastructure de base pour l'intégration des différents composants.

J'étais responsable administratif et technique pour la partie INRIA de ce projet. La tâche technique consistait à concevoir et implanter des outils Web pour la formulation de requêtes OQL étendue avec des fonctionnalités de recherche plein-texte [7]. Après la spécification du langage d'interrogation, il fallait créer un outil qui permet de formuler des requêtes structurées avec une connaissance partielle du schéma sous-jacent. Le résultat est une interface utilisateur qui a été développé dans le cadre d'un stage IIE [Car97] et d'un mémoire d'ingénieur [Tom98] qui permet d'interroger le schéma et les données.

1.5.3 Vues actives pour commerce électronique : ActiveViews

L'ambition du projet ActiveViews [AAC+99, AAA+99, Mig01] était de montrer que grâce à XML la technologie des bases de données s'adapte facilement aux applications Web et plus particulièrement, au commerce électronique sur le Web. L'approche ActiveViews part de l'observation qu'une telle applications peut être décrite par un ensemble de *vues actives* qui définissent des données et des activités contrôlées sur ces données. J'ai participé à la modélisation et au développement d'un prototype, qui met en oeuvre une

5. http://cedric.cnam.fr/ amann/MapView/CarteIHM.html
6. http://www.ri.silicomp.fr/wire
7. http://www.ri.silicomp.fr/wire/docs/query-interf/query-interf.html

base de données XML (Ardent Software) et les outils Web (Java RMI, JavaScript, HTML). Une description plus détaillée du système se trouve dans la section 2.1.3.

1.6 Collaborations

Tous ces projets m'ont permis de collaborer avec les membres de l'équipe Vertigo au CNAM (M. Scholl, P. Rigaux et D. Vodislav) et d'autres groupes de recherche extérieurs au Cedric, mais également d'avoir un contact direct avec des utilisateurs pour mieux comprendre les besoins réels :

1. Parmi mes relations scientifiques extérieures au laboratoire Cedric, la plus importante est celle avec le projet Verso (S. Abiteboul et S. Cluet) à l'INRIA-Rocquencourt (formant maintenant avec l'équipe IASI du LRI Orsay (M.C. Rousset) le groupe Gémo/INRIA-Futur) où j'ai pu effectuer ma thèse et collaborer ensuite dans le cadre des projets ActiveViews, Xylème, C-Web et ActiveXML en tant que collaborateur extérieur depuis 1997.
2. En deuxième lieu, je dois mentionner l'action de développement Samie à l'INRIA-Rocquencourt (resp. A. Michard) qui m'a accueilli pendant un an (2001) dans le cadre du projet MesMuses.
3. Des liens étroits existent avec le laboratoire ICS-Forth (Crête) en la personne de V. Christophides et D. Plexousakis, liens qui ont conduit à une publication commune [ACK+00]. Ce laboratoire était aussi notre partenaire dans les projets C-Web et MesMuses.
4. Dans le cadre du projet STYX, j'ai eu le plaisir de collaborer avec C. Beeri de l'Université de Jérusalem, qui était un grand soutien dans l'encadrement de la thèse d'I. Fundulaki. Cette collaboration a également conduit à des publications communes [AFS+01, ABFS02a, ABFS02b].
5. Finalement, je dois mentionner mes échanges avec l'ENST Bretagne (P. Picouet, B. Buffereau) et l'ENST Paris (J.M. Saglio) dans le cadre du projet MesMuses.

L'expérience et les connaissances acquises sur les technologies autour du Web m'ont permis de réaliser avec P. Rigaux un ouvrage de synthèse sur XML, XPath et XSLT publié chez O'Reilly [AR02].

1.7 Bilan

1.7.1 Publications

[Livre]

[AR02] B. Amann, P. Rigaux. Comprendre XSLT O'Reilly, 2002.

[Thèse]

[Ama94] B. Amann Interrogation d'hypertextes Thèse du 3e cycle, CNAM, 1994

[Revues internationales (avec comité de lecture)]

[AFS00] B. Amann, I. Fundulaki, M. Scholl, Integrating ontologies and thesauri for RDF schema creation and metadata querying. *Int. Journal of Digital Libraries (JODL'00)*.

[ASR95] B. Amann, M. Scholl, et A. Rizk. Schema-based authoring and querying of large hypertexts. *Int. Journal of Human-Computer Studies (IJHCS)*, 43(3):281–299, September 1995.

[Revues nationales (avec comité de lecture)]

[ABFS03] B. Amann, C. Beeri, I. Fundulaki, M. Scholl Interrogation de Ressources XML Concernant un Domaine d'Intérêt *Techniques et science informatiques - no spécial BDA'02*

[**Conférences internationales** (avec comité de lecture)**]**

[MAA+03] T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, et F. Dang Ngoc. Exchanging intensional xml data. In *SIGMOD*, 2003. (article en cours de soumission au journal ACM TODS)

[ABIS02b] B. Amann, C. Beeri, I.Fundulaki, et M. Scholl. Querying XML sources using an ontology-based mediator. In *CoopIS*, 2002.

[ABFS02] B. Amann, C. Beeri, I. Fundulaki, M. Scholl. Ontology-Based Integration of XML Web Resources In *Proceedings Intl. Semantic Web Conference 2002 (ISWC 2002)*, Sardinia, 2002.

[AFS+01] B. Amann, I. Fundulaki, M. Scholl, C. Beeri, et A.M. Vercoustre. Mapping xml fragments to community web ontologies. In *Proceedings Fourth International Workshop on the Web and Databases (WebDB'2001)*, Santa Barbara, California, 2001.

[ACK+00] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle, B. Amann, I. Fundulaki, M. Scholl, et A.M. Vercoustre. Managing rdf metadata for community webs. In *Workshop on the Web and Conceptual Modeling (WCM'2000)*, Salt Lake City, Utah, November 2000.

[AAC+99] S. Abiteboul, B. Amann, S. Cluet, A. Eyal, L. Mignet, T. Milo Active Views for Electronic Commerce *Int. Conf. on Very Large Databases (VLDB99)*, Edinburgh, 1999.

[AF99] B. Amann, I. Fundulaki Integrating Ontologies and Thesauri to Build RDF Schemas *European Conf. on Digital Libraries (ECDL'99)*, Paris, 1999.

[AVFC98] B. Amann, D. Vodislav, J. Fernandes, et G. Costes. Browsing SGML documents with maps : The French "Inventaire" experience. In *Proceedings of the Ninth International Conference on Data and Expert Systems Applications (DEXA'98)*, Vienna, Austria, August 1998. Springer Verlag.

[ARS94] B. Amann, A. Rizk, et M. Scholl. Querying typed hypertexts in Multicard/O_2. In *Proceedings of the ACM European Conference on Hypermedia Technology (ECHT'94)*, pages 4–10, Edinburgh, Scotland, September 1994.

[ACS93] B. Amann, V. Christophides, et M. Scholl. HyperPATH/O2: Integrating hypermedia systems with object-oriented database systems. In *Proceedings of the Fourth International Conference on Data and Expert Systems Applications (DEXA'93)*, Prague, Czech Republic, September 1993. Paru également dans *IXèmes Journées Bases de Données Avancées*, Toulouse, France, Septembre 1993.

[AS92a] B. Amann et M. Scholl. Application of a graph model to hypertext querying. In *Proceedings of the International Conference on Human-Computer Interaction (EWHCI'92)*, St.Petersburg, Russia, August 1992.

[AS92b] B. Amann et M. Scholl. Gram: A graph data model and query language. In *Proceedings of the Fourth ACM Conference on Hypertext and Hypermedia (ECHT'92)*, Milano, Italy, December 1992. Paru également dans *VIIIèmes Journées Bases de Données Avancées*, Trégastel, France, Septembre 1992.

[**Conférences nationales (avec comité de lecture)]**

[MPA+00] L. Mignet, M. Preda, S. Abiteboul, S. Ailleret, B. Amann, et A. Marian. Acquiring XML pages for a web house. In *XXIèmes Journées Bases de Données Avancées*, Blois, 2000.

[AAC+98] S. Abiteboul, B. Amann, S. Cluet, T. Milo, et V. Vianu. Active views for electronic commerce. In *XIVèmes Journées Bases de Données Avancées*, Hammamet, Tunisia, October 1998, version révisée soumise à la conférence internationale VLDB'99.

[**Rapports de contrats et rapports de recherche]**

[ACFSV98] Bernd Amann, Vassilis Christophides, Irini Fundulaki, Michel Scholl et Anne-Marie Vercoustre Intelligent Mediation of Cultural Information Sources. ERCIM News No.35 - October 1998.

[AM01] B. Amann et A. Michard. C-Web Functional Spécification, Mai 2001

[Ama97] B. Amann. Integrating GIS components with mediators and CORBA. Technical Report 97-09, Cedric-CNAM, CNAM, Paris, May 1997. URL : ftp://sikkim.cnam.fr/pub/Reports/GISMed.ps.gz

[Démonstrations]

[AC97] B. Amann et E. Carsenat OQL Query Designer. Démonstration à ICDE'97.

[FA02] I. Fundulaki, B. Amann, C. Beeri, et M. Scholl. STYX : Connecting the XML World to the World of Semantics. Démonstration à EDBT'02.

[AAB+02] Serge Abiteboul, B. Amann, J. Baumgarten, O. Benjelloun, F. Dang Ngoc and T. Milo. Schema-driven Customization of Web Services Démonstration à VLDB'03.

1.7.2 Projets et contrats de recherche

Dans le cadre des projets et contrats de recherche mentionnés dans les section 1.3 et 1.4 j'ai pu remplir diverses responsabilités administratives et techniques :

WIRE (projet ESPRIT, section 1.5.2) :

- responsable administratif (gestion de contrats, rapports d'avancement)
- responsable technique (délivrables et développement);

Eliot (contrat avec le Ministère de la Culture Français, section 1.3.1) :

- responsable du développement du prototype ELIOT
- participation à la conception d'une ontologie pour la Direction de L'Architecture et du Patrimoine (DAPA) du Ministère de la Culture français

C-Web/MesMuses (IST, section 1.3.2) :

- participation à la conception de la plate-forme C-Web
- participation à l'implantation du prototype MesMuses
- participation à l'élaboration d'une "ontologie du vivant" pour la Cité des Sciences et l'Industrie (CSI)

Edot (RNTL, section 1.4.4) :

- responsable INRIA

1.7.3 Prototypes

J'ai participé à la conception des plateformes ActiveViews, Xyleme, C-Web et ActiveXML et j'ai été responsable de la réalisation d'un certain nombre de prototypes qui ont été développés dans le cadre de différents stages d'étudiants et la thèse d'Irini Fundulaki :

STYX(section 1.4.2) : prototype développé par Irini Fundulaki dans le cadre de sa thèse que j'ai pu co-encadrer avec Michel Scholl (CNAM) et Catriel Beeri (Univ. de Jérusalem)

JaGo (section 1.5.1) : prototype développé par moi même et H. Khou dans le cadre de son stage de Maîtrise [Kho96].

OQL Query Designer (section 1.5.2) : prototype implanté dans le cadre d'un stage IIE [Car97] et d'un mémoire d'ingénieur [Tom98]. Il a été présenté aux divers revues ESPRIT, manifestations nationales (Fête de la Science) et la conférence ICDE'97 (Data Engineering) à Birmingham.

Eliot (section 1.3.1) : prototype développé dans le cadre du mémoire d'ingénieur de S. Radicevic [Rad00] et de la thèse d'Irini Fundulaki [Fun03]

Robot Xylème (section 1.4.1) : le premier robot Xylème à été conçu et développé sous ma responsabilité dans le cadre d'un stage stage Polytechnique (S. Ailleret) et d'un stages de DEA (A. Galland)

1.7.4 Encadrements

Thèses

– "Intégration et Interrogation de Ressources XML pour Communautés Web", Irini Fundulaki, thèse (CNAM), Janvier 2003. Thèse co-encadré avec Michel Scholl et Catriel Beeri.

Stages DEA

– "Typage de documents XML avec appels de services", F. Dang-Ngoc, stage de DEA (Univ. Paris VI), Septembre 2002
– "Conception d'un protocole de travail coopératif pour la réalisation d'une base de données scientifique sur le WEB", P. Saby, stage de DEA (Univ. Paris VI), Septembre 1998
– "Interopérabilité : Corba et le SGBD O_2", J. Thorner, stage de DEA (Univ. Paris VI), Septembre 1995

Stages en co-encadrement

– "Xyleme Loader", A. Galland, stage de DEA (Paris VI), co-encadré avec G. Ferran, INRIA.
– "Un robot pour le Web", S. Ailleret, stage école Polytechnique, Juillet 1999, projet Xyleme, INRIA.
– "Le Prototype ActiveViews", R. Dhaou, stage DEA (Paris VI), Août 1998, co-encadré avec S. Abiteboul, INRIA.
– "La notification sous Java/O2", S. Arnoud, stage École Polytechnique, Juillet 1998, co-encadré avec S. Abiteboul, INRIA.

Stages Maîtrise (et équivalent)

Les stages suivants ont été encadrés dans le cadre de mon activité de recherche :

– "Modélisation et implantation d'un répertoire de services", R. Pop, stage de fin d'études de l'Université Polytechnique de Bucarest (Politehnica), Septembre 2003.
– "Implantation d'un serveur Web avec O2Web", R. Tkito et S. El-Hanchi, stage de fin d'études (EMI), Maroc, Juillet 1998.
– "Interface d'interrogation graphique pour le Web", E. Carsenat, stage IIE, Juillet 1997.
– "Interface graphique pour SIG en Java", H. Khou, Stage de Maîtrise (Univ. de Versailles), Juillet 1996

Mémoires Ingénieur

Un mémoire ingénieur CNAM dure entre 9 et 12 mois et se termine par un rapport et une soutenance. J'ai encadré en tout un dizaine de mémoires d'ingénieur dont deux étaient liés à mon activité de recherche :

– "Mise en oeuvre XML d'un portail Web de fonds culturels", S. Radicevic, mémoire d'ingénieur CNAM, 2002.
– "Outils d'aide à la formulation de requêtes sur des schémas complexes", I. Tomescu, mémoire d'ingénieur CNAM, 1998.

1.7.5 Comités de programmes

Conférences internationales

- Very Large Databases (VLDB'03)
- International Conference on Extending Database Technology (EDBT'02)
- Database and Expert Systems Applications (Dexa'03)

Conférences nationales

- Journées Bases de Données Avancées (BDA'99)
- Journées Inforsid'99
- Journées Françaises de la Toile (JFT'03)

Workshops

- WebDB'03 (workshop associé à SIGMOD'03)
- XML Database Symposium (XSym'03) (workshop associé à VLDB'03)
- Data Integration over the Web (associé à CAiSE'01)
- Semantic Web (associé à ECDL'00)

Je suis membre du comité de rédaction de la revue http://www.Revue-I3.org/ et, depuis Janvier 2003 expert pour l'évaluation des propositions RNTL.

1.7.6 Écoles d'été

- Modelware : vers la modélisation et la sémantisation de l'information, École d'informatique CEA/EDF/INRIA, 16-27 juin 2003. J'ai donné dix heures de cours sur la Gestion de Contenus Web.

1.7.7 Organisation de colloques

- Responsable de l'organisation des démonstrations pour la conférence ECDL'99 (European Conference on Digital Libraries) à Paris.

1.8 Plan du Rapport

La suite de ce rapport est organisée en quatre chapitres. Le chapitre 2 décrit les deux approches complémentaires au problème du partage d'informations sur le Web qui sont concrétisées dans les différentes recommandations autour de XML et RDF [DvHB+00] du W3C. L'objectif de ce chapitre est une introduction générale pour situer mon activité de recherche dans son contexte scientifique. Le chapitre 3 présente mes contributions de recherche dans la construction d'entrepôts de métadonnées sémantiques. Le chapitre 4 étudie le problème de l'intégration de données sur le Web ainsi que les différentes architectures et approches proposées. Je décrirai également notre modèle STYXqui propose une solution originale pour l'intégration et l'interrogation de ressources XML. Je terminerai ce rapport avec une description de mon activité de recherche actuelle et de ses perspectives.

Chapitre 2

Données et connaissances sur le Web

Dans ce chapitre, nous décrirons deux visions complémentaires du problème de partage d'informations sur le Web qui sont concrétisées dans les deux recommandations XML [BPSMM00] et RDF [DvHB+00] du W3C. L'objectif de cette introduction générale est surtout de clarifier le contexte scientifique de mon activité de recherche. Pour une description plus complète des différentes approches et solutions en terme d'architecture et de modélisation nous conseillons au lecteur les nombreux travaux de synthèse qui existent sur le sujet [Hul97, Via01, BLHL01].

Ce chapitre est organisé en trois sections. La première section décrit l'utilisation de XML comme moyen de transformer le Web en source de données semi-structurées qui peuvent être interrogées et intégrées. Elle présente également deux projets auxquels j'ai participé et qui montrent l'utilisation concrète de XML pour la gestion de données sur le Web. La section 2.2 montre le besoin d'ontologies pour le partage d'informations sur le Web et le rôle de RDF dans création du Web Sémantique. Le chapitre se termine avec une discussion sur la double fonction des ontologies comme schéma de connaissances et schémas de données.

2.1 XML : Données sur le Web

À son origine, le Web a été conçu comme un moyen de publication et d'échange d'informations à grande échelle [BLCGP92]. La standardisation de HTML comme format de publication permettait la création et distribution de navigateurs gratuits et garantissait ainsi que chaque utilisateur pouvait facilement consulter les informations publiées sur un site Web. L'apparition d'architectures plus sophistiquées pour la génération dynamique de pages HTML à partir d'une base de données a transformé le Web en interface vers les systèmes d'information. Néanmoins, l'exploitation de cette information restait limitée à cause du format HTML. Même si une donnée est structurée initialement, sa traduction en HTML nécessite la destruction (partielle) de sa structure et rend une exploitation par un programme plus difficile. Par exemple, après avoir traduit une table relationnelle contenant des noms de produits et leurs prix en tableau HTML, l'extraction du prix d'un produit (qui correspond à une simple requête SQL) devient plus compliquée et doit prendre en compte des choix de présentation de l'information dans la page.

Le standard XML a généralisé la notion de publication grâce à la possibilité de définir des structures adaptées pour l'échange de données entre systèmes d'informations et d'autres applications. L'exemple 2.1 montre un document XML contenant des informations sur le peintre Van Gogh et deux de ses peintures. L'information est structurée grâce à des balises qui désignent le sens de l'information qu'elles entourent :

Exemple 2.1 *van-gogh.xml* : *Un document XML*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<Peintre>
```

```
  <Nom>Van Gogh</Nom>
  <Peintures>
    <Peinture titre='Portrait du Docteur Gachet'
              année='1890'>
      La pose choisie, avec la tête s'appuyant sur la main, est
      traditionnellement celle de la mélancolie, et l'expression de
      Gachet, la tristesse qui se lit dans ses yeux, l'impression
      d'affaissement du corps, tout concourt à créer une impression de
      malaise.
    </Peinture>
    <Peinture titre='La Chaise et la Pipe'
              année='1888' >
      En posant sur la paille du siège sa pipe et son tabac, Van Gogh
      passe d'une chaise à <<sa>> chaise. C'est un substitut
      d'autoportrait.
    </Peinture>
  </Peintures>
</Peintre>
```

L'origine de ce document n'a pas d'importance du point de vue de l'application qui veut le traiter. On peut ainsi s'imaginer qu'il a été créé (1) avec un éditeur de texte standard, (2) par une requête vers une base de données ou (2) par un appel à un service Web [SOA] ou (4) par une intégration de tous ces différents types de sources.

Grâce à la possibilité de définir des balises adaptées pour *désigner et structurer* les informations indépendamment de leur présentation, XML est considéré comme un successeur de SGML pour la représentation de documents structurés. Mais d'un point de vue base de données il propose surtout un modèle fondé sur la représentation de données sous forme d'arbres avec des noeuds étiquetés [1]. La figure 2.1 illustre les caractéristiques essentielles du modèle DOM [BPSMM00] sous-jacent à la syntaxe XML : un document XML est un arbre *ordonné* avec une racine de type **Document** et composé d'éléments (noeuds de type **Element**), d'attributs (noeuds de type **Attr**) et de noeuds de type **Texte**. Le lecteur est invité à consulter le site du W3C [2] et [AR02] pour une description plus détaillée de la syntaxe XML et du modèle DOM.

2.1.1 XML : Un modèle de données semi-structurées

Le modèle de XML fait partie de la catégorie des *modèles de données semi-structurées* [ABS99] qui partagent tous l'idée de représenter des données sous forme de *graphes étiquetés*. Ce choix permet une grande flexibilité dans la représentation d'informations structurées. Contrairement aux modèles de données classiques (modèle relationnel, modèle objet) qui représentent les données comme des instances d'un schéma, les modèles semi-structurées assouplissent cette séparation entre schémas et données en proposant une représentation unique pour stocker, échanger, interroger et transformer l'information et sa structure.

Une multitude de modèles de données semi-structurées ont été proposés dans la littérature [ABS99, QRS+95, BDHS96]. Il se distinguent essentiellement dans la fonction d'étiquetage, l'identification des noeuds et la structure des graphes. L'objectif de ces modèles est de fournir une représentation à la fois

1. XML permet également de créer des références entre des noeuds d'un même document (ID/IDREF) et des liens (XLink) entre des documents, mais les graphes créés ainsi sont généralement considérés comme structures secondaires par rapport à la structure abrorescente.

2. http://www.w3c.org

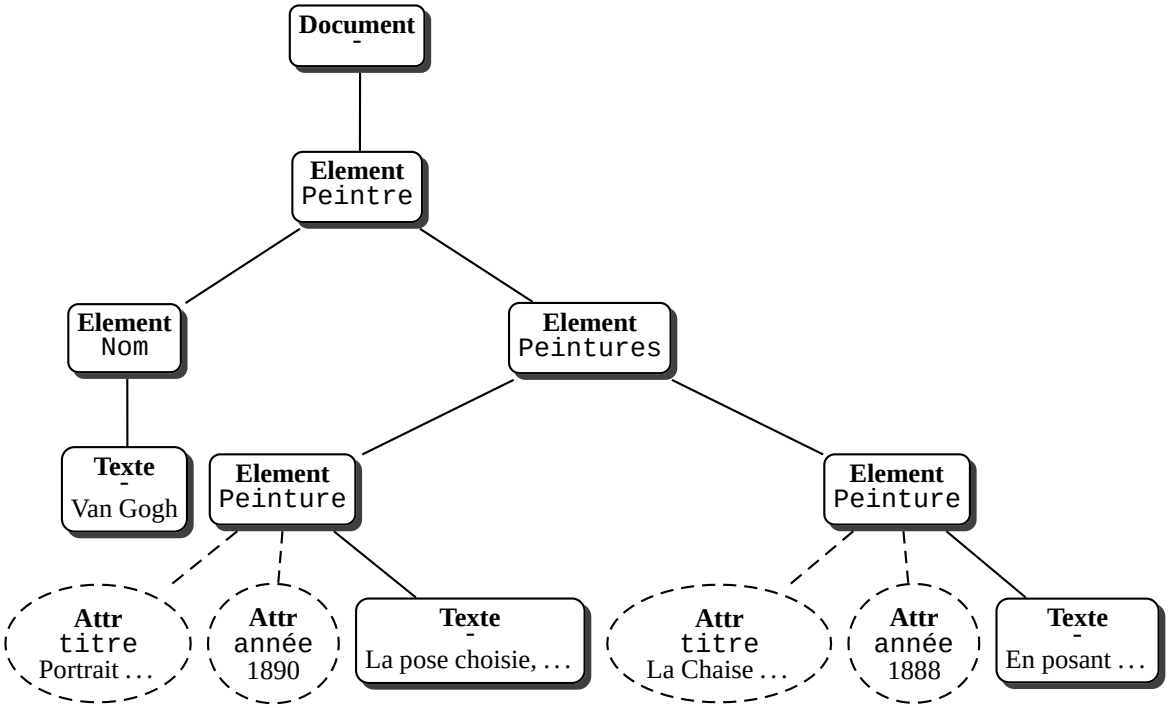

FIG. 2.1 – *Arbre DOM du document vang-gogh.xml*

flexible et précise pour des données semi-structurées. Par rapport à ces modèles, la norme XML met plus en valeur la notion de document :

- Bien qu'il soit possible de représenter des graphes (de données) par des références et des liens, XML privilégie la représentation de structures arborescentes (de documents).
- XML est sensible à l'ordre des noeuds dans un arbre. En effet, la notion d'ordre joue un rôle important quand il s'agit d'interroger ou de transformer un document, mais devient insignifiant et même gênant quand il s'agit des grand volumes de données (par exemple, l'absence d'un ordre parmi les n-uplets d'une relation facilite l'optimisation de requêtes et l'utilisation d'index).

2.1.2 XML et les bases de données

La rencontre entre le Web et les bases de données grâce au standard XML et les modèles de données semi-structurées a été un grand pas dans la modélisation des données sur le Web [ABS99, Via01]. Une multitude de systèmes a permis de montrer la conformité de l'approche bases de données pour le stockage, l'interrogation, l'intégration de données XML.

Stockage : Bien qu'il soit possible de stocker des documents XML sous leur forme sérialisée dans des fichiers textes standards, il est évident que cette solution est inefficace quand il s'agit de manipuler des grands volumes d'informations. Le stockage efficace de données XML a fait l'objet d'un grand nombre d'activités de recherches et une multitude de solutions ont été proposées. D'une manière générale, on peut distinguer entre deux approches. L'approche "natif" propose un modèle logique centré autour de la notion de document composé d'éléments et d'attributs tandis que l'approche "SGBD étendu" étend la technologie des systèmes de gestion bases de données relationnels et orientés-objet pour le traitement efficace de données XML. Tous ces systèmes ont leurs avantages et inconvénients (voir [AYF02] pour un état de l'art), mais ils montrent surtout que la technologie des bases de données a bien su s'adapter à ce nouveau type de données.

Une liste exhaustive de systèmes XML avec des liens vers les sites Web respectifs peut être trouvée sur le site http://www.rpbourret.com/xml/XMLDatabaseProds.htm.

Schémas : Les schémas jouent un rôle important dans la manipulation de données structurées. Ils ne garantissent pas seulement un certain degré de cohérence des données manipulées, mais rendent également le traitement des requêtes plus efficace. Comme tous les modèles de données semi-structurées, le standard XML n'oblige pas la description explicite de la structure (du type) d'un document. Néanmoins, une telle description s'avère utile pour le traitement (édition et interrogation) et l'échange de documents XML.

Formellement, la structure arborescente d'un document XML peut être décrite par un automate (ou une grammaire) d'arbre [CDG+, Via01]. On peut définir une hiérarchie de grammaires où chaque niveau inclut les grammaires des niveaux inférieurs [MLM01]. Les DTD sont suffisantes pour décrire la structure syntaxique d'un document XML, mais trop limitées pour modéliser des données plus riches. D'autres langages (XML Schema [TBMM01], Relax NG [rel]) séparent la définition des types et proposent des systèmes de typage plus riches (sous-typage, types atomiques).

Interrogation et transformation : La définition et l'implantation de langages de requêtes pour les données semi-structurées et XML en particulier représentent une partie importante de l'effort de recherche en bases de données pour le Web. Ainsi, il existe une multitude d'implantations de langages de requêtes XML (Galax[3], GMD-IPSI XQL[4], Kweelt[5], e-XMLMedia[6], XOQL [Agu02]) qui sont tous similaires et permettent d'inspecter les données et leur structure grâce à des expressions de chemins (par exemple XPath [CD99]). Parmi les contributions, il faut aussi mentionner l'effort de standardisation d'un langage de requête pour XML (XQuery [CFR+01]).

Le langage de transformation XSLT [Cla99, AR02] répond au besoin de restructuration de données XML avant d'effectuer d'autres traitements comme par exemple la publication sur un site Web ou l'importation dans une base de données [Mun00]. Il permet l'extraction d'informations mais également la génération de noeuds et d'identifiants.

Mises-à-jour : Il existe plusieurs moyens pour générer et modifier un document XML. Le moyen le plus simple est d'utiliser un éditeur de texte standard pour éditer la forme sérialisée du document. L'inconvénient de cette méthode est le manque de contrôle pendant l'édition du document. Les éditeurs XML comme XMLSpy[7] permettent de contrôler le processus d'édition à travers une DTD ou un schéma XML prédéfini. La définition de langages de mise-à-jour déclaratifs [LM00, TIHW01] a attiré jusqu'à maintenant moins d'attention que les langages de requêtes et de transformation. Ceci peut être expliqué par le fait que XML est souvent utilisé pour l'échange d'informations, mais n'a pas encore la maturité du modèle relationnelle pour la gestion de données. Par exemple, un problème important de la recherche actuelle est la définition de modèles de contraintes XML [FKS01b] pour garantir le même niveau de cohérence comme dans une base de données classique.

Vues : Les langages de vues pour XML (combinés avec les langages d'interrogation et de transformation) répondent à une multitude de besoins classiques comme la personnalisation des données et l'échange et l'intégration de données hétérogènes [Abi99]. Dans les deux sections suivantes je décrirai deux projets

3. http://db.bell-labs.com/galax/
4. http://www.ipsi.fraunhofer.de/oasys/projects/ipsi-xq/index_e.html
5. http://kweelt.sourceforge.net/
6. http://www.e-xmlmedia.com/home/index.htm
7. http://www.xmlspy.com/

auxquels j'ai participé et qui utilisent une combinaison des différentes technologies mentionnées, et plus particulièrement la notion de vue sur des documents XML.

2.1.3 ActiveViews : Vues Actives pour le commerce électronique

Le prototype ActiveViews [AAC+99, Mig01] propose un environnement de développement d'applications de commerce électronique fondé sur la notion de *vue active* pour le contrôle de mises-à-jour de données XML stockées dans un entrepôt XML [AAA+99]. Chaque application est un ensemble de vues actives définies par des expressions qui permettent de définir (1) les données manipulées par l'application, (2) un ensemble de méthodes qu'on peut appliquer à ces données, (3) un ensemble d'activités et (4) un ensemble de *règles actives*.

Le langage ActiveViews : AVL

Les données manipulées dans une vue active sont spécifiées par des expressions **let-be-with-mode** :

let *var*: *T*
be *Q*
with *liaisons*
mode *droits*

Cette expression définit une variable *var* de type *T* (clause **let**) et l'affecte avec les données XML extraites de l'entrepôt par la requête XOQL [Agu02] *Q* (clause **be**). Les deux clauses suivantes **with** et **mode** définissent les droits d'accès aux données de la vue : l'expression *liaisons* affecte des fragments XML à des variables qui sont ensuite utilisées dans l'expression *droits*. On distingue quatre types de droits d'accès : **read**, **write**, **append**, **remove** (par défaut toutes les données sont accessibles en lecture).

Comme exemple, l'expression suivante affecte une variable p de type $(Peintre)*$ avec les résultat d'une requête XOQL qui extrait tous les éléments de type `Peintre` d'une collection de documents XML désignée par le nom `ART`. La clause **with** définit deux variables n et x qui correspondent respectivement au nom d'un peintre et à l'ensemble de ses peintures. Finalement la clause **mode** spécifie que toutes les données sont visibles (**read all**) et qu'il est possible de modifier le nom du peintre (**write** n) et d'ajouter des nouvelles peintures pour chaque peintre (**append** x).

let p: (Peintre)*
be select a from a in ART/Peintre
with p.Nom n, p.Peintures x, p.*
mode **read all**, **write** n, **append** x

L'entrepôt XML utilisé par ActiveViews est construit au dessus du SGBD orienté-objet O_2 ce qui permet d'associer des méthodes aux données XML stockées sous forme d'objets. Ces méthodes peuvent être intégrées dans la vue par des expressions de la forme suivante :

method $signature$
is $message$
if $condition$

La clause **method** spécifie la $signature$ de la méthode "virtuelle" dans la vue. Le lien vers la méthode concrète est effectué par la clause **is** qui définit le message à envoyer. Finalement, la clause (optionnelle) **if** décrit des pré-conditions nécessaires pour pouvoir appeler la méthode.

Supposons qu'il soit possible d'emprunter une peinture pour une période donnée en appelant une méthode $emprunter(x : Periode)$ définie dans la classe $Peinture$. L'expression suivante utilise cette méthode pour définir une méthode "virtuelle" $empr\_peinture(p : Peinture,y : Periode)$ qui prend une peinture et une période en entrée :

method $empr\_peinture(p : Peinture,y : Periode)$
is $p \rightarrow emprunter(y)$
if $p \rightarrow est\_libre()$

La méthode booléenne $est\_libre()$ permet de savoir si une peinture peut être empruntée. Elle se traduit dans l'interface graphique par un bouton ***empr_peinture*** à coté de chaque peinture p qui peut être activé si $p \rightarrow est\_libre()$ retourne $vrai$.

Une deuxième manière de décrire le comportement d'une application ActiveViews est la définition de règles *événement-condition-action* où un événement généré par une vue peut déclencher des méthodes dans d'autres vues (voir [AAC+99] pour plus de détails).

Génération d'Interfaces

À partir d'une spécification AVL un compilateur génère automatiquement une application complète avec un ensemble de vues actives et d'interfaces utilisateur adaptées. Une telle interface est montrée dans la figure 2.2. Elle permet à un vendeur d'aider un client pendant l'achat d'un instrument de musique. L'interface

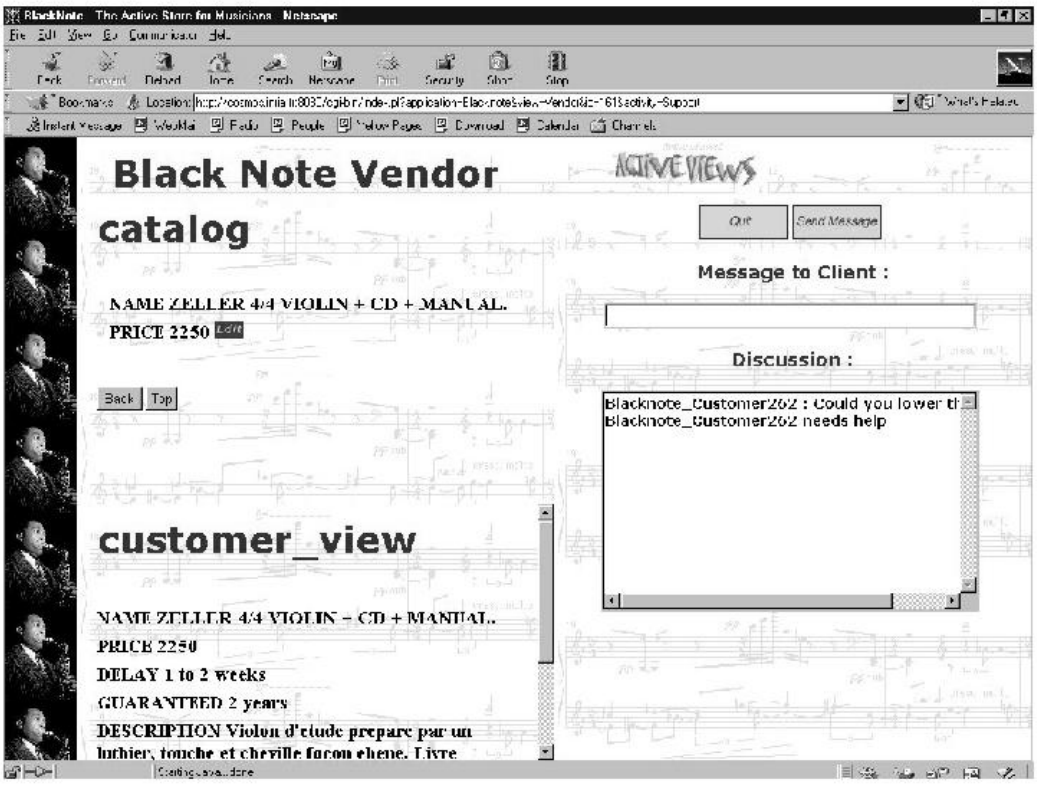

FIG. 2.2 – *Interface d'une vue ActiveViews*

est composé de trois parties. En haut à gauche, le vendeur peut naviguer dans le catalogue et choisir des instruments. La vue du client en contact avec le vendeur est montrée en bas à droite et les deux acteurs peuvent communiquer à travers un simple formulaire de communication affichée à gauche de l'écran.

En occurrence, le vendeur a choisi un violon Zeller au prix de 2250 Francs (l'exemple date de la période avant l'euro). Le bouton ***edit*** indique qu'il a le droit de changer le prix de l'instrument (ce qui n'est évidemment pas le cas pour le client). Voici la définition de cette partie de la vue vendeur sur le catalogue :

```
let fresh catalog : (CATEGORIES)*
 be {select $m.CATEGORIES.CATEGORY
     from $m in $catalog.META_CATEGORIES.META
     where $m.NAME == 'Musical Instruments'}
 with self.* X , X.ARTICLE Y, X.NAME Z, Y.PRICE P, Y.NAME N
 mode deferred read X, immediate read Y Z N, write P
```

Le paramètre `fresh` dans la déclaration de la vue indique que toutes les modifications du catalogue doivent immédiatement être transmises à l'interface par un mécanisme de notification. La vue distingue également entre deux modes de lecture. Le catalogue est lu en mode `deferred` : au lieu de charger tout le catalogue dans le client, le chargement est effectué d'une manière incrémentale pendant la navigation d'un produit vers l'autre. Chaque article est lu immédiatement (`immediate read`) avec son nom et son prix.

Architecture

L'architecture générale d'une application ActiveViews est montrée dans la figure 2.3. Elle est composée de quatre types de modules principaux :

1. L'*entrepôt XML* qui stocke toutes les données XML persistantes de l'application.
2. Un *gestionnaire d'application* qui intègre trois sous-modules pour le traitement des évènements générés par l'application :
   - Le module de mises-à-jour reçoit le flot des changements de l'entrepôt. Il avertit les vues appropriées des changements qui les concernent.
   - Le module de journalisation garde le journal des événements générés par les vues sous forme d'un document XML qui peut être interrogé.
   - Le module de règles actives gère un ensemble de règles qui sont déclenchées conformément aux événements générés par les vues.
3. Un ensemble d'instances de *vues actives* qui accèdent aux données de l'entrepôt XML. Ces instances ou clients sont *actifs* : premièrement, ils peuvent réagir à des modifications de données dans l'entrepôt grâce au mécanisme de notification de changements. Deuxièmement, ils peuvent déclencher et réagir à des évènements spécifiés par des règles actives.
4. Chaque instance d'une vue active est accessible à travers une *interface Web* standard. Cette interface est capable de réagir automatiquement à tous les changements d'état de la vue grâce à un protocole de communication entre le serveur de vues et les clients Web.

En résumé, on peut décrire ActiveViews comme un système qui intègre différents mécanismes (vues, droits d'accès, trigger) dans un langage déclaratif (AVL) et les adaptent au contexte XML. Une description plus détaillée du modèle et de l'architecture système ActiveViews peut être trouvée dans [AAC+99, Mig01].

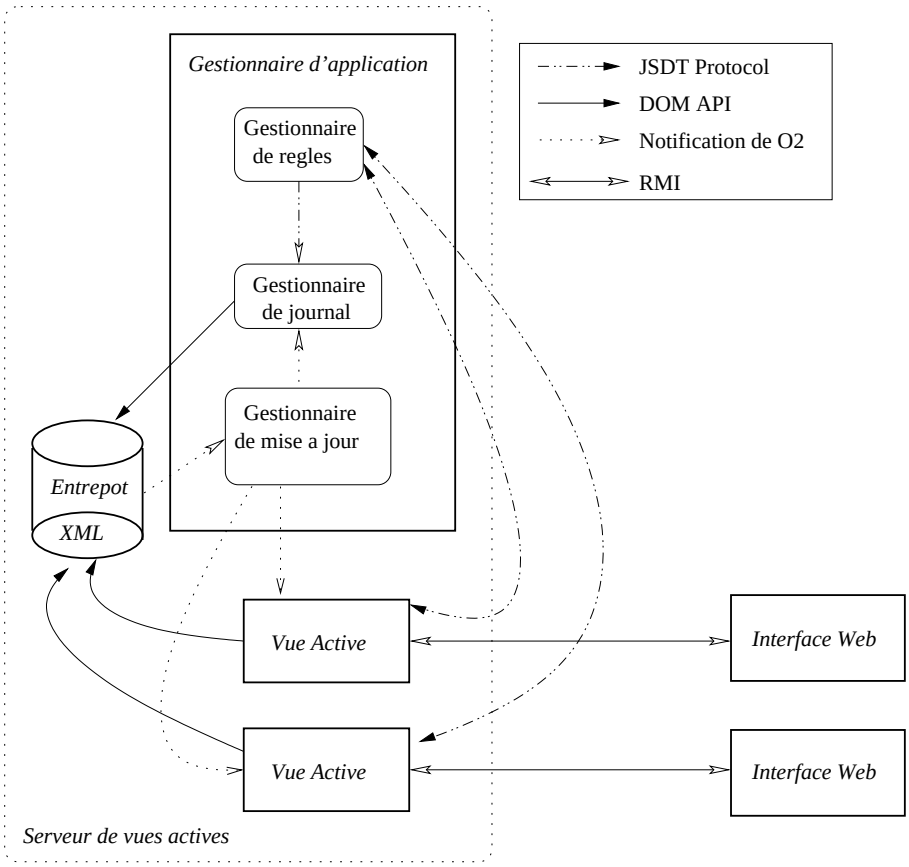

FIG. 2.3 – *Architecture du système ActiveViews*

2.1.4 Données et Services : ActiveXML

Dans le projet ActiveXML [ABM02, MAA+03] (à ne pas confondre avec ActiveViews) la notion de vue est incarnée par des *documents intensionnels* qui sont des documents XML contenant des données et des appels de services Web [SOA]. Ce type de documents permet d'intégrer d'une manière transparente à l'utilisateur des données XML avec des résultats d'appels de services qui peuvent ensuite être interrogés avec le langage de requête XOQL [Agu02].

Documents intensionnels

L'exemple suivant montre un document ActiveXML avec deux appels de service Web représentés sous forme d'éléments de type `int:fun` avec l'information nécessaire pour effectuer l'appel de service. Par exemple, le premier appel déclenche l'opération *Get_Temp* (indiquée par l'attribut `method_name`) du service *http://www.forecast.com/soap* (indiqué par l'attribut `endpointURL`).

```
<?xml version="1.0"?>
<newspaper
 xmlns:int="http://www.vwx.com/namespace/int">
 <title> The Sun </title>
 <date> 04/10/2002 </date>
 <int:fun
   endpointURL="http://www.forecast.com/soap"
   methodName="Get_Temp"
   namespaceURI="urn:xmethods-weather">
  <int:params>
    <int:param>
       <city>Paris</city>
    </int:param>
  </int:params>
 </int:fun>
 <int:fun
     endpointURL="http://www.timeout.com/paris"
     methodName="TimeOut">
     namespaceURI="urn:timeout-program">
  <int:params>
    <int:param> exhibits </param>
  </int:params>
 </int:fun>
</newspaper>
```

L'approche ActiveXML propose une architecture *pair-à-pair* (peer-to-peer) fondée sur l'échange de documents intensionnels [ABM02] entre différents pairs ActiveXML. La possibilité de mélanger des données et des appels de service dans un tel document permet à chaque pair de *matérialiser* (c.a.d. remplacer l'appel par son résultat) tous ou seulement une partie des appels de service avant l'échange du document. Les critères de ce choix sont multiples et dépendent des contraintes physiques et logiques du système et de l'application :

1. Par exemple, si la bande passante entre les deux pairs est faible, il est préférable de transmettre un appel de service à la place du résultat de cet appel qui a généralement une taille plus importante.
2. Un des deux pair est incapable d'appeler le service pour des raisons de droits d'accès insuffisants.
3. Un des deux pairs ne veut pas appeler des services "inconnus" pour des raisons de sécurité.

Schémas intensionnels

Afin de pouvoir spécifier le *degré de matérialisation* de documents ActiveXML échangés entre des pairs ActiveXML, nous avons étendu le langage XML Schema [TBMM01] avec deux nouveaux types de noeuds pour les appels de service [MAA+03, DN02] :

- Le type *function* permet de décrire un appel de service par les trois attributs mentionnés plus haut.
- Le type *functionPattern* donne la possibilité de créer des "motifs" qui décrivent des appels de service par les types de leurs paramètres et du résultat.

Cette extension est appelé XML Schema$_{int}$ et permet la définition de schémas intensionnels. Par exemple, le motif suivant décrit tous les appels prenant un élément de type `city` en entrée et renvoiant un élément de type `temp` :

```
<functionPattern id="Forecast">
  <params>
    <param> <element ref="city"/> </param>
  </params>
  <result> <element ref="temp"/> </result>
</functionPattern>
```

Les types d'appels de service sont utilisés comme les autres types de noeuds (éléments, attributs, ...) du langage XML Schema. En particulier, ils peuvent faire partie de types complexes comme c'est montré dans l'exemple suivant qui spécifie que le premier élément après la date peut être un appel de service de type `Forecast` *ou* un élément de type `temp` (c.a.d. le résultat de l'appel) :

```
<element name="newspaper">
  <complexType>
    <sequence>
      <element ref="title"/>
      <element ref="date"/>
      <choice>
        <functionPattern ref="Forecast"/>
        <element ref="temp"/>
      </choice>
      <functionPattern ref="TimeOut"/>
  </complexType>
</element>
```

Un document est valide par rapport à ce schéma intensionnel si l'élément de type `date` est suivi (1) d'un appel qui satisfait le motif `Forecast` ou d'un élément de type `temp` et (2) d'un appel de type `TimeOut`. La validation ne concerne pas seulement les types des paramètres, mais également le type du résultat (par exemple, l'opération `Get_Temp` doit retourner un élément de type `temp`).

Validation et réécriture : Le problème que nous nous sommes posé ensuite était de fournir et d'implanter un outil qui ne permet pas seulement de valider un document intensionnel par rapport à un schéma donné, mais également de proposer une liste des appels de service à effectuer pour *rendre un document valide*. Un tel algorithme de validation et de réécriture a été défini [MAA+03] et développé dans le cadre d'un stage DEA que j'ai encadré [DN02]. Formellement, cet algorithme utilise la théorie des automates pour résoudre le problème : un document d et toutes ses réécritures possibles peuvent être représentés par un automate A_d qui peut être comparé à l'automate du schéma A_s. Cette comparaison consiste dans la vérification s'il existe au moins un chemin dans A_d (une réécriture de d) qui est acceptée par A_s. Les détails de l'algorithme peuvent être trouvés dans [MAA+03].

2.2 RDF : Connaissances sur le Web

2.2.1 Le Web Sémantique

Grâce à leur faculté de représenter, d'interroger, de transformer et d'intégrer des documents et des données très diverses, les outils XML sont indispensables pour le partage d'informations sur le Web. Néanmoins XML ne répond que partiellement aux problèmes rencontrés dans le traitement d'informations sur le Web. Cette limite n'est pas d'ordre structurelle ou syntaxique (il est pratiquement possible de représenter toute information structurée en XML), mais concerne le niveau sémantique de l'information manipulée.

Tout traitement d'une donnée ou d'un document XML est fondé sur la compréhension de sa structure et de son contenu. Même s'il est possible de définir très précisément la structure de ce document par une DTD ou un schéma *XML Schema* [TBMM01], une telle description n'est pas toujours suffisante pour effectuer certains traitements. Par exemple, pour interroger le document *van-gogh.xml*, page 19, on suppose que l'utilisateur connaît la signification des balises `Peintre`, `Peinture` etc. mais également de la relation père/fils entre les différents noeuds de l'arbre (un élément fils de type `Peinture` d'un élément `Peintre` est compris comme une peinture créée par le peintre). Cette connaissance est utilisée d'une manière implicite et non-formelle dans la définition du schéma du document, ce qui peut créer des *conflits sémantiques* entre l'intention de l'utilisateur et de l'auteur du document. Par exemple, un utilisateur qui cherche des adresses de peintres industriels sera étonné de retrouver Vincent Van Gogh parmi les personnes qui peuvent repeindre sa maison. A l'inverse, un deuxième utilisateur qui cherche des noms d'artistes en utilisant la balise `Artiste` dans sa requête sera déçu de ne pas trouver Van Gogh.

L'initiative *Semantic Web*[8] du W3C essaie de répondre à ce problème en fournissant les outils nécessaires pour éviter ou résoudre ce type de conflits sémantiques. Parmi ces outils, les ontologies et le langage RDF jouent un rôle central pour la représentation de connaissances et la description sémantique des ressources Web.

Langages d'ontologies

La notion d'ontologie est ancienne et utilisée dans différentes disciplines pour désigner "des connaissances sur des objets". Dans le contexte des systèmes d'information [Gru93] une ontologie est une spécification explicite d'un ensemble de relations entre les instances d'un domaine. Il existe différentes approches pour la définition d'ontologies [Gua97, HVD02] dont chacune permet un certaine interopérabilité sémantique grâce à la possibilité de décrire des concepts et des relations sémantiques entre ces concepts. Les différents choix de représentation sont illustrés par les trois types d'ontologies suivants :

1. Un *thesaurus* est une conceptualisation où l'ensemble des relations entre les concepts décrits par des *termes* est prédéfini. Par exemple, le standard ISO-2788 [ISO86] restreint cet ensemble à 4 types de relations entre termes : (1) une relation de généralisation/spécialisation (broader term generic/narrower term generic), (2) une relation de composition (broader term partitive/narrower term partitive), (3) une relation d'équivalence (use/used-for) et (4) une relation de proximité sémantique (related-term). Ceci signifie qu'un thésaurus ne peut pas prendre en compte des relations spécifiques entre les concepts d'un domaine d'application (on peut dire qu'un peintre est un artiste, une peinture est une oeuvre d'art, mais pas qu'un peintre peint des peintures).
2. Les réseaux sémantiques généralisent la notion de relation entre concepts et donnent la possibilité de modéliser un domaine d'application à plusieurs niveaux d'abstraction. Ils permettent ainsi plus de flexibilité et de puissance dans la description de connaissances avec comme inconvénient une plus

8. http://www.w3.org/2001/sw/

grande complexité dans l'utilisation. Le langage RDF [LS99] est un représentant de ce type de langage d'ontologies (voir section 2.2.1).

3. La traduction des concepts et des relations d'une ontologie en prédicats logiques permet la définition et l'utilisation d'opérations logiques pour décrire et manipuler des connaissances. Par exemple, le langage OWL DL [MvH03] est fondée sur la théorie des *logiques de description* [BCM+02] pour définir des concepts complexes sous forme d'expressions logiques et de raisonner sur ces concepts (voir section 2.2.1).

Un critère important du choix d'un langage ou type d'ontologie est l'équilibre entre les besoins d'une application et la complexité du langage utilisé. Ainsi, bien que les thesaurus soient restreints à une ensemble limité de relations sémantiques entre des concepts, ils sont utilisés dans un grand nombre de domaines et d'applications grâce à leur simplicité et la possibilité d'implanter des traitements efficaces (voir chapitre 3).

On peut distinguer entre deux champs d'applications d'une ontologie dans le contexte du Web. Le premier type d'utilisation exploite surtout la richesse sémantique d'une ontologie pour la découverte et la classement automatique de ressources Web. Généralement, ceci consiste dans l'utilisation de thesauri linguistiques [Mil95, Mah96, KL94] combinés avec des ontologies spécialisées à un domaine d'application précis [TSW03]. Le deuxième champ d'application se sert des ontologies comme référentiel sémantique pour décrire et interroger des sources d'informations. Les thesaurus thématiques décrivant des vocabulaires spécifiques à un domaine d'application (par exemple AAT [AAT] pour le domaine de la culture) et d'autres types d'ontologies créées dans le contexte d'un domaine d'application précis sont généralement exploités dans ce cas.

Pendant ma recherche, je me suis surtout intéressé au deuxième type d'utilisation et plus particulièrement à l'utilisation d'ontologies pour la description et l'intégration de ressources Web.

RDF et OWL

Un objectif majeur de l'initiative *Semantic Web* est la définition de langages pour la description sémantique de ressources Web [LS99, HVD02, DCHM+01, MvH03]. Les métadonnées liées à une ressource Web peuvent être caractérisées par leur dépendance du contenu de la ressource et leur spécificité par rapport à un domaine d'application [She99]. Prenons, par exemple, la ressource *van-gogh.xml*, page 19. La taille de ce document est une métadonnée qui dépend du contenu, tandis que sa date de création en est indépendante. En même temps, ces métadonnées (taille, date de création) ne prennent pas en compte la sémantique de son contenu qui est situé dans le domaine de la culture. Pour comprendre le besoin de ce type de métadonnées sémantiques prenons le cas où un utilisateur cherche des ressources Web concernant la période néo-impressionniste. Une possibilité est d'utiliser un moteur de recherche ou, dans le cas de documents XML, le langage d'interrogation XQuery [CFR+01] pour trouver ces ressources. Indépendamment du langage utilisé, cette interrogation se limite au contenu du document (texte et balises) et ne permet pas d'utiliser d'autres critères de recherche que ceux préconisés par l'auteur du contenu. Par exemple, même si la ressource *van-gogh.xml* concerne cette période, le système ne pourra pas trouver ce document sans connaissances supplémentaires. Cette limite d'une "interrogation par contenu" est encore plus apparente quand on cherche des ressources audio-visuelles (images, son, vidéo) qui ne contiennent généralement aucune information sémantique sur leur contenu.

Pour répondre à ce problème, les moteurs de recherche comme Google ou AltaVista proposent un classement de ressources Web conforme à des hiérarchies de classification comme ODP [odp] dans lesquelles l'utilisateur peut naviguer afin de localiser des ressources. Ainsi, la ressource *van-gogh.xml* (ou une image de ce peintre) aurait pu être placée dans la catégorie *Art→Art History→Movements→Post-Impressionism* de l'ontologie ODP utilisée par Google.

La représentation d'un tel schéma de classification et l'organisation des ressources est possible grâce à un modèle d'ontologies comme celui proposé par la recommandation RDF [LS99] du W3C ou la norme TopicMaps [BBN99]. L'exemple suivant montre la définition RDF de la catégorie `Top/Arts` et de ses sous-catégories dans la hiérarchie ODP [9].

```
<RDF xmlns:r="http://www.w3.org/TR/RDF/"
     xmlns:d="http://purl.org/dc/elements/1.0/"
     xmlns="http://directory.mozilla.org/rdf">

<Topic r:id="Top/Arts">
  <tag catid="2"/>
  <d:Title>Arts</d:Title>
  <narrow r:resource="Top/Arts/Books"/>
  <narrow r:resource="Top/Arts/Music"/>
  <narrow r:resource="Top/Arts/Writing"/>
  ...
</Topic>
...
</RDF>
```

L'élément `Topic` rassemble tous les liens vers les ressources classées dans la catégorie correspondante. Chaque ressource est décrite individuellement dans un élément de type `ExternalPage` (la même ressource peut être classée dans différentes catégories).

```
<RDF xmlns:r="http://www.w3.org/TR/RDF/"
     xmlns:d="http://purl.org/dc/elements/1.0/"
     xmlns="http://directory.mozilla.org/rdf">

<Topic r:id="Art/Art History/Movements/Post-Impressionism">
  <tag catid="2"/>
  <d:Title>Post-Impressionism</d:Title>
  <link r:resource="http://www.vangoghgallery.com/"/>
  <link r:resource="http://www.ibiblio.org/wm/paint/auth/cezanne/"/>
  ...
</Topic>

<ExternalPage about="http://www.vangoghgallery.com/">
  <d:Title>Van Gogh Museum</d:Title>
  <d:Description>The artist's life and times, exhibitions
     and the Museum's collection.</d:Description>
</ExternalPage>
...
</RDF>
```

ODP utilise RDF pour définir un schéma avec des catégories et sous-catégories pour le classement des ressources Web ce qui correspond à la définition d'une sorte de thésaurus avec un nombre fixe de rélations sémantiques entre les catégories. Mais avec RDF, il est possible de créer des schémas plus riches avec des concepts et des rôles spécifiques à un domaine d'application. Par exemple, le fait que le document *van-gogh.xml* «concerne» peintures à l'huile sur toile de la période néo-impressionniste peut être représenté par le document XML suivant utilisant une terminologie provenant du domaine de l'art :

```
<Peinture about="van-gogh.xml">
   <de_la_période>
```

9. Une version RDF complète de ODP est disponible sur http://rdf.dmoz.org/

```
     <Période id='#néo-impressionnisme'/>
   </de_la_période>
   <matériaux_utilisé>huile sur toile</matériaux>
</Peinture>
```

Plus formellement, ce document est une représentation XML d'un graphe de ressources typées et reliées par différents types de propriétés à d'autres ressources ou valeurs. Ainsi, l'interprétation RDF de ce document décrit la ressource *van-gogh.xml* comme une instance de la classe *Peinture* reliée à l'instance *#néo-impressionnisme* de la classe *Période* par une propriété *de_la_période*. La valeur de la propriété *matériaux_utilisé* est la chaîne de caractères *"huile sur toile"*.

Les classes et propriétés d'une description RDF sont définies dans un *schéma de métadonnées* RDFS dont le rôle se situe entre le rôle d'un schéma de base de données et celui d'une ontologie : il permet la structuration des métadonnées et décrit d'une manière explicite des connaissances partagées sur un domaine. Voici un extrait d'un schéma RDFS pour la description précédente :

```
<rdfs:Class rdf:ID="Objet_Iconographique"/>
<rdfs:Class rdf:ID="Peinture">
   <rdfs:subclassOf rdf:resource="#Objet_Iconographique"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Période"/>
<rdf:Property rdf:ID="de_la_période>
   <rdfs:domain rdf:resource="#Objet_Iconographique"/>
   <rdfs:range rdf:resource="Période"/>
</rdf:Property>
<rdf:Property rdf:ID="matériau_utilisé">
   <rdfs:domain rdf:resource="Peinture"/>
   <rdfs:range rdf:resource="rdfs:Literal"/>
</rdf:Property>
```

Le langage RDFS permet de créer des hiérarchies de classes à travers la propriété *rdfs:subclassOf* avec la sémantique orienté-objet standard d'héritage de propriétés et d'inclusion des extensions. Ainsi, toutes les instances de la classe *Peinture* sont aussi des instances de la classe *Objet_Iconographique* et héritent les propriétés de cette classe.

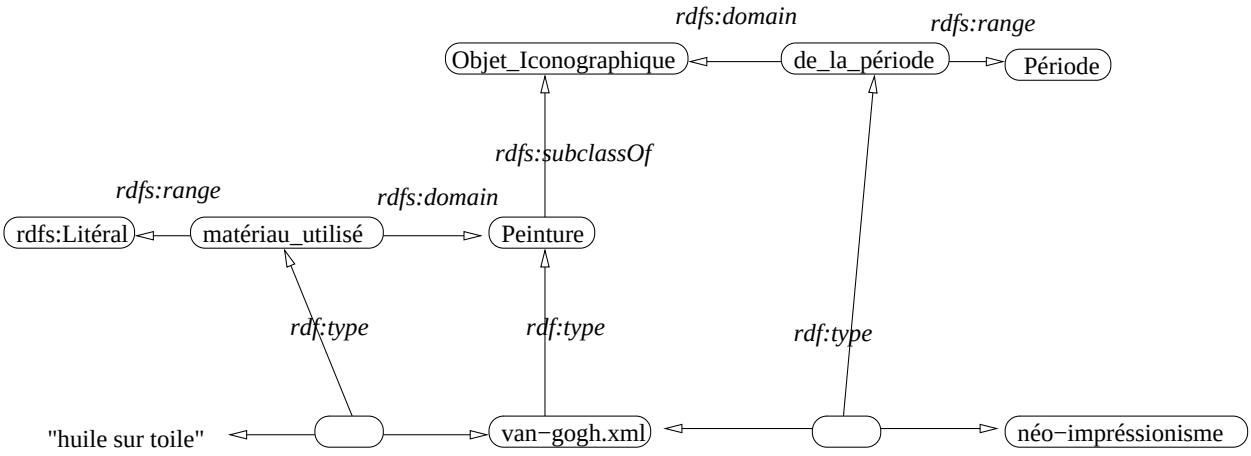

FIG. 2.4 – *Représentation d'un schéma RDFS sous forme de graphe*

L'originalité du modèle RDF se situe dans le fait que tous les composants du modèle (pages Web externes, objets locaux, classes, propriétés, types de propriétés etc.) sont considérés comme des *ressources* ou

instances de la classe *rdfs:Resource*. Ceci donne une grande flexibilité dans la définition d'un métaschéma, mais permet de conclure également que la ressource *rdfs:Class* est une instance de soi-même.

L'intérêt de définir un schéma RDFS n'est pas seulement de pouvoir contrôler la terminologie et la structure des descriptions RDF, mais également d'introduire la possibilité de raisonner sur les liens d'héritage qui existent entre le concepts et les propriété. Ceci est surtout utile quand on veut interroger des métadonnées pour découvrir des ressources. Voici un exemple de requête RQL [KAC+02], qui cherche toutes les ressources sur des objets iconographiques de la période néo-impressionniste :

```
select X
from Objet_Iconographique{X}.de_la_période.{Y}
where Y = "#néo-impressionnisme"
```

Cette requête prend en compte le fait que toutes les instances de la classe *Peinture* sont également des instances de la classe *Objet_Iconographique* et renvoie la ressource *van-gogh.xml*. Ceci rapproche RQL du langage standard OQL [BCD89] avec la possibilité supplémentaire de formuler des *expressions de chemins* qui interrogent en même temps le schéma et les données. Comme exemple, la requête suivante permet de trouver pour la ressource *van-gogh.xml* les classes `$C` dont il est une instance (dans RDF, une ressource peut être l'instance de plusieurs classes), ainsi que les types de propriétés `@P` :

```
select $C, @P
from $C{X}.@P
where X = "van-gogh.xml"
```

Le résultat de cette requête sur notre exemple est la relation montrée dans la figure 2.5 :

$C	**@P**
Peinture	de_la_période
Peinture	matériaux_utilisés

FIG. 2.5 – *Résultat de la requête RQL*

Même si RDF/RDFS permet de créer des graphes conceptuels très complexes, son pouvoir d'expression est trop limité pour certains applications qui ont besoin de raisonner sur les connaissances d'un domaine. Le langage DAML+OIL (March 2001) [10] et son successeur OWL [MvH03] répondent à ce besoin avec des primitives de modélisation plus riches. OWL (ou plus précisément OWL DL) est une extension de RDF/RDFS avec la théorie des *logiques de description* [BCM+02] pour décrire des concepts par des expressions logiques sur des concepts et leurs propriétés. Par exemple, l'expression OWL suivante signifie que toutes les instances de la classe *Peinture* ne sont pas des sculptures (*Peinture* est une sous-classe du complément de la classe *Sculpture*).

```
<owl:Class rdf:about="Peinture">
  <rdfs:comment>une peinture n'est pas une sculpture</rdfs:comment>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:complementOf rdf:resource="#Sculpture"/>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

10. http://www.daml.org/2001/03/reference

Il est également possible de décrire un concept à travers ses propriétés. Ainsi, l'expression suivante décrit les instances de la classe *Peintre* comme des personnes qui ont créé au moins une peinture.

```
<owl:Class rdf:ID="Peintre">
  <rdfs:subClassOf rdfs:resource="#Personne">
    <owl:Restriction>
      <owl:onProperty rdf:resource="#a_créé"/>
      <owl:hasClass rdf:resource="#Peinture"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

La définition de concepts par des expressions logiques permet de calculer des liens d'inclusion (subsomption) entre les concepts et de détecter des incohérences dans les schemas et les descriptions de ressources. Par exemple, si quelqu'un pose une requête qui cherche tous les peintres qui n'ont pas créé de peintures, les système peut déduire que la réponse est vide sans inspecter les données. Ce type de raisonnement est formalisé est dans les logiques de description et joue également un rôle important dans l'intégration sémantique de données [Lev01] (voir chapitre 4).

2.3 Schémas de données et de connaissances

Il est intéressant de comparer plus en détail les rôles de XML et de RDF dans la construction du Web Sémantique. A première vue leurs fonctions semblent bien séparées : XML propose une syntaxe et un modèle de données semi-structurées, tandis que RDF est un modèle de connaissances et de métadonnées qui utilise XML surtout comme syntaxe d'échange. Néanmoins, si on regarde de plus près, l'interaction entre XML et RDF est plus complexe [PSS02]. Afin de mieux comprendre cette interaction, il est intéressant de comparer le rôle d'une ontologie avec celui d'un schéma de données (voir figure 2.3).

Schéma de données	**Ontologie**
structures de données	concepts et relations sémantiques
monde fermé	monde ouvert
types partagés entre applications	conceptualisation partagée entre utilisateurs
évaluation de requêtes et mises-à-jour de données	raisonnement sur les concepts

FIG. 2.6 – *Schémas de données et ontologies*

D'un côté, un schéma de données définit la structure de données (semi-)structurées stockées dans une base de données. Chacune de ces bases de données est un monde fermé où toutes les extensions de classes ou de relations sont connues. Un schéma est partagé entre des applications et souvent invisible pour l'utilisateur. Il joue un rôle important dans l'évaluation de requêtes et les mise-à-jour et sa qualité est souvent mesurée dans ces deux termes (jointures, redondance, anomalies).

De l'autre côté, une ontologie décrit des relations sémantiques entre des objets du monde réel qui est un monde ouvert où les extensions des concepts ne sont pas connues dans leur totalité. Elle est d'abord une terminologie partagée par des utilisateurs pour la description explicite et cohérente de leurs connaissances. La représentation logique des concepts et relations (par exemple, sous forme d'une logique de description) permet également différents degrés de raisonnement.

Malgré cette séparation fonctionnelle il existe une multitude d'exemples qui illustrent des liens d'interaction entre ces deux notions. Un premier exemple est la modélisation conceptuel des bases de données. L'objectif de modèles sémantiques comme IFO [AH87] ou le modèle entité-association consiste dans la

description conceptuelle des informations manipulées dans un SGBD ou un système d'information. Les schémas sémantiques obtenus ressemblent à des ontologies qui représentent les relations sémantiques entre objets en utilisant une terminologie partagée entre les utilisateurs et les concepteurs du schéma. Un deuxième exemple d'interaction entre ontologies et schémas de données est incarné par les schémas de bases de données objet qui donnent la possibilité d'appliquer directement la description conceptuelle d'un domaine à l'organisation et la modélisation des données en définissant des classes avec des attributs et reliés par des rôles/relations et des liens d'héritage. Cette description autorise ensuite un raisonnement fondé sur l'héritage des propriétés (attributs, méthodes et relations) ainsi que l'inclusion entre extensions de classes dans le cas d'un langage de requêtes. La représentation d'ontologies sous forme de schémas orienté-objet permet également l'utilisation de langages de requêtes standards et bien maîtrisés comme OQL pour l'interrogation de métadonnées.

Le troisième lien, qui est le plus important dans le contexte du Web Sémantique, est l'utilisation des ontologies comme schéma d'intégration de données. Le principe d'un tel schéma est de fournir une interface unique pour l'interrogation de sources de données hétérogènes. Il joue un rôle clé pour la formulation de requêtes utilisateurs et ressemble ainsi à un schéma de bases de donnée classique. L'utilisation d'une ontologie comme schéma d'intégration admet la prise en compte de deux contraintes importantes liées à la dimension du Web. Premièrement, les sources Web à intégrer ne sont pas toujours connues au moment de la conception du schéma. Une ontologie décrit un domaine d'application indépendamment de l'information disponible et sa "traduction" en schéma permet de créer facilement une point de départ bien-fondé pour la description et l'intégration des sources. Deuxièmement, il est difficile à prévoir toutes les utilisations d'un schéma d'intégration qui servira non seulement pour la programmation d'applications, mais également comme interface d'interrogation pour les utilisateurs. Les ontologies représentent des concepts et des relations partagés par une communauté d'utilisateurs et facilitent ainsi la compréhension du schéma d'intégration généré.

Dans les deux chapitres suivants, nous mettons en oeuvre ce double rôle des ontologies – description de connaissances et schémas de données – pour la création d'entrepôts de métadonnées et d'intégration de ressources XML.

Chapitre 3

Entrepôts de métadonnées

Ce chapitre présente nos résultats concernant la définition et l'implantation d'entrepôts de métadonnées sémantiques. Cette activité a été menée dans le cadre d'un contrat avec le Ministère de la Culture et de deux contrats Européens (C-Web et MesMuses). Elle a également donné lieu à plusieurs publications dans des conférences et journaux nationaux et internationaux [AF99, AFS00, ACK+00] et des collaborations avec l'institut FORTH et l'INRIA Rocquencourt.

Le chapitre est organisé en trois sections. Dans la première section je présente brièvement les fonctionnalités principales d'un entrepôt de métadonnées que nous avons défini dans le cadre du projet C-Web [cwe99]. La section suivante décrit notre approche de création de schémas de métadonnées [AF99, AFS00] et je terminerai le chapitre avec la description de l'entrepôt de métadonnées Eliot [Rad00] qui utilise une base de données orienté-objet et le langage OQL pour le stockage et l'interrogation de descriptions de ressources [AFS00]. Un point important étudié dans le cadre de ce prototype concernait l'optimisation de requêtes sur des hiérarchies de concepts (thésaurus) [Laf00].

3.1 L'entrepôt de métadonnées C-Web

L'intention du projet C-Web (voir aussi section 1.3.2, page 10) était de spécifier une plate-forme qui fournit à des communautés d'utilisateurs ou *communautés Web* un ensemble de services qui facilitent la publication et l'échange d'informations sur le Web. Ces services sont fondés sur le fait que chaque communauté partage non seulement des ressources d'information mais également des ontologies qui peuvent être utilisées pour décrire et organiser ces ressources.

3.1.1 Le modèle C-Web

Nous distinguons trois composants fondamentaux d'une plate-forme C-Web [AM01] :

1. un ensemble de *ressources Web* partagées par une communauté d'utilisateurs;
2. une descriptions des connaissances d'un domaine sous forme d'une *ontologie*;
3. un *ensemble de métadonnées* qui décrivent les ressources dans les termes de cette ontologie;
4. un *langage de requêtes* qui permet de formuler des requêtes structurées sur les descriptions (métadonnées) des ressources.

L'ensemble de ces quatre composants est décrit par un *modèle d'entrepôt de métadonnées*. Le modèle d'entrepôt C-Web est fondé sur la notion de ressource RDF (voir section 2.2.1) pour la représentation des métadonnées associées à des ressources Web.

Contrairement à RDF, le modèle C-Web fait la distinction entre les ressources internes (descriptions) et les ressources externes (ressources Web) qu'elles décrivent. Une deuxième différence avec RDF est qu'un schéma C-Web est séparé en *schéma applicatif* qui décrit l'ontologie d'une application spécifique et un *schéma documentaire* qui permet d'associer des informations facilitant la gestion des ressources externes. Toutes les instances du schéma applicatif sont des ressources internes appelées *surrogats*. Les ressources Web externes sont des instances du schéma documentaire et reliées aux surrogats grâce à une propriété *concerne* (il est possible d'ajouter d'autres types de propriété plus précises si nécessaire). Ainsi, une ressource concernant le peintre Van Gogh n'est pas considérée comme une instance de la classe RDFS *Peintre*, mais comme une instance de la classe prédéfinie *RessourceWeb* qui est relié par une propriété *concerne* à une instance du concept *Peintre* (*surrogat*). La figure 3.1 illustre l'organisation des ressources dans le modèle C-Web (la définition formelle du modèle C-Web peut être trouvée dans [AM01]).

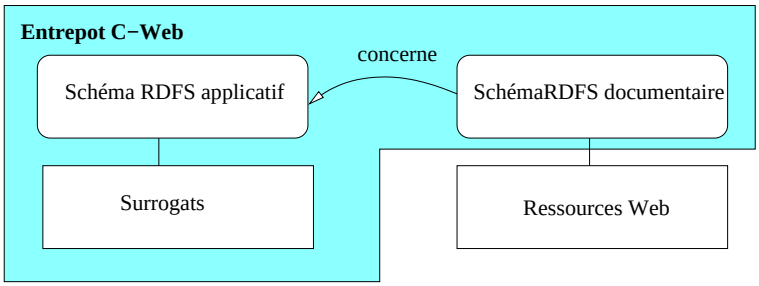

FIG. 3.1 – *Modèle de données C-Web*

La séparation entre ces différents types de ressources a deux avantages. Premièrement, il est naturel de faire la distinction entre une ressource Web et l'objet qu'elle décrit. En particulier, ceci permet d'associer différentes ressources au même objet. Deuxièmement, la séparation en deux catégories de ressources (internes et externes) permet plus facilement d'associer des traitements différents aux ressources Web et aux métadonnées qui les décrivent (voir section suivante).

3.1.2 Fonctionnalités et architecture C-Web

Dans le cadre du projet C-Web nous avons identifié et étudié un ensemble de services nécessaires pour l'utilisation et la maintenance d'un entrepôt de métadonnées [Chr00]. Ces services peuvent être classés par rapport à trois tâches fondamentales :

1. La première tâche concerne la *création du schéma de métadonnées de l'entrepôt*. L'acquisition et la représentation des connaissances partagées par une communauté Web est une tâche clé dans la construction d'un entrepôt de métadonnées. Elle implique typiquement la collaboration d'experts du domaine assistés par des spécialistes dans la représentation de connaissances. Notre approche à cette problématique sera décrite plus en détail dans la section 3.2.
2. La deuxième tâche concerne la publication de ressources à travers la *la création des métadonnées*. Une difficulté de cette tâche est de réduire l'effort demandé à l'utilisateur pour la description de ressources ou de collections de ressources dans les termes du schéma de l'entrepôt. Un deuxième problème est soulevé par l'autonomie des ressources et la nécessité de garantir un minimum de cohérence entre leur évolution et l'état de l'entrepôt.
3. La troisième tâche concerne l'*exploitation des métadonnées* à travers des outils de navigation et d'interrogation. Le langage de requête y joue un rôle primordial. Il doit permettre la formulation de requêtes

avec une connaissance incomplète du schéma et, plus particulièrement, l'interrogation simultanée du schéma et des données.

Un ensemble d'outils pour accomplir ces trois types de tâches a été implanté dans le cadre du projet Européen MesMuses autour de l'entrepôt de métadonnées RDFSuite développé par ICS/FORTH. Parmi ces outils nous pouvons mentionner un *éditeur de schémas C-Web*, un *éditeur de métadonnées* et un *générateur de portails*. Un aspect intéressant concernant l'édition des métadonnées est l'utilisation du protocole Web-DAV [web] pour gérer les ressources et leurs descriptions. Grâce à ce protocole il est possible de stocker les ressources avec leurs métadonnées sur des serveurs Web standard ce qui permet de garantir une certaine cohérence entre les ressources et leurs descriptions tout en gardant leur autonomie. Un protocole d'acquisition de métadonnées adapté a été spécifié et permet le rafraîchissement de l'entrepôt d'une manière synchrone (par notification de l'éditeur) ou d'une manière asynchrone (par souscription des serveurs WebDAV). Une description plus détaillée de ce protocole et des autres outils peut être trouvée dans [AM01].

3.2 Construction de schémas de métadonnées

La construction d'un schéma de métadonnées est une tâche difficile et demande la collaboration entre des experts du domaine qui sont souvent assistés par des spécialistes dans la représentation de connaissances. Une façon de réduire cet effort consiste dans la définition d'ontologies standards pour différents types d'applications. Comme exemples on peut citer l'ontologie ICOM/CIDOC [ICO] pour la description d'objets culturels ou le Dublin core [dub] pour la gestion documentaire. Ces ontologies représentent généralement des connaissances partagées par un grand nombre d'utilisateurs, mais doivent souvent être adaptées pour répondre aux besoins d'une application spécifique. L'adaptation devient possible grâce à des modèles standardisés pour la représentation et l'échange d'ontologies [MvH03, DCHM+01, Rou02].

Notre approche pour la création de schémas de métadonnées est fondée sur le principe d'intégration d'ontologies existants [MFR+00, MBDH02, TCS01]. L'originalité de notre proposition est que nous considérons la description d'un domaine d'application comme une composition d'un *schéma conceptuel* et d'un ensemble de *terminologies spécialisées* pour les différents concept de ce schéma. Afin d'illustrer notre approche nous nous servirons d'un exemple concret du domaine de la culture (les détails avec les définitions formelles sont décrits dans [AF99, AFS00]).

3.2.1 Schémas conceptuels et thésaurus

La figure 3.2 représente un schéma conceptuel pour la description d'objets culturels qui est inspiré d'une ontologie publiée par le Comité International pour la Documentation du Conseil International des Musées (ICOM-CIDOC) [1]. Le schéma est représenté sous forme d'un graphe dont les nœuds correspondent à des concepts et des types de valeurs et les arcs décrivent des rôles, des attributs et des liens d'héritage (isa) entre des concepts. Le schéma contient les neufs concepts *Personne*, *Artiste*, *Activité*, *Artefact*, *Objet Iconographique*, *Musée*, *Période*, *Style* et *Image*. Les concepts sont reliés par des rôles, tandis que les attributs partent toujours d'un concept vers un type de valeur. Par exemple, une personne a un nom (attribut *nom* de type *String*) et effectue des activités (rôle *effectue* vers le concept *Activité*) afin de produire des artefacts (rôle *produit* vers le concept *Artefact*). Chaque arc qui correspond à un rôle est étiqueté par le nom du rôle et le nom du rôle inverse en parenthèses. Les liens d'héritage sont représentés par des arcs en pointillé. Par exemple, les objets iconographiques (concept *Objet Iconographique*) sont des artefacts auxquels on peut associer un *Style*.

1. http://www.willpowerinfo.myby.co.uk/cidoc/

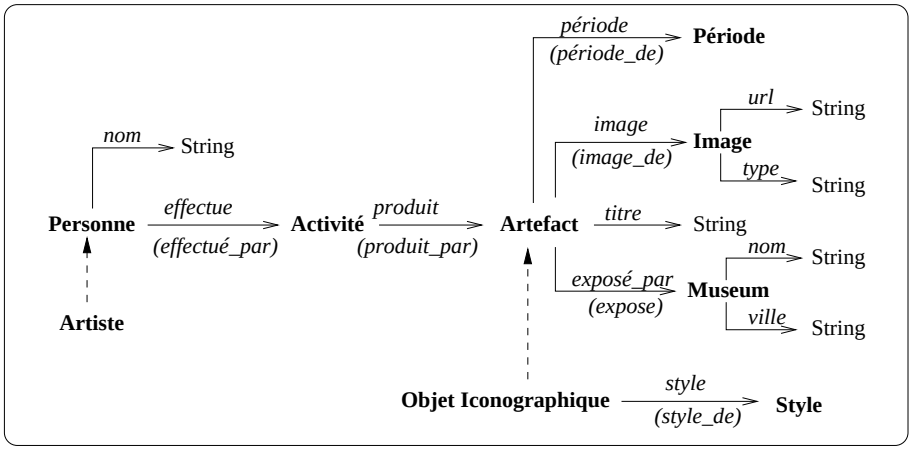

FIG. 3.2 – *Un schéma conceptuel pour la description d'objets culturels*

Un extrait d'un thésaurus de styles et périodes artistiques est montré dans la figure 3.3. Ce thésaurus est mono-hiérarchique (chaque terme, excepté la racine, a exactement un parent) et organise les styles et les périodes artistiques par leur région (sous-hiérarchie *<styles and periods>*) et en styles issus de mouvements internationaux après 1945 (sous-hiérarchie *<international post-1945 styles and movements>*). Les termes entre les symboles '<' et '>' ne désignent pas directement un style ou une période, mais servent à clarifier l'organisation de l'hiérarchie (termes structurants ou *guide terms*).

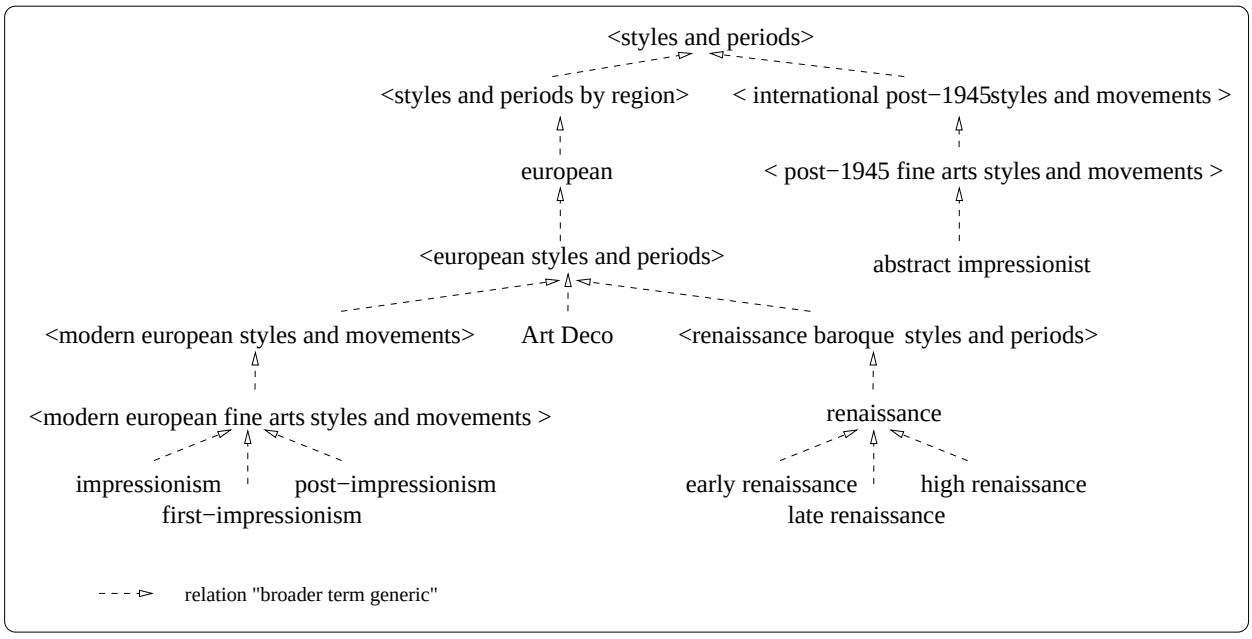

FIG. 3.3 – *Extrait du thésaurus* Art & Architecture Thesaurus

Les thésauri représentent une forme d'ontologie très largement utilisée pour l'organisation, l'indexation et la recherche de documents. Ainsi les thésaurus sont généralement développés et proposés par les grands organismes responsables de l'archivage ou la documentation, comme par exemple les systèmes de classi-

fication Library of Congress Subject Headings [2], ACM Computing Classification System [3], *Unified Medical Language System* [4] and the *Art & Architecture Thesaurus* [5] pour le domaine culturel. Le standard ISO 2788 [ISO86] définit en tout cinq types de relations (généralisation/spécialisation, synonymie, instantiation, partition et proximité sémantique) entre les termes d'un thésaurus :

- Généralisation (broader term generic - *btg*) : La relation *btg* est la plus utilisée dans les thésaurus existants et interprétée comme une inclusion entre les ensembles de documents indexées par les termes correspondants. Elle permet la création de *taxonomies*, i.e. d'hiérarchies de termes reliés par des relations de généralisation/spécialisation (voir figure 3.3).
- Synonymie (used for term - *uf*) : La relation de synonymie permet d'associer un terme à ses synonymes [6]. Cette relation est surtout utile pour l'intégration d'ontologies qui ont été développées indépendamment dans différents communautés ou pays. Un exemple du domaine de la culture est le terme *art nouveau* qui est aussi appelé *modern style*, *jugendstil*, *secession*, *modernissmo* etc.
- Instantiation (broader term - *bt*) : La relation d'instanciation permet de créer des hiérarchies d'abstraction qui considèrent un terme comme le nom d'une instance du concept désigné par son terme parent. Par exemple, la relation $bt('VanGogh','peintre')$ considère la personne désigné par le terme *Van Gogh* comme une instance du concept *peintre*.
- Partition (part-of (broader term partitive - *btp*) : Cette relation permet de spécifier qu'un terme désigne un objet qui fait partie de l'objet désigné par l'autre terme. Par exemple, la relation $btp('porte','maison')$ décrit une porte comme une partie d'une maison.
- Contexte sémantique (related term - *rt*) : Il est difficile de donner une sémantique formelle (e.g. ensembliste) à la relation *rt* qui sert à décrire le "contexte sémantique" d'un terme. Par exemple, afin de placer l'art moderne dans son contexte historique on peut créer une relation *rt* entre le terme *art moderne* et le terme *première guerre mondiale*. Cette relation montre les limites du pouvoir d'expression d'un thésaurus et la complémentarité des schémas conceptuels dans la description d'un domaine.

3.2.2 Intégration de schémas conceptuel et de thésauri

Les schémas conceptuels et les thésaurus sont deux types d'ontologies complémentaires pour décrire des connaissances d'un domaine. Une schéma conceptuel fournit une description *structurée, générique et partageable sous forme d'un graphe de concepts et de rôles* tandis qu'un thésaurus contient un *grand ensemble de termes avec une sémantique précise mais faiblement structurés*.

L'intégration contrôlée de ces deux types de structures sémantiques est possible grâce à une interprétation ensembliste des concepts d'une ontologie et des termes d'un thésaurus. Ainsi, nous considérons un terme de thésaurus comme une une référence vers un ensemble d'objets et la relations entre les termes comme relations entre des ensembles.

Nous avons choisi de restreindre notre approche de construction de schémas de métadonnées à l'utilisation de taxonomies définies par la relation *btg* :

- L'interprétation ensembliste de cette relation est bien-définie et très utile pour l'interrogation des métadonnées, car elle permet d'introduire un minimum de raisonnement contrôlé pendant l'évaluation d'une requête utilisateur. Par exemple, une recherche de ressources concernant la renaissance (mot

2. http://www.grci.com/services/library/libcongress/index.shtml
3. http://www.iicm.edu:8080/jucs_classification
4. http://gmedserv.nlm.nih.gov/research/umls
5. http://www.ahip.getty.edu/vocabulary/aat_intro.html
6. Le standard ISO 2788 sépare les termes d'un thésaurus en *descripteurs* et en *non-descripteurs*. Toutes les relations sauf la relation *uf* sont définies entre descripteurs, tandis que la relation *uf* relie des descripteurs à des termes non-descripteurs qui peuvent être utilisés comme synonymes.

clé de recherche: *renaissance*) permet également de trouver des documents indexés par les deux sous-termes *early renaissance, late renaissance* et *high renaissance*.

– Les hiérarchies de classification créées par la relation *btg* représentent la plus grande majorité des thésaurus utilisés aujourd'hui et s'intègrent naturellement avec la relation isa utilisés dans les schémas conceptuels.
– La relation *btg* (avec la relation d'équivalence *uf*) a une sémantique précise et peut facilement être intégrée dans la sémantique d'un langage de requêtes.

Nous suivons une approche à deux étapes pour la création de schémas de métadonnées. La première étape consiste dans le choix d'un schéma conceptuel qui décrit les concepts principaux du domaine et les relations qui existent entre ces concepts. Le schéma résultant est une description générique et contient généralement quelques dizaines de concepts reliés par des rôles. Dans la deuxième étape on choisit un certain nombre d'hiérarchies de classification qui peuvent être associées aux différents concepts du schéma. Cette association est définie sous forme d'une *relation de connexion* entre des termes d'un thésaurus et des concepts.

Une relation de connexion entre l'hiérarchie *Styles & Periods* de la figure 3.3 et les deux concepts Style et Period du schéma conceptuel de la figure 3.2 est montrée dans la figure 3.4

Term	Concept
impressionism	*Style*
post-impressionism	*Style*
abstract impressionism	*Style*
renaissance	*Style*
early renaissance	*Style*
first-impressionism	*Style*
late renaissance	*Style*
high renaissance	*Style*
renaissance	*Period*
early renaissance	*Period*
late renaissance	*Period*
high renaissance	*Period*

FIG. 3.4 – *Relation de connexion entre l'hiérarchie* Styles & Periods *et les concepts Style et Period du schéma conceptuel.*

Un couple $[t,c]$ dans cette relation crée une relation de généralisation entre le terme *t* et le concept *c* : l'ensemble d'objets désigné par le terme *t* est un sous-ensemble des instances désigné par le concept *c* et hérite tous les rôles de ce concept. On remarquera qu'une relation de connexion permet de connecter uniquement une partie des termes d'un thésaurus aux différents concepts du schéma conceptuel (dans l'exemple précédent nous n'avons pas connecté tout le thésaurus *Styles & Periods* aux concepts *Style* et *Period*). Nous avons adopté cette approches sélective pour plusieurs raisons. Une première raison évidente est que certains termes sont en dehors du domaine d'application qui doit être décrit par le schéma de métadonnées. Par exemple, si une application est uniquement concernée par la peinture, la relation de connexion ne prendra pas en compte les termes *sculpture* et *drawings*. Une autre raison pour choisir les termes explicitement est l'existence de de termes structurants (guide terms) qui n'ont pas de fonction descriptive, mais servent uniquement à organiser le thésaurus (par exemple <*international post-1945 styles and movements*>). Finalement, un thésaurus peut contenir des termes qui peuvent être associés à plusieurs concepts. Par exemple, les termes de l'hiérarchie *Styles & Periods* figure 3.3 décrivent des styles (e.g. *impressionism*), des périodes ou des styles et periods

(e.g. *renaissance*). La relation de connexion permet aux spécialistes de clarifier les différentes sémantiques d'un tel terme homonyme désignant deux concepts différents.

Après avoir défini la relation de connexion entre un ensemble de taxonomies et un ensemble de concepts d'un schéma conceptuel, il est possible d'extraire pour chaque concept une *taxonomie locale* qui permet de classer les instances de ce concept. Cette extraction correspond à une opération de "saturation" qui garantie que la terminologie d'un concept inclue la terminologie de tous ses sous-concepts. Ceci garantie qu'un terme qui décrit des instances d'un concept c' est également un terme du thésaurus local de tous les super-concepts de c'.

La figure 3.5 montre un exemple d'extraction de taxonomies locales. Les termes t, v et w sont connectés aux concepts c, d et e. L'opération d'extraction construit pour le concept c le thésaurus $\mathcal{T}_c$ qui contient le terme t (provenant de la relation de connexion) mais également les termes v et w qui sont connectés aux sous-concepts de c. On remarquera que le terme u a disparu, toute en gardant la relation de généralisation entre les termes w et t.

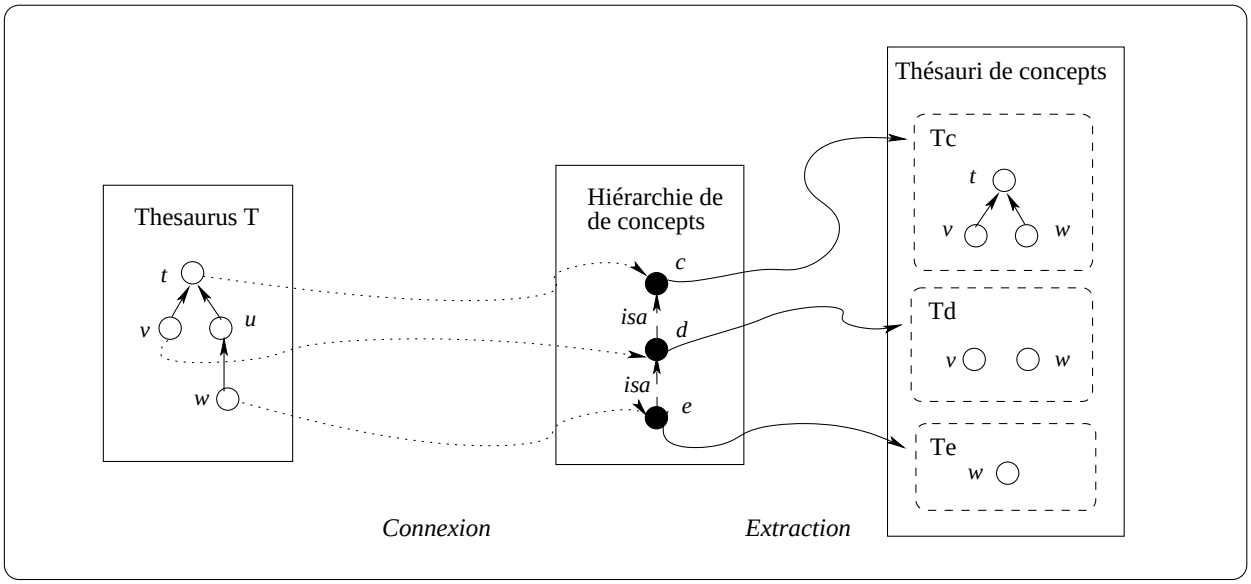

FIG. 3.5 – *Extraction de thésaurus*

Résumé Le premier résultat de notre modèle d'intégration d'ontologies est une méthodologie simple et modulaire pour la création de schémas de métadonnées. Cette méthodologie a été mise en pratique dans le cadre de deux contrats de recherche (le projet Européen IST MesMuses et un contrat avec le Ministère de la Culture). Notre expérience a montré que le découpage de la conception d'un schéma de métadonnées en deux étapes successives – définition du schéma conceptuel suivie de la création de thésaurus locaux pour certains concepts – facilite la tache de conception du schéma pour les utilisateurs finaux.

Un deuxième résultat de notre approche est la possibilité de générer et d'interroger des métadonnées précises en utilisant des termes de thésauri spécialisés. Ces métadonnées sont stockées dans un entrepôt qui fournit un point d'accès unique aux ressources publiées. L'utilisateur pose des requêtes dans les termes du schéma de métadonnées (schéma conceptuel et thésauri) et le système sélectionne les adresses des ressources dont la description correspond à la requête posée. Un tel entrepôt développé au dessus du SGBD orienté-objet O_2 sera décrit dans la section suivante.

3.2.3 Description de ressources

Nous avons montré qu'il est possible de traduire nos schémas de métadonnées en schémas RDFS [AFS00]. Dans cette traduction les taxonomies locales sont transformées en hiérarchies de classes RDFS. L'avantage d'une telle traduction est que les langages d'interrogation RDF comme RQL [KAC+02] prennent automatiquement en compte la relation `rdfs:subclassOf` dans l'interprétation d'une requête. Bien que cette traduction soit sémantiquement correcte, elle pose un problème en terme de performance : le coût de l'interprétation de la relation d'inclusion entre des classes RDFS augmente considérablement quand le schéma contient des hiérarchies de classes profondes.

Une possibilité pour diminuer le coût de l'évaluation d'une requête est de séparer l'hiérarchie des concepts de leurs thésauri et d'implanter des algorithmes spécifiques pour l'interprétation de la relation *btg* entre les termes. Notre modèle de description de ressources est fondé sur une telle séparation. Chaque concept c est représenté par une classe du même nom tandis que chaque thésaurus local d'un concept c est traduit en un type T_c qui rassemble tous les termes du thésaurus (si le concept c n'a pas de thésaurus local, l'extension de T_c est vide). Nous distinguons trois types de propriétés pour un classe c :

- Chaque classe a une propriété *terme* de type T_c.
- Une rôle r entre un concept c et un c' est un attribut de type c'.
- Tous les attributs de concepts sont directement traduits en attributs du même type.

Une description est un couple $[u,d]$ où u est l'URL de la source et d est une instance d'une classe c. Voici une description (sous forme RDF) de la ressource *http://www.marmottan.com/images/oeuvres/nympheas.jpg* qui est une reproduction des "Nymphéas" de Claude Monet :

```
<Artefact about="http://www.marmottan.com/images/oeuvres/nympheas.jpg"
  <terme>Peinture</terme>
  <titre>Nymphéas</titre>
  <style>
     <Style>
        <terme>impressionisme</terme>
     </Style>
  </style>
  <produit_par>
     <Activité>
        <effectué_par>
           <Peintre>
              <nom>Monet</nom>
           </Peintre>
        </effectué_par>
     </Activité>
  </produit_par>
</Artefact>
```

La description est une instance de la classe *Artefact*. L'attribut *terme* a comme valeur le terme *Peinture* de la taxonomie locale associée à ce concept. Le titre de l'artefact est un attribut de type `String`. L'objet est relié à un objet de type *Style*(avec un attribut *terme* indiquant le style de l'objet) et une instance de la classe *Peintre*.

Un entrepôt de métadonnées est un ensemble de descriptions de ressources publiées à un instant donné. Chaque ressource peut être décrite par plusieurs descriptions utilisant différents concepts du schéma de métadonnées. Nous appliquons l'interprétation classique de la relation classe/sous-classe (héritage d'attributs et inclusion ensembliste) aux descriptions avec une seule particularité qui provient de la sémantique associée aux hiérarchies de termes des thésauri locaux : si o correspond à une instance de la classe c spécialisé par

une terme t ($o.term = t$), alors pour tous les termes plus génériques t' dans le thésaurus de c, l'objet o' avec $o'.term = t'$ appartient également à l'extension de c. Ainsi, par exemple, si une source u est concerne une peinture à l'huile, alors elle décrit également une peinture.

3.3 Un entrepôt de métadonnées orienté-objet

3.3.1 Le prototype ELIOT

Le prototype ELIOT [Rad00] a été développé dans le cadre d'un contrat entre le Conservatoire National des Arts et Métiers (CNAM) et la Direction de L'Architecture et du Patrimoine (DAPA) du Ministère de la Culture français. Son architecture est montrée dans la figure 3.6.

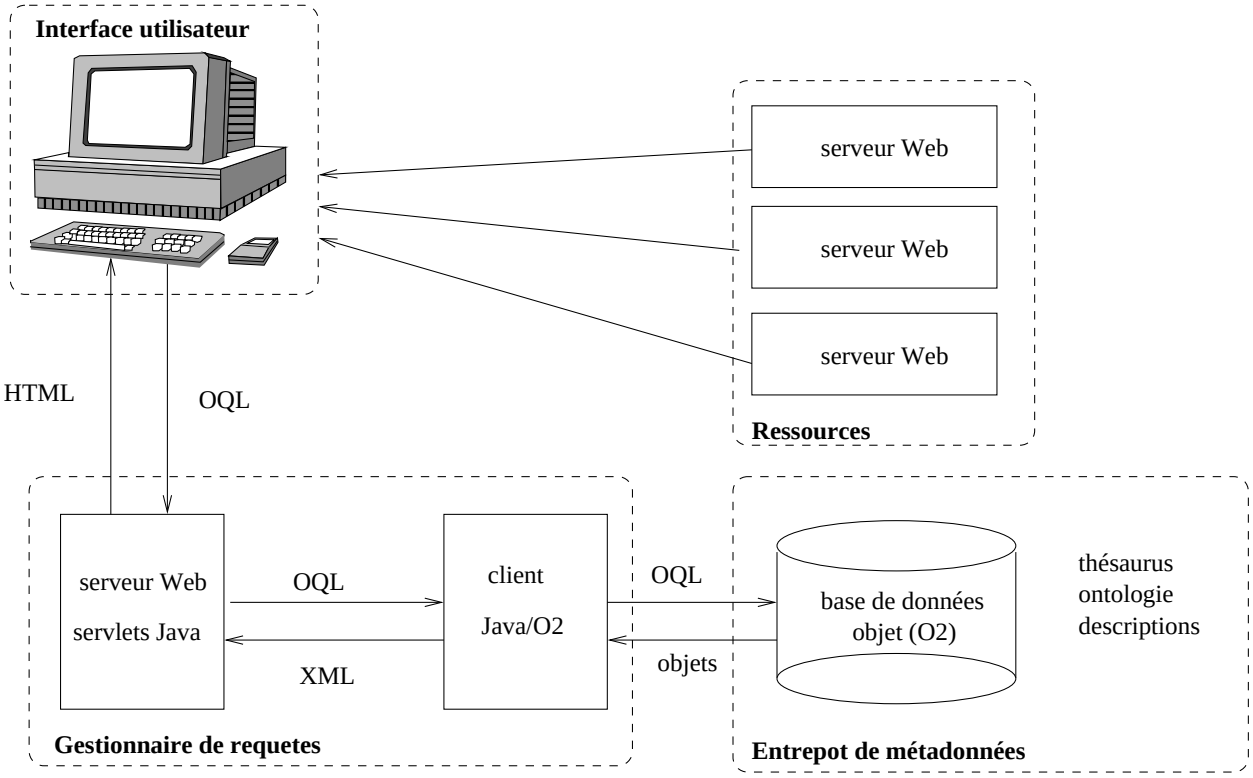

FIG. 3.6 – *Architecture du prototype ELIOT*

Le système est composé de trois modules : (i) un entrepôt de métadonnées implanté sous forme d'une base de données O_2 [FCP92], (ii) un gestionnaire de requêtes et (iii) une interface utilisateur Web pour la consultation des métadonnées et des ressources Web. Le gestionnaire de requêtes est composé d'un client de l'entrepôt qui communique avec un serveur Web à travers le protocole Java-RMI. Il transmets les requêtes de l'interface utilisateur au serveur O_2 et traduit les résultats retournés en HTML.

La figure 3.7 montre un formulaire d'interrogation pour la recherche de ressources sur des objets d'études. L'utilisateur cherche des ressources documentaires concernant des objets d'études de type *architecture religieuse*. L'interface est composé de deux parties : la partie de gauche affiche un formulaire qui permet de saisir des valeurs d'attributs du concept Objet_d_etude. Certains attributs prennent leurs valeurs dans un thésaurus. Un terme du thésaurus relié à l'attribut *dénomination* est montré dans dans la partie de

droite. L'utilisateur peut choisir le terme affiché ou élargir (restreindre) la requête en naviguant vers d'autres termes plus généraux (spécifiques). Le résultat de la requête est illustré dans la figure 3.8.

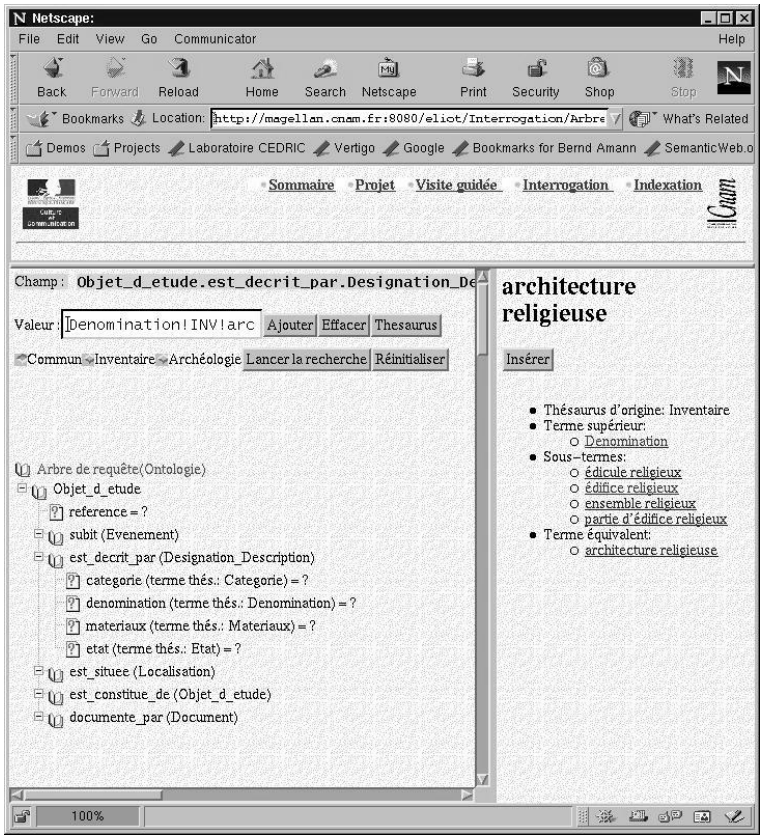

FIG. 3.7 – *Interface d'interrogation de métadonnées*

3.3.2 Évaluation de requêtes

L'entrepôt implante notre modèle de description décrit dans la section 3.2.3. L'introduction de terminologies dans les schémas de métadonnées ne permet pas seulement la création de métadonnées plus riches, mais augmente également le coût d'évaluation des requêtes qui doivent prendre en compte la relation de généralisation entre les termes. Nous avons étudié ce problème dans le dans le cadre du prototype ELIOT qui permet l'interrogation de métadonnées avec le langage standard OQL [Clu98]. Le schéma de métadonnées est traduit en schéma orienté-objet et l'entrepôt est une instance de ce schéma. Cette traduction inclue la traduction du thésaurus local de chaque concept et influe directement l'évaluation des requêtes.

Parcours de thésaurus

Une première solution naïve pour évaluer des requêtes thésaurus est d'effectuer un parcours de l'arborescence de termes du thésaurus. Cette arborescence est implantée sous forme d'un ensemble d'objets dont chacun contient le terme sous forme d'une chaîne de caractères, l'identificateur de son père et l'ensemble de ses enfants. Le parcours du thésaurus est implanté sous forme d'une méthode $tree(Term) \rightarrow set(Term)$

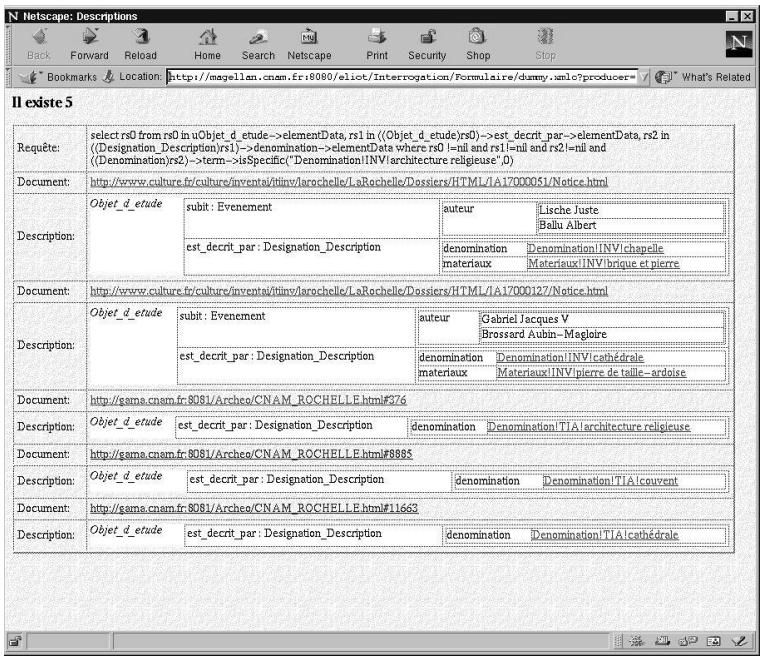

FIG. 3.8 – *Résultat d'une requête*

qui retourne pour un terme t donné, l'ensemble des termes plus spécifiques dans le thésaurus associé à la classe correspondante [7].

Grâce à la possibilité d'appeler des méthodes dans des requêtes OQL, on peut ensuite utiliser la méthode $tree$ dans la condition de sélection de la requête pour vérifier si une description concerne un terme donné. Par exemple, la requête suivante retourne tous les URLs de sources concernant des artistes :

Exemple 3.1

```
SELECT d.url
  FROM d IN Personnes
 WHERE d.term IN d.tree("artiste")
```

La racine de persistance `Personnes` contient toutes les descriptions concernant des personnes. Ces descriptions sont des instances de la classe *Personne* qui possède un thésaurus local avec différents métiers. La méthode $tree$ cherche d'abord le noeud (le terme) avec l'étiquette *artiste* et collectionne ensuite tous les descendants de ce noeud (les sous-termes) par un parcours récursif en profondeur de l'arborescence.

L'optimiseur de O_2 ne prend pas en compte les méthodes utilisées dans une requête OQL. En particulier, il n'est pas possible d'exploiter des index dans le corps de la méthode (par exemple pour comparer deux termes). En plus, chaque occurrence de la méthode $tree$ dans la clause `WHERE` génère *à chaque comparaison* le même ensemble de tous les sous-termes d'un terme donné. Cette ensemble peut devenir relativement gros si l'argument se trouve en haut d'une hiérarchie de plusieurs milliers de termes. Ainsi, même si l'interrogation de ces descriptions ne demande qu'une forme limitée de déduction sur la relation de généralisation entre les termes d'une taxonomie, cette facilité devient un problème central dans l'interrogation d'entrepôts utilisant des thésaurus avec plusieurs milliers de termes.

Un meilleur résultat peut être obtenu par une extension du modèle de données avec un nouveau type *thésaurus* (des extensions similaires ont été faites pour l'indexation plein texte de documents dans

7. La traduction exacte de notre modèle de description dans le modèle O_2 peut être trouvée dans [Laf00]

O2 [ACC+97]). Une autre solution est de choisir une représentation des thésaurus qui peut être interprété par l'optimiseur. Une telle solution est l'utilisation de techniques de *linéarisation d'arbres* pour traduire le domaine hiérarchique défini par le thésaurus en un ou plusieurs domaines linéaires.

Linéarisation de Thesaurus

Les techniques de linéarisation d'arbres ont été développées et étudiées dans différents contextes d'application et surtout pour l'évaluation de requête "ancêtre-descendants" sur des documents XML [CKM02, BKS02, Gru02, ZND+01]. L'idée est de transformer le parcours d'une arborescence en un test d'appartenance à une intervalle qui peut être évalué efficacement sans étendre le langage de requêtes.

Dans le contexte de notre application, la linéarisation d'un thésaurus consiste dans la définition d'un schéma d'étiquetage $label$ sur les termes de thésaurus qui permet de vérifier si un terme t est un sous-terme d'un autre terme t' par une simple comparaison de leurs étiquettes $label(t)$ et $label(t')$.

Différents schémas d'étiquetage ont été proposés dans la littérature. Un état de l'art récent peut être trouvé dans [CPST02]. Dans [AFS00] nous avons étudié le schéma d'étiquetage fondé sur le principe de la classification de Dewey [Dew94]. Ce schéma utilise la relation *préfixe* entre les étiquettes comme moyen de définition d'un ordre partiel entre des termes : un terme t est un ancêtre d'un autre terme t', si l'étiquette $label(t)$ est un préfixe de l'étiquette $label(t')$.

Plus formellement, on définit une étiquette comme un mot sur un alphabet A. Le nombre de symboles définit le degré maximum de l'arbre (le nombre maximal d'enfants par noeud). Un exemple d'arbre linéarisé avec l'alphabet $\{a,b,c\}$ est montré dans la figure 3.9.

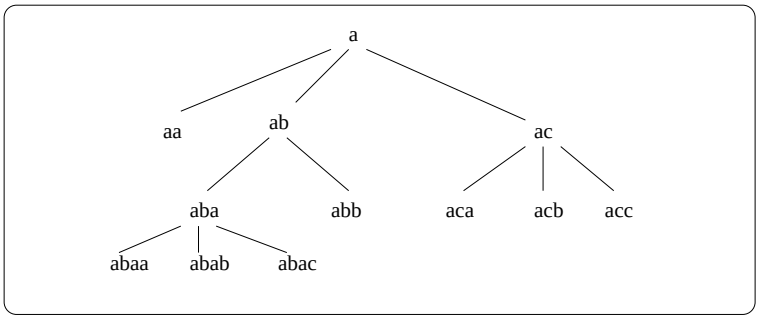

FIG. 3.9 – *Code Dewey*

Pour chaque noeud t, la longueur de l'étiquette $label(t)$ indique la profondeur du noeud dans l'arbre. L'ordre lexical entre les étiquettes correspond à l'ordre de parcours en profondeur de gauche à droite. Cette propriété permet de traduire un test de descendance entre deux termes t et t' en prédicat logique sur les étiquettes des termes : le terme t' est un descendant de t ssi $label(t')$ est dans l'intervalle $[label(t),next(t)[$ (où $next(t)$ est l'étiquette du frère juste à droite du terme t).

La traduction de la requête de l'exemple 3.1 utilisant ce schéma de codage est montré dans l'exemple suivant :

Exemple 3.2

```
SELECT d.url
  FROM d IN Personnes,
       t IN LPersonnes
 WHERE t.term = "Artiste"
   AND d.label >= t.label
   AND d.label <= t.next;
```

Dans cette requête, nous supposons que pour chaque thésaurus local d'un concept c il existe une racine de persistance Lc qui contient pour chaque terme t un n-uplet $[term : t, label : l, next : n]$ où l est l'étiquette de t et n est l'étiquette du frère droit de t (si t est le dernier noeud de son niveau, n a la même longueur que l et est strictement plus grand que l).

Même si cette requête semble plus compliquée que sa version naïve (il y a deux conditions de sélection de plus), elle est plus efficace car elle peut être optimisée par des techniques d'optimisation standard (par exemple en utilisant un arbre B+ sur les étiquettes des termes).

Requêtes multi-termes : Dans [AFS00] nous avons également proposé l'utilisation de techniques d'optimisation et d'évaluation de requêtes multi-dimensionnelles[RSV01, GG98, PRS99] pour répondre à des critères de sélection qui impliquent plusieurs termes : Une description qui implique n termes provenant de différents thésauri peut être considérée comme des points dans un espace à n dimensions. Ceci est illustré dans la Figure 3.10, où chaque point correspond à une description avec deux termes décrivant le type de l'objet (peinture, sculpture, ...) et son style (impressionisme, renaissance, ...). Une requête qui cherche toutes les descriptions de peintures impressionnistes devient une requête de fenêtrage [RSV01] sur ces deux dimensions. Si ces requêtes sont fréquentes, il est intéressant de créer un index 2D (e.g. arbre R) [8].

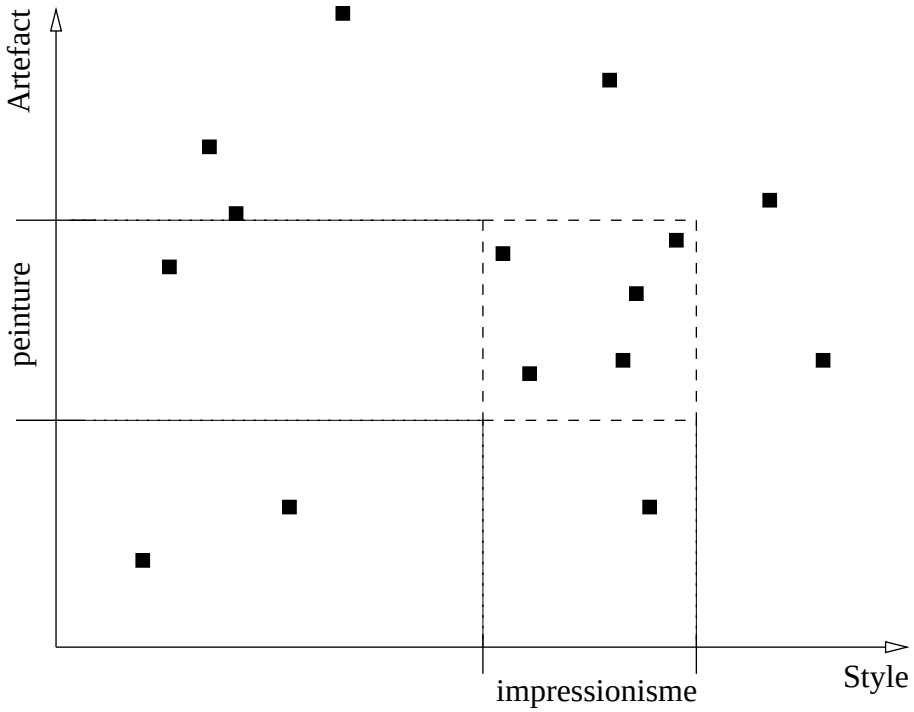

FIG. 3.10 – *Index bi-dimensionnel sur les thésaurus locaux du concept Artefact et Style*

Mesures de performances : Bien que le gain en performances de la linéarisation des thésaurus est évident (voir Figure 3.11 extraite de [Laf00]), le choix de la fonction de labélisation est soumis à plusieurs contraintes parmi lesquelles on peut citer la taille du code de linéarisation et la mise-à-jour des thésaurus

8. Bien que ce type d'index ne fasse pas encore partie du noyau d'un SGBD standard, il existe déjà des extensions efficaces comme par exemple l'extension spatiale de Oracle [Sha99].

(tous les changements dans un thésaurus peuvent mener à un reétiquetage des termes qui doit être propagé vers les descriptions utilisant des termes de ce thésaurus).

Nous avons commencé à étudier et comparer différents schémas dans le cadre du stage de DEA de S. Lafois [Laf00]. La figure 3.11 montre les temps de réponse en fonction de la sélectivité pour des requêtes de descendance pour trois codages différentes : le codage Dewey que j'ai décrit plus haut, le codage binaire qui correspond au code Dewey après une binarisation du thésaurus et une variante du codage postfix-préfixe [Die82] qui stocke pour chaque noeud sa position dans un parcours en préordre, son niveau et le nombre de descendants. Une description plus détaillée des différentes fonctions d'étiquetage et une

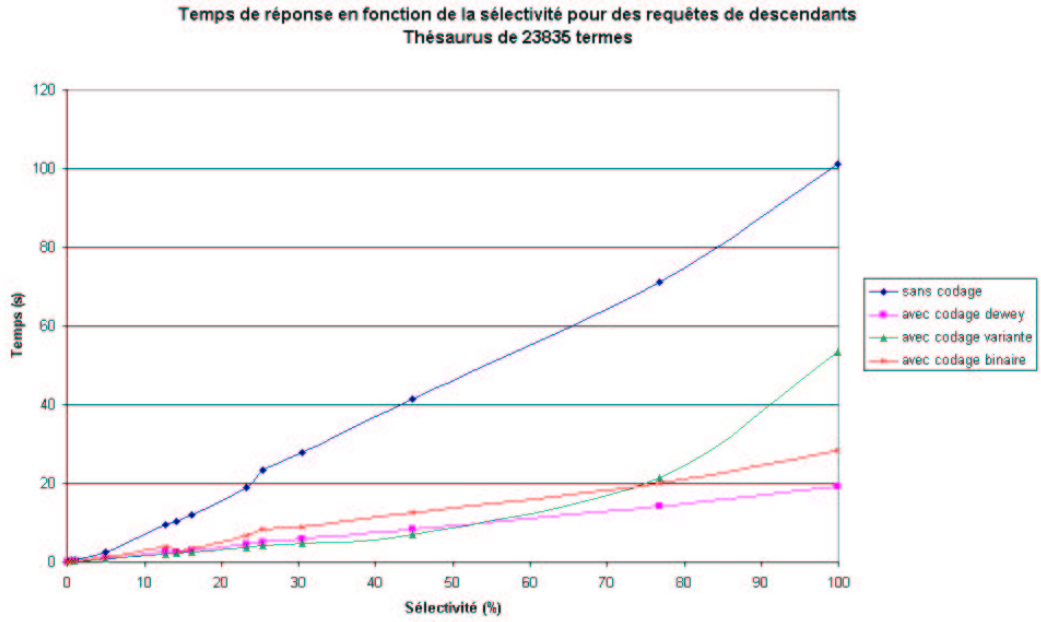

FIG. 3.11 – *Temps de réponse en fonction de la sélectivité pour des requêtes de descendance*

évaluation de leurs propriétés peuvent être trouvées dans [CPST02, KMS02].

3.4 Résumé

Notre approche de conception de schéma de métadonnées a prouvé son utilité dans deux applications de portails sémantiques pour *ressources culturelles et scientifiques*. La première application a été initiée par le Service de l'Inventaire[9] de la Direction de L'Architecture et du Patrimoine (DAPA) du Ministère de la Culture français et se place dans un projet plus général concernant la production et la diffusion électronique de la documentation par les différents services documentaires du ministère [Rad00, Fun03] (voir section 1.3.2). La deuxième application fait partie du projet MesMuses et consiste dans la création d'un entrepôt de métadonnées pour deux musées scientifiques : la Cité des Sciences et de l'Industrie (CSI) à Paris et l'Istituto E Museo Di Storia Della Scienza (IMSS) à Florence (voir section 1.3.2). Les deux expériences ont montré que l'approche est assez simple, bien comprise par les utilisateurs et que les schémas générés peuvent facilement être transposés dans les modèles RDF et orienté-objet.

9. http://www.culture.gouv.fr/culture/inventai/presenta/invent.htm

Un autre résultat important de ce travail est l'illustration que la technologie des bases de données est bien adaptée à la gestion et l'interrogation de métadonnées et joue un rôle important dans la création d'un Web Sémantique.

Chapitre 4

Intégration de données XML

Le partage de données sur le Web a redonné un intérêt nouveau au vieux problème de l'intéropérabilité des bases de données. Les premières recherches dans cette direction peuvent être situées dans les années 80 (voir [SL90] pour un état de l'art). Elles s'adressaient aux problèmes d'intégration de schémas de bases de données, de langages de requêtes, mais également des aspects systèmes comme la gestion de transactions distribuées et la communication entre SGBD. L'architecture prédominante était celle des SGBD fédérés qui permettaient une certaine autonomie des composants grâce à une hiérarchie de processus de transformation entre les schémas composants et un schéma fédéré.

L'apparition du Web a donné une nouvelle orientation à ce problème en s'intéressant à l'intégration de sources de données plus diverses (bases de données structurées, données scientifiques, documents) tout en respectant au maximum l'autonomie de chaque source. Le problème principal que nous avons étudié concerne l'intégration de données *sans mise-à-jour* : étant donné un ensemble de sources de données autonomes et hétérogènes, l'objectif est de permettre aux utilisateurs d'interroger les données comme une seule source avec un seul schéma global.

Le chapitre suivant est organisé en deux sections. La première section est un résumé du problème de l'intégration de données sur le Web et décrit les tâches principales ainsi que les architectures et modèles existants. La deuxième section présente STYX, un médiateur XML qui a été modélisé et implanté dans le cadre d'une collaboration avec C. Beeri de l'Université de Jérusalem et de la thèse d'Irini Fundulaki [Fun03] (que j'ai co-encadré avec M. Scholl).

4.1 Intégration de données sur le Web

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

1. Transformation de données: Un exemple typique de cette tâche est la transformation de données relationnelles en XML (ou l'inverse). Les problèmes importants qui doivent être résolus à ce niveau sont la perte d'information (par exemple dans le passage d'un modèle ordonné comme XML vers un modèle non-ordonné comme le modèle relationnel), la taille des données générées et la performance des traitements sur ces données. Ainsi, une multitude de représentations relationnelles pour des données XML avec des caractéristiques différentes en terme de redondance et de performances ont été proposées [AYF02]. L'exploitation de la structure de données à transformer (types, schémas) joue un rôle

crucial dans ce contexte.

2. Traduction de requêtes: La traduction de requêtes d'un langage (e.g. XQuery) en un autre langage (e.g. SQL) est liée au problème de transformation de données (la traduction d'une requête dépend de la transformation du modèle sous-jacent). Elle doit également prendre en compte la puissance d'expression du langage cible et nécessite souvent des extensions spécifiques afin d'obtenir la puissance d'expression du langage source. Par exemple, toutes les langages de requêtes XML utilisent des expressions de chemins avec un certain type de récursion (descendance) qui peut être "simulé" dans SQL en utilisant des techniques de linéarisation d'arbres (voir Section 3.3.2).
3. Réécriture de requêtes: Cette tâche est différente de la tâche précédente et généralement plus complexe car elle doit prendre en compte l'hétérogénéité structurelle et *sémantique* entre les schémas. Elle joue un rôle essentiel dans l'intégration de données sur le Web (voir plus bas et [Hal00] pour un état de l'art).
4. Fusion de données: La fusion des données essaye de répondre au problème de la représentation multiple d'une même information dans différentes sources. Elle fait partie de la tâche de réécriture de requêtes, mais se pose également dans un environnement où les données sont matérialisées (voir plus bas).

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

4.1.1 Architectures et modèles d'intégration

Un système d'intégration peut être caractérisé par son architecture et son modèle d'intégration. On peut distinguer entre deux architectures fondamentales pour l'intégration de données. L'approche *médiateur* [Wie95] est fondée sur la définition de *mapping* permettant la traduction de requêtes : une requête formulée par l'utilisateur dans les termes du schéma global est traduite en une ou plusieurs sous-requêtes qui sont évaluées sur les données sources. Les réponses sont combinées et transformées afin d'être compatibles avec le schéma global et conformes à la requête posée par l'utilisateur. L'architecture générale d'un tel système est montrée dans la Figure 4.1. L'approche *entrepôt* [Wid95] applique le principe des *vues matérialisées* et intègre les données en accord avec les schémas globaux. Le résultat est un entrepôt de données qui peut directement être interrogé à travers un langage adapté. L'architecture d'un entrepôt est montrée dans la figure 4.2.

Chacune de ces deux approches a des avantages et des inconvénients en termes de traitement des données intégrées. L'approche virtuelle favorise l'intégration de sources qui sont toujours disponibles avec de mises-à-jour fréquentes. L'approche matérialisée facilite l'intégration de sources qui ont des capacités d'interrogation limitées et/ou ne sont pas facilement accessibles. Elle pose le problème de rafraîchissement des données, mais permet des traitements de données indépendamment des capacités des sources.

Il existe également des approches hybrides qui essaient de combiner les avantages de ces deux approches en matérialisant les données dans un entrepôt et en utilisant l'approche virtuelle pour leur intégration. Ainsi le problème de disponibilité et de capacités des ressources est diminué tout en gardant la fléxibilité de l'approche virtuelle pour l'interrogation. Un exemple d'un tel système est Xylème où les données sources

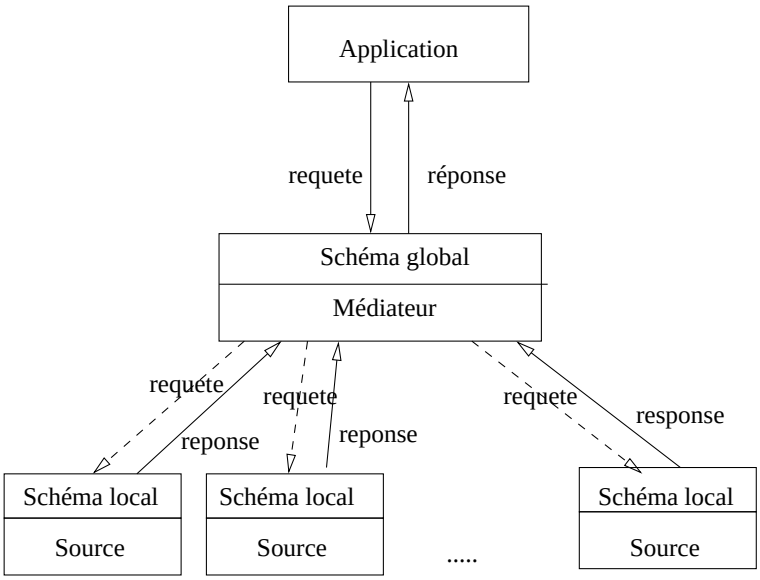

FIG. 4.1 – *Architecture Médiateur*

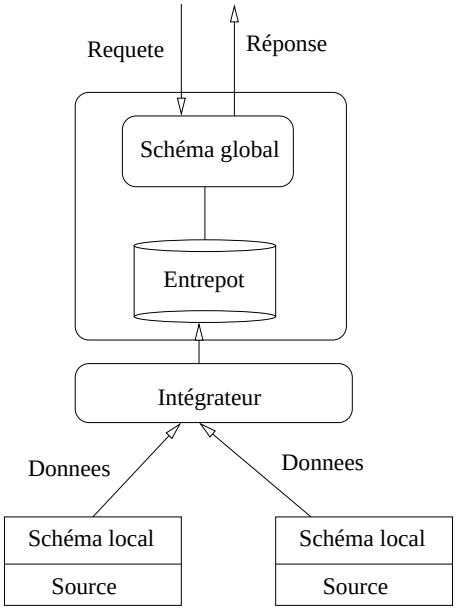

FIG. 4.2 – *Architecture Entrepôt*

sont stockées dans le format XML (sans être transformées) et intégrées à travers un mécanisme de vues entre les DTD concrètes des sources et des DTD abstraites montrées à l'utilisateur [CVV01].

4.1.2 Global ou Local comme Vue (LAV/GAV)

Le rôle d'un modèle d'intégration est de décrire le schéma global et son intégration avec les schémas locaux des sources. Il existe deux approches opposées pour l'intégration d'un ensemble de schémas locaux dans un schéma global. L'approche GAV (*global-as-view*) suppose que toutes les sources sont connues au moment de la définition du schéma et que l'ensemble des données interrogées correspond à l'union des données dans les sources (*hypothèse d'un monde fermé*). Le schéma global est défini commme une *vue globale* sur les schémas locaux. Ceci correspond à l'intégration classique utilisées dans les bases de données et des systèmes comme Hermes [AE95], Tsimmis [PGMW95], YAT [CCS00], MIX [BGL+99], Agora [MFK01] et XMLMedia [GMT02].

L'approche LAV (*local-as-view*) part de l'idée que le schéma global décrit un domaine d'intérêt spécifique (culture, sport, etc.) indépendamment des sources de données. Les sources sont ensuite intégrées comme *vues locales* sur le schéma global. Cette approche est utilisée dans les systèmes SIMS [AK93], Information Manifold [LRO96], Tukwila [PL00], Picsel [GLR00], Xyleme [Xyl01] et STYX [FABS02].

Les différents avantages et inconvénients de ces deux approches sont présentés dans [Lev01]. Un avantage important de l'approche GAV est la simplicité du processus de traduction d'une requête utilisateur qui consiste dans le remplacement des vues utilisées dans la requête par leur définition. Cette procédure relativement simple n'est plus possible dans l'approche LAV où une requête utilisateur doit être reformulée dans les termes de chaque schéma local (vue). Différents algorithmes pour la *réécriture de requêtes dans les termes des vues locales* ont été proposés dans ce contexte [LRO96, PL00] et une multitude de travaux théoriques ont été consacrés à l'étude de la compléxité de ce problème dans la cadre du modèle relationnel et des logiques de description [BLR97, Lev01, DG97].

La complexité algorithmique de l'approche LAV est compensée par une plus grande simplicité de l'utilisation du système. Dans l'approche GAV, la définition de la vue globale prend en compte l'ensemble des schémas de source et tout changement au niveau des sources (changement du schéma d'une source, apparition d'une nouvelle source, disparition d'une source) nécessite une vérification de la vue globale pour la détection des modifications nécessaires. Ainsi, la complexité de la vue globale croît avec le nombre de sources et l'approche GAV est seulement pratiquable si l'ensemble des sources à intégrer est petit et stable.

Un autre avantage de l'approche LAV est la possibilité de définir le schéma global indépendamment des données à intégrer. Cette liberté permet de se concentrer sur la description du domaine et de choisir éventuellement une ou plusieurs ontologies existantes pour la définition du schéma.

La figure 4.3 résume les différentes approches d'intégration que nous venons de décrire. Dans le contexte du Web, l'approche LAV est préférable : après avoir fixé un schéma global, il est possible d'effectuer différentes sortes de modifications sans être obligé de prendre en compte la totalité des sources. Même si dans certains cas, il peut être nécessaire d'étendre le schéma global, ce type de modification n'a généralement pas de conséquences importantes sur la définition des vues locales. Ainsi, comparée à l'approche GAV, l'approche LAV facilite surtout le passage à l'échelle.

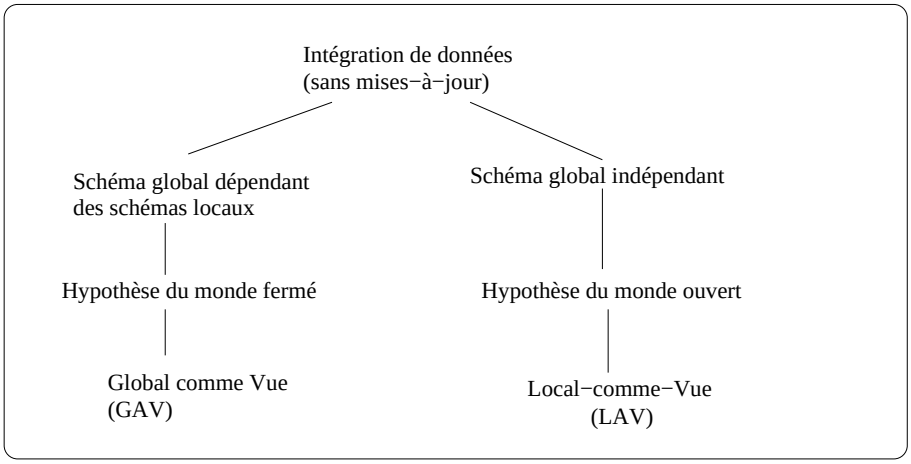

FIG. 4.3 – *Approches d'intégration de données GAV et LAV*

4.2 STYX : Un médiateur XML

XML est aujourd'hui reconnu comme le moyen standard pour la publication et l'échange de données sur le Web et le nombre de systèmes et d'application qui exportent leur données en XML et publient leur contenu sous forme de DTDs et de schémas XML s'agrandit chaque jour. Ainsi, indépendamment du format de stockage des données qui peut être XML, relationnel, orienté-objet ou autre, la vue présentée à l'extérieur a pour format XML.

Cette section présente notre approche d'intégration de sources XML [ABFS02b, AFS+01]. Nos contributions principales sont : (i) l'utilisation d'un certain type d'ontologies comme schéma global ; (ii) la définition d'un langage simple mais expressif pour la description de sources XML comme des vues sur le schéma global (approche LAV); (iii) la définition d'algorithmes pour le traitement des requêtes d'utilisateurs traduites en requêtes sur les sources et pour l'intégration des résultats obtenus; et (iv) la validation de cette approche à travers la réalisation d'un prototype appelé STYX [FABS02].

Nous illustrons notre approche à travers un exemple d'intégration de ressources XML concernant l'art et la culture. La source S_1=*http://www.paintings.com* est une ressource XML avec des documents sur des peintres et des peintures. Un exemple d'un document et de sa DTD est montré dans la figure 4.4 et la figure 4.5. La DTD d'une deuxième source S_2=*http://www.art.com* est décrite dans la figure 4.6.

```
<Peintre>
  <Nom>Van Gogh</Nom>
  <Peintures>
    <Peinture titre='Portrait du Docteur Gachet'
              année='1890'/>
    <Peinture titre='La Chaise et la Pipe'
              année='1888' />
    </Peinture>
  </Peintures>
</Peintre>
```

FIG. 4.4 – *Document XML dans* http://www.paintings.com

Dans un scénario d'intégration de données, une source isolée ne fournit généralement qu'une partie de l'information concernant un objet. Par exemple, la source S_1 contient pour chaque peinture son titre et son

```
<!ELEMENT Peintre  (Nom, Peintures)>
<!ELEMENT Nom #PCDATA>
<!ELEMENT Peintures (Peinture*)>
<!ATTLIST Peinture titre CDATA #IMPLIED
                  année  CDATA #IMPLIED>
```

FIG. 4.5 – *DTD de la source* S_1=http://www.paintings.com*)*

```
<!ELEMENT Musée (MuséeNom, Ville, Artefact+)>
<!ELEMENT Artefact (Désignation, Image*)>
<!ELEMENT Image EMPTY>
<!ATTLIST Image type #CDATA #IMPLIED
                url #CDATA #IMPLIED>
<!ELEMENT MuséeNom #PCDATA>
<!ELEMENT Ville #PCDATA>
<!ELEMENT Désignation #PCDATA>
```

FIG. 4.6 – *DTD de la source* S_2=http://www.art.com

année de création mais ne donne pas d'information sur son lieu d'exposition. De plus, les sources ne se distinguent pas uniquement par leur contenu, mais également par leur structure et la terminologie utilisée. Étant donné la structure hiérarchique des documents XML, ces différences structurelles sont encore plus significatives que dans une base de données relationnelle. Ceci est illustré par la source S_2 où les artefacts sont organisés par musée et non pas par artiste, comme c'est le cas dans la source S_1. Ainsi, tandis que dans la hiérarchie de S_1 un peintre apparaît comme *parent* de ses peintures, dans la source S_2 on serait obligé d'ajouter l'artiste comme *fils* de ses œuvres.

4.2.1 Schéma global et chemins conceptuels

La tâche principale d'un médiateur de requêtes est de fournir aux utilisateurs une interface unique sous forme d'un *schéma global* pour l'interrogation des données contenues dans les sources. Notre modèle d'intégration est fondé sur l'utilisation de *schémas conceptuels* comme schéma global d'intégration [1]. Un exemple de schéma conceptuel est montré dans la figure 3.2, page 40. L'information contenue dans une source est décrite par un ensemble de chemins dans le schéma. Nous distinguons deux types de chemins. D'abord, on veut decrire les différents fragments XML de la source par les concepts du schéma. L'utilisation de chemins partant d'un concept permet la définition de *concepts dérivés* qui décrivent l'information plus précisément. Par exemple, une source qui contient des artefacts produits par des artistes utilisera le concept dérivé Artiste.effectue.produit pour décrire ces objets. Ce concept décrit toutes les instances qu'on peut atteindre en suivant un chemin $x.r_1.r_2$ ou x est une instance du concept Artiste, r_1 est une instance du rôle *effectue* et r_2 est une instance du rôle *produit* (x, r_1 et r_2 sont virtuels et ne sont pas obligatoirement présents dans la source).

Ensuite, il est possible de composer des rôles pour créer des *rôles dérivés* qui décrivent des relations sémantiques entre des fragments XML. Par exemple, une source qui contient des artistes et leurs oeuvres peut utiliser le rôle dérivé effectue.produit pour relier les instances du concept *Artiste* aux instances du concept *Artefact* correspondant (la source n'est pas obligée de contenir des informations concernant les activités effectuées).

L'augmentation d'un schéma donné avec des rôles et des concepts dérivés donne comme résultat un

1. Nos motivations derrière ce choix sont expliquées dans la section 2.3.

schéma dérivé. La figure 4.7 montre un extrait du schéma dérivé du schéma de la figure 3.2, page 40 (voir [ABFS02a] pour une définition formelle). Il décrit éxactement les informations qui peuvent être four-

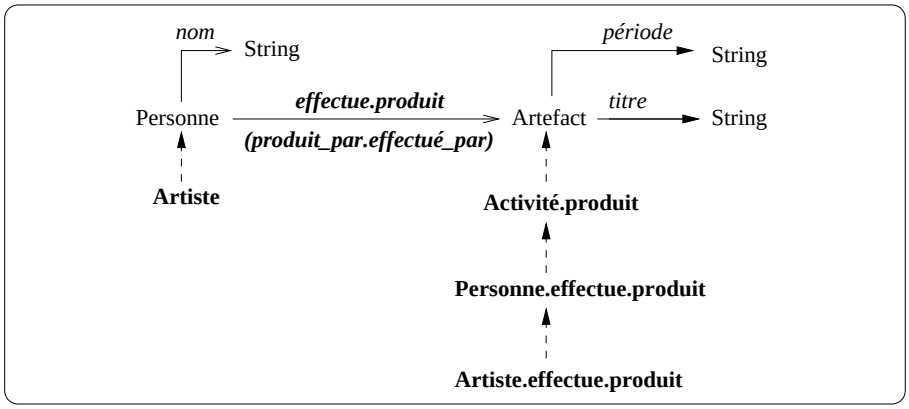

FIG. 4.7 – *Un schéma dérivé du schéma de la figure 3.2*

nies par une source qui permet de trouver des noms d'artistes et le titres et la période des artefacts qu'ils ont produits. L'exemple est seulement une extrait du schéma dérivé qui contient tous les concepts et rôles du schéma d'origine ainsi que tous les rôles et concept dérivés.

4.2.2 Description de ressources XML

Notre approche d'intégration décrit des sources XML comme des vues sur un schéma global dérivé. Une source est intégrée dans le système en fournissant un ensemble de *règles de description* qui décrivent les correspondances qui existent entre le schéma de la source et le schéma global. Les différentes façons pour définir ces correspondances se distinguent dans la taille et la précision de la définition, mais également dans la complexité de l'algorithme de réécriture [CVV01]. Nous avons choisi la même approche que [CVV01], qui associe des chemins dans le schéma global à des chemins dans le schéma source. Ceci nous permet d'associer des nœuds XML dans les sources à des concepts (dérivés) dans le schéma et de décrire la structure du document par des rôles (dérivés).

La description d'une source est composée d'un ensemble de règles qui associent des expressions XPath [CD99] à des chemins conceptuels dans le schéma global. Par exemple, les règles montrées dans la figure 4.8 décrivent la source S_1 (figure 4.5) dans les termes du schéma dérivé.

R_1:	`http://www.paintings.com/Peintre` as u_1	$\leftarrow$ `Artiste`
R_2:	u_1`/Nom` as u_2	$\leftarrow$ nom
R_3:	u_1`/Peintures/Peinture` as u_3	$\leftarrow$ effectue.produit
R_4:	u_3`/@titre` as u_4	$\leftarrow$ titre
R_5:	u_3`/@période` as u_5	$\leftarrow$ période

FIG. 4.8 – *Description de la source* S_1=http://www.paintings.com

Une règle est composée d'un nom et de deux expressions de chemins séparées par $\leftarrow$. L'expression de gauche est une expression XPath qui débute avec une URL concrète, comme dans la règle R_1, ou avec une variable, comme dans la règle R_2. Cette expression est suivie par une déclaration optionnelle d'une variable `as` u_i (l'utilisation des variables sera expliquée plus loin). L'expression XPath est appelée le *chemin source*

de la règle. L'expression de droite est un chemin dans le schéma global, appelé le *chemin conceptuel* de la règle. Une règle R est appelée une *règle relative* si sa racine est une variable, et une *règle absolue* sinon.

Descriptions : Étant donné un ensemble de règles de description pour une source s, on définit *l'accessibilité* pour les règles et les variables de la façon suivante :

- chaque règle absolue (avec comme racine l'URL de s) est accessible;
- chaque variable liée par une règle accessible est accessible;
- chaque règle relative avec une variable racine accessible est accessible;

Une *description* M d'une source s dans un schéma $\mathcal{S}$ est un ensemble de règles de description tel que chaque règle a un nom différent, toutes les règles et variables sont accessibles dans s, et les concepts, rôles et attributs utilisés dans ses règles apparaissent dans le schéma dérivé.

Interprétation : Les règles de description définissent des instances de concepts, des relations entre ces instances et des valeurs d'attributs. Par exemple, la règle R_1 dans la figure 4.8 décrit les éléments racines de type `Peintre` dans les documents de la source S_1 comme des instances du concept Artiste. La règle R_2 ajoute des valeurs de l'attributs *nom* à ces instances en évaluant l'expression XPath `Nom` sur les éléments x obtenus par la règle R_1. De la même manière, la règle R_3 relie les instances obtenues par la règle R_1 à des instances du concept Artefact par un chemin de type *effectue.produit* (voir [ABFS02a] pour une définition plus précise de l'interprétation des règles).

En accord avec l'approche LAV, les extensions XML de chaque concept sont considérées comme des sous-ensembles d'extensions réelles (mais inconnues). En effet, en ajoutant des nouvelles sources et règles de description, la taille de chaque extension peut augmenter.

4.2.3 Évaluation de requêtes

Langage de requêtes

Les utilisateurs formulent des requêtes en utilisant une variante simplifiée d'OQL, le standard pour l'interrogation de bases de données à objets [CB00]. Voici une requête Q_1 qui demande tous les *titres d'artefacts produits par Van Gogh* :

Q_1: **select** c
from Personne a,
a.nom b,
a.effectue.produit.titre c
where b = "Van Gogh"

Toutes les variables (a, b et c) sont liées dans la clause **from**. Le chemin conceptuel qui précède une variable x dans la clause **from** est appelé le *chemin d'affectation* de x. Par hypothèse, les requêtes satisfont les restrictions suivantes. Premièrement, le graphe qu'on obtient en reliant les variables de la requête par leurs chemins d'affectation tel que x est relié par p à y si l'expression $x.p\ y$ apparaît dans la clause **from** doit former un *arbre*. Deuxièmement, il n'est pas possible de créer de nouvelles structures dans la clause **select**. Bien que cette fonctionnalité augmente le pouvoir d'expression du langage, nous pensons qu'elle n'est pas indispensable pour notre application et qu'elle est orthogonale au problème de la recherche d'information. Troisièmement, la clause **where** est une conjonction de prédicats simples de la forme $x_i \theta d$ où $\theta \in \{=, <, >, \leq, \geq\}$ et d est une valeur atomique. Ainsi il n'est pas possible d'exprimer des jointures par des égalités $x_i = x_j$ entre des variables ce qui restreint la puissance d'expression du langage mais simplifie

la réécriture et l'évaluation des requêtes. Enfin, les chemins conceptuels apparaissent uniquement dans la clause **from** [2] et les requêtes ne contiennent pas de quantificateurs, agrégats, ou sous-requêtes [3].

Réécriture de requêtes

Il existe différentes options pour répondre à une requête étant donné un ensemble de sources S et l'ensemble de leurs descriptions M. La première solution est d'essayer d'évaluer la requête dans chacune des sources $s \in S$. Ceci signifie que nous devons réécrire la requête en une requête XML à laquelle s peut répondre. L'idée derrière notre algorithme de réécriture est de chercher tous les homomorphismes β qui existent entre les variables (où une partie des variables) V de la requête Q et l'ensemble des règles R_i de la description M. Un tel homomorphism est appelé une *liaison* et satisfait les contraintes suivantes :

- Si x est la variable racine, alors le chemin conceptuel de la règle $\beta(x)$ est un sous-concept du chemin d'affectation de x dans le schéma global dérivé. Ceci signifie que la règle $\beta(x)$ est une règle absolue qui trouve des instance de la variable x.
- Sinon il existe une expression $y.p\,x$ dans la clause **from** telle que la règle $\beta(x)$ permet de trouver des instances de x à partir d'instances de y reliés par le rôle dérivé p et
  1. $\beta(y)$ est défini,
  2. la racine de la règle $\beta(x)$ est liée par la règle $\beta(y)$,
  3. et le chemin conceptuel de $\beta(x)$ est égal au chemin d'affectation de x.

L'ensemble des variables liées par une liaison β est noté $dom(\beta)$. Une liaison est appelée *complète* si elle lie toutes les variables de la requête Q. Une liaison β est *maximale* s'il n'existe pas de liaison β' telle que $dom(\beta) \subset dom(\beta')$ et $\beta(x) = \beta'(x)$ pour tous $x \in dom(\beta)$.

Chaque liaison complète peut être utilisée pour la réécriture d'une requête en une requête XQuery qui peut être évaluée par la source XML. Par exemple pour la requête Q_1, la source S_1 retourne des instances de la variable a grâce à la règle R_1 (Artiste est un sous-concept du concepte Personne). Le même raisonnement est possible pour la variable b et la règle R_2. Par contre, il n'existe pas de règle dans la description de S_1 qui peut être associée à la variable c. La solution est de *concaténer* les règles R_3 et R_4 pour obtenir une nouvelle règle $R_3.R_4$ qui trouve les instances cherchées (voir après). La liaison obtenue ainsi est $\beta_1 = \{a \mapsto R_1, b \mapsto R_2, c \mapsto R_3.R_4\}$ et, en substituant les chemins conceptuels dans la requête par les chemins sources des règles correspondantes, nous obtenons une réécriture $\mathcal{R}(Q_1,\beta_1)$:

$\mathcal{R}(Q_1,\beta_1)$: **select** c
from `http://www.paintings.com/Peintre` a,
a`./Nom` b, a`./Peintures/Peinture/@titre` c,
where b = "Van Gogh"

La requête $\mathcal{R}(Q_1,\beta_1)$ peut facilement être traduite en une requête XQuery:

```
FOR    $a IN document("http:///www.paintings.com/Peintre")/Peintre,
       $b IN $a/Nom,
       $c IN $a/Peinture/@titre
WHERE  $b = "Van Gogh"
RETURN $c
```

2. Une requête avec des chemins conceptuels dans la clause **select** et/ou la clause **where** peut facilement être transformée en une requête équivalente où ces chemins sont "poussés" dans la clause **from**.

3. Certaines requêtes avec quantification existentielle peuvent être exprimées à travers des variables qui apparaîssent uniquement dans la clause **from** de la requête.

Clôture d'une description : L'exemple précédent montre que pour calculer une liaison β d'une requête Q vers une description M la liaison β doit prendre en compte des chemins conceptuels dans Q qui sont des concaténations de chemins conceptuels dans la description M. Deux règles de description $R_1 : a/q_1\ as\ v_1 \leftarrow p_1$, $R_2 : v_1/q_2\ [as\ v_2] \leftarrow p_2$, peuvent être *concaténées*, si la composition de leurs chemins conceptuels, $p_1 \circ p_2$, est bien définie[4]. Le résultat de la concaténation est la règle $R_1.R_2 : a/q_1/q_2\ [as\ v_2] \leftarrow p_1 \circ p_2$. Étant donné une description M, sa *clôture* est l'ensemble de toutes les règles qu'on peut obtenir à partir de M en appliquant, d'une manière répétée, l'opération de concaténation [5].

Algorithme de liaison : L'algorithme de liaison $\mathcal{B}(Q,s)$ qui permet de trouver toutes les liasons *maximales* pour une requête Q vers la description M d'une source s est décrit dans [ABFS02a]. Il peut être résumé ainsi :

1. D'abord il faut calculer la clôture M^* de la description M.
2. Ensuite nous cherchons pour la variable racine a de la requête toutes les règles de descriptions absolues dans M^* qui permettent d'instancier la variable (on choisit toutes les règles dont le chemin conceptuel décrit le concept ou un sous-concept du chemins d'affectation dans le schéma dérivé).
3. On obtient un premier ensemble de liaisons de variables $\beta_i = \{a \mapsto R_i\}$ qui est étendu ensuite (récursivement) en parcourant l'arbre de la requête : étant donnée une variable x qui a été liée à une règle R_i, on cherche à lier chaque enfant y de x à une règle R_j tel que le chemin d'affectation de y correspond au chemin conceptuel de R_j *et la racine de* R_j *est liée par la règle* R_i.

Réécriture avec liaisons complètes : Le processus de mise-en-correspondance/réécriture doit être appliqué à chaque source. Ensuite les réponses sont assemblées et retournées à l'utilisateur. Formellement, une *réécriture globale* $\mathcal{RG}(Q,S)$ d'une requête vers un ensemble de ressources S est l'union de toutes les réécritures $\mathcal{R}(Q,\beta)$, où β est une *liaison complète* de Q vers la description d'une source $s \in S$:

$$\mathcal{RG}(Q,S) = \bigcup_{s \in S} \bigcup_{\beta \in \mathcal{B}(Q,s)} \mathcal{R}(Q,\beta)$$

Décomposition de requêtes

Dans certains cas on ne peut pas obtenir une liaison complète pour une source. Il est alors nécessaire de *décomposer* la requête en plusieurs requêtes qui sont évaluées sur différentes sources. En d'autres termes, les liaisons β trouvées par l'algorithme $\mathcal{B}(Q,s)$ ne permettent à la source de fournir qu'une information incomplète pour une partie des variables de la requêtes. Afin de compléter cette information, nous *décomposons* la requête Q en une *requête préfixe* P_β qui peut être traitée par la source en utilisant β, et un *ensemble de requêtes suffixes* $\mathcal{S} = \{Q_1,Q_2,...,Q_n\}$ qui seront soumises à nouveau à l'algorithme de calcul des liaisons pour trouver des liaisons.

L'ensemble des requêtes suffixes $\mathcal{S}$ d'une requête préfixe P_β dans une requête Q est définie de la façon suivante. Soit $N = \mathcal{F}(Q,P_\beta)$ l'ensemble des variables dans P_β qui contiennent au moins un enfant dans Q qui n'est pas dans P_β (nous appelons N la *frontière* de P_β et toutes les variables dans N des *variables de jointure*). Alors, chaque variable x dans N génère une requête suffixe Q_i qui correspond au sous-arbre de Q avec la racine x et qui contient tous les descendants de x qui ne sont pas dans P_β.

Soit Q une requête, S un ensemble de sources et β une liaison maximale de Q dans la description de S. Une *décomposition* de Q est un couple $\mathcal{D}(Q,\beta) = [P_\beta,\mathcal{S}]$ où P_β est une requête préfixe de Q et $\mathcal{S}$ est un

4. Nous n'imposons pas de restriction sur la concaténation des expressions XPath.
5. M^* peut être obtenu par un calcul du point-fixe.

ensemble de requêtes suffixes de P_β dans Q (on peut noter que $\mathcal{S}$ est vide si β est une liaison complète de Q).

Prenons comme exemple la requête suivante qui demande les *titres des objets produits par Van Gogh, ainsi que le nom et la ville des musées dans lesquels ils sont exposés* :

Q_2: **select** d, f, g
from Personne a,
a.nom b,
a.effectue.produit c,
c.titre d,
c.exposé_par e,
e.nom f,
e.city g
where b = “Van Gogh”

Supposons que l'ensemble des sources $S = \{S_1,S_2\}$ décrit dans notre médiateur contient les deux sources S_1 et S_2 montrées dans la section 4.2.2. La liaison β_1 de la section 4.2.3 est une liaison maximale mais *partielle* de Q_2 vers la description de S_1. Cette liaison n'est pas complète : la source S_1 ne peut pas fournir à elle toute seule une réponse à cette requête car elle ne connaît pas les lieux d'exposition de ses peintures (pour une instance de la variable c on ne peut pas obtenir les instances de la variable e et ses descendants). S_1 est capable de fournir une *réponse partielle* en évaluant la requête P_{β_1} montrée dans la figure 4.9. La décomposition $\mathcal{D}(Q,\beta_1) = [P_{\beta_1},\mathcal{S}]$ est montrée dans la figure 4.9. L'information manquante peut être obtenue par la seule requête dans $\mathcal{S} = \{Q_2(b)\}$.

P_{β_1}: **select** d
from Personne a,
a.nom b
a.effectue.produit c,
c.titre d
where b = “Van Gogh”

$Q_2(b)$: **select** f, g
from Artefact c,
c.exposé_par e,
e.nom f, e.city g

FIG. 4.9 – *La requête préfixe P_{β_1} et l'unique requête suffix $Q_2(b)$*

Jointure des résultats

Si on suppose que $Q_2(b)$ peut être évaluée par une autre source (par exemple S_2), les résultats des deux requêtes sont joints sur la variable c afin d'obtenir une réponse complète à la requête initiale (évidemment, si une telle décomposition ne peut pas être trouvée, le mieux qu'on puisse faire est de présenter à l'utilisateur le résultat partiel de la requête P_{β_1}).

Pour joindre les fragments XML obtenus comme instances d'une variable de jointure, on a besoin d'un moyen pour décider si deux fragments représentent le même objet. On suppose que les sources sont hétérogènes et autonomes, et on ne s'attend pas à ce qu'elles fournissent des identificateurs d'objets persistants et valides pour toutes les sources. Les attributs XML de type `ID/IDREF` peuvent être utilisés comme références internes, mais pas comme clés globales. Même si les sources définissent des clés en terme d'attributs et d'éléments XML comme proposé dans [BDF+01, FKS01a, TBMM01], il n'est pas réaliste que des sources différentes et autonomes utilisent la même DTD avec la même structure et les mêmes noms d'éléments.

Une façon de résoudre ce problème est de définir des clés au niveau du schéma global sous forme

d'attributs dérivés[6], appelés *attributs clés*. Dans notre schéma, nous pouvons par exemple considérer que chaque instance du concept Personne est identifiée par son attribut *nom*. De la même façon, les instances du concept Artefact sont identifiées par une clé composée de leur titre et de leur année de création. Cette clé est représentée par l'ensemble $K_{\text{Artefact}} = \{titre,produit.date.anne\}$.

Jointure : Les *clés* permettent de joindre les résultats des requêtes P_{β_1} sur la source S_1 et $Q_2(b)$ sur la source S_2. La jointure s'effectue sur les instances des variables de jointures dans la frontière de $N = \mathcal{F}(Q,P_{\beta_1})$. Soit x une variable de jointure x dans une requête Q : si Q est une requête préfixe, x appartient à la frontière de Q; sinon x est la racine de Q. Pour faire la jointure, Q doit être étendue avec la clé K_c de telle façon que la nouvelle requête $ext(Q,x)$ retourne toutes les valeurs de Q *et* les valeurs clés des instances de x. Le résultat de $ext(Q,x)$ est de la forme $\{[a_1,a_2,\ldots a_n,x_1,x_2,\ldots,x_p]\}$ où $[x_1,x_2,\ldots,x_p]$ est une valeur clé d'une instance de x. Par exemple, la requête préfixe obtenue après l'extension de la requête P_{β_1} (figure 4.9) avec la clé du concept Artefact est la suivante :

$ext(P_{\beta_1},c)$: **select** *d, t, y*
from Personne *a*,
a.nom *b*,
a.effectue.produit *c*,
c.titre *d*,
c.titre t,
c.produit.date.année y
where *b* = "Van Gogh"

Soit Q_i une requête suffixe dans $\mathcal{S}$ avec la variable racine x. Alors $P_\beta \bowtie_x Q_i$ désigne la *jointure* entre $ext(P_\beta,x)$ et $ext(Q_i,x)$ sur les valeurs clés de la variable x. Par exemple, $P_{\beta_1} \bowtie_c Q_2(b)$ désigne la jointure entre le résultat de la requête $ext(P_{\beta_1},c)$ et la requête $ext(Q_2(b),c)$ sur les valeurs de clés de la variable c.

Réécritures de requêtes et plans d'éxecution

Nous pouvons maintenant donner une définition plus formelle de la réécriture d'une requête Q sur un ensemble de sources S. Nous allons appeler cette réécriture une *réécriture globale,* notée $\mathcal{RG}(Q,S)$. Soit $\mathcal{D}(Q,\beta) = [P_\beta,\mathcal{S}]$ la décomposition de Q pour β. Alors P_β peut directement être traduite en une requête source grâce à la liaison β (nous supposons que toutes les requêtes sont étendues avec les clés nécessaires pour la jointure) :

– Une *réécriture préfixe* $\mathcal{RP}(Q,\beta)$ pour une décomposition $\mathcal{D}(Q,\beta) = [P_\beta,\mathcal{S}]$ est la jointure entre la réécriture de la requête préfixe P_β et le réécritures globales de toutes ses requêtes suffixes $Q_i \in \mathcal{S}$, $1 \leq i \leq n$, dans $\mathcal{S}$. Soit x_i la variable racine de chaque requête suffixe Q_i (si β est une liaison complète, alors $\mathcal{RP}(Q,\beta) = \mathcal{R}(Q,\beta)$) :

$$\mathcal{RP}(Q,\beta) = \mathcal{R}(P_\beta\beta, \bowtie)_{x_1} \mathcal{RG}(Q_1,S) \bowtie_{x_2} \ldots \bowtie_{x_n} \mathcal{RG}(Q_n,S))$$

– Chaque requête suffixe dans $\mathcal{S}$ doit être réécrite à son tour. La *réécriture globale* $\mathcal{RG}$(Q,S) pour la requête initiale Q sur un ensemble de sources S est définie comme l'union de toutes les réécritures préfixes :

$$\mathcal{RG}(Q,S) = \bigcup_{s \in S} \bigcup_{\beta \in \mathcal{B}(Q,s)} \mathcal{RP}(Q,\beta)$$

6. Une séquence de rôles qui se termine par un attribut est appelé un attribut dérivé.

On voit qu'une réécriture globale est une union de jointures d'une requête préfixe avec des réécritures globales des requêtes suffixes.

Plan d'exécution : Une expression QEP est un *plan d'exécution* si (1) QEP est une requête qui peut être évaluée par une seule source (QEP est un plan *atomique*) *ou* (2) QEP est l'union de deux plans d'exécution *ou* (3) QEP est la jointure entre deux plans d'exécution. Les sources répondent aux plans d'exécution atomiques, tandis que le médiateur effectue les jointures et les unions. Un plan d'exécution peut contenir plusieurs plans atomiques destinés à la même source.

Entrée: une requête Q et un ensemble de sources S
Sortie: un plan d'exécution pour Q;
Algorithme: $QEP(Q,S) = \emptyset$;
pour chaque source $s \in S$ {
  si $\mathcal{B}(Q,s) \neq \emptyset$ {
    / il existe au moins une liaison maximale pour Q dans s */*
    pour chaque liaison $\beta \in \mathcal{B}(Q,s)$ {
      si β est une liaison complète
        $QEP(Q,\beta) := Q$;
      sinon {
        calculer décomposition $\mathcal{D}(Q,\beta) = [P_\beta,\mathcal{S}]$;
        $QEP(Q,\beta) := P_\beta$;
        pour chaque requête suffixe $Q' \in \mathcal{S}$
          si $QEP(Q,\beta) \neq \emptyset$
            / il existe un plan non-vide */*
            / pour chaque requête suffixe jusqu'à Q' */*
            si $QEP(Q',S) \neq \emptyset$
              / il existe un plan pour Q' */*
              $QEP(Q,\beta) := QEP(Q,\beta) \bowtie_{\vec{k}'} QEP(Q',S)$;
            sinon
              $QEP(Q,\beta) := \emptyset$;
      }
      $QEP(Q,S) = QEP(Q,S) \cup QEP(Q,\beta)$
    }
  }
}
retourne $QEP(Q,S)$;

FIG. 4.10 – *Algorithme de génération de plan d'exécution $QEP(Q,S)$*

Étant donné un ensemble de sources S et une requête Q, l'algorithme $QEP(Q,S)$ montré dans la figure 4.10 retourne un plan d'exécution pour Q et l'ensemble de sources S. Pour chaque source $s \in S$ et liaison maximale $\beta \in \mathcal{B}(Q,s)$, un plan d'exécution $QEP(Q,\beta)$ de la réécriture $\mathcal{RP}(Q,\beta)$ est calculé : si β est une liaison complète, le résultat est inclus dans le résultat de Q. Sinon, la requête Q est décomposée en une requête préfixe P_β et un ensemble de requêtes suffixes $\mathcal{S}$ (ces requêtes sont étendues par les requêtes clés comme nous l'avons montré plus haut). Le plan d'exécution $QEP(Q,\beta)$ généré par β est la jointure entre P_β et le plan d'exécution obtenu pour chaque requête suffixe $Q' \in \mathcal{S}$ par un appel récursif de $QEP(Q',S)$. Le plan général $QEP(Q,S)$ est l'union de tous les plans $QEP(Q,\beta)$.

L'algorithme définit deux raisons pour interrompre la construction d'un plan d'exécution pour une source s et une liaison β : une première raison triviale est l'absence de liaison maximale pour Q dans s. La deuxième raison est l'absence d'un plan d'exécution pour au moins une requête suffixe dans $\mathcal{S}$.

4.2.4 Le prototype STyX

Notre modèle à été implanté dans le prototype $STyX$ [FABS02] dont l'architecture est montrée dans la figure 4.11. Une requête utilisateur est d'abord analysée syntaxiquement et validée par rapport au schéma

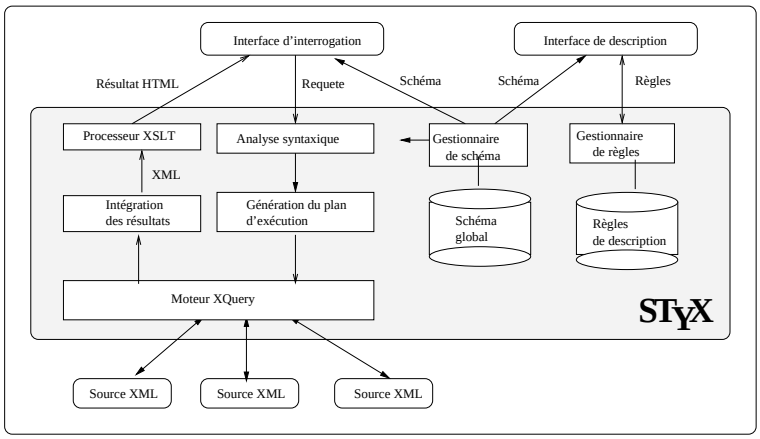

FIG. 4.11 – *Architecture de STyX*

global. Ensuite, le système génère un plan d'exécution dont les plans atomiques sont traduits en requêtes XQuery et évalués (dans le prototype, nous utilisons le moteur Kweelt [7]). Les résultats obtenus sont joints par le module d'intégration et renvoyés à un processeur XSLT (Cocoon [8]), qui effectue la transformation du résultat final en HTML (voir figure 4.12).

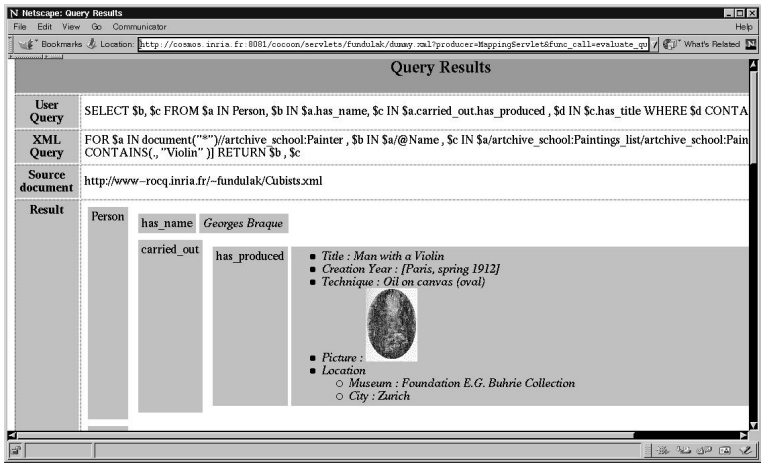

FIG. 4.12 – *Résultat d'une requête*

7. http://db.cis.upenn.edu/Kweelt/.
8. http://xml.apache.org/cocoon

Chapitre 5

Conclusion et Perspectives

Bien que le Web ait subi de grands changements depuis son apparition il y a bientôt 15 ans, son évolution est toujours fidèle à l'idée essentielle de faciliter le partage d'informations grâce à la définition d'abstractions et de standards qui permettent de cacher l'hétérogénéité des protocoles (URL), des interfaces utilisateurs (HTML) ou des modèles de données (XML). Plus récemment, cette approche a été adoptée par les *services Web* pour ouvrir les applications informatiques vers le Web.

Bien qu'on ne puisse pas encore parler d'une architecture de services Web (dans le sens d'une architecture CORBA), il existe déjà un ensemble de recommandations W3C [SOA, WSD] et d'autres propositions de standards [1] qui définissent le coeur d'une telle architecture. Ces efforts sont soutenus par des organismes comme Apache, ainsi que tous les grands industriels en informatique comme Microsoft, Sun et IBM sous forme de plateformes compatibles (.NET [2], J2EE [3], Apache Axis [4]).

Le principe des services Web est d'élargir la notion de ressource Web vers une interface qui permet l'accès à des applications complexes tout en faisant l'abstraction des aspects d'implantation. Ils héritent ainsi deux caractéristiques importantes généralement attribuées aux ressources Web :

Indépendance : Comme pour une page HTML qui peut être référencée, accédée et affichée sur n'importe quel poste informatique connecté à Internet, la localisation et l'implantation d'un service Web est indépendante du protocole de communication et de la localisation et l'implantation de son client. Ainsi les services Web représentent une brique importante pour la construction d'environnements de *calcul global* (global computing) où les données *et* les traitements peuvent être distribués sur différents noeuds d'un réseau.

Autonomie : Le choix des fonctions dans un système d'information classique est limité par les outils installés dans l'environnement du système. Ces limites disparaissent avec les services Web qui sont autonomes et évoluent indépendamment des applications qui les utilisent. Cette autonomie signifie surtout que les applications doivent être capables de s'adapter aux changements d'environnement (disparition d'un service, apparition de nouveaux services) et prendre en compte la qualité et la disponibilité des services.

Comme c'est le cas pour les ressources Web classiques, l'indépendance et l'autonomie des services Web facilitent leur déploiement et leur utilisation, mais ces deux propriétés créent également le besoin de nouveaux modèles, langages et outils pour la découverte et la composition de services. Dans le reste de ce chapitre, je présenterai deux de ces besoins auxquels je m'intéresse actuellement.

1. http://uddi.org/
2. Microsoft .NET, http://www.microsoft.com/net/
3. Java 2 Platform, Enterprise Edition, http://java.sun.com/j2ee
4. http://xml.apache.org/axis

5.1 Découverte de services

Le principe des moteurs de recherche est d'indexer le contenu textuel d'une ressource, ce qui les rend limités pour la découverte de ressources multimédias (son, image, vidéo) ou des ressources dynamiques qui sont accessibles à travers des formulaires et des services Web. Une solution à cette limitation est d'annoter ce type de ressources avec des métadonnées (par exemple RDF) qui décrivent leur contenu et leurs fonctionnalités et peuvent être interrogées avec les langages adaptés (par exemple RQL).

Il existe déjà une multitude de propositions et standards pour la description de services Web qui sont souvent destinés à un certain type d'utilisations. Ainsi, une description WSDL (Web Service Description Language) [WSD] contient toutes les informations nécessaires pour créer la connexion et les messages entre le client et le serveur du service ainsi que pour valider les paramètres et le résultat.

Si les descriptions WSDL sont des documents XML qui peuvent être stockés et interrogés pour trouver par exemple toutes les opérations d'un service, elles sont insuffisantes pour répondre à des requêtes comme *"Je cherche tous les services de réservation de transports en Europe"*. Le standard UDDI (Universal Description, Discovery and Integration) propose une solution à ce type de requêtes par un modèle pour la description des fournisseurs de services (businessEntity), des services (businessService) et de leurs implantations (bindingTemplate). Le concept clé de UDDI est la notion de *tModel* pour la construction de registres standardisés et partagés entre les fournisseurs de services et leurs clients. Chaque entrée de type *tModel* est identifiée par un identifiant universel (UUID) et peut désigner une activité industrielle (codes NAICS), un type de produit (UN/SPSC), mais aussi la définition WSDL d'un service. Ils servent à la catégorisation des activités économiques (pages jaunes), l'identification des fournisseurs de services (white pages), et la description des processus et services (green pages). UDDI est très orienté vers le commerce électronique et souvent critiqué pour son approche fondée sur l'utilisation d'identificateurs globaux et de registres centralisés pour la gestion et la description de services [VSS$^+$]. Il a subi plusieurs modifications/extensions à cause de ces critiques et se trouve déjà à la troisième version depuis Septembre 2000.

Mon expérience dans la description et l'intégration de ressources Web m'a donné envie d'étudier la problématique de découverte et d'intégration de services Web. L'idée d'un premier modèle simple est d'utiliser des taxonomies pour décrire les fonctionnalités et les interfaces (paramètres et résultats) des services. Un extrait d'une telle taxonomie pour le tourisme est montré dans la figure 5.1. Les descriptions sont stockées

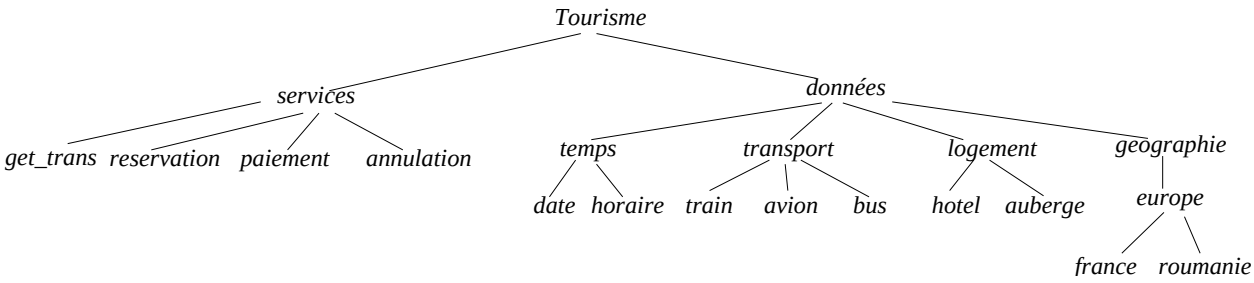

FIG. 5.1 – *Taxonomie pour le tourisme*

dans des *registres* et peuvent être interrogées par des requêtes prenant en compte les liens d'inclusion entre les termes de la taxonomie (par exemple, la requête précédente trouve également tous les services de reservation de trains en France). Dans le cadre d'un stage de fin d'études [Pop03] nous sommes en train d'implanter un prototype de ce modèle au dessus du système ActiveXML (section 2.1.4). Les taxonomies et registres de services sont représentés sous forme de documents ActiveXML qui peuvent être interrogés à travers un service Web. L'utilisation du paradigme de *document intensionnel* donne plusieurs ouvertures intéressantes

dans le déploiement des taxonomies et des registres de services que nous envisageons d'explorer :

– Grâce à la possibilité de mélanger d'une manière transparente des données statiques et des résultats d'appels de services, la distribution des taxonomies de services et de données devient naturelle : chaque taxonomie est accessible à travers un service Web qui peut directement être intégrée dans d'autres taxonomies sous forme d'un appel de service. Par exemple un service $S1$ peut proposer l'accès à une ontologie pour le voyage sous forme d'un document ActiveXML, tandis qu'un autre service $S2$ est un point d'accès spécialisé aux noms de lieux géographiques. Ces deux ontologies peuvent être intégrées dans un troisième document ActiveXML (sous forme d'appels de services) qui devient ainsi un point d'accès unique pour décrire des services proposant des voyages vers des lieux géographiques.
– Une deuxième application, qui est plus intéressante dans le contexte des services Web, concerne la distribution des registres de services (ebXML Version 3.0 [Oas] et [VSS$^+$]) : Un registre est un document ActiveXML qui est accessible à travers des appels de services et peut à son tour contenir des appels vers d'autres registres. Le résultat est un registre distribué sur différents pairs ActiveXML. L'avantage est une plus grande autonomie des fournisseurs de services (chaque fournisseur peut maintenir son registre de services) avec une plus grande flexibilité de gestion.

5.2 Composition de services

Un objectif des langages de description de services Web est de faciliter leur composition pour créer des nouveaux services. Des exemples de ce type de services "complexes" sont les services de reservation de voyages (qui incluent la reservation des moyens de transport et des logements) ou des service du commerce électronique comme par exemple l'achat d'un produit (services de commande, de livraison et de facturation). À un niveau abstrait, une agrégation de services (élémentaires ou complexes) peut être représentée par un graphe dont les noeuds correspondent à des appels de services et les arcs aux flux de contrôle (livraison après la commande) et les échanges de données (numéro du bon de commande) entre les appels reliés. Ce modèle générique est mis en pratique dans une multitude de langages de composition de services Web (BPEL4WS [ACD$^+$03], WSFL [IBM], ebXML [Oas] et BPML [Ark02]). Un aspect important de ces langages est leur "degré de déclarativité" qui indique les possibilités de créer et de raisonner sur le comportement de services complexes [BFHS03].

Une voie que j'envisage d'explorer est la composition dynamique de services Web par interrogation. L'idée est d'intégrer des requêtes de services dans la spécification d'un service complexe. Par exemple, une agence de voyages veut proposer un nouveau service qui intégre différents services pour reserver des vols et des chambres d'hotels et pour louer des voitures [5]. Une solution est de choisir "manuellement" trois services dans le registre (un par type de service) et de les intégrer en utilisant un langage de composition de services comme par exemple BPML. Cette solution est statique et ne s'adapte pas à la disparition des services utilisés ou à l'apparitation de nouveaux services dans le registre. Une autre solution est d'introduire des requêtes de services dans la définition du nouveau service : au lieu d'appeler des services prédéfinis, le nouveau service cherche d'abord les services disponibles pour ensuite en choisir celui ou ceux qu'il veut appeler. Ceci rend le nouveau service plus indépendant des autres services et s'adapte plus facilement aux changements de son environnement.

Cette activité est encore dans ses débuts et se place dans le cadre de différentes collaborations en cours ou en préparation.

– Le développement d'un permier registre de services sur la plateforme ActiveXML est en cours [Pop03]. Ce registre peut être considéré comme une application du système ActiveXML, mais

5. Ce type de service est notamment proposé sur le site de la SNCF, http://www.voyages-sncf.com.

également comme une extension du même système pour faciliter la génération de documents intensionnels.

– D'autres collaborations au niveau national sont envisagées dans le cadre d'une Action Spécifique CNRS sur les services Web qui impliquent différents laboratoires comme le CLIPS-IMAG (M.C. Fauvet), le CNAM, France Télécom (A. Léger), le LRI (C. Reynaud, N. Spyratos), le LIRIS (M.-S. Hacid) et le PriSM (M. Bouzeghoub) (demande en cours).
– Au niveau Européen j'ai participé à une proposition de Reséau d'Excellence (NoE) sur le Web Sémantique et les Services Web. Même si cette proposition n'est pas retenue finalement, elle permet de réunir différentes équipes de recherche Européens qui s'intéressent à cette problématique.

Bibliographie

[AAA+99] Serge Abiteboul, Vincent Aguilera, Sebastien Ailleret, Bernd Amann, Sophie Cluet, Brendan Hills, Frederic Hubert, Jean-Claude Mamou, Amelie Marian, Laurent Mignet, Tova Milo, Cassio Souza Dos Santos, Bruno Tessier, and Anne-Marie Vercoustre. Xml repository and active views demonstration. Online (PDF), 1999. VLDB.

[AAC+99] S. Abiteboul, B. Amann, S. Cluet, A. Eayl, L. Mignet, and T. Milo. Active views for electronic commerce. In *Proceedings of the 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland, September 1999.

[AAT] The Art & Architecture Thesaurus. http://www.ahip.getty.edu/vocabulary/aat_intro.html.

[ABFS02a] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-based integration of xml web resources. In *International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.

[ABFS02b] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Querying xml sources using an ontology-based mediator. In *CoopIS*, 2002.

[Abi99] Serge Abiteboul. Views and xml. In *Proc. ACM Symp. on Principles of Database Systems*, 1999.

[ABM02] Serge Abiteboul, Omar Benjelloun, and Tova Milo. Web services and data integration. In *International Conference on Web Information Systems Engineering*, 2002.

[ABS99] S. Abiteboul, P. Buneman, and D. Suciu. *Data On the Web: From Relations to Semistructured Data and XML*. Morgan kaufmann, October 1999.

[ACC+97] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, and J. Siméon. Querying documents in object databases. *Int. J. on Digital Libraries*, 1(1), 1997.

[ACD+03] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business process execution language for web services. URL : ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf, may 2003.

[ACFR01] S. Abiteboul, S. Cluet, G. Ferran, and M.-C. Rousset. The xyleme project, 2001.

[ACK+00] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, K. Tolle, B. Amann, I. Fundulaki, M. Scholl, and A.M. Vercoustre. Managing rdf metadata for community webs. In *Workshop on the Web and Conceptual Modeling (WCM'2000)*, Salt Lake City, Utah, November 2000.

[ACW01] V. Aguilera, S. Cluet, and F. Wattez. Querying the XML Documents of the Web. In *SEBD*, 2001. http://www.xyleme.com/publications.html.

[AE95] Sibel Adali and Ross Emery. A uniform framework for integrating knowledge in heterogeneous knowledge systems. In *Data Engineering*, 1995. http://www.cs.umd.edu/projects/hermes/.

[AF99] B. Amann and I. Fundulaki. Integrating ontologies and thesauri to build rdf schemas. In *Proceedings of the 3rd European Conf. on Digital Libraries (ECDL'99)*, Paris, France, September 1999.

[AFS00] B. Amann, I. Fundulaki, and M. Scholl. Integrating ontologies and thesauri for rdf schema creation and metadata querying. *Intl. Journal of Digital Libraries (JODL)*, 3(3), 2000.

[AFS+01] B. Amann, I. Fundulaki, M. Scholl, C. Beeri, and A.M. Vercoustre. Mapping xml fragments to community web ontologies. In *Proceedings Fourth International Workshop on the Web and Databases (WebDB'2001)*, Santa Barbara, California, 2001.

[Agu02] V. Aguilera. *Interrogation de documents XML*. PhD thesis, Ecole National des Ponts et Chaussees, 2002.

[AH87] S. Abiteboul and R. Hull. IFO: A formal semantic database model. *ACM Transactions on Database Systems*, 12(4):525–565, December 1987.

[AK93] Yigal Arens and Craig A. Knoblock. Sims: Retrieving and integrating information from multiple sources. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993*, pages 562–563. ACM Press, 1993.

[AM01] B. Amann and A. Michard. C-web functional specification (v1.0), 2001. http://cweb.inria.fr/Resources/C-Webtm.

[Ama94] B. Amann. *Interrogation d'Hypertextes*. PhD thesis, Conservatoire National des Arts et Métiers, Paris, France, February 1994.

[Ama97] B. Amann. Integrating GIS components with mediators and CORBA. Technical Report 97-09, Cedric-CNAM, CNAM, Paris, May 1997. *ftp://sikkim.cnam.fr/pub/Reports/GISMed.ps.gz*.

[AR02] B. Amann and P. Rigaux. *Comprendre XSLT*. O'Reilly, 2002.

[Ark02] Assaf Arkin. Business process modeling language. URL : http://www.bpmi.org/bpml-spec.esp, nov 2002.

[AYF02] Sihem Amer-Yahia and Mary Fernandez. Overview of existing xml storage techniques. soumis à publication, 2002. http://www.research.att.com/ sihem/publications/SIGRECORD02.pdf.

[BBN99] M. Biezunski, M. Bryan, and S. R. Newcomb. Topic maps: Information technology – document description and markup languages. ISO/IEC 13250:2000, 1999.

[BCD89] F. Bancilhon, S. Cluet, and C. Delobel. A query language for the O_2 object-oriented database system. In *Proceedings of the Second International Workshop on Database Programming Languages (DBPL'89)*, 1989.

[BCM+02] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.

[BDF+01] P. Buneman, S. B. Davidson, W. Fan, C. S. Hara, and W. C. Tan. Keys for XML. In *Proc. WWW10*, pages 201–210, 2001.

[BDHS96] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan Suciu. A query language and optimization techniques for unstructured data. In *SIGMOD*, pages 505–516, 1996.

[Ber00] M.K. Bergman. The deep web: Surfacing hidden value. White Paper, July 2000. http://www.brightplanet.com/deepcontent/tutorials/deepweb/index.asp.

[BFHS03] T. Bultan, X. Fu, R. Hull, and J. Su. Conversation specification: A new approach to design and analysis of e-service composition. In *Twelth Intl. World Wide Web Conference (WWW2003)*, 2003.

[BGL+99] C. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-based information mediation with MIX. In *Demonstrations, ACM/SIGMOD*, pages 597–599, 1999.

[BKS02] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: optimal xml pattern matching. In *SIGMOD Conference*, 2002.

[BLCGP92] T. Berners-Lee, R. Cailliau, J.F. Groff, and B. Pollermann. World-Wide Web: The information universe. *Electronic Networking: Research, Applications and Policy*, 2(1):52–58, 1992. Meckler Publishing, CT, USA.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *The Scientific American*, May 2001.

[BLR97] C. Beeri, A. Levy, and M-C. Rousset. Rewriting Queries Using Views in Description Logics. In *Proc. PODS*, pages 99–108, Tucson, Arizona, May 1997.

[BPSMM00] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler. Extensible markup language (xml) 1.0 (second edition). W3C Recommendation, 2000. http://www.w3.org/TR/REC-xml.

[Car97] E. Carsenat. Interface d'interrogation graphique pour le web. Master's thesis, Stage IIE, 1997.

[CB00] R.G.G. Cattell and D.K. Barry. *The Object Data Standard : ODMG 3.0*. Morgan Kaufmann, 2000.

[CCS00] V. Christophides, S. Cluet, and J. Simeon. On Wrapping Query Languages and Efficient XML Integration. In *Proc. of ACM SIGMOD*, Dallas, USA, May 2000.

[CD99] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. W3C Recommendation, November 1999. http://www.w3c.org/TR/xpath.

[CDG+] H. Comon, M. Dauchet, R. Gilleron, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. draft.

[CFR+01] D. Chamberlin, D. Florescu, J. Robie, J. Simeon, and L. Stefanescu. XQuery: A Query Language for XML. http://www.w3.org/TR/xquery, February 2001.

[Chr00] V. Christophides. Community webs (c-webs): Technological assessment and system architecture, 2000. C-WEB IST-1999-13479.

[CKM02] E. Cohen, H. Kaplan, and T. Milo. Labeling dynamic xml trees. In *PODS*, 2002.

[Cla99] J. Clark. XSL Transformation (XSLT) Version 1.0. W3C Recommendation, November 1999. http://www.w3c.org/TR/xslt.

[Clu98] S. Cluet. Designing OQL: Allowing Objects to be Queried. *Information Systems*, 23(5), 1998.

[CPST02] V. Christophides, D. Plexousakis, M. Scholl, and S. Tourtounis. On labeling schemes for the semantic web. In *WWW Conf.*, 2002.

[CVV01] S. Cluet, P. Veltri, and D. Vodislav. Views in a Large Scale XML Repository. In *Proc. VLDB*, Rome, Italy, September 2001.

[cwe99] C-web. http://cweb.inria.fr/cwebproj.html, 1999.

[DCHM+01] F. van Harmelen D. Connolly, I. Horrocks, D.L. McGuinness, P.F. Patel Schneider, and L.A. Stein. DAML+OIL (march 2001) reference description. W3C Note, December 2001. http://www.w3.org/TR/daml+oil-reference.

[Dew94] M. Deway. *Classification décimale de Dewey et index*. Editions ASTED, 1994.

[DG97] O.M. Duschka and M.R. Genesereth. Answering recursive queries using views. In *Proceeding of PODS 1997*, pages 109–116, Tuscon, Arizona, 1997.

[Die82] Paul F. Dietz. Maintaining order in a linked list. In *STOC*, 1982.

[DN02] F. Dang-Ngoc. Typage de documents xml avec appels de services. Master's thesis, Univ. Paris VI, 2002.

[dub] Dublin core metadata initiative. http://dublincore.org/.

[DvHB+00] S. Decker, F. van Harmelen, J. Broekstra, M. Erdmann, D. Fensel, I. Horrocks, M. Klein, and S. Melnik. The semantic web on the respective roles of xml and rdf. *IEEE Internet Computing*, 2000.

[FABS02] I. Fundulaki, B. Amann, C. Beeri, and M. Scholl. STYX : Connecting the XML World to the World of Semantics, 2002. Demonstration at EDBT'2002.

[FCP92] Bancilhon Francois, Delobel Claude, and Kanellakis Paris. *Building an Object-Oriented Database System : The Story of O_2*. Morgan Kaufman, 1992.

[FKS01a] W. Fan, G. Kooper, and J. Simeon. A Unified Constraint Model for XML. In *Proc. WWW10*, Hong-Kong, China, May 2001.

[FKS01b] W. Fan, Gabriel M. Kuper, and J. Simeon. A unified constraint model for XML. In *World Wide Web*, pages 179–190, 2001.

[FLM98] Daniela Florescu, Alon Y. Levy, and Alberto O. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.

[Fun03] I. Fundulaki. *Intégration et Interrogation de Ressources XML pour communautés Web*. PhD thesis, CNAM, 2003. Thèse de doctorat.

[GG98] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[GLR00] F. Goasdoué, V. Lattés, and M-C. Rousset. The use of CARIN language and algorithms for information integration: The PICSEL System. *International Journal on Cooperative Information Systems*, 2000.

[GMT02] Georges Gardarin, Antoine Mensch, and Anthony Tomasic. An introduction to the e-xml data integration suite. In *8th International Conference on Extending Database Technology (EDBT)*, 2002. http://www.e-xmlmedia.com/home/.

[Gru93] T.R Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.

[Gru02] T. Grust. Accelerating xpath location steps. In *SIGMOD*, 2002.

[Gua97] N. Guarino. Understanding, Building, and Using Ontologies. *International Journal of Human and Computer Studies*, 46(2/3):293–310, 1997.

[Hal00] A. Halevy. Theory of answering queries using views. *SIGMOD Record*, 29(4):40–47, 2000.

[HK] Richard Hull and Roger King. Reference architecture for the intelligent integration of information. http://www.isse.gmu.edu/I3_Arch/.

[Hul97] Richard Hull. Managing Semantic Heterogeneity in Databases : A Theoretical Perspective. In *PODS 1997*, pages 51–61, Tuscon, Arizona, May 1997.

[HVD02] J. Heflin, R. Volz, and J. Dale. Requirements for a web ontology language. http://www.w3.org/TR/2002/WD-webont-req-20020307/, July 2002.

[IBM] IBM. Wsfl. URL : http://www.alphaworks.ibm.com/tech/wspmt.

[ICO] International Guidelines for Museum Object Information: The CIDOC Information Categories. http://www.cidoc.icom.org/guide/.

[ISO86] Documentation - Guidelines for the establishment and development of monolingual thesauri. International Organization for Standardization, 11 1986. Ref. No ISO 2788-1986.

[KAC+02] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. Rql: a declarative query language for rdf. In *WWW*, 2002.

[Kho96] H. Khou. Interface graphique pour sig en java. Master's thesis, Univ. de Versailles, 1996. Stage de Maîtrise.

[KL94] K. Knight and S. Luk. Building a large knowledge base for machine translation. In *Proceedings of AAAI-94*, 1994.

[KM00] Carl-Christian Kanne and Guido Moerkotte. Efficient storage of XML data. In *ICDE*, page 198, 2000.

[KMS02] H. Kaplan, T. Milo, and R. Shabo. A comparison of labeling schemes for ancestor queries. In *SODA*, 2002.

[KS98] V. Kashyap and A. Sheth. *Cooperative Information Systems, Trends and Directions*, chapter Semantic Heterogeneity in Global Information Systems: the Role of Metadata, Context and Ontologies. Academic Press, 1998.

[Laf00] S. Lafois. Interrogation de données structurées en arbre : application à c-web. Master's thesis, Univ. Paris VI, 2000. Stage de DEA.

[Lev01] A. Levy. Answering queries using views: a survey. *VLDB Journal*, 2001. http://www.cs.washington.edu/homes/alon/site/files/view-survey.ps.

[LM00] Andreas Laux and Lars Martin. Xupdate working draft. XML:DB Working Draft, sep 2000. http://www.xmldb.org/xupdate/xupdate-wd.html.

[LRO96] A. Levy, A. Rajaraman, and J. Ordille. Querying Heterogeneous Information Sources Using Source Descriptions. In *Proc. VLDB*, pages 251–262, Mumbai (Bombay), India, September 1996.

[LS99] O. Lassila and R.R. Swick. Resource Description Framework (RDF) model and syntax specification. W3C Recommendation, February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[MAA+03] T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, and F. Dang Ngoc. Exchanging intensional xml data. In *SIGMOD*, 2003.

[MAAV01] Laurent Mignet, Vincent Aguilera, Sebastien Ailleret, and Pierangelo Veltri. Xyro: The xyleme robot architecture. In *DIWeb*, pages 91–99, 2001.

[MACM01] Amelie Marian, Serge Abiteboul, Gregory Cobena, and Laurent Mignet. Change-centric management of versions in an XML warehouse. In *The VLDB Journal*, pages 581–590, 2001.

[Mah96] K. Mahesh. Ontology development for machine translation: Ideology and methodology, 1996.

[MBDH02] J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *18th Nat. Conf. on Artificial Intelligence (AAAI'2002)*, 2002.

[MFK01] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML Queries over Hterogeneous Data Sources. In *Proc. VLDB*, Rome, Italy, September 2001.

[MFR+00] D.-L. McGuinness, R. Fikes, J. Rice, , and S. Wilder. An environment for merging and testing large ontologies. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, 2000.

[Mig01] L. Mignet. *Contrôle des changements de données semi-structurées*. PhD thesis, CNAM, 2001.

[Mil95] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[MLM01] M. Murata, D. Lee, and M. Mani. "Taxonomy of XML Schema Languages using Formal Language Theory". In *Extreme Markup Languages*, Montreal, Canada, 2001.

[MPA+00] L. Mignet, M. Preda, S. Abiteboul, S. Ailleret, B. Amann, and A. Marian. Acquiring XML pages for a web house. In *XVIèmes Journées Bases de Données Avancées*, Blois, 2000.

[Mun00] S. Munch. *Building Oracle XML Applications*. O'Reilly, 2000.

[MvH03] D.L. McGuinness and F. van Harmelen. Owl web ontology language overview. W3C Working Draft, 2003.

[NVVH03] B. Nguyen, M. Vazirgiannis, I. Varlamis, and M. Halkidi. Organizing web documents into thematic subsets using an ontology. *VLDB journal*, 2003. special issue on "Semantic Web".

[Oas] Oasis. ebxml. URL : http://www.ebxml.org/specs/index.htm.

[odp] Open directory rdf dump. rdf.dmoz.org.

[PGMW95] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object Exchange Across Heterogeneous Information Sources. In *Proc. ICDE Conf.*, March 1995. TSIMMIS project: http://www-db.stanford.edu/tsimmis.

[PL00] R. Pottinger and A. Levy. A Scalable Algorithm for Answering Queries Using Views. In *Proc. VLDB*, pages 484–495, Cairo, Egypt, September 2000.

[Pop03] Radu Pop. Description et intégration de services web. Master's thesis, École Polytechnique de Bucarest, 2003.

[PRS99] A. Papadopoulos, P. Rigaux, and M. Scholl. A performance evaluation of spatial join processing strategies. In F. Lochovsky R. H. Gueting, D. Papadias, editor, *Proc. of SSD'99*, Springer, LNCS 1651, pages 286–307, Hong-Kong, July 1999.

[PSS02] Peter F. Patel-Schneider and Jérôme Siméon. The yin/yang web: Xml syntax and rdf semantics. In *WWW*, 2002.

[QRS+95] Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, and Jennifer Widom. Querying semistructured heterogeneous information. In *Deductive and Object-Oriented Databases*, pages 319–344, 1995.

[Rad00] S. Radicevic. Mise en oeuvre xml d un portail web de fonds culturels. Master's thesis, CNAM, 2000.

[rel] Relax-ng homepage. http://relaxng.org/.

[Rou02] M.-C. Rousset. The semantic web needs languages for representing (complex) mappings between (simple) ontologies. *IEEE Intelligent Systems*, 17, 2002.

[RSV01] P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases*. Morgan Kaufman Publishers, 2001.

[Sha99] J. Sharma. Oracle8ispatial: Experiences with extensible databases. An Oracle Technical White Paper, May 1999.

[She99] A. Sheth. *Interoperating Geographic Information Systems*, chapter Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. Kluwer, 1999.

[SL90] A.P. Sheth and J.A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(1):183–236, March 1990.

[SOA] Simple Object Access Protocol (SOAP) 1.1. `http://www.w3.org/TR/SOAP`.

[TBMM01] H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn. XML Schema Part 1: Structures. W3C Recommendation, May 2001. http://www.w3.org/TR/XML-schema-1.

[TCS01] Y. Tzitzikas, P. Constantopoulos, and N. Spyratos. Mediators over ontology-based information sources. In *WISE*, 2001.

[TIHW01] Igor Tatarinov, Zachary G. Ives, Alon Y. Halevy, and Daniel S. Weld. Updating XML. In *SIGMOD Conference*, 2001.

[Tom98] I. Tomescu. Outils d'aide à la formulation de requêtes sur des schémas complexes. Master's thesis, Mémoire CNAM, Paris, 1998.

[TSW03] M. Theobald, R. Schenkel, and G. Weikum. Exploiting structure, annotation, and ontological knowledge for automatic classification of xml data. In *WebDB*, 2003.

[Via01] Victor Vianu. A web odyssey: From codd to XML. In *Symposium on Principles of Database Systems*, 2001.

[VSS+] K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. Meteor-s wsdi: A scalable infrastructure of registries for semantic publication and discovery of web services. *Journal of Information Technology and Management*.

[web] WebDAV resources. http://www.webdav.org/.

[Wid95] J. Widom. Research problems in data warehousing. In *Int'l Conf. on Information and Knowledge Management*, 1995.

[Wie95] G. Wiederhold. Mediaton in information systems. *ACM Computing Surveys*, 27(2):265–267, June 1995.

[WSD] Web Services Definition Language (WSDL). `http://www.w3.org/TR/wsdl`.

[Xyl01] L. Xyleme. A dynamic warehouse for xml data on the web. *IEEE Data Eng. Bulletin*, 24(2):40–47, 2001.

[ZND+01] Chun Zhang, Jeffrey F. Naughton, David J. DeWitt, Qiong Luo, and Guy M. Lohman. On supporting containment queries in relational database management systems. In *SIGMOD*, 2001.