XML et les systèmes d'intégration de données

Zohra Bellahsène — Xavier Baril

UMR 5506 CNRS - Université Montpellier II
LIRMM - Laboratoire d'Informatique, de Robotique et de Micro-électronique
161, Rue Ada - F-34392 Montpellier cedex 5
{*bella,baril*}*@lirmm.fr*

RÉSUMÉ. *La comunauté Bases de données n'est pas seulement intéressée par XML en tant que langage d'échange, mais aussi comme modèle de représentation des données. En effet, en associant un modèle de données à XML on peut décrire non seulement la structure syntaxique, mais aussi la sémantique des données semi-structurées. Par ailleurs, l'existence de langages de requêtes permet l'interrogation des documents XML sur leur structure et leur contenu, ce qui présente une avancée importante pour la recherche d'information sur le web. Cependant, étant donné la grande hétérogénité des sources de données, cette interrogation par le contenu n'est possible qu'en fournissant une vision unifiée des sources de données. Cette fonctionnalité est fournie par les sytèmes d'intégration de données. Mieux encore, l'objectif de ces systèmes est de fournir un accès uniforme et qui rend transparent la localisation et l'hétérogénéité des sources de données. Dans cet article, nous proposons une synthèse des qualités du langage XML et de son aptitude à représenter des données semi-structurées. Nous étudions également le rôle d'XML dans les techniques d'intégration. Enfin, pour mieux comprendre le fonctionnement des systèmes d'intégration, nous présentons quelques exemples de prototypes en cours de développement et de plates-formes déjà sur le marché.*

ABSTRACT. *The database community is interested by XML not only as the new standard for exchanging data on the web but also as a data model. Indeed, associating a data model to XML allows to describe both syntactic structure and semantics of semistructured data. These semantics are useful for querying XML documents by content. However, due to the heterogeneity of the data sources, querying XML documents by content is not possible unless providing a unified view of all the data sources. This feature is provided by data integration systems. Furthermore, the aim of these systems is providing a uniform access that hides both the heterogeneity and the localization of the data sources. In this paper, we propose a synthesis of the XML language qualities and show how it is adequate for representing semistructured data. Furthermore, we analyze the role of XML in the integration techniques. Moreover, we present some prototypes to better understand the principles and the foundation of these systems.*

MOTS-CLÉS : *XML, données semi-structurées, intégration de données.*

KEYWORDS: *XML, semistructured data, data integration.*

1. Introduction

XML est le langage standard émergeant adopté par le W3C (*World Wide Web Consortium*) [W3C], pour la représentation et l'échange de données sur le web. Il permet de séparer la présentation, le contenu et le stockage des données, contrairement à HTML. Il est capable de représenter des données avec une structure irrégulière.

Une grande partie des données présentes sur le web proviennent de bases de données et sont formatées en HTML pour être publiées. Cependant cette solution n'est pas satisfaisante car le traitement des données est rendu difficile, à cause de la confusion entre la présentation et le contenu des données. XML [XML 00a] (*eXtensible Markup Language*) apporte une solution à ce problème en séparant le contenu et la présentation des documents. XML décrit la structure logique et le contenu du document, sans se charger de la présentation. Il utilise pour cela un langage de présentation appelé XSL (*Xml Stylesheet Language*). Son universalité et sa simplicité expliquent son succès grandissant, qui feront de lui dans un futur proche «le langage du web».

La séparation de la structure logique et de la présentation du contenu permet de voir, en un document XML, un contenu logique et structuré, analogue aux bases de données. La structure logique est obtenue en XML en encadrant le contenu par des balises, sans obéir nécessairement à un schéma prédéfini, comme pour les données semi-structurées. Par conséquent, on peut voir un document XML comme une base de données semi-structurées. De plus, le contenu d'un document XML est indépendant du stockage des données. On peut donc dire qu'il garantit l'indépendance des données.

1.1. *Principe ou problématique des systèmes d'intégration de données*

Le développement d'applications utilisant intensivement le web est très difficile à cause du grand nombre de sources de données et de l'hétérogénéité existant à différents niveaux : au niveau des protocoles, des formats de données et des langages de recherche d'information. En effet, d'une part la quantité d'informations disponibles au travers du web croît d'une façon exponentielle. D'autre part, la recherche aussi bien que l'exploitation de ces informations deviennent extrêmement difficiles. De plus, l'information sur le web est placée d'une façon indépendante par différentes organisations. Par conséquent, les documents contenant ces informations peuvent apparaître dans des sites web différents et dans des formats différents.

L'objectif de l'intégration de données est de fournir une vue unifiée des différentes sources de données avec une seule interface, un seul schéma et un seul langage de requêtes. L'intégration est nécessaire dans la majorité des applications utilisant le web ; citons comme exemple la construction de site web culturel à partir de bibliothèques digitales, de places de marchés pour le commerce électronique, etc.

Une architecture générique et normalisée DARPA I3 [WID 92] a été proposée. Cette architecture s'articule autour de trois couches : la couche données, la couche médiation

et la couche application qui offre des services d'interrogation pour les clients du système.

La couche données résoud les problèmes d'extraction de données dans des sources hétérogènes. L'extraction des données consiste à activer une requête sur la source qui spécifie les données exportables par cette source. Ces données sont ensuite transformées en format commun par un wrapper.

La couche médiateur fournit une vue intégrée des différentes sources et décompose des requêtes sur le schéma médiateur en sous-requêtes sur les schémas locaux. La majorité des travaux de recherche actuels et des prototypes travaillent avec l'hypothèse que le modèle pivot est XML. Le schéma médiateur est défini grâce à un mécanisme de vues. Dans les prototypes actuels, le schéma global est appelé schéma médiateur. Ce dernier fournit une «vue» uniforme de sources multiples et hétérogènes, ainsi l'utilisateur pose ses requêtes sur ce schéma sans connaître les sources sous-jacentes. L'interopérabilité s'effectue donc par l'utilisation d'un langage de requêtes.

Un des problèmes importants dans ce contexte concerne la reformulation de requêtes pour déterminer quelles sont les sources de données susceptibles de fournir une réponse à une requête utilisateur qui, elle, est posée sur le schéma médiateur défini comme un ensemble de vues. Il s'agit donc de réécrire les requêtes en utilisant des vues [BEE 97].

1.2. *Motivations et objectifs*

La comunauté Bases de données n'est pas seulement intéressée par XML en tant que langage d'échange, mais aussi comme modèle de représentation des données. En effet, en associant un modèle de données à XML on peut décrire non seulement la structure syntaxique, mais aussi la sémantique des données semi-structurées.

De nombreux travaux ont été consacrés à la définition de modèles de données semi-structurées [PAP 95a, SUC 98a, MCH 97]. Aujourd'hui XML semble s'imposer pour représenter de telles données [SUC 98b]. De nombreux prototypes capables de traiter des données semistructurées provenant de sources hétérogènes ont été développés ces dernières années [GAR 95, LAH 99]. XML a une influence très importante sur ces différents travaux, c'est ce que nous montrons dans cet article. L'objectif de l'article est de faire le point sur l'apport d'XML dans la communauté Base de données, et dans le domaine des systèmes d'intégration de données nous étudions plus particulièrement le rôle d'XML dans les techniques d'intégration de données.

L'article est organisé comme suit. Les données semi-structurées et XML sont présentées dans la section 2. Le modèle de données et les langages de requêtes associés sont aussi présentés dans cette section. La section 3 présente les problèmes liés à l'intégration de données hétérogènes et montre le rôle de XML dans ce contexte. La section 4 présente les différentes approches pour intégrer des données et donne des exemples de

prototypes de systèmes d'intégration de données. Enfin, la section 5 contient la conclusion.

2. XML et les données semi-structurées

2.1. *Principes du langage XML*

Dans cette sous-section nous présentons succinctement les principes généraux du langage XML ainsi qu'un modèle de données. Nous donnons également quelques éléments de syntaxe et des exemples simples de documents XML. Pour une description plus complète, on pourra se référer aux ouvrages [MIC 98, ABI 99, GAR 99a], ou à la norme officielle du langage [XML 00a], disponible sur le site web du W3C.

Comme HTML, XML est un langage de balisage. Les balises permettent de structurer le document en *éléments*. Contrairement à HTML, il n'y a pas de balises prédéfinies, l'auteur d'un document définit ses propres balises. Les éléments peuvent éventuellement être complétés par un ou plusieurs *attributs*. Un attribut est une paire (nom, valeur) contenant une information relative à l'élément.

Le langage XML distingue deux classes de documents : les documents *bien formés* et les documents *valides*. Les documents bien formés respectent les règles syntaxiques du langage XML, les documents valides sont en plus conformes à un type de document.

2.2. *Les données semi-structurées et le modèle OEM*

Les données du web sont des données semi-structurées : le schéma et les données sont mélangées, et leur structure est bien souvent irrégulière. Les données semi-structurées présentent les caractéristiques suivantes :

– elles sont *auto-décrites*, c'est-à-dire que le schéma et les données sont mélangés. Classiquement, quand on veut stocker ou traiter des données par programme, on décrit d'abord la structure (type, schéma), puis on crée de nouvelles instances de ce type. Avec les données semi-structurées, il n'y a pas de séparation entre la description et les données : la description est contenue dans les données ;

– leur structure est *irrégulière*, c'est à dire que la description de plusieurs entités d'un même type peut être différente. Par exemple, une adresse peut être décrite par une rue et une ville, alors qu'une autre adresse sera décrite par une résidence, un bâtiment, une rue et une ville.

Les modèles de données semi-structurées présentent de nombreux avantages. Ils permettent de représenter aisément des données issues de sources hétérogènes. Ces sources de données appartiennent souvent à des organisations externes, et leur structure est seulement connue partiellement, et peut changer sans notification. De telles données pourraient être modélisées comme des données orientées objet, mais leur structure est irrégulière : certains objets peuvent avoir des attributs manquant, multivalués, ou un

même attribut peut avoir un type différent selon l'objet dans lequel il est présent. Les données semi-structurées permettent de représenter des données pour lesquelles il est difficile de définir un schéma fixe. Par exemple, les données bibliographiques au format bibtex sont des données semi-structurées. Un des apports non moins important d'XML pour les applications du web est la possibilité de gérer l'information indépendamment de tout programme.

Le modèle OEM a été un des premiers modèles de données dédié à la représentation des données semistructurées. Il a été introduit à l'origine pour le projet TSIMMIS (*The Standford-IBM Manager of Multiple Informations Sources*) [GAR 95], un système d'intégration d'informations. Il est possible de voir OEM comme un modèle orienté objet, car le concept fondamental d'OEM est l'objet. Un objet OEM a quatre composants :

– un identifiant (oid),

– une étiquette qui indique ce que l'objet représente,

– un type qui peut être soit de type *complexe*, soit un type atomique comme *string*, *int* …

– une valeur qui peut être soit atomique, soit un ensemble d'objets dans le cas d'un objet complexe.

OEM ou des variantes du modèle ont été utilisées dans de nombreux projets. Par conséquent, OEM est devenu le standard *de facto* pour les données semi-structurées. Bien qu'OEM soit un modèle dit « objet », l'aspect comportement n'est pas présent dans le modèle.

2.3. *Un modèle de données XML*

La syntaxe d'XML est bien adaptée pour représenter des données semi-structurées. En effet, dans le langage XML, les éléments sont organisés sous forme d'arbre. Par conséquent, XML peut facilement représenter des données semi-structurées quand leur graphe est un arbre. Un mécanisme permet de relier des nœuds de l'arbre pour passer à un graphe : c'est le typage des attributs. Le type `ID` spécifie que la valeur de l'attribut permet d'identifier de manière unique l'élément. Le type `IDREF(S)` spécifie que la valeur de l'attribut permet de référencer un (plusieurs) élément(s) existant dans le document. C'est ce mécanisme de liens intra-document qui permet d'étendre l'arbre des éléments en un graphe, et représenter ainsi un graphe de données semi-structurées. Cependant, la spécification d'XML ne comporte que des règles syntaxiques et il est nécessaire d'associer un modèle de données à XML pour ajouter une sémantique aux données. Cette sémantique sera utilisée pour l'interrogation et l'intégration de documents XML.

Il n'y a pas encore de normes pour le modèle de données associé à XML. Les modèles de données proposés pour XML considèrent le document comme un graphe orienté, étiqueté ayant une racine. Les variations concernent : le placement des étiquettes sur

les nœuds ou les arcs du graphe ; la prise en compte de l'ordre des éléments ; la distinction entre sous-élément et attribut ; la représentation des attributs IDREF.

Nous présentons ici le modèle de données que nous avons défini dans [BAR 00] pour notre modèle de vues. Les données XML sont représentées par un graphe ordonné ayant une racine. Nous avons choisi un modèle ordonné, car les éléments dans un document XML sont naturellement ordonnés par leur ordre d'apparition. Nous avons choisi de distinguer les attributs et les sous-éléments, afin que le modèle de données respecte bien les choix de conception du document. Enfin, notre modèle représente les attributs de type `IDREF(S)` comme des liens entre les éléments, ce qui permet de retrouver facilement les éléments référencés. Le modèle de données est donc un graphe G défini comme suit :

– G a deux sortes de nœuds :

- les nœuds qui représentent les éléments du document. Ces nœuds sont étiquetés avec les attributs de l'élément, sauf pour les attributs IDREF(S) ;

- les nœuds qui représentent les chaînes de caractères. Ces nœuds sont toujours des nœuds fils, possèdent une valeur et ne sont pas étiquetés ;

– G a trois sortes d'arcs :

- les arcs qui représentent les liens de composition, c'est-à-dire la structure d'arbre des éléments du document. Ces arcs sont étiquetés avec les noms des éléments ;

- les arcs qui représentent les liens de référence, c'est-à-dire les liens vers des éléments partagés à l'aide des attributs de type `IDREF(S)`. Ces arcs sont étiquetés avec le noms des attributs `IDREF` ;

- les arcs qui représentent les liens vers des éléments chaînes de caractères. Ces liens ne sont pas étiquetés ;

– G a un nœud racine qui représente l'élément racine du document.

La figure 1 présente un exemple de données XML avec le graphe associé. Les nœuds éléments sont représentés par des cercles, l'ordre des nœuds est noté à l'intérieur du nœud. Les nœuds chaînes de caractères sont représentés par leur valeur. Les liens de composition sont représentés par des flèches avec un trait plein et une étiquette. Les liens de référence sont représentés par des flèches avec un trait en pointillé et une étiquette. Les liens vers des chaînes de caractères sont représentés par des flèches en trait plein, sans étiquette. Pour une meilleure lisibilité, certaines parties du graphe ne sont pas dessinées et remplacées par des points de suspension.

Le tableau 1 présente les similitudes entre le graphe OEM et notre modèle de données pour XML. On voit que le graphe XML permet de représenter les caractéristiques du graphe OEM en rapprochant les notions d'objet OEM et d'élément XML, avec les liens définis sur le graphe XML. Au niveau du typage, on voit qu'OEM est plus riche qu'XML, car seul le type chaîne de caractères est présent pour XML. Toutefois, XML-Schéma [XML 00c], présenté plus haut, permet de typer plus finement le contenu des

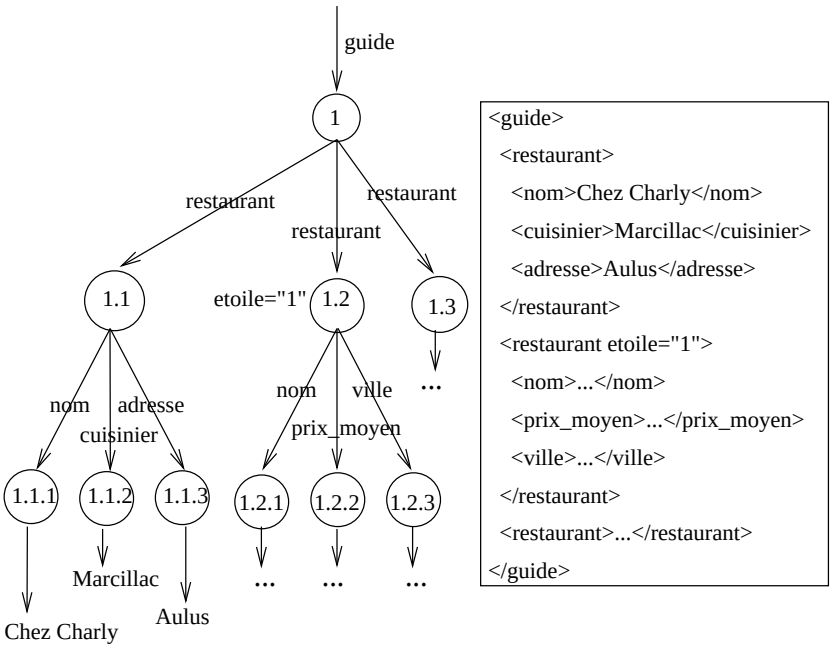

Figure 1. *Exemple de graphe XML*

Graphe OEM	*Graphe XML*
Objet	Elément
Identifiant d'objet : oid	Attribut de type `ID` identifiant un élément
Lien entre deux objets	Lien de composition ou lien de référence ou lien chaîne de carcactères
Point d'entrée dans le graphe	Elément racine
Type atomique : string, int, float, etc.	Chaîne de caractères seulement
Pas de mécanisme correspondant	Différenciation entre attribut et sous-élément

Tableau 1. *Similitudes entre le graphe OEM et notre graphe XML*

éléments XML. Enfin, XML permet de différencier attribut et sous-élément, il n'y a pas de mécanisme équivalent avec OEM.

2.4. *Langages de requêtes pour XML*

Pourquoi un langage de requêtes pour des documents XML ? Le succès d'XML laisse présager que bientôt beaucoup de données du web seront au format XML. Il devient alors important de pouvoir interroger ces données. Jusqu'à présent, les requêtes sur les données du web (à partir de moteurs de recherche), fournissaient comme résultat un ensemble de pages HTML, en utilisant des mécanismes de recherche par mot-clé. Aucune interrogation sur la structure des pages n'était possible. En associant un modèle de données à XML, on peut interroger les données XML sur leur structure et leur contenu, ce qui présente une avancée notoire pour la recherche d'information sur le web. De plus, les langages de requête pour XML sont un élément important des systèmes d'intégration de données du web pour deux raisons :

1. l'utilisateur exprime ses requêtes sur un schéma XML et ignore totalement le format et la localisation des données sources ;

2. l'existence de langages de requête sert de support pour la définition de vues sur les sources de données. En effet, aujourd'hui tous les systèmes d'intégration utilisent le mécanisme de vues pour définir leur schéma médiateur comme une ensemble de vues sur lequel l'utilisateur peut formuler des requêtes.

Pour l'instant, il n'existe pas de norme pour un langage d'interrogation de documents XML. Cependant un groupe de travail du W3C (*XML Query Working Group*) y travaille activement. Ce groupe a proposé une liste de *desiderata* concernant un futur langage de requête pour XML [XML 00b] que nous rapportons ci-dessous et une proposition pour un langage de requête qui est actuellement à l'étude [CHA 01].

1. *Déclarativité* : le langage doit être déclaratif et ne doit pas impliquer une stratégie d'évaluation particulière.

2. *Disponibilité du schéma* : le langage doit pouvoir exprimer une requête, même s'il n'y a pas de schéma associé aux données (on entend ici par schéma une DTD ou un XML-Schéma).

3. *Exploitation du schéma* : le langage doit exploiter le schéma s'il est présent.

4. *Préservation de la structure* : le langage doit préserver la structure hiérarchique des documents.

5. *Transformation de la structure* : le langage doit être capable de transformer la structure, ou de créer de nouvelles structures hiérarchiques.

6. *Fermeture* : le langage doit produire en sortie des documents XML, avec le même modèle de données que celui utilisé en entrée.

3. Rôle de XML dans les systèmes d'intégration de données

Dans cette section, nous analysons le rôle de XML dans les systèmes d'intégration de données. Tout d'abord, nous étudions sa capacité à jouer le rôle d'un modèle commun. Ensuite, nous examinerons son rôle dans le processus d'intégration proprement

dite ainsi que ses avantages et ses faiblesses pour résoudre les problèmes de conflits structurels et sémantiques.

3.1. *Propriétés d'un modèle commun*

Afin de faciliter cette intégration, on procède à la traduction de schéma. Ce processus de pré-intégration de schéma, consiste à traduire les schémas des différentes sources dans un même modèle appelé modèle de données commun. La traduction des schémas des sources dans le modèle commun est capitale en vue de leur intégration. Elle a pour objectif de fournir un ensemble d'abstractions partagées, comprises et supportées par tous les modèles. La traduction de schémas est nécessaire lorsque le modèle des sources est différent du modèle de données du médiateur. Elle permet de réduire le nombre de traductions entre modèles : le nombre de traductions est de N (où N est le nombre de modèles).

Le modèle commun doit posséder les propriétés suivantes [PIT 95] :

1. il doit être suffisamment puissant pour (i) exprimer les modèles existants, (ii) capturer la sémantique exprimée explicitement et implicitement dans les différents schémas des sources de données ;

2. il doit être flexible pour pouvoir intégrer de nouveaux modèles ;

3. il doit être minimal : comprendre un ensemble de constructeurs permettant aux différents modèles de réaliser des combinaisons ;

4. il doit être simple pour faciliter la création de schéma médiateur.

Rappelons que les principaux atouts de XML sont (i) c'est un standard pour l'échange de données sur le web, il facilite l'interopérabilité et (ii) il est très flexible grâce à sa structure d'arbre qui permet de représenter n'importe quelle donnée (structurée ou semi-structurée). De plus, il est aisé de convertir des données dans un format donné en format XML et cela d'une façon générique.

Cependant, XML en tant qu'uniquement format d'échange ne peut être utilisé comme modèle commun car il ne permettrait pas de définir le schéma médiateur comme un ensemble de vues. C'est donc bien XML en tant que modèle de données et muni d'un langage de requêtes qui est utilisé aujourd'hui comme modèle commun dans les systèmes d'intégration de données.

3.2. *XML et l'intégration proprement dite*

Après la traduction des différents schémas dans un modèle commun, la phase d'intégration de schémas pourrait se résumer à faire l'union des schémas des sources si les mêmes concepts n'étaient pas représentés dans les différentes sources et éventuellement d'une façon différente si les modèles de données utilisés sont

différents. Les types de conflits classiques rencontrés lors de l'intégration sont résumés comme suit [PIT 95] :

– conflits de schéma. On distingue les conflits de noms :

- homonymes : le même nom est utilisé pour désigner des concepts différents,

- synonymes : le même concept est décrit par plusieurs noms différents ;

– conflits structurels : le même concept est représenté par des constructeurs différents du modèle de données. Même lorsque le même concept est représenté par le même constructeur, il subsiste des différences au niveau des opérations ou des relations ;

– conflits sémantiques : le même concept est interprété différemment dans les différentes bases locales ;

– conflits de données :

- identité : la même entité est représentée plusieurs fois,

- incohérence : des données décrivant la même entité sont différentes ou contradictoires.

Dans les systèmes de bases de données fédérées, l'intégration de schémas est réalisée en utilisant des méthodes d'intégration complexes [GRE 92, KAU 92, KIM 93, KLA 95, KRI 91, LAR 89, LIT 90, MAN 92, MOT 87, SHE 87]. Dans le contexte des systèmes d'intégration de données, le processus d'intégration est réalisé à deux niveaux : au niveau des wrappeurs et au niveau du médiateur.

Pour comprendre les problèmes de l'intégration de données dans le contexte du web, il faut se référer à l'architecture des systèmes d'intégration de données et commencer par comprendre le fonctionnement des adaptateurs (wrappers). Le travail des wrappers constitue véritablement une première phase d'intégration. En effet, les wrappers effectuent l'extraction des données et leur traduction dans un modèle de données cible (en général XML).

Contrairement, au contexte des bases de données fédérées, les sources sur le web n'exportent pas des données mais fournissent des capacités d'interrogation. Pour cela, il faut être en mesure de comprendre leurs langages d'interrogation afin de pouvoir les intégrer. TSIMMIS a été le premier système à fournir des moyens pour représenter les capacités d'interrogation des sources de données avec la notion de *query template* [PAP 95b]. Plus récemment, un langage de description des sources a été proposé dans le système YAT [CHR 00]. Basé sur une algèbre pour XML, celui-ci est capable de traduire des requêtes full texte ou structurées, comme par exemple des requêtes SQL ou OQL. Par ailleurs, les wrappeurs doivent extraire le type des données. La difficulté réside dans le fait que la plupart des sources ne fournissent pas le type des données à cause du manque de standard d'échange de type. En effet, les DTD semblent insuffisantes pour capturer des types riches et ne peuvent donc pas remplacer les schémas (au sens classique des bases de données). XML-Schéma pourrait constituer une solution s'il est adopté comme standard.

La phase principale d'intégration, dans les systèmes d'intégration de données basés sur XML, consiste à définir un schéma médiateur au moyen de langages déclaratifs ; par exemple, MSL dans TSIMMIS, YATL dans le système YAT et XMAS dans le système MIX. Cette approche consiste à construire le schéma médiateur comme un ensemble de vues en utilisant un langage de définition de vues déclaratif basé sur un langage de requêtes de type XML-QL. Elle est plus rapide à mettre en œuvre que les méthodes d'intégration dans les bases de données fédérées et permet une maintenance plus aisée. Cependant, XML ne permet pas d'éviter le problème de conflits structurels et de conflits sémantiques. En effet, à cause de sa flexibilité et de l'existence de la notion d'élément et d'attribut, deux documents traitant du même sujet peuvent être structurés différemment et donc contenir des balises différentes. Les DTD des documents constituent une grande aide pour résoudre le problème de conflits structurels. Dans certaines approches comme par exemple dans Xylème, l'intégration des différentes DTD permet de fournir un schéma médiateur qui donne une vue unifiée des sources de données hétérogènes.

Les problèmes de conflits sémantiques existent lorsque plusieurs documents différents peuvent traiter de la même thématique. La question est comment reconnaître qu'un document appartient à un thème donné. Par exemple, le concept de film est nommé *film* dans un document et *cinéma* dans un autre document issu d'une source de données différente. De même qu'un film peut être représenté par un document XML entier et par ailleurs par un attribut dans un autre document issu d'une autre source de données.

4. Différentes approches pour l'intégration de données

Il existe deux principales approches pour intégrer des données : l'approche virtuelle (souvent appelée approche par médiateur) et l'approche matérialisée (approche par entrepôt). La figure 2 présente ces deux approches. Dans l'approche par médiateur (a), les données sont virtuelles, c'est-à-dire qu'un accès aux sources est nécessaire pour répondre à chaque requête. Dans l'approche entrepôt (b), les données sont matérialisées, c'est-à-dire qu'elles sont dupliquées dans l'entrepôt et qu'il n'est pas nécessaire d'accéder aux sources pour répondre aux requêtes. Cependant, pour des raisons d'optimisation, il est possible de combiner l'approche virtuelle avec l'approche matérialisée, en matérialisant uniquement certaines vues. Dans ce cas (c), une partie des données est stockée au niveau du médiateur. Naturellement, la question de savoir quelles vues doivent être matérialisées est primordiale. Plusieurs heuristiques sont utilisées [CHI 01, KOT 99, BEL 00] : fréquences de requêtes, optimisation du coût de maintenance de ces dernières, optimisation du temps de réponse, espace de stockage, etc.

Après avoir décrit les deux principales approches, nous présenterons quelques prototypes de systèmes d'intégration de données. Tous ces systèmes ont en commun l'utilisation d'XML comme modèle commun. Ils fournissent tous un schéma médiateur défini comme un ensemble de vues XML.

Un mécanisme de vues permet de restructurer les données selon différents points de vues. Cela permet ainsi de personnaliser les données pour statisfaire les besoins des utilisateurs et des contraintes de sécurité (masquer des données confidentielles). De plus, dans l'environnement semi-structuré, les vues permettent d'ajouter une structure aux données facilitant ainsi l'optimisation de requêtes et l'utilisation de langages de programmation pour le développement d'applications.

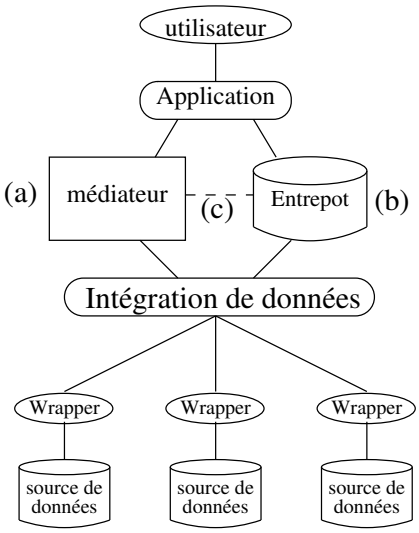

Figure 2. *Architecture générale : (a) approche médiateur (b) approche entrepôt (c) approche mixte*

4.1. *Les principales approches*

4.1.1. *L'approche virtuelle*

L'approche virtuelle est fondée sur une hiérarchie de médiateurs, correspondant à des vues virtuelles : les données ne sont stockées que dans leur source d'origine. L'utilisateur a une «vue intégrée» des différentes sources. Il exprime ses requêtes sur un schéma médiateur qui correspond en fait à un ensemble de vues décrivant les sources de données accessibles *via* ce médiateur. Ces requêtes sont adressées au médiateur qui les décompose et les envoie après les avoir optimisées, aux différentes sources concernées. L'avantage de cette approche réside dans le fait qu'elle n'entraîne pas de coût de maintenance étant donnée que les vues sont virtuelles au niveau du médiateur. En contrepartie, le temps de réponse des requêtes est pénalisé par le fait de la distribution des données et de la nécessité de recomposer les résultats des différentes sous-requêtes rendues par les sources avant de présenter un résultat global à l'utilisateur.

4.1.2. *L'approche matérialisée*

Dans les approches matérialisées, les données sont effectivement extraites (par copie sélective [1]), nettoyées [GAL 00], intégrées et stockées dans un entrepôt. Les requêtes utilisateurs sont traitées directement par l'entrepôt de données sans accéder aux sources. Les avantages d'une telle approche sont : l'amélioration du temps de réponse des requêtes et la disponibilité permanente des données. En contrepartie, elle entraîne un coût de maintenance élevé en cas de mise à jour des sources. A l'origine, les entrepôts de données ont été implémentés sur des SGBD relationnels pour des raisons d'efficacité et aussi parce que la principale application est l'aide à la décision dans les entreprises. Aujourd'hui on voit apparaître de nouvelles applications puisant leurs données sur le web. Cependant, il est très difficile de développer des applications car les pages du web changent en permanence. D'où l'idée de construire des entrepôts où les données stockées sont plus stables (néanmoins à jour).

4.2. *Différents systèmes d'intégration*

4.2.1. *Une approche virtuelle : MIX*

Le système MIX [MIX] (*Mediation of Information using XML*) a été développé pour la médiation de sources de données hétérogènes. Dans ce projet, le web est considéré comme une grande base de données distribuée. L'architecture du système est classique, utilisant des wrappers et un médiateur. XML est utilisé comme modèle commun pour intégrer les données. Nous présentons ici les rôles des composants dans la résolution des problèmes d'intégration.

Les wrappers permettent d'avoir une vue logique en XML d'une source d'information (relationnelle, collection de pages HTML, base de données hiérarchique). Pour cela, des techniques permettant de représenter en XML des sources de données décrites dans un autre format, par exemple relationnel [BAR 99], ont été développées. Un wrapper est aussi capable de traduire une requête XMAS vers un langage compréhensible pour la source concernée et de traduire le résultat de cette requête en XML. Ces wrappers permettent de résoudre les problèmes syntaxiques et de fournir un schéma XML (sous forme de DTD) de la source concernée.

Le schéma médiateur du système est construit comme un ensemble de vues. Ces vues sont définies avec le langage XMAS [LUD 99] qui est fortement inspiré de XML-QL. Ce langage de définition de vues possède la caractéristique de générer une DTD, ce qui permet d'avoir un schéma XML des sources de données intégrées. Enfin, une interface graphique (BBQ) a été développée pour faciliter la définition des vues. Cette interface graphique utilise les DTD fournies par les wrappers pour guider l'utilisateur dans la conception de la vue.

---

1. Les données copiées sont éventuellement filtrées par des requêtes.

4.2.2. *Une approche à la carte basée sur des composants : e-XML média*

Une société issue de la recherche, e-XMLMédia [XML], a vu le jour assez récemment. Elle propose des composants dédiés à l'intégration, utilisant XML comme modèle commun [GAR 99b]. Ces technologies ont été développées à l'origine dans le cadre du projet Miro-Web [FAN 98]. Les composants commercialisés permettent de répondre aux problèmes d'intégration de sources de données hétérogènes et de stockage de données XML. Ils peuvent fonctionner de manière indépendante et sont au nombre de trois. Nous les présentons ici.

– *e-XML Mediator* est un outil de requêtes sur des sources de données multiples et hétérogènes. En fédérant ces sources, il interroge et présente de manière homogène toutes les données accessibles. Le schéma médiateur est défini par des métadonnées, peu de détails sont donnés à ce propos dans la documentation. Les résultats des requêtes sont fournis sous forme de documents XML dont la structure est définie dans les requêtes. Le langage utilisé est XQuery. Le composant possède également une interface graphique pour l'aide à la spécification des requêtes.

– *e-XMLizer* joue le rôle de wrapper. Ce composant permet l'extraction de données, leur transformation au format XML et l'insertion d'information XML dans des tables relationnelles. C'est en fait un wrapper pour des sources relationnelles qui se configure à l'aide de scripts SQL pour l'extraction des données et de règles de gestion pour l'insertion de données XML dans des tables relationnelles. Ce composant résoud donc les problèmes d'intégration syntaxique. De plus, la définition de requêtes SQL pour l'extraction de données permet de résoudre certains conflits de noms en renommant des attributs.

– *e-XML Repository* permet de stocker et d'interroger des documents XML dans une base de données relationnelle. Le stockage peut être réalisé de manière générique pour des documents dont la structure n'est pas connue *a priori*. Lorsque la structure du document est connue (XML-Schéma), le composant peut générer un schéma relationnel spécifique pour le stockage. Au niveau de l'interrogation, deux solutions sont possibles. Le composant permet d'interroger les documents en SQL (cela nécessite toutefois de connaître le schéma utilisé pour le stockage du document) ou en XQuery.

Ces composants indépendants peuvent être mis en place pour construire un système d'intégration de données. Ils permettent de mettre en œuvre une approche virtuelle ou matérialisée.

4.2.3. *Stocker toutes les données XML du web : Xylème*

Xyleme [Xyl] est un système d'entrepôt de données qui a pour objectif de stocker toutes les données XML du web et de les maintenir à jour. Même si cet objectif semble utopique, les problèmatiques qu'il soulève sont intéressantes. De plus, du point de vue pratique, il est possible de construire des entrepôts spécialisés (appelés *datamart*) pour les besoins spécifiques d'une ou plusieurs applications. Xyleme est un projet de recherche entre plusieurs équipes (INRIA, U. de Mannheim, LRI, laboratoire CEDRIC du CNAM) qui a débouché sur une start-up proposant des services sur internet. Tout

d'abord, les acteurs du projet proposent de nouvelles techniques de stockage et d'aquisition de pages XML [MIG 00]. Les pages sont stockées dans un SGBD natif (NATIX [KAN 99]) dédié aux données XML. Maintenir ces données à jour (ou le plus possible !) est aussi une tâche ardue [MAR 00]. Les pages sont rafraîchies plus ou moins fréquemment, principalement selon trois critères : (1) les pages importantes (beaucoup référencées) sont rafraîchies plus fréquemment ; (2) les pages changeant très rarement (pages d'archives) sont rafraîchies peu fréquemment ; (3) les pages qui changent très fréquemment sont rafraîchies peu fréquemment car il est impossible de les maintenir à jour dans l'entrepôt (les cours de bourse par exemple peuvent être rafraîchis chaque minute).

Le schéma médiateur dans Xylème est défini par un mécanisme de vues et correspond à une structure qui donne un résumé d'un domaine d'intérêt pour un groupe d'utilisateur [REY 01, CLU 01]. Une vue dans Xylème est définie par un couple $\{C, C'\}$ où :

– C est un chemin dans la DTD abstraite virtuelle,

– C' est un chemin dans une DTD des documents stockés dans l'entrepôt.

Le choix de chemin à chemin permet de préserver le contexte d'interprétation des nœuds abstraits et concrets. L'intégration sémantique dans Xylème est basée sur une relation de correspondance entre les chemins du type de l'arbre abstrait modélisant le schéma médiateur et des chemins des types de l'arbre concret servant de schéma pour les données stockées dans l'entrepôt. Pour chaque domaine, l'intégration sémantique est réalisée par la définition de correspondances entre les éléments des DTD concrètes et la DTD (abstraite) du domaine.

Les données et les structures de données sont représentées sous forme d'arbre. Le schéma médiateur est représenté également sous forme d'arbre. Ce choix est motivé par le besoin de rendre compatible le schéma médiateur avec une DTD XML concrète et de fournir une interface simple et visuelle sur laquelle l'utilisateur peut formuler des requêtes. Par ailleurs, aucune sémantique spécifique n'est attachée aux liens internœuds de l'arbre, excepté le fait qu'un nœud fils doit être interprété dans le contexte du sens de son nœud parent. Ceci permet de résoudre les problèmes de conflit de noms. En effet, un même nom d'une propriété ou d'un concept peut apparaître plus d'une fois sans qu'il y ait d'ambiguïté car leur interprétation est faite relativement au domaine de leur nœud parent.

4.2.4. *Définir un entrepôt comme un ensemble de vues*

Dans cette section, nous présentons un deuxième exemple d'entrepôt de données XML. Cependant, en termes de fonctionnalités, c'est un prototype qui est en cours de développement, il est moins complet que les autres systèmes présentés dans la même section. L'architecture de cette approche (figure 3), suit les recommandations proposées par Gio Widerhold [WID 92] pour l'architecture DARPA I3. L'entrepôt de données est défini ici comme un ensemble de vues XML.

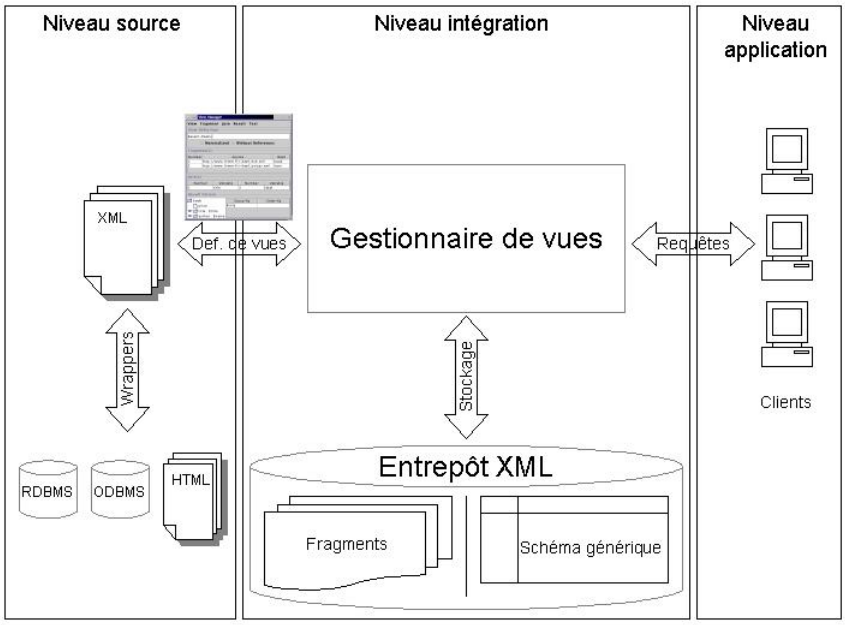

Figure 3. *Architecture d'un entrepôt de données XML*

Nous avons défini un modèle de vues pour XML [BAR 00], consistué d'un modèle de données et d'un langage de définition de vues.

Nous supposons que nos sources sont capables d'exporter leurs données au format XML. Cette hypothèse est tout à fait raisonnable étant donné l'essor actuel d'XML. L'intégration est rendue possible par le fait qu'une vue peut être définie sur plusieurs documents XML et que chacun des documents peut être issu d'une source différente. Une telle vue sera exprimée avec une jointure entre plusieurs documents, c'est l'originalité de notre modèle de vues. C'est l'ensemble des ces vues qui constitue le schéma médiateur de l'entrepôt. Cependant, les conflits sémantiques sont supposés résolus en amont avant la définition des vues.

Ce langage de vues, proche de XML-QL est déclaratif. Afin d'aider l'utilisateur dans la tâche de spécification des vues, nous avons développé un outil graphique convivial [BAR 01] ne nécessitant pas la connaissance du langage de définition des vues. L'originalité de cet outil est de fournir une aide au concepteur de la vue, pour définir les données à retrouver dans les sources. Ces aides permettent au concepteur, s'il ignore la structure des sources, de la découvrir au fur et à mesure de la spécification de la vue. Ces aides sont basées sur deux sources de données : une extension du concept de *dataguide* pour XML et la DTD du document source (si elle existe).

Enfin, au niveau du stockage, nous utilisons la métamodélisation pour stocker les éléments du résultat de la vue dans un SGBD relationnel. Les vues sont décomposées en fragments, le résultat de chaque fragment étant représenté par une relation. Un fragment est une partie de la définition de la vue correspondant à une forme définie sur une source. Il y a autant de fragments dans une vue que de sources associés à cette vue. Un fragment définit des variables qui sont associées à des chemins dans le document source. Dans l'entrepôt, chaque fragment est stocké dans une relation. Chaque colonne de la relation correspond à une variable du fragment. Chaque ligne de la relation correspond à une instanciation des variables dans la source. Les éléments XML résultant de l'instanciation des variables sur les sources sont stockés dans une base de données utilisant un schéma générique, décrit dans la figure 4.

$Document(\mathbf{docID}, url)$
$Element(\mathbf{elemID}, type, docID)$
$Attribut(\mathbf{attID}, name, docID)$
$XmlNode(\mathbf{xmlNodeID}, elemNodeID, textNodeID)$
$ElementNode(\mathbf{elemNodeID}, elemID)$
$TextNode(\mathbf{textNodeID}, value)$
$AttributNode(\mathbf{attID}, \mathbf{elemNodeID}, \mathbf{xmlNodeID}, type)$
$Children(\mathbf{elemNodeID}, \mathbf{xmlNodeID}, rank)$

Figure 4. *Schéma générique pour le stockage des éléments XML*

Cette métamodélisation est basée sur la structure des données à stocker. La relation $Document$ contient les URL des sources de données. Les relations $Element$ et $Attribut$ sont des dictionnaires qui contiennent respectivement les types d'éléments et d'attributs existant dans les données. Ces dictionnaires, associés à la création d'index sur les clés étrangères permettent d'accélérer les recherches pour l'interrogation. Les nœuds éléments et les nœuds textes du documents sont stockés dans les relations $ElementNode$ et $TextNode$. La relation $XmlNode$ généralise la notion de nœud élément et texte dans un document XML, pour permettre de les référencer indifféremment. La relation $Children$ permet de conserver la hiérarchie et l'ordre des nœuds dans le document. Les attributs sont stockés dans la relation $AttributeNode$ contenant le type de l'attribut, le nœud élément auquel il appartient et sa valeur. La valeur d'un attribut `CDATA` est un nœud texte, alors que la valeur d'un attribut `IDREF` est un nœud élément. Les attributs `IDREFS` sont multivalués, ils sont représentés par plusieurs lignes dans la relation $AttributeNode$, une ligne correspondant à chaque valeur.

4.3. *Comparaison des différentes approches*

Les principales problématiques de recherche concernant les systèmes d'intégrations utilisant XML sont :

– la définition de mécanismes permettant de traduire les données décrites dans un format donné (fichier, relationnel, objet, HTML, etc.) en XML. C'est la fonction des wrappers,

– la construction du schéma médiateur de ces systèmes,

– l'intégration sémantique des données,

– le stockage des données (dans un SGBD ou un système natif XML).

Dans cette sous-section nous discutons ces problématiques et comparons les propositions que nous avons présentées.

L'écriture des wrappers est une tâche répétitive, et des efforts d'automatisation ont été faits. MIX propose un mécanisme pour définir des vues XML sur des données relationnelles [BAR 99]. XML-Média propose un langage de scripts à base de règles pour générer des wrappers. Cette traduction n'est pas considérée dans Xylème ni dans notre approche car les données à intégrer sont déjà au format XML. Cette étape permet de résoudre l'intégration syntaxique, en utilisant XML comme modèle de données commun.

La définition du schéma médiateur du système permet de fournir une vue unifiée des différentes sources de données. Ce schéma est généralement défini par un ensemble de vues. Ces mécanismes de vues sont fondés sur les langages de requêtes pour XML. Par exemple, XMAS et notre langage de vues [BAR 00] sont fortement inspirés par XML-QL [DEU 99].

Dans Xylème, le schéma médiateur est constitué par l'ensemble des DTD virtuelles des différentes domaines. En effet, une DTD virtuelle est construite pour chaque domaine. Enfin, dans XML-Média le schéma médiateur est construit à l'aide de métadonnées sur les sources.

Plusieurs approches sont possibles pour le stockage de données XML. Tout d'abord, on peut construire un système natif dédié aux données XML. C'est l'approche qui a été utilisée dans Xylème, avec le système NATIX [KAN 99]. L'autre solution consiste à utiliser un SGBD existant pour stocker ces données. Il faut alors décrire les données à stocker dans le modèle du SGBD utilisé : cette méthode est appelée métamodélisation. Lorsque le schéma utilisé est indépendant des données XML à stocker, on dit qu'il est générique. Notre approche utilise un schéma générique pour stocker des données XML dans un SGBD relationnel. Le composant *XML-repository* offre la possibilité d'utiliser un schéma générique, ou bien, lorsque les données XML sont validées par une DTD, de construire un schéma dédié à cette DTD. Enfin, dans MIX, la problématique du stockage n'est pas abordée car le système repose sur une approche totalement virtuelle.

Le premier point commun à toutes ces approches réside dans l'utilisation d'XML comme modèle de données commun. Le second point est l'utilisation du mécanisme de

vues pour la définition du schéma médiateur (hormis dans e-XML où ce point n'est pas clairement décrit dans les documentations). Tous les systèmes étudiés n'utilisent pas la même technique pour stocker les données XML. Deux de ces systèmes (XML-Média et le nôtre) utilisent des SGBD relationnels, alors que Xylème utilise NATIX [KAN 99]. Finalement, l'intégration sémantique reste encore une question ouverte même si des solutions ont été proposées dans certains systèmes étudiés (par exemple, dans Xylème).

5. Conclusion

Pour conclure nous mettons en avant les principaux apports de XML en tant que langage standard d'échange et de représentation des données ainsi que son rôle dans les sytèmes d'intégration des données. Grâce à sa structure d'arbre il permet de représenter n'importe quelle donnée (structurée ou semi-structurée). Le statut de langage standard d'XML laisse présager que bientôt beaucoup de données du web seront au format XML. Jusqu'à présent, les requêtes sur les données du web (à partir de moteurs de recherche), fournissaient comme résultat un ensemble de pages HTML, en utilisant des mécanismes de recherche par mot-clé. En associant un modèle de données à XML, on peut interroger les données XML à l'aide de langage de requêtes sur leur structure et leur contenu, ce qui présente une avancée notoire pour la recherche d'information sur le web.

En plus de sa souplesse de représentation, XML permet de modéliser des données indépendamment de leur présentation et de leur stockage. Ces deux propriétés sont nécessaires et utiles pour le développement d'applications dans des environnements ouverts comme le web. L'existence de nombreux langages et outils qui sont basés sur XML est un autre atout pour l'interopérabilité. En particulier, nous notons l'existence de XMI (*XML Metadata Interchange*). C'est un modèle d'échange de métadonnées standardisé par l'ODMG qui unifie XML, UML et MOF (*Meta Object Facilities*).

Par ailleurs, l'existence de langage de requêtes sert de support pour la définition de vues sur les sources de données. En effet, aujourd'hui tous les systèmes d'intégration de données utilisent le mécanisme de vues pour définir leur schéma médiateur comme un ensemble de vues sur lequel l'utilisateur peut formuler des requêtes.

Tous ces sytèmes utililisent XML comme modèle pivot car il est aisé de convertir des données dans un format donné en format XML et cela d'une façon générique. De nombreux wrappers capables de réaliser cette tache de traduction ont été développés. Ce n'est pas XML en tant que langage d'échange qui permet de réaliser l'intégration mais l'utilisation conjuguée du modèle de données XML comme modèle pivot et des langages de requêtes pour XML qui permet de définir une vue intégrée sur un ensemble de sources de données hétérogènes. Cependant, XML ne permet pas d'éviter les problèmes de conflits structurels et sémantiques.

Les systèmes d'intégration de données basés sur XML sont très prometteurs en termes de rapidité et d'intégration relativement peu coûteuse par rapport aux méthodes d'intégration utilisées dans les bases de données fédérées. C'est une technologie très

utile pour développer des applications sur le web. Cependant, plusieurs questions restent ouvertes ou sont en cours d'étude : intégration sémantique, résolution et optimisation de requêtes en utilisant des vues, développer et expérimenter des implémentations robustes.

6. Bibliographie

[ABI 99] ABITEBOUL S., BUNEMAN P., SUCIU D., *Data on the Web : From Relations to Semistructured Data and XML*, Morgan Kaufmann Publishers, 1999.

[BAR 99] BARU C., « XViews : XML views of relational schemas », *Proceedings of the 10th International Workshop on Database and Expert System Applications*, september 1999, p. 700-705.

[BAR 00] BARIL X., BELLAHSÈNE Z., « A View Model for XML Documents », *6th International Conference on Object Oriented Information Systems (OOIS'2000)*, Springer-Verlag, 2000, p. 429-441.

[BAR 01] BARIL X., BELLAHSÈNE Z., « A Browser for Specifying XML Views », *7th International Conference on Object Oriented Information Systems (OOIS'2001)*, Springer-Verlag, 2001, p. 164-174.

[BEE 97] BEERI C., LEVY A. Y., ROUSSET M.-C., « Rewriting Queries Using Views in Description Logics », *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 12-14, 1997, Tucson, Arizona*, ACM Press, 1997, p. 99-108.

[BEL 00] BELLAHSÈNE Z., MARROT P., « Materializing a Set of Views: Dynamic Strategies and Performance Evaluation », *International Database Engineering and Applications Symposium*, Yokohoma, Japan, September 2000.

[CHA 01] CHAMBERLIN D., CLARK J., FLORESCU D., ROBIE J., SIMÉON J., STEFANESCU M., « XQuery 1.0 : An XML Query Language », 07 june 2001, http://www.w3.org/TR/2001/WD-xquery-20010607.

[CHI 01] CHIRKOVA R., HALEVY A., SUCIU D., « A Formal Perspective on the View Selection Problem », *Proceedings of the International Conference on Very Large Databases*, Roma, Italy, 2001.

[CHR 00] CHRISTOPHIDES V., CLUET S., SIMÉON J., « On Wrapping Query Languages and Efficient XML Integration », CHEN W., NAUGHTON J. F., BERNSTEIN P. A., Eds., *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, vol. 29, ACM, 2000, p. 141-152.

[CLU 01] CLUET S., VELTRI P., VODISLAV D., « Views in a Large Scale XML Repository », *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, Morgan Kaufmann, 2001, p. 271-280.

[DEU 99] DEUTSCH A., FERNANDEZ M., FLORESCU D., LEVY A., SUCIU D., « A Query Langage for XML », *Proceedings of the 8th International World Wide Web Conference*, 1999, p. 1155-1169.

[FAN 98] FANKHAUSER P., GARDARIN G., LOPEZ M., MUNTZ J., TOMASIC A., « Experiences in Federated Databases: From IRO-DB to MIRO-DB », *Proceedings of the 24th International Conference on Very Large Data Bases*, 1998, p. 655-658.

[GAL 00] GALHARDAS H., FLORESCU D., SHASHA D., SIMON E., « Declaratively cleanning your data using AJAX », *Proceedings of the 16th journées Bases de Données Avancées*, 2000, p. 97-116.

[GAR 95] GARCIA-MOLINA H., HAMMER J., IRELAND K., PAPAKONSTANTINOU Y., ULLMAN J., WIDOM J., « Integrating and Accessing Heterogeneous Information Sources in TSIMMIS », *Proceedings of the AAAI Symposium on Information Gathering*, 1995.

[GAR 99a] GARDARIN G., *Internet/internet et bases de données*, Eyrolles, 1999.

[GAR 99b] GARDARIN G., SHA F., NGOC T., « XML-Based Components for Federating Multiple Heterogeneous Data Sources », *Proceedings of ER '99, 18th International Conference on Conceptual Modeling*, 1999, p. 506-519.

[GRE 92] GRELLER J., PERL Y., NEUHOLD E., SHETH A., « Structural schema integration with full and partial correspondance using the dual model », *Information Systems*, vol. 17, 1992, p. 443-464.

[KAN 99] KANNE C., MOERKOTTE G., « Efficient Storage of XML data », rapport n° 899, 1999, Mannheim University.

[KAU 92] KAUL M., DROSTEN K., NEUHOLD E., « Viewsystem: Integrating heterogeneous information based by object-oriented views », *IEEE International Conference on Data Engineering*, 1992, p. 2-10.

[KIM 93] KIM W., CHOI I., GALA S., SCHEEVEL M., « On resolving schematic heterogeneity in multidatabase systems », *International Journal on Parallel Distributed Databases*, vol. 1, 1993, p. 251-279.

[KLA 95] KLAS W., FANKHAUSER P., MUTH P., RAKOW T., NEUHOLD E., « Database integration using the open object-oriented database system VODAK », *Object Oriented Multidatabases*, Prentice Hall, 1995.

[KOT 99] KOTIDIS Y., ROUSSOPOULOS N., « DynaMat : A Dynamic View Management System for Data Warehouses », *Proceedings of SIGMOD*, 1999, p. 371-382.

[KRI 91] KRISHNAMURTY R., LITWIN W., KENT W., « Language features for interoperability of databases with schematic discrepencies », *Proceedings of the ACM SIGMOD*, 1991, p. 40-49.

[LAH 99] LAHIRI T., ABITEBOUL S., WIDOM J., « Ozone : Integrating Structured and Semistructured Data », *Proceedings of the International Workshop on Database Programming Languages*, 1999.

[LAR 89] LARSON J., NAVATHE S., ELMARSI R., « A theory of attribute equivalence in databases with application to schema integration », *IEEE Transformation Software Engineering*, vol. 15, 1989, p. 449-463.

[LIT 90] LITWIN W., MARK L., ROUSSOPOULOS N., « Interoperability of multiple autonomous databases », *ACM Computing Survey*, vol. 22, 1990, p. 267-293.

[LUD 99] LUDÄSCHER B., PAPAKONSTANTINOU Y., VELIKHOV P., VIANU V., « View Definition and DTD Inference for XML », *Post ICDT Workshop on Query Processing Conference for Semistructured Data and Non Standard Data Formats*, 1999.

[MAN 92] MANOLA F., HEILER S., « An approch to interoperable object models », *Proceedings of the International Workshop on Distributed Object Management*, Edmonton, Canada, 1992, p. 326-330.

[MAR 00] MARIAN A., ABITEBOUL S., MIGNET L., « Change-centric management of versions in an XML Warehouse », *Proceedings of the 16th journées Bases de Données Avancées*, 2000, p. 281-303.

[MCH 97] MCHUGH J., ABITEBOUL S., GOLDMAN R., QUASS D., WIDOM J., « Lore: A Database Management System for Semistructured Data », *SIGMOD Record*, 1997, p. 54-66.

[MIC 98] MICHARD A., *XML : langage et applications*, Eyrolles, 1998.

[MIG 00] MIGNET L., ABITEBOUL S., AILLERET S., AMANN B., MARIAN A., PREDA M., « Acquiring XML pages for a Webhouse », *Proceedings of the 16th journées Bases de Données Avancées*, 2000, p. 241-263.

[MIX] « Web Site of the MIX Project », http://www.npaci.edu/DICE/MIX/.

[MOT 87] MOTRO A., « Superviews: Virtual integration of multiple databases », *IEEE Transformation Software Engineering*, vol. 13, 1987, p. 785-798.

[PAP 95a] PAPAKONSTANTINOU Y., GARCIA-MOLINA H., WIDOM J., « Objetct Exchange Across Heterogeneous Information Source », *Proceedings of the International Conference on Data Engineering*, 1995, p. 251-260.

[PAP 95b] PAPAKONSTANTINOU Y., GUPTA A., GARCIA-MOLINA H., ULLMAN J. D., « A Query Translation Scheme for Rapid Implementation of Wrappers », *Deductive and Object-Oriented Databases, Fourth International Conference, DOOD'95, Singapore, December 4-7, 1995*, vol. 1013, Springer, 1995, p. 161-186.

[PIT 95] PITOURA E., BUKHRES O., ELMAGARMID A., « Object Orientation in Multidatabase Systems », *ACM Computing Surveys*, vol. 27(2), 1995, p. 141–195.

[REY 01] REYNAUD C., SIROT J., VODISLAV D., « Semantic Integration of XML Heterogeneous Data Sources », *Proceedings of International Database Engineering and Applications Symposium, IDEAS01, July 16-18, 2001, Grenoble, France*, IEEE Computer Society, 2001.

[SHE 87] SHETH A. P., LARSON J., CORNELIO A., NAVATHE S. B., « A tool for integrating conceptual schemas and user views », *Proceedings of the 4th International Conference on Data Engineering*, 1987, p. 176-183.

[SUC 98a] SUCIU D., « An Overview of Semistructured Data », *SIGACT News*, 1998, p. 28-38.

[SUC 98b] SUCIU D., « Semistructured Data and XML », *Proceedings of the 5th International Conference of Foundations of Data Organization (FODO'98*, 1998, Invited talk.

[W3C] W3C, « Web Site of the World Wide Web Consortium (W3C) », http://www.w3c.org.

[WID 92] WIDERHOLD G., « Intelligent Integration of Diverse Information », *Keynote presented at CIKM*, 1992.

[XML] « Web Site of e-XMLMedia », http://www.e-xmlmedia.fr/.

[XML 00a] « Extensible Markup Language (XML) 1.0 (Second Edition) », 6 october 2000, http://www.w3c.org/TR/REC-xml.

[XML 00b] « XML Query Requirements », 15 august 2000, http://www.w3c.org/TR/-xmlquery-req.

[XML 00c] « XML Schema Part 2 : Datatypes », 24 october 2000, http://www.w3c.org/TR/-xmlschema-2.

[Xyl] « Web site of the Xyleme project », http://www-rocq.inria.fr/xyleme/.